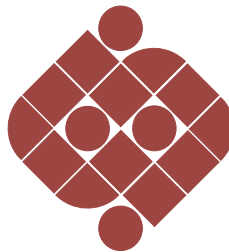# Recent Advances in Networking

Hamed Haddadi, Olivier Bonaventure (Editors)

ACM SIGCOMM

August 2013

Introduction

Professors often rely on textbooks to teach undergraduate and graduate networking courses. While there are many good introductory textbooks, there are very few books on advanced networking topics that could be suitable to graduate courses in networking. To fill this gap, SIGCOMM Education Committee has launched a community project to develop a high-quality, open-source, edited eBook on "Recent Advances in Networking". This eBook will be distributed online via the SIGCOMM website.

The first edition of this eBook is composed of nine chapters chosen after a highly selective review process by the editorial board. The selected chapters cover advanced networking topics and accompanying teaching material (slides and exercises). All the source code of the eBook and the teaching material are kept on a version controlled repository that will be accessible for the entire SIGCOMM community. We expect that releasing such high quality teaching material will be beneficial for a large number of students and professors. The teaching material will be updated on a regular basis to reflect new advances in our field. We will also be adding new chapters on more emerging topics in the future volumes.

We wish to thank all the authors for providing such high quality chapters. We are also grateful to the reviewers and the editorial board for spending many hours on each chapter to ensure a coherent level amongst all the chapters. We hope you enjoy reading this eBook and find it a useful resource.

Hamed Haddadi, Olivier Bonaventure

In the chapter on Internet Topology Research Redux we revisit some of the interesting properties of the critical infrastructure at router, AS and PoP level which today underpin how the world is knot together.

On top of this ever evolving topology, traffic sources are regulated by end-to-end protocols such as TCP - these too see continual changes, and the chapter on Recent Advances in Transport Protocols covers some of the ways TCP is being enhanced to cope with multihoming, and take advantage of multiple active routes between ends, whether in the data center or in the pocket.

We can put together the traffic sources, and the topology under one framework, that of Internet Traffic Matrices. The next chapter introduces a primer to this topic.

One general framework for understanding dynamic traffic management in the Internet is to think of the system as one of continual optimization. The chapter on Optimizing and Modeling Dynamics in Networks revisits the goals of fairness and the control problem and its solution space.

It has been said that all is fair in love and war, but the only certainties in the world are death and taxes. Life may not be fair, but we can enforce a different kind of fairness on the Internet through traffic pricing. The next chapter discusses Smart Data Pricing (SDP) and covers a range of Economic Solutions to Network Congestion, bringing game theory to bear on the problem where the previous chapter engaged with the weapons of optimization.

At a coarser grain (in time and space) we can control traffic by partitioning our network into VPNs - the next chapter looks at the practical tools available to the operator to manage a set of such disjoint systems and their capacity, by focusing on MPLS and Virtual Private Networks.

For some time, the capacity of the net has been consumed largely by people downloading or live streaming content. Content Distribution Networks are overlays that allow management of the load. A less rigorous but nearly as popular family of tools exist based on the famous peer-to-peer paradigm. The next chapter looks at how the world has moved on from fighting, to embracing Collaboration Opportunities for Content Providers and Network Infrastructures.

The true explosion in Internet access in the developing world (and now dominant form of access in the developed world too) has been through mobile devices (smart phones, tablets and the rest). The next chapter looks at the Design Space of Network Mobility, and how seamlessness is achieved (or at least as close to seamless as we can get today).

Finally, there have been sporadic attempts to build provider-less networks that are build out of mobile wireless nodes only. It is quite a challenge, and a number of breakthroughs over the last few years have led to engineering solutions for Enabling Multihop Communication in Spontaneous Wireless Networks, which are starting to look like a viable alternative to managed wireless networks in some niche areas.

This is an exciting time to be learning about communications and building and extending systems for communications. Planet Earth is not far away from being 100% connected, and the capacity and functionality that has been achieved over the last few decades is quite astounding. It does not look like we have hit any fundamental limits, nor will for some time to come!

Read this book by some of the world's leading lights in the area of communications and enjoy.

Cambridge, UK, *Jon Crowcroft*
August 2013

Editorial Board

Review Board

List of Chapters

1. **Internet Topology Research Redux**
   Walter Willinger, Matthew Roughan

2. **Recent Advances in Reliable Transport Protocols**
   Olivier Bonaventure, Janardhan Iyengar, Costin Raiciu

3. **Internet Traffic Matrices: A Primer**
   Paul Tune, Matthew Roughan

4. **Optimizing and Modeling Dynamics in Networks**
   Ibrahim Matta

5. **Smart Data Pricing (SDP): Economic Solutions to Network Congestion**
   Soumya Sen, Carlee Joe-Wong, Sangtae Ha, Mung Chiang

6. **MPLS Virtual Private Networks**
   Luca Cittadini, Giuseppe Di Battista, Maurizio Patrignani

7. **Collaboration Opportunities for Content Delivery and Network Infrastructures**
   Benjamin Frank, Ingmar Poese, Georgios Smaragdakis, Anja Feldmann, Bruce M.
   Maggs, Steve Uhlig, Vinay Aggarwal, and Fabian Schneider

8. **The Design Space of Network Mobility**
   Pamela Zave, Jennifer Rexford

9. **Enabling Multihop Communication in Spontaneous Wireless Networks**
   Juan Antonio Cordero, Jiazi Yi, Thomas Clausen, Emmanuel Baccelli

Internet Topology Research Redux

Walter Willinger   Matthew Roughan

1   Introduction

Internet topology research is concerned with the study of the various types of connectivity structures that are enabled by the layered architecture of the Internet. More than a decade of Internet topology research has produced a number of high-profile "discoveries" that continue to fascinate the scientific community, even though (or, especially because) they have been simultaneously touted by different segments of that community as either seminal, controversial, seriously flawed, or simply wrong. Among these highly-popularized discoveries are the observed power-law relationships of the Internet topology, the network's scale-free nature, and its extreme vulnerability to attacks that target the highly-connected nodes in its core (*i.e.,* the Achilles' heel of the Internet).

The purpose of this chapter is to bring order to the current state of Internet topology research and separate "the wheat from the chaff". In particular, by relying on carefully vetted data and readily available domain knowledge, we re-examine the reported discoveries and expose them to higher standards with respect to statistical inference and model validation. In the process, we reveal the superficial nature of many of these discoveries and provide alternative solutions that reflect networking reality and do not collapse under scrutiny with high-quality data or when examined in detail by domain experts.

1.1   The many facets of Internet connectivity

Internet topology research is concerned with the study of the various types of connectivity structures that are enabled by the layered architecture of the Internet. These structures include the inherently physical components of the Internet's infrastructure (*e.g.,* routers and switches and the fiber cables connecting them) as well as a wealth of more logical topologies that can be defined and studied at the higher layers of the Internet's TCP/IP protocol stack (*e.g.,* IP-level graph, AS-level network, Web-graph, P2P networks, Online Social Networks or OSNs).

As early as the ARPANET, researchers were drawing maps of the network representing its connectivity [25]. The earliest date to 1969. In those days, the entire network was simply enough to draw on the back of an envelope[1], and accurate maps could be drawn because every piece of equipment was expensive, installation was a major task, and only a few people worked on the network.

As the network grew, its complexity also grew, until the point where no one person could draw such a map. At that point, automated strategies started to arise for measuring the topology. The earliest Internet topology studies date back to the time of the NSFNET and focused mainly on the network's physical infrastructure consisting of routers, switches and the physical links connecting them (*e.g.,* see [23, 119]). The decommissioning of the NSFNET around 1995 led to a transition of the Internet from a largely monolithic network structure (*i.e.,* NSFNET) to a genuinely diverse "network of networks." Also known

---

[1]For instance see http://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/roberts_arpanet_large.gif

as Autonomous Systems (ASes), together these individual networks form what we now call the "public Internet" and are owned by a diverse set of organizations and companies that includes large and small Internet Service Providers (ISPs), transit providers, network service providers, Fortune 500 companies and small businesses, academic and research organizations, content providers, Content Distribution Networks (CDNs), Web hosting companies, and cloud providers.

With this transition came an increasing fascination of the research community with a largely economics-driven connectivity structure commonly referred to as the Internet's AS-graph; that is, the logical Internet topology where nodes represent individual ASes and edges reflect observed relationships among the ASes (*e.g.,* customer-provider, peer-peer, or sibling-sibling relationship). It is important to note that the AS-graph says little about how two ASes connect with one another at the physical level; in particular, it says nothing about if or how they exchange actual traffic. Nevertheless, starting shortly after 1995, this fascination with the AS-graph has resulted in thousands of research publications covering a range of aspects related to measuring, modeling, and analyzing the AS-level topology of the Internet and its evolution over time [60, 135].

At the application layer, the emergence of the World Wide Web (WWW) in the late 1990 as a killer application generated general interest in exploring the Web-graph, where nodes represent web pages and edges denote hyperlinks [18]. While this overlay network or logical connectivity structure says nothing about how the servers hosting the web pages are connected at the physical or AS level, its scale and dynamics differ drastically from its physical-based or economics-driven underlays – a typical Web-graph has billions of nodes and even more edges and is highly dynamic; a large ISP's router-level topology consists of some thousands of routers, and today's AS-level Internet is made up of some 30,000-40,000 actively routed ASes and an order of magnitude more links.

Other applications that give rise to their own "overlay" or logical connectivity structure and have attracted some attention among researchers include email and various P2P systems such as Gnutella, Kad, eDonkey, and BitTorrent. More recently, the enormous popularity of Online Social Networks (OSNs) has resulted in a staggering number of research papers dealing with all different aspects of measuring, modeling, analyzing, and designing OSNs. Data from large-scale crawls or, in rare circumstances, OSN-provided data have been used to examine snapshots of many real-world OSNs or OSN-type systems, where the snapshots are generally simple graphs with nodes representing individual users and edges denoting some implicit or explicit friendship relationship among the users.

1.2 Many interested parties with different objectives

The above-mentioned list of possible connectivity structures that exist in today's Internet is by no means complete, but illustrates how these structures arise naturally within the Internet's layered architecture. It also highlights the many different meanings of the term "Internet topology," and sensible use of this term requires explicitly specifying which facet of Internet connectivity is considered because the differences are critical.

The list also reflects the different motivations that different researchers have for studying Internet-related graphs or networks. For example, engineers are mainly concerned with the physical facets of Internet connectivity, where technological issues generally dominate over economic and social aspects. However, the more economics-minded researchers are particularly interested in the Internet's AS-level structure where business considerations and market forces mix with technological innovation and societal considerations and shape the very structure and evolution of this logical topology. Moreover, social scientists see in the application-level connectivity structures that result from large-scale crawls of the various OSNs new and exciting opportunities for studying different aspects of human behavior and

technology-enabled inter-personal communication at previously unheard of scale.

In addition, mathematicians are interested in the different connectivity structures mainly because of their many novel features and properties that tend to require new and creative modeling and analysis methodologies. From the perspective of many computer scientists, the challenges posed by many of these intricate connectivity structures are algorithmic in nature and arise from trying to solve specific problems involving a particular topological structure. For yet another motivation, many physicists turned network scientists see the Internet as one of many examples of large-scale complex networks that awaits the discovery of universal properties that do not depend on system-specific details and advance our understanding of these complex networks irrespective of the domain in which they arose in the first place.

1.3 More than a decade of Internet topology research

When trying to assess the large body of literature in the area of Internet topology research that has accumulated since about 1995 and has experienced enormous growth especially during the last 10+ years, the picture that emerges is at best murky.

On the one hand, there are high-volume datasets of detailed network measurements that have been collected by domain experts. These datasets have been made publicly available so other researchers can use them. As a result, Internet topology research has become a prime example of a measurement-driven research effort, where third-party studies of the available datasets abound and have contributed to a general excitement about the topic area, mainly because many of the inferred connectivity structures have been reported to exhibit surprising properties (*e.g.,* power-law relationships for inferred quantities such as node degree [49]). In turn, these surprising discoveries have led network scientists and mathematicians alike to develop new network models that are provably consistent with some of this highly-publicized empirical evidence. Partly due to their simplicity and partly due to their strong predictive power, these newly proposed network models have become very popular within the larger scientific community [5, 11, 12]. For example, they have resulted in claims about the Internet that have made their way into standard textbooks on complex networks, where they are also used to support the view that a bottom-up approach dominated by domain-specific details and knowledge is largely doomed when trying to match the insight and understanding that a top-down approach centered around a general quest for "universality" promises to provide [10, 45, 113, 123].

On the other hand, there is a body of work within the networking research literature that argues essentially just the opposite and presents the necessary evidence in support of a inherently engineering-oriented approach filled with domain-specific details and knowledge [7, 93, 156]. In contrast to being measurement-driven, this approach is first and foremost concerned with notions such as a network's purpose or functionality, the hard technological constraints that the different devices used to build a network's physical infrastructure have to obey, or the sources of uncertainty in a network's "environment" with respect to which the built network should be robust. As for the measurements that have been key to the top-down approach, the reliance on domain knowledge reveals the data's sub-par quality and highlights how errors of various forms occur and can add up to produce results and claims that create excitement among non-experts but quickly collapse when scrutinized or examined by domain experts. While there exist currently no textbooks that document these failures of applying detail- and domain knowledge-agnostic perspective to the Internet, there is an increasing number of papers in the published networking research literature that detail the various mis-steps and show why findings and claims that look at first glance impressive and conclusive to a science-minded reader turn out to be simply wrong or completely meaningless when examined closely by domain experts [6, 85, 156].

In short, a survey of the existing literature on Internet topology research leaves one with the distinct

impression that "too many cooks spoil the broth." We hope that in the not-too-distant future, this impression will be replaced by "many hand make light work", and we see this chapter as a first step towards achieving this goal.

1.4 Themes

In writing this chapter there are a number of themes that emerge, and it is our intention to highlight them to bring out in the open the main differences between a detail-oriented engineering approach to Internet topology modeling versus an approach that has become a hallmark of network science and aims at abstracting away as many details as possible to uncover "universal" laws that govern the behavior of large-scale complex networks irrespective of the domains that specify those networks in the first place.

Theme 1: When studying highly-engineered systems such as the Internet, "details" in the form of proto-cols, architecture, functionality, and purpose matter.

Theme 2: When analyzing Internet measurements, examining the "hygiene" of the available measure-ments (*i.e.,* an in-depth recounting of the potential pitfalls associated with producing the measure-ments in question) is critical.

Theme 3: When validating proposed topology models, it is necessary to treat network modeling as an exercise in reverse-engineering and not as an exercise in model-fitting.

Theme 4: When modeling highly-engineered systems such as the Internet, beware of M.L. Mencken's quote "For every complex problem there is an answer that is clear, simple, and wrong."

2 Primer

We start first by defining some common ideas, motives, and problems within the scope of network topology modelling.

2.1 A Graph Primer

In this context *topology* usually refers to the structure of the *graph* representation of a network. That is, the common notion used to describe network topology is the mathematical *graph*. A graph $\mathcal{G}$ is defined by a set of nodes $\mathcal{N}$ (often called vertices) and edges (or links) $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$, so we usually write $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. Here, we shall denote the number of nodes $N = |\mathcal{N}|$, and the number of edges $E = |\mathcal{E}|$.

Nodes are usually associated with some logical of physical structure in a network: a router, switch, PoP, or AS. Edges are associated with the appropriate type of logical or physical link between these nodes.

A graph describes connectivity between logical resources such as routers, or IP address, but simple connectivity is rarely as useful as when additional information such as names, capacity or distance are attached to these abstract objects. Such can easily be included in these descriptions by creating labelling functions of the node or edge sets, in the form: $f : \mathcal{N} \to \mathbb{R}$ or $f : \mathcal{E} \to \mathbb{R}$ in the case of real-valued labels. We could similarly defined labelling functions with text labels, or integer or vector values, and so on. However, it is naive to treat labels as an "add-on" as they carry semantics that can be important in the network. For instance, when we define link distances (be these geographic or semantic), that can change the notion of distance in the network as a whole.

We can also define functions of groupings of nodes or edges, though in this case it is not as conceptually obvious why we might. However, an exemplary case is that of "on-net" where we might define a function that classifies pairs of nodes as on the same subnet or not. Thus, such functions can ascribe meaning to groupings of nodes.

Many of the Internet graphs have symmetric links (that is, if $i \rightarrow j$ is a link, then $j \rightarrow i$ is also a link) and so these networks are *undirected*, but we also need sometimes to represent asymmetric links, and do so with a *directed graph* or *digraph*, and we call the links in such a digraph *arcs*.

In the study of network topology we might come across the more generalized graph concepts of the *multi-graph* and *hyper-graph*.

- *hypergraph*: links connect more than two nodes

  - *e.g.,* where you have a connective medium (rather than a wire), for instance in a wireless network.

- *multigraph* or *pseudograph*: has multiple parallel links between two nodes

  - *e.g.,* it is easy to have two links between two routers.

We'll exclude these cases unless explicitly stated, but it is worth noting that each of these do apply to particular aspects of the Internet.

We say two nodes are *connected* if a path exists between them, and that a graph is connected if all pairs of nodes are connected. A graph is $k-$node connected if the graph remains connected after the removal of any set of $k-1$ or fewer nodes (and corresponding links) and $k-$edge connected if the graph remains connected after the removal of any $k-1$ edges.

For an undirected graph G, define the *neighborhood* of node i by

$$N_i = \left\{ j \mid (i, j) \in E \right\},$$

i.e., the set of adjacent nodes to i, and we define the degree of the node to be the number of elements in the neighborhood to be

$$k_i = |N_i|.$$

In a directed graph, we define two concepts: the *in-degree* (the number of links connecting to the link) and the *out-degree* (the number of links originating from it).

$$
\begin{aligned}
\text{in-degree}(i) &= \left| \{(j, i) | (j, i) \in \mathscr{E}\} \right|, \\
\text{out-degree}(i) &= \left| \{(i, j) | (i, j) \in \mathscr{E}\} \right|.
\end{aligned}
$$

We often consider statistics of the degree distribution p_k (which gives the probability that a node has degree k), the average node degree being the most obvious such. It can be easily calculated from the sum of degrees, which has the interesting property

$$\sum_{i \in N} k_i = 2|E|,$$

generally referred to the Handshake lemma.

The node-degree distribution provides a common characterization of a graph (though by no means a complete characterization). It is noteworthy, however, that although this distribution is frequently discussed, the concept is somewhat ill-defined. It can be directly measured for a real network, in which

case p_k is the probability that a randomly selected node from the measured graph has degree k. However, it is often used in the context of a set of simulated graphs, where it is used to mean the probability that a node in the ensemble of networks has degree k with this probability. The difference is subtle, but it is worth keeping track of such discrepancies.

There are many other graph *metrics*. For instance, the *distance*[2] between two connected nodes in an unweighted graphs is generally defined to be the number of edges in the shortest path connecting them. We can then examine quantities such as the average distance, and the *diameter* of the network (the maximum distance).

And there are many other metrics: assortativity, clustering coefficient, centrality, and so on. They are all attempts to capture the nature of a graph in a small set of measures, and as such provide simpler, seemingly more intuitive ways to consider graphs. For other discussions of these, and comparisons in the context of Internet topologies see [68, 79]. We must be wary though, as it should be clear that the potential for problems is immediate. No small set of numbers can truly represent graphs. For instance, consider the Hamiltonian cycle[3] problem. The problem of determining if a network has such a path is well known to be NP-complete, and as such no small set of statistics of the graph will provide a characterization that is sufficient to consider this problem. Thus, these simple statistics must miss important properties of the network.

It may be useful to the reader to consider some of the tools that are available for working with graphs. They have different sets of feature, but perhaps the most important is whether they are used in conjunction with a programming language and which one, so we have listed a (no doubt incomplete) set below with some very basic information.

- MatlabBGL http://www.stanford.edu/~dgleich/programs/matlab_bgl/

    - Graph libraries for Matlab,
    - using Boost Graph Library (BGL)
      http://www.boost.org/doc/libs/1_42_0/libs/graph/doc/index.html
- igraph http://igraph.sourceforge.net/

    - Libraries for working with graphs in R or Python
- GraphVis http://www.graphviz.org/

    - Toolkit for visualization of graphs
- NetworkX http://networkx.lanl.gov/

    - Python toolkit for working with graphs
- GDToolkit http://www.dia.uniroma3.it/~gdt/gdt4/index.php

    - OO C++ library for handling and drawing graphs
- JUNG http://jung.sourceforge.net/

    - Java universal network/graph framework
- IGen http://informatique.umons.ac.be/networks/igen/

    - A toolkit for generating IP network topologies based on network design heuristics.

---

[2]The graph distance has a long history. In mathematics, perhaps the best known example is the Erdős number, which is the distance of a author from Erdős in the co-authorship graph. In popular culture there is an equivalent: the Bacon number, or the distance between actors in the graph of co-appearances.

[3]A Hamiltonian cycle is a path (on the graph) that visits each node exactly once, and then returns to the start point.

2.2 Motivations for network topology investigations

Underlying the whole research area is often an only vague notion of why we study topology. The motives for such studies are in fact quite diverse, and the implications are important. Topology studies motivated by network managers with operational imperatives have profoundly different requirements to those of the scientific research community. Broadly speaking, we can divide the motivations as follows:

- **Scientific:** Despite the fact that computer networks are designed (rather than grown as organic networks), little is known about their generic properties. Such knowledge would be useful (apart from satisfying simple curiosity) because there are many future protocols (for instance multicast protocols), and network-engineering algorithms (for instance see [54]) whose design could benefit from an understanding of typical networks, rather than the typically very small, and unrealistic test examples often employed.

- **Adversarial:** Some network operators (though not all [84]) believe that commercial rivals might gain advantage through obtaining proprietary information about their network design. Similarly, there is a general belief that such information might facilitate an attack on the network. For instance, knowledge of a competitor's customers might be used by an adversary to target Denial of Service (DoS) attacks. One possible motivation for topology discovery is for just such an adversary to gain information to target attacks.

- **Managerial:** It is often assumed that a network operator has a "database of record" that contains all the important information about their network. While this may be true in some cases, it is more often the case that such databases are hard to keep up-to-date and consistent with other data sources. This is actually a common problem in database management [37]. Such databases must be maintained by humans, and as soon as they grow large, and complex (particularly when they are dynamic), it becomes exceedingly difficult to eradicate all human errors. In addition, when the network undergoes change, particularly unplanned changes (failures), the database is unlikely to be accurate. Hence, for management of complex, dynamic networks, another source of data about the network is needed. It should be no surprise that obtaining this data from the network itself is the best solution for ensuring up-to-date, accurate records.

- **Informational:** This is a fairly general category of motivations, but differs from pure scientific curiosity in the immediacy of its application. For instance, customers of network operators often desire information about the networks to which they currently subscribe (in order to debug services, or obtain quality of service measurements), and also about networks to which they might subscribe (to help make choices about who could provide them with the most effective service). Often, a customer may not entirely trust its provider, or potential provider, and wish to verify information themselves. Hence, there is a need for such customers to be able to discover the networks to which they might subscribe, or where they should place services.

Each of these motivations imposes different requirements on our study in terms of accuracy, immediacy, and the types of measurements available. High degrees of accuracy are needed for network management; and certainly the measurements used scientifically have rarely been very accurate (though this is a significant problem with that research).

2.3   Type of network

The other core aspect we should consider is the type of topology to be examined, as it can have a drastic affect on both observations and behavior of the network. Two obvious dimensions are

Physical vs virtual: Physical networks are built from hardware: routers and cables (copper or optical fiber for the Internet), transformers and cables for electricity, or cities and roads for the road network. Virtual networks may have physical nodes, but virtual edges (as in a social network), or virtual nodes and edges (as in online social networks, or the WWW).

The main difference is that there are usually large costs in building, or adding to a physical network. Virtual networks, on the other hand, have much smaller costs (an HTML link costs almost nothing to create). Costs have a profound affect on network designs, as we shall later see, but also on dynamic behavior. If it is easier to change a network, it can change more quickly.

The other major issue for a physical network is that it is often bound by physical constraints, and this also profoundly affect their design.

Figure 1 illustrates the differences, and also makes the point that it isn't really a binary difference. Networks lie on a spectrum.
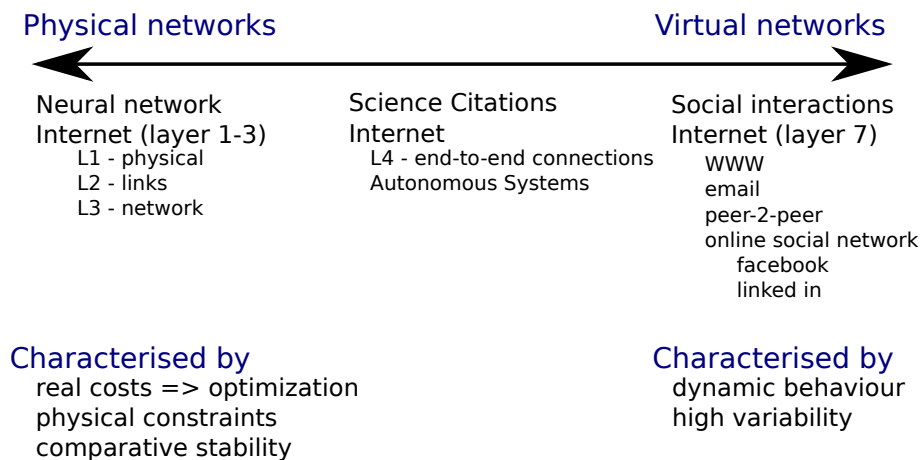
Figure 1: *Physical vs virtual networks.*

Transport vs Information flows: A more subtle differentiator is between what the network carries. Some networks physically transport some type of material (cars, water, ...) whereas the flows in other networks are (almost) pure information (the Internet, ...).

The importance of this distinction for networks may be less immediately obvious, but it certainly does have implications. When physical transport is involved in a network, the constraints on that network are likely to be even more stringent, and the ability to change the network even more limited. Costs for changing the road network, for instance, are usually higher than changing the equivalent proportion of a IP network.

Within this chapter, we are primarily interested in the "Internet", and that includes both physical (OSI layer 1-3) networks, and virtual networks (MPLS, WWW, online social networks, etc.). However, all of the

Transport networks   Information networks

Cars/buses/pedestrians   Social interactions
Distribution   Internet
    water   WWW
    electricity   email
                          etc.

Characterised by   Characterised by
   transport of physical objects   transport of information
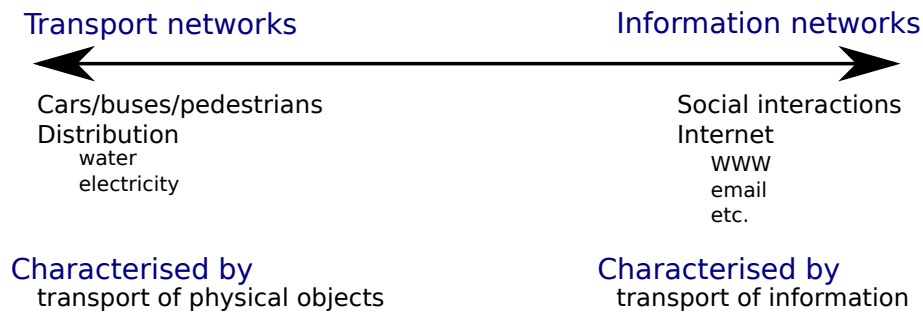
Figure 2: *Transport vs information flows.*

networks considered here are information transportation networks.

There are other dimensions on which networks could be classified. For instance, by the nature of the transport. Does it come in discrete chunks (*e.g.,* cars, packets, or the post) or continuously (*e.g.,* water or electricity)? Is the transport connection oriented (*e.g.,* the telephone network) or packet oriented (*e.g.,* the Internet)?

And there are other general issues we need to deal with:

- Physical networks are embedded in geography, but logical networks often aren't, and yet the same terminology is often applied to each.

- Connectivity often changes over time, with the time-scale varying depending on the type of network.

- The Internet is often said to be a "network of networks". It is often hard to consider one network in isolation, they have relationships, but the situations is even more complicated than often imagined.

  peers Networks may be connected to *peers, i.e.,* similar networks that may be competing or co-operating (or both in some cases), *e.g.,* two ISPs operating in the same region.

  parents Networks may have a parent-child relationship in the sense that one network controls the other, *e.g.,* the SS7 network with respect to traditional telephone network.

  layers A single network may have multiple layers, each of which can be represented by a different graph, *e.g.,* the physical- vs the network-layers in the Internet.

  external There is substantial interaction between notionally separated networks, *e.g.,* the power grid and the Internet, both because the Internet uses electricity, but also because spikes in electricity demand could potentially be caused by network flash crowds (certainly TV programs have a very important impact on electricity usage).

That brings us naturally to the particular object of discussion here – the Internet (and its topology). The term "Internet" means (many) different things to (many) different people. Even within the networking community, the term is often used ambiguously, leading to misunderstandings and confusion and creating roadblocks for a genuinely scientific treatment of an engineered system that has revolutionized the way we live.

While mathematics in the form of graph theory has been equally culpable in adopting the use of this vague nomenclature, the "new science of networks" has popularized it to the point where phrases like

"topology of the Internet" or "Internet graph" have entered the mainstream science literature, even though they are essentially meaningless without precisely-stated definitions. For one, "Internet topology" could refer to the connectivity structures encountered in any of layers in the protocol stack, or at various levels of aggregation. Common examples are

1. **Router-level (layer 3):** An often sought topology is the router level. Somewhat ambiguously, this may also be called the network level, or IP level, but "network" is a heavily overloaded term here, and the IP level can also be ambiguous. For instance, IP level could refer to the way IP addresses are connected, that is it could refer to the interfaces of one router as separate nodes [19], but that is rarely what is useful for network operations or research. We could also add at layer 3, in addition to *interface-level* topology described above, the *subnet-level* topology [19, 67, 81, 148, 149], describing the interconnectivity of logical subnets (often described by an IP-level prefix), but here we focus on the more commonly considered router level.

   The router-level graph shows a range of interesting implementation details of a network. This type of information is critical for network management applications, as much of Internet management rests at the IP layer, and it is of great importance for network adversaries. For instance, developing tools to measure network traffic requires an understanding of the router-layer topology, in order to match traffic to links. Similarly traffic engineering, and reliability analyses are carried out at this level. One complication of this layer is that we sometimes wish to obtain the topology extending out to end-hosts, which are not technically routers, but we shall include these in our definition of router-layer topology, unless otherwise specified.

2. **Switch-level (layer 2):** A single IP layer logical link may hide several layer-2 devices (hubs and switches). The increasing prevalence of Ethernet, and the ability to provide redundancy at reasonable cost, has led to a proliferation of such devices, and most Local Area Networks (LANs) are based around such. Hence, very many networks which have trivial, or simple IP layer topologies have complex and interesting layer-2 topologies. Multi-Protocol Label Switching (MPLS) further complicates the situation by creating logical layer-2 networks without physical devices, often in the form of cliques. Measurements often see only one layer, creating misunderstandings of a network's true resilience and more general graph properties. For instance, layer-2 devices can connect large numbers of routers, making them appear to have higher degree at layer-3 [104] (for more detailed discussion see §3.2.3).

3. **Physical-level (layer 1):** Below the link layer (layer 2), lies the physical layer. Again, many physical devices may underly a single logical link. Discovery of this layer is of critical importance in network management tasks associated with reliability. In particular, the concept of Shared Risk Link Groups (SRLG) requires knowledge of which links are carried on which fibers (using Wavelength Division Multiplexing), in which conduits. If a backhoe digs up a single conduit, it will cause a bundle of fibers to fail, and so connections that are in the same SRLG will all fail simultaneously. Clearly redundant links need to be in different SRLG, and discovery of the physical topology is important to ensure that this is the case.

4. **PoP-level:** A Point-of-Presence (PoP) is a loosely defined grouping of devices, often defined by a metropolitan area. PoP level topologies are quite useful, essentially because these graphs describe the logical structure of the network as the designer intended, rather than its particular implementation in terms of individual routers. Such topologies are ideal for understanding tradeoffs between connectivity and redundancy, and also provide the most essential information to competitors or

customers (about where a network is based, or who has the best access network in a region). Network maps are often drawn at this level because it is an easy level for humans to comprehend.

5. **Application layer:** There has been significant interest in logical topologies of the application layer, *e.g.,* for the Web (using HTTP, and HTML), and the P2P applications.

6. **AS-level:** AS topologies have generated much interest in the scientific literature [5, 13, 161], because they appear to show interesting properties (such as power-laws) in common with other un-engineered networks such as biological networks. Also, much data on AS topologies is publicly available. While of interest in the scientific literature, this data's use is confused by many myths and misunderstandings [135]. The data may provide mild competitive benefits, in allowing operators to determine who peers with who, but the measured data often comes without attributes that would make the data truly useful in this regard. Finally, it is hard to see how such data could be used in an attack, although much publicized reports such as [161] suggest, incorrectly (see [93]), that the observed structure of the AS graph may lead to an "Achilles heel" of the Internet.

The number of possible topologies we might wish to discover highlights the complexity of this problem, and why discovery is so valuable for network management. In this chapter we will consider the router-level topology in detail, and then discuss some of the similarities and differences with respect to AS- and PoP-level topologies.

In addition to understanding the Internet network as a simple graph, there are many other features of the graph that one would also wish to know, for instance, its routing, link capacities, and geographic locations. We describe such qualities as graph attributes, and find that most can either be attributed to edges of the graph, for instance

- link capacities,
- link length,
- routing weights (*e.g.,* for shortest-path routing),
- link utilizations,
- link performance (for example, bit-error-rate, delay, loss, jitter, reordering, buffer utilization),
- link status (up/down), and
- a link's lower layer properties (*e.g.,* number of physical hops),

or to the nodes of the graph

- geographic location,
- type of node, *e.g.,* brand of router, or version of software,
- performance measures (*e.g.,* CPU utilization), and
- node status (up/down).

We could further divide this list into *intrinsic* network properties, such as node location, or link capacity (things that cannot change easily), and *extrinsic* properties, such as performance, or traffic related properties, which can change dramatically despite there being no change in the underlying network.

3   Router-level topology

3.1   A look back

Since the early days of the ARPANET, networking researchers have been interested in drawing the type of networks they designed [71]. An early map of the ARPANET is reproduced in Figure 3 and shows the network's logical connectivity structures in April 1971, when the network as a whole consisted of some 15 nodes and a node was little more than one or two state-of-the-art computers connected via an Interface Message Processor or IMP (the modern equivalent would be a router). In this context, logical structure refers to a detailed drawing of the connectivity among those nodes and of node-specific details (*e.g.,* type of machine), by and large ignoring geography. In contrast, geographic structure refers to a map of the US that includes the actual locations of the network's physical nodes and shows the physical connectivity among those nodes. Such accurate maps could be drawn because at that time, each piece of equipment was expensive and needed to be accounted for, only a few groups with a small number of researchers were involved in the design and installation of the network, and the network changed relatively slowly.
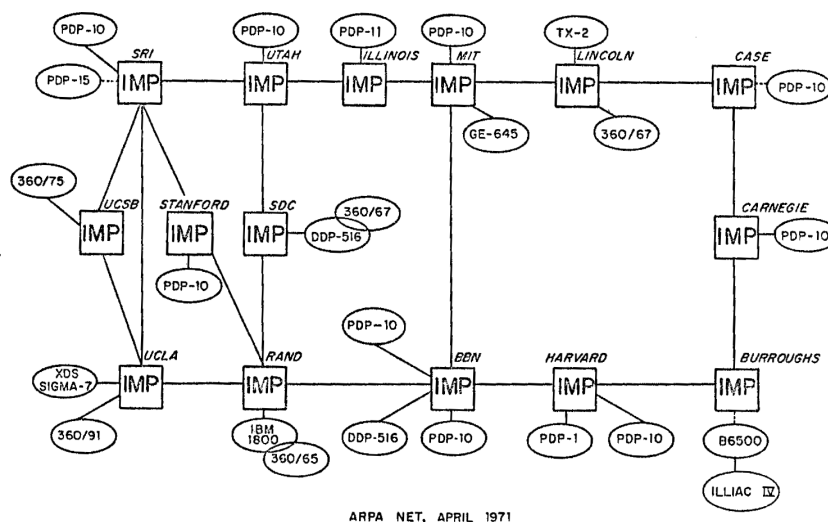

Figure 3: *The ARPANET in 1971 (reprinted from [25]; ©1990 ACM, Inc. Included here by permission.)*

The network quickly grew in size and complexity. For instance, Figure 4 shows the geographic counterpart from 1984 of the ARPANET map depicted in Figure 3. Manually accounting for the increasing number of components quickly became prohibitive and motivated the adoption of automatic strategies for obtaining some of the available connectivity as well as traffic information. A prime example for effectively visualizing this collected information is reproduced from [55] and shown in Figure 5, which depicts a 3D rendering of the (US portion of the) NSFNET around 1991, annotated with traffic-related information. At that time, the NSFNET backbone consisted of some 14 nodes that were interconnected with T1 links as shown and, in turn, connected to a number of different campus networks (*e.g.,* collections of interconnected LANs). However, even though the internal structure of the backbone nodes was well-known (*i.e.,* each node was composed of nine IBM RTs linked by two token rings with an Ethernet interface to attached

12

Figure 4: *The early ARPANET (reprinted from [25]; ©1990 ACM, Inc. Included here by permission.)*

networks), nobody had any longer access to the internals of all the different campus networks and as a result, drawing the 1991 NSFNET equivalent of the ARPANET's logical connectivity structure (Figure 3) was no longer possible.

With the decommissioning of the NSFNET in 1995 and the rise of the "public Internet", the researchers' ability to obtain detailed connectivity and component information about the internals of the different networks that formed the emerging "network of networks" further diminished and generated renewed interest in the development of abstract, yet informed, models for router-topology evaluation and generation. For example, the Waxman model [155], a variation of the classical Erdös-Rényi random graph model [47] was the first popular topology generator commonly-used for network simulation studies at the router level. However, it was largely abandoned in the late 1990s in favor of models that attempted to explicitly account for non-random structure as part of the network design. The arguments that favored structure over randomness were largely empirical in nature and reflected the fact that the inspection of real-world router-level ISP networks showed clear signs of non-random structures in the form of the presence of backbones, the appearance of hierarchical designs, and the importance of locality. These arguments also favored the notion that a topology generator should reflect the design principles in common use; *e.g.,* to achieve some desired performance objectives, the physical networks must satisfy certain connectivity and redundancy requirements, properties which are generally not guaranteed in random network topologies. These principles were, for example, advanced in [23, 164, 165] and were ultimately integrated into the popular Georgia Tech Internetwork Topology Models (GT-ITM) [65].

These more structure-oriented router topology generators were viewed as the state-of-the-art until around 2000 when, in turn, they were largely abandoned in favor of a new class of random graph models whose trademark was the ability to reproduce the newly discovered power-law relationship in the observed connectivity (*i.e.,* node degree) of router-level graphs of the Internet. This discovery was originally reported

13
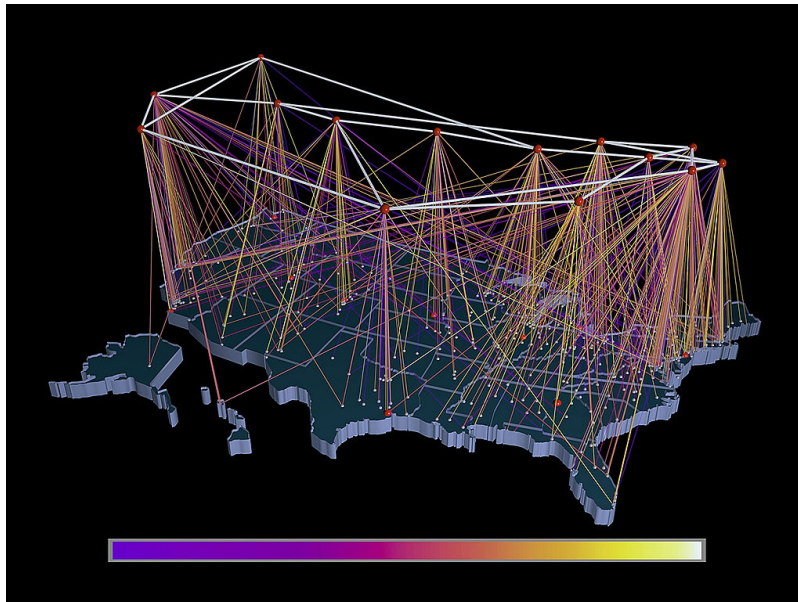
Figure 5: *A visualization of the NSFNET circa 1991 (by Donna Cox and Robert Patterson, National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign. See also* `http://en.wikipedia.org/wiki/File:NSFNET-traffic-visualization-1991.jpg`*).*

in the seminal paper by Faloutsos *et al.* [49], who used a router-level graph constructed from data that was collected a few years earlier by Pansiot and Grad [119] for the purpose of obtaining some experimental data on the actual shape of multicast trees in the Internet. The Boston university Representative Internet Topology gEnerator (BRITE) [103] became a popular representative of this new class of models, in part also because it combined the more structure-oriented perspective of the GT-ITM generator with the new focus that emphasized the ability to reproduce certain metrics or statistics of measured router topologies (*e.g.,* node degree distribution).

One of the hallmarks of networks that have power-law degree distributions and that are generated according to any of a number of different probabilistic mechanisms (*e.g.,* preferential attachment [13], random graphs with a given expected degree sequence [30], power-law random graphs [3]) is that they can be shown to have a few centrally located and highly connected *hubs* through which essentially most traffic must flow. When using these models to represent the router-level topology of the Internet, the presence of these highly connected central nodes has been touted the Internet's "Achilles-heel" because network connectivity is highly vulnerable to attacks that target the high-degree hub nodes [5]. It has been similarly argued that these high-degree hubs are a primary reason for the epidemic spread of computer worms and viruses [112, 122]. Importantly, the presence of highly connected central nodes in a network having a power-law degree distribution is the essence of the so-called scale-free network models. They have been a highly popular theme in the study of complex networks, particularly among researchers inspired by statistical physics [4], and have fuelled the rise of a new scientific discipline that has become known as "Network Science" [12]. In the process, they have also seen wide-spread use among Internet topology researchers.

However, as the general fascination with and popularity of network science in general and scale free network modeling in particular grew, so did the arguments that were voiced by Internet researchers and questioned the appropriateness and relevance of the scale-free modeling approach for studying highly-engineered systems such as the Internet's router topology. In fact, at around 2010, when the number of publications in the area of network science reached a new height, the number of papers that were published in the networking research literature and applied scale-free network models to describe or study router-level topologies of the Internet was close to zero. This begs the question "What happened?", and the answer provided in the next section is really a classic lesson in how errors of various forms occur and can add up to produce results and claims that create excitement among non-networking researchers, but quickly collapse when scrutinized with real data or examined by domain experts.

3.2   Know your measurements

Between 1990 and 2000, Internet topology research underwent a drastic change from being a data-starved discipline to becoming a prime example of a largely measurement-driven research activity. As described earlier, even though the development of abstract, yet informed, models for network topology evaluation and generation has always been a give and take between theoreticians and empiricists, for router topology modeling, the essential role that measurements have started to play came into full focus in a sequence of three seminal papers that appeared between 1998-2000.

3.2.1   Three seminal papers on router topology modeling

The key papers that turned router topology modeling into a full-fledged measurement-driven research activity cover the whole spectrum of modeling activities, from measurement experiments to model construction and validation to graph-theoretical network analysis, and are listed below:

(i) "On routes and multicast trees in the Internet" by J.-J. Pansiot and D. Grad (1998) [119] described the original measurement experiment that was performed in mid-1995 and produced data on actual routes taken by packets in the Internet. This data was subsequently used to construct a router graph of the Internet.

(ii) "On power-law relationships of the Internet topology" by M. Faloutsos *et al.* (1999) [49] reported (among other observations) on the observed power-law relationship in the connectivity of the router-level topology of the Internet measured by Pansiot and Grad [119].

(iii) "Error and attack tolerance of complex networks" by R. Albert *et al.* (2000) [5] proposed a scale-free network model to describe the router topology of the Internet and argued for its validity on the basis of the latest findings by Faloutsos *et al.* [49]. It touted the new model's exemplary predictive power by reporting on the discovery of a fundamental weakness of the Internet (a property that was became known as the Internet's "Achilles' heel") that went apparently unnoticed by the engineers and researchers who have designed, deployed, and studied this large-scale, critical infrastructure, but followed directly from the newly proposed scale-free modeling approach.
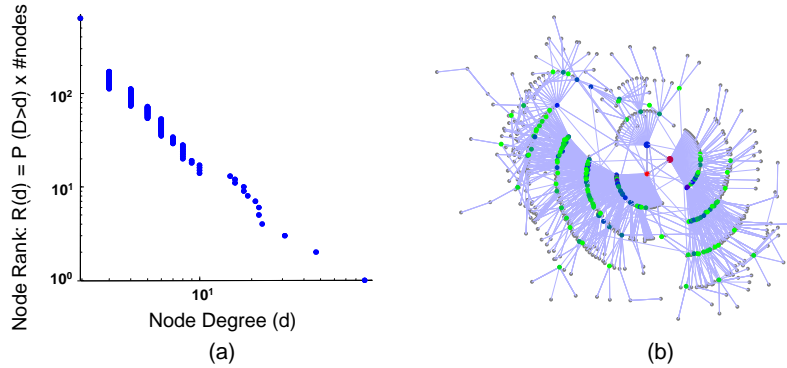
Figure 6: *A toy example of a scale-free network of the preferential attachment type (b) generated to match a power-law type node degree distribution (a). (First published in Notices of the American Mathematical Society, Volume 56, No.3 (May 2009): 586-599 [156]. Included here by permission.)*

At first glance, the combination of these three papers appears to show network modeling at its best – firmly based on experimental data, following modeling practices steeped in tradition, and discovering surprisingly and previously unknown properties of the modeled network. An example of a toy network resulting from taking the findings from these seminal papers at face value is shown in Figure 6. However, one of the beauties of studying man-made systems such as the Internet is that – because of their highly-engineered architectures, a thorough understanding of its component technologies, and the availability of extensive (but not necessarily very accurate) measurement capabilities – they provide a unique setting in which most claims about their properties, structure, and functionality can be unambiguously resolved, though perhaps not without substantial efforts. In the remainder of this section, we will illustrate how in the context of the Internet's router topology, applying readily available domain knowledge in the form of original design principles, existing technological constraints, and available measurement methodologies reveals a drastically different picture from that painted in these three seminal papers. In fact, we will

expose the specious nature of scale-free network models that may appeal to more mathematically inclined researchers because of their simplicity or generality, but besides having no bearing on the Internet's router topology are also resulting in wrong claims about the Internet as a whole.

3.2.2   A first sanity check: Using publicly available information

A first indication of apparent inconsistencies between the proposed scale-free models for the Internet's router topology and the actual Internet comes from the inspection of the router topologies of actual networks that make the details of their network internals publicly available. For example, networks such as Internet2 [77] or GÉANT [57] show no evidence that there exist any centrally located and highly connected "hubs" through which essentially most traffic must flow. Instead, what they typically show is the presence of a more or less pronounced "backbone" network that is fed by tree-like access networks, with additional connections at various places to provide a degree of redundancy and robustness to components failures[4].

This design pattern is fully consistent with even just a cursory reading of the most recent product catalogs or white papers published by the main router vendors [32,33,80]. For one, the most expensive and fastest or highest-capacity pieces of equipment are explicitly marketed as backbone routers. Moreover, due to inherent technological limitations in how many packets or bytes a router can handle in a given time interval, even the latest models of backbone routers can support only a small number of very high-bandwidth connections, typically to connect to other backbone routers. At the same time, a wide range of cheaper, slower or lower-capacity products are offered by the different router vendors and are targeted primarily at to support network access. On the access side, a typical router will have many lower-bandwidth connections for the purpose of aggregating customer traffic from the network's edge and subsequently forwarding that traffic towards the backbone. In short, even the latest models advertised by today's router vendors are limited by existing technologies, and even for the top-of-the-line backbone routers, it is technologically infeasible to have hundreds or thousands of high-bandwidth connections. At the same time, while technically feasible, deploying some of the most expensive equipment and configuring it to support hundreds or thousands of low-bandwidth connections would be considered an overall bad engineering decision (*e.g.,* excessively costly, highly inefficient, and causing serious bottlenecks in the network).

However, the root cause for these outward signs of a clear mismatch between the modeled and actual router topology of the Internet goes deeper and lies in the original design philosophy of the Internet. As detailed in [34], while the top level goal for the original DARPA Internet architecture was *"to develop an effective technique for multiplexed utilization of existing interconnected networks"*, the requirement that *"Internet communication must continue despite loss of networks or gateways"* topped the list of second level goals. To survive in the face of components failing off, the architecture was to mask completely any transient failure, and to achieve this goal, state information which describes an existing connection must be protected. To this end, the architecture adopted the "fate-sharing" model that gathers this state information at the endpoints of connections, at the entities that are utilizing the service of the network. Under this model, it is acceptable to lose the state information associated with an entity if, at the same time, the entity itself is lost; that is, there exists no longer any physical path over which any sort of communication with that entity can be achieved (*i.e.,* total partition). Ironically, these original design principles outlined in [34] favor precisely the opposite of what the scale-free modeling approach yields – no centrally located and highly connected "hubs" because their removal makes partitioning the network easy.

---

[4]This is not a universal phenomena. For instance [84] notes that some networks do exhibit hub-like structure, but it is the lack of universality that is important here, as exhibited by these and other counter examples.

3.2.3 An in-depth look at traceroute:
Examining a popular measurement technique

While the above-mentioned empirical, technological, and architectural arguments cast some serious doubts on the scale-free network modeling approach for the router topology of the Internet, they say nothing about the measurements that form the basis of this approach and has given it a sense of legitimacy among scientists in general and networking researchers in particular. To appreciate the full role that measurements play in this discussion, it is informative to revisit the original paper by Pansiot and Grad [119] that describes the measurement experiment, discusses the measurement technique used, and provides a detailed account of the quality of the data that form the basis of the scale-free approach towards modeling the Internet's router topology.

In essence, [119] describes the first (at that time) large-scale traceroute campaign performed for the main purpose of constructing a router graph of the Internet from actual Internet routes. Although traceroute-based, the authors of [119] quickly point out that their purpose of using the traceroute tool (*i.e.,* obtaining actual Internet routes to construct a router graph) differed from what V. Jacobson [78] had in mind when he originally designed the tool (*i.e.,* tracing a route from a source to a destination for diagnostic purposes). As a result, a number of serious issues arise that highlight why using the traceroute technique for the purpose of constructing a router graph is little more than an "engineering hack" and can certainly not be called a well-understood "measurement methodology."

IP alias resolution problem: One serious problem explained in detail in [119] with using traceroute-based data for constructing router graphs is that the traceroute tool only returns the IP addresses of the interface cards of the routers that the probe packets encountered on their route from the source to their destination. However, most routers have many interface cards, and despite many years of research efforts that have produced a series of increasingly sophisticated heuristics [15, 67, 140], the networking community still lacks rigorous and accurate methods for resolving the IP alias resolution problem; that is, determining whether two different interface IP addresses belong to or can be mapped to the same router. While the essence of this problem is illustrated in Figure 7, the impact it can have when trying to map a router topology of an actual network is shown in Figure 8.

(a)   (b)
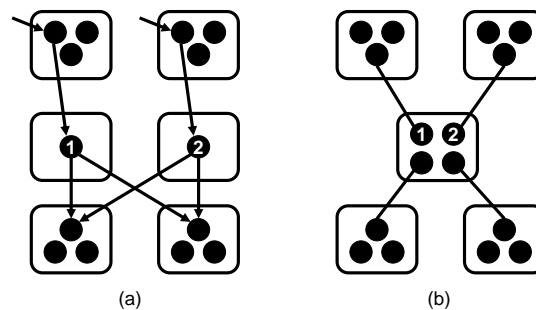
Figure 7: *The IP alias resolution problem. Paraphrasing Fig. 4 of [144], traceroute does not list routers (boxes) along paths but IP addresses of input interfaces (circles), and alias resolution refers to the correct mapping of interfaces to routers to reveal the actual topology. In the case where interfaces 1 and 2 are aliases, (b) depicts the actual topology while (a) yields an "inflated" topology with more routers and links. (First published in Notices of the American Mathematical Society, Volume 56, No.3 (May 2009): 586-599 [156]. Included here by permission.)*

Figure 8: *The IP alias resolution problem in practice. Shown is a comparison between the Abilene/Internet2 topology inferred by Rocketfuel (left) and the actual topology (top right). Rectangles represent routers with interior ovals denoting interfaces. The histograms of the corresponding node degrees are shown in the bottom right plot. (Reprinted from [141]; ©2008 ACM. Inc. Included here by permission.)*

Lesson 1: *Due to the absence of accurate and rigorous methods for solving the IP alias resolution problem, the actual values of the connectivity of each router (i.e., node degrees) inferred from traceroute measurements cannot be taken at face value.*

Opaque Layer-2 clouds: Another serious issue with using generic traceroute-based measurements for construction router graphs is also discussed at length in [119] and illustrated in Figure 9. Being strictly limited to IP or layer-3, the problem with traceroute is that it is incapable of tracing through opaque layer-2 clouds that feature circuit technologies such as Asynchronous Transfer Mode (ATM) or Multiprotocol Label Switching (MPLS). These technologies have the explicit and intended purpose of hiding the network's physical infrastructure from IP, so from the perspective of traceroute, a network that runs these technologies will appear to provide direct connectivity between routers that are separated by local, regional, national, or even global physical network infrastructures. An example of using traceroute to map a network that uses MPLS is depicted in Figure 9 and shows an essentially completely connected graph at Layer 3 with multiple high-degree nodes, even though the physical router topology is very sparse. Similarly, if traceroute encounters an ATM cloud, it falsely "discovers" a high-degree node that is really a logical entity – often an entire network potentially spanning many hosts or great distances – rather than a physical node of the Internet's router-level topology. Donnet *et al.* [44] found that at least 30% of the paths they tested traversed an MPLS tunnel.

Recent extensions of the ICMP protocol, using traceroute to trace through opaque MPLS clouds have become technically feasible [16], but operators often configure their routers to hide the MPLS tunnels by turning off this option [142]. Even then it may be possible to detect the MPLS tunnels [44], but the inference techniques are not a guarantee, and are quite particular to MPLS, which is not the only technique for creating tunnels, so there may still be some opaque networks to deal with. More to the point, even where such inferences are possible, most data sets do not contain this type of analysis, and most subsequent analyses of the data have ignored the issue.

Lesson 2: *Due to an inability of the generic traceroute technique to trace through opaque Layer-2 clouds, or understand the connectivity created by Layer-2 devices [104], the inferred high-degree nodes (i.e., routers with a large number of connections) are typically fictitious, an artifact of an imperfect measurement tool.*

Limited vantage points: We have commented earlier that since a router is fundamentally limited in terms of the number of packets it can process in any time interval, there is an inherent tradeoff in router configuration: it can support either a few high-throughput connections or many low-throughput connections. Thus, for any given router technology, a high-connectivity router in the core reflects a poor design decision – it will either have poor performance due to its slow connections or be prohibitively expensive relative to other options. Conversely, a good design choice is to deploy cheap high-degree router near the edge of the network and rely on the very technology that supports easy multiplexing of a large number of relatively low-bandwidth links. Unfortunately, neither the original traceroute-based study of Pansiot and Grad [119] nor any of the larger-scale campaigns that were subsequently performed by various network research groups have the ability to detect those actual high-degree nodes. The simple reason is that these campaigns lack access to a sufficient number of vantage points (*i.e.,* sources for launching traceroute probes and targets) in any local end-system to reveal these actual high connectivity patterns at the network's edge.

Lesson 3: *If there were high-degree nodes in the network, existing router technology relegates them to the edge of the network where no generic traceroute-based measurement campaigns is able to detect them because of a lack of vantage points nearby.*

There are other issues with large-scale traceroute campaigns that impact the quality of the resulting measurements and have received some attention in the literature. For example, the use of traceroute has been shown to make experimental data susceptible to a type of measurement bias in which some nodes
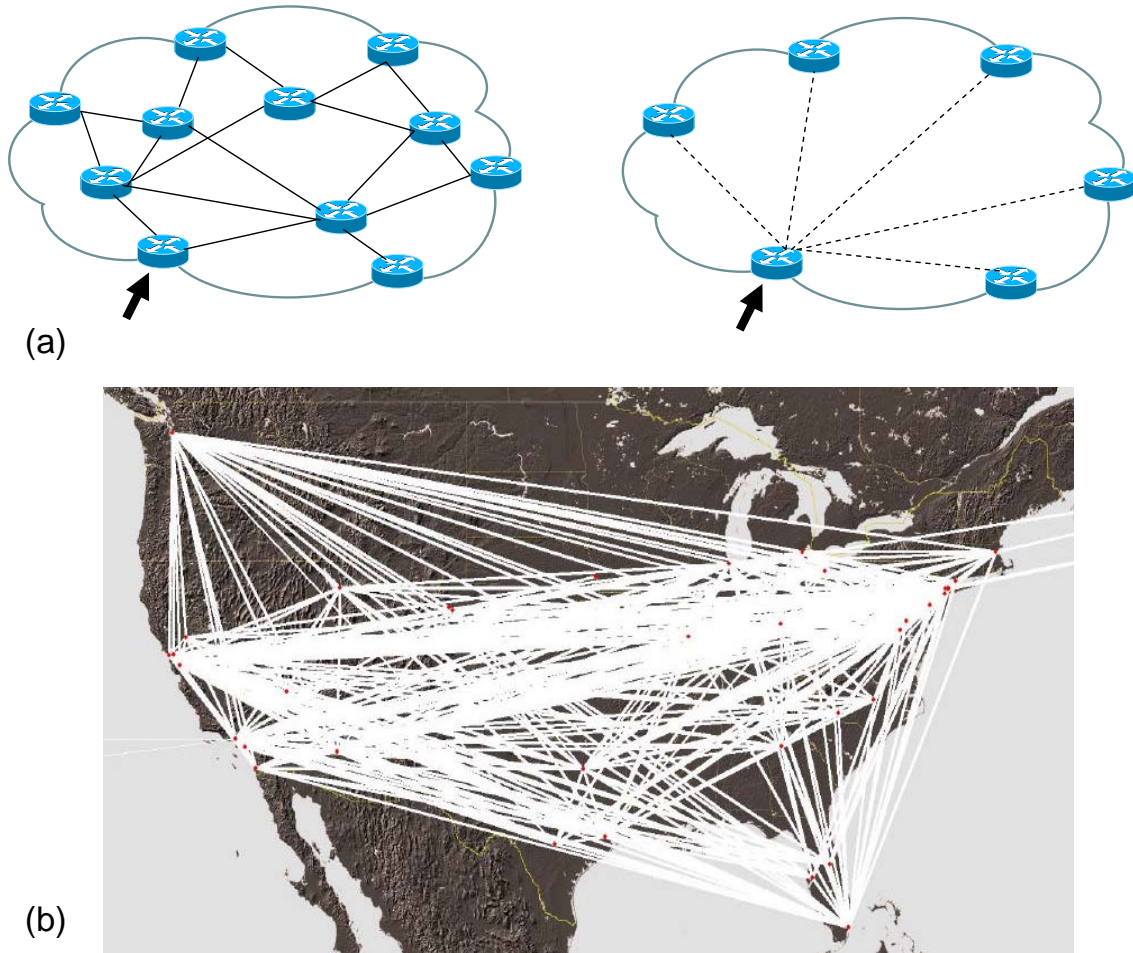
Figure 9: *How traceroute detects fictitious high-degree nodes in the network core. (a) The actual connectivity of an opaque layer-2 cloud, i.e., a router-level network running a technology such as ATM or MPLS (left) and the connectivity inferred by traceroute probes entering the network at the marked router (right). (b) The Rocketfuel-inferred backbone topology of AS3356 (Level3), a Tier-1 Internet service provider and leader in the deployment of MPLS. (Figure (b) reprinted from [144]; ©2002 ACM, Inc. Included here by permission.)*

of the network are oversampled, while others are undersampled. However, while this feature has received considerable attention [1, 91], in the presence of systematic errors due to an inability to perform accurate IP alias resolution or trace through opaque Layer-2 clouds, this work is largely of theoretical interest and of little practical relevance for modeling the Internet's router topology.

3.2.4   Just the facts: power-law scaling and router-level topologies

When applying lessons 1-3 to the main findings reported in the seminal papers discussed in §3.2.1 we are faced with the following facts:

Fact 1: A very typical but largely ignored fact about Internet-related measurements in general and traceroute measurements in particular is that what we can measure in an Internet-like environment is generally not the same as what we really want to measure (or what we think we actually measure). This is mainly because as a decentralized and distributed system, the Internet lacks a central authority and does not support third-party measurements.

Fact 2: A particularly ironic fact about traceroute is that the high-degree nodes it detects in the network core are necessarily fictitious and represent entire opaque layer-2 clouds, and if there are actual high-degree nodes in the network, existing technology relegates them to the edge of the network where no generic traceroute-based measurement experiment will ever detect them.

Fact 3: In particular, due to the inherent inability of traceroute to (i) reveal unambiguously the actual connectivity (*i.e.,* node degree) of any router, and (ii) correctly identify even the mere absence or presence of high-degree nodes (let alone their actual values), statistical statements such as those made in [49] claiming that the Internet's router connectivity is well described by a power-law distribution (or, for that case, any other type of distribution) cannot be justified with any reasonable degree of statistical confidence.

Fact 4: Since historical traceroute-based measurements cannot be taken at face value when (mis)using them for inferring router topologies and the inference results obtained from such data cannot be trusted, the claims that have been made about the (router-level) Internet in [5] are without substance and collapse under careful scrutiny.

In short, after almost 15 years of examining the idiosyncrasies of the traceroute tool, there exists overwhelming evidence that the sort of generic and raw traceroute measurements that have been used to date to infer the Internet's router topology are seriously flawed to the point of being essentially of no use for performing scientifically sound inferences. Yet, the myth that started with [49]; *i.e.,* the router topology of the Internet exhibits power-law degree distributions persists and continues to be especially popular with researchers that typically work in the field of network science and show in general little interest in domain-specific "details" such as traceroute's idiosyncrasies.

At the same time, it is worthwhile pointing out that most of the above-mentioned flaws and shortcomings of traceroute-based measurements are neither new nor controversial with networking researchers. In fact, when discussing the use of the traceroute tool as part of their original measurement experiment, the authors of [119] described many of the issues discussed in this section in great detail and commented on the possible implications that these inherently traceroute-related issues can have for constructing router graphs of the Internet. In this sense, [119] is an early example of an exemplary measurement paper, but unfortunately, it has been largely ignored and essentially forgotten. For one, [49], which critically relies on the data described in [119] for their power law claim for the Internet's router topology, fails to recognize the

relevance of these issues and does not even comment on them. Moreover, the majority of papers that have appeared in this area after the publication of [49] typically cite only [49] and don't even mention [119].

Traceroute-based measurements are not the only approach for obtaining router-level topologies, just the most commonly presented in the research literature. Network operators can obtain measurements of their own networks using much more accurate methods: for instance, from configuration files [52], or using route monitors [137], but those techniques require privileged access to the network, and so haven't been used widely for research. More recently, the mrinfo tool [109] has been used to measure topologies using IGMP (the Internet Group Management Protocol) [105, 120]. IGMP has the advantage that routers that respond provide much more complete information on their interfaces than those responding to traceroutes (so aliasing is less an issue), but there are still coverage problems created by lack of support, or deliberate filtering or rate limiting of responses to the protocol [102].

3.3   Network modeling: An exercise in reverse-engineering

The conclusion from the previous section that the available traceroute measurements are of insufficient quality to infer any statistical quantity of the data (including node degree distribution) with sufficient statistical confidence is a show-stopper for traditional network modeling. Indeed, given that the data cannot be trusted, relying on statistics of unknown accuracy (*e.g.,* by and large arbitrary node degrees) makes model selection precarious, and model validation in the sense of checking if the selected model describes the data "well" is an oxymoron – providing a "good" fit for statistical quantities of unknown accuracy is meaningless.

As such, the scale-free approach to modeling the Internet's router topology advanced in [5] is an example of what can go wrong if serious flaws of the underlying data are ignored and the available measurements are taken at face value. It should therefore come as no surprise that the resulting modeling framework and ensuing claims quickly collapse when scrutinized with readily available domain knowledge or vetted against alternative and solid sources of information. However, the lessons learned from this ill-fated approach to router topology modeling rises the question: What are viable alternative approaches to modeling the Internet's router topology that are by and large independent of the available but problematic traceroute measurements?

3.3.1   Router topology modeling as a constrained optimization problem

Having dismissed traceroute-based data as a source for informing our approach to modeling the Internet's router topology, we turn to readily available domain knowledge as critical alternate information source. To this end, we focus on the specific problem of modeling the physical infrastructure of a regional, national, or global Internet Service Provider (ISP).

The first key ingredient of this "first-principles" approach is the realization that ISPs design their physical infrastructures for a purpose; that is, their decisions are driven by possibly ISP-specific objectives and reflect trade-offs between what is feasible and what is desirable. While in general it may be difficult if not impossible to define or capture the precise meaning of a particular ISP's purpose for designing its network, an objective that expresses a desire to provide connectivity to the rest of the Internet for its end users and an ability to carry an expected traffic demand efficiently and effectively, subject to prevailing economic and technological constraints, is unlikely to be far from the "true" purpose.

The second ingredient concerns the trade-offs an ISP has to made between what is feasible (in terms of available products sold by the different router vendors) and what is desirable (in terms of cost, performance, ease-of-management or other criteria for the built-out router topology). In particular, router

technology constraints are a significant force shaping network connectivity at the router-level and, in turn, router topology design. Due to hard physical limits, even the most expensive and highest-capacity router models available on the market in any given year operate within a "feasible region" and corresponding "efficiency frontier" of possible bandwidth-degree combinations; that is, they can be configured to either have only a few high-bandwidth connections and perform at their capacity or have many low-bandwidth connections and tolerate a performance hit due to the overhead that results from the increased connectivity.

Similarly, economic considerations also affect network connectivity and router topology design. For example, the cost of installing and operating physical links in a network can often dominate the cost of the overall router infrastructure. In essence, this observation creates enormous practical incentives to design the physical plant of an ISP so as to keep the number of links small and avoid whenever possible long-haul connections due to their high cost. These incentives to share costs via multiplexing impact and are impacted by available router technologies and argue for a design principle for an ISP's router topology that favors aggregating traffic at all levels of network hierarchy, from its periphery all the way to its core.

The third and final key ingredient of the proposed first-principle alternative to router topology modeling is concerned with the role that randomness plays in this approach. Recall that the traditional approach is typically graph theory-based where randomness is explicit and appears in the form of a series of coin tosses (using potentially bias coins as in the case of scale-free networks of the preferential attachment type) that determine whether or not two nodes (*i.e.,* routers) are connected by a physical link, irrespective of the type of routers involved or link considered. In stark contrast, in our approach, randomness enters in a very different and less explicit manner, namely in terms of the uncertainty that exists about the "environment" (*i.e.,* the traffic demand that the network is expected to carry). Moreover, irrespective of the model chosen for quantifying this uncertainty, the resulting network design is expected to exhibit strong robustness properties with respect to changes in this environment.

When combining all three ingredients to formulate an ISP's router topology design problem, the mathematical modeling language that naturally reflects the objectives of an ISP, its need to adhere to existing technology constraints and respect economic considerations, and its desire to operate effectively and efficiently in light of the uncertainly in the environment is *constrained optimization.* Thus, we have changed network modeling from an exercise in model fitting into an exercise in reverse-engineering and seek a solution to a constrained optimization problem formulation that captures by and large what the ISP can afford to build, operate, and manage (*i.e.,* economic considerations), satisfies the hard constraints that technology imposes on the network's physical entities (*i.e.,* routers and links), and is robust to changes in the expected traffic that is supposed to handle

3.3.2 Heuristically optimal router topologies

In the process of formulating the design of an ISP's router topology as a constrained optimization problem, we alluded to a synergy that exists between the technological and economic design issues with respect to the network core and the network edge. The all-important objective to multiplex traffic is supported by the types of routers available on the market. In turn, the use of these products re-enforces traffic aggregation everywhere in the network. Thus, the trade-offs that an ISP has to make between what is technologically feasible versus economically sensible can be expected to yield router topologies where individual link capacities tend to increase while the degree of connectivity tends to decrease as one moves from the network edge to its core.

This consistent picture with regard to the forces that by and large govern the built-out and provisioning of an ISP's router topology and include aspects such as equipment constraints, link costs, and bandwidth

demands suggests that the following type of topology is a reasonably "good" design for a single ISP's physical plant: (i) Construct a core as a loose mesh of expensive, high-capacity, low-connectivity routers which carry heavily aggregated traffic over high-bandwidth links. (ii) Support this mesh-like core with hierarchical tree-like structures at the edge of the network for the purpose of aggregating traffic from end users via cheaper, lower-capacity, high-connectivity routers. (iii) Augment the resulting structure with additional connections at various selectively-chosen places to provide a degree of redundancy and robustness to component failures. The result is a topology that has a more or less pronounced backbone, which is fed by tree-like access networks, with additional links added for redundancy and resilience. We refer to this design as *heuristically optimal* to reflect its consistency with real design considerations and call the resulting "solutions" *heuristically optimal topologies*, or HOT for short. Note that such HOT models have been discussed earlier in the context of highly organized/optimized tolerances/tradeoffs [24, 48].

An important aspect of the proposed HOT models is that even though we have formulated the design of an ISP's router topology as a constrained optimization problem that could in principle be solved optimally, we are typically not concerned with a network design that is "optimal" in a strictly mathematical sense and is also likely to be NP-hard. Instead, our interest is in solutions that are "heuristically optimal" in the sense that they result in "good" performance. The main reason for not pursuing optimal solutions more aggressively is the imprecise nature of essentially all ingredients of the constrained optimization problem of interest. For one, it is unrealistic to expect that an ISP's true objective for building out and provisioning its physical infrastructure can be fully expressed in mathematical terms as an objective function. Furthermore, a bewildering number of different types of routers and connections make it practically impossible to account for the nuances of the relevant feasible regions or efficiency frontiers. Finally, any stochastic model for describing the expected traffic demand is an approximation of reality or at best based on imprecise forecasts. Given this approximate nature of the underlying constrained optimization problem, we seek solutions that captures by and large what the ISP can afford to build, operate, and manage (*i.e.,* economic considerations), satisfy some of the more critical hard constraints that technology imposes on the network's physical entities (*i.e.,* routers and links), and exhibit strong robustness properties with fluctuations in the expected traffic demands (*i.e.,* insensitivity to changes in the uncertain environment).
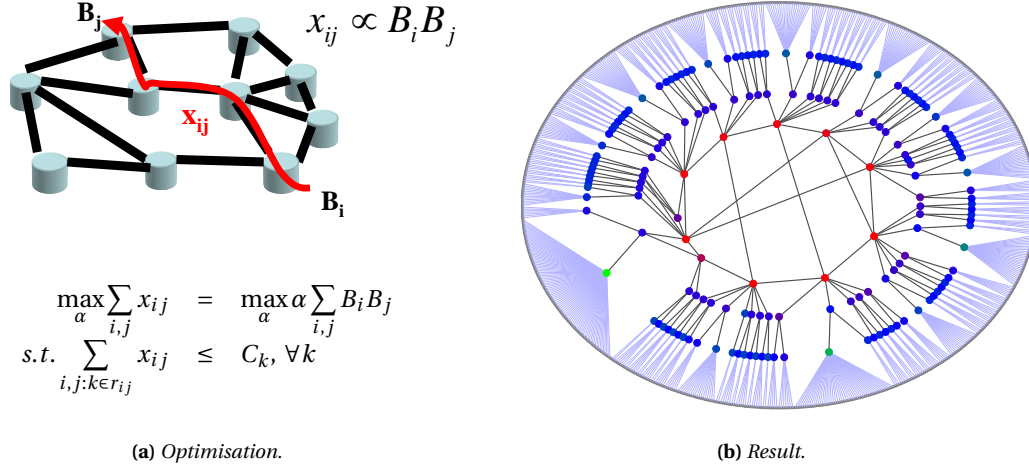
3.3.3   A toy example of a HOT router topology

To illustrate the proposed HOT approach, we use a toy example that is rich enough to highlight the key ingredients of the outlined first-principles methodology and demonstrate its relevance for router topology modeling as compared to the popular model-fitting approach. It's toy nature is mainly due to a number of simplifying assumptions we make that facilitate the problem formulation. For one, by simply equating throughput with revenues, we select as our objective function the maximum throughput that the network can achieve for a given traffic demand and use it as a metric for quantifying the performance of our solutions. Second, considering an arbitrary distribution of end-user traffic demand B_i, we assume a gravity model for the unknown traffic demand; that is, assuming shortest-path routing, the demands are given by the traffic matrix element $x_{i,j} = \alpha B_i B_j$ between routers i and j for some constant α. Lastly, we consider only one type of router and its associated technologically feasible region; that is, (router degree, router capacity)-pairs that are achievable with the considered router type, and implicitly avoid long-haul connections due to their high cost.

The resulting constrained optimization problem can be written in the form

$$\max_{\rho} \sum x_{i,j} \quad \text{such that} \quad A\mathscr{X} \leq C, \tag{1}$$

25

where $\mathscr{X}$ is the vector obtained by stacking all the demands $x_{i,j}$; A is the routing matrix obtained by using standard shortest path routing and defined by $A_{k,l} = 1$ or 0, depending on whether or not demand l passes through router k; and C is the vector consisting of the router degree-bandwidths constraints imposed by the technologically feasible region of the router at hand.


$$x_{ij} \propto B_i B_j$$

$$\max_{\alpha} \sum_{i,j} x_{ij} = \max_{\alpha} \alpha \sum_{i,j} B_i B_j$$
$$s.t. \sum_{i,j:k \in r_{ij}} x_{ij} \leq C_k, \forall k$$

(a) *Optimisation.* **(b)** *Result.*

Figure 10: *Generating networks using constrained optimization. (a) Engineers view network structure as the solution to a design problem that measures performance in terms of the ability to satisfy traffic demand while adhering to node and arc capacity constraints. (b) A network resulting from heuristically optimized tradeoffs (HOT). This network has very different structural and behavioral properties, even when it has the same number of nodes, links, and degree distribution as a scale free network shown in Figure 9. (First published in Notices of the American Mathematical Society, Volume 56, No.3 (May 2009): 586-599 [156]. Included here by permission.)*

While all the simplifying assumptions can easily be relaxed to allow for more realistic objective functions, more heterogeneity in the constraints, or more accurate descriptions of the uncertainty in the environment, Figure 10 illustrates the key characteristics inherent in a heuristically optimal solution of such a problem. First, the cost-effective handling of end user demands avoids long-haul connections (due to their high cost) and is achieved through traffic aggregation starting at the edge of the network via the use of high-degree routers that support the multiplexing of many low-bandwidth connections. Second, this aggregated traffic is then sent toward the *backbone* that consists of the fastest or highest-capacity routers (*i.e.,* having a small number of very high-bandwidth connections) and that forms the network's mesh-like core. The result is a network of the form described earlier: a more or less explicit backbone representing the network core and tree-like access networks surrounding this core, with additional connections as backup in case of failures or congestion.

The realism of this reverse-engineering approach to router topology modeling is demonstrated in Figure 11 which shows the router topologies of two actual networks – CENIC (circa 2004) and Abeline (circa 2003).

3.3.4 On the (ir)relevance of node degree distributions

The above description of our engineering-based first-principles approach to router topology modeling shows that node degree distributions in general and power low-type node degree distributions in particular
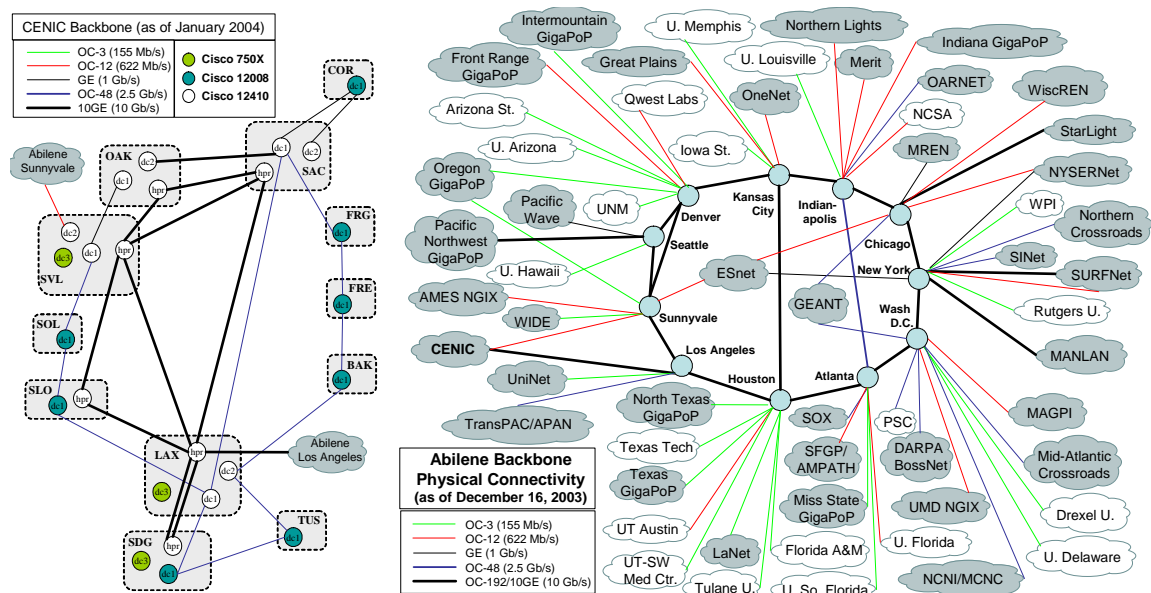
Figure 11: *CENIC and Abilene networks. (Left): CENIC backbone. The CENIC backbone is comprised of two backbone networks in parallel – a high performance (HPR) network supporting the University of California system and other universities, and the digital California (DC) network supporting K-12 educational initiatives and local governments. Connectivity within each POP is provided by Layer-2 technologies, and connectivity to the network edge is not shown. (Right): Abilene network. Each node represents a router, and each link represents a physical connection between Abilene and another network. End user networks are represented in white, while peer networks (other backbones and exchange points) are represented in gray. Each router has only a few high bandwidth connections, however each physical connection can support many virtual connections that give the appearance of greater connectivity to higher levels of the Internet protocol stack. ESnet and GÉANT are other backbone networks. (Reprinted from [93]; ©2004 ACM, Inc. Included here by permission.)*

are clearly a non-issue and play no role whatsoever in our formulation of an ISP router topology design as a constrained optimization problem. Thus, we achieved our goal of developing a network modeling approach that does not rely in any way on the type of measurements that have informed previous network modeling approaches but have been shown earlier to be of insufficient quality to be trusted to form the basis of any scientifically rigorous modeling pursuit.

However, even if the available traceroute measurements could be trusted and taken at face value, the popular approach to network modeling that views it as an exercise in model fitting is by itself seriously flawed, unless it is accompanied by a rigorous validation effort. For example, assuming that the data can be trusted so that a statistic like an inferred node degree distribution is indeed solid and reliable. In this case, who is to say that a proposed model's ability to match this or any other commonly considered statistics of the data argues for its validity, which is in essence the argument advanced by traditional approaches that treat network modeling as an exercise in model fitting? It is well known in the mathematics literature that there can be many different graph realizations for any particular node degree sequence and there are often significant structural differences between graphs having the same degree sequence. Thus, two models that match the data equally well with respect to some statistics can still be radically different in terms of other properties, their structures, or their functionality. A clear sign of the rather precarious current state of network-related modeling that is rooted in the almost exclusive focus on model fitting is that the same underlying data set can give rise to very different, but apparently equally "good" models, which in turn can give rise to completely opposite scientific claims and theories concerning one and the same observed phenomenon. Clearly, network modeling and especially model validation ought to mean more than being able to match the data if we want to be confident that the results that we derive from our models are relevant in practice.

To illustrate these points, Figure 12 depicts five representative toy networks, constructed explicitly to have one and the same node degree distribution. This distribution is shown in plot (a) and happens to be the one of our HOT router topology example in Figure 10. While plots (b) and (c) show two scale-free networks constructed according to the preferential attachment method and general random graph method, respectively, plots (d)-(f) are three different HOT examples, including our earlier example in Figure 10 (plot (e)) and a sub-optimal or poorly-engineered HOT topology in (f). While the differences among these five topologies with identical node degree distributions are already apparent when comparing their connectivity structures, they can be further highlighted by considering both a performance-related and a connectivity-only topology metric. In particular, the performance-related metric *Perf(g)* for a given network g is defined as *Perf(g)* $= max_\alpha \sum x_{i,j}$ s.t. $RX \le C$ and represents the maximum throughput with gravity flows of the network g. In contrast, the connectivity-only topology metric $S(g)$ is the network likelihood of g defined as $S(g) = \sum_{(i,j)\in E(g)} \omega_i \omega_j / s_{max}$ where ω_i denotes the degree of node i, $E(g)$ is the set of all edges in g, and s_{max} is a normalization constant. For a justification of using the $S(g)$ metric to differentiate between random networks having one and the same node degree sequence, we refer to [92].

While computing for each of the five networks their *Perf(g)*-value is straightforward, evaluating their network performance requires further care so as to ensure that the different network have the same total "cost", where cost is measured in number of routers. When simultaneously plotting network performance versus network likelihood for all five networks models in Figure 13, a striking contrast is observed. The well-engineered HOT networks (d) and (e) have high performance and low likelihood while the random degree-based networks (b) and (c) have high likelihood but low performance. To contrast, network (f) has both low performance and low likelihood and is proof that networks can be designed to have poor performance. The main reason for the degree-based models to have such poor performance is exactly the presence of the highly connected "hubs" that create low-bandwidth bottlenecks. The two HOT models' mesh-like cores, like real ISP router topologies, aggregate traffic and disperse it across multiple
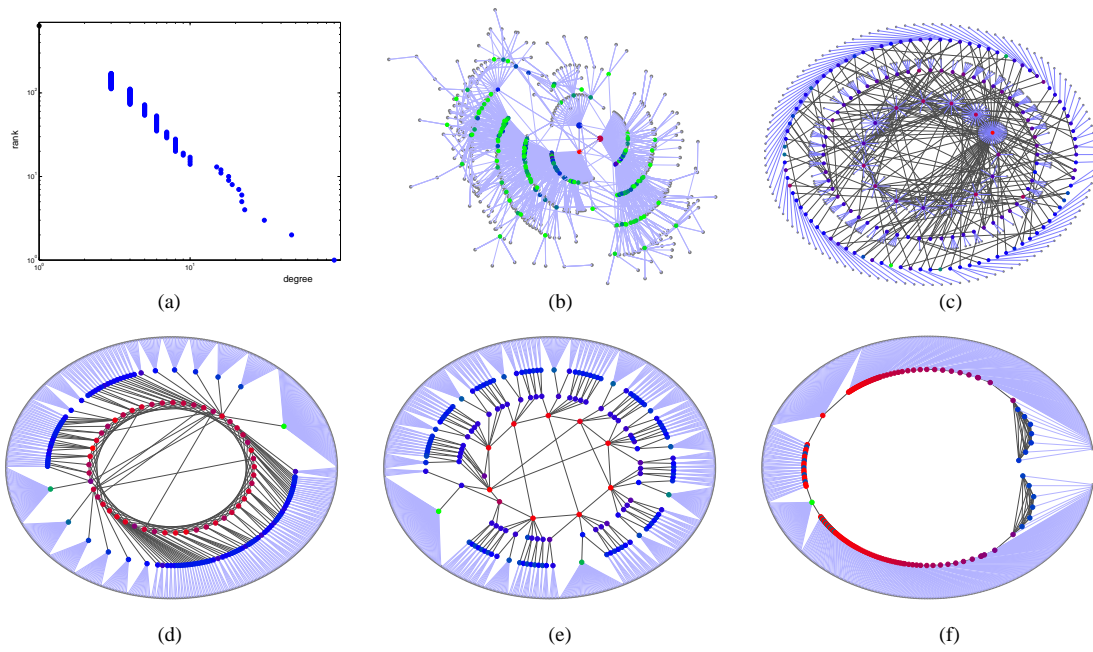
Figure 12: *Five networks having the same node degree distribution. (a) Common node degree distribution (degree versus rank on log-log scale); (b) Network resulting from preferential attachment; (c) Network resulting from the GRG method; (d) Heuristically optimal topology; (e) Abilene-inspired topology; (f) Sub-optimally designed topology. (Reprinted from [93]; ©2004 ACM, Inc. Included here by permission.)*

high-bandwidth routers.
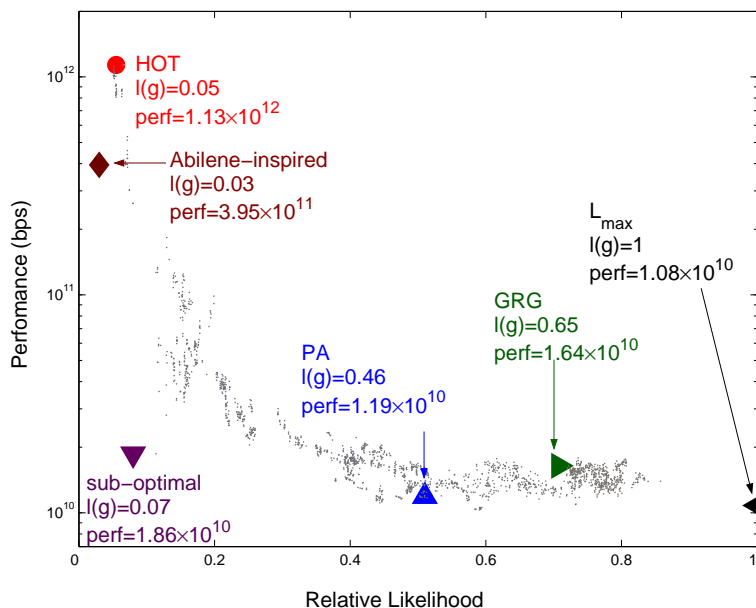
Figure 13: *Performance vs. likelihood for each of the topologies in Figure 12, plus other networks (grey dots) having the same node degree distribution obtained by pairwise random rewiring of links. (Reprinted from [93]; ©2004 ACM, Inc. Included here by permission.)*

The interpretation of this picture is that a careful design process explicitly incorporating technological constraints can yield high-performance topologies, but these are extremely rare from a probabilistic graph point of view. In contrast, equivalent scale-free networks constructed by generic degree-based probabilistic constructions result in more likely, but poorly-performing topologies. Consistent with this, the "most likely" network (included in Figure 13) has also sub-par performance. This picture can be further enhanced when considering alternative performance measures such as the distribution of end user bandwidths and router utilization. As detailed in [93], the heuristically optimal networks (d) and (e) achieve high utilization in their core routers and support a wide range of end-user bandwidth requirements. In contrast, the random degree-based networks (b) and (c) saturate only their "hub" nodes and leave all other routers severely underutilized, thus providing uniformly low bandwidth and poor performance to their end-users. A main lesson from this comparison of five different networks with identical node degree distributions for network modeling is that *functionality (*e.g., *performance) trumps structure (*e.g., *connectivity).* That is, connectivity-only metrics are weak discriminators among all graph of a given size with the same node degree distribution, and it requires appropriate performance-related metrics to separate "the wheat from the chaff."

We explained earlier that on the basis of currently available traceroute measurements, claims of power-law relationships have no substance as far as the Internet's router topology is concerned. However, by examining available router technologies and models, we have also shown that it is certainly conceivable that the actual node degrees of deployed routers in an actual ISP can span a range of 2-3

orders of magnitude; that is, the corresponding node degree distribution exhibits high variability, without necessarily conforming to a power law-type distribution. At the same time, Figure 12 illustrates that irrespective of the type of node degree distribution, graphs with identical node degree distributions can be very different in their structure and differ even more drastically in terms of their functionality (*e.g.*, performance). What is also true is that the same core network design can support many different end-user bandwidth distributions and that by and large, the variability in end-user bandwidth demands determines the variability of the node degrees in the resulting network. To illustrate, consider the simple example presented in Figure 14, where the same network core supports different types of variability in end user bandwidths at the edge (and thus yields different overall node degree distributions). The network in Figure 14(a) provides uniformly high bandwidth to end users; the network in Figure 14(b) supports end user bandwidth demands that are highly variable; and the network in Figure 14(c) provides uniformly low bandwidth to end users. Thus, from an engineering perspective, not only is there not necessarily any implied relationship between a network node degree distribution and its core structure, there is also no implied relationship between a network's core structure and its overall degree distribution.

Thus, the proposed engineering-based first-principles approach to modeling the Internet router topology demystifies power law-type node degree distributions altogether by identifying its root cause in the form of high variability in end-user bandwidth demands. In view of such a simple physical explanation of the origins of node degree variability in the Internet's router-level topology, Strogatz' question, paraphrasing Shakespeare's Macbeth, "... power-law scaling, full of sound and fury, signifying nothing?" [145] has a resounding affirmative answer.

3.4   A look ahead

Late in the last century, when router-level topology modeling started to turn into a measurement-driven research activity, the conventional wisdom was to start with traceroute-based measurements, use them to infer router-level connectivity, and argue for the validity of a proposed model if it faithfully reproduces certain statistics of the inferred connectivity structure (*e.g.,* node degree distribution). However, the last decade of Internet topology research has shown that this traditional and widely-used approach to router-

(a)   (b)   (c)
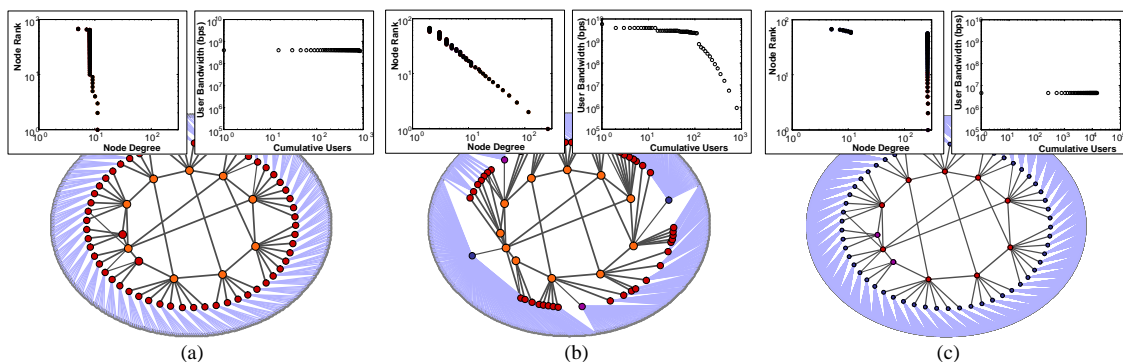
Figure 14: *Distribution of node degree and end-user bandwidths for several topologies having the same core structure: (a) uniformly high bandwidth end users, (b) highly variable bandwidth end users, (c) uniformly low bandwidth end users. (Reprinted from [93]; ©2004 ACM, Inc. Included here by permission.)*

topology modeling is flawed in more than one way, and we have collected and presented this gradually accumulating evidence in this section – the underlying measurements are highly ambiguous (§3.2), the inferred connectivity structures are erroneous (§3.3), and the resulting models are infeasible and/or do not make sense from an engineering perspective because they are either too costly, have extremely poor performance, or cannot be built with from existing technology in the first place.

This section also describes and reports on an alternative design-based approach to router-level topology modeling and generation that has come into focus during the last 5-10 years and represents a clean break with tradition. The most visible sign of this break is the emergence of constrained optimization as new modeling language, essentially replacing the traditional language of random graph theory. While the latter treats nodes and links as largely generic objects and focuses almost exclusively on structural aspects such as connectivity, the former supports a much richer treatment of topologies – nodes and links are components with their own structure, constraints, and functionality, and their assembly into a topology that is supposed to achieve a certain overall objective and should do so efficiently and effectively within the given constraints on the individual components or the system as a whole is the essence of the constrained optimization-based network design approach. In essence, this approach echoes what was articulated some 15 years ago in [23, 42, 165], but it goes beyond this prior work in terms of empirical evidence, problem formulation, and solution approach. As a result, the described design-based approach has by and large put an end to graph theory-based router topology modeling for the Internet.

At the same time, the design-based approach that has been developed and reported in bits and pieces in the existing literature and is presented in this section for the benefit of the reader in one piece has also far-reaching implications for Internet topology research in general and router-level topology modeling, analysis, and generation in particular. For one, it shows the largely superficial nature of router-level topology generators that are based on graph-theoretic models. As appealing as they may be to a user because of their simplicity (after all, all a user has to specify is in general the size of the graph), they are by and large of no use for any real application where details like traffic, routing, capacities, or functionality matter.

Second, while the design-based approach yields realistic router-level topology models that are inherently generative in nature, it puts at the same time an end to the popular request for a largely generic black-box-type topology generator. Users in real need for synthetic router-level maps have to recognize that this need doesn't come for free. Instead, it comes with the responsibility to provide detailed input in terms of expected customers – their geographic dispersion, and the traffic matrix (see [150] for more details) – design objectives and constraints, etc. In addition, the level of detail required of a generated ISP router-level topology (*e.g.,* POP-, router-, interface card-level) depends critically on and cannot be separated from the purpose for which these generated maps will be used. Again, this puts a considerable burden on the user of a synthetically generated map and tests her understanding of the relevant issues to a degree unheard of in Internet topology modeling in the past.

Third, the explicit focus of the design-based approach on ISPs as crucial decision makers renders the commonly-expressed desire for synthetic router-level maps of the global Internet largely pointless. The Internet is a network of networks, with the sovereign entities being the autonomous systems (ASes). A subset of these ASes that are in the business of providing network service to other ASes or Internet access to end users are owning and operating their networks that together make up much of the physical infrastructure of the global Internet. As a result, a key first step in understanding the structure and temporal evolution of the Internet at the different physical and logical layers is to study the physical infrastructures of the service and access providers' networks and how they react in response to changes in the environment, technology, economy, etc.

Finally, once we have a more-or-less complete picture of the router-level topology for the individual

ISPs, we can start interconnecting them at common locations, thereby bringing ISP router-level and AS-level topology under one umbrella. In the process, it will be critical to collapse the detailed router-level topologies into their corresponding PoP-level maps which are essentially the geographic maps mentioned in the context of the ARPANET in §3.1 and serve as glue between the detailed router-level topologies and an appropriately defined and constructed AS-level topology of the Internet. For a faithful and realistic modeling of this combined router-, POP-, and AS-level structure of the Internet, it will be important to account for the rich structure that exists in support of network interconnections in practice. This structure includes features such as third-party colocation facilities that house the PoPs of multiple ASes in one and the same physical building. It also includes components of the Internet's infrastructure such as Internet eXchange Points (IXPs). This existing structure is inherently non-random but is a reflection of the incentives that exist, on the one hand, for network and access providers to build, manage, and evolve their physical infrastructures and, on the other hand, for content providers, CDNs, and cloud providers to establish peering relationships with interested parties. Importantly, neither such structures nor incentives precludes an application of the described constrained optimization-based approach to network design; they merely require being creative with respect to formulating a proper objective function, identifying the nature of the most critical constraints, and being able to pinpoint the main sources of uncertainty in the environment.

3.5 Notes

The primary sources for the material presented in this section are:

[93] L. Li, D. Alderson, J.C. Doyle and W. Willinger. A first principles approach to understanding the Internet's router-level topology, *Proc. ACM SIGCOMM'04, ACM Computer Communication Review* 34(4), 2004.

[46] J. C. Doyle, D. L. Alderson, L. Li, S. Low, M. Roughan, S. Shalunov, R. Tanaka, and W. Willinger. The "robust yet fragile" nature of the Internet, *PNAS* 102(41), 2005.

[156] W. Willinger, D. Alderson, and J.C. Doyle. Mathematics and the Internet: A Source of Enormous Confusion and Great Potential, *Notices of the AMS* 56(5), 2009.

An excellent short treatment of the discussion in §3.3 about network modeling as an exercise in reverse-engineering vs. as an exercise in model fitting can be found in Chapter 10 of the recent book

[29] M. Chiang. Networked Life: 20 Questions and Answers. Cambridge University Press, 2012.

For additional and more in-depth reading materials we point to

[7] Alderson, D., Li, L., Willinger, W., and Doyle, J.C. Understanding Internet Topology: Principles, Models, and Validation, *IEEE/ACM Transactions on Networking* 13(6): 1205-1218, 2005.

[85] B. Krishnamurthy and W. Willinger. What are our standards for validation of measurement-based networking research? *Computer Communications* 34, 2011.

For some "food for thought" regarding topics such as power law distributions and scale-free networks, and network science, we recommend

[157] W. Willinger, D. Alderson, J.C. Doyle, and L. Li. More "normal" than normal: Scaling distributions and complex systems. In: R.G. Ingalls, M.D. Rossetti, J.S. Smith, and B.A. Peters (Editors). IEEE. Proc. of the 2004 Winter Simulation Conference, Piscataway, NJ, 2004.

[106] M. Mitzenmacher. A Brief History of Generative Models for Power Law and Lognormal Distributions, Internet Mathematics 1(2):226-251, 2004.

[82] E. Fox Keller. Revisiting "scale-free" networks, BioEssays 27(1)):1060-1068, 2005.

[11] A.-L. Barabasi. Scale-free Networks: A Decade and Beyond, Science 325, 2009.

[12] A.-L. Barabasi. The network takeover, Nature Physics 8, pp. 14-16, 2012.

[107] M. Mitzenmacher. Editorial: The Future of Power Law Research. *Internet Mathematics*, vol. 2. no. 4, pp. 525-534, 2006.

4 AS-level topology

When trying to establish a precise meaning or interpretation of the use of "Internet topology," in much of the existing literature, we find that the phrase has often been taken to mean a virtual construct or graph created by the Border Gateway Protocol (BGP) routing protocol. Commonly referred to as the inter-domain or Autonomous-System (AS) topology — named after the logical blocks (ASes) that are used in BGP to designate the origin and path of routing announcements — it is this particular connectivity structure that we focus on in this section, though we will see that the notion of the AS-topology is more slippery than commonly imagined. In particular, we will discuss some of the main issues that arise in the context of studying the Internet's AS topology (ranging from proper definitions and interpretations of this construct to measurements) and focus less on modeling-related aspects as they are still in their infancy, especially when compared to the advances in router-topology modeling described in §3.

4.1 A look back

As far as we know, the first researchers to use BGP-based measurements in the form of route monitor data for topology-related work were Govindan and Reddy [60], who introduced the notion of the *inter-domain topology* defined as "the graph of domains and the inter-domain peering relationships." However, although they were quite specific in regard to being interested in routing, the concept was reused when Faloutsos *et al.* [49] coined the term "Internet topology", a paper that is more widely cited (at least outside of the network research literature) than [60]. The paper [49] is responsible for advancing the alluring notion that the inter-domain topology of the Internet is a well-defined object and can be *accurately* obtained and reconstructed from the available BGP route monitor data. As we shall see, this is not at all the case, and it has fed into a large subsequent scientific literature, already discussed earlier, *e.g.,* [13, 161].

The problems lie in the very definitions of the AS-topology and the measurements that have been used to study this topology (we return to the measurements in §4.2 below). In terms of definitions, in the context of the AS-level Internet, it is tempting to simply equate a node with an AS, but this begs the question what an AS really is. The term refers formally to the AS number (ASN) allocated by IANA (Internet Assigned Numbers Authority) or the Regional Internet Registries (RIRs). An ASN is tendered to enable routing using BGP. This is **not** equivalent to the popular view that associates an AS with a set of routers that appear to the outside as if they formed a single coherent system with a 1:1 mapping between it and some administering company.

For instance, an organization may often own a router which has at least one interface IP address belonging to another organization. In fact, many point-to-point IP links occur across a "/30" subnet. When the link joins two networks, this subnet must be allocated from the IP blocks of one or the other

connecting network, and so most such connections result in IP addresses from neighboring ASes appearing locally.

Another problem arises from the fact that although an AS is often considered to correspond to a single technical administrative domain, *i.e.,* a network run by one organization, it is common practice for a single organization to manage multiple ASes, each with their own ASN [22]. For instance, Verizon Business (formerly known as UUNET) uses ASNs 701, 702, 703 to separate its E-BGP network into three geographic regions, but runs a single IGP instance throughout its whole network. In terms of defining nodes of a graph, these three networks are all under the same operational administrative control, and hence should be viewed as a single node. On the other hand, as far as ASNs are concerned, they are different and should be treated as three separate nodes. The situation is actually more complex since corporations like Verizon Business own some 200+ ASNs [22] (not all are actually used, though). In many of these cases, a clear boundary between these multiple ASes may not really exist, thus blurring the definition of the meaning of a node in an AS graph. Similar problems can arise when a single AS is managed by multiple administrative authorities which consist of individuals from different corporations. For example, AS 2914 is run partially by NTT/America and partially by NTT/Asia.

All this presumes that an AS is a uniform, contiguous entity, but that is not necessarily true [110,111]. An AS may very well announce different sets of prefixes at different exit points of its network, or use BGP to balance traffic across overloaded links (other reasons for heterogeneous configurations are reported in [21]). Figure 15 illustrates the problem. The AS-graph simplifies, in some cases grossly, the very complicated structure of the entities involved, which are often heterogeneous, and not necessarily even contiguous either geographically or logically.

For all these reasons, it should be clear that modeling an AS as a single *atomic* node without internal (or external) structure is overly simplistic for most practical problems. Moreover, these issues cannot simply be addressed by moving towards graph representations that can account for some internal node structure (such as in [110]), mainly because BGP is unlikely to reveal sufficient information to infer the internal structure for the purpose of faithful modeling.

Moreover, the AS-graph treats ASes as nodes, with connecting edges, but the real situation is much more complex. ASes are complex networks in their own right, and are connected sometimes by multiple edges (Mérindol *et al.* [105] found that over half of the ASes they studied were connected by multiple links), and sometimes through Internet eXchange Points (IXPs) that connect multiple ASes. In fact, the traditional approach of modeling the AS-level Internet as a simple connected di-graph is an abstraction incapable of capturing important facets of the rich semantics of real-world inter-AS relationships, including different interconnections for different policies and/or different interconnection points [110,111]. The implications of such abstractions need to be recognized before attributing network-specific meaning to findings derived from the resulting models.

4.2   Know your measurements

In studying the AS-level Internet, there are some critical differences compared to looking at the router-level Internet:

- No-one "owns" the AS structure. There isn't anyone with the type of privileged view that a network operator has of its own network. There are tens of thousands of ASes, and so we can't reasonably expect to consult all of them to collate a picture either.

- ASes are not "nodes". They are complex in their own right, so viewing the AS-level Internet as an AS-graph is a big abstraction of reality.

(a) *A simple section of the "AS-graph".*   (b) *A possible picture of the router-level connectivity.*
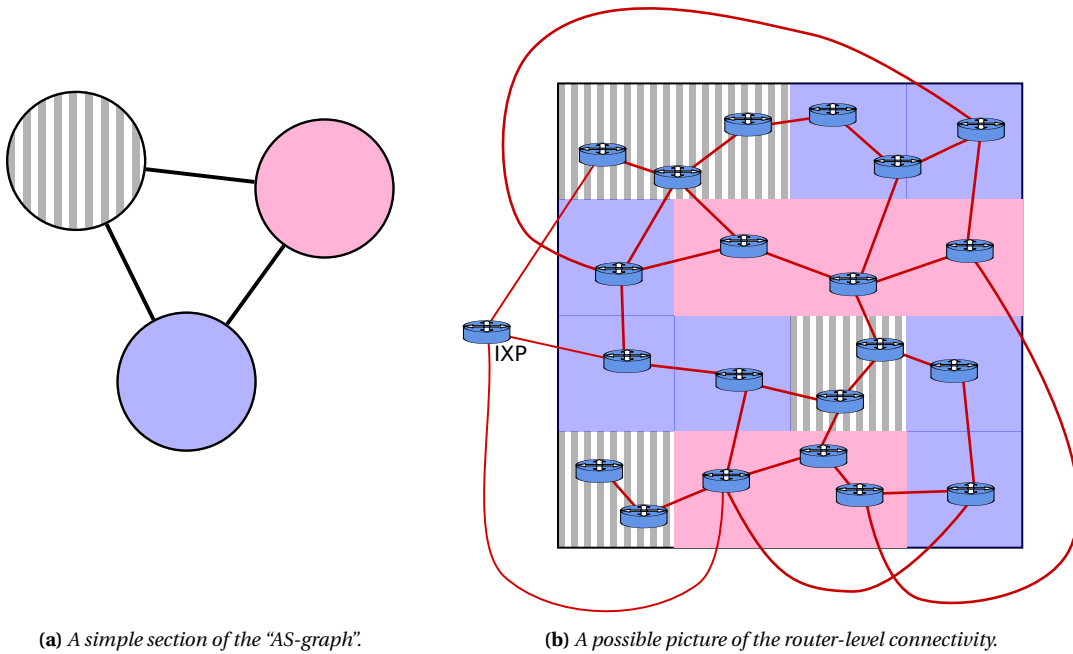
Figure 15: *An illustration of the obfuscation of the AS-graph (in the vein of [61]). The graph may appear simple, but hides heterogeneous, non-atomic, dis-contiguous entities and interconnects. At the minimum, this should illustrate the dangers of talking about the "Internet" graph.*

- Routing between ASes is very different from routing within ASes and highlights the difference between graph representations that reflect "reachability" vs. "connectivity" information.

These differences create interesting problems and opportunities for measurements, some with parallels to the router-level measurement problems and others without any such parallels.

4.2.1 Data-plane vs. control-plane measurements

As discussed in §3.2, despite all its deficiencies, traceroute has been the method-of-choice for obtaining router-level measurements. As a prime example of an active measurement tool that is confined to the data plane (*i.e.,* probe packets take the same paths as generic data packets), traceroute has also been used to obtain information about the AS topology but has additional problems in this domain.

Apart from the already problematic issues (*e.g.,* load-balancing, aliasing, missing data), IP addresses along traceroute paths must now be mapped to ASes. This mapping is even harder than the mapping to routers, not just because the data for doing so is inaccurate or incomplete (*e.g.,* IP to organization allocations may not work because an organization does not directly correspond to an AS), but also because the border of an AS is not well-defined in terms of IP addresses. It is common for a link between two ASes to come from a subnet allocated by one of the ASes, resulting in an interface in the other network with an address that is not its own [100, 101]. The problem is further complicated by variations such as anycast or Multiple Origin ASes [167], which provide yet another set of counter-examples to a straightforward mapping between AS and address space. Some work has concentrated on trying to improve

the mapping [120], and these represent technical advances, but it is important to understand that the fundamental difficultly lies in the fact that the boundaries of the "business" are not equivalent to the AS boundaries.

The other major alternative to obtaining information about the AS topology is to collect control plane data in the form of directly measured routing information. The primary example of such control plane data are BGP-derived measurements. BGP is a path-vector routing protocol, and as such each node transmits to its neighbors information about the *best* path that it knows to a destination. Each node then takes the information it has received about best paths, and computes its own best path, which it transmits to its neighbors. A route monitor receives this information as would any router, and from the transmitted path information, can infer links between ASes. The two best known projects that rely on BGP route monitors, Oregon RouteViews [118], and RIPE (Réseaux IP Européens)'s Routing Information Service [131] both use this approach, and each connects to a few dozen different ASes.

However, by its very design, BGP is an information-hiding rather than an information-revealing routing protocol. In addition, by its very design, BGP is all about reachability and **not** connectivity. Using it for mapping the Internet inter-domain topology is a "hack", and so it should come as no surprise that it has its own set of problems, including the following:

- The AS-path information in the announcements is primarily included for loop detection and does not have to correspond to reality. It is easy (and not uncommon) to insert additional ASes into a path for various purposes, *e.g.,* traffic engineering or measurement [21, 35], and moreover, the AS-path does not have to represent the data path.

- Path-vector protocols do not transmit information on every path in the network. For instance, backup paths may never appear in any routing announcements (unless there is a failure), and so may not be seen by a route monitor.

- Path-vector protocols only transmit "best" paths, and so there is a large loss of visibility from any one viewpoint. It is sometimes argued that a large number of viewpoints would alleviate this, but the viewpoint locations are highly biased towards larger networks, and this known "vantage point problem" severely biases the possible views of the network [134].

The BGP measurement data being provided by RIPE and RouteViews was originally intended to help debug networks, not for mapping. While this data collections have been invaluable for that intended original purpose, it is unsurprising that it is inadequate when used for a rather different purpose such as mapping the AS Internet. However, when this aspect is carefully taken into account, good work can be done but requires a critical evaluation of the data. Problems arise primarily when this data is used uncritically. Other useful sources of AS-level measurements such as looking glass servers and route registries suffer from similar problems [69, 96], and do so for similar reasons: they weren't intended to draw a map of the AS Internet.

4.2.2 Attribute discovery

The AS topology may be interesting to scientists in itself, but to be useful to network engineers, the routing policies that accompany it should also be known. It has been common to approximate the range of policies between ASes by a simple set of three relationships: (a) customer-provider, (b) peer-peer, and (c) siblings. This reduction was at least in part motivated by Huston [72, 73] and has been used in various places [146, 153, 159]. While many relationships fall into these three categories, there are frequent exceptions [75, 110, 126], for instance, in the form of partial transit in a particular region [115, 163].

Forgetting for the moment the simplification in assuming all policies fit this model and the simplifications the AS-graph itself makes, the relationships can be represented in the graph by providing simple labels for each edge. Typically, the next step after inferring network topology is to infer policies between ASes. The most common approach to this problem is to assume the universality of the peer-peer, customer-provider, sibling-sibling model, and to infer the policies by finding an allocation of policies consistent with the observed routing [14, 41, 56, 153, 159].

Once relationships are established, a seemingly reasonable next step is to estimate the hierarchical structure as in [146]. However, the effect of large numbers of (biased) missing links has not really been considered in these algorithms. In fact, the tier structure of the Internet seems to be largely an illusion. Recent work has shown that there is little value in the model at present [58, 88]; but, in contrast to the claims of these papers, there is no strong evidence that the situation has actually changed or that the tier model was ever a good model (except maybe in the early stage of the "public" Internet in the latter 20th century) particularly in light of the problems in the data.

Alternatively, we can infer a generic set of policies consistent with routing observations using a more detailed set of routing measurements [98, 110] and estimate performance by comparing predicted routes to real routes (held back from the inference process).

4.2.3   The "missing link" problem: Extent and impact

Perhaps the most obvious problem that results from relying on BGP measurement data to map the AS-level Internet is that there are many missing links in the resulting AS-graph. To illustrate the extent of this problem, years of concentrated research efforts that relied on a combination of improved inference methods and additional data sources [9, 27, 28, 39, 41, 69, 70, 110, 117, 134, 136, 166] have produced a picture of the Internet's AS topology that — as of 2011 — consisted of some 35,000-40,000 ASes (nodes) and about 115,000-135,000 edges (AS links), with about 80,000-90,000 of them being of the customer-provider type and 35,000-45,000 of the peer-peer type.

More recently, this supposedly up-to-date and most complete view of the AS-level Internet changed drastically thanks to [2] that relied on ground truth data from one of the largest IXPs in Europe (and worldwide) that had at the time of this study almost 400 member ASes. The main finding of this recent study is that in this single location, the number of actively used AS links of the peer-peer type was more the 50,000 — larger than the number of all AS links of the peer-peer type in the entire Internet known as of 2011. Moreover, being extremely conservative when extrapolating from this IXP to the Internet as a whole, [2] shows that there are easily more than 200,000 AS links of the peer-peer type in the entire Internet, more than twice the number of all AS links of the customer-provider type Internet-wide. Importantly, the main reason for this abundance of AS links of the peer-peer type at IXPs is well understood — many IXPs, especially the larger ones, offer as free service to their member ASes the use of their route server. This service greatly facilitates the establishment of peer-peer links between the members of an IXP and has become enormously popular with members that have an "open" (as compared to restrictive or selective) peering policy. Especially for the larger IXPs, such networks typically constitute the vast majority of IXP member ASes. Figure 16 provides an illustration of the connectivity through this IXP and shows that a majority of its member ASes have an open peering policy (some 300+ members) and establish AS links of the peer-peer type with one another.

In short, for many years, researchers have worked with AS-graphs that are typically complete in terms of nodes, but easily miss more than half the edges. Importantly, these graphs have generally a 2:1 ratio of customer-provider type vs. peer-peer type links when a 1:3 ratio is much more likely to reflect Internet reality. Clearly, for gaining any economic-based understanding of the AS Internet, getting
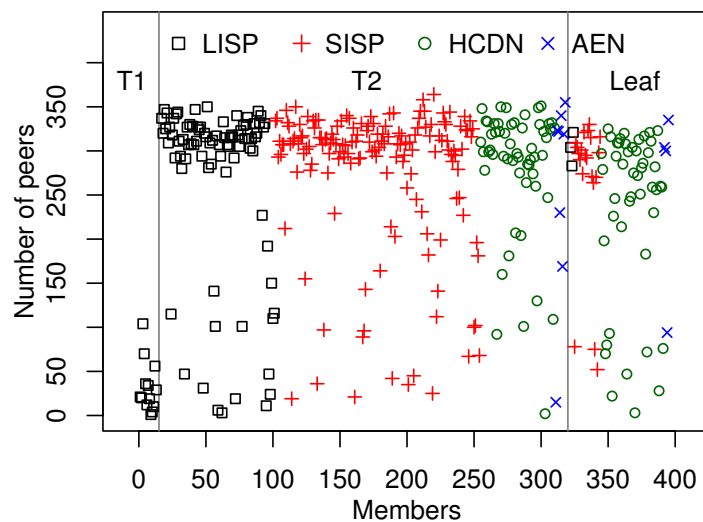
Figure 16: *Scatter-plot of number of peers per member, based on a classification of the member ASes in the four business categories defined above: LISP (Large ISP), SISP (Small ISP), HCDN (Hosting/service and Content Distribution Network)), and AEN (Academic and Enterprise Networks), and by tier. (Reprinted from [2]; ©2012 ACM, Inc. Included here by permission.)*

39

that ratio approximately correct is paramount because it is directly impacting how money flows in the Internet — while in a customer-provider relationship, the former pays the latter for bandwidth, peer-peer relationships are typically settlement-free (*i.e.,* no money is exchanged between the involved parties).

Besides their immediate economic impact, the above missing edges cause also significant problems in inferring the AS graph. For instance, it is a requirement that a network be multi-homed to obtain an ASN. This means the AS needs to intent to connect to at least two upstream providers. In this sense a "single-homed stub-AS" does not exist. Without any doubt, there are exceptions to this rule. However, the second link is often a backup link which is invisible to BGP outside of the immediate connection, because of BGP's information hiding[5]. Thus, it may appear as if a large number of ASes are single-homed stubs.

In [117], the authors separate the missing links into *hidden* and *invisible*. Whereas the latter are links that are missing from the data for structural reasons (*i.e.,* it is not just a question of quantity (*i.e.,* numbers of monitors) but quality (*i.e.,* location of monitor)), the *hidden* links may be found with enough measurements (over time, or multiple viewpoints). In [134] the authors extend that by dividing links into a number of classes based on their observability.

The "missing link" problem in the AS context is much more serious than if those links were "missing at random". In particular, the bias in the type of links that are missing [134] is critical when calculating some metrics on the graph, such as distances, precisely because such links are often *designed* to cut down on the number of ASes traffic must traverse. The missing data is also crucial for understanding reliability: for instance, papers such as [5] that argue that high-degree nodes create vulnerabilities in the Internet ignore the backup links that are invisible in these dataset, but obviously crucial when studying the resilience of the network.

4.3   The Internet's AS-level topologies

Despite the limitations of measurements, there is a considerable amount known about the AS-level topology of the Internet, and we talk here about the issues in defining and modelling that topology. We have seen that the definition of an AS is fraught with problems. Assuming for the time being that the concept of an AS is well defined so that it makes sense to equate each AS with a node in a graph, then what is the set of links? Unfortunately, the question of which ASes are "adjacent" also has no simple answer, and defining the meaning of a "link" between two ASes requires further consideration.

Does a link mean the ASes have a business relationship, physical connectivity, connecting BGP session, or that they share traffic? All the above are reasonable definitions, and none are equivalent. A common definition is that two ASes are said to be connected (at a particular time), if they can exchange routing data (and presumably IP traffic) without the help of an intermediary AS that provides *transit*. However, this says little about the true business relationships that are sometimes discussed as a matter of course when the AS-graph is considered. Moreover, this abstraction looses considerable information. In reality there are multiple topologies we want to model, each with its own meaning, structure, potential applications, and inference problems.

- *Business relationship graph:* in its simplest form this graph simply indicates (by an edge) that a business relationship exists between the corporations that own two ASNs. Edges could be usefully labelled by the type of business relationship, and we list a small subset of the possible relationships in Table 1.

- *Physical link-level graph:* this graph indicates whether two ASNs have a physical (layer 1) connection, and how many such connections they have. The multiple nature of such connections leads this

---

[5]Note that complex BGP policies may play a role in this as well [36, 63].

Graph	Edge Annotation	Graph Type
business relationship	subsidiary, partner, customer,...	directed graph
physical link-level	link capacity	multi- hyper-graph
connectivity graph	-	multigraph
BGP routing graph	-	undirected graph
policy graph	BGP policies	directed multigraph
traffic graph	traffic volumes	directed graph

Table 1: Example elements of the set of AS graphs.

to being a multigraph, as it is very common for two ASes to be connected by multiple links and in different geographic locations [94, 114, 143]. The idea is clearly illustrated by Figure 1 in [94], which shows a "pancake" diagram of the North American Internet backbone. Perhaps the reason this critical aspect of the topology is typically ignored is that it is very hard to measure—BGP monitor data is in general blind to this facet of the topology. In addition, this graph should really be a hypergraph. A single "edge" can connect multiple ASes, for example through an IXP [9, 75, 160]. One might argue that they are joined by a switch/router, each using point-to-point links, but in at least some cases, that switch has no place in a AS graph (*i.e.,* it has no ASN). The graph's edges could be usefully annotated with link capacity and potentially other features such as geographic location.

- *Connectivity graph:* this graph indicates that layer-2 connectivity exists between two ASNs. In many cases the layer-2 connectivity between ASNs would be congruent with the layer-1 connectivity, but with recent advances in network virtualization this may not hold for long [154].

- *BGP routing graph:* the edges in this graph indicate pairs of ASes that have an active BGP session exchanging routing information (*i.e.,* a BGP session that is in the 'established' state [130]).

- *Policy graph:* the edges in this graph are the same as those in the BGP routing graph, but include directed policy annotations [62]. We define this separately from the BGP routing graph because it may require a multigraph to allow for policy differences between different regions.

- *Traffic graph:* it is the same as the BGP routing graph, but the edges are annotated with the amount of traffic exchanged between the corresponding ASes.

This is hardly a complete set of possibilities, but already we can see the potential complexity here. Nevertheless, it appears unusual for studies to even define precisely what graph they examine (exceptions being papers such as [60, 117] where the BGP routing graph is explicitly considered). In Table 1, we list some of the possible graphs, and their basic properties. There is no clean 1:1 mapping between "network" and "organization" and "AS" [22, 75], and so it is highly non-trivial to map between these graphs, and they are certainly not equivalent.

4.4 A look ahead

Given our list of problems described here, one might be tempted to think that the AS-graph and routing data in general are useless until these datasets are drastically improved. However, apart from their operational utility, RouteViews and RIPE RIS have provided the essential ingredients for many important studies that match those services' goals [116]. A number of these studies have improved the Internet

significantly, and in the majority of such successful papers there is no need to exploit the "graph" view of the network. Examples include: (a) The discovery of slow convergence and persistence oscillation in routing protocols [64, 86, 87, 89, 90, 151, 152]. (b) Understanding of the impacts (positive and negative) of route flap dampening [97, 124]. (c) Determining how much address space and how many ASNs are being actively used [74]. (d) Looking for routing "Bogons" often related to Internet address hijacking [17, 40, 50, 128, 147]. (e) Debugging network problems [20, 53, 133].

On the measurement side, there have also been many advancements towards improving our view of AS topology. For instance:

1. As BGP routing changes, often multiple potential paths are explored and these paths (which are unlikely to actually be used as a final choice) can show some of the alternative routes available in the network [166], and thus a more complete topology.

2. Missing edges can be found using additional datasets, *e.g.,* RIRs and looking glasses [27, 69, 70, 166], or IXP data [9, 69, 70, 136], though care must be exercised with any additional dataset.

3. A routing beacon [21, 89, 99] is just a router that advertises and withdraws certain prefixes on a regular schedule. Examination of the observed announcements and withdrawals by various route monitors then allows estimates of protocol behavior such as convergence time.

4. Route poisoning prevents announcement from reaching certain parts of the Internet. As with beacons, it allows one to examine the behavior of BGP in a more controlled manner. This is perhaps the only way to see (some) backup paths, or to understand whether an ISP uses default routing [21, 35].

5. There are also attempts to not just estimate the topology but derive some quality measure for the resultant AS-graph [76, 134, 158].

There is often an unfortunate side-effect to some of these types of measurement in form of a Heisenberg-like uncertainty principle. That is, it is not clear whether observed changes are due to the micro-phenomenon of path exploration or macro-phenomena of link changes, new entrants, etc. The longer we make observations, the more complete they may seem, but we then do not know whether all of those links existed at the same time. Such uncertainty principles appear to be present in a number of Internet measurement contexts [132] where we trade off "accuracy" of the measurements against "time localization". In any case, this approach does not overcome the structural bias mentioned earlier.

At the same time, the above-mentioned and other advances on the measurement side suggest that the missing link problem may be improved, providing "more complete" AS graphs. However, there is a profound need (illustrated by the above) for better data accuracy measurements, and better response to data quality issues from subsequent users of the data. Obvious ways to improve are to conduct sensitivity analysis (of results) to missing or incorrect input data.

In addition, it is to be hoped that more controlled experiments are conducted (*i.e.,* experiments that have a "control" sample against which the experimental data can be compared) in order to precisely derive which factors of interest affect which variables. Controls allow one to discriminate alternative explanations for results, and prevent the affects of one confounding factor drowning out the affects of others (see [21, 99]). This is basic tenet of the scientific method, but seems to have been ignored in this area of research. Most studies have been "observational", and while there is a valid role for such experiments, for instance in epidemiology, they are intrinsically harder to interpret.

Lastly, another aspect of this richer set of AS topologies is that it should be obvious by now that economic or commercial objectives by and large determine and shape the structure and evolution of

their real-world counterparts, and that these constructs are once again naturally expressed through optimization rather than random graph models, though in this case the optimization problems may come from game theory or economics rather than mathematical programming.

4.5 Notes

The primary sources for the material presented in this section are

[135] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush. 10 Lessons from 10 Years of Measuring and Modeling the Internet's Autonomous Systems, in: *IEEE Journal on Selected Areas in Communications* 29(9):1810-1821, 2011.

[2] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger. Anatomy of a large European IXP, in: *Proc. ACM SIGCOMM'12, ACM Computer Communication Review* 42(4), 2012.

and they contain lengthier discussions of many of the issues touched upon here.

For additional and more in-depth reading materials (in addition to the references indicated throughout) we point to

[26] H. Chang. Modeling the Internet's Inter-Domain Topology and Traffic Demand Based on Internet Business Characterization, PhD Thesis, University of Michigan, 2006.

[43] B.Donnet and T. Friedman, Internet Topology Discovery: A Survey, IEEE Communications Surveys & Tutorials, 9(4), pp.56-69, 2007.

[38] A. Dhamdhere. Understanding the Evolution of the AS-level Internet Ecosystem, PhD Thesis, Georgia Institute of Technology, 2008.

[68] H. Haddadi, M. Rio, G. Iannaccone, A. Moore and R. Mortier, Network topologies: inference, modeling and generation, IEEE Communications Surveys, 10(2), 2008.

[39] A. Dhamdhere and C. Dovrolis. Twelve Years in the Evolution of the Internet Ecosystem, in: *IEEE/ACM Transactions on Networking* 19(5), 2011.

5 PoP-level topology

When designing or reconfiguring the physical infrastructure of an ISP, network operators are often guided by a design principle that emphasizes *hierarchy* [31, 59, 108]. There are two main reasons for implementing hierarchical network designs: *scalability* and *simplicity*. Compared to non-hierarchical designs, hierarchical networks can often be built at scale, mainly because hierarchy makes a network easier to visualize — a key feature towards making it easier to manage. The situation is analogous to modularity in programming languages — ideally it allows consideration of network components in isolation.

A common form of hierarchy in IP networks is based on the concept of the PoP (or Point of Presence). A PoP is a loosely defined term. Some providers may use the term to mean a physical building (housing a group of routers, switches and other devices), whereas others mean a metropolitan area where service is provided. However it is defined, though, it is a useful construct because it describes the logical structure of the network as the designer intended, rather than its particular implementation in terms of individual routers. Moreover, irrespective of the meaning, PoPs have an explicit geography (*e.g.,* street address or

city/metropolitan area). This then leads to our third major category of "Internet topology" — the PoP-level topology.

PoP-level topologies are ideal for understanding tradeoffs between connectivity and redundancy, and also provide the most essential information to competitors or customers (about where a network is based, or who has the best access network in a region). Additional reasons why the PoP-level view of networks is interesting include

- Network maps are often drawn at this level because it is an easy level for humans to comprehend.

- Network optimization is often conducted at this level because the problem size is generally reasonable (*e.g.,* dozens of PoPs as compared to potentially hundreds of routers) and because inter-PoP links are much more expensive than intra-PoP links.

- The internal design of PoPs is almost completely determined by simple templates [31, 59, 108].

- Networks change less frequently at the PoP level than at the router level [138]; and

- The PoP level is the more interesting level for many activities because it is less dependent on the details of protocol implementations, router vendor and model, and other technological details.
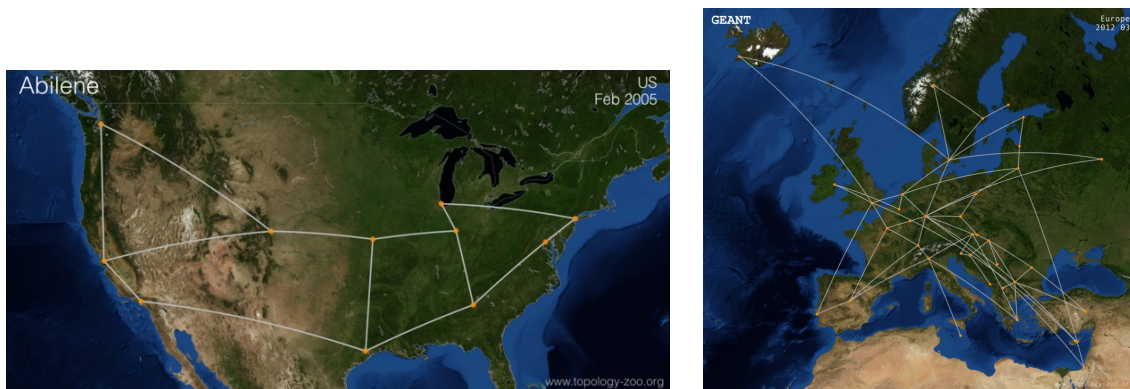
Figure 17: *PoP-level network topologies taken from* `www.topology-zoo.org`.

The last point is subtle but important for modelling. For instance, when using a network as part of a simulation, one would like to have a network that is *invariant* to the method being tested. If a network designer might change his/her network in response to a new protocol, say a routing or traffic engineering algorithm, then the test will be ambiguous if it uses existing networks as models. PoP-level networks are less sensitive to these details than router-level networks, because routers impose physical and technological constraints that are almost completely dependent on the details of the router vendor, model and even the version of software running on the router.

Two examples of PoP-level topologies are depicted in Figure 17, showing the structure of two of the largest research networks (Abilene, and GÉANT) in the world.

5.1 A look back

The study of PoP-level topologies has a briefer history than the major alternatives. Although the concept has existed for almost as long as networks, the work on modelling and measurement has typically focussed on routers (or their equivalent) though it is noteworthy that in simple networks (with one router per PoP) the two are the same.

The first steps were taken when real data-networks were observed and it was noted that they had structure in the form of hierarchy that was not well represented in simple random graphs. This observation led to the development of *structural topology generators* [165], based on the idea that a topology generator should reflect the obvious hierarchical structures visible in real networks (*e.g.,* the Georgia Tech Internetwork Topology Models (GT-ITM) [66] generator). However, this model was not specifically aimed at modelling the PoP-level. PoPs have been used in more advanced structural topology generators such as IGen [127]. IGen explicitly treats network design as an optimization, rather than following simple stochastic rules, in order to mimic the manner in which real networks are designed. IGen uses this not only for topology but also to create some of the other important details of the network, such as the link metrics or iBGP configurations.

The Rocketfuel project [144] sought to measure (using traceroute with all the problems described earlier) individual ISPs, and as part of this, presented data and maps at the PoP level. The idea was extended by the iPlane project [95], and by DIMES [139]. However, the flaws in traceroute make this data and the ensuing maps suspect from the start. However, [144] raised the bar with respect to validating the obtained PoP-level maps by trying to solicit feedback from the operators in charge of the mapped ISPs.

More recently, the concept of a PoP has been explicitly used to help guide measurement approaches, in the hope to overcome some of the limitations of traceroute [51, 138, 162]. However, in the absence of strong validation and given traceroute's many problems, it is likely that most of the known issues still exist in these approaches.

Alternatively, Knight *et al.* [84] have collected a set of over 200 maps published by ISPs themselves, and transcribed these into an open set of data available from `www.topology-zoo.org`. Much of this data is at the PoP-level, indicating the importance of this level of network representation to ISPs. For a similar but complementary effort, see [8].

5.2 Know your measurements

The problems in measuring PoPs are essentially the same as in any traceroute-based survey, though it is thought (or perhaps just hoped) that mapping IP addresses to PoPs is more straight-forward than mapping IPs to either routers or to ASes. To the best of our knowledge, no rigorous testing of this hypothesis has been conducted to date, but there are some indications (*e.g.,* see [84]) that the PoP-level maps provided by traceroute are no better than their router- or AS-level counter-parts.

The topology-zoo data [84], on the other hand, is provided by ISPs themselves and should in principle be much more accurate. However, this dataset must also be treated carefully (as should all data) because of potential transcription errors, or mistakes or approximations in the maps provided by the ISPs. While such errors are much less frequent than those encountered in measured networks, an added concern with respect to ISP-provided maps is stale data because there is little incentive to provide up-to-date maps.

As mentioned earlier, another important component of many PoP-level topologies is the geographic element. Such topologies are much easier to visualize geographically [83], and so a frequent interest is geolocation of the PoPs. While this is not a chapter on geolocation, it suffices to say that many research papers have been written on the problems of accurately mapping IP addresses to their geolocation (*e.g.,* see [125]). Moreover, while the routers and switches of a PoP are typically located in a single location,

city, or metropolitan area, the "eyeballs" (*i.e.,* end users) connected to the PoP will be spread over some area [129]. However, if the researcher is willing to diligently mine various data sources, there is hope of at least being able to geolocate PoPs as they house potentially hundreds or thousands of IP addresses and reside in locations with known physical addresses (*e.g.,* carrier hotels) [8].

5.3   The nature of the PoP-level graph

There is a common meme in network modelling that the design of a US ISP backbone network involves simply selecting the NFL cities, and then joining them up with a few lines on a map. While the real process of network design is rarely so trite, the picture above isn't entirely unfair.

Most notably, PoPs are usually selected based on commercial criteria (*e.g.,* the desirability and size of the potential customer base in an area). So network engineers get little choice over the locations that they are connecting. They could be the NFL cities in the US, or the larger cities of another country, or the capitals of countries on a continent, and so on. Once the PoP locations have been selected, they need to be connected in some redundant fashion to ensure some degree of robustness to node or link failures. Historically, connecting these PoPs may have been done in a mostly *ad hoc* manner; see for instance http://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/roberts_arpanet_large.gif.

However, since the burst of the Internet bubble (*circa* 2000), capital investment has become harder to obtain, and network operators and engineers had to justify such investment more carefully. At this point, network capacity started to be more carefully planned, not always using formal mathematical optimization, but certainly using traffic measurements to ensure less wasted capacity. Much of this planning and optimization is performed at the PoP-level simply because the router-level is much more complicated (in size, and complexity of constraints), and because inter- and intra-PoP link costs vary by a large margin.

We have discussed network design by constrained optimization extensively in §3 (see also references such as [93]), and so here we shall only consider the main differences for PoP-level design (apart from those already listed above).

Perhaps the most important difference is that the physical and engineering constraints on a router do not directly apply for a PoP. At least in theory, a PoP can use as many routers as needed to provide sufficient number of ports for any arbitrary node degree and sufficient throughput per port. Naturally, the constraints in this case will arise in the form of costs, and optimization-based formulations of the PoP-level network design problem will feature budget constraints to reflect this aspect. As budget constraints can vary greatly among different companies, when we look at actual PoP-level ISP backbone networks, we see a wide variety of designs ranging from the meshy designs with high-degree nodes only at the edge predicted by the HOT model, to hub and spoke like networks [84]. In fact, the sheer variety of network designs we observe in reality suggest that while some network operators seem to aim at optimizing performance (given some lenient cost constraints), others appear to be willing to sacrifice performance in order to keep costs low. Moreover, network operators in different countries can encounter very different link costs depending on the local geographic and commercial environment.

A critical but rarely discussed property of the PoP-level topology is that it provides the "glue" between the more physical router-level topology on the one hand and the more logical AS-level topology on the other hand. Functioning as an "intermediary" between these two topologies highlights the important aspect of the PoP-level topology that its granularity is "just right" for many networking problems of practical interest — not too coarse as to ignore important context (*e.g.,* the case with various AS topologies) but also not too fine as to be overwhelmed with unnecessary details (*e.g.,* the case with various physical

topologies). We next discuss this property in more detail.

5.3.1   From PoP-level to router-level connectivity

Given a PoP-level network, there is an additional interesting question: "Can we map this network down to the router level?" The GT-ITM model addressed this through random generation of its subnetworks, but in practice the design process of network engineers in this case is a text-book application of repeating patterns [31, 59, 108] and hence anything but random.

The main reason for following this design process is that network designers often apply "cookie cutter" methods to design networks as a whole or the internals of PoPs, though that term unnecessarily trivializes the importance of repeated patterns. Repetition makes network operations vastly simpler: the management of two PoPs requires the same skills. Equipment can be bulk purchased, debugging is easier, and adding new PoPs is simpler. Finally, networks based on templated design lead to simple design methodologies. For instance, the inter-PoP level network topology can be optimized relatively simply, as details such as redundancy will be supplied by the provision of pairs of redundant routers in each PoP, with redundant links between them. Design often refers to the graph topology of router interconnections, but templated design can extend to other details, such as physical configuration within racks, connections with external networks, or additional servers such as Domain Name Service or Network Management Systems.

This type of design can be mathematically described using graph-products, though for more details, we invite the reader to consult [121].

5.3.2   From PoP-level to AS-level connectivity: The pancake-view of the Internet

Until now we have only really discussed the PoP-level topology of a single network. However, there is considerable interest in how these networks interconnect.

The most prominent and commonly-accepted view of the Internet is as a "network of networks" or ASes in the AS-graph representation discussed in §4. A much neglected and rarely-mentioned representation is the "pancake view" where we consider each network to be a layer and where the different layers (networks) are stacked on top of one another to form a pancake-like structure [94]. To show where the different networks inter-connect, we add links across layers; intra-network connectivity is shown as links within each layer. For a set of "peer" networks, one advantage of this pancake view is that these networks often cover similar geographic areas and inter-connect in multiple locations, but at a limited set of cities (determined either by where private interconnects are seen as commercially viable, or where IXPs are available). Importantly, depending on the types of networks, many of them host their PoPs in one and the same commercial colocation facilities whose street addresses are generally known[6]. As such, the pancake view allows one to visualize not only this connectivity inside and between such providers but also the geography of their PoP-level topologies.

However, as far as we are aware, there has been no significant work studying this pancake view together with the different inter-connections, other than noting that it exists. The dearth of studies and models perhaps stems from the problems in obtaining the measurements necessary for constructing this view (see the discussions in §4), but it is perhaps one of the most interesting areas for future Internet topology research.

---

[6]One problem in establishing such a view lies in the limitations of current IP geolocation services [125].

47

5.4 A look ahead

We have focused in this section and the earlier sections mostly on traditional ISPs or network service providers which operate networks that have more or less pronounced backbones and cover geographic areas ranging from individual countries to entire geographic regions to the entire globe. However, there are many other networks that are not ISPs and consist of PoPs without their own physical infrastructures to connect them (*e.g.,* content providers, CDNs, Web-hosting companies). The PoPs of these networks are typically located in commercial colocation facilities or data centers that are operated by third-party companies for the explicit purpose of interconnecting such infrastructureless networks among one another or with ISPs or network service providers.

The importance of the role of such dedicated Internet infrastructure in the form of colocation facilities is best illustrated with a concrete example. As of December 2012, Equinix `www.equinix.com`, one of the leading companies in global interconnection and data centers, owned and operated in 14 countries in 5 continents some 30 colocation facilities. In these 30+ colocation facilities that are located in the major cities around the world, more than 4,000 networks connected directly to their customers and partners.

Another largely under-researched topic concerns the fact that in this chapter, we treated router-, PoP-, and AS-level topologies as static objects, when in reality, they evolve over time. In particular, it is rarely the case that a network operator designs a new network from scratch. Network design typically has to include as important aspects awareness and knowledge of the existing network that the operator intends to change due to, for example, drastic changes in the traffic demand that the original network design can no longer handle efficiently.

In view of these an other added challenges, the PoP-level view promises to be one of the more useful and interesting direction for future Internet topology research. In particular, measurements and models at this level have considerable scope for the future, and extensions of HOT-like optimization models may provide much more realistic synthetic networks than are currently available. At the same time, work on interconnection of networks at this level also provides considerable scope, but will require significant advances in our ability to measure and model the flow of traffic across the different networks as it is the traffic that ultimately determines much of the structure and evolution of the different topologies that we discussed in this chapter.

5.5 Notes

The primary source for the material presented in this section (and a much lengthier discussion of many of the issues) is

[84] S.Knight, H.Nguyen, N.Falkner, R.Bowden, M.Roughan, The Internet topology zoo. IEEE Journal on Selected Areas in Communications (JSAC) 29, 9, 1765-1775, October 2011.

For additional and more in-depth reading materials (in addition to the references indicated throughout) we point to

[139] Y.Shavitt and N.Zilberman, Geographical Internet PoP-level maps. In Proceedings of the 4th international conference on Traffic Monitoring and Analysis (Berlin, Heidelberg), TMA'12, Springer-Verlag, pp. 121-124, 2012.

[121] E.Parsonage, H.Nguyen, R.Bowden, K.Knight, N.Falkner, M.Roughan, Generalized graph products for network design and analysis. In 19th IEEE International Conference on Network Protocols (ICNP) (Vancouver, CA), October 2011.

[138] Y.Shavitt and N.Zilberman, A structural approach for PoP geo-location. In IEEE Infocom 2010.

[162] K.Yoshida, Y.Kikuchi, M.Yamamoto, Y.Fujii, K.Nagami, I.Nakagawa and H.Esaki, Inferring PoP-level ISP topology through end-to-end delay measurement. In PAM, pp. 35-44, 2009.

6 Conclusion

This chapter has aimed at clarifying the state of the art in Internet topology measurement and modelling, and correcting a number of clear and present flaws in reasoning. As we outlined in the introduction, we can see a number of themes recurring at multiple levels of hierarchy in topology modelling:

Theme 1: When studying highly-engineered systems such as the Internet, "details" in the form of protocols, architecture, functionality, and purpose matter.

Theme 2: When analyzing Internet measurements, examining the "hygiene" of the available measurements (*i.e.,* an in-depth recounting of the potential pitfalls associated with producing the measurements in question) is critical.

Theme 3: When validating proposed topology models, it is necessary to treat network modeling as an exercise in reverse-engineering and not as an exercise in model-fitting.

Theme 4: When modeling highly-engineered systems such as the Internet, beware of M.L. Mencken's quote "For every complex problem there is an answer that is clear, simple, and wrong."

We have not tried to survey the entire literature in this area, and we apologize to those whose work has not appeared here, but there are other extant surveys mentioned at the relevant points throughout this chapter, for specific components of the work. We also have not tried to critique every model, but rather tried to provide general guidance about modelling. It is intended that the readers could themselves critique existing and new models based on the ideas presented here.

In addition, we do not claim to have covered every type of topology associated with the Internet. Specifically, we have avoided topologies at the applications layer, for instance those associated with the WWW or online social networks. We made this choice simply because these topologies are (despite being "Internet" topologies) profoundly different from the topologies we have included. They are almost purely virtual whereas all of the networks considered here have a physical component, which leads to the arguments for optimization as their underlying construction. An important open problem in this context is the role that societal-related factors play over more economic- or technology-based drivers in the formation and evolution of these virtual topologies.

Finally, in each section, we have aimed at illuminating some of the current problems and identifying hopefully fruitful directions for future research in this area.

References

[1] ACHLIOPTAS, D., CLAUSET, A., KEMPE, D., AND MOORE, C. On the bias of traceroute sampling: or, power-law degree distributions in regular graphs. In *ACM Symposium on Theory of Computing (STOC)* (2005), ACM, pp. 694–703.

[2] AGER, B., CHATZIS, N., FELDMANN, A., SARRAR, N., UHLIG, S., AND WILLINGER, W. Anatomy of a large European IXP. In *ACM SIGCOMM* (2012).

49

[3] AIELLO, W., CHUNG, F., AND LU, L. A random graph model for massive graphs. In *ACM Symposium on Theory of Computing (STOC)* (2000), ACM, pp. 171–180.

[4] ALBERT, R., AND BARABÁSI, A.-L. Statistical mechanics of complex networks. *Reviews of Modern Physics 74* (2002), 47–97.

[5] ALBERT, R., H.JEONG, AND BARABÁSI, A.-L. Error and attack tolerance of complex networks. *Nature 406* (2000), 378–382.

[6] ALDERSON, D., AND DOYLE., J. Contrasting views of complexity and their implications for network-centric infrastructures. *IEEE Transactions on Systems, Man, and Cybernetics – Part A 40*, 4 (2010).

[7] ALDERSON, D., LI, L., WILLINGER, W., AND DOYLE, J. C. Understanding Internet topology: principles, models, and validation. *IEEE/ACM Transactions on Networking 13* (December 2005), 1205–1218.

[8] Internet Atlas. http://atlas.wail.wisc.edu/about-us.jsp.

[9] AUGUSTIN, B., KRISHNAMURTHY, B., AND WILLINGER, W. IXPs: Mapped? In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2009), pp. 336–349.

[10] BARABÁSI, A.-L. *Linked: How Everything Is Connected to Everything Else and What it Means for Business, Science, and Everyday Life.* Perseus Publishing, Cambridge, MA, 2002.

[11] BARABÁSI, A.-L. Scale-free networks: A decade and beyond. *Science 325* (2009), 412–413.

[12] BARABÁSI, A.-L. The network takeover. *Nature Physics 8* (2012), 14–16.

[13] BARABÁSI, A.-L., AND ALBERT, R. Emergence of scaling in random networks. *Science 286* (1999).

[14] BATTISTA, G., PATRIGNANI, M., AND PIZZONIA, M. Computing the types of the relationships between autonomous systems. In *IEEE INFOCOM* (2003).

[15] BENDER, A., SHERWOOD, R., AND SPRING, N. Fixing Ally's growing pains with velocity modeling. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2008).

[16] BONICA, R., GAN, D., TAPPAN, D., AND PIGNATARO, C. ICMP extensions for multiprotocol label switching. IETF, Network Working Group, Request for Comments: 4950, August 2007.

[17] BOOTHE, P., HIEBERT, J., AND BUSH, R. How prevalent is prefix hijacking on the Internet? *NANOG 36* (February 2006).

[18] BRODER, A., KUMAR, R., MAGHOUL, F., RAGHAVAN, P., RAJAGOPALAN, S., STATA, R., TOMKINS, A., AND WIENER, J. Graph structure in the web. *Computer Networks 33*, 1-6 (June 2000), 309–320.

[19] BROIDO, A., AND CLAFFY, K. Internet topology: connectivity of IP graphs. In *SPIE International Symposium on Convergence of IT and Communication* (August 2001), pp. 172–187.

[20] BUSH, R., HIEBERT, J., MAENNEL, O., ROUGHAN, M., AND UHLIG, S. Testing the reachability of (new) address space. In *ACM SIGCOMM workshop on Internet network management (INM'07)* (2007), pp. 236–241.

[21] BUSH, R., MAENNEL, O., ROUGHAN, M., AND UHLIG, S. Internet optometry: assessing the broken glasses in Internet reachability. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2009), pp. 242–253.

[22] CAI, X., HEIDEMANN, J., KRISHNAMURTHY, B., AND WILLINGER, W. Towards an AS-to-organization map. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2010).

[23] CALVERT, K., DOAR, M., AND ZEGURA, E. Modeling Internet topology. *IEEE Communications Magazine 35*, 6 (June 1997), 160–163.

[24] CARLSON, J., AND DOYLE, J. Complexity and robustness. *Proceedings of the National Academy of Sciences of the USA (PNAS) 99*, Suppl. 1 (2002), 2538–2545.

[25] CERF, V., AND KAHN, B. Selected ARPANET maps. *ACM SIGCOMM Computer Communications Review (CCR) 20* (1990), 81–110.

[26] CHANG, H. *Modeling Internet's Inter-Domain Topology and Traffic Demand Based on Internet Business Characterization.* PhD thesis, University of Michigan, 2006.

[27] CHANG, H., GOVINDAN, R., JAMIN, S., SHENKER, S. J., AND WILLINGER, W. Towards capturing representative AS-level Internet topologies. *Computer Networks 44*, 6 (2004), 737–755.

[28] CHEN, K., CHOFFNES, D., POTHARAJU, R., CHEN, Y., BUSTAMANTE, F., PAI, D., AND ZHAO, Y. Where the sidewalks ends: Extending the Internet AS graph using traceroutes from P2P users. In *ACM SIGCOMM CoNEXT* (2009).

[29] CHIANG, M. *Networked Life: 20 Questions and Answers.* Cambridge University Press, 2012.

[30] CHUNG, F., AND LU, L. The average distance in a random graph with given expected degrees. *Internet Mathematics 1*, 1 (2004), 91–113.

[31] ISP network design, Cisco Systems. ISOC ISP/IXP Workshop, 2005.

[32] Building cost effective and scalable CORE networks using an elastic architecture. Cisco white paper, 2013. http://www.cisco.com/en/US/prod/collateral/routers/ps5763/white_paper_c11-727983.pdf.

[33] Converged transport architecture: Improving scale and efficiency in service provider. Cisco white paper, 2013. http://www.cisco.com/en/US/prod/collateral/routers/ps5763/white_paper_c11-728242.pdf.

[34] CLARKE, D. The design principles of the DARPA Internet protocols. *ACM SIGCOMM Computer Communications Review (CCR) 25*, 1 (January 1995).

[35] COLITTI, L. *Internet Topology Discovery Using Active Probing.* PhD thesis, University di Roma Tre, 2006.

[36] BGP cost community, Cisco Systems, 2005. http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/s_bgpcc.html.

[37] DASU, T., AND JOHNSON, T. *Exploratory Data Mining and Data Cleaning.* Wiley, New York, 2003.

[38] DHAMDHERE, A. *Understanding the Evolution of the AS-level Internet Ecosystem.* PhD thesis, Georgia Institute of Technology, 2008.

[39] DHAMDHERE, A., AND DOVROLIS, C. Twelve years in the evolution of the Internet ecosystem. *IEEE/ACM Transactions on Networking 19*, 5 (2011), 1420–1433.

[40] DIAZ, J. GIZMODO: China's Internet hijacking uncovered, 2010. http://gizmodo.com/5692217/chinas-secret-internet-hijacking-uncovered.

[41] DIMITROPOULOS, X., KRIOUKOV, D., FOMENKOV, M., HUFFAKER, B., HYUN, Y., KC CLAFFY, AND RILEY, G. AS relationships: Inference and validation. *ACM SIGCOMM Computer Communications Review (CCR) 37*, 1 (2007), 29–40.

[42] DOAR, M. B., AND NEXION, A. A better model for generating test networks. In *IEEE GLOBECOM* (1996), pp. 86–93.

[43] DONNET, B., AND FRIEDMAN, T. Internet topology discovery: A survey. *IEEE Communications Surveys & Tutorials 9* (2007), 56–69.

[44] DONNET, B., LUCKIE, M., MÉRINDOL, P., AND PANSIOT, J.-J. Revealing MPLS tunnels obscured from traceroute. *ACM SIGCOMM Computer Communications Review (CCR)* (April 2012). http://www.sigcomm.org/ccr/papers/2012/April/2185376.2185388.

[45] DOROGOVTSEV, S., AND MENDES, J. *Evolution of Networks: From Biological Nets to the Internet and WWW.* Oxford University Press, 2003.

[46] DOYLE, J. C., ALDERSON, D. L., LI, L., LOW, S., ROUGHAN, M., SHALUNOV, S., TANAKA, R., AND WILLINGER, W. The "robust yet fragile" nature of the Internet. *Proceedings of the National Academy of Sciences of the USA (PNAS) 102*, 41 (October 2005), 14497–502.

[47] ERDÖS, P., AND RÉNYI, A. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences 5* (1960), 17–61.

[48] FABRIKANT, A., KOUTSOUPIAS, E., AND PAPADIMITRIOU, C. Heuristically optimized trade-offs: A new paradigm for power laws in the internet. In *Automata, Languages and Programming*, vol. 2380 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2002, pp. 781–781.

[49] FALOUTSOS, M., FALOUTSOS, P., AND FALOUTSOS, C. On power-law relationships of the Internet topology. In *ACM SIGCOMM* (1999), pp. 251–262.

[50] FEAMSTER, N., JUNG, J., AND BALAKRISHNAN, H. An empirical study of bogon route advertisements. *ACM SIGCOMM Computer Communications Review (CCR) 35*, 1 (2005), 63–70.

[51] FELDMAN, D., AND SHAVITT, Y. Automatic large scale generation of Internet PoP level maps. In *IEEE GLOBECOM* (2008), pp. 2426–2431.

[52] FELDMANN, A., GREENBERG, A., LUND, C., REINGOLD, N., AND REXFORD, J. Netscope: Traffic engineering for IP networks. *IEEE Network Magazine* (March/April 2000), 11–19.

[53] FELDMANN, A., MAENNEL, O., MAO, Z. M., BERGER, A., AND MAGGS, B. Locating Internet routing instabilities. In *ACM SIGCOMM* (2004).

[54] Fortz, B., and Thorup, M. Internet traffic engineering by optimizing OSPF weights. In *IEEE INFOCOM* (2000), pp. 519–528.

[55] Frazer, K. Merit's history, the NSFNET backbone project 1987-1995, Merit Network, Inc. http://www.livinginternet.com/doc/merit.edu/phenom.html.

[56] Gao, L. On Inferring Autonomous System Relationships in the Internet. *Global Telecommunications Internet Mini-Conference* (2000).

[57] GÉANT. http://www.geant.net/Network/The_Network/Pages/Network-Topology.aspx.

[58] Gill, P., Arlitt, M., Li, Z., and Mahanti, A. The flattening Internet topology: Natural evolution, unsightly barnacles or contrived collapse? In *Passive and Active Measurement Conference (PAM)* (2008), pp. 1–10. http://www.springerlink.com/content/1255p8g3k6766242/fulltext.pdf.

[59] Gill, V. Analysis of design decisions in a 10G backbone. www.nanog.org/meetings/nanog34/presentations/gill.pdf.

[60] Govindan, R., and Reddy, A. An analysis of Internet inter-domain topology and route stability. In *IEEE INFOCOM* (1997), pp. 850–857.

[61] Griffin, T. G. Understanding the Border Gateway Protocol (BGP). ICNP Tutorial, http://www.cl.cam.ac.uk/~tgg22/talks/, 2002.

[62] Griffin, T. G. The stratified shortest-paths problem (invited paper). In *International Conference on Communications Systems & Networks (COMSNETS)* (January 2010).

[63] Griffin, T. G., and Huston, G. BGP wedgies. RFC 4264, 2005.

[64] Griffin, T. G., and Wilfong, G. An analysis of the MED oscillation problem in BGP. In *IEEE International Conference on Network Protocols (ICNP)* (2002).

[65] GT-ITM: Georgia Tech Internetwork Topology Models. http://www.cc.gatech.edu/projects/gtitm/.

[66] Modeling topology of large internetworks, 2000. http://www.cc.gatech.edu/projects/gtitm/.

[67] Gunes, M., and Sarac, K. Resolving IP aliases in building traceroute-based Internet maps. *IEEE/ACM Transactions on Networking 17*, 6 (2009), 1738–1751.

[68] Haddadi, H., Rio, M., Iannaccone, G., Moore, A., and Mortier, R. Network topologies: inference, modeling and generation. *IEEE Communications Surveys 10*, 2 (2008).

[69] He, Y., Siganos, G., Faloutsos, M., and Krishnamurthy, S. V. A systematic framework for unearthing the missing links: Measurements and impact. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (April 2007).

[70] He, Y., Siganos, G., Faloutsos, M., and Krishnamurthy, S. V. Lord of the links: A framework for discovering missing links in the Internet topology. *IEEE/ACM Transactions on Networking 17*, 2 (2009), 391–404.

[71] HEART, F., MCKENZIE, A., MCQUILLIAN, J., AND WALDEN, D. ARPANET completion report. Tech. rep., Bolt, Beranek and Newman, Burlington, MA, 1978. http://www.cs.utexas.edu/users/chris/DIGITAL_ARCHIVE/ARPANET/DARPA4799.pdf.

[72] HUSTON, G. Peering and settlements - part I. *The Internet Protocol Journal 2*, 1 (March 1999).

[73] HUSTON, G. Peering and settlements - part II. *The Internet Protocol Journal 2*, 2 (June 1999).

[74] HUSTON, G. IPv4 address report, 2007. http://www.potaroo.net/tools/ipv4/index.html.

[75] HYUN, Y., BROIDO, A., AND K.C. CLAFFY. Traceroute and BGP AS path incongruities. Tech. rep., UCSD CAIDA, 2003. http://www.caida.org/publications/papers/2003/ASP/.

[76] Internet AS-level topology construction & analysis. http://topology.neclab.eu/.

[77] Internet 2. http://www.internet2.edu/pubs/networkmap-connectors-participants.pdf.

[78] JACOBSON, V. Traceroute. ftp://ftp.ee.lbl.gov/traceroute.tar.gz, 1989-04.

[79] JAMAKOVIC, A., AND UHLIG, S. On the relationships between topological metrics in real-world networks. *Networks and Heterogeneous Media 3*, 2 (June 2008), 345–359.

[80] Evolving backbone networks using with an MPLS supercore. Juniper Networks white paper, 2013. http://www.juniper.net/us/en/local/pdf/whitepapers/2000392-en.pdf.

[81] KARDES, H., OZ, T., AND GUNES, M. H. Cheleby: A subnet-level Internet topology mapping system. In *International Conference on Communications Systems & Networks (COMSNETS)* (2012).

[82] KELLER, E. F. Revisiting "scale-free" networks. *BioEssays 27*, 1 (2005), 1060–1068.

[83] KNIGHT, S., FALKNER, N., NGUYEN, H., TUNE, P., AND ROUGHAN, M. I can see for miles: Re-visualizing the Internet. *IEEE Network 26*, 6 (2012), 26–32.

[84] KNIGHT, S., NGUYEN, H., FALKNER, N., BOWDEN, R., AND ROUGHAN, M. The Internet topology zoo. *IEEE Journal on Selected Areas in Communications (JSAC) 29*, 9 (October 2011), 1765–1775.

[85] KRISHNAMURTHY, B., AND WILLINGER, W. What are our standards for validation of measurement-based networking research? *ACM SIGMETRICS Performance Evaluation Review 36* (2008), 64–69.

[86] LABOVITZ, C., AHUJA, A., BOSE, A., AND JAHANIAN, F. Delayed Internet routing convergence. In *ACM SIGCOMM* (2000).

[87] LABOVITZ, C., AHUJA, A., AND JAHANIAN, F. Experimental study of Internet stability and wide-area network failures. In *International Symposium on Fault-Tolerant Computing (FTCS)* (1999).

[88] LABOVITZ, C., IEKEL-JOHNSON, S., MCPHERSON, D., OBERHEIDE, J., AND JAHANIAN, F. Internet inter-domain traffic. In *ACM SIGCOMM* (2010), pp. 75–86.

[89] LABOVITZ, C., MALAN, R., AND JAHANIAN, F. Internet routing stability. In *ACM SIGCOMM* (1997).

[90] LABOVITZ, C., MALAN, R., AND JAHANIAN, F. Origins of Internet routing instability. In *IEEE INFOCOM* (1999).

[91] LAKHINA, A., BYERS, J., CROVELLA, M., AND XIE, P. Sampling biases in IP topology measurements. In *IEEE INFOCOM* (April 2003).

[92] LI, L., ALDERSON, D., DOYLE, J., AND WILLINGER, W. Towards a theory of scale-free graphs: Definitions, properties, and implications. *Internet Mathematics 2*, 4 (2005), 431–523.

[93] LI, L., ALDERSON, D., WILLINGER, W., AND DOYLE, J. A first-principles approach to understanding the Internet's router-level topology. In *ACM SIGCOMM* (2004), pp. 3–14.

[94] LILJENSTAM, M., LIU, J., AND NICOL, D. Development of an Internet backbone topology for large-scale network simulations. In *Winter Simulation Conference* (2003).

[95] MADHYASTHA, H. V., ISDAL, T., PIATEK, M., DIXON, C., ANDERSON, T., KRISHNAMURTHY, A., AND VENKATARAMANI, A. iPlane: An information plane for distributed services. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (November 2006).

[96] MAHADEVAN, P., KRIOUKOV, D., FOMENKOV, M., DIMITROPOULOS, X., CLAFFY, K. C., AND VAHDAT, A. The Internet AS-level topology: three data sources and one definitive metric. *ACM SIGCOMM Computer Communications Review (CCR) 36* (January 2006), 17–26.

[97] MAO, Z., GOVINDAN, R., VARGHESE, G., AND KATZ, R. Route flap dampening exacerbates Internet routing convergence. In *ACM SIGCOMM* (2002).

[98] MAO, Z., QIU, L., WANG, J., AND ZHANG, Y. On AS-level path inference. In *ACM SIGMETRICS* (2005).

[99] MAO, Z. M., BUSH, R., GRIFFIN, T. G., AND ROUGHAN, M. BGP beacons. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (October 2003).

[100] MAO, Z. M., REXFORD, J., WANG, J., AND KATZ, R. Towards an accurate AS-level traceroute tool. In *ACM SIGCOMM* (August 2003).

[101] MARCHETTA, P., DE DONATO, W., AND PESCAPE, A. Detecting third-party addresses in traceroute traces with IP timestamp option. In *Passive and Active Measurement Conference (PAM)* (2013).

[102] MARCHETTA, P., MERINDOL, P., DONNET, B., PESCAPE, A., AND PANSIOT, J. Quantifying and mitigating IGMP filtering in topology discovery. In *IEEE GLOBECOM* (2012), pp. 1871–1876.

[103] MEDINA, A., LAKHINA, A., MATTA, I., AND BYERS, J. BRITE: an approach to universal topology generation. In *Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (2001), pp. 346–353.

[104] MÉRINDOL, P., DONNET, B., BONAVENTURE, O., AND PANSIOT, J.-J. On the impact of layer-2 on node degree distribution. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2010), pp. 179–191.

[105] MÉRINDOL, P., VAN DEN SCHRIECK, V., DONNET, B., BONAVENTURE, O., AND PANSIOT, J.-J. Quantifying ASes multiconnectivity using multicast information. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2009), pp. 370–376.

[106] MITZENMACHER, M. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics 1* (2001), 226–251.

[107] MITZENMACHER, M. Editorial: The future of power law research. *Internet Mathematics 2*, 4 (2006), 525–534.

[108] MORRIS, M. Network design templates. `www.networkworld.com/community/blog/network-design-templates`, July 2007.

[109] The MRINFO and MERLIN project. `http://svnet.u-strasbg.fr/mrinfo/index.html`.

[110] MÜHLBAUER, W., FELDMANN, A., MAENNEL, O., ROUGHAN, M., AND UHLIG, S. Building an AS-topology model that captures route diversity. In *ACM SIGCOMM* (2006).

[111] MÜHLBAUER, W., UHLIG, S., FU, B., MEULLE, M., AND MAENNEL, O. In search for an appropriate granularity to model routing policies. In *ACM SIGCOMM* (2007).

[112] N. BERGER, C. BORGS, J. T. C., AND SABERI, A. On the spread of viruses on the Internet. In *16th ACM-SIAM Symposium on Discrete Algorithm (SODA)* (2005), pp. 301–310.

[113] NEWMAN, M. *Networks: An Introduction*. Oxford University Press, 2010.

[114] NORTON, W. B. A study of 28 peering policies. `http://drpeering.net/white-papers/Peering-Policies/A-Study-of-28-Peering-Policies.html`.

[115] NORTON, W. B. Internet service providers and peering, 2010. `http://drpeering.net/white-papers/Internet-Service-Providers-And-Peering.html`.

[116] OF THE ROUTING INFORMATION SERVICE, G. `http://www.ripe.net/ripe/docs/ripe-200`.

[117] OLIVEIRA, R., PEI, D., WILLINGER, W., ZHANG, B., AND ZHANG, L. The (in)completeness of the observed Internet AS-level structure. *IEEE/ACM Transactions on Networking 18*, 1 (2010), 109–122.

[118] University of Oregon Route Views Archive Project. `www.routeviews.org`.

[119] PANSIOT, J.-J., AND GRAD, D. On routes and multicast trees in the internet. *ACM SIGCOMM Computer Communications Review (CCR) 28*, 1 (1998), 41–50.

[120] PANSIOT, J.-J., MÉRINDOL, P., DONNET, B., AND BONAVENTURE, O. Extracting intra-domain topology from mrinfo probing. In *Passive and Active Measurement Conference (PAM)* (2010), pp. 81–90.

[121] PARSONAGEQUE, E., NGUYEN, H. X., BOWDEN, R., KNIGHT, S., FALKNER, N. J., AND ROUGHAN, M. Generalized graph products for network design and analysis. In *IEEE International Conference on Network Protocols (ICNP)* (October 2011).

[122] PASTOR-SATORRAS, R., AND VESPIGNANI, A. Epidemic spreading in scale-free networks. *Physical Review Letters 86*, 14 (2001), 3200–3203.

[123] PASTOR-SATORRAS, R., AND VESPIGNANI, A. *Evolution and Structure of the Internet: A Statistical Physics Approach*. Cambridge University Press, 2004.

[124] PELSSER, C., MAENNEL, O., MOHAPATRA, P., BUSH, R., AND PATEL, K. Route flap damping made useful. In *Passive and Active Measurement Conference (PAM)* (2011).

[125] POESE, I., UHLIG, S., KAAFAR, M. A., DONNET, B., AND GUEYE, B. IP geolocation databases: unreliable? *ACM SIGCOMM Computer Communications Review (CCR) 41*, 2 (April 2011), 53–56.

[126] QIU, S. Y., MCDANIEL, P. D., AND MONROSE, F. Toward valley-free inter-domain routing. In *IEEE International Conference on Communications* (2007).

[127] QUOITIN, B., VAN DEN SCHRIECK, V., FRANÃǦOIS, P., AND BONAVENTURE, O. IGen: generation of router-level internet topologies through network design heuristics. In *International Teletraffic Congress* (2009), pp. 1–8.

[128] RAMACHANDRAN, A., AND FEAMSTER, N. Understanding the network-level behavior of spammers. In *ACM SIGCOMM* (2006), pp. 291–302.

[129] RASTI, A. H., MAGHAREI, N., REJAIE, R., AND WILLINGER, W. Eyeball ASes: From geography to connectivity. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2010).

[130] REKHTER, Y., AND LI, T. A border gateway protocol (BGP-4). RFC 4271, January 2006.

[131] Ripe NCC: routing information service. http://www.ripe.net/projects/ris/.

[132] ROUGHAN, M. Fundamental bounds on the accuracy of network performance measurements. In *ACM SIGMETRICS* (June 2005), pp. 253–264.

[133] ROUGHAN, M., GRIFFIN, T., MAO, M., GREENBERG, A., AND FREEMAN, B. IP forwarding anomalies and improving their detection using multiple data sources. In *ACM SIGCOMM Workshop on Network Troubleshooting* (September 2004), pp. 307–312.

[134] ROUGHAN, M., TUKE, J., AND MAENNEL, O. Bigfoot, Sasquatch, the Yeti and other missing links: what we don't know about the AS graph. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (October 2008).

[135] ROUGHAN, M., WILLINGER, W., MAENNEL, O., PEROULI, D., AND BUSH, R. 10 lessons from 10 years of measuring and modeling the Internet's autonomous systems. *IEEE Journal on Selected Areas in Communications (JSAC) 29* (2011), 1810–1821.

[136] SANCHEZ, M., OTTO, J., BISCHOF, Z., CHOFFNES, D., BUSTAMANTE, F., KRISHNAMURTHY, B., AND WILLINGER, W. Dasu: Pushing experiments to the Internet's edge. In *10th USENIX NSDI* (2013).

[137] SHAIKH, A., AND GREENBERG, A. Experience in black-box OSPF measurement. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2001), pp. 113–125.

[138] SHAVITT, Y., AND ZILBERMAN, N. A structural approach for PoP geo-location. In *IEEE INFOCOM* (2010).

[139] SHAVITT, Y., AND ZILBERMAN, N. Geographical Internet PoP-level maps. In *4th international conference on Traffic Monitoring and Analysis* (2012), TMA'12, pp. 121–124.

[140] SHERRY, J., KATZ-BASSETT, E., PIMENOVA, M., MADHYASTHA, H., ANDERSON, T., AND KRISHNAMURTHY, A. Resolving IP aliases with prespecified timestamps. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2010).

[141] SHERWOOD, R., BENDER, A., AND SPRING, N. DisCarte: a disjunctive Internet cartographer. In *ACM SIGCOMM* (August 2008).

[142] SOMMERS, J., ERIKSSON, B., AND BARFORD, P. On the prevalence and characteristics of MPLS deployments in the open internet. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2011).

[143] SPRING, N., MAHAJAN, R., AND ANDERSON, T. Quantifying the causes of path inflation. In *ACM SIGCOMM* (2003).

[144] SPRING, N., MAHAJAN, R., AND WETHERALL, D. Measuring ISP topologies with Rocketfuel. In *ACM SIGCOMM* (August 2002).

[145] STROGATZ, S. Romanesque networks. *Nature 433* (2005).

[146] SUBRAMANIAN, L., AGARWAL, S., REXFORD, J., AND KATZ, R. Characterizing the Internet hierarchy from multiple vantage points. In *IEEE INFOCOM* (2002), vol. 2, pp. 618–627.

[147] The Team Cymru bogon reference page. http://www.cymru.com/Bogons/.

[148] TOZAL, M. E., AND SARAC, K. TraceNET: an Internet topology data collector. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2010), pp. 356–368.

[149] TOZAL, M. E., AND SARAC, K. Estimating network layer subnet characteristics via statistical sampling. In *11th international IFIP TC 6 conference on Networking – Volume Part I* (2012), IFIP'12, pp. 274–288.

[150] TUNE, P., AND ROUGHAN, M. *SIGCOMM eBook on Recent Advances in Networking*, vol. 1. ACM, 2013, ch. Internet Traffic Matrices: A Primer.

[151] VARADHAN, K., GOVINDAN, R., AND ESTRIN, D. Persistent route oscillations in inter-domain routing. Tech. rep., 96-631, USC/ISI, 1996.

[152] VARADHAN, K., GOVINDAN, R., AND ESTRIN, D. Persistent route oscillations in inter-domain routing. *Computer Networks* (March 2000).

[153] WANG, F., AND GAO, L. On inferring and characterizing Internet routing policies. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (October 2003).

[154] WANG, Y., KELLER, E., BISKEBORN, B., VAN DER MERWE, J., AND REXFORD, J. Virtual routers on the move: live router migration as a network-management primitive. In *ACM SIGCOMM* (2008), pp. 231–242.

[155] WAXMAN, B. M. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications (JSAC) 6*, 9 (December 1988), 1617–1622.

[156] WILLINGER, W., ALDERSON, D., AND DOYLE, J. Mathematics and the Internet: A source of enormous confusion and great potential. *Notices of the AMS 56*, 5 (2009), 586–599. http://www.ams.org/notices/200905/rtx090500586p.pdf.

[157] WILLINGER, W., ALDERSON, D., DOYLE, J. C., AND LI, L. More "normal" than normal: scaling distributions and complex systems. In *36th cWinter simulation Conference* (2004), pp. 130–141.

[158] WINTER, R. Modeling the Internet routing topology with a known degree of accuracy – in less than 24h. In *ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS)* (2009).

[159] XIA, J., AND GAO, L. On the evaluation of AS relationship inferences. In *Globecom* (2004).

[160] XU, K., DUAN, Z., ZHANG, Z.-L., AND CHANDRASHEKAR, J. On properties of Internet exchange points and their impact on AS topology and relationship. *LNCS 3042* (2004), 284–295. http://www.springerlink.com/content/umu88q1eewk0e8yy/fulltext.pdf.

[161] YOOK, S.-H., H.JEONG, AND BARABÁSI, A.-L. Modeling the Internet's large-scale topology. *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 99 (2002), 13382–13386.

[162] YOSHIDA, K., KIKUCHI, Y., YAMAMOTO, M., FUJII, Y., NAGAMI, K., NAKAGAWA, I., AND ESAKI, H. Inferring PoP-level ISP topology through end-to-end delay measurement. In *Passive and Active Measurement Conference (PAM)* (2009), pp. 35–44.

[163] YOSHINOBU, M. What makes our policy messy, 2010. http://www.attn.jp/maz/p/c/bgpworkshop200904/bgpworkshop-policy.pdf.

[164] ZEGURA, E., CALVERT, K., AND BHATTACHARJEE, S. How to model an internetwork. In *IEEE INFOCOM* (1996), vol. 2, pp. 594–602.

[165] ZEGURA, E. W., CALVERT, K. L., AND DONAHOO, M. J. A quantitative comparison of graph-based models for Internet topology. *IEEE/ACM Transactions on Networking 5* (December 1997), 770–783.

[166] ZHANG, B., LIU, R., MASSEY, D., AND ZHANG, L. Collecting the Internet AS-level topology. *ACM SIGCOMM Computer Communications Review (CCR)* (January 2005).

[167] ZHAO, X., PEI, D., WANG, L., MASSEY, D., MANKIN, A., WU, S. F., AND ZHANG, L. An analysis of BGP multiple origin AS (MOAS) conflicts. In *ACM SIGCOMM Internet Measurement Workshop (IMW)* (2001), pp. 31–35.

Recent Advances in Reliable Transport Protocols*

Costin Raiciu, Janardhan Iyengar, Olivier Bonaventure

Abstract

Transport protocols play a critical role in today's Internet. This chapter first looks at the recent of the reliable transport protocols. It then explains the growing impact of middleboxes on the evolvability of these protocols. Two recent protocol extensions, Multipath TCP and Minion, which were both designed to extend the current Transport Layer in the Internet are then described.

1 Introduction

The first computer networks often used ad-hoc and proprietary protocols to interconnect different hosts. During the 1970s and 1980s, the architecture of many of these networks evolved towards a layered architecture. The two most popular ones are the seven-layer OSI reference model [119] and the five-layer Internet architecture [27]. In these architectures, the transport layer plays a key role. It enables applications to reliably exchange data. A transport protocol can be characterized by the service that it provides to the upper layer (usually the application). Several transport services have been defined:

- a connectionless service

- a connection-oriented bytestream service

- a connection-oriented message-oriented service

- a message-oriented request-response service

- an unreliable delivery service for multimedia applications

The connectionless service is the simplest service that can be provided by a transport layer protocol. The User Datagram Protocol (UDP) [87] is an example of a protocol that provides this service.

Over the years, the connection-oriented bytestream service has proven to be the transport layer service used by most applications. This service is currently provided by the Transmission Control Protocol (TCP) [89] in the Internet. TCP is the dominant transport protocol in today's Internet, but other protocols have provided similar services [60].

Several transport protocols have been designed to support multimedia applications. The Real-Time Transport protocol (RTP) [101], provides many features required by multimedia applications. Some of the functions provided by RTP are part of the transport layer while others correspond to the presentation layer of the OSI reference model. The Datagram Congestion Control Protocol (DCCP) [67] is another protocol that provides functions suitable for applications that do not require a fully reliable service.

---

*Parts of the text in this chapter have appeared in the following publications by the same author(s): [17], [117], [94], [78] and [61].

The rest of this chapter is organized as follows. We first describe the main services provided by the transport layer in section 2. This will enable us to look back at the evolution of the reliable Internet transport protocols during the past decades. Section 3 then describes the organization of today's Internet, the important role played by various types of middleboxes, and the constraints that these middleboxes impose of the evolution of transport protocols. Finally, we describe the design of two recent TCP extensions, both of which evolve the transport layer of the Internet while remaining backward compatible with middleboxes. Multipath TCP, described in section 4, enables transmission of data segments within a transport connection over multiple network paths. Minion, described in section 5, extends TCP and SSL/TLS [35] to provide richer services to the application—unordered message delivery and multi-streaming—without changing the protocols' wire-format.

2 Providing the transport service

As explained earlier, several services have been defined in the transport layer. In this section, we first review the connectionless service. Then we present in more detail how TCP and SCTP provide a connection-oriented service and highlight the recent evolution of the key functions of these protocols. This section concludes with a discussion of the request-response service.

2.1 Providing the connectionless service

To provide a connectionless service, the transport layer mainly needs to provide some multiplexing on top of the underlying network layer. In UDP, this multiplexing is achieved by using port numbers. The 8 byte UDP header contains the source and destination port numbers that identify the applications that exchange data on the two communicating hosts. In addition to these port numbers, the UDP header contains a checksum that optionally covers the payload, the UDP header and a part of the IP header. When UDP is used above IPv4, the checksum is optional [88]. The sending application decides whether the UDP payload will be protected by a checksum. If not, the checksum field is set to zero. When UDP is used above IPv6, the checksum is mandatory and cannot be disabled by the sender.

UDP has barely changed since the publication of [88]. The only significant modification has been the UDP-lite protocol [70]. UDP-lite was designed for applications that could benefit from the delivery of possibly corrupted data. For this, UDP-lite allows the application to specify which part of the payload must be covered by a checksum. The UDP-lite header includes a checksum coverage field that indicates the part of the payload that is covered by the checksum.

2.2 Providing the connection-oriented service

The connection-oriented service is both more complex and also more frequently used. TCP and SCTP are examples of current Internet protocols that provide this service. Older protocols like TP4 or XTP [108] also provide a connection-oriented service.

The connection-oriented service can be divided in three phases :

- the establishment of the connection

- the data transfer

- the release of the connection

61

2.2.1 Connection establishment

The first objective of the transport layer is to multiplex connections initiated by different applications. This requires the ability to unambiguously identify different connections on the same host. TCP uses four fields that are present in the IP and TCP headers to uniquely identify a connection:

- the source IP address

- the destination IP address

- the source port

- the destination port

The source and destination addresses are the network layer addresses (e.g. IPv4 or IPv6 in the case of TCP) that have been allocated to the communicating hosts. When a connection is established by a client, the destination port is usually a well-known port number that is bound to the server application. On the other hand, the source port is often chosen randomly by the client [69]. This random selection of the source port by the client has some security implications as discussed in [5]. Since a TCP connection is identified unambiguously by using this four-tuple, a client can establish multiple connections to the same server by using different source ports on each of these connections.

The classical way of establishing a connection between two transport entities is the three-way handshake which is used by TCP [89]. This three handshake was mainly designed to deal with host crashes. It assumes that the underlying network is able to guarantee that a packet will never remain inside the network for longer than the Maximum Segment Lifetime (MSL)[1]. Furthermore, a host should not immediately reuse the same port number for subsequent connections to the same host. The TCP header contains flags that specify the role of each segment. For example, the ACK flag indicates that the segment contains a valid acknowledgment number while the SYN flag is used during the three-way handshake. To establish a connection, the original TCP specification [89] required the client to send a TCP segment with the SYN flag sent, including an initial sequence number extracted from a clock. According to [89], this clock had to be implemented by using a 32-bit counter incremented at least once every 4 microseconds and after each TCP connection establishment attempt. While this solution was sufficient to deal with random host crashes, it was not acceptable from a security viewpoint [48]. When a clock is used to generate the initial sequence number for each TCP connection, an attacker that wishes to inject segments inside an established connection could easily guess the sequence number to be used. To solve this problem, modern TCP implementations generate a random initial sequence number [48].

An example of the TCP three-way handshake is presented in figure 1. The client sends a segment with the SYN flag set (also called a SYN segment). The server replies with a segment that has both the SYN and the ACK flags set (a SYN+ACK segment). This SYN+ACK segment contains contains a random sequence number chosen by the server and its acknowledgment number is set to the value of the initial sequence number chosen by the client incremented by one[2]. The client replies to the SYN+ACK segment with an ACK segment that acknowledges the received SYN+ACK segment. This concludes the three-way handshake and the TCP connection is established.

The duration of the three-way handshake is important for applications that exchange small amount of data such as requests for small web objects. It can become longer if losses occur because TCP can only

---

[1]The default MSL duration is 2 minutes [89].

[2]TCP's acknowledgment number always contains the next expected sequence number and the SYN flag consumes one sequence number.

SYN,seq=1234, Option

SYN+ACK,ack=1235,seq=5678, Option
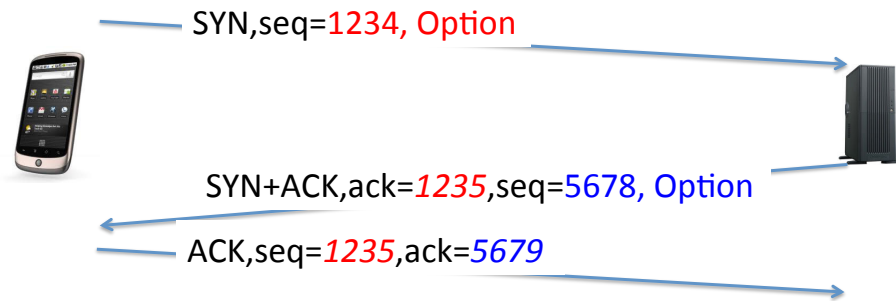
ACK,seq=1235,ack=5679

Figure 1: TCP three-way handshake

rely on its retransmission timer to recover from the loss of a `SYN` or `SYN+ACK` segment. When a client sends a `SYN` segment to a server, it can only rely on the initial value of its retransmission timer to recover from losses[3]. Most TCP/IP stacks have used an initial retransmission timer set to 3 seconds [18]. This conservative value was chosen in the 1980s and confirmed in the early 2000s [81]. However, this default implies that with many TCP/IP stacks, the loss of any of the first two segments of a three-way handshake will cause a delay of 3 seconds on a connection that may normally be shorter than that. Measurements conducted on large web farms showed that this initial timer had a severe impact on the performance perceived by the end users [24]. This convinced the IETF to decrease the recommended initial value for the retransmission timer to 1 second [82]. Some researchers proposed to decrease even more the value of the retransmission timer, notably in datacenter environments [111].

Another utilization of the three-way handshake is to negotiate options that are applicable for this connection. TCP was designed to be extensible. Although it does not carry a version field, in contrast with IP for example, TCP supports the utilization of options to both negotiate parameters and extend the protocol. TCP options in the `SYN` segment allow to negotiate the utilization of a particular TCP extension. To enable a particular extension, the client places the corresponding option inside the `SYN` segment. If the server replies with a similar option in the `SYN+ACK` segment, the extension is enabled. Otherwise, the extension is disabled on this particular connection. This is illustrated in figure 1.

Each TCP option is encoded by using a Type-Length-Value (TLV) format, which enables a receiver to silently discard the options that it does not understand. Unfortunately, there is a limit to the maximum number of TCP options that can be placed inside the TCP header. This limit comes from the `Data Offset` field of the TCP header that indicates the position of the first byte of the payload measured as an integer number of four bytes word starting from the beginning of the TCP header. Since this field is encoded in four bits, the TCP header cannot be longer than 60 bytes, including all options. This size was considered to be large enough by the designers of the TCP protocol, but is becoming a severe limitation to the extensibility of TCP.

A last point to note about the three-way handshake is that the first TCP implementations created state upon reception of a `SYN` segment. Many of these implementations also used a small queue to store the TCP connections that had received a `SYN` segment but not yet the third `ACK`. For a normal TCP connection, the

---

[3]If the client has sent packets earlier to the same server, it might have stored some information from the previous connection [109, 11] and use this information to bootstrap its initial timer. Recent Linux TCP/IP stacks preserve some state variables between connections.

delay between the reception of a `SYN` segment and the reception of the third `ACK` is equivalent to a round-trip-time, usually much less than a second. For this reason, most early TCP/IP implementations chose a small fixed size for this queue. Once the queue was full, these implementations dropped all incoming `SYN` segments. This fixed-sized queue was exploited by attackers to cause denial of service attacks. They sent a stream of spoofed `SYN` segment[4] to a server. Once the queue was full, the server stopped accepting `SYN` segments from legitimate clients [37]. To solve this problem, recent TCP/IP stacks try to avoid maintaining state upon reception of a `SYN` segment. This solution is often called *syn cookies*.

The principles behind *syn cookies* are simple. To accept a TCP connection without maintaining state upon reception of the `SYN` segment, the server must be able to check the validity of the third `ACK` by using only the information stored inside this `ACK`. A simple way to do this is to compute the initial sequence number used by the server from a hash that includes the source and destination addresses and ports and some random secret known only by the server. The low order bits of this hash are then sent as the initial sequence number of the returned `SYN+ACK` segment. When the third `ACK` comes back, the server can check the validity of the acknowledgment number by recomputing its initial sequence number by using the same hash [37]. Recent TCP/IP stacks use more complex techniques to deal notably with the options that are placed inside the `SYN` and need to be recovered from the information contained in the third `ACK` that usually does not contain any option.

At this stage, it is interesting to look at the connection establishment scheme used by the SCTP protocol [105]. SCTP was designed more than two decades after TCP and thus has benefited from several of the lessons learned from the experience with TCP. A first difference between TCP and SCTP are the segments that these protocols use. The SCTP header format is both simpler and more extensible than the TCP header.

The first four fields of the SCTP header (Source and Destination ports, Verification tag and Checksum) are present in all SCTP segments. The source and destination ports play the same role as in TCP. The verification tag is a random number chosen when the SCTP connection is created and placed in all subsequent segments. This verification tag is used to prevent some forms of packet spoofing attacks [105]. This is an improvement compared to TCP where the validation of a received segment must be performed by checking the sequence numbers, acknowledgment numbers and other fields of the header [47]. The SCTP checksum is a 32 bits CRC that provides stronger error detection properties than the Internet checksum used by TCP [107]. Each SCTP segment can contain a variable number of chunks and there is no apriori limit to the number of chunks that appear inside a segment, except that a segment should not be longer than the maximum packet length of the underlying network layer.

The SCTP connection establishment uses several of these chunks to specify the values of some parameters that are exchanged. A detailed discussion of all these chunks is outside the scope of this document and may be found in [105]. The SCTP four-way handshake uses four segments as shown in figure 2. The first segment contains the `INIT` chunk. To establish an SCTP connection with a server, the client first creates some local state for this connection. The most important parameter of the `INIT` chunk is the *Initiation tag*. This value is a random number that is used to identify the connection on the client host for its entire lifetime. This *Initiation tag* is placed as the *Verification tag* in all segments sent by the server. This is an important change compared to TCP where only the source and destination ports are used to identify a given connection. The `INIT` chunk may also contain the other addresses owned by the client. The server responds by sending an `INIT-ACK` chunk. This chunk also contains an *Initiation tag* chosen by the server and a copy of the *Initiation tag* chosen by the client. The `INIT` and `INIT-ACK` chunks also contain an initial sequence number. A key difference between TCP's three-way handshake and SCTP's four-way hand-

---

[4]An IP packet is said to be spoofed if it contains a source address which is different from the IP address of the sending host. Several techniques can be used by network operators to prevent such attacks [41], but measurements show that they are not always deployed [14].
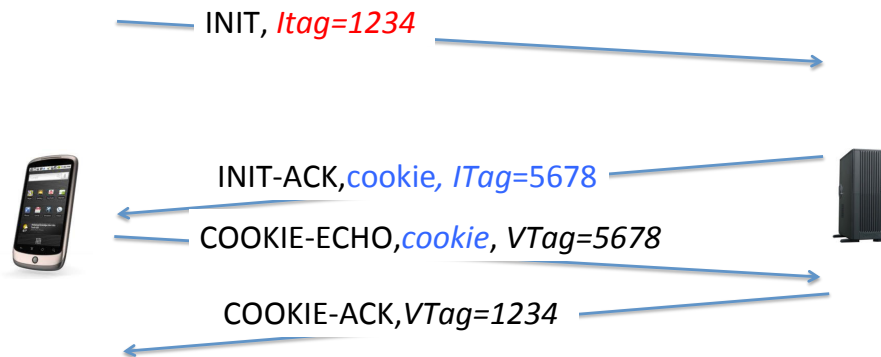
Figure 2: The four-way handshake used by SCTP

shake is that an SCTP server does not create any state when receiving an `INIT` chunk. For this, the server places inside the `INIT-ACK` reply a *State cookie* chunk. This *State cookie* is an opaque block of data that contains information from the `INIT` and `INIT-ACK` chunks that the server would have had stored locally, some lifetime information and a signature. The format of the *State cookie* is flexible and the server could in theory place almost any information inside this chunk. The only requirement is that the *State cookie* must be echoed back by the client to confirm the establishment of the connection. Upon reception of the `COOKIE-ECHO` chunk, the server verifies the signature of the *State cookie*. The client may provide some user data and an initial sequence number inside the `COOKIE-ECHO` chunk. The server then responds with a `COOKIE-ACK` chunk that acknowledges the `COOKIE-ECHO` chunk. The SCTP connection between the client and the server is now established. This four-way handshake is both more secure and more flexible than the three-way handshake used by TCP.

2.2.2 Data transfer

Before looking at the techniques that are used by transport protocols to transfer data, it is useful to look at their service models. TCP has the simplest service model. Once a TCP connection has been established, two bytestreams are available. The first bytestream allows the client to send data to the server and the second bytestream provides data transfer in the opposite direction. TCP guarantees the reliable delivery of the data during the lifetime of the TCP connection provided that it is gracefully released.

SCTP provides a slightly different service model [79]. Once an SCTP connection has been established, the communicating hosts can access two or more message streams. A message stream is a stream of variable length messages. Each message is composed of an integer number of bytes. The connection-oriented service provided by SCTP preserves the message boundaries. This implies that if an application sends a message of N bytes, the receiving application will also receive it as a single message of N bytes. This is in contrast with TCP that only supports a bytestream. Furthermore, SCTP allows the applications to use multiple streams to exchange data. The number of streams that are supported on a given connection is negotiated during connection establishment. When multiple streams have been negotiated, each application can send data over any of these streams and SCTP will deliver the data from the different streams independently without any head-of-line blocking.

While most usages of SCTP may assume an in-order delivery of the data, SCTP supports unordered

65

delivery of messages at the receiver. Another extension to SCTP [106] supports partially-reliable delivery. With this extension, an SCTP sender can be instructed to "expire" data based on one of several events, such as a timeout, the sender can signal the SCTP receiver to move on without waiting for the "expired" data. This partially reliable service could be useful to provide timed delivery for example. With this service, there is an upper limit on the time required to deliver a message to the receiver. If the transport layer cannot deliver the data within the specified delay, the data is discarded by the sender without causing any stall in the stream.

To provide a reliable delivery of the data, transport protocols rely on various mechanisms that have been well studied and discussed in the literature : sequence numbers, acknowledgments, windows, checksums and retransmission techniques. A detailed explanation of these techniques may be found in standard textbooks [16, 99, 40]. We assume that the reader is familiar with them and discuss only some recent changes.

TCP tries to pack as much data as possible inside each segment [89]. Recent TCP stacks combine this technique with Path MTU discovery to detect the MTU to be used over a given path [73]. SCTP uses a more complex but also more flexible strategy to build its segments. It also relies on Path MTU Discovery to detect the MTU on each path. SCTP then places various chunks inside each segment. The control chunks, that are required for the correct operation of the protocol, are placed first. Data chunks are then added. SCTP can split a message in several chunks before transmission and also the bundling of different data chunks inside the same segment.

Acknowledgments allow the receiver to inform the sender of the correct reception of data. TCP initially relied exclusively on cumulative acknowledgments. Each TCP segment contains an acknowledgment number that indicates the next sequence number that is expected by the receiver. Selective acknowledgments were added later as an extension to TCP [74]. A selective acknowledgment can be sent by a receiver when there are gaps in the received data. A selective acknowledgment is simply a sequence of pairs of sequence numbers, each pair indicating the beginning and the end of a received block of data. SCTP also supports cumulative and selective acknowledgments. Selective acknowledgments are an integral part of SCTP and not an extension which is negotiated at the beginning of the connection. In SCTP, selective acknowledgments are encoded as a control chunk that may be placed inside any segment. In TCP, selective acknowledgments are encoded as TCP options. Unfortunately, given the utilization of the TCP options (notably the timestamp option [63]) and the limited space for options inside a TCP segment, a TCP segment cannot report more than three blocks of data. This adds some complexity to the handling and utilization of selective acknowledgments by TCP.

Current TCP and SCTP stacks try to detect segment losses as quickly as possible. For this, they implement various heuristics that allow to retransmit a segment once several duplicate acknowledgments have been received [40]. Selective acknowledgment also aid to improve the retransmission heuristics. If these heuristics fail, both protocols rely on a retransmission timer whose value is fixed in function of the round-trip time measured over the connection [82].

Last but not least, the transport protocols on the Internet perform congestion control. The original TCP congestion control scheme was proposed in [62]. Since then, it has evolved and various congestion control schemes have been proposed. Although the IETF recommends a single congestion control scheme [8], recent TCP stacks support different congestion control schemes and some allow the user to select the preferred one. A detailed discussion of the TCP congestion control schemes may be found in [3]. SCTP's congestion control scheme is largely similar to TCP's congestion control scheme.

Additional details about recent advances in SCTP may be found in [21]. [36] lists recent IETF documents that are relevant for TCP. [40] contains a detailed explanation of some of the recent changes to the TCP/IP protocol stack.
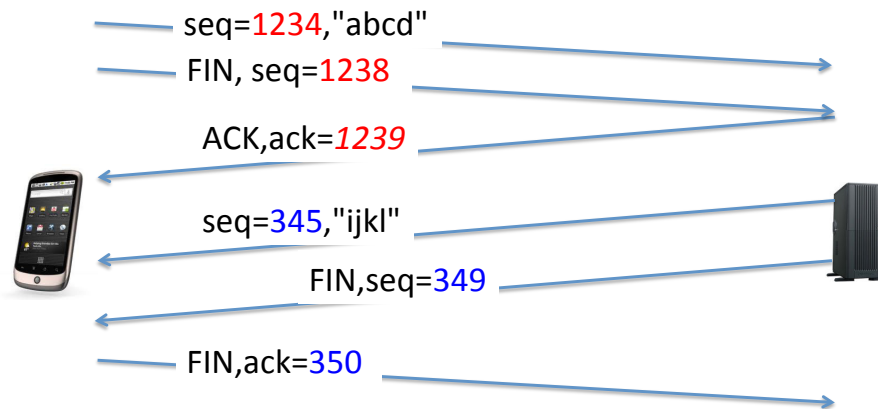
Figure 3: The four-way handshake used to close a TCP connection

2.2.3 Connection release

This phase occurs when either both hosts have exchanged all the required data or when one host needs to stop the connection for any reason (application request, lack of resources, ...). TCP supports two mechanisms to release a connection. The main one is the four-way handshake. This handshake uses the `FIN` flag in the TCP header. Each host can release its own direction of data transfer. When the application wishes to gracefully close a connection, it requests the TCP entity to send a `FIN` segment. This segment marks the end of the data transfer in the outgoing direction and the sequence number that corresponds to the `FIN` flag (which consumes one sequence number) is the last one to be sent over this connection. The outgoing stream is closed as soon as the sequence number corresponding to the `FIN` flag is acknowledged. The remote TCP entity can use the same technique to close the other direction [89]. This graceful connection release has one advantage and one drawback. On the positive side, TCP provides a reliable delivery of all the data provided that the connection is gracefully closed. On the negative side, the utilization of the graceful release forces the TCP entity that sent the last segment on a given connection to maintain state for some time. On busy servers, a large number of connections can remain for a long time [39]. To avoid maintaining such state after a connection has been closed, web servers and some browsers send a `RST` segment to abruptly close TCP connections. In this case, the underlying TCP connection is closed once all the data has been transferred. This is faster, but there is no guarantee about the reliable delivery of the data.

SCTP uses a different approach to terminante connections. When an application requests a shutdown of a connection, SCTP performs a three-way handshake. This handshake uses the `SHUTDOWN`, `SHUTDOWN-ACK` and `SHUTDOWN-COMPLETE` chunks. The `SHUTDOWN` chunk is sent once all outgoing data has been acknowledged. It contains the last cumulative sequence number. Upon reception of a `SHUTDOWN` chunk, an SCTP entity informs its application that it cannot accept anymore data over this connection. It then ensures that all outstanding data have been delivered correctly. At that point, it sends a `SHUTDOWN-ACK` to confirm the reception of the `SHUTDOWN` segment. The three-way handshake completes with the transmission of the `SHUTDOWN-COMPLETE` chunk [105].

SCTP also provides the equivalent to TCP's `RST` segment. The `ABORT` chunk can be used to refuse a connection, react to the reception of an invalid segment or immediately close a connection (e.g. due to lack
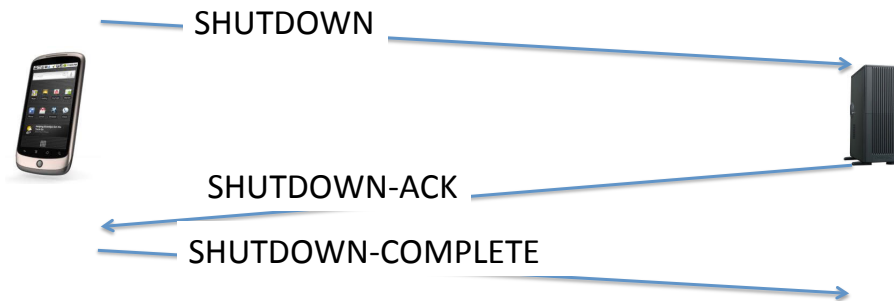
Figure 4: The three-way handshake used to close an SCTP connection

of resources).

2.3 Providing the request-response service

The request-response service has been a popular service since the 1980s. At that time, many request-response applications were built above the connectionless service, typically UDP [15]. A request-response application is very simple. The client sends a request to a server and blocks waiting for the response. The server processes the request and returns a response to the client. This paradigm is often called Remote Procedure Call (RPC) since often the client calls a procedure running on the server.

The first implementations of RPC relied almost exclusively on UDP to transmit the request and responses. In this case, the size of the requests and responses was often restricted to one MTU. In the 1980s and the beginning of the 1990s, UDP was a suitable protocol to transport RPCs because they were mainly used in Ethernet LANs. Few users were considering the utilization of RPC over the WAN. In such networks, CSMA/CD regulated the access to the LAN and there were almost no losses. Over the years, the introduction of Ethernet switches has both allowed Ethernet networks to grow in size but also implied a growing number of packet losses. Unfortunately, RPC running over UDP does not deal efficiently with packet losses because many implementation use large timeouts to recover for packet losses. TCP could deal with losses, but it was considered to be too costly for request-response applications. Before sending a request, the client must first initiate the connection. This requires a three-way handshake and thus "wastes" one round-trip-time. Then, TCP can transfer the request and receive the response over the established connection. Eventually, it performs a graceful shutdown of the connection. This connection release requires the exchange of four (small) segments, but also forces the client to remain in the TIME_WAIT state for a duration of 240 seconds, which limits the number of connections (and thus RPCs) that it can establish with a given server.

TCP for Transactions or T/TCP [19] was a first attempt to enable TCP to better support request/response applications. T/TCP solved the above problem by using three TCP options. These options were mainly used to allow each host to maintain an additional state variable, Connection Count (CC) that is incremented by one for every connection. This state variable is sent in the SYN segment and cached by the server. If a SYN received from a client contains a CC that is larger than the cached one, the new connection is immediately established and data can be exchanged directly (already in the SYN). Otherwise, a normal three-way handshake is used. The use of this state variable allowed T/TCP to reduce the duration of the TIME_WAIT state. T/TCP used SYN and FIN flags in the segment sent by the client and returned by the
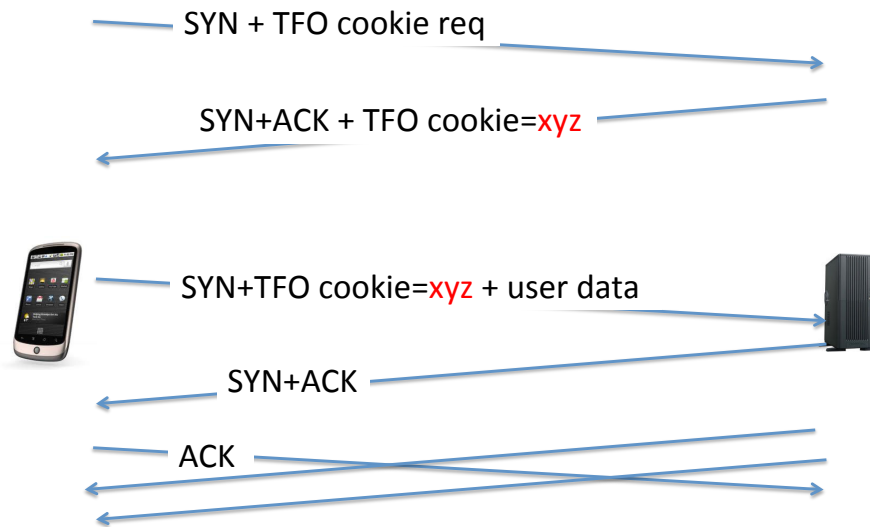
Figure 5: TCP fast open

server, which led to a two segment connection, the best solution from a delay viewpoint for RPC applications. Unfortunately, T/TCP was vulnerable to spoofing attacks [32]. An attacker could observe the Connection Count by capturing packets. Since the server only checked that the value of the CC state variable contained in a `SYN` segment was higher than the cached one, it was easy to inject new segments. Due to this security problem, T/TCP is now deprecated.

Improving the performance of TCP for request/response applications continued to be an objective for many researchers. However, recently the focus of the optimizations moved from the LANs that were typical for RPC applications to the global Internet. The motivation for several of the recent changes to the TCP protocol was the perceived performance of TCP with web search applications [24]. A typical web search is also a very short TCP connection during which a small HTTP request and a small HTTP response are exchanged. A first change to TCP was the increase of the initial congestion window [25]. For many years, TCP used an initial window between 2 and 4 segments [6]. This was smaller than the typical HTTP response from a web search engine [24]. Recent TCP stacks use an initial congestion window of 10 segments [25].

Another change that has been motivated by web search applications is the TCP Fast Open (TFO) extension [91]. This extension can be considered as a replacement for T/TCP. TCP fast open also enables a client to send data inside the `SYN` segment. TCP fast open relies on state sharing between the client and the server, but the state is more secure than the simple counter used by T/TCP. To enable the utilization of TCP fast open, the client must first obtain a *cookie* from the server. This is done by sending a `SYN` segment with the TFO cookie request option. The server then generates a secure cookie by encrypting the IP address of the client with a local secret [91]. The encrypted information is returned inside a TFO cookie option in the `SYN+ACK` segment. The client caches the cookie and associates it with the server's IP address. The subsequent connections initiated by the client will benefit from TCP fast open. The client includes the cached cookie and optional data inside its `SYN` segment. The server can validate the segment by decrypting its cookie. If the cookie is valid, the server acknowledges the `SYN` and the data that it contains. Otherwise,

the optional data is ignored and a normal TCP three-way handshake is used. This is illustrated in figure 5.

3   Today's Internet

The TCP/IP protocol suite was designed with the end-to-end principle [100] in mind. TCP and SCTP are no exception to this rule. They both assume that the network contains relays that operate only at the physical, datalink and network layers of the reference models.

In such an end-to-end Internet, the payload of an IP packet is never modified inside the network and any transport protocol can be used above IPv4 or IPv6. Today, this behavior corresponds to some islands in the Internet like research backbones and some university networks. Measurements performed in enterprise, cellular and other types of commercial networks reveal that IP packets are processed differently in deployed networks [58, 115].

In addition to the classical repeaters, switches and routers, currently deployed networks contain various types of middleboxes [22]. Middleboxes were not part of the original TCP/IP architecture and they have evolved mainly during the last decade. A recent survey in enterprise networks reveals that such networks contain sometimes as many middleboxes as routers [103].

A detailed survey of all possible middleboxes is outside the scope of this chapter, but it is useful to study the operation of some important types of middleboxes to understand their impact on transport protocols and how transport protocols have to cope with them.

3.1   Firewalls

Firewalls perform checks on the received packets and decide to accept or discard them based on configured security policies. Firewalls play an important role in delimiting network boundaries and controlling incoming and outgoing traffic in enterprise networks. In theory, firewalls should not directly affect transport protocols, but in practice, they may block the deployment of new protocols or extensions to existing ones. Firewalls can either filter packets on the basis of a white list, i.e. an explicit list of allowed communication flows, or a black list, i.e. an explicit list of all forbidden communication flows. Most enterprise firewalls use a white list approach. The network administrator defines a set of allowed communication flows, based on the high-level security policies of the enterprise and configures the low-level filtering rules of the firewall to implement these policies. With such a whitelist, all flows that have not been explicitly defined are forbidden and the firewall discards all packets that do not match an accepted communication flow. This unfortunately implies that a packet that contains a different *Protocol* than the classical TCP, ICMP and UDP protocols will usually not be accepted by such a firewall. This is a major hurdle for the deployment of new transport protocols like SCTP.

Some firewalls can perform more detailed verification and maintain state for each established TCP connection. Some of these stateful firewalls are capable of verifying whether a packet that arrives for an accepted TCP connection contains a valid sequence number. For this, the firewall maintains state for each TCP connection that it accepts and when a new data packet arrives, it verifies that it belongs to an established connection and that its sequence number fits inside the advertised receive window. This verification is intended to protect the hosts that reside behind the firewall from packet injection attacks despite the fact that these hosts also need to perform the same verification.

Stateful firewalls may also limit the extensibility of protocols like TCP. To understand the problem, let us consider the large windows extension defined in [63]. This extension fixes one limitation of the original TCP specification. The TCP header [89] includes a 16-bits field that encodes the receive window in the TCP header. A consequence of this choice is that the standard TCP cannot support a receive window larger

70

than 64 KBytes. This is not large enough for high bandwidth networks. To allow hosts to use a larger window, [63] changes the semantics of the *receive window* field of the TCP header on a per-connection basis. [63] defines the `WScale` TCP option that can only be used inside the `SYN` and `SYN+ACK` segments. This extension allows the communicating hosts to maintain their receive window as a 32 bits field. The `WScale` option contains as parameter the number of bits that will be used to shift the 32-bits window to the right before placing the lower 16 bits in the TCP header. This shift is used on a TCP connection provided that both the client and the server have included the `WScale` option in the `SYN` and `SYN+ACK` segments.

Unfortunately, a stateful firewall that does not understand the `WScale` option, may cause problems. Consider for example a client and a server that use a very large window. During the three-way handshake, they indicate with the `WScale` option that they will shift their window by 14 bits to the right. When the connection starts, each host reserves 2^{17} bytes of memory for its receive window[5]. Given the negotiated shift, each host will send in the TCP header a window field set to `0000000000000100`. If the stateful firewall does understand the `WScale` option used in the `SYN` and `SYN+ACK` segments, it will assume a window of 4 bytes and will discard all received segments. Unfortunately, there are still today stateful firewalls[6] that do not understand this TCP option defined in 1992.

Stateful firewalls can perform more detailed verification of the packets exchanged during a TCP connection. For example, intrusion detection and intrusion prevention systems are often combined with traffic normalizers [113, 53]. A traffic normalizer is a middlebox that verifies that all packets obey the protocol specification. When used upstream of an intrusion detection system, a traffic normalizer can for example buffer the packets that are received out-of-order and forward them to the IDS once they are in-sequence.

3.2 Network Address Translators

A second, and widely deployed middlebox, is the Network Address Translator (NAT). The NAT was defined in [38] and various extensions have been developed over the years. One of the initial motivation for NAT was to preserve public IP addresses and allow a group of users to share a single IP address. This enabled the deployment of many networks that use so-called *private addresses* [96] internally and rely on NATs to reach the global Internet. At some point, it was expected that the deployment of IPv6 would render NAT obsolete. This never happened and IPv6 deployment is still very slow [34]. Furthermore, some network administrators have perceived several benefits with the deployment of NATs [51] including : reducing the exposure of addresses, topology hiding, independence from providers, ... Some of these perceived advantages have caused the IETF to consider NAT for IPv6 as well, despite the available address space. Furthermore, the IETF, vendors and operators are considering the deployment of large scale Carrier Grade NATs (CGN) to continue to use IPv4 despite the depletion of the addressing space [83] and also to ease the deployment of IPv6 [64]. NATs will remain a key element of the deployed networks for the foreseeable future.

There are different types of NATs depending on the number of addresses that they support and how they maintain state. In this section, we concentrate on a simple but widely deployed NAT that serves a large number of users on a LAN and uses a single public IP address. In this case, the NAT needs to map several private IP addresses on a single public address. To perform this translation and still allow several internal hosts to communicate simultaneously, the NAT must understand the transport protocol that is used by the internal hosts. For TCP, the NAT needs to maintain a pool of TCP port numbers and use one of the available port as the source port for each new connection initiated by an internal host. Upon reception of a packet, the NAT needs to update the source and destination addresses, the source (or destination) port number and

---

[5]It is common to start a TCP connection with a small receive window/buffer and automatically increase the buffer size during the transfer [102].

[6]See e.g. http://support.microsoft.com/kb/934430

also the IP and TCP checksums. The NAT performs this modification transparently in both directions. It is important to note that a NAT can only change the header of the transport protocol that it supports. Most deployed NATs only support TCP [49], UDP [9] and ICMP [104]. Supporting another transport protocol on a NAT requires software changes [55] and few NAT vendors implement those changes. This often forces users of new transport protocol to tunnel their protocol on top of UDP to traverse NATs and other middleboxes [85, 110]. This limits the ability to innovate in the transport layer.

NAT can be used transparently by most Internet applications. Unfortunately, some applications cannot easily be used over NATs [57]. The textbook example of this problem is the File Transfer Protocol (FTP) [90]. An FTP client uses two types of TCP connections : a control connection and data connections. The control connection is used to send commands to the server. One of these is the `PORT` command that allows to specify the IP address and the port numbers that will be used for the data connection to transfer a file. The parameters of the `PORT` command are sent using a special ASCII syntax [90]. To preserve the operation of the FTP protocol, a NAT can translate the IP addresses and ports that appear in the IP and TCP headers, but also as parameters of the `PORT` command exchanged over the data connection. Many deployed NATs include Application Level Gateways (ALG) [57] that implement part of the application level protocol and modify the payload of segments. For FTP, after translation a `PORT` command may be longer or shorter than the original one. This implies that the FTP ALG needs to maintain state and will have to modify the sequence/acknowledgment numbers of all segments sent over a connection after having translated a `PORT` command. This is transparent for the FTP application, but has influenced the design of Multipath TCP although recent FTP implementations rarely use the `PORT` command[7].

3.3 Proxies

The last middlebox that we cover is the proxy. A proxy is a middlebox that resides on a path and terminates TCP connections. A proxy can be explicit or transparent. The SOCKS5 protocol [71] is an example of the utilization of an explicit proxy. SOCKS5 proxies are often used in enterprise networks to authenticate the establishment of TCP connections. A classical example of transparent proxies are the HTTP proxies that are deployed in various commercial networks to cache and speedup HTTP requests. In this case, some routers in the network are configured to intercept the TCP connections and redirect them to a proxy server [75]. This redirection is transparent for the application, but from a transport viewpoint, the proxy acts as a relay between the two communicating hosts and there are two different TCP connections[7]. The first one is initiated by the client and terminates at the proxy. The second one is initiated by the proxy and terminates at the server. The data exchanged over the first connection is passed to the second one, but the TCP options are not necessarily preserved. In some deployments, the proxy can use different options than the client and/or the server.

3.4 How prevalent are middleboxes?

We've learnt that middleboxes of various kinds exist, but are they really deployed in today's networks? Answering this question can guide the right way to develop new protocols and enhance existing ones.

Here we briefly review measurement studies that have attempted to paint an accurate image of middlebox deployments in use. The conclusion is that middleboxes are widespread to the point where end-to-end paths without middleboxes have become exceptions, rather than the norm.

---

[7]Some deployments use several proxies in cascade. This allows the utilization of compression techniques and other non-standard TCP extensions on the connection between two proxies.

There are two broad types of middlebox studies. The first type of study uses ground-truth topology information from providers and other organizations. While accurate, these studies may overestimate the influence of middleboxes on end-to-end traffic because certain behavior is only triggered in rare, corner cases (e.g. an intrusion prevention system may only affect traffic carrying known worm signatures). These studies do not tell us exactly *what operations* are applied to packets by middleboxes. One recent survey study has found that the 57 enterprise networks surveyed deploy as many middleboxes as routers [103].

Active measurements probe end-to-end paths to trigger "known" behaviors of middleboxes. Such measurements are very accurate, pinpointing exactly what middleboxes do in certain scenarios. However, they offer a lower bound of middlebox deployments, as they may pass through other middleboxes without triggering them. Additionally, such studies are limited to probing a full path (cannot probe path segments) thus cannot tell how many middleboxes are deployed on a path exhibiting middlebox behavior. The data surveyed below comes from such studies.

Network Address Translators are easy to test for: the traffic source needs to compare its local source address with the source address of its packets reaching an external site (e.g. what's my IP). When the addresses differ, a NAT has been deployed on path. Using this basic technique, existing studies have shown that NATs are deployed almost universally by (or for) end-users, be they home or mobile:

- Most cellular providers use them to cope with address space shortage and to provide some level of security. A study surveying 107 cellular networks across the globe found that 82 of them used NATs [115].

- Home users receive a single public IP address (perhaps via DHCP) from their access providers, and deploy NATs to support multiple devices. The typical middlebox here is the "wireless router", an access point that aggregates all home traffic onto the access link. A study using peer-to-peer clients found that only 12% of the peers have a public IP address [30].

Testing for stateless firewalls is equally simple: the client generates traffic using different transport protocols and port numbers, and the server acks it back. As long as some tests work, the client knows the endpoint is reachable, and that the failed tests are most likely due to firewall behavior. There is a chance that the failed tests are due to the stochastic packet loss inherent in the Internet; that is why tests are interleaved and run multiple times.

Firewalls are equally widespread in the Internet, being deployed by most cellular operators [115]. Home routers often act as firewalls, blocking new protocols and even existing ones (e.g. UDP) [30]. Most servers also deploy firewalls to restrict in-bound traffic [103]. Additionally, many modern operating systems come with "default-on" firewalls. For instance, the Windows 7 firewall explicitly asks users to allow incoming connections and to whitelist applications allowed to make outgoing connections.

Testing for explicit proxies can be done by comparing the segments that leave the source with the ones arriving at the destination. A proxy will change sequence numbers, perhaps segment packets differently, modify the receive window, and so forth. Volunteers from various parts of the globe ran TCPExposure, a tool that aims to detect such proxies and other middlebox behavior [58]. The tests cover 142 access networks (including cellular, DSL, public hotspots and offices) in 24 countries. Proxying behavior was seen on 10% of the tested paths[58].

Beyond basic reachability, middleboxes such as traffic normalizers and stateful firewalls have some expectations on the packets they see: what exactly do these middleboxes do, and how widespread are they? The same study sheds some light into this matter:

- Middlebox behavior depends on the ports used. Most middleboxes are active on port 80 (HTTP traffic).

73

- A third of paths keep TCP flow state and use it to actively correct acknowledgments for data the middlebox has not seen. To probe this behavior, TCPExposure sends a few TCP segments leaving a gap in the sequence number, while the server acknowledgment also covers the gap.

- 14% of paths remove unknown options from SYN packets. These paths will not allow TCP extensions to be deployed.

- 18% of paths modify sequence numbers; of these, a third seem to be proxies, and the other are most likely firewalls than randomize the initial sequence number to protect vulnerable end hosts against in-window injection attacks.

3.5 The Internet is Ossified

Deploying a new IP protocol requires a lot of investment to change all the deployed hardware. Experience with IPv6 after 15 years since it has been standardized is not encouraging: only a minute fraction of the Internet has migrated to v6, and there are no signs of it becoming "the Internet" anytime soon.

Changing IPv4 itself is in theory possible with IP options. Unfortunately, it has been known for a while now that "IP options are not an option" [42]. This is because existing routers implement forwarding in hardware for efficiency reasons; packets carrying unknown IP options are treated as exceptions that are processed in software. To avoid a denial-of-service on routers' CPU's, such packets are dropped by most routers.

In a nutshell, we can't really touch IP - but have we also lost our ability to change transport protocols as well? The high level picture emerging from existing middlebox studies is that of a network that is highly tailored to today's traffic to the point it is ossified: *changing existing transport protocols is challenging as it needs to carefully consider middlebox interactions*. Further, *deploying new transport protocols natively is almost impossible*.

Luckily, changing transport protocols is still possible, albeit great care must be taken when doing so. Firewalls will block *any* traffic they do not understand, so deploying new protocols must necessarily use existing ones just to get through the network. This observation has recently lead to the development of Minion, a container protocol that enables basic connectivity above TCP while avoiding its in-order, reliable bytestream semantics at the expense of slightly increased bandwidth usage. We discuss Minion in Section 5.

Even changing TCP is very difficult. The semantics of TCP are embedded in the network fabric, and new extensions must function within the confines of these semantics, or they will fail. In section 4 we discuss Multipath TCP, another recent extension to TCP, that was designed explicitly to be middlebox compatible.

4   Multipath TCP

Today's networks are multipath: mobile devices have multiple wireless interfaces, datacenters have many redundant paths between servers and multi-homing has become the norm for big server farms. Meanwhile, TCP is essentially a single path protocol: when a TCP connection is established, it is bound to the IP addresses of the two communicating hosts. If one of these addresses changes, for whatever reason, the connection fails. In fact a TCP connection cannot even be load-balanced across more than one path within the network, because this results in packet reordering and TCP misinterprets this reordering as congestion and slows down.

This mismatch between today's multipath networks and TCP's single-path design creates tangible problems. For instance, if a smartphone's WiFi interface loses signal, the TCP connections associated with it

stall - there is no way to migrate them to other working interfaces, such as 3G. This makes mobility a frustrating experience for users. Modern datacenters are another example: many paths are available between two endpoints, and equal cost multipath routing randomly picks one for a particular TCP connection. This can cause collisions where multiple flows get placed on the same link, hurting throughput - to such an extent that average throughput is halved in some scenarios.

Multipath TCP (MPTCP) [13] is a major modification to TCP that allows multiple paths to be used simultaneously by a single connection. Multipath TCP circumvents the issues above and several others that affect TCP. Changing TCP to use multiple paths is not a new idea: it was originally proposed more than fifteen years ago by Christian Huitema in the Internet Engineering Task Force (IETF) [59], and there have been a half-dozen more proposals since then to similar effect. Multipath TCP draws on the experience gathered in previous work, and goes further to solve issues of fairness when competing with regular TCP and deployment issues due to middleboxes in today's Internet.

4.1 Overview of Multipath TCP

The design of Multipath TCP has been influenced by many requirements, but there are two that stand out: application compatibility and network compatibility. Application compatibility implies that applications that today run over TCP should work without any change over Multipath TCP. Next, Multipath TCP must operate over any Internet path where the TCP protocol operates.

As explained earlier, many paths on today's Internet include middleboxes that, unlike routers, know about the TCP connections they forward, and affect them in special ways. Designing TCP extensions that can safely traverse all these middleboxes has proven to be challenging.

Multipath TCP allows multiple "subflows" to be combined to form a single MPTCP session. An MPTCP session starts with an initial subflow which is very similar to a regular TCP connection, with a three way handshake. After the first MPTCP subflow is set up, additional subflows can be established. Each subflow also looks very similar to a regular TCP connection, complete with three-way handshake and graceful teardown, but rather than being a separate connection it is bound into an existing MPTCP session. Data for the connection can then be sent over any of the active subflows that has the capacity to take it.

To examine Multipath TCP in more detail, let us consider a very simple scenario with a smartphone client and a single-homed server. The smartphone has two network interfaces: a WiFi interface and a 3G interface; each has its own IP address. The server, being single-homed, has a single IP address. In this environment, Multipath TCP would allow an application on the mobile device to use a single MPTCP session that can use both the WiFi and the 3G interfaces to communicate with the server. The application opens a regular TCP socket, and the kernel enables MPTCP by default if the remote end supports it, using both paths. The application does not need to concern itself with which radio interface is working best at any instant; MPTCP handles that for it. In fact, Multipath TCP can work when both endpoints are multihomed (in this case subflows are opened between all pairs of "compatible" IP addresses), or even in the case when both endpoints are single homed (in this case different subflows will use different port numbers, and can be routed differently by multipath routing in the network).

Connection Setup. Let us walk through the establishment of an MPTCP connection. Assume that the mobile device chooses its 3G interface to open the connection. It first sends a `SYN` segment to the server. This segment contains the `MP_CAPABLE` TCP option indicating that the mobile device supports Multipath TCP. This option also contains a key which is chosen by the mobile device. The server replies with a `SYN+ACK` segment containing the `MP_CAPABLE` option and the key chosen by the server. The mobile device completes the handshake by sending an `ACK` segment.
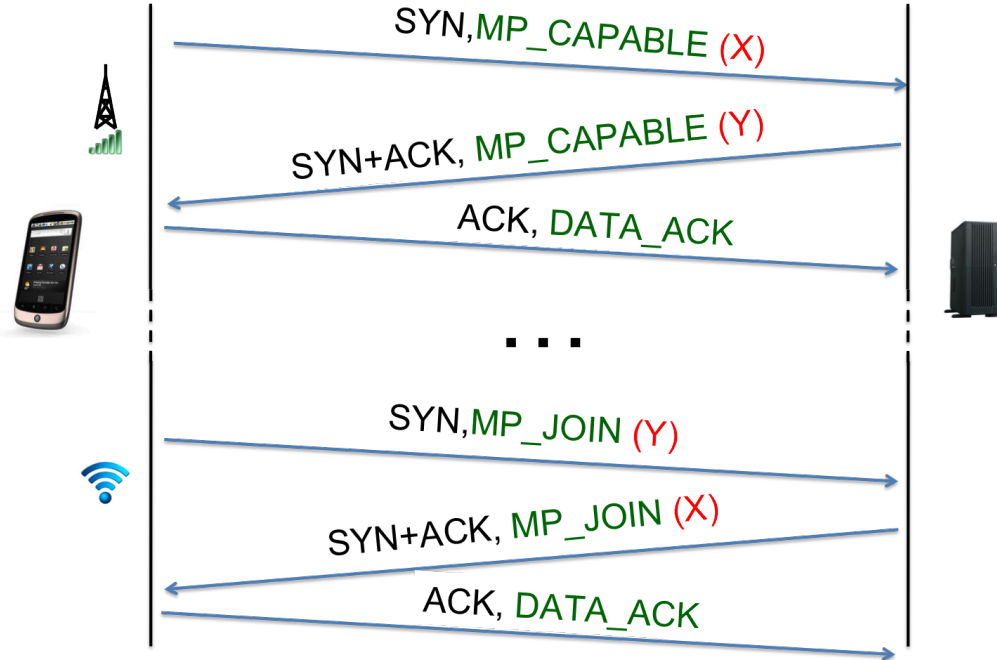
Figure 6: Multipath TCP handshake: multiple subflows can be added and removed after the initial connection is setup and connection identifiers are exchanged.

The initial MPTCP connection setup is shown graphically in the top part of Figure 6, where the segments are regular TCP segments carrying new multipath TCP-related options (shown in green).

At this point the Multipath TCP connection is established and the client and server can exchange TCP segments via the 3G path. How could the mobile device also send data through this Multipath TCP session over its WiFi interface?

Naively, it could simply send some of the segments over the WiFi interface. However most ISPs will drop these packets, as they would have the source address of the 3G interface. Perhaps the client could tell the server the IP address of the WiFi interface, and use that when it sends over WiFi? Unfortunately this will rarely work: firewalls and similar stateful middleboxes on the WiFi path expect to see a SYN segment before they see data segment. The only solution that will work reliably is to perform a regular three-way handshake on the WiFi path before sending any packets that way, so this is what Multipath TCP does. This handshake carries the MP_JOIN TCP option, providing information to the server that can securely identify the correct connection to associate this additional subflow with. The server replies with MP_JOIN in the SYN+ACK, and the new subflow is established (this is shown in the bottom part of Figure 6).

An important point about Multipath TCP, especially in the context of mobile devices, is that the set of subflows that are associated to a Multipath TCP connection is not fixed. Subflows can be dynamically added and removed from a Multipath TCP connection throughout its lifetime, without affecting the bytestream transported on behalf of the application. If the mobile device moves to another WiFi network, it will receive a new IP address. At that time, it will open a new subflow using its newly allocated address and tell the server that its old address is not usable anymore. The server will now send data towards the new address.

These options allow mobile devices to easily move through different wireless connections without breaking their Multipath TCP connections [80].

Multipath TCP also implements mechanisms that allows to inform the remote host of the addition/removal of addresses even when an endpoint operates behind a NAT, or when a subflow using a different address family is needed (e.g. IPv6). Endpoints can send an ADD_ADDR option that contains an address identifier together with an address. The address identifier is unique at the sender, and allows it to identify its addresses even when it is behind a NAT. Upon receiving an advertisement, the endpoint may initiate a new subflow to the new address. An address withdrawal mechanism is also provided via the REMOVE_ADDR option that also carries an address identifier.

Data Transfer. Assume now that two subflows have been established over WiFi and 3G: the mobile device can send and receive data segments over both. Just like TCP, Multipath TCP provides a bytestream service to the application. In fact, standard applications can function over MPTCP without being aware of it - MPTCP provides the same socket interface as TCP.

Since the two paths will often have different delay characteristics, the data segments sent over the two subflows will not be received in order. Regular TCP uses the sequence number in the TCP header to put data back into the original order. A simple solution for Multipath TCP would be to just reuse this sequence number as is.

Unfortunately, this simple solution would create problems with some existing middleboxes such as firewalls. On each path, a middlebox would only see half of the packets, so it would observe many gaps in the TCP sequence space. Measurements indicate that some middleboxes react in strange ways when faced with gaps in TCP sequence numbers [58]. Some discard the out-of-sequence segments while others try to update the TCP acknowledgments in order to "recover" some of these gaps. With such middleboxes on a path, Multipath TCP cannot safely send TCP segments with gaps in the TCP sequence number space. On the other hand, Multipath TCP also cannot send every data segment over all subflows: that would be a waste of resources.

To deal with this problem, Multipath TCP uses its own sequence numbering space. Each segment sent by Multipath TCP contains two sequence numbers: the subflow sequence number inside the regular TCP header and an additional Data Sequence Number (DSN).

This solution ensures that the segments sent on any given subflow have consecutive sequence numbers and do not upset middleboxes. Multipath TCP can then send some data sequence numbers on one path and the remainder on the other path; the DSN will be used by the Multipath TCP receiver to reorder the bytestream before it is given to the receiving application.

Before we explain the way the Data Sequence Number is encoded, we first need to discuss two other key parts of Multipath TCP that are affected by the additional sequence number space—flow control and acknowledgements.

Flow Control. TCP's receive window indicates the number of bytes beyond the sequence number from the acknowledgment field that the receiver can buffer. The sender is not permitted to send more than this amount of additional data. Multipath TCP also needs to implement flow control, although segments can arrive over multiple subflows. If we inherit TCP's interpretation of receive window, this would imply an MPTCP receiver maintains a pool of buffering per subflow, with receive window indicating per-subflow buffer occupancy. Unfortunately such an interpretation can lead to deadlocks:

1. The next segment that needs to be passed to the application was sent on subflow 1, but was lost.

2. In the meantime subflow 2 continues delivering data, and fills its receive window.

3. Subflow 1 fails silently.

4. The missing data needs to be re-sent on subflow 2, but there is no space left in the receive window, resulting in a deadlock.

The correct solution is to generalize TCP's receive window semantics to MPTCP. For each connection *a single receive buffer pool should be shared between all subflows*. The receive window then indicates the maximum data sequence number that can be sent rather than the maximum subflow sequence number. As a segment resent on a different subflow always occupies the same data sequence space, deadlocks cannot occur.

The problem for an MPTCP sender is that to calculate the highest data sequence number that can be sent, the receive window needs to be added to the highest data sequence number acknowledged. However the ACK field in the TCP header of an MPTCP subflow must, by necessity, indicate only subflow sequence numbers to cope with middleboxes. Does MPTCP need to add an extra data acknowledgment field for the receive window to be interpreted correctly?

Acknowledgments. The answer is positive: MPTCP needs and uses *explicit connection-level acknowledgments* or *DATA_ACKs*. The alternative is to infer connection-level acknowledgments from subflow acknowledgments, by using a scoreboard maintained by the sender that maps subflow sequence numbers to data sequence numbers. Unfortunately, MPTCP segments and their associated ACKs will be reordered as they travel on different paths, making it impossible to correctly infer the connection-level acknowledgments from subflow-level ones [94].

Encoding. We have seen that in the forward path we need to encode a mapping of subflow bytes into the data sequence space, and in the reverse path we need to encode cumulative data acknowledgments. There are two viable choices for encoding this additional data:

- Send the additional data in TCP options.

- Carry the additional data within the TCP payload, using a chunked or escaped encoding to separate control data from payload data.

For the forward path there aren't compelling arguments either way, but the reverse path is a different matter. Consider a hypothetical encoding that divides the payload into chunks where each chunk has a TLV (type-length-value) header. A data acknowledgment can then be embedded into the payload using its own chunk type. Under most circumstances this works fine. However, unlike TCP's pure ACK, anything embedded in the payload must be treated as data. In particular:

- It must be subject to flow control because the receiver must buffer data to decode the TLV encoding.

- If lost, it must be retransmitted consistently, so that middleboxes can track sequence state correctly[8]

- If packets before it are lost, it might be necessary to wait for retransmissions before the data can be parsed - causing head-of-line blocking.

Flow control presents the most obvious problem for the chunked payload encoding. Figure 7 provides an example. Client C is pipelining requests to server S; meanwhile S's application is busy sending the large

---

[8]TCP proxies re-send the original content they see a "retransmission" with different data.
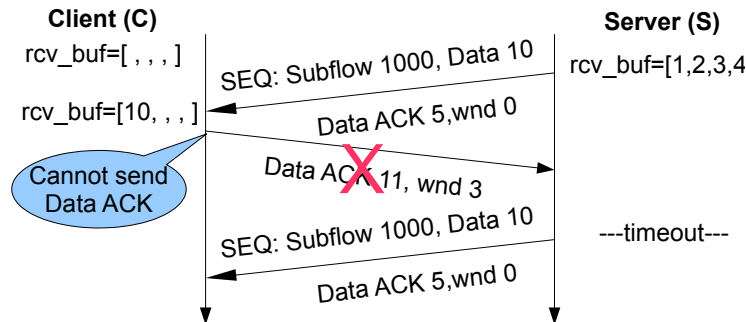
Figure 7: Flow Control on the path from C to S inadvertently stops the data flow from S to C

response to the first request so it isn't yet ready to read the subsequent requests. At this point, S's receive buffer fills up.

S sends segment 10, C receives it and wants to send the DATA_ACK, but cannot: flow control imposed by S's receive window stops him. Because no DATA_ACKs are received from C, S cannot free his send buffer, so this fills up and blocks the sending application on S. S's application will only read when it has finished sending data to C, but it cannot do so because its send buffer is full. The send buffer can only empty when S receives the DATA_ACK from C, but C cannot send this until S's application reads. This is a classic deadlock cycle. As no DATA_ACK is received, S will eventually time out the data it sent to C and will retransmit it; after many retransmits the whole connection will time out.

The conclusion is that DATA_ACKs cannot be safely encoded in the payload. The only real alternative is to encode them in TCP options which (on a pure ACK packet) are not subject to flow control.

The Data Sequence Mapping. If MPTCP must use options to encode DATA_ACKs, it is simplest to also encode the mapping from subflow sequence numbers to data sequence numbers in a TCP option. This is the *data sequence mapping* or DSM.

At first glance it seems the DSM option simply needs to carry the data sequence number corresponding to the start of the MPTCP segment. Unfortunately middleboxes and interfaces that implement TSO or LRO make this far from simple.

Middleboxes that re-segment data would cause a problem. TCP Segmentation Offload (TSO) hardware in the network interface card (NIC) also re-segments data and is commonly used to improve performance. The basic idea is that the OS sends large segments and the NIC re-segments them to match the receiver's MSS. What does TSO do with TCP options? A test of 12 NICs supporting TSO from four different vendors showed that all of them copy a TCP option sent by the OS on a large segment into all the split segments [94].

If MPTCP's DSM option only listed the data sequence number, TSO would copy the same DSM to more than one segment, breaking the mapping. Instead the DSM option must say precisely which subflow bytes map to which data sequence numbers. But this is further complicated by middleboxes that modify the initial sequence number of TCP connections and consequently rewrite all sequence numbers (many firewalls behave like this). Instead, the DSM option must map the offset from the subflow's initial sequence number

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +---------------+---------------+-------+---------------------+
   |     Kind      |    Length     |Subtype| (reserved) |F|m|M|a|A|
   +---------------+---------------+-------+---------------------+
   |            Data ACK (4 or 8 octets, depending on flags)     |
   +------------------------------------------------------------+
   |    Data Sequence Number (4 or 8 octets, depending on flags) |
   +------------------------------------------------------------+
   |              Subflow Sequence Number (4 octets)            |
   +----------------------------+-------------------------------+
   |  Data-level Length (2 octets) |     Checksum (2 octets)     |
   +----------------------------+-------------------------------+
```
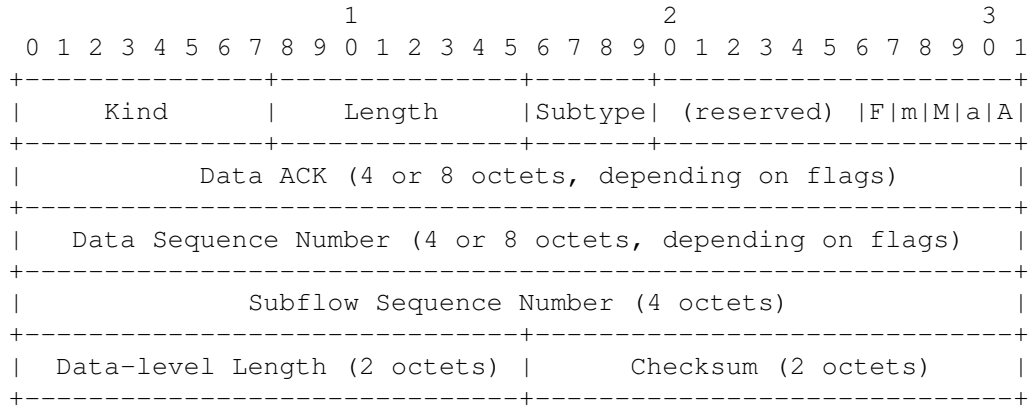
Figure 8: The Data Sequence Signal option in MPTCP that carries the Data Sequence Mapping information, the Data ACK, the Data FIN and connection-fall-back options.

to the data sequence number, as the offset is unaffected by sequence number rewriting. The option must also contain the length of the mapping. This is robust - as long as the option is received, it does not greatly matter which packet carries it, so duplicate mappings caused by TSO are not a problem.

Dealing with Content-Modifying Middleboxes.   Multipath TCP and content-modifying middleboxes (such as application-level NATs, e.g. for FTP) have the potential to interact badly. In particular, due to FTP's ASCII encoding, re-writing an IP address in the payload can necessitate changing the length of the payload. Subsequent sequence and ACK numbers are then fixed up by the middlebox so they are consistent from the point of view of the end systems.

Such length changes break the DSM option mapping - subflow bytes can be mapped to the wrong place in the data stream. They also break every other possible mapping mechanism, including chunked payloads. There is no easy way to handle such middleboxes.

That is why MPTCP includes an optional checksum in the DSM mapping to detect such content changes. If an MPTCP host receives a segment with an invalid DSM checksum, it rejects the segment and triggers a fallback process: if any other subflows exists, MPTCP terminates the subflow on which the modification occurred; if no other subflow exists, MPTCP drops back to regular TCP behavior for the remainder of the connection, allowing the middlebox to perform rewriting as it wishes. This fallback mechanism preserves connectivity in the presence of middleboxes.

For efficiency reasons, MPTCP uses the same 16-bit ones complement checksum used in the TCP header. This allows the checksum over the payload to be calculated only once. The payload checksum is added to a checksum of an MPTCP pseudo header covering the DSM mapping values and then inserted into the DSM option. The same payload checksum is added to the checksum of the TCP pseudo-header and then used in the TCP checksum field. MPTCP allows checksums to be disabled for high performance environments such as data-centers where there is no chance of encountering such an application-level gateway.

The fall-back-to-TCP process, triggered by a checksum failure, can also be triggered in other circumstances. For example, if a routing change moves an MPTCP subflow to a path where a middlebox removes DSM options, this also triggers the fall-back procedure.

Connection Release. Multipath TCP must allow a connection to survive even though its subflows are coming and going. Subflows in MPTCP can be torn down by means of a four-way handshake as regular TCP flows—this ensures MPTCP allows middleboxes to clear their state when a subflow is not used anymore.

MPTCP uses an explicit four-way handshake for connection tear-down indicated by a DATA_FIN option. The DATA_FIN is MPTCP's equivalent to TCP's FIN, and it occupies one byte in the data-sequence space. A DATA_ACK will be used to acknowledge the receipt of the DATA_FIN. MPTCP requires that the segment(s) carrying a DATA_FIN must also have the FIN flag set - this ensures all subflows are also closed when the MPTCP connection is being closed.

For reference, we show the wire format of the option used by MPTCP for data exchange in Figure 8. This option encodes the Data Sequence Mapping, Data ACK, Data FIN and the fall-back options. The flags specify which parts of the option are valid, and help reduce option space usage.

4.2 Congestion Control

One of the most important components in TCP is its congestion controller which enables it to adapt its throughput dynamically in response to changing network conditions. To perform this functionality, each TCP sender maintains a congestion window w which governs the amount of packets that the sender can send without waiting for an acknowledgment. The congestion window is updated dynamically according to the following rules:

- On each ACK, increase the window w by $1/w$.

- Each loss decrease the window w by $w/2$.

TCP congestion control ensures fairness: when multiple connections utilize the same congested link each of them will independently converge to the same average value of the congestion window.

What is the equivalent of TCP congestion control for multipath transport? The obvious question to ask is why not just run regular TCP congestion control on each subflow? Consider the scenario in Fig. 9. If multipath TCP ran regular TCP congestion control on both paths, then the multipath flow would obtain twice as much throughput as the single path flow (assuming all RTTs are equal). This is unfair. To solve this problem, one solution is to try and detect shared bottlenecks but that is unreliable; a better solution is be less aggressive on each subflow (i.e. increase window slower) such that in aggregate the MPTCP connection is no more aggressive than a single regular TCP connection.

Before we describe solutions to MPTCP congestion control, let's discuss the three goals that multipath congestion control must obey [117]:

Fairness If several subflows of the same MPTCP connection share a bottleneck link with other TCP connections, MPTCP should not get more throughput than TCP.

Deployability The performance of all the Multipath TCP subflows together should be at least that of regular TCP on any of the paths used by a Multipath TCP connection. This ensures that there is an incentive to deploy Multipath TCP.

Efficiency A final, most important goal is that Multipath TCP should prefer efficient paths, which means it should send more of its traffic on paths experiencing less congestion.

Intuitively, this last goal ensures wide-area load balancing of traffic: when a multipath connection is using two paths loaded unevenly (such as Figure 10), the multipath transport will prefer the unloaded path
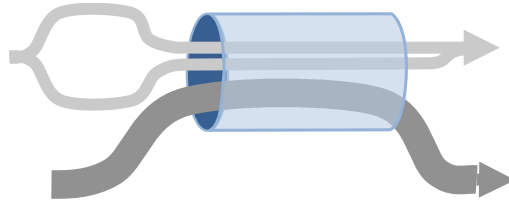
Figure 9: A scenario which shows the importance of weighting the aggressiveness of subflows (Reprinted from [117]. Included here by permission).

and push most of its traffic there; this will decrease the load on the congested link and increase it on the less congested one.

If a large enough fraction of flows are multipath, congestion will spread out evenly across collections of links, creating "resource pools": links that act together as if they are a single, larger capacity link shared by all flows. This effect is called resource pooling [116]. Resource pooling brings two major benefits, discussed in the paragraphs below.

Increased Fairness. Consider the example shown in Figure 10: congestion balancing ensures that all flows have **the same throughput**, making the two links of 20 pkt/s act like a single pooled link with capacity 40 pkt/s shared fairly by the four flows. If more MPTCP flows would be added, the two links would still behave as a pool, sharing capacity fairly among all flows. Conversely, if we remove the Multipath TCP flow, the links no longer form a pool, and the throughput allocation is unfair as the TCP connection using the top path gets twice as much throughput as the TCP connections using the bottom path.

Increased Throughput. Consider the somewhat contrived scenario in Fig.11, and suppose that the three links each have capacity 12Mb/s. If each flow split its traffic evenly across its two paths subflow would get 4Mb/s hence each flow would get 8Mb/s. But if each flow used only the one-hop shortest path, it could get 12Mb/s: this is because two-hop paths consume double the resources of one-hop paths, and in a congested network it makes sense to only use the one-hop paths.

In an idle network, however, using all available paths is much better: consider the case when only the blue connection is using the links. In this case this connection would get 24Mb/s throughput; using the one hop path alone would only provide 12Mb/s.

In summary, the endpoints need to be able to dynamically decide which paths to use based on conditions in the network. A solution has been devised in the theoretical literature on congestion control, independently by Kelly and Voice [65] and Han et al. [52]. The core idea is that a multipath flow should shift all its traffic onto the least-congested path. In a situation like Fig. 11 the two-hop paths will have higher drop probability than the one-hop paths, so applying the core idea will yield the efficient allocation. Surprisingly it turns out that this can be achieved by doing independent congestion control at endpoints.

Multipath TCP Congestion Control. The theoretical work on multipath congestion control [65, 52] assumes a rate-based protocol, with exponential increases of the rate. TCP, in contrast, is a packet-based protocol, sending w packets every round-trip time (i.e. the rate is w/RTT); a new packet is sent only when an acknowledgment is received, confirming that an existing packet has left the network. This property is called ACK-clocking and is nice because it has good stability properties: when congestion occurs round-trip times increase (due to buffering), which *automatically reduces the effective rate* [62].
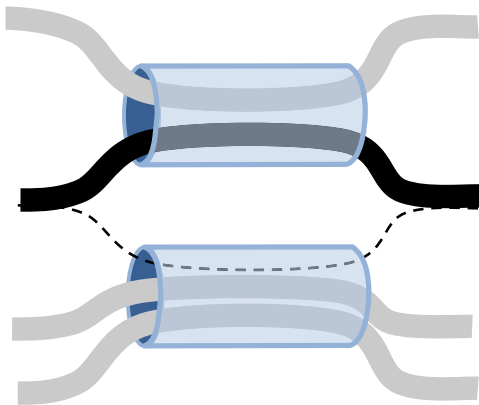
Figure 10: Two links each with capacity 20 pkts/s. The top link is used by a single TCP connection, and the bottom link is used by two TCP connections. A Multipath TCP connection uses both links. Multipath TCP pushes most of its traffic onto less congested top link, making the two links behave like a resource pool of capacity 40 pkts/s. Capacity is divided equally, with each flow having throughput 10 pkts/s. (Reprinted from [117]. Included here by permission)

Converting a theoretical rate-based exponential protocol to a practical packet-based protocol fair to TCP turned out to be more difficult than expected. There are two problems that appear [117]:

- When loss rates are equal on all paths, the theoretical algorithm will place all of the window on one path or the other, not on both—this effect was termed "flappiness" and it appears because of the discrete (stochastic) nature of packet losses which are not captured by the differential equations used in theory.

- The ideal algorithm always prefers paths with lower loss rate, but in practice these may have poor performance. Consider a mobile phone with WiFi and 3G links: 3G links have very low loss rates and huge round-trip times, resulting in poor throughput. WiFi is lossy, has shorter round-trip times and
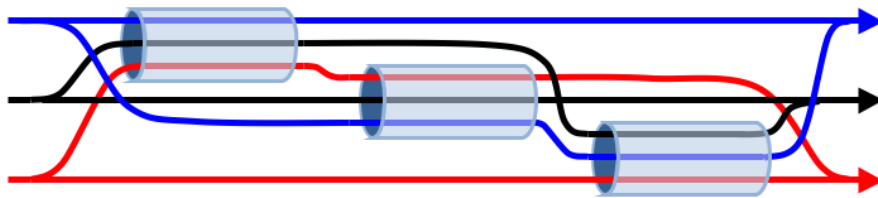
Figure 11: A scenario to illustrate the importance of choosing the less-congested path. (Reprinted from [117]. Included here by permission.)

83

typically offers much better throughput. In this common case, a "perfect" controller would place all traffic on the 3G path, violating the second goal (deployability).

The pragmatic choice is to sacrifice some load-balancing ability to ensure greater stability and to offer incentives for deployment. This is what Multipath TCP congestion control does.

Multipath TCP congestion control is a series of simple changes to the standard TCP congestion control mechanism. Each subflow has its own congestion window, that is halved when packets are lost, as in standard TCP [117].

Congestion balancing is implemented in the increase phase of congestion control: here Multipath TCP will allow less congested subflows to increase proportionally more than congested ones. Finally, the total increase of Multipath TCP across all of its subflows is dynamically chosen in such a way that it achieves the first and second goals above.

The exact algorithm is described below and it satisfies the goals we've discussed:

- Upon ACK on subflow r, increase the window w_r by $\min(a/w_{total}), 1/w_r)$.

- Upon loss on subflow r, decrease the window w_r by $w_r/2$.

Here

$$a = w_{total} \frac{\max_r w_r/RTT_r^2}{(\sum_r w_r/RTT_r)^2}, \tag{1}$$

w_r is the current window size on subflow r and w_{total} is the sum of windows across all subflows.

The algorithm biases the increase towards uncongested paths: these will receive more ACKs and will increase accordingly. However, MPTCP does keep some traffic even on the highly congested paths; this ensures stability and allows it to quickly detect when path conditions improve.

a is a term that is computed dynamically upon each packet drop. Its purpose is to make sure that MPTCP gets at least as much throughput as TCP on the best path. To achieve this goal, a is computed by estimating how much TCP would get on each MPTCP path (this is easy, as round-trip time and loss-rates estimates are known) and ensuring that MPTCP in stable state gets at least that much. A detailed discussion on the design of the MPTCP congestion control algorithm is provided in [117].

For example, in the three-path example above, the flow will put 45% of its weight on each of the less congested path and 10% on the more congested path. This is intermediate between regular TCP (33% on each path) and a perfect load balancing algorithm (0% on the more congested path) that is impossible to implement in practice.

The window increase is capped at $1/w_r$, which ensures that the multipath flow can take no more capacity on either path than a single-path TCP flow would.

In setting the a parameter, Multipath TCP congestion control uses subflow delays to compute the target rate. Using delay for congestion control is not a novel idea: TCP Vegas [20], for instance, performs congestion control only by tracking RTT values. Vegas treats increasing RTTs as a sign of congestion, instead of relying on packet losses as regular TCP does. That is why Vegas loses out to regular TCP at shared bottlenecks, and is probably the reason for its lack of adoption. MPTCP does not treat delay as congestion: it just uses it to figure out the effective rate of a TCP connection on a given path. This allows MPTCP to compete fairly with TCP.

Alternative Congestion Controllers for Multipath TCP. The standardized Multipath TCP congestion control algorithm chooses a trade-off between load balancing, stability and the ability to quickly detect available capacity. The biggest contribution of this work is the clearly defined goals for what multipath congestion control should do, and an instantiation that achieves (most of) the stated goals in practice.
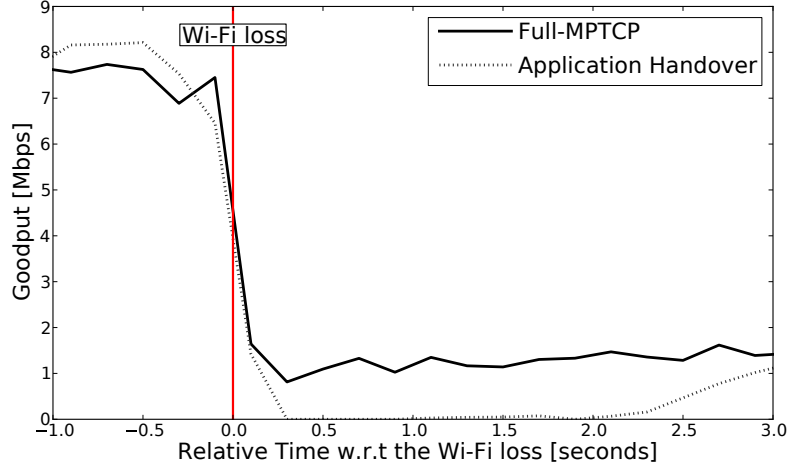
Figure 12: (Mobility) A mobile device is using both its WiFi and 3G interfaces, and then the WiFi interface fails. We plot the instantaneous throughputs of Multipath TCP and application-layer handover. (Reprinted from [80]; ©2012, ACM. Included here by permission.)

This research area is relatively new, and it is likely that more work will lead to better algorithms—if not generally applicable, then at least tailored to some practical use-cases. A new and interesting congestion controller called Opportunistic Linked Increases Algorithm (OLIA) has already been proposed [66] that offers better load balancing with seemingly few drawbacks.

We expect this area to be very active in the near future; of particular interest are designing multipath versions of high-speed congestion control variants deployed in practice, such as Cubic or Compound TCP.

4.3 Implementation and performance

We now briefly cover two of the most compelling use cases for Multipath TCP by showing a few evaluation results. We focus on mobile devices and datacenters but note that Multipath TCP can also help in other scenarios. For example, multi-homed web-servers can perform fine-grained load-balancing across their uplinks, while dual-stack hosts can use both IPv4 and IPv6 within a single Multipath TCP connection.

The full Multipath TCP protocol has been implemented in the Linux kernel; its congestion controller has also been implemented in the ns2 and htsim network simulators. The results presented in here are from the Linux kernel implementation [94].

The mobile measurements focus on a typical mode of operation where the device is connected to WiFi, the connection goes down and the phone switches to using 3G. The setup uses a Linux laptop connected to a WiFi and a 3G network, downloading a file using HTTP.

3G to Wifi handover is implemented in today's phones by changing the application to monitor the network interfaces. When the app detects the loss of the interface, it creates a new TCP connection to the server and the connection can resume. This solution, while simple, is not applicable to all applications because some of the bytes succesfully delivered by the old connection may be resent by the new one.

Applications that rely on HTTP GET requests (with no side effects) are, however, easy to change. The HTTP range header allows a client to resume the download of a file from a specified offset. This is the de-facto standard for today's apps.
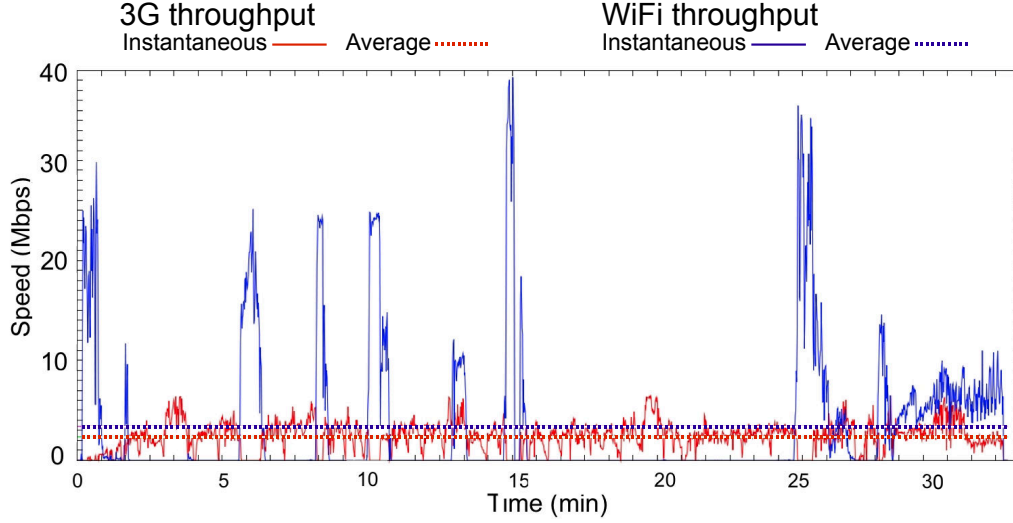
Figure 13: (3G to WiFi Handover) A Linux distribution is downloaded by a mobile user while on the Bucharest subway. 3G coverage is ubiquitous. WiFi is available only in stations.

Figure 12 compares application layer-handover (with HTTP-range) against Multipath TCP. The figure shows a smooth handover with Multipath TCP, as data keeps flowing despite the interface change. With application-layer handover there is a downtime of 4 seconds where the transfer stops-this is because it takes time for the application to detect the interface down event, and it takes time for 3G to ramp up.

Multipath TCP enables unmodified mobile applications to survive interface changes with little disruption. Selected apps can be modified today to support handover, but their performance is worse than with MPTCP. A more detailed discussion of the utilization of Multipath TCP in WiFi/3G environments may be found in [80].

We also present a real mobility trace in Figure 13, where a mobile user downloads a Linux distro on his laptop while travelling on the Bucharest underground. The laptop uses a 3G dongle to connect to the cellular network and its WiFi NIC to connect to access points available in stations.

The figure shows the download speeds of the two interfaces during a 30 minute underground ride. 3G throughput is stable: the average is 2.3Mbps (shown with a dotted line on the graph), and the instantaneous throughput varies inversely proportional with the distance to the 3G cell. WiFi throughput is much more bursty: in some stations the throughput soars to 40Mbps, while in others it is zero, as the laptop doesn't manage to associate and obtain an IP address quickly enough. The average WiFi throughput (3.3Mbps) is higher than the average 3G throughput.

While MPTCP uses both interfaces in this experiment, using just WiFi when it is available may be preferable as it reduces 3G data bills and reduces the load of the cellular network. This is easy to implement with MPTCP, as it supports a simple prioritization mechanism that allows the client to inform the server to send data via preferred subflow(s) [9].

---

[9]MPTCP sends an MP_PRIO option to inform the remote end about changes in priority for the subflows.
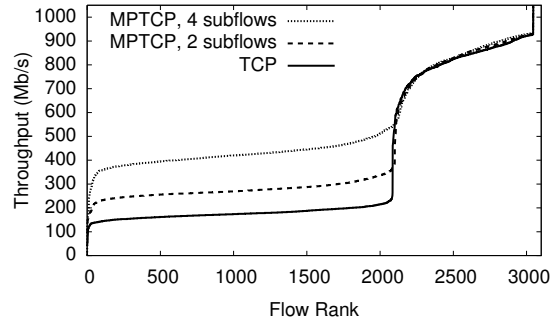
Figure 14: (Datacenter load-balancing) This graph compares standard TCP with MPTCP with two and four flows, when tested on an EC2 testbed with 40 instances. Each host uses iperf sequentially to all other hosts. We plot the performance of all flows (Y axis) in increasing order of their throughputs (X axis). (Reprinted from [92], ©ACM, 2011. Included here by permission)

Datacenters. We also show results from running Multipath TCP in a different scenario: the EC2 datacenter. Like most datacenters today, EC2 uses a redundant network topology where many paths are available between any pair of endpoints, and where connections are placed randomly onto available paths. In EC2, 40 machines (or instances) ran the Multipath TCP kernel. A simple experiment was run where every machine measured the throughput sequentially to every other machine using first TCP, then Multipath TCP with two and with four subflows. Figure 14 shows the sorted throughputs measured over 12 hours. The results show that Multipath TCP brings significant improvements compared to TCP in this scenario. Because the EC2 network is essentially a black-box , it is difficult to pinpoint the root cause for the improvements; however, a detailed analysis of the cases where Multipath TCP can help and why in is presented in [92].

4.4 Impact of Multipath TCP

MPTCP is deployable today, as it was designed and shown to work over existing networks and unmodified applications. Whether MPTCP will be adopted or not is unclear at this point; only time will tell. If MPTCP is adopted, however, it will likely affect the development of both the network and the applications running on top of it. Here we briefly speculate on what the impact might be. Multipath TCP embeds two key mechanisms: load balancing to avoid congested paths, and the ability to use different IP addresses in the same connection. Having these mechanisms implemented at the end-hosts means other layers (e.g. the network) may become simpler, and more efficient.

For instance, MPTCP seamlessly uses available bandwidth even if it is short-lived (e.g. WiFi in underground stations), and provides robustness when a subset of paths fail by quickly resending data on working paths. This could take some of the load imposed on BGP, that today must quickly reconverge in the case of a failure. With MPTCP at the endpoints failures would be "masked" automatically, allowing BGP to react to failures slowly which will ensure there are no oscillations. However, one must ensure that the paths used by the end-systems are disjoint and that the failure does not affect all of them.

Another example is layer 2 mobility, implemented in both WiFi and cellular networks, that aims to do a fast handover from one access point to the next, or from one cell to the next. With MPTCP as a transport protocol, a WiFi client could connect simultaneously to two access points or cells (provided the L2 protocol allows it), and load balance traffic on the best link. MPTCP could replace fast handovers with slow handovers

that are more robust. Of course, some changes may be needed at layer 2 to support such load balancing.

Network topologies may be designed differently if MPTCP is the transport protocol. An example is GRIN [4], a work that proposes to change existing datacenter networks by randomly interconnecting servers in the same rack directly, using their free NIC ports. This allows a server to opportunistically send traffic at speeds larger than its access link by having its idle neighbor relay some traffic on its behalf. With a minor topology change and MPTCP, GRIN manages to better utilize the datacenter network core.

The mechanisms that allow MPTCP to use different IPs in the same transport connection (i.e. the connection identifier) also allow connections to be migrated across physical machines with different IP addresses. One direct application is seamless virtual machine migration: an MPTCP-enabled guest virtual machine can just resume its connections after it is migrated.

On the application side many optimizations are possible. Applications like video and audio streaming only need a certain bitrate to ensure user satisfaction - they rarely fully utilize the wireless capacity. Our mobility graph in the previous section shows that, in principle, WiFi has enough capacity to support these apps, even if it is only available in stations. A simple optimization would be to download as much of the video as possible while on WiFi, and only use 3G when the playout buffer falls below a certain threshold. This would ensure a smooth viewing experience while pushing as much traffic as possible over WiFi.

The examples shown in this section are only meant to be illustrative, and their potential impact or even feasibility is not clear at this point. Further, the list is not meant to be exhaustive. We have only provided it to show that Multipath TCP benefits go beyond than increasing throughput, and could shape both the lower and the upper layers of the protocol stack in the future.

5   Minion

As the Internet has grown and evolved over the past few decades, congestion control algorithms and extensions such as MPTCP continue to reflect an evolving TCP. However, a proliferation of middleboxes such as Network Address Translators (NATs), Firewalls, and Performance Enhancing Proxies (PEPs), has arguably stretched the waist of the Internet hourglass upwards from IP to include TCP and UDP [98, 45, 86], making it increasingly difficult to deploy new transports and to use anything but TCP and UDP on the Internet.

TCP [89] was originally designed to offer applications a convenient, high-level communication abstraction with semantics emulating Unix file I/O or pipes: a reliable, ordered bytestream, through an end-to-end channel (or connection). As the Internet has evolved, however, applications needed better abstractions from the transport. We start this section by examining how TCP's role in the network has evolved from a communication *abstraction* to a communication *substrate*, why its in-order delivery model makes TCP a poor substrate, why other OS-level transports have failed to replace TCP in this role, and why UDP is inadequate as the only alternative substrate to TCP.

5.1   Rise of Application-Level Transports

The transport layer's traditional role in a network stack is to build high-level communication abstractions convenient to applications, atop the network layer's basic packet delivery service. TCP's reliable, stream-oriented design [89] exemplified this principle, by offering an inter-host communication abstraction modeled on Unix pipes, which were the standard *intra-host* communication abstraction at the time of TCP's design. The Unix tradition of implementing TCP in the OS kernel offered further convenience, allowing much application code to ignore the difference between an open disk file, an intra-host pipe, or an inter-host TCP socket.
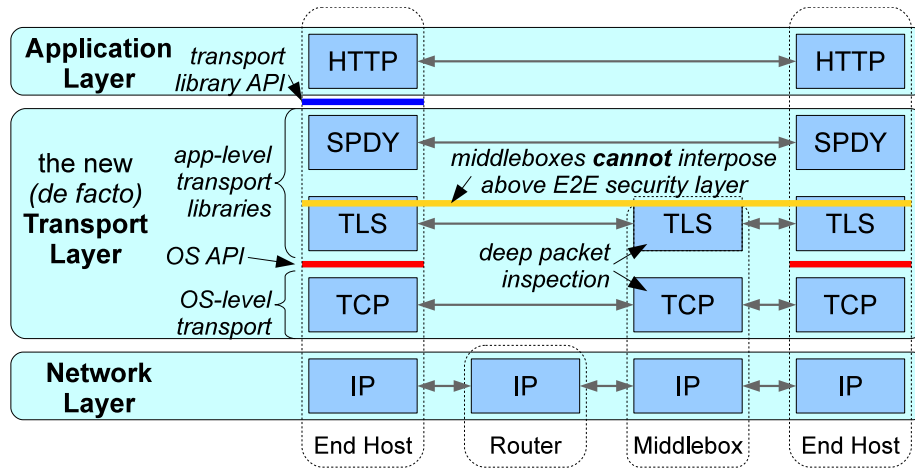
Figure 15: Today's "*de facto* transport layer" is effectively split between OS and application code. (Reprinted from [78]. Included here by permission.)

Instead of building directly atop traditional OS-level transports such as TCP or UDP, however, today's applications frequently introduce additional transport-like protocol layers at user-level, typically implemented via application-linked libraries. Examples include the ubiquitous SSL/TLS [35], media transports such as RTP [101], and experimental multi-streaming transports such as SST [44], SPDY [1], and ØMQ [2]. Applications increasingly use HTTP or HTTPS over TCP as a substrate [86]; this is also illustrated by the W3C's WebSocket interface [114], which offers general bidirectional communication between browser-based applications and Web servers atop HTTP and HTTPS.

In this increasingly common design pattern, the "transport layer" as a whole has in effect become a stack of protocols straddling the OS-application boundary. Figure 15 illustrates one example stack, representing Google's experimental Chrome browser, which inserts SPDY for multi-streaming and TLS for security at application level, atop the OS-level TCP.

One can debate whether a given application-level protocol fits some definition of "transport" functionality. The important point, however, is that *today's applications no longer need, or expect, the underlying OS to provide "convenient" communication abstractions: an application simply links in libraries, frameworks, or middleware offering the abstractions it desires.* What today's applications need from the OS is not convenience, but *an efficient substrate* atop which application-level libraries can build the desired abstractions.

5.2 TCP's Latency Tax

While TCP has proven to be a popular substrate for application-level transports, using TCP in this role converts its delivery model from a blessing into a curse. Application-level transports are just as capable as the kernel of sequencing and reassembling packets into a logical data unit or "frame" [29]. By delaying any segment's delivery to the application until all prior segments are received and delivered, however, TCP imposes a "latency tax" on all segments arriving within one round-trip time (RTT) after any single lost segment.

This latency tax is a fundamental byproduct of TCP's in-order delivery model, and is irreducible, in that an application-level transport cannot "claw back" the time a potentially useful segment has wasted in TCP's

buffers. The best the application can do is simply to *expect* higher latencies to be common. A conferencing application can use a longer jitter buffer, for example, at the cost of increasing user-perceptible lag. Network hardware advances are unlikely to address this issue, since TCP's latency tax depends on RTT, which is lower-bounded by the speed of light for long-distance communications.

5.3 Alternative OS-level Transports

All standardized OS-level transports since TCP, including UDP [87], RDP [112], DCCP [67], and SCTP [105], support out-of-order delivery. The Internet's evolution has created strong barriers against the widespread deployment of new transports other than the original TCP and UDP, however. These barriers are detailed elsewhere [98, 45, 86], but we summarize two key issues here.

First, adding or enhancing a "native" transport built atop IP involves modifying popular OSes, effectively increasing the bar for widespread deployment and making it more difficult to evolve transport functionality below the red line representing the OS API in Figure 15. Second, the Internet's original "dumb network" design, in which routers that "see" only up to the IP layer, has evolved into a "smart network" in which pervasive middleboxes perform deep packet inspection and interposition in transport and higher layers. Firewalls tend to block "anything unfamiliar" for security reasons, and Network Address Translators (NATs) rewrite the port number in the transport header, making both incapable of allowing traffic from a new transport without explicit support for that transport. Any packet content not protected by end-to-end security such as TLS—the yellow line in Figure 15—has become "fair game" for middleboxes to inspect and interpose on [95], making it more difficult to evolve transport functionality anywhere below that line.

5.4 Why Not UDP?

As the only widely-supported transport with out-of-order delivery, UDP offers a natural substrate for application-level transports. Even applications otherwise well-suited to UDP's delivery model often favor TCP as a substrate, however.

A recent study found over 70% of streaming media using TCP [50], and even latency-sensitive conferencing applications such as Skype often use TCP [12]. Firewalls can monitor the TCP state machine to help thwart network attacks [54], but determining an application session's state atop UDP requires application-specific analysis [77], creating an incentive for firewalls to block UDP in general.

In general, network middleboxes support UDP widely but not *universally*. For this reason, latency-sensitive applications seeking maximal connectivity "in the wild" often fall back to TCP when UDP connectivity fails. Skype [12] and Microsoft's DirectAccess VPN [31], for example, support UDP but can masquerade as HTTP or HTTPS streams atop TCP when required for connectivity.

TCP can offer performance advantages over UDP as well. For applications requiring congestion control, an OS-level implementation in TCP may be more timing-accurate than an application-level implementation in a UDP-based protocol, because the OS kernel can avoid the timing artifacts of system calls and process scheduling [118]. Hardware TCP offload engines can optimize common-case efficiency in end hosts [76], and performance enhancing proxies can optimize TCP throughput across diverse networks [22, 26]. Since middleboxes can track TCP's state machine, they impose much longer idle timeouts on open TCP connections—nominally two hours [49]—whereas UDP-based applications must send keepalives every two minutes to keep an idle connection open [9], draining power on mobile devices.
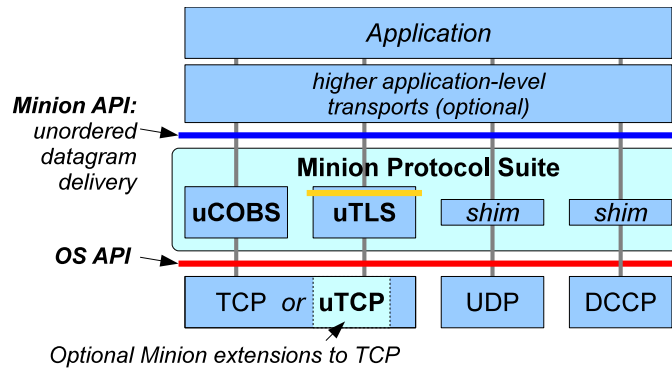
Figure 16: Minion architecture (Reprinted from [78]. Included here by permission.)

5.5 Breaking Out Of the Transport Logjam

Applications and application developers care most about services that the networking infrastructure offers to them and not how packets look on the wire; that is, they care about new transport *services*, not new transport *protocols*. On the other hand, middleboxes care most about how packets look on the wire, and generally do not care about what services are offered to the applications; that is, changing the transport protocol's bits on the wire will require changing middleboxes to respond to these changes as well.

For application developers, TCP versus UDP represents an "all-or-nothing" choice on the spectrum of services applications need. Applications desiring some but not all of TCP's services, such as congestion control but unordered delivery, must reimplement and tune all other services atop UDP or suffer TCP's performance penalties.

Without dismissing UDP's usefulness as a truly "least-common-denominator" substrate, we believe the factors above suggest that TCP will also remain a popular substrate—even for latency-sensitive applications that can benefit from out-of-order delivery—and that a deployable, backward-compatible workaround to TCP's latency tax can significantly benefit such applications.

Recognizing that the development of application-level transports and the use of TCP as a substrate under them is likely to continue and expand, we now describe *Minion*, an architecture for efficient but backward-compatible unordered delivery over extant transports, including TCP. Minion consists of *uTCP*, a small OS extension adding basic unordered delivery primitives to TCP, and two application-level protocols implementing datagram-oriented delivery services that function on either *u*TCP or unmodified TCP stacks.

While building a new transport on UDP or using alternative OS-level transports where available are both perfectly reasonable design approaches, Minion offers a uniform interface for applications to use along with an alternative option where the ubiquitous TCP protocol is adequate, but the sockets API is not; where a new transport service can be offered using TCP's bits on the wire.

5.6 Minion Architecture Overview

Minion is an architecture and protocol suite designed to provide efficient unordered delivery built atop existing transports. Minion itself offers no high-level abstractions: its goal is to serve applications and higher application-level transports, by acting as a "packhorse" carrying raw datagrams as reliably and efficiently as possible across today's diverse and change-averse Internet.

Figure 16 illustrates Minion's architecture. Applications and higher application-level transports link in and use Minion in the same way as they already use existing application-level transports such as DTLS [97], the datagram-oriented analog of SSL/TLS [35]. In contrast with DTLS's goal of layering security atop datagram transports such as UDP or DCCP, Minion's goal is to offer efficient datagram delivery atop *any* available OS-level substrate, including TCP.

Minion consists of several application-level transport protocols, together with a set of optional enhancements to end hosts' OS-level TCP implementations.

Minion's enhanced OS-level TCP stack, called *uTCP* ("unordered TCP"), includes sender- and receiver-side API features supporting unordered delivery and prioritization, as detailed later in this section. These enhancements affect only the OS API through which application-level transports such as Minion interact with the TCP stack, and make *no* changes to TCP's wire protocol.

Minion's application-level protocol suite currently consists of the following main components:

- *u**COBS** is a protocol that implements a minimal unordered datagram delivery service atop either un-modified TCP or *u*TCP, using *Consistent-Overhead Byte Stuffing*, or COBS encoding [23] to facilitate out-of-order datagram delimiting and prioritized delivery, as described later in Section 5.8.3.

- *u**TLS** is a modification of the traditionally stream-oriented TLS [35], offering a secure, unordered datagram delivery service atop TCP or *u*TCP. The wire-encoding of *u*TLS streams is designed to be indistinguishable in the network from conventional, encrypted TLS-over-TCP streams (e.g., HTTPS), offering a maximally conservative design point that makes no network-visible changes "below the yellow line" in Figure 16. Section 5.8.3 describes *u*TLS.

- Minion adds shim layers atop OS-level datagram transports, such as UDP and DCCP, to offer applications a consistent API for unordered delivery across multiple OS-level transports. Since these shims are merely wrappers for OS transports already offering unordered delivery, this paper does not discuss them in detail.

Minion currently leaves to the application the decision of *which* protocol to use for a given connection: e.g., *u*COBS or *u*TLS atop TCP/*u*TCP, or OS-level UDP or DCCP via Minion's shims. Other ongoing work explores *negotiation protocols* to explore the protocol configuration space dynamically, optimizing protocol selection and configuration for the application's needs and the network's constraints [46]. Many applications already incorporate simple negotiation schemes, however—e.g., attempting a UDP connection first and falling back to TCP if that fails—and adapting these mechanisms to engage Minion's protocols according to application-defined preferences and decision criteria should be straightforward.

5.7 Compatibility and Deployability

Minion addresses the key barriers to transport evolution, outlined in Section 5.3, by creating a backward-compatible, incrementally deployable substrate for new application-layer transports desiring unordered delivery. Minion's deployability rests on the fact that it can, when necessary, avoid relying on changes either "below the red line" in the end hosts (the OS API in Figure 16), or "below the yellow line" in the network (the end-to-end security layer in Figure 16).

While Minion's *u*COBS and *u*TLS protocols offer maximum performance benefits from out-of-order delivery when both endpoints include OS support for Minion's *u*TCP enhancements, *u*COBS and *u*TLS still function and interoperate correctly even if neither endpoint supports *u*TCP, and the application need not know or care whether the underlying OS supports *u*TCP. If only one endpoint OS supports *u*TCP,

Minion still offers incremental performance benefits, since uTCP's sender-side and receiver-side enhancements are independent. A uCOBS or uTLS connection atop a mixed TCP/uTCP endpoint-pair benefits from uTCP's sender-side enhancements for datagrams sent by the uTCP endpoint, and the connection benefits from uTCP's receiver-side enhancements for datagrams arriving at the uTCP host.

Addressing the challenge of network-compatibility with middleboxes that filter new OS-level transports and sometimes UDP, Minion offers application-level transports a continuum of substrates representing different tradeoffs between suitability to the application's needs and compatibility with the network.

An application can use unordered OS-level transports such as UDP, DCCP [67], or SCTP [105], for paths on which they operate, but Minion offers an unordered delivery alternative wire-compatible not only with TCP, but with the ubiquitous TLS-over-TCP streams on which HTTPS (and hence Web security and E-commerce) are based, likely to operate in almost any network environment purporting to offer "Internet access."

5.8 Minion's Unordered Delivery using TCP and TLS

We now briefly discuss Minion's true unordered delivery over bytestream protocols; we describe how Minion provides true unordered datagram delivery without modifying the TCP and TLS wire-formats.

Minion enhances the OS's TCP stack with API enhancements supporting unordered delivery in both TCP's send and receive paths, enabling applications to reduce transmission latency at both the sender- and receiver-side end hosts when both endpoints support uTCP. Since uTCP makes no change to TCP's wire protocol, two endpoints need not "agree" on whether to use uTCP: one endpoint gains latency benefits from uTCP even if the other endpoint does not support it. Further, an OS may choose independently whether to support the sender- and receiver-side enhancements, and when available, applications can activate them independently.

uTCP does *not* seek to offer "convenient" or "clean" unordered delivery abstractions directly at the OS API. Instead, uTCP's design is motivated by the goals of maintaining exact compatibility with TCP's existing wire-visible protocol and behavior, and facilitating deployability by minimizing the extent and complexity of changes to the OS's TCP stack.

5.8.1 uTCP: Receiver-Side Modifications

A conventional TCP receiver delivers data in-order to the receiving application, holding back any data that is received out of order. uTCP modifies the TCP receive path, enabling a receiving application to request immediate delivery of data that is received by uTCP, both in order and out of order.

uTCP makes two modifications to a conventional TCP receiver. First, whereas a conventional TCP stack delivers received data to the application only when prior gaps in the TCP sequence space are filled, the uTCP receiver makes data segments available to the application immediately upon receipt, skipping TCP's usual reordering queue. The data the uTCP stack delivers to the application in successive application reads may skip forward and backward in the transmitted byte stream, and uTCP may even deliver portions of the transmitted stream multiple times. uTCP guarantees only that the data returned by each application read corresponds to *some* contiguous sequence of bytes in the sender's transmitted stream.

Second, when servicing an application's read, the uTCP receiver also delivers the logical offset of the first returned byte in the sender's original byte stream—information that a TCP receiver must maintain to arrange received segments in order.

Figure 17 illustrates uTCP's receive-side behavior, in a simple scenario where three TCP segments arrive in succession: first an in-order segment, then an out-of-order segment, and finally a segment filling the gap
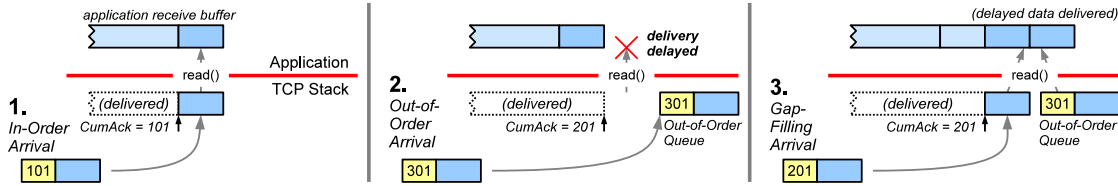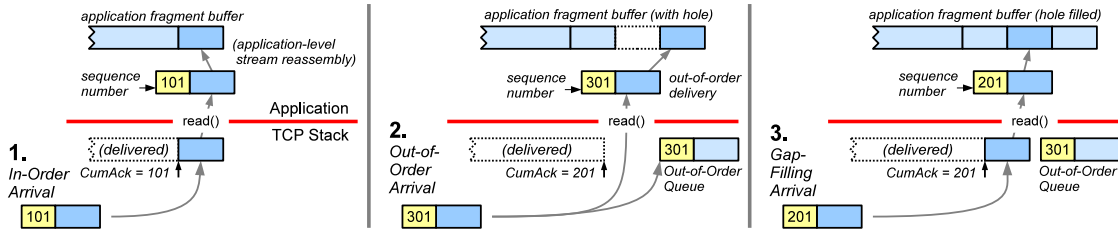
Figure 17: Delivery behavior of (a) standard TCP, and (b) uTCP, upon receipt of in-order and out-of-order segments. (Reprinted from [78]. Included here by permission.)

between the first two. With uTCP, the application receives each segment as soon as it arrives, along with the sequence number information it needs to reconstruct a complete internal view of whichever fragments of the TCP stream have arrived.

5.8.2 uTCP: Sender-Side Modifications

While uTCP's receiver-side enhancements address the "latency tax" on segments waiting in TCP's reordering buffer, TCP's sender-side queue can also introduce latency, as segments the application has already written to a TCP socket—and hence "committed" to the network—wait until TCP's flow and congestion control allow their transmission. Many applications can benefit from the ability to "late-bind" their decision on *what* to send until the last possible moment, and also from being able to transmit a message of higher priority that bypasses any lower priority messages in the sender-side queue.

A uTCP sender allows a sending application to specify a tag with each application write, which the uTCP sender currently interprets as a priority level. Instead of unconditionally placing the newly-written data at the tail of the send queue as TCP normally would, uTCP *inserts* the newly-written data into the send queue just *before* any lower-priority data in the send queue not yet transmitted.

With these modifications to a TCP stack, none of which require changes to the TCP wire-format, uTCP offers an interface which, while not convenient for applications, is powerful. In the next Section, we discuss how we build a userspace library that uses this interface that provides a simple unordered delivery service, unordered delivery of encrypted messages, and logically separate data streams within a single uTCP connection.

5.8.3 Datagrams atop uTCP

Applications built on datagram substrates such as UDP generally assume the underlying layer preserves datagram boundaries. TCP's stream-oriented semantics do not preserve any application-relevant frame bound-

aries within a stream, however. Both the TCP sender and network middleboxes can and do coalesce TCP segments or re-segment TCP streams in unpredictable ways [58].

Atop $uTCP$, a userspace library can reconstruct contiguous fragments in the received data stream using the metadata sequence number information that $uTCP$ passes along at the receiver. However, providing unordered message delivery service atop $uTCP$ requires delimiting application messages in the bytestream. While record delimiting is commonly done by application protocols such as HTTP, SIP, and many others, a key property that we require to provide a true unordered delivery service is that a receiver must be able to extract a given message independently of other messages. That is, as soon as a complete message is received, the message delimiting mechanism must allow for extraction of the message from the bytestream fragment, without relying on the receipt of earlier messages.

Minion implements self-delimiting messages in two ways:

1. To encode application datagrams efficiently, the userspace library employs *Consistent-Overhead Byte Stuffing*, or COBS [23] to delimit and extract messages. COBS is a binary encoding which eliminates *exactly* one byte value from a record's encoding with minimal bandwidth overhead. To encode an application record, COBS first scans the record for *runs* of contiguous marker-free data followed by exactly one marker byte. COBS then removes the trailing marker, instead *prepending* a non-marker byte indicating the run length. A special run-length value indicates a run of 254 bytes *not* followed by a marker in the original data, enabling COBS to divide arbitrary-length runs into 254-byte runs encoded into 255 bytes each, yielding a worst-case expansion of only 0.4%.

2. The userspace library coaxes out-of-order delivery from the *existing* TCP-oriented TLS wire format, producing an encrypted datagram substrate indistinguishable on the wire from standard TLS connections. TLS [35] already breaks its communication into *records*, encrypts and authenticates each record, and prepends a header for transmission on the underlying TCP stream. TLS was designed to decrypt records strictly in-order, however, creating challenges which the userspace library overcomes [78]. Run on port 443, our encrypted stream atop $uTCP$ is indistinguishable from HTTPS—regardless of whether the application actually uses HTTP headers, since the HTTP portion of HTTPS streams are TLS-encrypted anyway. Deployed this way, Minion effectively offers an end-to-end protected substrate in the "HTTP as the new narrow waist" philosophy [86].

5.9   Impact on Real Applications

Minion's unordered delivery service benefits a number of applications; we refer the reader to detailed discussion and experiments in [78]. Of these applications, we briefly discuss how Minion fits within ongoing efforts to develop a next-generation transport for the web, such as SPDY[1] and the Internet Engineering Task Force (IETF)'s HTTP/2.0 (httpbis) effort. Developing a next-generation HTTP requires either submitting to TCP's latency tax for backward compatibility, as with SPDY's use of TLS/TCP, or developing and deploying new transports atop UDP, neither of which, as we discussed earlier in this section, is a satisfying alternative.

Minion bridges this gap and demonstrates that it is possible to obtain unordered delivery from wire-compatible TCP and TLS streams with surprisingly small changes to TCP stacks and application-level code. These protocols offer latency-sensitive applications performance benefits comparable to UDP or DCCP, with the compatibility benefits of TCP and TLS. Without discounting the value of UDP and newer OS-level transports, Minion offers a more conservative path toward the performance benefits of unordered delivery, which we expect to be useful to applications that use TCP for a variety of pragmatic reasons.

6 Conclusion

The Transport Layer in the Internet evolved for nearly two decades, but it has been stuck for over a decade now. A proliferation of middleboxes in the Internet, devices in the network that look past the IP header, has shifted the waist of the Internet hourglass upward from IP to include UDP and TCP, the legacy workhorses of the Internet. While popular for many different reasons, middleboxes thus deviate from the Internets end-to-end design, creating large deployment black-holes—singularities where legacy transports get through, but any new transport technology or protocol fails, severely limiting transport protocol evolution. The fallout of this ossification is that new transport protocols, such as SCTP and DCCP, that were developed to offer much needed richer end-to-end services to applications, have had trouble getting deployed since they require changes to extant middleboxes.

Multipath TCP is perhaps the most significant change to TCP in the past twenty years. It allows existing TCP applications to achieve better performance and robustness over today's networks, and it has been standardized at the IETF. The Linux kernel implementation shows that these benefits can be obtained in practice. However, as with any change to TCP, the deployment bar for Multipath TCP is very high: only time will tell whether the benefits it brings will outweigh the added complexity it brings in the end-host stacks.

The design of Multipath TCP has been a lengthy, painful process that took around five years. Most of the difficulty came from the need to support existing middlebox behaviors, while offering the exact same service to applications as TCP. Although the design space seemed wide open in the beginning, in the end we were *just* able to evolve TCP this way: for many of the design choices there was only one viable option that could be used. When the next major TCP extension is designed in a network with even more middleboxes, will we, as a community, be as lucky?

A pragmatic answer to the inability to deploy new transport protocols is Minion. It allows deploying new transport services by being backward compatible with middleboxes by encapsulating new protocols inside TCP. Minion demonstrates that it is possible to obtain unordered delivery and multistreaming from wire-compatible TCP and TLS streams with surprisingly small changes to TCP stacks and application-level code. Minion offers a path toward the performance benefits of unordered delivery, which we expect to be useful to applications that use TCP for a variety of pragmatic reasons.

Early in the Internet's history, all IP packets could travel freely through the Internet, as IP was the narrow waist of the protocol stack. Eventually, apps started using UDP and TCP exclusively, and some, such as Skype, used them adaptively, probably due to security concerns in addition to the increasing proliferation of middleboxes that allowed only UDP and TCP through. As a result, UDP and TCP over IP were then perceived to constitute the new waist of the Internet. (We'll note that HTTP has also recently been suggested as the new waist [86].)

Our observation is that whatever the new waist is, middleboxes will embrace it and optimize for it: if MPTCP and/or Minion become popular, it is likely that middleboxes will be devised that understand these protocols to optimize for the most successful use-case of these protocols and to help protect any vulnerable applications using them. One immediate answer from an application would be to use the encrypted communication proposed in Minion—but actively hiding information from a network operator can potentially encourage the network operator to embed middleboxes that intercept encrypted connections, effectively mounting man-in-the-middle attacks to control traffic over their network, as is already being done in several corporate firewalls [72]. To bypass these middleboxes, new applications may encapsulate their data *even* deeper, leading to a vicious circle resembling an "arms race" for control over network use.

This "arms race" is a symptom of a fundamental tussle between end-hosts and the network: end-hosts will always want to deploy new applications and services, while the network will always want to allow and optimize only existing ones [28]. To break out of this vicious circle, we propose that end-hosts and the

network must co-operate, and that they must build cooperation into their protocols. Designing and providing protocols and incentives for this cooperation may hold the key to creating a truly evolvable transport (and Internet) architecture.

Acknowledgements

References

[1] SPDY: An Experimental Protocol For a Faster Web. `http://www.chromium.org/spdy/spdy-whitepaper`.

[2] ZeroMQ: The intelligent transport layer. `http://www.zeromq.org`.

[3] AFANASYEV, A., TILLEY, N., REIHER, P., AND KLEINROCK, L. Host-to-Host Congestion Control for TCP. *IEEE Communications Surveys & Tutorials 12*, 3 (2012), 304–342.

[4] AGACHE, A., AND RAICIU, C. GRIN: utilizing the empty half of full bisection networks. In *Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing* (Berkeley, CA, USA, 2012), HotCloud'12, USENIX Association, pp. 7–7.

[5] ALLMAN, M. Comments on selecting ephemeral ports. *SIGCOMM Comput. Commun. Rev. 39*, 2 (Mar. 2009), 13–19.

[6] ALLMAN, M., FLOYD, S., AND PARTRIDGE, C. Increasing TCP's Initial Window. RFC 3390 (Proposed Standard), Oct. 2002.

[7] ALLMAN, M., OSTERMANN, S., AND METZ, C. FTP Extensions for IPv6 and NATs. RFC 2428 (Proposed Standard), Sept. 1998.

[8] ALLMAN, M., PAXSON, V., AND BLANTON, E. TCP Congestion Control. RFC 5681 (Draft Standard), Sept. 2009.

[9] AUDET, F., AND JENNINGS, C. Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. RFC 4787 (Best Current Practice), Jan. 2007.

[10] BAKER, F. Requirements for IP Version 4 Routers. RFC 1812 (Proposed Standard), June 1995.

[11] BALAKRISHNAN, H., RAHUL, H., AND SESHAN, S. An integrated congestion management architecture for internet hosts. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication* (New York, NY, USA, 1999), SIGCOMM '99, ACM, pp. 175–187.

[12] BASET, S. A., AND SCHULZRINNE, H. An analysis of the Skype peer-to-peer Internet telephony protocol. In *IEEE INFOCOM* (Apr. 2006).

[13] BEGEN, A., WING, D., AND CAENEGEM, T. V. Port Mapping between Unicast and Multicast RTP Sessions. RFC 6284 (Proposed Standard), June 2011.

[14] BEVERLY, R., BERGER, A., HYUN, Y., AND CLAFFY, K. Understanding the efficacy of deployed internet source address validation filtering. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference* (New York, NY, USA, 2009), IMC '09, ACM, pp. 356–369.

[15] BIRRELL, A., AND NELSON, B. Implementing remote procedure calls. *ACM Trans. Comput. Syst. 2*, 1 (Feb. 1984), 39–59.

[16] BONAVENTURE, O. *Computer Networking : Principles, Protocols and Practice*. Saylor foundation, 2012. Available from http://inl.info.ucl.ac.be/cnp3.

[17] BONAVENTURE, O., HANDLEY, M., AND RAICIU, C. An Overview of Multipath TCP. *Usenix ;login: magazine 37*, 5 (Oct. 2012).

[18] BRADEN, R. Requirements for Internet Hosts - Communication Layers. RFC 1122 (INTERNET STANDARD), Oct. 1989.

[19] BRADEN, R. T/TCP – TCP Extensions for Transactions Functional Specification. RFC 1644 (Historic), July 1994.

[20] BRAKMO, L., O'MALLEY, S., AND PETERSON, L. TCP Vegas: new techniques for congestion detection and avoidance. In *Proceedings of the conference on Communications architectures, protocols and applications* (New York, NY, USA, 1994), SIGCOMM '94, ACM, pp. 24–35.

[21] BUDZISZ, L., GARCIA, J., BRUNSTROM, A., AND FERRÚS, R. A taxonomy and survey of SCTP research. *ACM Computing Surveys 44*, 4 (Aug. 2012), 1–36.

[22] CARPENTER, B., AND BRIM, S. Middleboxes: Taxonomy and Issues. RFC 3234 (Informational), Feb. 2002.

[23] CHESHIRE, S., AND BAKER, M. Consistent overhead byte stuffing. In *Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication* (New York, NY, USA, 1997), SIGCOMM '97, ACM, pp. 209–220.

[24] CHU, J. Tuning TCP Parameters for the 21st Century. Presented at IETF75, July 2009.

[25] CHU, J., DUKKIPATI, N., CHENG, Y., AND MATHIS, M. Increasing TCP's Initial Window. Internet draft, draft-ietf-tcpm-initcwnd, work in progress, February 2013.

[26] CISCO. Rate-Based Satellite Control Protocol, 2006.

[27] CLARK, D. The design philosophy of the darpa internet protocols. In *Symposium proceedings on Communications architectures and protocols* (New York, NY, USA, 1988), SIGCOMM '88, ACM, pp. 106–114.

[28] CLARK, D., WROCLAWSKI, J., SOLLINS, K., AND BRADEN, R. Tussle in cyberspace: defining tomorrow's internet. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2002), SIGCOMM '02, ACM, pp. 347–356.

[29] CLARK, D. D., AND TENNENHOUSE, D. L. Architectural considerations for a new generation of protocols. In *Proceedings of the ACM symposium on Communications architectures & protocols* (New York, NY, USA, 1990), SIGCOMM '90, ACM, pp. 200–208.

[30] D'ACUNTO, L., POUWELSE, J., AND SIPS., H. A Measurement of NAT and Firewall Characteristics in Peer-to-Peer Systems. In *Proceedings of the ASCI Conference* (2009).

[31] DAVIES, J. DirectAccess and the thin edge network. *Microsoft TechNet Magazine* (May 2009).

[32] DE VIVO, M., DE VIVO, G., KOENEKE, R., AND ISERN, G. Internet vulnerabilities related to TCP/IP and T/TCP. *SIGCOMM Comput. Commun. Rev. 29*, 1 (Jan. 1999), 81–85.

[33] DETAL, G. tracebox. http://www.tracebox.org.

[34] DHAMDHERE, A., LUCKIE, M., HUFFAKER, B., CLAFFY, K., ELMOKASHFI, A., AND ABEN, E. Measuring the deployment of IPv6: topology, routing and performance. In *Proceedings of the 2012 ACM conference on Internet measurement conference* (New York, NY, USA, 2012), IMC '12, ACM, pp. 537–550.

[35] DIERKS, T., AND RESCORLA, E. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008.

[36] DUKE, M., BRADEN, R., EDDY, W., AND BLANTON, E. A Roadmap for Transmission Control Protocol (TCP) Specification Documents. RFC 4614 (Informational), Sept. 2006.

[37] EDDY, W. TCP SYN Flooding Attacks and Common Mitigations. RFC 4987 (Informational), Aug. 2007.

[38] EGEVANG, K., AND FRANCIS, P. The IP Network Address Translator (NAT). RFC 1631 (Informational), May 1994.

[39] FABER, T., TOUCH, J., AND YUE, W. The TIME-WAIT state in TCP and its effect on busy servers. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (1999), vol. 3, IEEE, pp. 1573–1583.

[40] FALL, K., AND STEVENS, R. *TCP/IP Illustrated, Volume 1: The Protocols*, vol. 1. Addison-Wesley Professional, 2011.

[41] FERGUSON, P., AND SENIE, D. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827 (Best Current Practice), May 2000.

[42] FONSECA, R., PORTER, G., KATZ, R., SHENKER, S., AND STOICA, I. IP options are not an option. Tech. Rep. UCB/EECS-2005-24, UC Berkeley, Berkeley, CA, 2005.

[43] FORD, A., RAICIU, C., HANDLEY, M., AND BONAVENTURE, O. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824 (Experimental), Jan. 2013.

[44] FORD, B. Structured streams: a new transport abstraction. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2007), SIGCOMM '07, ACM, pp. 361–372.

[45] FORD, B., AND IYENGAR, J. Breaking up the transport logjam. In *7th Workshop on Hot Topics in Networks (HotNets-VII)* (Oct. 2008).

[46] FORD, B., AND IYENGAR, J. Efficient cross-layer negotiation. In *8th Workshop on Hot Topics in Networks (HotNets-VIII)* (Oct. 2009).

[47] GONT, F. Survey of Security Hardening Methods for Transmission Control Protocol (TCP) Implementations. Internet draft, draft-ietf-tcpm-tcp-security, work in progress, March 2012.

[48] GONT, F., AND BELLOVIN, S. Defending against Sequence Number Attacks. RFC 6528 (Proposed Standard), Feb. 2012.

[49] GUHA, S., BISWAS, K., FORD, B., SIVAKUMAR, S., AND SRISURESH, P. NAT Behavioral Requirements for TCP. RFC 5382 (Best Current Practice), Oct. 2008.

[50] GUO, L., TAN, E., CHEN, S., XIAO, Z., SPATSCHECK, O., AND ZHANG, X. Delving into internet streaming media delivery: a quality and resource utilization perspective. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement* (New York, NY, USA, 2006), IMC '06, ACM, pp. 217–230.

[51] HAIN, T. Architectural Implications of NAT. RFC 2993 (Informational), Nov. 2000.

[52] HAN, H., SHAKKOTTAI, S., HOLLOT, C., SRIKANT, R., AND TOWSLEY, D. Multi-path TCP: a joint congestion control and routing scheme to exploit path diversity in the Internet. *IEEE/ACM Trans. Networking 14*, 6 (2006).

[53] HANDLEY, M., PAXSON, V., AND KREIBICH, C. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *Proc. USENIX Security Symposium* (2001), pp. 9–9.

[54] HANDLEY, M., PAXSON, V., AND KREIBICH, C. Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-end Protocol Semantics. In *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium* (Berkeley, CA, USA, 2001), USENIX Association, pp. 9–9.

[55] HAYES, D., BUT, J., AND ARMITAGE, G. Issues with network address translation for SCTP. *SIGCOMM Comput. Commun. Rev. 39*, 1 (Dec. 2008), 23–33.

[56] HESMANS, B. Click elements to model middleboxes. https://bitbucket.org/bhesmans/click.

[57] HOLDREGE, M., AND SRISURESH, P. Protocol Complications with the IP Network Address Translator. RFC 3027 (Informational), Jan. 2001.

[58] HONDA, M., NISHIDA, Y., RAICIU, C., GREENHALGH, A., HANDLEY, M., AND TOKUDA, H. Is it still possible to extend TCP? In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference* (New York, NY, USA, 2011), IMC '11, ACM, pp. 181–194.

[59] HUITEMA, C. Multi-homed TCP. Internet draft, work in progress, 1995.

[60] IREN, S., AMER, P., AND CONRAD, P. The transport layer: tutorial and survey. *ACM Comput. Surv. 31*, 4 (Dec. 1999), 360–404.

[61] IYENGAR, J., FORD, B., AILAWADI, D., AMIN, S. O., NOWLAN, M., TIWARI, N., AND WISE, J. Minion—an All-Terrain Packet Packhorse to Jump-Start Stalled Internet Transports. In *PFLDNeT 2010* (Nov. 2010).

[62] JACOBSON, V. Congestion avoidance and control. In *Symposium proceedings on Communications architectures and protocols* (New York, NY, USA, 1988), SIGCOMM '88, ACM, pp. 314–329.

[63] JACOBSON, V., BRADEN, R., AND BORMAN, D. TCP Extensions for High Performance. RFC 1323 (Proposed Standard), May 1992.

[64] JIANG, S., GUO, D., AND CARPENTER, B. An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition. RFC 6264 (Informational), June 2011.

[65] KELLY, F., AND VOICE, T. Stability of end-to-end algorithms for joint routing and rate control. *SIGCOMM Comput. Commun. Rev. 35*, 2 (Apr. 2005), 5–12.

[66] KHALILI, R., GAST, N., POPOVIC, M., UPADHYAY, U., AND LE BOUDEC, J.-Y. MPTCP is not pareto-optimal: performance issues and a possible solution. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies* (New York, NY, USA, 2012), CoNEXT '12, ACM, pp. 1–12.

[67] KOHLER, E., HANDLEY, M., AND FLOYD, S. Datagram Congestion Control Protocol (DCCP). RFC 4340 (Proposed Standard), Mar. 2006.

[68] KOHLER, E., MORRIS, R., CHEN, B., JANNOTTI, J., AND KAASHOEK, F. The click modular router. *ACM Trans. Comput. Syst. 18*, 3 (Aug. 2000), 263–297.

[69] LARSEN, M., AND GONT, F. Recommendations for Transport-Protocol Port Randomization. RFC 6056 (Best Current Practice), Jan. 2011.

[70] LARZON, L.-A., DEGERMARK, M., PINK, S., JONSSON, L.-E., AND FAIRHURST, G. The Lightweight User Datagram Protocol (UDP-Lite). RFC 3828 (Proposed Standard), July 2004.

[71] LEECH, M., GANIS, M., LEE, Y., KURIS, R., KOBLAS, D., AND JONES, L. SOCKS Protocol Version 5. RFC 1928 (Proposed Standard), Mar. 1996.

[72] MARKO, K. Using SSL Proxies To Block Unauthorized SSL VPNs. *Processor Magazine, www.processor.com 32*, 16 (July 2010), 23.

[73] MATHIS, M., AND HEFFNER, J. Packetization Layer Path MTU Discovery. RFC 4821 (Proposed Standard), Mar. 2007.

[74] MATHIS, M., MAHDAVI, J., FLOYD, S., AND ROMANOW, A. TCP Selective Acknowledgment Options. RFC 2018 (Proposed Standard), Oct. 1996.

[75] MCLAGGAN, D. Web Cache Communication Protocol V2, Revision 1. Internet draft, draft-mclaggan-wccp-v2rev1, work in progress, August 2012.

[76] MOGUL, J. TCP offload is a dumb idea whose time has come. In *HotOS IX* (May 2003).

[77] NORTHCUTT, S., ZELTSER, L., WINTERS, S., KENT, K., AND RITCHEY, R. *Inside Network Perimeter Security*. SAMS Publishing, 2005.

[78] NOWLAN, M., TIWARI, N., IYENGAR, J., AMIN, S. O., AND FORD, B. Fitting square pegs through round pipes: Unordered delivery wire-compatible with TCP and TLS. In *NSDI* (Apr. 2012), vol. 12.

[79] ONG, L., AND YOAKUM, J. An Introduction to the Stream Control Transmission Protocol (SCTP). RFC 3286 (Informational), May 2002.

[80] PAASCH, C., DETAL, G., DUCHENE, F., RAICIU, C., AND BONAVENTURE, O. Exploring mobile/WiFi handover with multipath TCP. In *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design* (New York, NY, USA, 2012), CellNet '12, ACM, pp. 31–36.

[81] PAXSON, V., AND ALLMAN, M. Computing TCP's Retransmission Timer. RFC 2988 (Proposed Standard), Nov. 2000.

[82] PAXSON, V., ALLMAN, M., CHU, J., AND SARGENT, M. Computing TCP's Retransmission Timer. RFC 6298 (Proposed Standard), June 2011.

[83] PERREAULT, S., YAMAGATA, I., MIYAKAWA, S., NAKAGAWA, A., AND ASHIDA, H. Common Requirements for Carrier-Grade NATs (CGNs). RFC 6888 (Best Current Practice), Apr. 2013.

[84] PFEIFFER, R. Measuring TCP Congestion Windows. Linux Gazette, March 2007.

[85] PHELAN, T., FAIRHURST, G., AND PERKINS, C. DCCP-UDP: A Datagram Congestion Control Protocol UDP Encapsulation for NAT Traversal. RFC 6773 (Proposed Standard), Nov. 2012.

[86] POPA, L., GHODSI, A., AND STOICA, I. Http as the narrow waist of the future internet. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* (New York, NY, USA, 2010), Hotnets-IX, ACM, pp. 6:1–6:6.

[87] POSTEL, J. User Datagram Protocol. RFC 768 (INTERNET STANDARD), Aug. 1980.

[88] POSTEL, J. Internet Protocol. RFC 791 (INTERNET STANDARD), Sept. 1981.

[89] POSTEL, J. Transmission Control Protocol. RFC 793 (INTERNET STANDARD), Sept. 1981.

[90] POSTEL, J., AND REYNOLDS, J. File Transfer Protocol. RFC 959 (INTERNET STANDARD), Oct. 1985.

[91] RADHAKRISHNAN, S., CHENG, Y., CHU, J., JAIN, A., AND RAGHAVAN, B. Tcp fast open. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies* (New York, NY, USA, 2011), CoNEXT '11, ACM, pp. 21:1–21:12.

[92] RAICIU, C., BARRE, S., PLUNTKE, C., GREENHALGH, A., WISCHIK, D., AND HANDLEY, M. Improving datacenter performance and robustness with multipath tcp. In *Proceedings of the ACM SIGCOMM 2011 conference* (New York, NY, USA, 2011), SIGCOMM '11, ACM, pp. 266–277.

[93] RAICIU, C., HANDLEY, M., AND WISCHIK, D. Coupled Congestion Control for Multipath Transport Protocols. RFC 6356 (Experimental), Oct. 2011.

[94] RAICIU, C., PAASCH, C., BARRE, S., FORD, A., HONDA, M., DUCHENE, F., BONAVENTURE, O., AND HANDLEY, M. How hard can it be? designing and implementing a deployable multipath TCP. In *NSDI* (2012), vol. 12, pp. 29–29.

[95] REIS, C., ET AL. Detecting in-flight page changes with web tripwires. In *Symposium on Networked System Design and Implementation (NSDI)* (Apr. 2008).

[96] REKHTER, Y., MOSKOWITZ, B., KARRENBERG, D., DE GROOT, G. J., AND LEAR, E. Address Allocation for Private Internets. RFC 1918 (Best Current Practice), Feb. 1996.

[97] RESCORLA, E., AND MODADUGU, N. Datagram Transport Layer Security. RFC 4347 (Proposed Standard), Apr. 2006.

[98] ROSENBERG, J. UDP and TCP as the new waist of the Internet hourglass, Feb. 2008. Internet-Draft (Work in Progress).

[99] ROSS, K., AND KUROSE, J. *Computer networking : A top-down Approach Featuring the Internet*. Addison Wesley, 2012.

[100] SALTZER, J., REED, D., AND CLARK, D. End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS) 2*, 4 (1984), 277–288.

[101] SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (INTERNET STANDARD), July 2003.

[102] SEMKE, J., MAHDAVI, J., AND MATHIS, M. Automatic tcp buffer tuning. In *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication* (New York, NY, USA, 1998), SIGCOMM '98, ACM, pp. 315–323.

[103] SHERRY, J., HASAN, S., SCOTT, C., KRISHNAMURTHY, A., RATNASAMY, S., AND SEKAR, V. Making middleboxes someone else's problem: network processing as a cloud service. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication* (New York, NY, USA, 2012), SIGCOMM '12, ACM, pp. 13–24.

[104] SRISURESH, P., FORD, B., SIVAKUMAR, S., AND GUHA, S. NAT Behavioral Requirements for ICMP. RFC 5508 (Best Current Practice), Apr. 2009.

[105] STEWART, R. Stream Control Transmission Protocol. RFC 4960 (Proposed Standard), Sept. 2007.

[106] STEWART, R., RAMALHO, M., XIE, Q., TUEXEN, M., AND CONRAD, P. Stream Control Transmission Protocol (SCTP) Partial Reliability Extension. RFC 3758 (Proposed Standard), May 2004.

[107] STONE, J., AND PARTRIDGE, C. When the CRC and TCP checksum disagree. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (New York, NY, USA, 2000), SIGCOMM '00, ACM, pp. 309–319.

[108] STRAYER, T., DEMPSEY, B., AND WEAVER, A. *XTP: The Xpress transfer protocol*. Addison-Wesley Publishing Company, 1992.

[109] TOUCH, J. TCP Control Block Interdependence. RFC 2140 (Informational), Apr. 1997.

[110] TUEXEN, M., AND STEWART, R. UDP Encapsulation of SCTP Packets for End-Host to End-Host Communication. Internet draft, draft-ietf-tsvwg-sctp-udp-encaps, work in progress, March 2013.

[111] VASUDEVAN, V., PHANISHAYEE, A., SHAH, H., KREVAT, E., ANDERSEN, D., GANGER, G., GIBSON, G., AND MUELLER, B. Safe and effective fine-grained tcp retransmissions for datacenter communication. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication* (New York, NY, USA, 2009), SIGCOMM '09, ACM, pp. 303–314.

[112] VELTEN, D., HINDEN, R., AND SAX, J. Reliable Data Protocol. RFC 908 (Experimental), July 1984.

[113] VUTUKURU, M., BALAKRISHNAN, H., AND PAXSON, V. Efficient and Robust TCP Stream Normalization. In *IEEE Symposium on Security and Privacy* (2008), IEEE, pp. 96–110.

[114] W3C. The WebSocket API (draft), 2011. http://dev.w3.org/html5/websockets/.

[115] WANG, Z., QIAN, Z., XU, Q., MAO, Z., AND ZHANG, M. An untold story of middleboxes in cellular networks. In *Proceedings of the ACM SIGCOMM 2011 conference* (New York, NY, USA, 2011), SIGCOMM '11, ACM, pp. 374–385.

[116] WISCHIK, D., HANDLEY, M., AND BAGNULO, M. The resource pooling principle. *SIGCOMM Comput. Commun. Rev. 38*, 5 (Sept. 2008), 47–52.

[117] WISCHIK, D., RAICIU, C., GREENHALGH, A., AND HANDLEY, M. Design, implementation and evaluation of congestion control for Multipath TCP. In *Proceedings of the 8th USENIX conference on Networked systems design and implementation* (Berkeley, CA, USA, 2011), NSDI'11, USENIX Association, pp. 8–8.

[118] ZEC, M., MIKUC, M., AND ZAGAR, M. Estimating the impact of interrupt coalescing delays on steady state TCP throughput. In *SoftCOM* (2002).

[119] ZIMMERMANN, H. OSI reference model–The ISO model of architecture for open systems interconnection. *Communications, IEEE Transactions on 28*, 4 (1980), 425–432.

A Exercises

This appendix contains a few exercises on the evolution of transport protocols and their interactions with middleboxes.

A.1 Transport protocols

1. TCP provides a reliable transport service. Assuming that you control the two endpoints of a connection, how would you modify the TCP protocol to provide an unreliable service ? Explore two variants of such a transport service :

   - an unreliable bytestream where bytes can be corrupted but where there are no losses
   - an unreliable bytestream that prevents data corruption but can deliver holes in the bytestream

2. Same question as above, but for SCTP.

3. TCP provides a connection-oriented bytestream service. How would you modify TCP to support a message-oriented service. Consider two variants of this service :

   - A connection-oriented message-mode service that supports only small messages, i.e. all messages are smaller than one segment.
   - A connection-oriented message-mode service that supports any message length.

4. The large windows extension for TCP defined in [63] uses the `WScale` option to negotiate a scaling factor which is valid for the entire duration of the connection. Propose another method to transport a larger window by using a new type of option inside each segment. What are the advantages/drawbacks of this approach compared to [63] assuming that there are no middleboxes ?

5. One of the issues that limits the extensibility of TCP is the limited amount of space to encode TCP options. This limited space is particularly penalizing in the SYN segment. Explore two possible ways of reducing the consumption of precious TCP option-space

   • Define a new option that packs several related options in a smaller option. For example try to combine SACK with Timestamp and Window scale in a small option for the SYN segment.

   • Define a compression scheme that allows to pack TCP options in fewer bytes. The utilisation of this compression scheme would of course need to be negotiated during the three way handshake.

A.2 Middleboxes

Middleboxes may perform various changes and checks on the packets that they process. Testing real middleboxes can be difficult because it involves installing complex and sometimes costly devices. However, getting an understanding of the interactions between middleboxes and transport protocols can be useful for protocol designers.

A first approach to understand the impact of middleboxes on transport protocols is to emulate the interference caused by middleboxes.This can be performed by using `click` [68] elements that emulate the operation of middleboxes [56] :

   • `ChangeSeqElement` changes the sequence number in the TCP header of processed segments to model a firewall that randomises sequence numbers

   • `RemoveTCPOptionElement` selectively removes a chosen option from processed TCP segments

   • `SegSplitElement` selectively splits a TCP segment in two different segments and copies the options in one or both segments

   • `SegCoalElement` selectively coalesces consecutive segments and uses the TCP option from the first/second segment for the coalesced one

Using some of these `click` elements, perform the following tests with one TCP implementation.

1. Using a TCP implementation that supports the timestamp option defined in [63] evaluate the effect of removing this option in the SYN, SYN+ACK or regular TCP segments with the `RemoveTCPOptionElement` `click` element.

2. Using a TCP implementation that supports the selective acknowledgement option defined in [74] predict the effect randomizing the sequence number in the TCP header without updating anything in this option as done by some firewalls. Use the `ChangeSeqElement` `click` element to experimentally verify your answer. Instead of using random sequence numbers, evaluate the impact of logarithmically increasing/decreasing the sequence numbers (i.e. +10, +100, +1000, +1000, . . .)

3. Recent TCP implementations support the large windows extension defined in [63]. This extension uses the `WScale` option in the SYN and SYN+ACK segments. Evaluate the impact of removing this option in one of these segments with the `RemoveTCPOptionElement` element. For the experiments, try to force the utilisation of a large receive window by configuring your TCP stack.

4. Some middleboxes split or coalesce segments. Considering Multipath TCP, discuss the impact of splitting and coalescing segments on the correct operation of the protocol. Use the Multipath TCP implementation in the Linux kernel and the `SegCoalElement` and `SegSplitElement` `click` elements to experimentally verify your answer.

5. The extensibility of SCTP depends on the utilisation of chunks. Consider an SCTP-aware middlebox that recognizes the standard SCTP chunks but drops the new ones. Consider for example the partial-reliability extension defined in [106]. Develop a `click` element that allows to selectively remove a chunk from processed segments and evaluate experimentally its impact on SCTP.

Another way to evaluate middleboxes is to try to infer their presence in a network by sending probe packets. This is the approach used by Michio Honda and his colleagues in [58]. However, the TCPExposure software requires the utilisation of a special server and thus only allows to probe the path towards this particular server. An alternative is to use `tracebox` [33]. `tracebox` is an extension to the popular `traceroute` tool that allows to detect middleboxes on (almost) any path. `tracebox` sends TCP and UDP segments inside IP packets that have different Time-To-Live values like `traceroute`. When an IPv4 router receives an IPv4 packet whose TTL is going to expire, it returns an ICMPv4 *Time Exceeded* packet that contains the offending packet. Older routers return in the ICMP the IP header of the original packet and the first 64 bits of the payload of this packet. When the packet contains a TCP segment, these first 64 bits correspond to the source and destination ports and the sequence number. However, recent measurements show that a large fraction of IP routers in the Internet, notably in the core, comply with [10] and thus return the complete original packet. `tracebox` compares the packet returned inside the ICMP message with the original one to detect any modification performed by middleboxes. All the packets sent and received by `tracebox` are recorded as a libpcap file that can be easily processed by using `tcpdump` or `wireshark`.

1. Use `tracebox` to detect whether the TCP sequence numbers of the segments that your host sends are modified by intermediate firewalls or proxies.

2. Use `tracebox` behind a Network Address Translator to see whether `tracebox` is able to detect the modifications performed by the NAT. Try with TCP, UDP and regular IP packets to see whether the results vary with the protocol. Analyse the collected packet traces.

3. Some firewalls and middleboxes change the `MSS` option in the `SYN` segments that they process. Can you explain a possible reason for this change ? Use `tracebox` to verify whether there is a middlebox that performs this change inside your network.

4. Use `tracebox` to detect whether the middleboxes that are deployed in your network allow new TCP options, such as the ones used by Multipath TCP, to pass through.

5. Extend `tracebox` so that it supports the transmission of SCTP segments containing various types of chunks.

A.3 Multipath TCP

Although Multipath TCP is a relatively young extension to TCP, it is already possible to perform interesting experiments and simulations with it. The following resources can be useful to experiment with Multipath TCP :

- http://www.multipath-tcp.org provides an implementation of Multipath TCP in the Linux kernel with complete source code and binary packages. This implementation covers most of [43] and supports the coupled congestion control [93] and OLIA [66]. Mininet and netkit images containing the Multipath TCP kernel are available from the above website.

- http://caia.swin.edu.au/urp/newtcp/mptcp/ provides a kernel patch that enables Multipath TCP in the FreeBSD-10.x kernel. This implementation only supports a subset of [43]

- The `ns-3` network simulator[10] contains two forms of support for Multipath TCP. The first one is by using a Multipath TCP model[11]. The second is by executing a modified Linux kernel inside `ns-3` by using Direct Code Execution[12].

Most of the exercises below can be performed by using one of the above mentioned simulators or implementation.

1. Several congestion control schemes have been proposed for Multipath TCP and some of them have been implemented. Compare the performance of the congestion control algorithms that your implementation supports.

2. The Multipath TCP congestion control scheme was designed to move traffic away from congested paths. TCP detects congestion through losses. Devise an experiment using one of the above mentioned simulators/implementation to analyse the performance of Multipath TCP when losses occur.

3. The non-standard `TCP_INFO` socket option[84] in the Linux kernel allows to collect information about any active TCP connection. Develop an application that uses `TCP_INFO` to study the evolution of the Multipath TCP congestion windows.

4. Using the Multipath TCP Mininet or netkit image, experiment with Multipath TCP's fallback mechanism by using `ftp` to transfer files through a NAT that includes an application level gateway. Collect the packet trace and verify that the fallback works correctly.

---

[10]See http://www.nsnam.org/.
[11]See https://code.google.com/p/mptcp-ns3/
[12]See http://www.nsnam.org/projects/direct-code-execution/

Internet Traffic Matrices: A Primer

Paul Tune and Matthew Roughan

Abstract

The increasing demand of various services from the Internet has led to an exponential growth of Internet traffic in the last decade, and that growth is likely to continue. With this demand comes the increasing importance of network operations management, planning, provisioning and traffic engineering. A key input into these processes is the *traffic matrix*, and this is the focus of this chapter.

The traffic matrix represents the volumes of traffic from sources to destinations in a network. Here, we first explore the various issues involved in measuring and characterising these matrices. The insights obtained are used to develop models of the traffic, depending on the properties of traffic to be captured: temporal, spatial or spatio-temporal properties. The models are then used in various applications, such as the recovery of traffic matrices, network optimisation and engineering activities, anomaly detection and the synthesis of artificial traffic matrices for testing routing protocols. We conclude the chapter by summarising open questions in Internet traffic matrix research and providing a list resources useful for the researcher and practitioner.

1 Introduction

In the era of the telephone voice traffic dominated physical telecommunication lines. With the birth of the Internet, and its subsequent adoption as a key means of communication, data traffic has taken over (though some of this data traffic is actually re-badged voice traffic using Voice over IP). With the advent of new applications such as video streaming, and the rapid growth of data traffic from mobile devices, we are witnessing a global data explosion.

Given the ever increasing importance of the Internet, knowledge of its traffic has become increasingly critical. The Internet, however, is just an all-encompassing term to describe the vast global collection of networks, and so it largely falls on individual network providers to determine their own traffic. This knowledge is vital for continued operations because it allows network operators to perform important tasks such as providing enough network capacity to carry the current traffic, as well as to predict and prepare for future trends. Traffic data is also important in network maintenance, which is necessary if services and content are to be provided to customers with minimal interruption.

The focus of this chapter is on the *traffic matrix*, which, in a nutshell, is an abstract representation of the traffic volume flowing between sets of source and destination pairs. Each element in the matrix denotes the amount of traffic between a source and destination pair. There are many variants: depending on the network layer under study, sources and destinations could be routers or even whole networks. And "Amount" is generally measured in the number of bytes or packets, but could refer to other quantities such as connections.

Traffic matrices, as will be clearer below, are utilised for a variety of network engineering goals, such as prediction of future traffic trends, network optimisation, protocol design and anomaly detection. Considering

the widespread use of these matrices, the first objective of this chapter is to provide an entry level for graduate students and new researchers to current research on traffic matrices. To that end, the material is organised in a tutorial-like fashion, so as to ensure key concepts can be easily understood.

1.1 Motivation

Why study Internet traffic matrices? Simply because their implications for network operators are vast. If the traffic matrix of a network is exactly known, then, armed with topology and routing information of the network, the operator knows *exactly* what is going on in the network, rendering network management tasks relatively easy. If the traffic matrix is completely unknown, then a network operator is blind. Subtle faults may plague their network, substantially reducing performance. Congestion may be rife, or sudden shifts in traffic may cause transient traffic losses.

The issues are becoming more important. The dominant philosophy in the early days of the Internet was best effort delivery. Most applications did not have high quality of service (QoS) requirements, and had some tolerance to packet drops and link failures. In recent years, however, the landscape of the Internet is changing quickly with the introduction of streaming content such as video, high definition television and Voice over IP (VoIP), which have more stringent QoS requirements. For example, excessive packet drops and delays would produce highly noticeable artefacts in streamed video. These changes are driven primarily by user demands, with the introduction of new applications, such as online social networking, entertainment services and online multi-player gaming. Furthermore, with the increasing computational power of mobile devices and increasing wireless speeds, it is evident that a significant portion of future traffic will be generated by these devices. These trends are making measurements more and more critical.

For most operators, the state of their measurements is somewhere between extremes. Most operators have some traffic data (if only link counts). However, it is rare (in our experience) to find a network operator who knows everything about their traffic. As such, one of the tasks we shall consider here is how to measure these matrices[1], but also how to obtain estimates of traffic matrices when presented with incomplete data. The traffic matrix *inference* or *completion* or *recovery* problem is one of the major areas of research into these interesting creatures, and it is intricately related to modelling the matrices, both as measurements supply data to populate the models, and because models are used to perform the inference. Even those operators with extensive measurements and exact knowledge of today's traffic matrix may be interested in methods to predict their matrices for use in future planning, and this can be seen as another form of matrix completion.

Traffic matrices have many uses, apart from the simple fact that this type of business intelligence is critical to understanding your network. The more direct uses include network optimisation, anomaly detection and protocol design.

There are three common optimisation problems on networks. Capacity planning is needed to ensure there is adequate bandwidth for users in the present and future, but at minimal cost. There are two types of network planning: evolutionary planning and green fields planning; see §5 for details. Traffic engineering tasks include day-to-day maintenance of the network as well as predicting growth trends and anticipating traffic demands [16,45,51,52,74,75,79,91,115]. Routing involves organising traffic flow in the network, as well as modelling routing of large networks [88]. This includes functions such as finding the shortest paths for flows but also, importantly, load balancing to ensure links remain uncongested. In all these cases, the traffic matrix is a key input to perform the tasks effectively and efficiently.

---

[1]This chapter is not really a primer on measurement tools as such, so much as the principles that underlie those tools. We will not tell the reader how to set up NetFlow on the reader's particular router, but instead we will aim to inform the reader what could be achieved with this type of flow-level measurements.

Traffic matrices can also be used to detect sudden shifts in traffic due to anomalies. Anomalies include sudden unexpected events, such as network failures, or more malicious events, such as the September 11 World Trade Centre attack, worm infections and distributed denial of service (DDoS) attacks [99]. Regardless, these anomalies need to be detected so as to develop appropriate measures against possible threats to the network.

Traffic matrices may also be used to conduct reliability analyses, where the effect (on traffic) of network failures is considered. A basic task in most network design is to create redundant paths to carry traffic in case of failure, but if high reliability is required, then an operator should also ensure that there is sufficient capacity in the network to carry this traffic along its alternate paths.

Further, the performance of many network protocols depends on the traffic they carry, and the cross-traffic which forms their background. Design of new protocols therefore requires realistic measurements, or models of such traffic. Models can be used to test protocols on artificially synthesised traffic. In this way, the limitations of a protocol may be understood in a controlled environment before running it on an actual network.

These issues will be examined in-depth in §5, where algorithms utilising traffic matrices to perform these tasks will be discussed.

1.2   A primer on modelling

Motivated by a lack of understanding of Internet traffic, and as a response to the recent shifting landscape of traffic demands, various traffic models have been developed over the last decade or so. There are hundreds of papers describing data traffic modelling, however, here we shall focus on one group of such models that are particularly useful to operators: models for traffic matrices. There is no single model that captures all observed properties of once and future traffic matrices, and so here we examine the issues that go into deciding on appropriate models for particular tasks.

The ultimate goal is to use these models in various networking tasks, and the "essential" qualities that the model must capture depend critically on the task. For instance, traffic properties are highly dependent on time scale: in very short time scales (seconds), the traffic volume distributions have been shown to exhibit complex statistical behaviour such as high variability, long-range dependence, and multi-fractal behaviour. On long time scales (hours to days to weeks) traffic exhibits strong periodicities: there are distinct diurnal and weekly cycles with traffic peaks occurring around mid-day or in the evening and troughs in the early morning. This pattern correlates to user behaviour, where they access the Internet during the day for work or school and sleep in the night. There is also an observable long-term growth trend underlying the traffic volume when measured over years, corresponding to increasing global traffic demand. However, most traffic matrices are measured at some intermediate time scale, often at 5 to 15 minute intervals.

The time scale is not just a property of the measurements though. The tasks we wish to perform usually have an associated time scale as well. In capacity planning, we are often concerned with a "busy hour" – certainly peak measures over some intermediate period are important. However, anomaly detection needs to act at fine time scales: minutes, or perhaps even seconds.

We also have to consider the *planning horizon* of a task. Capacity planning may be aimed at determining the correct network design six months in advance or more, whereas traffic engineering is sometimes conducted on scales as short as a day. The planning horizon determines how far in advance we must predict traffic matrices.

We can start to see that modelling traffic matrices is a challenging task. Moreover, there are complexities layered upon complexities. The Internet is designed in terms of layers, with different protocols overlaid atop each other. Such a paradigm was adopted to ensure that each layer works independently, in theory,

of each other, although there are some overlaps between these layers in practice. The basic properties of traffic matrices will depend on the network level, and traffic can be measured between logical or physical source/destinations, or at different levels of aggregation.

Measurements of networks clearly serve as the foundation in any model development. The caveat, however, is that the measurements themselves are subject to errors and inconsistency which may lead to an incorrect model. Moreover, several hypotheses may fit a particular observation of the network, leading to several possible models explaining the same observation. To argue for the use of one model over another requires additional knowledge from new data or from domain knowledge. And when new information become available, there is a question of how to incorporate it into the model. There are a variety of possible approaches, all equally valid [7].

To further compound the problem, underlying all Internet traffic is the fact that all traffic is driven by consumer demands. Unfortunately, human behaviour is inherently difficult to model let alone understand. Furthermore, changing trends in network usage, deployment of Content Distribution Networks (CDNs), and increasing mobile traffic all have implications on traffic measurement. Although measurements could be made extremely accurate, if one overlooks the costs involved, the observations themselves may be outdated very quickly. Therefore, a complicated model based on these measurements may be accurate for today, but fail in predicting traffic demands for the next few years or so, given fluctuations in demand and unexpected changes in traffic patterns [113].

Furthermore, it is misleading to talk about a "correct" traffic matrix model. As pointed out in [7], just because a model replicates some set of properties of the observed data does not necessarily mean the model is "correct". At the least, there is the dangerous possibility of over-fitting. After all, a better fit is always achievable simply by adding more parameters to the model. Several information criteria address the over-fitting problem, for instance the Akaike information criterion (AIC) [6], Bayesian information criterion (BIC) [102], Minimum Message Length (MML) [120] or the Minimum Description Length (MDL) [95]. While these criteria are beyond the scope of our discussion, the basic principle is to choose the simplest explanation (measured in some information metric) amongst all competing explanations of the data.

It is for these reasons models should be evaluated not just on their accuracy in making predictions of particular statistics, but also their simplicity, robustness and consistency in relation to the realities of network operations. Model assumptions should "make sense" to an operator as well as be empirically tested on various datasets to understand their reliability and pitfalls, which is not to say we cannot learn new ideas and principles from measurements. We just need to keep in mind their scope of application, and the usefulness of these principles in practice may be limited. It is often preferable to have simple, robust models, in preference to precise, but fragile ones.

At a fundamental level we need to accept that models are all unrealistic in some way. A model describing the properties of a smaller scale network, such as a Local Area Network (LAN), may be unsuitable at the backbone network level. The underlying assumptions of one may not hold in the other. Models are simplifications of the glorious complexity that comprises humanity's primary means of telecommunication. We must, instead, reread the adage, by George E. P. Box: "All models are wrong, but some are useful". Some models have been more successfully used in real networks, and it is to these that we shall devote the most time here. However, we shall endeavour to cover the majority of simple models with the view that individuals should use the best model *for their application* without fear or favour.

No doubt, the murkiness and apparent self-contradictions of this discussion have left our readers no wiser, as yet. Modelling is a topic that could be discussed endlessly. It is our aim that through consideration of the qualities of various traffic matrix models, we shall not only inform about these particular models, but also bring the reader to a new understanding of modelling to make these issues a little less opaque.

1.3 Chapter outline

The theme of this chapter is to directly report on all key works in the field. No attempt will be made to provide overt commentary on what techniques or models are good or bad, as the objective of this chapter is to present comprehensive summaries of existing work. It is important to note that the techniques, algorithms and models presented here were developed as tools to suit specific applications. Hence, it is the onus of the practitioner to evaluate and decide which of these are applicable to his or her situation. Whenever possible, the strengths and weaknesses of the inference techniques and models are discussed objectively, so as to minimise the problem of a practitioner treating a particular tool as the silver hammer for all proverbial nails.

The chapter is organised as follows. In §2, an overview is provided on traffic matrices: the basic definitions and some illustrations to give a better handle on the topic. §3 discusses how data on traffic matrices are collected in practice. §4 discusses the various models proposed in literature that aims to capture the statistical properties of traffic matrices. §5 goes into a more in-depth treatment of the applications of traffic matrices, in particular, how traffic matrix models are used for inference, network optimisation applications, anomaly detection and traffic matrix synthesis. Not everything is known about these matrices, and §6 summarises some open questions and concludes by giving some thoughts on what the future holds for traffic matrix research. The chapter is concluded in §7.

2 Definitions and Notation

In this section, a formal approach to networks and traffic matrices is defined. Traffic originates from a *source* and is delivered to a *destination* (or several destinations). The traffic traverses a set of *links* between some set of sources and destinations. The links connecting these define the *topology* of the network, and the paths chosen by traffic flows determine the *routing*. Traffic may be split across multiple paths by load balancing, or may keep to a single path. Often, sources and destinations are identified with network devices such as switches or routers, but they can also refer to a location in a logical space attached to the network, for instance IP addresses of prefix blocks.

Let Ω denote the non-empty set of all sources and destinations in a network and let $|\Omega| = N$. The sources and destinations often have a physical, geographic location, and so we regard their indices as *spatial variables*, even when they are actually logical entities, such as Autonomous Systems (ASes), or cannot be directly identified as network devices, as with IP addresses.

A traffic matrix is naturally represented by a three dimensional, nonnegative hyper-matrix $\mathbf{X}(t)$, with i, j-th entry $X_{i,j}(t)$. Each entry represents the traffic volume, or demand, measured in terms of bytes or packets, from source i to destination j in time interval $[t, t + \Delta t) \subset \mathcal{T}$, the full measurement interval being denoted by $\mathcal{T}$. As an aside, a matrix representation is useful for the representation of other aspects of the network, for instance, delay, jitter, loss, bottleneck-bandwidth and distance [72], but throughout the chapter, traffic will be the focus. Whenever the context is clear, for example, when considering only the spatial structure of the matrix, the time index t is dropped. An abstract example of the traffic matrix is presented in Figure 1.

A closely related concept is the *demand matrix*, distinct from the traffic matrix because the former is *offered load*, and the latter *carried load* [47]. They may be the same, but may differ where congestion limits the carried traffic, or rate limiting is used on some traffic streams. In general we cannot measure offered traffic, only carried traffic, and so almost all empirical research has concentrated on traffic matrices, but it is important to note that many of the assumptions of traffic matrix models are actually motivated by intuition about demand matrices, and these may not apply where the two differ substantially[2].

---

[2]Many works make no distinction between demand and traffic matrices, leading to confusion. Here, we shall try to keep the two

$$X = \begin{pmatrix} x_{AA} & x_{AB} & x_{AC} & \cdots \\ x_{BA} & x_{BB} & x_{BC} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$
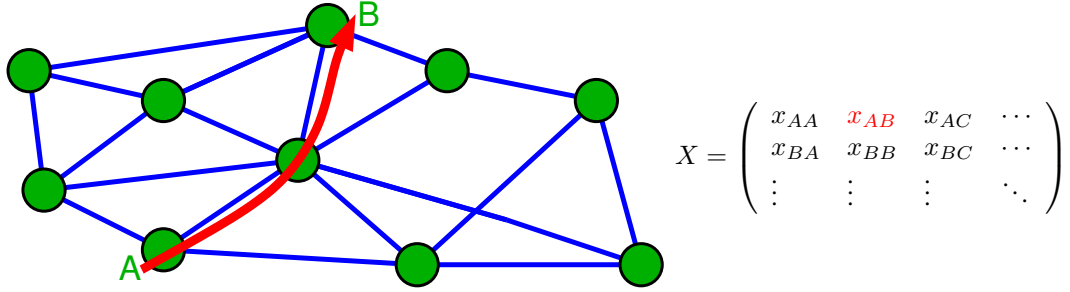
Figure 1: *An example of a traffic matrix. Note that often the diagonal elements, $x_{AA}, x_{BB}, \ldots$ are zero as this traffic does not cross the network, however, in almost as many cases it is non-zero because a "node" in the graph represents an aggregation of devices such as a PoP or an AS. In these cases we often do wish to measure these diagonals, even though they may not cross the logical links pictured, because they affect traffic engineering within the PoP, for example.*

Large-scale, real-time monitoring of traffic is intractable at present, thus limiting measurements to the average traffic in a discrete time interval. Shorter time intervals *i.e.,* small Δt, benefit anomaly detection applications, the tradeoff being a possibility of uncertainty from traffic burstiness at shorter time scales combined with larger potential measurement or sampling errors[3]. Longer time intervals result in "smoother" traffic, averaging out measurement errors. However, this smooths out real variability in the traffic as well, and can result in meaningless estimates in the presence of strong non-stationarity. Hence, the choice of Δt depends on the application and available measurements. Common choices range from 5 minutes to an hour. Further discussion on the temporal properties of traffic flows is found in §4.

There are two popular definitions of traffic matrices: the origin–destination (OD) matrix and the ingress–egress (IE) matrix.

(i) **OD traffic matrix**: this matrix measures traffic from true source to destination, *i.e.,* the point that generates a packet to the point that receives it. In the Internet, it is perhaps most reasonably defined in terms of Internet Protocol (IP) addresses. However, if Ω is defined over the entire IPv4 address space of 2^{32} addresses (with even more for IPv6), this poses storage and computational problems. Moreover, the matrix would be very sparse. What's more, protocols such as NAT, HTTP proxies and firewalls may obscure the true IP address mappings. One way to overcome some of these deficiencies is to aggregate the traffic matrix into blocks of IP addresses, frequently using routing prefixes. Bharti *et al.* [11] defined the idea of *atomic aggregation* to partially circumvent the problem of size of an OD matrix. If the logical indexing chosen is *atomic*, then a non-zero element of an OD traffic matrix implies that all flows between the particular source and destination pair is fully visible to the network operator (see [11] for details).

(ii) **IE traffic matrix**: any single network operator sees only a small proportion of the Internet OD matrix. Thus this matrix is not just unknown, but unmeasurable (by a single operator). Instead, many operators find that using their edge routers (or even the edge links) as sources and destinations results in a local traffic matrix of great use. We call this the IE traffic matrix as the set Ω includes ingress, *i.e.,* traffic going into the network, and egress, *i.e.,* traffic flowing out of the network, points as proxies for sources and destinations. A single ingress or egress "node" may denote a router, a collection of

---

distinct: when we say traffic matrix, we are referring to carried traffic.

[3]Traffic is typically sampled at the backbone network level to cope with the tremendous volumes of data that could be collected.

physically co-located routers called a Point of Presence (PoP), or some other abstract collection of traffic ingress/egress points depending on the level of coarseness required in the modelling process. The PoP level convention is often adopted as it provides a simple visualisation of the network its operators.

IE traffic matrices can be obtained in a number of ways. They can be formed from OD traffic matrices simply by mapping IP prefixes to ingress/egress locations in the network, but this assumes knowledge of all flows traversing the ingress/egress nodes. Traffic at egress nodes may be inferred from router data (see the next section) and measurements of ingress traffic, but typically, the converse is difficult. Likewise, it is usually difficult to form an OD matrix from IE matrices.

Consequently, the IE traffic matrix is frequently adopted for network optimisation applications as it is more practical to measure, and because in aggregating traffic the OD flows are "bundled" together into locally meaningful groupings. A network may carry flows between billions of IP address pairs and millions of prefix pairs, but only thousands of router pairs, and hundreds of PoP pairs. In this way, the IE matrix is a more compact representation, but more importantly, the aggregation of the traffic into large bundles results in a smoothing effect on the data, reducing the number of independent parameters that may have to be estimated. At the PoP level, the aggregation of flows results in averaging out sampling error (similar to the choice of Δt above). This is highly beneficial for numerical iterative algorithms used to estimate the traffic matrices, as aggregation leads to better conditioning of the traffic matrix. The trade-off, unfortunately, is the loss of fine grained data, as one can no longer observe IP-level flow data, or examine application profiles.

Another consideration worthy of concern in applying traffic matrices is *invariance*. A good representation of the traffic has to be invariant to other network aspects, such as routing and the network topology. For instance, if the traffic matrix for a network changes in response to changes in link placement, then the matrix is not terribly useful for network design. IE matrices are subject, for example, to large changes due to routing shifts, and this means that they are less useful to operators compared to OD matrices. However, the practicalities of measurements mean that IE matrices may be all that is measurable.

2.1 Example

As an example we present the Abilene (Internet2) data from 2004 that was used in several studies [7,96,124]. The backbone network is located in North America (see Figure 2a). The traffic data contains averages over 5 minute intervals, from March 1st to September 11th, 2004 (though there are some missing periods). The dataset is an Ingress-Egress, router-router[4] traffic matrix. Table 1 shows one example traffic matrix from that period, along with row and column sums.

We can immediately see that the matrix is not symmetric, and that there are quite a range of values. Also notable, the diagonals are not zero, as there is some measured traffic that enters Abilene, and then exits at the same PoP.

A timeseries view of the data is seen in Figure 2. The top figure shows the total traffic during each time interval for the whole dataset, and the bottom figure shows the first two weeks of the data, overlaid so as to illustrate the cyclic nature of the data. We can clearly pick out peak traffic hours, and weekdays from the weekend. However, the variation between peak and off-peak loads is perhaps smaller than is typical in commercial environments.

We have endeavoured to put this data into an easily read format, and placed it onto a web server at http://www.maths.adelaide.edu.au/matthew.roughan/traffic_matrices.html, where we will also place other TM datasets as far as possible, for comparison studies, or as test datasets for education.

---

[4]In Abilene, at the time, the routers almost corresponded to PoPs.

| | Destination | | | | | | | | | | | | |
Source	1	2	3	4	5	6	7	8	9	10	11	12	Row sum
1	0.07	0.07	0.43	0.00	0.06	0.12	0.06	0.00	0.05	0.00	0.00	0.25	1.12
2	0.00	4.09	6.42	0.06	7.07	4.42	1.59	0.02	3.24	0.03	0.16	11.09	38.18
3	0.00	4.70	25.48	4.11	13.99	11.53	3.31	87.27	5.22	0.01	0.08	7.70	163.38
4	0.00	1.93	10.25	1.68	5.63	6.11	2.59	0.01	4.11	2.60	0.04	5.92	40.88
5	0.00	4.76	0.25	0.01	24.06	0.04	0.01	0.02	1.24	0.02	0.03	18.05	48.49
6	0.00	2.87	23.73	1.55	13.53	4.78	2.89	0.01	9.45	0.08	0.50	7.64	67.02
7	0.00	0.67	4.79	1.92	3.50	2.24	1.25	0.00	0.93	0.02	0.03	3.31	18.67
8	0.00	4.18	2.58	5.80	26.35	0.17	0.16	1.41	10.88	2.11	3.64	16.67	73.97
9	0.00	8.61	12.34	5.71	18.21	11.05	3.84	0.41	36.36	0.02	0.52	17.31	114.37
10	0.00	0.18	0.04	1.71	1.69	0.00	0.06	5.61	0.96	1.82	8.44	0.36	20.86
11	0.00	3.47	3.28	0.54	8.60	0.13	0.93	3.92	1.77	0.81	0.61	2.32	26.38
12	0.00	18.20	16.04	0.83	34.03	11.18	5.64	0.09	25.57	0.08	0.80	47.02	159.47
Column sum	0.07	53.74	105.61	23.94	156.73	51.76	22.34	98.77	99.77	7.59	14.84	137.65	772.80

Table 1: An Abilene 5 minute traffic matrix from April 15th, 2004 from 16:25–16:30, in Mbps.
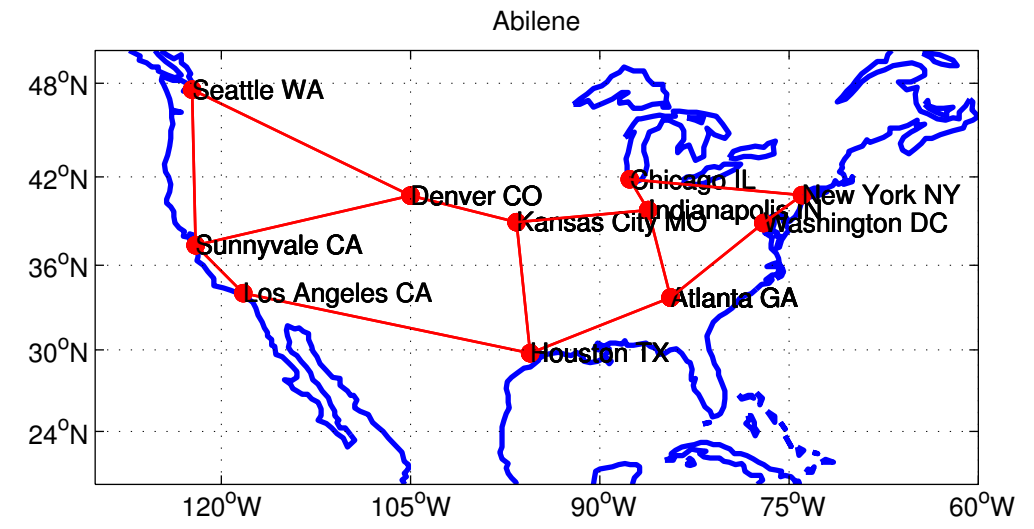
(a) *Map of Abilene PoPs.* *Table 1* *is a router to router traffic matrix, but there is one router per PoP except in Atlanta, where there is a second router. The router numbers in the table are alphabetical.*
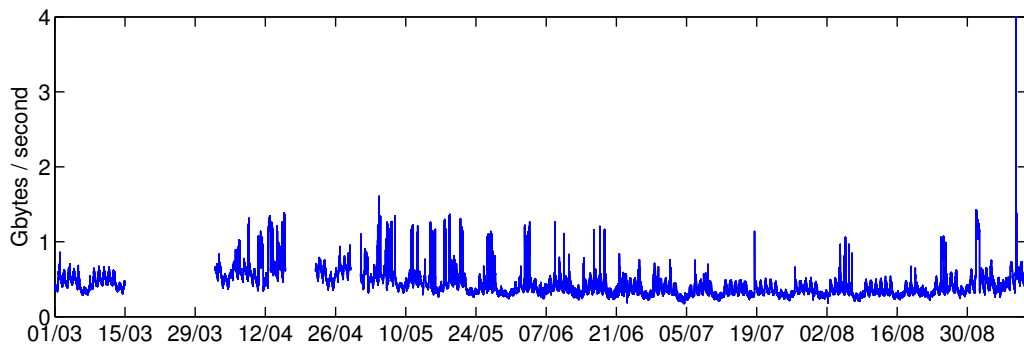
(b) *Abilene 5 minute totals of the traffic matrix from March 1st to September 11th, 2004.*
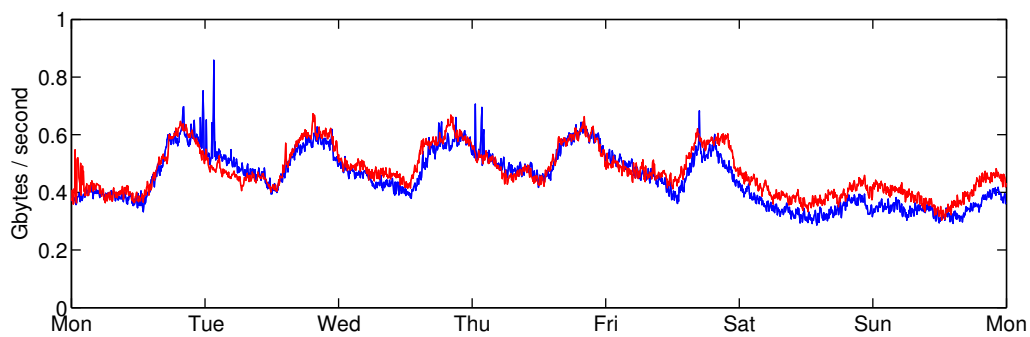
(c) *The weeks starting March 1st (blue), and March 8th (red).*

Figure 2: *Abilene, 2004*

116

The data illustrates a number of properties of the data, perhaps most notably the fact that many datasets have anomalous "spikes" (these are often real, but may also be artefacts in the data), and periods of missing data. The question of how these arise then naturally leads to our next chapter on measuring traffic matrices.

3 Measurements

In theory, it is possible to collect accurate data of Internet traffic from a network. In reality, however, many issues confound such measurements. Budget constraints driven by the cost of collection, or the massive data storage facilities required due to the sheer amount of data traversing a backbone network limit what can be achieved. And even good measurement systems can suffer from errors and missing data. Added to this, current operator practice rarely includes any significant calibration of measurement apparatus, so often the degree of accuracy of the measurements is unknown.

There are several well-known strategies for collecting traffic measurements. A *packet trace* is a collection of packets headers (perhaps with some payload) and timestamps. A packet trace can be collected through various means, for example, through hardware such as a splitter placed strategically in optical fibre; adding a monitor port on a router; or through software tools such as `tcpdump` executed on several hosts in a shared network. An OD traffic matrix can be constructed from such a trace by simple consideration of the IP address in the packet headers (with the caveats mentioned earlier).

Such an approach is ideal in many respects: we have almost complete information, and the matrices may be drawn at almost any time resolution. However, collecting packet traces is expensive due to the need for dedicated hardware, and the huge amount of storage required: potentially, over a terabyte of data per hour on OC48 links (2.5 Gbps) is needed. It is rare for any but the smallest network to be able to completely instrument its network at this level of detail.

Fortunately, constructing a traffic matrix does not require such detailed information. Perhaps the most common alternative is *flow-level aggregation* where packets are aggregated according to a common flow key. One popular definition of the key is the 5-tuple comprised of the IP source and destination address, TCP/UDP source and destination port numbers and protocol number. A series of packets possessing a common key is called a *flow*, and we maintain simple statistics (byte and packet counters, and start and stop times) for each flow. The aggregation of packets into flows reduces the number of records needed to be stored by removing redundancies of data from a packet trace. Flow-level collection is generally performed in 15 minute time bins[5] and is often an in-built function of a router. The only additional infrastructure required is the Network Management Station (NMS) and flow records themselves are usually compressed by the router before being exported to the NMS. Despite this reduction, flows arrive at a router at rapid rates and the formation and storage of flow records at a router often burdens the router's CPU.

To further reduce the number of flow records at a router, sampling methods are employed. The most popular sampling method is packet sampling, where incoming packets are sampled based on predetermined sampling patterns, used, for instance by Cisco's NetFlow [1]. Such pseudo-random patterns have a similar effect to independently picking incoming packets given sufficient mixing of traffic. The sampling rates can be adjusted depending on the capacity of the incoming links with recommendations such as 1 in 256 packets for OC192 links (10 Gbps). Higher capacity links require aggressive data reduction, and so lower sampling rates are used.

Packet sampling reduces the number of flows significantly by omitting many, but it is important to realise that although it may select *packets* in an unbiased way, it is not unbiased with respect to *flows*. Packet

---

[5]The issue of timing of flows is actually somewhat more complicated, but readers should refer to detailed descriptions of specific flow-capture protocols for information on their particular flow capture.

sampling has a strong bias towards long flows, since it is more likely to have sampled packets from a long flow than a short one. Furthermore, the sampled flow size is not the true size of the flow and there are several works [37, 38] proposing methods to sample and estimate the true size of a sampled flow. While the strong bias may be a problem for some applications, there is usually no problem in using sampled flows to form the IE traffic matrix.

In addition to packet sampling, we may also sample a set of flows, and these sampled flows can then be used to create traffic matrices. The resulting reduction in intermediate storage and processing can be substantial, particularly if the sampling is done in a clever way [37, 38].

It must be remembered that sampling is an inherently lossy process, regardless of the underlying sampling method used. The loss of information translates into errors or noise in the data. The size of these errors should, in best practice, be estimated for a given setup, but most operators do not undertake such procedures due to the difficulties involved in obtaining ground-truth data with which to compare to the sampled data. In many cases it is simply assumed that these errors, once the data is aggregated further into traffic matrices, are negligible, but this assumption is rarely validated.

Furthermore, there is the problem of possible multiple counting of traffic flows from the aggregated records of flows. The problem arises if the aggregated flow records come from internal backbone routers of the network, since a single flow may be recorded more than once by several routers if it traverses several links. One way to get around this relies on the placement of the measurement points. One simply performs traffic measurements at the ingress points. Specifically, the measurements are performed on the backbone routers connected to the access routers, or on dedicated devices placed at the links connecting the access routers to the backbone routers. In this way, the total incoming traffic of the network may be reliably measured from the flow records. There is then no longer a problem of multiple counting of flows.

Trajectory sampling [39, 40] is another method. It exploits the pseudorandomness of a hash function to simulate random sampling of packets. However, if a flow was sampled at one router, it will be consistently sampled at every other router the flow traverses by the deterministic operation of hash functions. Since the tracking of flows is now feasible, it is then also straightforward to disambiguate their identities and eliminate multiple counting. The downside is that a network operator must deploy trajectory sampling throughout the whole network, and configure the hash function on each router each time before a measurement interval begins.

A less costly alternative is easily obtainable link counts. A link count, or link load measurement, gives the volume (in bytes or packets) of traffic on a link during a particular time interval. Link counts are obtainable from measurement data by the Simple Network Management Protocol (SNMP) [25], defined as part of the IETF standard and present on many Internet devices, including most routers. SNMP data from a single router provides two measurements for each interface, the incoming and outgoing byte counts. SNMP data is obtained by an NMS by periodically polling requests through an interface, typically UDP port 161. The polling period varies from 1 minute to several minutes, but the default seems to be 5 minutes. SNMP data is highly susceptible to error, due to the following factors:

(i) **missing link observations**: data is transmitted via unreliable UDP packets and there may be errors when copying the data into the observer's archive,

(ii) **incorrect link observations**: poor router vendor implementations causes inaccuracies, and

(iii) **sampling coarseness**: polling times are often inaccurate either due to poor NMS or SNMP agent implementations, high loads, or network delays.

As with flow-level data, link count data should be calibrated, but rarely is. There is only one experiment of which we are aware that does so [97]. The study showed that in one network, SNMP errors were typically

low (90% of measurements had an error of less than 1%), but a small number of measurements were very large, some as large as 100%. This type of heavy-tailed distribution causes problems for some estimation approaches and should be considered in context.

Another drawback of SNMP data is that it only provides aggregate link statistics, omitting details such as types of traffic on the link and the traffic source and destination. Despite all these issues, SNMP data is, at present, the easiest and perhaps most common way to obtain large-scale traffic data.

The observed link counts provide some information about the traffic matrix, but only in an indirect manner. Thus, the traffic matrix has to be inferred through a process called *tomography*. Network tomography was first introduced by Vardi [117], with the inspiration taken from inference techniques for medical tomography as both problems are similar in nature. Vardi's work was subsequently expanded upon by Tebaldi and West [111] and Cao *et al.* [24]. Various other works in network tomography measure other properties of the network, such as link delays (see [26, 30] for an overview) via active packet probing, but for traffic matrix estimation, we are only concerned with the link count observations from SNMP data.

Mapping traffic to links requires topological and routing data in the form of a *routing matrix*. The routing matrix $\mathbf{A}$ is defined by

$$A_{i,r} = \begin{cases} F_{i,r}, & \text{if traffic for } r \text{ traverses link } i, \\ 0, & \text{otherwise.} \end{cases}$$

where $F_{i,r} \in (0, 1]$ is the fraction of traffic from source-destination pair $r = (s, d)$ traversing link i. Fractional values occur in cases when load balancing is used, but it has often been assumed that $F_{i,r} = 1$, resulting in $A_{i,r} \in \{0, 1\}$. The size of the routing matrix depends on the network: with N nodes and L links, the routing matrix has size $L \times N(N-1)$ (traffic from a node is assumed not rerouted to itself).

Information on the routing matrix can be obtained from several different sources (router configuration files, traceroutes, or from the routing protocols themselves), but the collection of such information is not the topic of the chapter. The interested reader is referred to [46, 47].

A common assumption is that the routing matrix remains stable during the measurement interval, thus the temporal dependence is dropped, *i.e.,* $\mathbf{A}(t) = \mathbf{A}$ for all $t \in \mathcal{T}$. However, changes in routing may occur if there are link or router failures, necessitating traffic reroutes. Generally, it is assumed the measurement interval is chosen over a period of time (minutes to hours) when $\mathbf{A}$ is stable enough to be considered static, justified by observations in [85], however in at least one case it is proposed that the changes be created, and exploited [107] for traffic matrix inference.

All the link counts may be grouped into an $L \times 1$ vector $\mathbf{y}$. Then, based on link observations in one measurement interval, the SNMP link counts may be expressed as

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \tag{1}$$

where $\mathbf{x}$ is the $N(N-1) \times 1$ vectorised version of the traffic matrix $\mathbf{X}$, with its columns stacked upon one another.

Figure 3 presents an example of (1). It shows how the traffic on a single link y_1, is built from the sum of traffic routed across the link $x_1 + x_2$. We can see that by stacking each of these equations we would get

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \tag{2}$$

which, written in matrix notation, is just (1). Note that in this case the routing matrix $\mathbf{A}$ is invertible, so the problem of inferring the traffic matrix from link measurements is easy, but this is rarely the case. Usually, L is usually much smaller than $N(N-1)$, and so the problem is highly underconstrained.

(a) *Link labels* **(b)** *Traffic labels and routes*
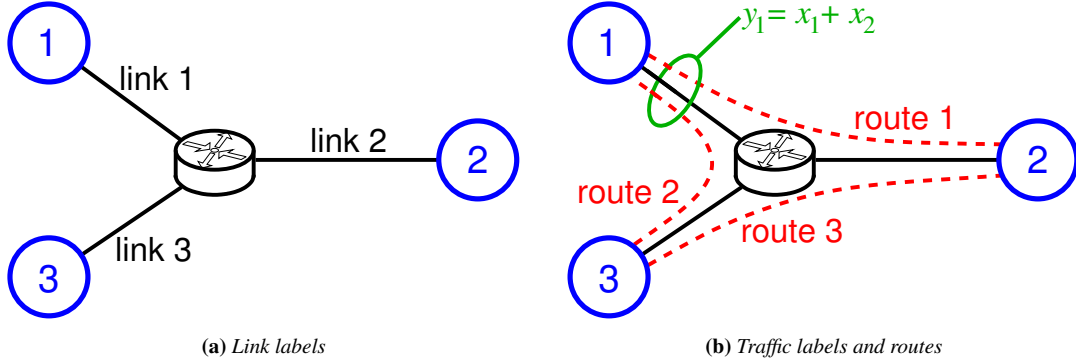
Figure 3: *A simplified network and traffic (the main simplifications are that we only consider a single router, and only consider unidirectional traffic, not bidirectional as in real networks).*

There are two main assumptions implicit in this observation model. It assumes the traffic matrix is stationary, *i.e.,* its statistical properties remain stable throughout the measurement interval and there are no errors in the measurement. Stationarity is preserved by choosing an appropriate measurement interval, say 1 hour (see §4). Moreover, in reality, errors do occur and to account for it, the model (1) is extended to

$$\mathbf{y} = \mathbf{Ax} + \mathbf{z}. \tag{3}$$

The second model is a simple noise additive model often used to test the robustness of an inference technique. Each element of the additive noise term $\mathbf{z}$ is typically chosen to be independent and identically distributed (i.i.d.) white Gaussian noise with mean zero and variance σ^2. Often σ^2 is kept small, as large values would result in some elements of $\mathbf{y}$ violating the non-negativity constraint. It is for this reason other distributions may be used, for example, log-normal or gamma distributions. Additionally, due to the problem of missing link information due to poor SNMP implementation, some of the elements of $\mathbf{y}$ may be missing. Finally, if the given routing matrix $\mathbf{A}$ is incorrect, the observations $\mathbf{y}$ would significantly depart from the true SNMP link counts. However, most works assume an accurate $\mathbf{A}$ because there are reliable methods for obtaining routing information [104].

There are usually many less links than the total number of IE pairs, and so the inverse problem above is highly underconstrained. Whether noise is present or not there may be an infinite number of solutions $\mathbf{x}$ that fit the observations (1). Figure 4 shows a picture of such a network, where we only measure at the bottleneck. Now, even in this very simple network the equations $y_1 = x_1 + x_2$ are underconstrained. In order to make progress, some additional information needs to be assumed, usually in the form of a traffic matrix model, and we shall consider some of these in the following section.

Before moving on to the modelling aspects of traffic matrices, note that there are other strategies for measurement. For instance, if MPLS is being used, this creates a set of tables (in many implementations) that can almost be read directly to obtain the matrix (*e.g.,* see [12]). Alternatively, the network operator may have more accurate local traffic matrices, obtained through specific functionality at the routers. It is, in principle, easy for a router to keep counts of its decisions [118], essentially amounting to a table of the volumes of traffic between pairs of interfaces. Locality here is defined in the sense of the matrix's restriction to a single router – we essentially see an IE traffic matrix of the single router's interface. These local matrices from all routers in the network can be used to improve the estimation of the IE traffic matrix; see §5. On some special cases, such as a star network, a single local matrix would be equivalent to the traffic matrix,
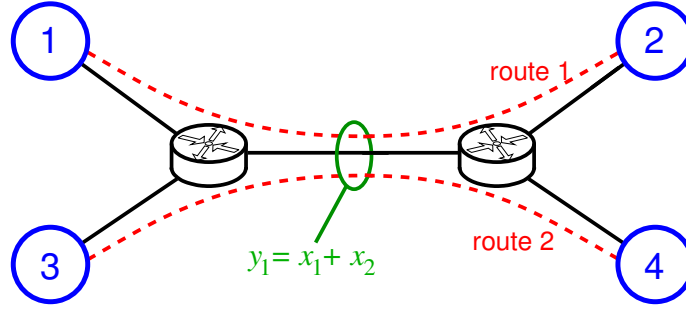
Figure 4: *A harder inference problem where we only have one measurement, but two traffic elements to estimate. There is therefore a fundamental ambiguity in this problem.*

serving to highlight the information gain from local traffic matrices. These matrices provide greater than a two-fold increase in accuracy of the tomography estimation schemes by [73, 125].

4 Models

In this section, several canonical as well as recently proposed models are presented. Modelling is divided into three categories: purely temporal modelling, spatial modelling and spatio-temporal modelling.

4.1 Temporal Modelling

Purely temporal models only focus on the time series properties of the traffic matrix. A key application of these models is in anomaly detection, so as to be able to pinpoint the time and location of the anomalous event. This is especially vital in detecting attacks on the network or worm outbreaks. However, temporal behaviour is also important in prediction, say for planning capacity of a future network.

Before delving into the models, some basic issues regarding the temporal properties of OD flows need to be understood. It is commonly agreed that IP data traffic is rising exponentially, and has been for more than a decade [2, 29, 55, 82]. There was much early controversy about such growth estimates, because the growth rate was vastly overestimated based on a small sample of data. However, these days, exponential growth is considered the common case (though with a much lower rate of growth), and is justified both by data, and as a consequence of increasing computational and networking speeds, governed by Moore's and Gilder's laws respectively. As of the present, the introduction of mobile devices and the rapid growth of traffic from these devices are set to further increase data traffic in the years to come. Cisco (who admittedly have a vested interest in a high forecast) estimate that "Annual global IP traffic will surpass the zettabyte threshold (1.3 zettabytes) by the end of 2016" [2].

Relatively few countries appear to monitor their national traffic, but Australia is an exception. The Australian Bureau of Statistics have collected and published traffic statistics for many years [3]. Figure 5 shows the growth of traffic in Australia from 2000 to 2012.

Regardless of your belief about the rate of growth, and/or the best model (exponential is common, but linear and logistic models may also be appropriate in many cases), any model of long-term traffic needs to be able to incorporate such growth.

Second, most network traffic is human generated. Therefore, it stands to reason that traffic is influenced by human activity in a 24 hour cycle. In fact, distinct diurnal patterns have been observed, with peak traffic occurring around mid-day or in the evening and dips during the night, for example see Figure 2c. This correlates to the daily schedule of an average human being, where mid-day traffic is generated for work or school purposes, while the lack of traffic during the night correlates to sleeping periods. Peak traffic rate is also noticeably less on the weekends. The regularity of this behaviour can be quite strong as shown in the figure where two successive weeks' data are overlaid so one can see how closely they match. Most traffic measurements see some measure of cyclic behaviour, with the degree of regularity determined by the type of traffic (is it made up of many individuals or a few large flows) and the scale.

Third, and leading from the last point, the traffic volume itself is dependent on the measurement period and the aggregation level of traffic. At very short scales, in milli-seconds or seconds, the traffic distribution is highly variable, and shows strong dependencies, making use of such measures statistically non-trivial, even if such measurements were easy to collect. A common paradigm is to consider a measurement interval of minutes to an hour, where measurement is easy. Also important is the fact that at these times scales *stationarity*[6] of the traffic volume distribution is often a reasonable assumption, though we can see the limits of this in Figure 2c (stationarity clearly doesn't hold for more than a couple of hours), however it was shown that *cyclo-stationarity* holds to a large extent [106].

Fourth, there are natural variations in traffic over time, and these are often modelled as a random (or stochastic) process. This random process could have all sorts of features, but there are some basics that

---

[6]Stationarity refers to the concept that the statistics of the traffic (for instance, the mean and variance, but in general including all statistics) are constant with respect to the time at which they are measured. Wide-sense stationarity is when only the mean and variance of the traffic satisfies the stationarity property. In Internet traffic data it is only ever approximately true. Moreover, it is hard to test for stationarity when traffic has long-term correlations, and so we can only ever talk about the degree of stationarity.
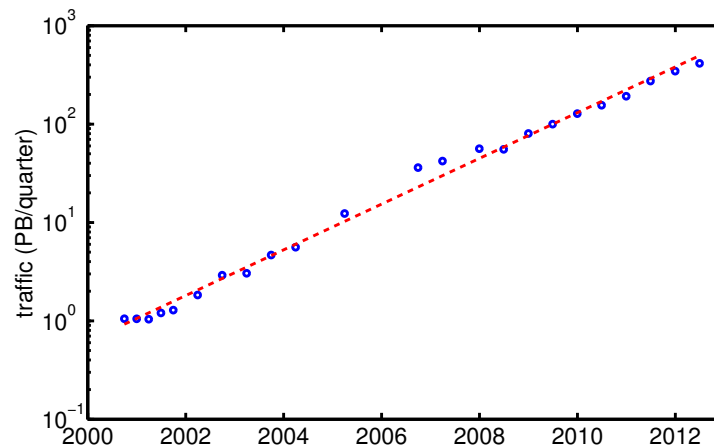
Figure 5: *Australian Internet traffic volumes based on data from the Australian Bureau of Statistics from 2000-2012. The dashed line shows a linear fit to the (log) data. Note the log y-axis, so the plot shows quite a reasonable fit to exponential growth with a doubling period of 465 days. Over the same period the growth in broadband subscribers has been almost exactly linear (soon this trend must decrease as a large proportion of Australia's population are now connected), so most of the growth has come from growth in the amount downloaded per customer.*

should be observed by all reasonable models. For instance, network operators aggregate traffic from multiple sources, which is known as *multiplexing*. Multiplexing is used to boost the efficiency of the links in a network by "smoothing" out variations in traffic. The apparent smoothness is a result of decreases in the relative variance, as predicted by the central limit theorem [23]. The more OD flows multiplexed on a link, the higher the efficiency and smoothing effect, provided the aggregated bandwidth does not exceed link capacity. Thus, any model for the large traffic rates in a network must be consistent under multiplexing, for example, when the number of flows being multiplexed is increased the relative variance should decrease in a predictable manner. Furthermore, the statistical properties of the aggregated traffic must also be consistent with the statistical assumptions of the traffic from a single user.

Finally, although rare, sometimes there may be sudden "spikes" in traffic. Such a component may arise from unusual traffic behaviour, such as DDoS attacks, worm propagation or Border Gateway Protocol (BGP) routing instability from misconfiguration. Flash crowds are also an example of this behaviour, which happens when there is a significant jump in the number of clients to a particular web server or CDN. Extreme unforeseen events, such as the September 11 attacks on the World Trade Centre in 2001 may instead cause a significant drop of traffic rates. In any case, a massive shift in traffic rates would be of interest to a network operator.

One temporal model of OD flows traversing backbone routers was proposed by Roughan *et al.* [99] by generalising the Norros model [78], originally used for modelling LAN traffic. Each OD flow is assumed to be generated from an independent source. The model is characterised by the following components (at time t):

(i) $L(t)$, the long term traffic trend,

(ii) $S(t)$, the seasonal (cyclical) component,

(iii) $W(t)$, random fluctuations, and

(iv) $I(t)$, the anomaly component.

These components correspond to the observations of traffic described earlier.

The long term trend, $L(t)$, depends on the observed underlying traffic growth in the data. This factor captures the overall growth of traffic over a long time period. An exponential growth model for instance, could be found by fitting $L(t) = A\exp(ct)$ to the data, with the parameters A and c easily estimated via log-linear regression as in Figure 5.

The component $S(t)$ may be any periodic function, such that $S(t+kT_s) = S(t)$ for all integers k, where the period T_s would typically be 24 hours, or one week.

The component $W(t)$ is assumed to be a stochastic process with zero mean and unit variance, to model the small fluctuating component of observed traffic. Finally, $I(t)$ captures the large variability of traffic from anomalies, say, a massive upsurge or downsurge in traffic. These events were captured in an individual component to separate their influence from traffic under the network's normal operating conditions.

Let $x(t)$ denote the volume of an OD flow at time t. The model takes the following form,

$$x(t) = m(t) + \sqrt{am(t)}W(t) + I(t), \tag{4}$$

where $m(t) = S(t)\,L(t)$ is the mean of the OD flow, assumed to be a product of the seasonal and long term trends, and a is the *peakedness* of the traffic. The average is modelled in such a way because as large OD flows have a larger range of variation in the size of their cycles. The parameter a controls the smoothness of the OD flow's volume in a way that is consistent given multiplexing of aggregated flows.

Figure 6 shows the decomposition in action on the set of Abilene data shown in Figure 2c, extending the analysis beyond into the section of missing data. The estimated long-term trend $\hat{L}(t)$ is not shown as it is almost constant over this period, but we show the estimated mean $\hat{m}(t) = \hat{S}(t)\,\hat{L}(t)$ (derived using a 5 term seasonal moving average, further smoothed with a 13 term moving average), shown in green. Normal bounds on this, obtained by modelling the random fluctuations $W(t)$ as a Gaussian process are shown as dashed lines, and where the traffic falls outside these bounds, we have indicated an anomaly.

We have deliberately looked at this period, which is followed by period of missing data to illustrate one of the advantages of this approach, which is the ability to fill in missing gaps in the data, though more sophisticated approaches to solve that problem also exist, *e.g.,* see [128].
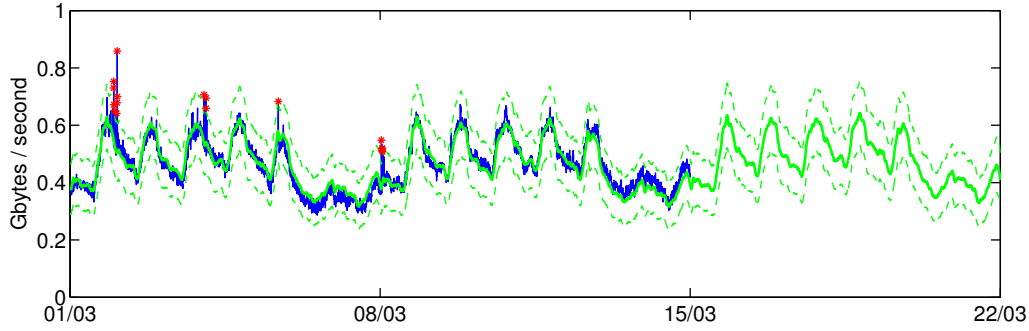
Figure 6: *Results of decomposing traffic into components. The blue curve shows the original Abilene traffic from Figure 2c (note that the data is missing for the third week). The solid green line shows $\hat{m}(t) = \hat{S}(t)\,\hat{L}(t)$, the estimated mean, and the dashed green lines show the bounds of normal variation around this mean. The red asterisks indicate the anomalies $I(t)$.*

Another feature of this model is the preservation of the properties of the model through a linear combination, advantageous when looking at the aggregated behaviour of the OD flows. Consider K aggregated OD flows, then

$$x_{\mathrm{agg}}(t) = \sum_{i=1}^{K} m_i(t) + \sum_{i=1}^{K} \sqrt{a_i m_i(t)}\, W_i(t) + \sum_{i=1}^{K} I_i(t).$$

The mean of $x_{\mathrm{agg}}(t)$ is simply $m_{\mathrm{agg}}(t) = \sum_{i=1}^{K} m_i(t)$, and the peakedness is the weighted average of the component peakedness, $a_{\mathrm{agg}} = \frac{1}{m_{\mathrm{agg}}(t)} \sum_{i=1}^{K} a_i m_i(t)$. The linearity properties allow $x_{\mathrm{agg}}(t)$ to be expressed in the same form as (4) with the new parameters $m_{\mathrm{agg}}(t)$ and a_{agg}. The linearity property enables the consistent computation of the variances of the aggregated traffic, which is useful for network planning and analysis. Besides this, [99] demonstrated the ease of estimating the parameters of model (4), via simple estimators and filtering.

The cyclical nature of the aggregated OD flows is also amenable to Fourier analysis. The Fourier transform decomposes a periodic signal into a weighted sum of sinusoids with distinct frequencies and phases. It would be reasonable to assume the observed cycles can be represented by a small number of Fourier coefficients (this does not require that the signal be exactly sinusoidal as any periodic signal can be approximated by a limited number of sinusoids). Indeed, it has been demonstrated this is the case with the traffic volumes [41, 106], where as little as just 5 Fourier coefficients were needed to achieve low error in fitting large

OD flows of a Tier 1 network, demonstrating the relatively few significant frequencies present in a diurnal cycle of the OD flows.

Figure 7 shows a similar analysis of Abilene data from Figure 2c. It shows a simple example of the excellent degree of approximation to traffic we can obtain using only a very small number of Fourier coefficients corresponding to daily periods. Figure 7a shows the periodogram of the data (the absolute magnitude of the Discrete Fourier Transform (DFT)). We can see that only a few peaks (with frequencies of 1, 2, 7, and 14 cycles per week, corresponding to daily and weekly cycles and their harmonics) are large enough to matter for gross features. Figure 7b shows the cumulative power contained in the largest of the Fourier coefficients, clearly showing that a small number of these represent the power of the signal well. Figure 7c shows an approximation of the original signal using only 10 coefficients. We can see that the cyclic components of the data are represented well, though the actual signal varies around that noisily. Including additional components provides a better approximation (in the sense of fitting the data more accurately), but we can see that we are simply recreating the noise. There is a clear tradeoff here between noise and signal, with no absolute standard for the correct choice, but the underlying promise of Fourier analysis is obvious.

The choice of time interval to use in Fourier analysis/approximation is interesting. A longer interval provides more data, and hence better estimates if the data is truly *cyclostationary*[7]. However, as we discussed earlier, there are noticeable trends in the overall volume, and so it is reasonable to assume that there will sometimes be significant shifts in the pattern (with respect to time of day or time of week). In these cases, extending the length of the dataset can confuse the statistical variability with non-stationary effects, resulting in biased estimates. The best tradeoff appears to depend on the dataset, but periods of perhaps a month seem to work reasonably well for estimating weekly cycles.

However, there are alternative tools for such analysis, designed to provide some flexibility in the tradeoff between time and frequency. The simplest is perhaps the Short-Time Fourier Transform, in which the signal is *windowed*, and the standard DFT is applied to the windows of data to produce a *spectrogram*. The technique has been applied in Figure 8 to a longer (6 weeks) sequence of the Abilene data. This segment of the data is more challenging, as it contains many anomalies, *e.g.,* the frequent spikes in the traffic, or the drop in traffic on the Independence Day holiday, observed on July 5th in 2004[8]. Figure 8a shows the set of data under consideration, along with its approximation using 10 coefficients. Figure 8b shows a spectrogram with windows of length 1 week, and here we can still clearly see the weekly and daily cycles appearing in most of the windows. Figure 8c shows a spectrogram using 1 day windows, with overlapping windows to smooth the results. The resulting picture is poor at showing the cyclical behaviour of the traffic, but clearly highlights the large anomalous spikes in the traffic in second week of the data.

An even more powerful set of techniques that have been applied to the temporal analysis of traffic are the Wavelet transforms [10, 83, 119], which provide a more flexible set of time/frequency tradeoffs. Wavelets have also been applied to spatial analysis of traffic matrices [31, 34, 93, 121, 123], which we will consider in a moment. However, wavelets are a relatively complicated set of techniques, and it is outside the scope of this chapter to provide an introduction to that material. See for instance [69] for more information.

Principal components analysis (PCA) has also been employed to quantify the temporal correlations of the traffic matrix. If $\mathcal{X}$ is a matrix where the rows represent a measurement (for instance an OD flow) and columns represented traffic volumes at time t, then temporal PCA decomposes the matrix $\mathcal{X}^{\mathrm{T}}\mathcal{X}$ into its corresponding components of eigenvalues and eigenvectors. Often, each column is *centred*, simply by subtracting the mean vector $\bar{\mathrm{x}}$, the average of all columns, from each column in $\mathcal{X}$. In what follows, $\mathcal{X}$ is

---

[7]A cyclostationary process can be thought of as one whose component processes formed from times embedded at multiples of the fundamental period from stationary sequences.
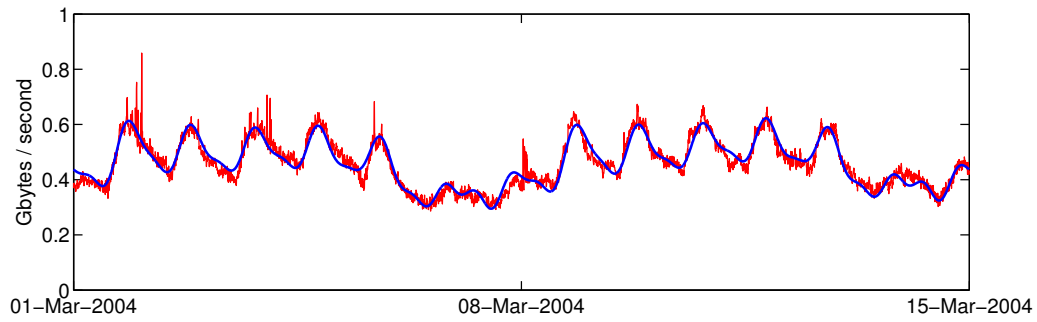
[8]Independence Day is actually celebrated on July 4th, but in 2004, the day falls on a Sunday. Thus, the following day was declared a public holiday as well.
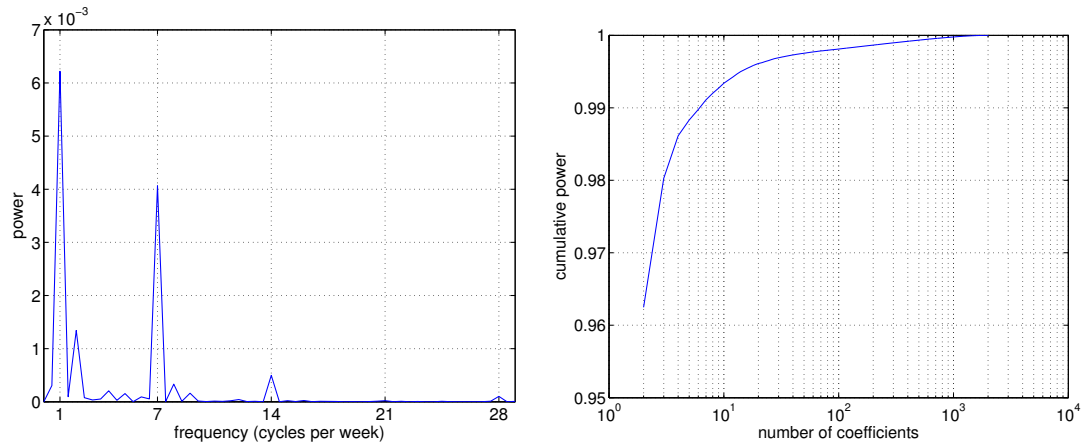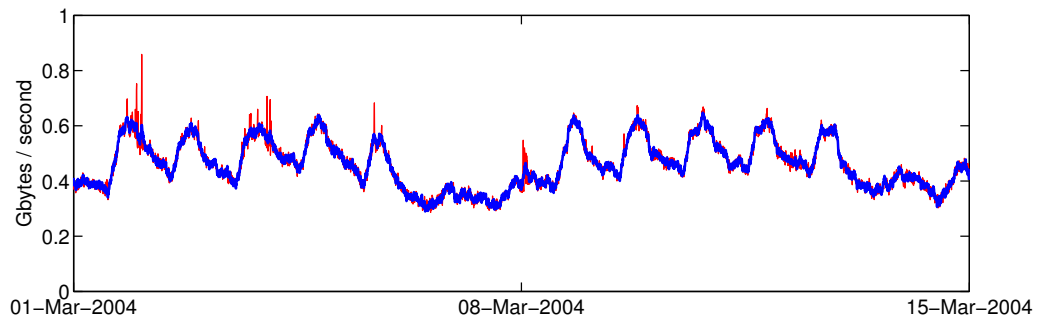
(a) *Fourier spectra (technically a periodogram) of the Abilene data.*

(b) *Cumulative power captured by Fourier components (sorted in descending order).*
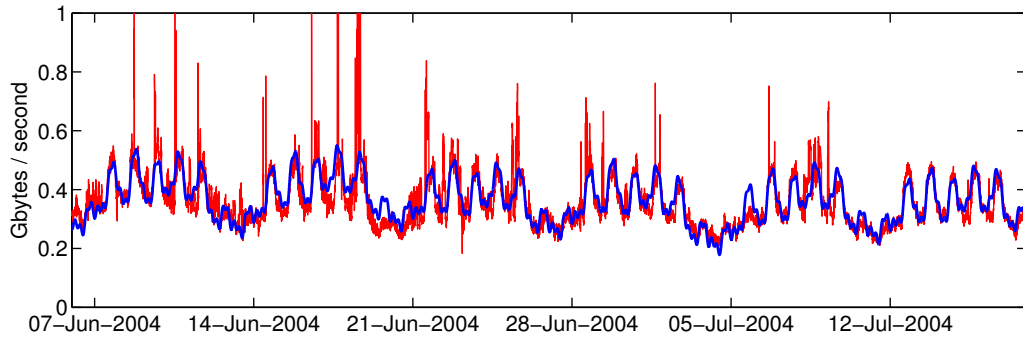

(c) *Fourier approximation to the data using 10 (complex) coefficients.*
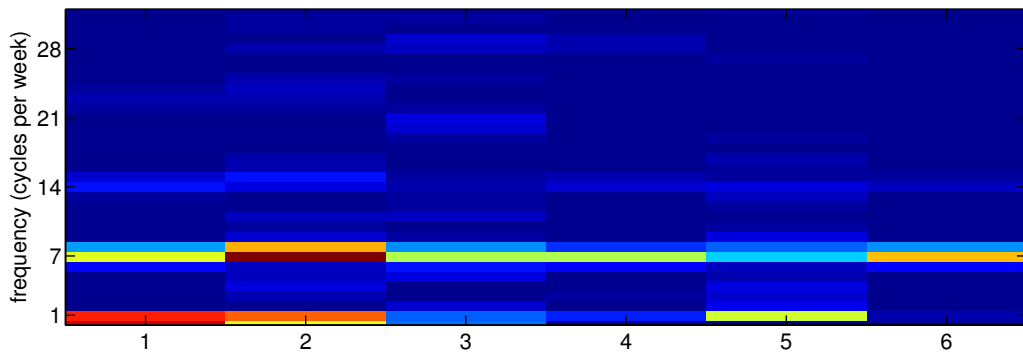

**(d)** *Fourier approximation to the data using 100 (complex) coefficients.*

Figure 7: *Fourier analysis of Abilene data shown in Figure 2c. Blue curves represent the Fourier approximation of the traffic, indicated by the red curves.*

(a) *Six weeks of Abilene data.*

(b) *Spectrogram with weekly windows. Brighter colours indicate more power. Again we see strong power at 1 and 7 cycles per week, though the strength of these varies per week. For instance, in week 5 (when the Independence Day holiday was held) there was a week day, whose traffic more closely resembled weekend traffic, breaking the weekly cycle, and pushing more power into the daily cycle.*
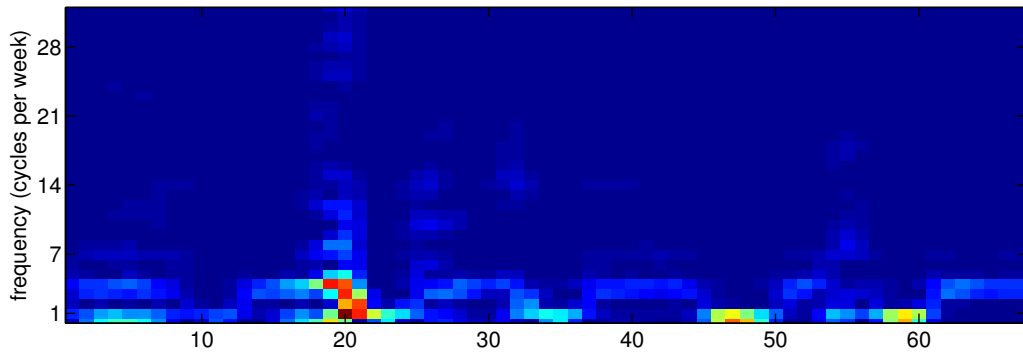
(c) *Spectrogram with daily windows. Note that the time-resolution in this case is actually poorer than the image would suggest, accounting for the poor resolution of the daily and weekly cycles in this figure. However, the large anomalous spikes of traffic in the second week stand out clearly in this view, as they spread power across a range of frequencies.*

Figure 8: *Short-Time Fourier analysis of Abilene data.*

assumed to be centred.

The matrix $\boldsymbol{\mathcal{X}}^{\mathrm{T}}\boldsymbol{\mathcal{X}}$ is *positive semidefinite*. Visualising this geometrically, if the columns of the matrix is reinterpreted as a set of points, then they trace out an ellipsoid. Alternatively, $\boldsymbol{\mathcal{X}}^{\mathrm{T}}\boldsymbol{\mathcal{X}}$ may be viewed as the *empirical covariance matrix* of the columns of $\boldsymbol{\mathcal{X}}$, in effect computing temporal correlations in traffic.

PCA is used to find the directions of greatest variance of $\boldsymbol{\mathcal{X}}^{\mathrm{T}}\boldsymbol{\mathcal{X}}$ by decomposing $\boldsymbol{\mathcal{X}}^{\mathrm{T}}\boldsymbol{\mathcal{X}} = \mathbf{W}\mathbf{D}\mathbf{W}^{\mathrm{T}}$, where $\mathbf{W}$ is an orthonormal matrix containing the eigenvectors of $\boldsymbol{\mathcal{X}}^{\mathrm{T}}\boldsymbol{\mathcal{X}}$ and $\mathbf{D}$ the diagonal matrix containing the eigenvalues of $\boldsymbol{\mathcal{X}}^{\mathrm{T}}\boldsymbol{\mathcal{X}}$. The eigenvectors are known collectively as the *principal axes*. The eigenvectors are ordered in a non-decreasing order with respect to their associated eigenvalues, starting from the eigenvector associated with the largest eigenvalue to the smallest. An equivalent view is that PCA essentially performs a Singular Value Decomposition (SVD) of the matrix $\boldsymbol{\mathcal{X}}$, by computing only its right singular basis $\mathbf{W}$.

Thus, every row of $\boldsymbol{\mathcal{X}}$ can be expressed as $\mathbf{x}_k = \mathbf{a}_k^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}$, *i.e.,* a linear combination of a coefficient vector $\mathbf{a}_k$, called the *principal components*. Here, $\mathbf{W}$ is equivalent to a linear transform, post-multiplied to the data. Intuitively, if the size of the set of principal axes with large principal components are small, then this is evidence there are high temporal correlations between the traffic flows. In practice, it is common to focus on the few largest principal axes for the purpose of data reduction. Basically, this means choosing the first few significant columns of $\mathbf{W}$ to approximate each $\mathbf{x}_k$ with $\tilde{\mathbf{x}}_k$ such that $\|\mathbf{x}_k - \tilde{\mathbf{x}}_k\|_2 < \epsilon$, for some small error $\epsilon > 0$.

As an aside, PCA may be performed on $\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{X}}^{\mathrm{T}}$, in effect computing the spatial correlations of $\boldsymbol{\mathcal{X}}$ instead. Here, we have $\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{X}}^{\mathrm{T}} = \mathbf{V}\tilde{\mathbf{D}}\mathbf{V}^{\mathrm{T}}$, with each column $\bar{\mathbf{x}}_k = \mathbf{V}\tilde{\mathbf{a}}_k$, equivalent to $\mathbf{V}$ pre-multiplied with the data. Spatial PCA was used in the context of anomaly detection [63–65] but there are problems with this approach. These discussions are deferred to §5.
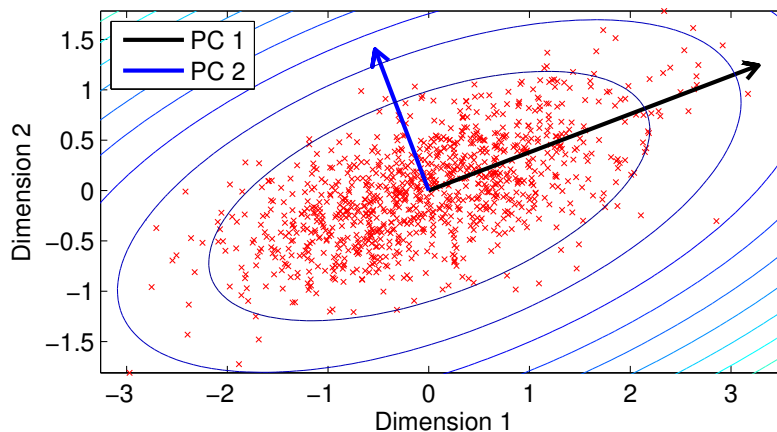
Figure 9: *Principal components analysis of the empirical covariance matrix of a two dimensional data matrix of 1000 centred points $\boldsymbol{\mathcal{X}}$. Here, "PC 1" and "PC 2" are the principal components. Note the elliptical shape of the contours of the density with semi-major and semi-minor axis given by the first and second principal components, respectively.*

Figure 9 demonstrates an example of PCA performed on the covariance matrix of 1000 two dimensional data points with zero mean, *i.e.,* $\boldsymbol{\mathcal{X}}$ has 2 rows and 1000 columns. The matrix $\boldsymbol{\mathcal{X}}^{\mathrm{T}}\boldsymbol{\mathcal{X}}$ formed by the data points vaguely resembles an ellipse. Here, there are two principal components, denoted by "PC 1" and "PC 2", with the higher variance captured by PC 1. This is clear from the way the points on the figure are

distributed. The key point to take away is that both components capture the direction of highest variance and are orthogonal to each other. Moreover, the principal components matrix $\mathbf{W}$ has PC 1 and PC 2 as its first and second columns respectively. Each data point can be expressed as linear combination of these two components. The concept is easily extended beyond two dimensions to the larger dimensions typically encountered with traffic matrices.

PCA was performed by Lakhina *et al.* on empirical data from two backbone networks show that OD flows are a combination of no more than 35 "eigenflows" (the principal axes), and in fact, fewer than this in general [66], out of over 100 OD flows. These eigenflows belonged to one of three categories, depending on their properties:

(i) **deterministic or d-eigenflow**: generally the significant diurnal component of the largest OD flows. Although present in smaller OD flows, these eigenflows are less significant. These eigenflows have a cyclo-stationary property and suggests that these eigenflows may be approximated by a small number of Fourier coefficients. These eigenflows account for the majority of the total traffic of the OD flow.

(ii) **spike or s-eigenflow**: medium sized eigenflows with a spikiness behaviour in time, with values ranging up to 5 standard deviations from the mean of the OD flow. This suggests these contributions come from bursty processes and may be modelled by a wideband Fourier process.

(iii) **noise or n-eigenflow**: small eigenflows behaving like stationary additive white Gaussian noise. These eigenflows have small energy and their contribution to overall traffic is negligible. The majority of eigenflows from Lakhina *et al.*'s datasets belong to this category.

There are several eigenflows belonging to two or more categories, but these eigenflows are rare [66]. For the most part, these categories are very distinct for almost all eigenflows. The low number of eigenflows compared to the dimension of the traffic matrices under study suggests low intrinsic dimensionality of traffic matrices, although the upper bound of 35 eigenflows indicates that the OD traffic is only "approximately" low rank. The results show a power law-type distribution of the principal components [66]. The decay of the distribution varies depending on the ISP, with some distributions exhibiting a very fast decay and some much slower decay.

In many senses PCA confirms the previous analysis and modelling, but it is interesting because its approach simply looks for correlations across different sets of measurements, and uses a different set of assumptions from, for instance, Fourier analysis which can be performed on a single time series.

Finally, it is important to note that the full data needs to be available (no missing entries in $\mathcal{X}$) in order to perform PCA. Furthermore, the basic flavour of PCA as described above is not a robust method, since it is an entirely data driven method and is therefore sensitive to outliers [94]. Robust variants have been proposed but they necessarily complicate the basic version of PCA presented here, since these modifications entail constructing methods to identify and exclude outliers. Despite these disadvantages, in its purest form, PCA is a useful tool to learn the temporal structure of traffic flows.

4.2 Spatial Modelling

Spatial models only focus on the properties of traffic between source and destination pairs, typically within a single measurement interval, without regard to how the traffic changes in time. The models presented here are the gravity model and its generalisations, the discrete choices model, the independent connections model and low rank spatial model. However, we shall start with the simplest test models. For ease of exposition in this section, the set of sources and destinations are assumed to be sets of PoP ingress and egress nodes, denoted by $\mathcal{I}$ and $\mathcal{E}$ respectively. The set Ω represents the set of all nodes in the network, *i.e.*, $\Omega = \mathcal{I} \cup \mathcal{E}$.

4.2.1 Simple *test* models

We must remember that the purpose of models is not always to "realistically" represent a network's traffic. Their purpose is to provide inputs to other tasks. One common task is to assess the sensitivity of a network to different types of traffic, and to that end, engineers can consider the effect of various artificial *test* models.

Three such are the uniform traffic model, peak load model, and focussed overload model. They are extremely simple:

uniform this simple model assigns the same value to all traffic matrix elements. It is used to provide a base load in some experiments, or to see the behaviour of a network under one extreme (the most uniform extreme) of traffic [18, Chapter 4.5.1].

peak load this model is equally simple, and equally extreme. It has zero for all loads except one OD flow. It simulates the opposite extreme where the aim is to see the effect of one dominant flow.

focussed overload this type of traffic matrix simulates the effect of a *focussed overload*, or *flash crowd*[9], where many users become interested in one location or resource and the traffic to this single location from all other sources is the dominant effect in the network. As a result, the focussed overload can be represented by a matrix with all elements zero, except for one row. We can likewise represent a focussed traffic load arising from a single point (say as response traffic to a focussed set of queries) by a matrix with a single non-zero column.

The advantage of each of the models lies in its simplicity. The simplicity means that the effect of the traffic is easy to interpret, and thus gain insights from these models where a more complex model would perhaps confound us with multiple potential causes for some results. For instance, in each of the above models we can gradually increase the traffic to see when capacity bounds are reached, and where those bounds would be reached in order to identify potential bottlenecks in a network.

Other test models based on classical distributions such as the Poisson and Gaussian distributions were proposed by Vardi [117], and Tebaldi and West [111] and Cao *et al.* [24], respectively. Their well-known properties make it easy to analyse results and provide insights, at the cost of a departure from real traffic properties.

4.2.2 Gravity model

The gravity model is perhaps the next simplest type of model, but it has a great deal to offer. Here, traffic from the source to the destination are modelled as a random process. In its simplest form it assumes, any packet originating a source to a destination nodes are *independent* of other packets. Depending on context, this could be the origin and destination, or ingress and egress nodes respectively. Consequently, the traffic between two nodes is *proportional* to the total traffic from the source node to the destination node. The gravity model is amongst one of the most well-studied models and is considered a canonical first generation model.

The name of the model derives from Newton's model of gravitation, where the gravitational force is proportional to the product of the mass of two objects divided by the distance between them squared. The general formulation of the gravity model is defined by two forces: the *repulsive* force (factor) R_i, associated with "leaving" from i and the *attractive* force (factor) A_j, associated with "going" into j. Its general form is described by the following equation:

$$X_{i,j} = \frac{R_i \, A_j}{f_{i,j}}, \tag{5}$$

---

[9]See also the slashdot effect.

where $f_{i,j}$ represents the *friction factor*, which describes the weakening of the forces (akin to distance in Newton's model), depending on the physical structure of the modelled phenomenon. The model has been used extensively in various fields, for instance the modelling of street traffic [86].

In the context of Internet traffic matrix modelling, the friction factors have typically been taken to be constant. That is, distance is assumed to have little effect on network traffic. That certainly seemed to be true even at a fairly large scale in the past, but it is unknown to what extent the deployment of CDNs (Content Distribution Networks) over the last few years has changed distance dependence, since CDNs locate traffic closer to the end user to avoid paying for transit costs across the network, or how inter-country matrices are affected by distance (for instance through language barriers). Where distance is ignored, equation (5) becomes

$$X_{i,j} = \frac{X_i^{\text{in}} X_j^{\text{out}}}{X^{\text{total}}}, \tag{6}$$

where X_i^{in} is the total traffic entering the network through i, X_j^{out} is the total traffic exiting the network through j and X^{total} is the total traffic across the network [125]. The model can be expressed succinctly as the single rank matrix

$$\mathbf{X} = \frac{\mathbf{x}^{\text{in}} \, \mathbf{x}^{\text{out T}}}{X^{\text{total}}}. \tag{7}$$

The popularity of the model stems from the ease of estimating the X_i^{in} and X_j^{out} for each node pair (i, j), and especially at the PoP or backbone level, since the level of traffic aggregation mitigates errors in the estimation of these quantities from sampled traffic.

The gravity model only captures the spatial structure of the traffic. The key assumption of the gravity model is the independence between each source i and destination j. Coupled with the assumption that none of the nodes act as a source or sink of traffic (*i.e.,* that traffic is conserved in the network) $X^{\text{total}} = \sum_{k \in \mathcal{I}} X_k^{\text{in}} = \sum_{\ell \in \mathcal{E}} X_\ell^{\text{out}}$. Under normal operating conditions in most backbone routers, where congestion is kept to a minimum, the conservation assumption appears reasonable. With this assumption,

$$X_{i,j} = X^{\text{total}} \, p_i^{\text{in}} \, p_j^{\text{out}}, \tag{8}$$

where

$$p_i^{\text{in}} = \frac{X_i^{\text{in}}}{\sum_{k \in \mathcal{I}} X_k^{\text{in}}}, \quad \text{and} \quad p_j^{\text{out}} = \frac{X_j^{\text{out}}}{\sum_{\ell \in \mathcal{E}} X_\ell^{\text{out}}},$$

are the proportions of traffic entering the ingress and exiting the egress nodes respectively, called *fanouts*. The formulation (8) is known as the *fanout* formulation because it describes how a packet entering via node i is distributed to several nodes $j \in \mathcal{E}$. Fanout has been demonstrated to be close to a constant over several measurement intervals, compared to the traffic matrix [73], suggesting the fanout may be a better alternative to measure and use in, for instance, anomaly detection, than the raw traffic volumes.

Observe the implication of independence between the source and destination in (8): $\Pr(\mathcal{I}, \mathcal{E}) = p_{\mathcal{I}}^{\text{in}} p_{\mathcal{E}}^{\text{out}}$. An immediate consequence is $\Pr(\mathcal{E} \,|\, \mathcal{I}) = P_d(\mathcal{E})$, where $P_d(\mathcal{E})$ is the marginal distribution of the traffic demand distribution at the destinations. The assumption of independence between the source and destination leads to two important properties of the gravity model making it well suited to traffic matrix modelling.

Theorem 1 (Independence)**.** *Independence between the source and destination traffic holds for any randomly chosen submatrix of the model.*

Proof. The independence property implies $\Pr(s, d) = p_s^{\text{in}} p_d^{\text{out}}$, holding for every $s \in \mathcal{I}$ and $d \in \mathcal{E}$. This condition would also hold for a subsample of locations in $\mathcal{I}$ and $\mathcal{E}$. $\qquad\square$

131

Theorem 2 (Aggregation). *An aggregate of the gravity model is itself also a gravity model.*

Proof. Let all nodes be partitioned into N subsets $\{\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_N\}$, with $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for $i \neq j$ and $\cup_{i=1}^{N} \mathcal{S}_i = \Omega$. The aggregated traffic matrix is defined as

$$X_{\mathcal{S}_i, \mathcal{S}_j} = \sum_{i \in \mathcal{S}_i} \sum_{j \in \mathcal{S}_j} X_{i,j}. \tag{9}$$

The independence condition implies

$$X_{i,j} = \frac{X_{i,\Omega} \, X_{\Omega,j}}{X^{\text{total}}}. \tag{10}$$

Substituting (10) into (9),

$$X_{\mathcal{S}_i, \mathcal{S}_j} = \sum_{i \in \mathcal{S}_i} \sum_{j \in \mathcal{S}_j} \frac{X_{i,\Omega} \, X_{\Omega,j}}{X^{\text{total}}} = \frac{1}{X^{\text{total}}} \sum_{i \in \mathcal{S}_i} X_{i,\Omega} \sum_{j \in \mathcal{S}_j} X_{\Omega,j}$$
$$= \frac{X_{\mathcal{S}_i, \Omega} \, X_{\Omega, \mathcal{S}_j}}{X^{\text{total}}}.$$

which is also a gravity model. $\square$

These are not just theoretical results. Any model should be consistent in the sense that if the data to which it applies is viewed in a different way (for instance by sampling or aggregation) then the model should still apply (though its parameter values may change). It seems like an obvious requirement, and yet there are many models to which it does not apply.

The utility of the gravity model is not just restricted to network measurement. It is used in various areas: teletraffic modelling [61, 67], economy and trade [87, 114], epidemiology [49, 76, 122], sociology [109], the retail industry, specifically Reilly's law of retail gravitation [32, 59, 92], and in vehicular traffic modelling [42]. More advanced discussion on the gravity model (albeit with an economics flavour) is found in [103].

The gravity model can be interpreted in terms of the *principle of maximum entropy*. Entropy here is the Shannon entropy from information theory parlance [33]. The principle is closely related to Occam's Razor, essentially choosing the parsimonious explanation of the data amongst competing explanations. With little information regarding the traffic matrix besides the total traffic information, it turns out that the best one can do, according to the principle, is to describe the observations with a model promoting independence and symmetry, consistent with known constraints. In this way, the model enjoys robustness compared to other models, as the gravity model seeks to minimise deviation from what has already been observed.

The model, however, is not without its drawbacks. The main critique against the gravity model is in its main assumption: the independence of the ingress and egress nodes[10]. It has been pointed out in several papers [43] that this assumption does not hold true. Most traffic between node pairs are determined by connections, for example TCP initiated sessions, so there exist dependencies between node pairs. The second is the violation of the conservation of traffic assumption, for *e.g.,* when there is high congestion, causing packets to be dropped from router queues.

Actual traffic matrices are generally asymmetric, violating the main assumption of gravity models. For example, forward traffic volumes of a source-destination pair of nodes do not typically match up with the volume of reverse traffic. Even if the OD traffic matrix matches the gravity model well, the corresponding IE traffic matrix may be vastly different, due to hot potato routing [113].

---

[10]The difference between OD and IE traffic matrices becomes critical here.
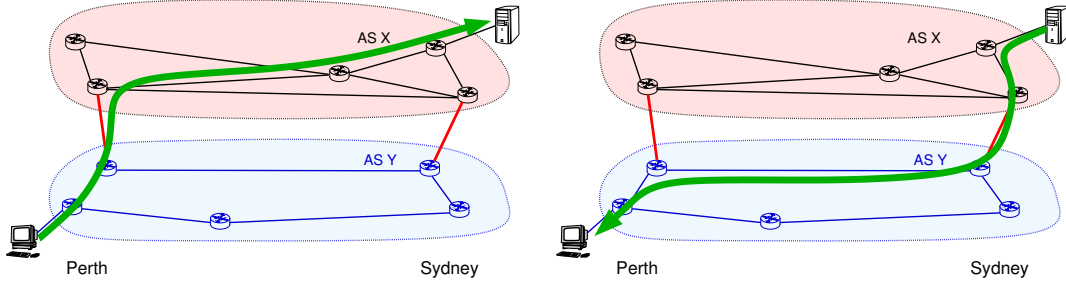
Figure 10: *Traffic flow between two ASes, one in Perth and the other in Sydney. Note the asymmetry in traffic: due to the action of hot potato routing, the path taken by a traffic flow from Perth to Sydney differs from the reverse path, since by BGP's implementation, the closest external link of an AS is always chosen to route traffic out from the AS.*

Hot potato routing is implemented as a part of the BGP decision process [90]. BGP allows network operators to choose the egress points of traffic at the prefix level. The decision may also vary across a network so that traffic at different points can end up being routed to different egress points. The idea of hot potato routing comes from its namesake: traffic is the "hot potato" in this case and the network tries to get rid of the "hot potato" as quickly as possible to avoid costs of transiting it over long distances. Therefore, traffic is sent on the shortest external route connecting an ingress to egress point. BGP provides less control over ingress points and this is what leads to the fundamental asymmetry in the IE traffic matrix.

An example of hot potato routing is in Figure 10. Here, there is a clear asymmetry since the paths taken by traffic flows from Perth to Sydney differ from Sydney to Perth. To further understand hot potato routing, we refer the reader to [17] for a basic understanding of BGP routing policy.

Thus, although the source-destination independence assumption may hold for OD traffic matrices, it may not necessarily hold for IE traffic matrices, due to distortion by inter-domain routing. Consider a simple toy example of a network in Figure 11 (originally from [7]). The ASes A, B and C are assumed to be connected, with A having three routers: 1, 2 and 3. The inter-domain routing protocol between these ASes uses hot potato routing, seeking the shortest path between these ASes.

Suppose $X^{\text{total}} = 9$. Consider an OD traffic matrix with the form of a gravity model, with even spread of traffic over each internal router 1, 2 and 3, with $\mathbf{x}^{\text{in}} = \mathbf{x}^{\text{out}} = \mathbf{x}$. The OD traffic matrix has the form $\mathbf{X}_{\text{OD}} = \mathbf{x}\mathbf{x}^{\text{T}}/X^{\text{total}}$, with $\mathbf{x} = (1,1,1,3,3)^{\text{T}}$, and written explicitly as

$$
\mathbf{X}_{\text{OD}} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \text{B} \\ \text{C} \end{array}
\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & \text{B} & \text{C} \end{array} \\
\left(\begin{array}{ccccc}
1/9 & 1/9 & 1/9 & 1/3 & 1/3 \\
1/9 & 1/9 & 1/9 & 1/3 & 1/3 \\
1/9 & 1/9 & 1/9 & 1/3 & 1/3 \\
1/3 & 1/3 & 1/3 & 1 & 1 \\
1/3 & 1/3 & 1/3 & 1 & 1
\end{array} \right) \end{array} . \tag{11}
$$

By Theorem 2, the gravity model for the aggregated OD matrix, comprising OD traffic volumes between ASes A, B and C, is given by

$$
\mathbf{X}'_{\text{OD}} = \begin{array}{c} \\ \text{A} \\ \text{B} \\ \text{C} \end{array}
\begin{array}{c} \begin{array}{ccc} \text{A} & \text{B} & \text{C} \end{array} \\
\left(\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{array} \right) \end{array} , \tag{12}
$$

133

simply by summing the traffic in the internal nodes. In this case, $\mathbf{X}'_{OD} = \mathbf{x}\mathbf{x}^T/X^{total}$, with $\mathbf{x} = (3, 3, 3)^T$, still a gravity model.

In order to construct the IE traffic matrix, the ingress and egress points of the network in A needs to be determined. The following assumptions are made:

  (i) A, B and C are peers,

 (ii) the shortest AS path protocol is used for inter-domain routing,

(iii) hot potato routing is used internally by A, and

(iv) the Interior Gateway Protocol (IGP) weights are all equal.

Suppose ingress and egress points are defined by the following routing table ($*$ represents a wildcard character)

Origin router	Destination	Egress router
1	B	2
1	C	3
2	*	2
3	*	3

The path for each traffic flow in the network, therefore, differs depending on its source and destination.

All traffic flows between the PoPs may be decomposed into four components: internal traffic within A, traffic departing A, traffic coming into A and traffic external to A, shown in Figure 12. The internal traffic of A (Figure 12a) is just the top-left 3×3 submatrix of $\mathbf{X}_{OD}$, which is

$$\mathbf{X}_{internal} = \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \begin{pmatrix} 1 & 2 & 3 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix} \tag{13}$$

Traffic bound for A, as seen in Figure 12b to be specifically for router 1 in this instance, from its peers has entry points controlled by B and C, given the above routing table. Hence, from A's point of view, the traffic behaves as if the traffic randomly distributed across ingress links. Assuming the traffic is evenly spread, the traffic matrix is

$$\mathbf{X}_{arriving} = \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \begin{pmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{pmatrix} \tag{14}$$

Traffic departing from A, seen in Figure 12c as originating from router 1, and routed by hot potato routing, is described by

$$\mathbf{X}_{departing} = \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1/3 & 1/3 \\ 0 & 2/3 & 0 \\ 0 & 0 & 2/3 \end{pmatrix} \tag{15}$$

Since A does not provide transit for B and C, traffic external to A, *i.e.,* between B and C, should not appear on A, the traffic will remain unseen by A (Figure 12d). Thus, the total IE traffic matrix is the sum of the
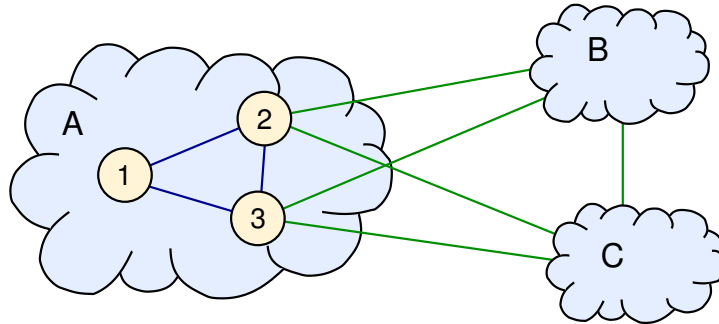
Figure 11: *Example toy network with three ASes: A, B and C are all assumed to be peers. The routers 1, 2 and 3 are internal to A.*

(a) *Internal traffic within A.*

(b) *Incoming traffic to A.*

(c) *Outgoing traffic from A.*

(d) *Traffic external to A.*
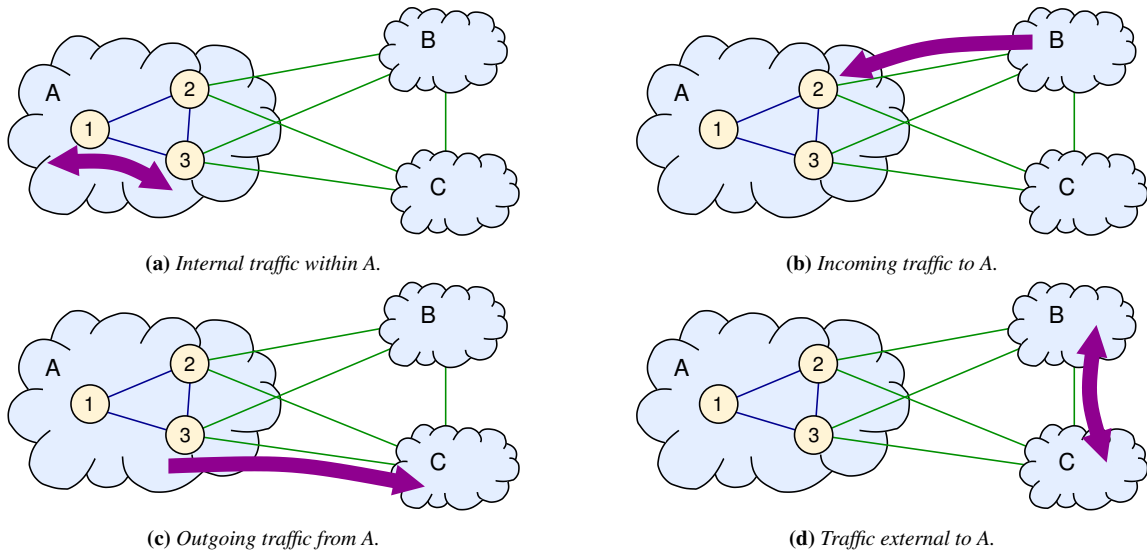
Figure 12: *Traffic flows within the network of Figure 11, classified into four components.*

component traffic above, so that the entry and exit points match, and is given by

$$\mathbf{X}_{\text{IE}} = \left(\begin{array}{ccc} 1/9 & 4/9 & 4/9 \\ 4/9 & 10/9 & 4/9 \\ 4/9 & 4/9 & 10/9 \end{array} \right) . \tag{16}$$

The matrix $\mathbf{X}_{\text{IE}}$ is not equal to $\mathbf{X}'_{\text{OD}}$ in (12), simply due to traffic asymmetry resulting from hot potato routing. Moreover, the assumption of the conservation of traffic no longer holds, since the total traffic of $\mathbf{X}'_{\text{IE}}$ is not equal to X^{total}. The diagonal terms, for example, are much larger than in $\mathbf{X}'_{\text{OD}}$. This example demonstrates that even if the OD traffic matrix is generated from the gravity model, the IE traffic matrix does not necessarily have a structure that conforms to the gravity model.

For large backbone networks where large aggregates of traffic are observed, the gravity model performs admirably, as evident from the results of [125] and its use in AT&T's backbone network for traffic engineering purposes. On smaller, local area networks, however, its effectiveness is limited. The friction factor $f_{i,j}$ may not necessarily be constant in actual traffic matrices, possibly due to different time zones [98], especially for a global spanning network, language barriers, or the increased deployment of CDNs[11]. There may also a distance dependency present between ingress and egress points [7].

The gravity model by itself incurs significant estimation error as the estimates obtained typically do not match the observed link counts. Due to violations of these assumptions, the gravity model turns out to be inaccurate when used in traffic matrix estimation. For example, it was reported to have $\pm 39\%$ accuracy when used in estimating traffic matrices [96].

Despite the flaws mentioned, the gravity model was reported to be a good initial estimate to more sophisticated methods. The model was paired with SNMP link measurements to develop the so-called *tomogravity* technique [125]. The gravity model is also surprisingly useful in the synthesis of traffic matrices. When proposed as a method for synthesising traffic matrices by Roughan [96], the gravity model serves as an excellent first order model for generating the cumulative distribution function of the traffic demands, closely mimicking the statistical properties of actual traffic matrices. While the basic gravity model may not necessarily be an optimal model, it is a simple and good first order model for estimation and synthesis purposes, and it can be improved to take into account the factors described above.

4.2.3 Generalised gravity model

In order to improve the efficacy of the basic gravity model and to address its deficiencies, a generalisation of the gravity model was developed [126, 127]. In a nutshell, the assumption of independent ingress and egress nodes was relaxed by dividing traffic into several classes of ingress and egress nodes, evident from the example in the previous section. Independence only applies to traffic belonging within a certain class, effectively enforcing a *conditional independence* criterion. Such an assumption is closer to actual conditions between ingress-egress pairs in a network.

In particular, the model now accounts for asymmetry of the IE traffic matrix. To account for the effect from hot potato routing, traffic is separated into classes based on peering and access links. Consider again the network in Figure 11. From the figure, two classes can be defined: internal and external classes. There are then four types of source-destination links (see Figure 12): *internal to internal, internal to external, external to internal* and *external to external*.

---

[11] The deployment of CDNs exacerbates this effect since they are located close to the end user so as to avoid having to pay for their traffic transiting other networks.

In the generalised gravity model, independence between nodes are only assumed between the internal to internal class and the external to external class. Thus, routers 1,2 and 3 in AS A are independent to each other, and so are ASes A, B and C to one another, but not traffic from 2 to B, for instance.

Thus, in the generalised gravity model, a modification is made by ensuring the independence assumption still holds, but only when conditioned within each traffic class. In terms of probabilities, traffic is *conditionally independent*, as formulated below for the joint fanout distribution of the sets of access nodes of the network of interest $\mathcal{A}$ and peering nodes $\mathcal{P}$ respectively:

$$p_{S,D}(s,d) = \begin{cases} \frac{p_S(s)}{p_S(\mathcal{A})}\frac{p_D(d)}{p_D(\mathcal{A})}(1 - p_S(\mathcal{P}) - p_D(\mathcal{P})), & \text{for } s \in \mathcal{A}, d \in \mathcal{A}, \\ p_S(s)\frac{p_D(d)}{p_D(\mathcal{A})}, & \text{for } s \in \mathcal{P}, d \in \mathcal{A}, \\ \frac{p_S(s)}{p_S(\mathcal{A})}p_D(d), & \text{for } s \in \mathcal{A}, d \in \mathcal{P}, \\ 0, & \text{for } s \in \mathcal{P}, d \in \mathcal{P}. \end{cases} \tag{17}$$

The four probabilities corresponds to the four cases in Figure 12. In particular, as per intuition, peering traffic is set to zero, since this class does not transit the network of interest.

The stratification of traffic into several classes results in an improved model. Its performance in the traffic matrix estimation results is significantly better than the basic gravity model [126]. Further stratification beyond separating peering and access nodes is possible. For example, the origin of the traffic, whether from a fixed location or mobile device, or the destination of the traffic, depending on application profiles, may be defined as new classes in the model. However, further classification in this manner is only possible with more side information available.

The generalised gravity model is superior to the basic model, but gravity models in general have been somewhat tarnished by the same brush. Most works benchmarking the performance of various models, for instance, for estimation, compare against only the simple gravity model, but make confusing statements that could lead one to believe that all such models are faulty. In fact, the generalised gravity model is vastly superior, but rarely used outside of the company at which it was first developed — AT&T. The chief reason is that the model requires additional topological and routing data, and for the external traffic flows to be mapped using this data [127]. This is a non-trivial task. In addition, in many external studies researchers have not had access to, in particular, knowledge on access and peering links in the network under study. Network operators are not open to releasing information on their networks to the public, however, the Abilene dataset, used in [127], and the GÉANT dataset [116] are publicly available, and contains enough information to make such comparisons.

4.2.4 Discrete choice models

Another proposed model is the *choice model*, introduced by Medina *et al.* [73]. The basis of the discrete choice model (DCM) is the theory of choice models for decision behaviour, originally developed in psychology, and later expanded upon by researchers in other fields, more recently in economics, by Daniel McFadden, for which he won the Nobel Prize in Economics in 2000 (see for example, [71]).

Choice models are popular in econometric applications as the model is used to describe a simplified underlying mechanism of rational decision behaviour. It has been used for transportation analysis, econometrics, marketing and consumer theory. The main inspiration for its use in Internet modelling comes from [110], where a choice model is used in the context of modelling the behaviour of travellers between the cities of Maceio and Sao Paulo, two cities in Brazil, as it parallels traffic traversing PoPs.

The choice model is defined by four elements:

(i) the decision makers,

(ii) the set of alternatives (choices),

(iii) the attributes of the decision maker and the set of alternatives, and

(iv) the decision rules.

All these elements play a key role in ultimately determining the decision process. The *decision makers* represent the agents making the decisions on which choices to go for. The *set of alternatives* characterise the set of possible actions the agents can choose. Each decision maker executes several choices based on its own inherent properties, or *attributes*, as well as the attributes of the set of alternatives. These attributes predispose a decision maker to certain alternatives. Finally, the *decision rules* determine how choices are made. How good a choice is, is measured by a standard based on a set of criteria. The rules establish constraints on the choices of the decision makers, enforcing consistency in the entire system. All four elements of the model aim to capture how agents would naturally decide on several differing choices in a system, in a rational and consistent way, based on a set of rules.

In the context of network traffic modelling, there are two interdependent factors influencing choices. First, the network users' behaviours determine much of how traffic flows are generated, as discussed in §4.1. Second, the network design and configuration plays a very important role in how traffic flows are delivered on the network. Routing protocols, policies, QoS as determined by the network operator and the geographical local of routers and PoPs determine how traffic is transported within the network and between networks. One could visualise this as a two level process: users generate the traffic flows, whereupon the flows are routed through the network, based on the network's design and policies, to the flows' destinations.

All four elements have direct analogues in the context of network traffic modelling. The decision makers are the set of ingress nodes, aggregating all information about the users' behaviours and network design and policies. The set of alternatives are the set of egress nodes, which aggregates the information about the users connected to these nodes. Thus, each decision maker i has a choice set $\mathcal{C} \subseteq \mathcal{E}$. Each node i, a decision maker, is modelled by the equation, for all $j \in \mathcal{C}$,

$$U_j^i = V_j^i + \varepsilon_j^i, \tag{18}$$

where U_j^i denotes the utility between the node pair i and j, V_j^i aggregates the information from the user behaviour and network design, which is deterministic, and ε_j^i is a random component to account for missing information from unknown factors. The term V_j^i can be thought of accounting for the level of attractivity of a destination node j. In [73], the authors proposed M-attributes per decision maker-choice pair, such that

$$V_j^i = \sum_{m=1}^{M} \mu_m \omega_j^i(m) + \gamma_j, \tag{19}$$

where $\omega_j^i(m)$ denotes the m-th attribute, μ_m are weights to account for the relative importance of the m-th attribute, and γ_j is a scaling term for other factors for attractivity, besides all M attributes. An attribute $\omega_j^i(m)$ could be the size of the destination node PoP, since a large egress PoP is more likely to have traffic exiting from it, or the number of peering links the destination node j has.

Based on the above, Medina *et al.* [73] proposed a traffic matrix model that assumes the decomposition

$$X_{i,j} = O_i \alpha_{i,j}. \tag{20}$$

The parameters O_i and $\alpha_{i,j}$, $\forall j$ denote the total outgoing traffic volume from node i and the *fanout* of node i respectively. For each i, $\sum_j \alpha_{i,j} = 1$. The total traffic from a node O_i is known from SNMP data.

Observe that the traffic matrix is now parameterised by the fanout distribution which has a direct analogy in the gravity model. In inference applications, it is the fanout distribution being estimated, thus indirectly inferring the traffic matrix, rather than directly estimating the traffic demands. Fanouts have been shown to be generally stable over a measurement period (several hours), compared to traffic demands [57], which is advantageous in traffic matrix estimation, since the stability contributes to more accurate inference.

The fanout distribution is determined by a decision rule. In [73], a utility maximisation criterion was used,

$$\alpha_{i,j} = \Pr\left(U_j^i = \max_{k \in \mathcal{C}}\{U_k^i\}\right). \tag{21}$$

Now, $\alpha_{i,j}$ is a random quantity as it depends on $\epsilon_{i,j}$, as observed from equation (18). A natural starting point is to assume ε_j^i is i.i.d. Gaussian distributed with mean 0 and variance 1. This transforms (21) to the well-known multiple normal probability unit or m-probit model [70]. However, there is no closed form for (21) under this assumption. Instead, by assuming ε_j^i is i.i.d. distributed following the Gumbel distribution, the m-probit model can be approximated, with (21) now having a closed form. This model is popularly called the multiple logistic probability unit or m-logit model [70]. The closed form is simply

$$\alpha_{i,j} = \frac{\exp(V_j^i)}{\sum_{k \in \mathcal{C}} \exp(V_k^i)}, \tag{22}$$

implying that

$$X_{i,j} = O_i \frac{\exp(V_j^i)}{\sum_{k \in \mathcal{C}} \exp(V_k^i)}. \tag{23}$$

The difficulty lies in determining what attributes should be included. The authors considered two models which they empirically validated:

  (i) $V_j^i = \mu_1 \omega_j(1) + \gamma_j$, where $\omega_j(1)$ denotes the total incoming bytes to an egress PoP j, and

  (ii) $V_j^i = \mu_1 \omega_j(1) + \mu_2 \omega^i(2) + \gamma_j$, where in addition, $\omega^i(2)$ denotes the total bytes leaving the ingress PoP i.

In general, the second model is more accurate, owing to the additional attribute, but it is not known if it is just a case of overfitting or the new parameter is truly useful.

The choice model is a variation of the gravity model. In particular, looking back at equation (20), the total traffic outflowing from ingress i may be regarded as the *repulsion factor*, while the parameters $\alpha_{i,j}$ combining both the *attractiveness factor* and the *friction factor*. A quick comparison of the choice model to (8) highlights the strong link between both models. The choice model, however, has a larger number of parameters to account for the attributes of the decision maker and set of alternatives.

4.2.5 Independent connections model

The independent connections model (ICM) was introduced in [43,44]. Unlike the gravity model, this model discards the assumption of independence between the ingress and egress nodes, and instead focuses on the *connections* between nodes. More specifically, the model differentiates between *initiators*, nodes that initiate a traffic connection, such as a TCP connection, and *responders*, the nodes that accept these connections. The independence assumption comes in by assuming that each initiator and responder are independent, in effect, resulting in independent *connections*.

The inspiration for the ICM comes from traffic characterisation studies, specifically on TCP behaviour. TCP creates two-way connections in response to a SYN packet, the packet used to initialise a connection. Although it is common for the majority of traffic to flow in one direction, there is also a smaller reverse flow. Common examples include an HTTP query, which involves query packets flowing in one direction, and a much larger set of data flowing in the other as a response, or an FTP transaction which may involve mainly data flow in one direction, but the forward packets require acknowledgement packets in the reverse direction. Therefore, the model uses the notion of a connection: a two-way exchange of packets between an *initiator* and a *responder*, corresponding to the ingress and egress nodes, without necessarily assuming symmetry of the two-way traffic.

Three parameters were defined as a product of these studies. The first parameter, the forward traffic proportion $f_{i,j}$ is the normalised proportion of forward traffic from a connection between ingress i to egress j, measured in packets or bytes and $0 \leq f_{i,j} \leq 1$, $\forall i \in \mathcal{I}$ and $j \in \mathcal{E}$. The second parameter A_i describes the activity level of the users at i (the A stands for 'activity'). Finally, some nodes may be chosen for connection more than others, and thus, P_j (stands for 'preference') denotes the preference for node j.

The main assumption of the model is that the probability that a connection responder belongs to node j depends on j only. The values of P_j for $j \in \mathcal{E}$ are unnormalised. They are divided by the sum $\sum_{k \in \Omega} P_k$ in order to treat them as the probability a node j is a connection responder. The parameters A_i and P_j were shown to be uncorrelated on empirical data, providing some evidence these parameters describe two very different underlying quantities.

The model is expressed by

$$X_{i,j} = \frac{f_{i,j} \, A_i \, P_j}{\sum_{k \in \Omega} P_k} + \frac{(1 - f_{j,i}) \, A_j \, P_i}{\sum_{k \in \Omega} P_k}. \tag{24}$$

The first term captures the forward traffic of the connection between initiator i and responder j while the second term its reverse traffic, generated by the users from i and j respectively. The model may be viewed as a weighted sum of two gravity models, with one gravity model characterising the forward traffic, while the other the reverse traffic. Thus potential asymmetries in traffic can be accounted for.

The model is sufficiently flexible to accommodate variations. For example, the *simple IC model* modifies one parameter of model (24) by setting $f_{i,j} = f$, where f is a constant as it has been observed that f is fairly stable from week to week (at least on the Abilene dataset [44]) simplifying the model considerably. Another variation, the *time-varying IC model* includes temporal variation of the parameters, *i.e.,*

$$X_{i,j}(t) = \frac{f(t) \, A_i(t) \, P_j(t)}{\sum_{k \in \Omega} P_k(t)} + \frac{(1 - f(t)) \, A_j(t) \, P_i(t)}{\sum_{k \in \Omega} P_k(t)},$$

and the *stable-fP IC model* removes the time dependency of f and the preferences $\{P_j\}_{j \in \mathcal{I}}$, while the *stable-$f$ IC model* only removes the temporal dependence of f. These variations allows trade-offs between the degrees of freedom of the model and computational complexity, especially when used for the synthesis or inference of traffic matrices. With less parameters, which was shown to be less than the basic gravity model, the model is easier to compute.

The parameters $\{A_i\}_{i \in \mathcal{I}}$ and $\{P_j\}_{j \in \mathcal{E}}$ were validated on actual data. Activity levels $\{A_i\}_{i \in \Omega}$ possess diurnal patterns, corresponding to user access patterns, and a periodic pattern on a weekly timescale. In particular, activity levels are higher on weekdays compared to the weekend, matching observations such as Figure 2c. There is also a more prominent periodic pattern when considering larger nodes, as this effect is due to aggregation, as it captures the users with higher activity levels. These observations are consistent with the temporal properties discussed of traffic matrices discussed in §2. The model was shown to be

effective in estimating traffic matrices, improving over the basic gravity model by $20\% - 25\%$ for the GÉANT dataset [116] and almost 10% for the Totem dataset [43, 44]. These results and observations show that average user behaviour is largely stable and predictable, a great boon to traffic modelling development.

In some ways, the ICM is similar to the DCM, in that both models include parameters to describe the underlying user behaviour, unlike the basic gravity model. For example, both models have a parameter to quantify the level of attractiveness of one node (connection) to another. The DCM is also able to incorporate features of the ICM as well. The differences end there, however. For one, the ICM has a slightly richer description of the flow connections between nodes, such as the forward and reverse traffic flows between nodes, whereas the DCM aggregates the information in a single parameter. The ICM seeks to capture the behaviour of each connection made, rather than merely model the relationship between nodes, emphasising a different focus compared to the DCM. Thus, the ICM may account for hot potato routing and other asymmetries in traffic flow.

4.2.6   Low-rank spatial models

The very noticeable feature of both DCM and ICMs is that they can better represent traffic matrices, but are more highly parameterised. It is, in general, possible to fit a data set more accurately when more parameters are available, but this presents a difficulty – does one accept the more complex, more highly parameterised model, or the simpler, perhaps more robust model?

In the previous cases, this was an "all or none" decision (at least we had to decide on the type of model we used, of not the exact number of choices involved), whereas the gravity model is fixed in its parameterisation. However, there are concepts that easily extend the gravity model.

The low-rank model somewhat new, made popular by its use in matrix completion problems [19, 20, 22, 89]. Low rank models assume the traffic matrix is well-represented by the low rank approximation

$$X_r = \sum_{i=1}^{r} \sigma_i^2 \mathbf{u}_i \mathbf{v}_i^{\mathrm{T}}, \tag{25}$$

where σ_i denotes the i-th singular value, with all singular values arranged in order of descending order, *i.e.,* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$. The famous Eckhart-Young theorem [108, Theorem 4.32, p. 70] states this is the best rank-r approximation, in the sense of the Frobenius norm[12], of a matrix $\mathbf{A}$ given by retaining the largest r singular values of its Singular Value Decomposition (SVD). The theorem, however, assumes that the target matrix for approximation is already known. In low-rank matrix recovery, however, the target matrix is unknown.

In the context of traffic matrix modelling, low-rank models are a relatively recent introduction, beginning with work in [128], although this model was spatio-temporal (see further below). Low rank purely spatial models were proposed and used to good effect by Bharti *et al.* [11]. The choice of a low rank model has strong empirical backing by the earlier results of PCA applied to traffic data, for instance in [66, 124].

In essence, we can see (25) as expressing a traffic matrix as a weighted sum of gravity models, *i.e.,* each single rank component looks exactly the same as that expressed in (7). It seems a logical approach simply because the Internet is not a homogenous entity. In particular there are many types of applications running across the network: from interactive session, to voice, to HTTP, to streamed video. We might imagine that a class of traffic, say streaming video, satisfies the gravity law, but with different row and column sums to, say, voice traffic. Given this, it seems that a weighted sum of gravity matrices is a natural extension.

---

[12] The Frobenius norm is the Euclidean norm applied to matrices *i.e.,* $\|\mathbf{X}\|_F = \sqrt{\sum_{i,j} x_{i,j}^2}$.

Previous models actually turn out to be special cases of this low-rank model. The gravity and discrete choice model are spatial rank-1 models. The generalised gravity model and the ICM are spatial rank-2 models, the latter of which can be observed from the summation of the forward and reverse traffic contributions in equation (24). The low-rank model may be viewed as a general model for providing a fundamental framework for further model development. We shall consider this idea in more detail below in the context of spatio-temporal modelling.

4.3 Spatio-Temporal modelling

Spatio-temporal models aim to describe spatial and temporal structure jointly. Considering the rich structure traffic matrices have both spatially and temporally, these models would be more sophisticated than their purely temporal or spatial counterparts. There has been relatively little work performed on this type of modelling as yet, but there is considerable suggestion that it will be fruitful in the future.

4.3.1 Low-rank spatio-temporal models

The main idea in spatio-temporal modelling of traffic matrices so far has been to exploit the low-rank models mentioned above, but in this context to apply it to the stacked representation of a series of traffic matrices denoted here by $\mathcal{X}$.

Low-rank models assume the traffic matrix is well-represented by the low-rank approximation

$$\mathcal{X}_r = \sum_{i=1}^{r} \bar{\sigma}_i^2 \mathbf{u}_i \mathbf{v}_i^{\mathrm{T}}, \tag{26}$$

where $\bar{\sigma}_i$ denotes the i-th singular value. As before, all singular values are arranged in a descending order. Note the difference between model (26) and the above is the low rank assumption also applies to the temporal structure of the traffic matrix.

In the context of traffic matrix modelling, low-rank models are a relatively recent introduction, beginning with work in [128]. Besides spatial correlations (exploited by the models proposed previously), traffic matrices are known to exhibit temporal correlations, resulting in a low-rank structure both spatially and temporally, justifying the rationale behind the model. The objective of the work is to approximate the time series of traffic matrices $\mathcal{X}$ by a rank-r model $\mathcal{X}_r$. The model proposed here is *spatio-temporal*, in contrast to the models discussed previously, which are only spatial in nature.

Simply insisting on low rank, however, is missing another important point, which is that matrices also exhibit locality, *i.e.,* elements that are close in time (where this might mean time of day, not absolute time), or space exhibit strong correlation. It turns out that the model (26) is greatly enhanced with additional simple constraints on the temporal and spatial structure to reflect the smoothness property of Internet traffic, under normal operating conditions.

The low-rank construction also proved relatively easy to use in practical applications such as matrix completion, and [128] showed that it could be used to do matrix inference from link data, impute missing data (from as little as a few percent of extant values), or be used to predict matrices into the future.

Despite the demonstration of its effectiveness in traffic matrix estimation, low-rank models are still not well understood. Unlike the previous models, where the parameters are, by design, quantitative measures of an underlying network property, low rankedness (in spatio-temporal matrices) does not correspond to any particular network aspect, such as user behaviour. It is just a measure of the spatio-temporal correlation between traffic flows. It does, however, hint that OD traffic flows are *clustered*, if one considers the allocation

of IP prefixes. A better interpretation is necessary to understand the properties of the model, and work in [11, 66] may provide clues in the right direction.

Furthermore, in the recovery of traffic matrices using the low-rank model, theoretical work on the minimum number of measurements required for recovery under structured losses of rows and columns of $\mathcal{X}$, which occurs frequently in the networking context, is left open. At present, the current focus is on random erasures of the elements of $\mathcal{X}$ [19, 20, 22, 89]. Overcoming structured losses is far more important than random erasures, as such a scenario is frequently encountered in real networks. For example, a router failure may result in missing data for an entire row of the traffic matrix. The results of [128] show much promise, as the method is largely immune to structured losses. The challenge now is to construct a theory as to why this is so and as to what extent structured losses may be recovered.

Low-rank models hold much promise for the development of more sophisticated models. More work is required to understand the spatio-temporal properties of the traffic matrix, but as preliminary results indicate, there is a potentially rich structure to exploit.

4.3.2 Tensors and hyper-matrices

A time series of purely spatial traffic matrices is simply a 3-dimensional array, which is sometimes also called a hyper-matrix. Such a representation would be a more natural representation as it would theoretically preserve spatio-temporal properties better than the stacked matrix, as well as track the evolution of the traffic demands throughout the measurement interval. A tensor representation of traffic is even better as it is invariant to changes of basis, unlike hyper-matrices. The difficulty, however, is identifying the type of decomposition of the tensor that would produce low-rank structures, or a beneficial, exploitable structure. There are many proposed methods for tensor decomposition, but the two most popular are the Canonical Polyadic (CP) or PARAFAC decomposition and the Tucker or multilinear decomposition [60]. Tensor decomposition requires a large number of computations, which may be an obstacle to its adoption in traffic matrix recovery. At present the one work exploiting the tensor structure of network traffic to impute missing entries of network traffic tensor is found in [4].

5 Applications

5.1 Traffic Matrix Recovery

As noted earlier it may be difficult to measure traffic matrices directly, but we need to recover traffic matrices from measurements before they can be used. The technique will obviously depend on the available measurements, but there is a glut of works on the recovery of traffic matrices, whether at the OD level, IE level or AS level. Amongst these traffic matrices, IE traffic matrices are relatively easier to recover, as measurements of these matrices are available through SNMP link counts. Recovery of OD level traffic matrices are fraught with challenges because at any point in time, only a subset of IP traffic is seen by a network. There is no way to know what goes on in the entire IPv4 address range, unless all measurements of the global network were combined, but even so, the data from such an endeavour would be massive and computationally intractable to analyse, let alone accurate in the first place. Similarly, for AS level traffic matrices, the lack of measurements as well as error prone measurement tools lead to inaccurate recovery of these matrices.

The major challenge in recovering the IE traffic matrix from SNMP measurements is that the problem is highly *underconstrained*. The set of linear equations (1) is under-determined, *i.e.,* there are many solutions to the observations. Recall that the measurements themselves are subject to error possibly due to poor data collection methods and poor vendor implementation of SNMP polling. They are also not fine-grained since

polling of the measurements is performed every 5 minutes (and the polling intervals may not be perfectly synchronised across a whole network). Clearly, any inference method is required to be robust against these errors and uncertainties.

The underconstrainedness of the problem may be mitigated by active measures. One is direct measurement, using dedicated monitors or in-built measurement software on routers such as NetFlow [1]. Direct measurements at even a single point of ingress results in measurement of an entire row of the traffic matrix, drastically reducing the number of missing matrix entries. Another interesting proposal is to change the IGP (Interior Gateway Protocol) link weights over several snapshots within the measurement interval to provide fresh sets of observations, thereby resulting in a system of linear equations with a unique solution (full rank) out of the SNMP measurements [80, 107]. Both these techniques may be impractical, either being too costly in the case of direct measurements, or requiring direct intervention by the network operator for IGP weight changes. Most proposals simply avoid these by settling on a passive approach of inferring the traffic matrix straight from SNMP data.

There are two main approaches to traffic matrix inference. The first is the deterministic approach, where $\mathbf{y}$ is assumed to provide hard constraints, rather than statistical data. Goldschmidt [54] formulated this as a Linear Program (LP) where the objective was designed to find bounds on traffic matrix elements. In simple terms, the LP finds the traffic matrix with the worst case upper and lower bound on the traffic demand subject to constraints. Recall the vectorised traffic matrix $\mathbf{x}$ has size $N(N-1)$. For the upper bound, the LP model is defined with the objective function

$$\max_{\mathbf{x}} \sum_{j=1}^{N(N-1)} \omega_j x_j, \tag{27}$$

where ω_j is a weight for an OD pair j, also called the coefficient of demand. There are three constraints to satisfy, namely,

(i) **observation constraints**:
$$\sum_{j=1}^{N(N-1)} A_{ij} x_j \leq y_i, \ i = 1, 2, \cdots, L, \tag{28}$$

(ii) **flow conservation constraints**:

$$\sum_{\substack{\ell_1=i,\ell_2=j, \\ \ell_1 \neq \ell_2}} y_{\ell_1} A_{\ell_2 k} - \sum_{\substack{\ell_1=j,\ell_2=i, \\ \ell_1 \neq \ell_2}} y_{\ell_1} A_{\ell_2 k} = \begin{cases} x_k, & \text{if } j \text{ is the source of } k, \\ -x_k, & \text{if } j \text{ is the destination of } k, \\ 0, & \text{otherwise.} \end{cases} \tag{29}$$

(iii) **non-negativity constraints**: $x_j \geq 0, j = 1, 2, \cdots, N(N-1)$.

Similarly, the lower bound is found by substituting the maximisation operation in (27) with a minimisation operation. The LP only produces a nontrivial solution if the lower bound and upper bound on the traffic demand is greater than zero and less than the observed total link count, *i.e.*, $\sum_j y_j$.

Unfortunately, the utility of the LP is only restricted to small toy problems. First, two linear programs have to be solved each time to obtain the upper and lower bounds on traffic demands, which is computationally expensive for large N. Second, the LP was shown to have terrible performance when tested on several types of traffic matrices [73]. Estimates of some traffic matrix entries were in excess of 200%, with most in excess of 100% error, proving that while the LP may be useful for certain small topologies, in general it

is not considered a practical estimation method. The reason for this is because the LP sets many estimated values to zero, resulting in overcompensation for the rest of the estimated values in order to meet the total traffic constraints. Third, there is a high sensitivity of the solution to weight choices, which implies that different solutions will be obtained depending on the chosen weights.

Instead, a more successful alternative is the use of statistical models and regularisation, *i.e.,* treating the traffic matrix as a realisation of a random process generated from a model. Regularisation refers to the inference technique of imposing additional structural assumptions on the problem to reduce undercon-strainedness. Regularisation methods are defined by four components:

(i) a **prior solution**, generated from a *model*,

(ii) a **model deviation** measure, used to compute the deviation of a feasible solution from the model,

(iii) a **distortion** measure, used to compare the deviation of the model with the observations, and

(iv) an **adjustment step**, to ensure the constraints on the total traffic entering and exiting all ingress and egress nodes respectively, as well as non-negativity constraints, are satisfied.

In terms of an optimisation procedure, solving the tomography problem is equivalent to

$$\mathbf{x}^\star = \underset{\mathbf{x} \in \mathbb{R}^{N(N-1)}}{\operatorname{argmin}} \quad R(\mathbf{x}, \mathbf{y}) + \lambda d(\mathbf{x}, \mathcal{M}), \tag{30}$$

where $R(\cdot, \cdot)$ denotes the distortion measure, $d(\cdot, \cdot)$ denotes the model deviation measure and $\lambda \geq 0$ is the penalty constant that amplifies the penalisation of a feasible solution which strays too far away from the model[13]. Typically, $R(\mathbf{x}, \mathbf{y}) = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2$. Regularisation techniques are *biased* to a particular prior model. Thus, if the model is inconsistent, then the estimator (30) would be inconsistent as well. However, if the prior model chosen describes the final solution somewhat accurately, then it is expected that the final estimate would be fairly accurate.

As an example, suppose the prior model, $\mathbf{x}^{(0)}$, used is the gravity model, which can be derived from link measurements by calculating the ingress and egress traffic volumes (by summing link measures on the edge-links of the network). One proposed penalty [127] is defined as

$$d(\hat{\mathbf{x}}, \mathbf{x}^{(0)}) = H(\hat{\mathbf{x}}) + H(\mathbf{x}^{(0)}) - H(\hat{\mathbf{x}}, \mathbf{x}^{(0)}), \tag{31}$$

where

$$H(\mathbf{x}) = - \sum_{j=1}^{N(N-1)} \frac{x_j}{\sum_{k=1}^{N(N-1)} x_k} \log \frac{x_j}{\sum_{k=1}^{N(N-1)} x_k}, \tag{32}$$

is the empirical entropy, while

$$H(\mathbf{x}, \mathbf{x}^{(0)}) = \sum_{j=1}^{N(N-1)} \frac{x_j}{\sum_{k=1}^{N(N-1)} x_k} \log\left(\frac{x_j}{\sum_{k=1}^{N(N-1)} x_k} \middle/ \frac{x_j^{(0)}}{\sum_{k=1}^{N(N-1)} x_k^{(0)}} \right), \tag{33}$$

is the joint empirical entropy, between the estimate and the prior model. The penalty function (31) measures the uncertainty between the quantities $\mathbf{x}$ and $\mathbf{x}^{(0)}$, and is commonly known as the *mutual information* [33].

---

[13]Technically, $\mathbf{x}$ comprises non-negative integers, but a relaxation to real numbers is used as it is easier to compute, especially when considering large traffic matrices.

The joint entropy term $H(\mathbf{x}, \mathbf{x}^{(0)})$ quantifies the uncertainty between $\mathbf{x}$ and $\mathbf{x}^{(0)}$. If $H(\mathbf{x}, \mathbf{x}^{(0)}) = 0$, then $\mathbf{x}$ is statistically independent of $\mathbf{x}^{(0)}$.

The approach is highly flexible: it can deal with the generalised gravity model simply by using a new prior model and constraints (17) are added to account for the different traffic classes (access and peering traffic).

The penalty can be rewritten and thought of as the Kullback-Leibler distance [33] between the estimate $\hat{\mathbf{x}}$ and the prior model $\mathbf{x}^{(0)}$, implying that the estimation objective seeks to preserve as much prior information from $\mathbf{x}^{(0)}$ as possible, while minimising $R(\mathbf{x}, \mathbf{y})$. This can be used directly, or approximated, for instance as a weighted quadratic [125, 126].

Using suitable models, most of the existing inference methods can be described in this framework (see [127] for details). Or, other penalties can be used, such as the nuclear norm, given by

$$d(\mathbf{X}) = \|\mathbf{X}\|_* = \sum_{i=1}^{r} \sigma_i \tag{34}$$

for low rank model recovery.

The solution of the optimisation procedure is often adjusted after regularisation using Iterative Proportional Fitting (IPF) [35], so as to satisfy the observed total traffic constraints and non-negativity constraints (those that were not included in the regularisation for computational reasons). In practice, the IPF is a very simple algorithm, performing fast even on large traffic matrices.
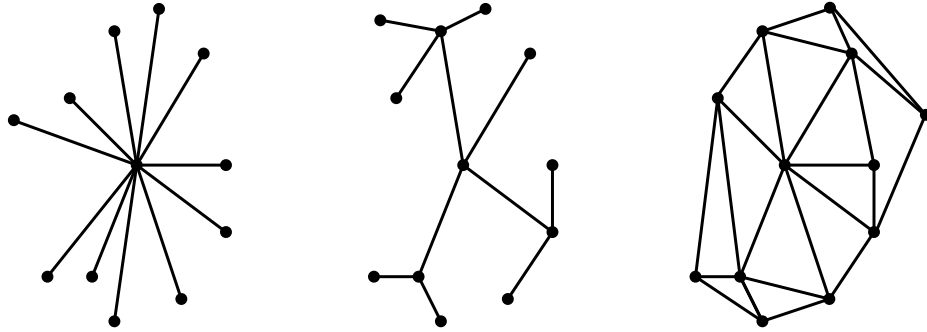
Figure 13: *Three example topologies where local traffic matrices provide benefits (motivated by the seminal figure in [9]).* **Left:** *centralised, or star, topology,* **Centre:** *decentralised topology,* **Right:** *distributed topology.*

Additional information can also be used, for instance, if some rows of the matrix are known from measurements, then this eases the number of variables to be estimated, making the problem a little simpler. Another source of potential data is the collection of *local traffic matrices* [118], providing information on traffic between interfaces of routers. We can see why this is useful by considering the three network topologies in Figure 13, with a centralised or star, a decentralised and a distributed topology. If the network has a star topology, then the entire traffic matrix is known if the local traffic matrix of the router right in the centre is obtained. For the other two topologies, collection of local traffic matrices in strategic places of the topology is likely to reduce the underconstrainedness of the original inference problem, though less (relative) information is provided the more distributed the topology. Local traffic matrices have been demonstrated to provide a significant information boost in [126], especially if the interfaces are well-connected, and that is highly dependent on the underlying network topology. If direct flow measurements from dedicated monitors are available, they provide a huge boon as an entire row of an IE traffic matrix would be revealed. In practice,

however, these are generally not available as they are deemed expensive. The advantage of the regularisation method is that these additional information may be incorporated easily via constraints.

Another issue is their computational tractability. Speed is an issue for these algorithms, since traffic matrices are often large. Most model deviation and distortion measures are chosen to be convex[14], with linear constraints. In this way, problem (30) becomes a convex optimisation problem, where many fast, scalable and efficient algorithms have been developed to solve such problems [13].

The discussions here only considered point-to-point traffic matrices. For IE matrices, the point-to-multipoint matrix may be more useful instead. Recall from the above that an ideal traffic matrix is invariant to other network aspects to be useful for network design. Unlike the point-to-point traffic matrix, the point-to-multipoint matrix contains records on the amount of traffic from one ingress point to a set of egress points. These sets are chosen to preserve invariance under changes in the egress point, a property much more useful for network planning. Inference of the point-to-multipoint traffic matrices may be done in a similar fashion to point-to-point IE matrices [126].

5.2   Network Optimisation

An ISP of a backbone network must ensure each link in the network has adequate capacity. The consequences of failure to provide such capacity is congestion, and a resulting loss of quality service, which in severe cases would result in loss of customers. However, over-provisioning can be wasteful, and so optimisation is used to strike the right balance between cost and capacity.

Network optimisation and engineering [16, 45, 51, 52, 74, 75, 79, 91, 115] involves several tasks with varying planning horizons. Assuming node locations are fixed, in the long term, we plan a network by considering the link locations, and capacity. We refer to this as *network planning*. It may be categorised into two common scenarios: *incremental planning* on existing networks, or *green-fields planning* [98]. In the former, the planning takes an evolutionary route, since the network designer is constrained by the current existing network. Any upgrades to the network are deliberately incremental, so as not to disrupt current operations. Green-fields planning, as the name suggests, starts from scratch: the entire network is designed from ground up. Shorter-term network optimisation tasks include traffic engineering [100] and some potential routing schemes.

As an illustration of the use of a traffic matrix for testing routing schemes, consider a simple example of a topology with three nodes in Figure 14. The traffic matrix is given by

$$\mathbf{X} = \left(\begin{array}{ccc} 0 & 1 & 4 \\ 2 & 0 & 2 \\ 3 & 2 & 0 \end{array} \right).$$

The path distances from PoP 1 to 2, 1 to 3 and 2 to 3 is 1, 2 and 4 respectively. The goal is to find the shortest path routing based on the traffic matrix and the topology. Recall that the link loads may be expressed as $\mathbf{y} = \mathbf{Ax}$, where $\mathbf{x} = (1, 4, 1, 2, 3, 2)^{\mathrm{T}}$ is the vectorised form of $\mathbf{X}$, without including self-traffic. The routing matrix $\mathbf{A}$ here is size 3 by 6, since there are 3 links and 6 flows, as we do not consider self-traffic. Each element is either 0 or 1 as the flows are assumed to be delivered whole. The capacity of each link is assumed to be infinite, or limitless.

---

[14]Convex relaxations of a non-convex objective function is often used as a substitute, for *e.g.,*, the nuclear norm is used in place of the rank function to recover low rank matrices. Under these circumstances, it is crucial to note the assumptions of the model, so as to know when the recovered solution is an excellent approximation of the true solution.
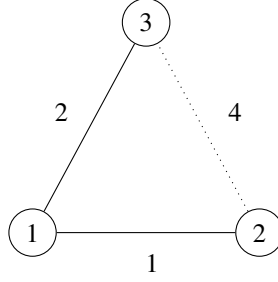
Figure 14: *Example of a simple network of three PoPs and path distances of each link. The shortest path routes chosen are from PoP 1 to 2, and from PoP 1 to 3, as these paths have the lowest distance (corresponding to the thick lines).*

Thus, finding the routes may be cast as the optimisation problem (integer program) with a simple cost function based on path distance

$$\mathbf{A}^\star = \underset{A_{i,j} \in \{0,1\}, \forall i,j}{\text{argmin}} \quad y_{1,2} + 2y_{1,3} + 4y_{2,3} \qquad (35)$$
$$\text{subject to} \qquad \mathbf{y} = \mathbf{A}\mathbf{x}.$$

The solution in this case is a simple one with

$$\mathbf{A}^\star = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The example presented is a simple, trivial one. In practice, networks have a larger topology and additional costs may be included, such as bandwidth utilisation of each link so as to minimise the maximum utilisation of the network, QoS constraints, and constraints on the maximum capacity of each link. For instance, if the maximum capacity of the link from PoP 1 to 3 is at most 4 *i.e.,* $y_{1,3} \leq 4$, then the routing matrix above is no longer a feasible solution. The new solution must incorporate the link from PoP 2 to 3 to conform to constraints (assuming the link 2 to 3 has adequate capacity).

In all of the above mentioned tasks, one of the key ingredients is the traffic matrix. The reason these matrices are so useful comes down to *invariance*. The traffic on a particular link obviously varies as the links in the network change. However, an ideal traffic matrix is invariant to other aspects of the network such as the network topology and underlying routing protocols. Invariance allows the design to be varied without the inputs to the process changing. To a certain extent, the IE traffic matrix satisfies the invariance property, but it is far from perfect for some tasks, most notably it is highly sensitive to external routing changes and some internal changes [112, 113]. The OD matrix is in some sense preferable [7], but harder to measure in most cases. The point-to-multipoint IE matrix (discussed in the previous section) is a useful compromise.

Furthermore, a network operator would require a prediction of the traffic matrix out to the level of the planning horizon for a task. Any forecast depends on the time scale involved and the underlying model used. At short time scales, say minutes, stationarity may be a reasonable approximation, and there are therefore many time series approaches the problem. On time scales of hours to days to weeks, the cyclostationary nature of the data must be included. The temporal models presented earlier can provide such predictions. For instance, with the model (4), we can estimate the average traffic at some time in the future by simply

extrapolating the mean. Longer-term prediction often focusses purely on the large-scale trend $L(t)$ (see §4.1), often captured using a simple growth model (linear or exponential) and regression. In all cases, historical data is needed, usually several times as long as the prediction interval.

In addition, whenever performing prediction we should provide an estimate of variances, or confidence intervals, though this component of the problem has not been well-studied in the specific context of traffic matrices.

5.3   Reliability Analysis

Traffic matrices may also be used to conduct reliability analyses, where the effect (on traffic) or network failures is considered. A basic task in most network design is to create redundant paths to carry traffic in case of failure, but if high reliability is required, then an operator should also ensure that there is sufficient capacity in the network to carry this traffic along its alternate paths. For more details of this task see [98].

5.4   Anomaly Detection

In reality, not all traffic of a network is legitimate. Various attacks may be launched: DDoS attacks [84] or worm outbreaks such as the Nimda worm [27]. Non-malicious, but equally violent spikes in traffic may be caused by a flash crowd or implementation bugs. We call these surges *anomalies*, and if they catch a network operator by surprise they can congest networks, causing untold damage to daily network activities. Other types of anomalies may cause drops in traffic, again resulting in performance problems.

All these anomalies may be rare, but the potential damage can be tremendous. It is for these reasons network operators strife to detect anomalies, with the hope of protecting their networks from these harmful effects.

Although network equipment vendors do provide some form of fast detection and diagnosis mechanisms, these features are generally not adequate for the problems listed above. Consequently, methods were developed to counter anomalies. One approach is to use detailed (packet level) traces and signature-based detection to detect known attacks, but this does not help if the attack is unknown (in advance) or if the necessary measurements are not available. Other techniques infer statistical anomalies from the traffic flow data or SNMP measurements. Traffic matrices play an important role in this respect, since these matrices record traffic volumes across a whole network.

The basic principle of anomaly detection is to define a *baseline operating condition* of the network, by establishing normal conditions of the traffic. The baseline could be from a model, say a gravity model, for example. There are many approaches, such as entropy-based methods [14,56], as network anomaly detection itself is a vast topic, but here, the focus is on the direct use of traffic matrices for anomaly detection.

Deviations from the baseline predictions of the traffic are quantified with a chosen norm, and one is flagged as an anomaly if it exceeds a predefined threshold. There are two sources of error: *false positives*, when normal traffic is flagged as an anomaly, and *false negatives*, when a detector does not flag an anomaly. The latter type of error, where we miss a potential anomaly, are apparently a more serious problem (given the serious nature of anomalies). However, if too many false positives occur, then operators can be overwhelmed, and will typically ignore the alarm system. The false positive problem is exacerbated if the number of tests is large, and in traffic matrix analysis (where we might conduct one test per traffic matrix element, per time interval) that number can be very large, requiring a very low false positive rate.

We consider the tradeoff between the two in an ROC (Receiver Operating Characteristic) curve which shows the two types of errors plotted against each other as a function of the chosen threshold (or other

suitable tuning parameter). However, proper assessment of an approach requires ground-truth data, which is, by the nature of anomalies, hard to obtain in the volumes required.

Models themselves can be modified to account for anomalous traffic. In §4.1, the model (4) itself has a term to account for sudden spikes in traffic, which was demonstrated empirically to be useful in detecting large shifts in traffic. Low rank models [128] were shown to be highly effective in detecting anomalies as well.

Many anomaly detection proposals may be broadly classified as methods to preprocess measurement data via a linear transformation, in order to separate normal traffic from anomalous traffic. This was observed in [124]. Their *anomography* (a portmanteau of "anomaly" and "tomography") framework is easy to understand and is aimed at providing a framework for discussing these types of techniques. It proceeds as follows: start by assuming the routing matrix is static in the entire duration of the measurements. Given a series of SNMP measurements, $\mathbf{Y} = \mathbf{A}\mathcal{X}$, a new inference problem is obtained by multiplying $\mathbf{Y}$ with a linear transform $\mathbf{T}$ to obtain $\tilde{\mathbf{Y}} = \mathbf{A}\tilde{\mathcal{X}}$, which are the anomalous link loads. Whether the focus is on spatial or temporal anomalies depends on whether $\mathbf{Y}$ is pre- or post-multiplied with $\mathbf{T}$:

(i) *spatial anomography*: pre-multiplication, *i.e.,* $\tilde{\mathbf{Y}} = \mathbf{TY}$, uses the spatial relationships between traffic at particular points in time to find traffic that is unusual with respect to other flows at the same time; and

(ii) *temporal anomography*: post-multiplication, *i.e.,* $\tilde{\mathbf{Y}} = \mathbf{YT}$; uses the relationships between traffic at different times to determine if traffic is unusual for its point in time.

The two have been combined to create spatio-temporal anomaly detection [128], though the full details of this go beyond the scope of this chapter.

The above assumes the routing matrix is static over the series of measurements. The models themselves have to be modified to account for possible route changes. Some models are less amenable to modification, requiring a large number of constraints that scale with the number of measurements [124], which makes them undesirable for practitioners.

Anomaly detection employing SNMP data took off with a series of papers [63–65, 68], where the low intrinsic spatial dimensionality of traffic matrices was exploited via a PCA-based anomaly detector. In the spatial PCA-based method, the principal components and axes of $\mathbf{Y}$ are computed from its columns and ordered from most to least significant component, to obtain a subspace $\mathbf{P} = [\mathbf{v}_1\,\mathbf{v}_2\,\cdots\mathbf{v}_m]$. The traffic space is then divided to a *normal subspace* and an *anomalous subspace*. The traffic time series is then projected on each principal axes, starting from $\mathbf{v}_1$ and so forth, and the projection magnitude is compared to a simple hard threshold of three standard deviations from the mean. Once there exists a projection exceeding this threshold, say at some $\mathbf{v}_K$, this component and subsequent components are classified as belonging the anomalous subspace $\mathbf{P}_A = [\mathbf{v}_K\,\mathbf{v}_{K+1}\,\cdots\,\mathbf{v}_m]$. The anomalous traffic is identified by projecting the time series onto the anomalous subspace and projecting the traffic back to obtain $\tilde{\mathbf{Y}}$.

Lakhina *et al.*'s spatial PCA method fits in the framework since the last step of extracting the anomalous traffic involves the projection $\tilde{\mathbf{Y}} = (\mathbf{P}_A^\mathrm{T}\mathbf{P}_A)\mathbf{Y}$ so the linear transformation is $\mathbf{T} = (\mathbf{P}_A^\mathrm{T}\mathbf{P}_A)$. However, its shortcomings have been the subject of scrutiny. Implementation of spatial PCA to network traffic is likely to be ineffective due to several drawbacks [94, 124]. Spatial PCA can be contaminated with a large anomaly[15], rendering it unable to detect the anomaly. In fact, PCA has been known to flag an entire measurement interval although there is only one anomaly present [124]. Additionally, PCA is very sensitive to the underlying data: adequate measurements are required and there must be a sufficient level of traffic aggregation before

---

[15]Though in fact this is a problem in general for anomaly detection, and has not received the attention it deserves.

underlying trends can be detected by PCA. It is not robust enough in practice, requiring much fine tuning. Finally, there is a high computational cost involved in computing the principal components of a traffic matrix.

Other alternatives exist: wavelet transformations [10, 83], Fourier transformations, autoregressive integrated moving average or ARIMA [124], and temporal PCA [124], where PCA is applied to the rows of $\mathbf{Y}$ (the temporal dimension) instead. In all these techniques, the baseline traffic flows are assumed to follow the prescribed model. In the Fourier model, baseline traffic is assumed to be composed of low frequencies. High frequencies may potentially indicated the presence of anomalies since these correspond to sudden changes in the traffic. Thus, the transformation filters out low frequencies and examines the remaining high frequencies to determine if any of these frequencies exceed a predetermined threshold. A similar rationale holds for the wavelet transform model. The ARIMA model [15] is very well-known in time series analysis, providing flexibility in the choice of parameters. The model generalises popular models such as a model with a built-in Holt-Winters smoothing, the random walk model and exponentially weighted moving average models. It also allows memory and long range dependency [119] to be built into the model via fractional ARIMA, as evidenced and used to great effect in [101][16].

After $\tilde{\mathbf{Y}}$ is obtained, the anomalous traffic $\tilde{\boldsymbol{\mathcal{X}}}$ has to be recovered. The choice of a particular inference algorithm would depend on the model. The spatial PCA method uses a greedy algorithm to find the largest anomaly in each time bin [63]. Other methods include the use of ℓ_1 regularisation, inspired by compressive sensing [21, 36], which was shown, when coupled with the ARIMA model, outperforms other methods, including PCA and wavelet-based anomaly detection [124].

In short, the model of the traffic matrix matters in anomaly detection. It serves as a baseline. However, it also needs to consistently allow for anomalies. One problem with approaches such as PCA is the models implied by the approach are often left unstated (implicit) and do not allow the anomalies to be separated as part of estimation (thus they can pollute the estimation process). Good techniques, going on into the future, need to be able to perform such separation consistently.

5.5 Traffic Matrix Synthesis

Synthesis of the traffic matrix is an important area, motivated by the lack of real world traffic matrices available, due to the proprietary nature of most traffic data. Publicly available data are often obtained from networks operated and maintained by research institutions and universities, such as GÉANT [116] and Abilene [77], which are limited in scope and is certainly biased towards research and educational networks. Thus, network operations stand to gain much from artificially synthesised traffic matrices, via a combination of good models.

Synthesis is not demanding in some ways. Traffic matrices are usually relatively small compared to other types of traffic data when measured at a reasonable level of aggregation and time scale. However, in other ways these matrices are quite challenging. For instance:

(i) we have few sets of traffic matrix data, and even fewer that are public, and somehow need to use these to estimate properties of these complex, high-dimensional objects;

(ii) there is a real relationship between topology and traffic (although we would like a traffic matrix to be invariant to the topology, there are clear cases where, particularly IE matrices are not);

(iii) traffic matrices come in a wide variety of types (at different levels of aggregation, for particular applications and so on) and it is unlikely that one model fits all; and

---

[16]See [15] for a good introduction to time series analysis.

(iv) there are a number of conflicting goals in synthesis, *e.g.,* to generate variability, but well "matched" to real traffic matrices.

The major use of synthesis is in simulation. Artificial traffic matrices may be used in capacity planning to stress test network topologies to see if they stand up to heavy loads without ending up with congestion. Monte Carlo-type simulations may be used to produce estimates of the behaviour of networks. Synthesised traffic matrices provide an avenue to explore the limitations of a protocol in a controlled environment before running it on an actual network. A good model simplifies these simulations, as parameters of the model can be tuned to generate a variety of scenarios for testing protocols.

In many cases a traffic matrix is enough for a simulation, but in others, we need to translate this into packets (or at least connections). The analogue in transportation modelling is often called a *micro-simulation* model [8, 58]. Here, the problem becomes one of taking a *demand* matrix (remember, most of the work here is related to traffic, not demand), and translating this into carried load. We know how to do that (using simulation tools such as ns) but doing it efficiently is difficult. One paper [105] starts to tackle this problem, but as in the work of transportation modelling, there is considerable scope for advanced scalable micro-simulation of traffic.

Another use for synthetic traffic matrices is in the further task of synthetic topology generation, but we shall leave discussion of this topic to Chapter 7 of this book.

Unfortunately, there is a dearth of work on traffic matrix synthesis, apart from [81, 96] and a brief mention regarding synthesis of matrices from the independent connections model [43, 44]. The problem of synthesising traffic matrices is the inverse of the inference problem. In synthesis, the topology of the network matters, as the generated entries of the artificial traffic matrix must not exceed the link capacity it is mapped to. Some models, such as the gravity model, automatically satisfy bandwidth constraints naturally [96]. However, if the generated entries do not conform to these constraints, there are algorithms to solve these problems [81], albeit with added computational complexity.

Computational complexity of the model is the other important issue, dependent on the number of parameters. Hence, there is an inherent tradeoff between the descriptive power of the model and the ease of synthesising traffic matrices. A guideline is to preferably choose the model with as little parameters as possible but enough proven descriptive power (focussed on the traffic aspect the practitioner intends to capture) measured via an information criterion, such as the AIC [6].

We here describe the simple approach of [96] motived by works such as [50], in order to provide a starting point for future work on such synthesis. We start by taking $\mathbf{x}^{in}$ and $\mathbf{x}^{out}$ to be vectors of N i.i.d. exponential random variables with mean one. The traffic matrix is then generated using (7). We can then adjust the total traffic to match the desired total by simple scaling. This method is extremely simple (an exponential distribution has only one parameter to estimate), and we need generate only $2N$ random variables. Yet it matches observed statistics for both Abilene and GÉANT data extremely well [7].

Synthesis of traffic matrices is an open area, and is important to further explore due to the benefits of using traffic matrices in traffic planning and engineering. There is considerable hope that progress can be made in terms of generating synthetic matrices, both for green-fields network design [61], and for simulation in general.

6 Future Directions

There are some interesting tasks left for traffic matrix research. The various algorithms and techniques described here could be improved, though in many cases the improvements may be relatively incremental

given the success of existing approaches. More interest may be found in extending the ideas and techniques used here to new domains, and to evolving Internet traffic.

There are a few obvious cases (and no doubt many less obvious cases that we have not thought of), for instance: multicast traffic has not, to our knowledge, been studied in this way. Multicast is interesting because it violates the *traffic conservation* assumption that lies underneath many techniques for estimation and modelling of traffic matrices. We could imagine modelling it by considering the "flow" to be the traffic on a multicast group, from say, one source, to a set of destinations, and then stacking a vector with these. The routing matrices now include elements for every link used (no longer following a single path). The traffic "matrix" could then be a column vector of the traffic on each of these flows. So the idea of multicast traffic can fit into the structure we have talked about here, but appropriate models for performing tasks such as inference do not seem to exist.

It would also be very interesting to understand the way that CDNs are affecting network traffic. A step in this direction, although not directly on traffic, but more on the discovery of the content hosts is [5]. A CDN's typical goal is to bring content closer to the user, thereby reducing network traffic. However, that explicitly violates the "friction free" assumption in most gravity models, and introduces distance as something to be modelled.

That leads naturally to the consideration of global traffic. Almost all studies of traffic have concentrated on a single network no larger than the national scale. That may still be very large – for example, several studies looked at Tier 1 providers in the USA, which for some time dominated Internet traffic. However, although large, it was still relatively homogeneous traffic between people speaking much the same language(s) from place to place in the network. When we consider the Internet globally, we may see that there are language or cultural clusters where large groupings of traffic are focussed.

On a large scale, time zones also play a significant role. Traffic patterns show strong cyclic behaviour based on user activity, but such activity is strongly dependent on the local time zone. If traffic is flowing from user to user, then this can result in strong apparent locality effects, simply because people in the same time zone are more likely to be awake at the same time [53].

While language and cultural focussing may be geographic in nature, it might also be considered per network, and that leads to another topic of some interest. Very few papers have tried to consider inter-AS (also known as inter-domain) traffic in any detail. Exceptions are Chang *et al.* [28] (which presents suggestions for estimating traffic based on models derived from business models and resulting usage); Bharti *et al.* [11] (which considered inference of hidden elements of this matrix using a subset of data), Feldman *et al.* [48] (which aimed to estimate a global traffic matrix, but only in the limited domain of WWW traffic), and Labovitz *et al.* [62] (which looked at inter-domain traffic from 110 network operators over a two-year period, though not in the form of a matrix). Study of the Internet's global traffic matrix is made difficult by the sheer scale of the project: Labovitz *et al.* studied 110 network operators over a two-year period[17] and to do so, collected over 200 Exabytes of data. Many network operators do not collect or store data of the type required for such a study, and many more regard it as proprietary or covered by privacy legislation with provisions such that no researcher is ever likely to see it. So we can see that study of the inter-domain matrix is likely to be a long-term, and rather challenging project.

In addition, we know that the *traffic profile* (or mix of applications) has changed fairly rapidly over time. It is likely this trend will continue, and there are bound to be effects on traffic patterns as a result. Peer-2-peer traffic significantly altered traffic patterns when it appeared because it was more symmetric than traditional (at the time) WWW traffic. However, in addition, peer-2-peer applications have the potential to exploit locality information to download from sources closer to the destination. This could potentially change the "no friction" assumption in much the same way that CDNs can, though in the early days it did not appear to

---

[17]To put this in context, there are tens of thousands of ASes in the Internet.

be the case [53]. An example traffic matrix (drawn from [53]) showing normalised[18] traffic between regions in a cable-network operator is given in Table 2. The major deviation, in this data, from a pure gravity model seemed to be time-zone differences.

From/To	R1	R2	R3	R4	R5	R6	R7	R8
R1	-	0.180	0.140	0.126	0.174	0.128	0.124	0.127
R2	0.172	-	0.141	0.126	0.190	0.132	0.118	0.120
R3	0.132	0.120	-	0.189	0.135	0.145	0.139	0.140
R4	0.107	0.111	0.182	-	0.124	0.163	0.155	0.158
R5	0.161	0.180	0.136	0.132	-	0.135	0.127	0.129
R6	0.107	0.108	0.145	0.155	0.125	-	0.187	0.173
R7	0.107	0.106	0.137	0.157	0.127	0.182	-	0.184
R8	0.109	0.111	0.127	0.161	0.128	0.178	0.185	-

Table 2: Normalised inter-regional traffic matrix from [53].

New traffic classes may change traffic matrices in the future, and modelling these will be interesting. On the other side, applications such as anomaly detection are likely to remain interesting due to their immediate benefits to operators, but the most overlooked task is traffic matrix synthesis. As mentioned in the previous section, the lack of real world traffic matrix datasets motivates the use of artificial traffic matrices to provide some degree of approximation in network traffic planning, provisioning and engineering tasks. It is an important area to concentrate on, considering its usefulness to network operators.

7 Conclusion

This chapter has been aimed at introducing the reader to the state-of-the-art in Internet traffic matrix measurement, modelling and applications. It is not a complete survey of all research about Internet traffic matrices as such a survey would necessarily consume a much larger amount of space, and be less digestible, and we apologise to those whose work has not been referenced.

The chapter has also aimed to clarify a set of common terminology in a field which has occasionally been confounded by ambiguous or confusing terms. We have discussed how difficult it is a task to measure and model traffic matrices, due to the intricate complexities of the layering of network protocols. Moreover, these complexities also contribute to challenges in determining the focus a model should take.

Models presented here were classified in three categories, depending on the traffic properties they aim to capture: purely temporal, purely spatial and spatio-temporal models. Several examples of popular models of varying complexity were discussed: Fourier and wavelet models, PCA, the modified Norros model, uniform and Gaussian priors, the gravity model and its variants, the independent connections model, the discrete choice model and low rank models. There are many more models, as the list of models presented here is not meant to be exhaustive. All models have their strengths and weaknesses, and they may only be used in circumstances where the assumptions from which they derive from hold. Furthermore, practitioners should always keep in mind the particular application the model is being used for.

Several applications of traffic matrices and their models were discussed, including their recovery, network optimisation and engineering, anomaly detection and synthesis. These tasks emphasise the practical usefulness of traffic matrices to a network operator.

---

[18]The elements have been normalised by dividing each row by the row-sum, so that each element actually represents the probability that a packet enters the network at a given region i will depart the network at region j.

There are several more open questions, considering the shift in current traffic trends. Multicast traffic has not been studied in-depth. Content delivery networks have a profound impact on traffic behaviour, but to our knowledge, has not garnered much attention with regard to studying their traffic behaviour. Other areas to understand further are the global traffic properties of traffic matrices, application profiles of the traffic and traffic matrix synthesis.

Understanding and modelling traffic matrices is a difficult problem, exacerbated by the lack of publicly available datasets. It is our aim to also provide, as an adjunct to this chapter, links to the most commonly used datasets in this domain, and code to perform some of the commonest tasks. In this way, we hope to provide a firm foundation for future work in the area, and to help those who just want to use traffic matrices in their research.

References

[1] Cisco NetFlow. http://www.cisco.com/go/netflow.

[2] Cisco visual networking index: Forecast and methodology, 2011-2016. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html.

[3] Internet Activity, Australia. http://www.abs.gov.au/AUSSTATS/abs@.nsf/DOSSbyTopic/0444532C5EBD3B76CA256BD0002833B6, 2013.

[4] ACAR, E., KOLDA, T. G., DUNLAVY, D. M., AND MØRUP, M. Scalable tensor factorizations for incomplete data. In *SIAM International Conference on Data Mining (SDM) 2010* (April 2010), pp. 701–712.

[5] AGER, B., MÜHLBAUER, W., SMARAGDAKIS, G., AND UHLIG, S. Web content cartography. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (November 2011), pp. 585–600.

[6] AKAIKE, H. A new look at statistical model identification. *IEEE Trans. Autom. Control 19*, 6 (December 1974), 716–723.

[7] ALDERSON, D. L., CHANG, H., ROUGHAN, M., UHLIG, S., AND WILLINGER, W. The many facets of Internet topology and traffic. *Networks and Heterogeneous Media 1*, 4 (December 2006), 569–600.

[8] ALGERS, S., BERNAUER, E., BOERO, M., BREHERET, L., DI TARANTO, C., DOUGHERTY, M., FOX, K., AND GABARD, J.-F. Review of micro-simulation models. Tech. rep., Simulation Modelling Applied to Road Transport European Scheme Tests (SMARTEST), Leeds University, 1997. http://www.its.leeds.ac.uk/smartest.

[9] BARAN, P. On distributed communications: 1. Introduction to distributed communications network. RAND Memorandum, August 1964.

[10] BARFORD, P., KLINE, J., PLONKA, D., AND RON, A. A signal analysis of network traffic anomalies. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement* (2002), pp. 71–82.

[11] BHARTI, V., KANKAR, P., SETIA, L., GÜRSUN, G., LAKHINA, A., AND CROVELLA, M. Inferring invisible traffic. In *Proceedings of the 6th International Conference* (2010), Co-NEXT '10, pp. 22:1–22:12.

[12] BLILI, R., AND MAGHBOULEH, A. Best practices for determining traffic matrices in IP networks V 4.0. Tutorial in NANOG 43, http://www.nanog.org/meetings/nanog43/presentations/Blili_trafficmatrix_N43.pdf, 2008.

[13] BOYD, S., AND VANDENBERGHE, L. *Convex Optimization*. Cambridge University Press, 2004.

[14] BRAUCKHOFF, D., DIMITROPOULOS, X., WAGNER, A., AND SALAMATIAN, K. Anomaly extraction in backbone networks using association rules. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2009), pp. 28–34.

[15] BROCKWELL, P. J., AND DAVIS, R. A. *Introduction to Time Series and Forecasting*, 2nd ed. Springer, March 2002.

[16] BURIOL, L. S., RESENDE, M. G. C., RIBEIRO, C., AND THORUP, M. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Optimization Online* (2003).

[17] CAESAR, M., AND REXFORD, J. BGP routing policies in ISP networks. *IEEE Network 19*, 6 (2005), 5–11.

[18] CAHN, R. S. *Wide Area Network Design*. Morgan Kaufmann, 1998.

[19] CANDES, E., AND PLAN, Y. Matrix completion with noise. *Proc. IEEE 98*, 6 (June 2010), 925–936.

[20] CANDES, E., AND RECHT, B. Exact matrix completion via convex optimization. *Found. of Comput. Math. 9* (2008), 717–772.

[21] CANDES, E., AND TAO, T. Near optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Info. Theory 52*, 12 (December 2006), 5406–5425.

[22] CANDES, E., AND TAO, T. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Info. Theory 56*, 5 (May 2010), 2053–2080.

[23] CAO, J., CLEVELAND, W. S., LIN, D., AND SUN, D. X. On the nonstationarity of Internet traffic. In *ACM SIGMETRICS 2001* (New York, NY, USA, 2001), pp. 102–112.

[24] CAO, J., DAVIS, D., WIEL, S. V., AND YU, B. Time-varying network tomography: Router link data. *J. Am. Statist. Assoc. 95*, 452 (December 2000), 1063–1075.

[25] CASE, J. D., FEDOR, M., SCHOFFSTALL, M. L., AND DAVIN, J. R. A simple network management protocol (SNMP). Tech. Rep. RFC 1157, IETF, May 1990. http://www.ietf.org/rfc/rfc1157.txt.

[26] CASTRO, R., COATES, M., LIANG, G., NOWAK, R., AND YU, B. Network Tomography: Recent Developments. *Statistical Science Magazine 19*, 3 (August 2004), 499–517.

[27] CERT/CC. CERT Advisory CA-2001-26 Nimda Worm: An Overview. http://www.cert.org/advisories/CA-2001-26.html, September 2001.

[28] CHANG, H., JAMIN, S., MAO, Z., AND WILLINGER, W. An empirical approach to modeling inter-AS traffic matrices. In *ACM/SIGCOMM Internet Measurement Conference (IMC) 2005* (October 2005), pp. 139–152.

[29] CLAFFY, K., BRAUN, H.-W., AND POLYZOS, G. Tracking long-term growth of the NSFNET. Cooperative Association for Internet Data Analysis - CAIDA, http://www.caida.org/publications/papers/1994/tlg/, 1994.

[30] COATES, M., HERO, A., NOWAK, R., AND YU, B. Internet tomography. *Signal Processing Magazine 19*, 3 (May 2002), 47–65.

[31] COATES, M., POINTURIER, Y., AND RABBAT, M. Compressed network monitoring for IP and all-optical networks. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2007), pp. 241–252.

[32] CONVERSE, P. D. New laws of retail gravitation. *The Journal of Marketing 14*, 3 (October 1949), 379–384.

[33] COVER, T. M., AND THOMAS, J. A. *Elements of Information Theory*, 2nd ed. John Wiley and Sons, Inc., 2006.

[34] CROVELLA, M., AND KOLACZYK, E. Graph wavelets for spatial traffic analysis. In *Proceedings of IEEE Infocom* (April 2003), pp. 1848–1857.

[35] DEMING, W. E., AND STEPHAN, F. F. On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. *Ann. Math. Stat 11*, 4 (1940), 427–444.

[36] DONOHO, D. Compressed sensing. *IEEE Trans. Info. Theory 52*, 4 (April 2006), 1289–1306.

[37] DUFFIELD, N., LUND, C., AND THORUP, M. Estimating flow distributions from sampled flow statistics. *IEEE/ACM Trans. Networking 13*, 5 (2005), 933–946.

[38] DUFFIELD, N., LUND, C., AND THORUP, M. Learn more, sample less: control of volume and variance in network measurement. *IEEE Trans. Info. Theory 51*, 5 (2005), 1756–1775.

[39] DUFFIELD, N. G., AND GROSSGLAUSER, M. Trajectory sampling for direct observation. *IEEE/ACM Trans. Networking 9*, 3 (June 2001), 280–292.

[40] DUFFIELD, N. G., AND GROSSGLAUSER, M. Trajectory sampling with unreliable reporting. In *INFOCOM 2004* (2004), pp. 1570–1581.

[41] ERIKSSON, B., BARFORD, P., BOWDEN, R., ROUGHAN, M., DUFFIELD, N., AND SOMMERS, J. BasisDetect : A model-based network event detection framework. In *ACM SIGCOMM Internet Measurement Conference* (Melbourne, Australia, 2010).

[42] ERLANDER, S., AND STEWART, N. F. *The gravity model in transportation analysis: Theory and extensions*. Topics in Transportation. International Science, 1990.

[43] ERRAMILLI, V., CROVELLA, M., AND TAFT, N. An independent-connection model for traffic matrices. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (October 2006), pp. 251–256.

[44] ERRAMILLI, V., CROVELLA, M., AND TAFT, N. An independent-connection model for traffic matrices. Tech. Rep. BUCS-TR-2006-022, Department of Computer Science, Boston University, September 2006.

[45] FEAMSTER, N., BORKENHAGEN, J., AND REXFORD, J. Guidelines for interdomain traffic engineering. *ACM SIGCOMM Computer Communications Review 33*, 5 (October 2003), 19–30.

[46] FELDMANN, A., GREENBERG, A., LUND, C., REINGOLD, N., AND REXFORD, J. Netscope: Traffic engineering for IP networks. *IEEE Net. Magazine 14* (March 2000), 11–19.

[47] FELDMANN, A., GREENBERG, A., LUND, C., REINGOLD, N., REXFORD, J., AND TRUE, F. Deriving demands for operational IP networks: Methodology and experience. *IEEE/ACM Trans. Netw. 9* (June 2001), 265–280.

[48] FELDMANN, A., KAMMENHUBER, N., MAENNEL, O., MAGGS, B., DE PRISCO, R., AND SUNDARAM, R. A methodology for estimating interdomain web traffic demand. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (October 2004), pp. 322–335.

[49] FERRARI, M. J., BJORNSTAD, O. N., PARTAIN, J. L., AND ANTONOVICS, J. A gravity model for the spread of a pollinator-borne plant pathogen. *American Naturalist 168*, 3 (September 2006), 294–303.

[50] FORTZ, B., REXFORD, J., AND THORUP, M. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine 40*, 10 (October 2002), 118–124.

[51] FORTZ, B., AND THORUP, M. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications 20*, 4 (2002), 756–767.

[52] FORTZ, B., AND THORUP, M. Robust optimization of OSPF/IS-IS weights. In *Proc. INOC 2003* (October 2003), pp. 225–230.

[53] GERBER, A., HOULE, J., NGUYEN, H., ROUGHAN, M., AND SEN, S. P2P The Gorilla in the Cable. In *National Cable & Telecommunications Association(NCTA) 2003 National Show* (Chicago,IL, June 2003).

[54] GOLDSCHMIDT, O. ISP backbone inference methods to support traffic engineering: Methodology and experience. In *Internet Statistics and Metrics Analysis (ISMA) Workshop* (December 2000).

[55] GROSCHWITZ, N. K., AND POLYZOS, G. C. A time series model of long-term NSFNET backbone traffic. Cooperative Association for Internet Data Analysis - CAIDA, http://www.caida.org/publications/papers/1994/tsm/, 1994.

[56] GU, Y., MCCALLUM, A., AND TOWSLEY, D. Detecting anomalies in network traffic using maximum entropy estimation. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2005), pp. 32–32.

[57] GUNNAR, A., JOHANSSON, M., AND TELKAMP, T. Traffic matrix estimation: A comparison on real data. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (October 2004), pp. 149–160.

[58] HOOGENDOORN, S. P., AND BOVY, P. H. L. State-of-the-art of vehicular traffic flow modelling. In *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* (2001), pp. 283–303.

[59] JUNG, A. F. Is Reilly's law of retail gravitation always true? *The Journal of Marketing 24*, 2 (October 1959), 62–63.

[60] KOLDA, T. G., AND BADER, B. W. Tensor decompositions and applications. *SIAM Review 51*, 3 (September 2009), 455–500.

[61] KOWALSKI, J., AND WARFIELD, B. Modeling traffic demand between nodes in a telecommunications network. In *ATNAC '95* (1995).

[62] LABOVITZ, C., IEKEL-JOHNSON, S., MCPHERSON, D., OBERHEIDE, J., AND JAHANIAN, F. Internet inter-domain traffic. In *ACM SIGCOMM* (2010), pp. 75–86.

[63] LAKHINA, A., CROVELLA, M., AND DIOT, C. Characterization of network-wide anomalies in traffic flows. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2004), pp. 201–206.

[64] LAKHINA, A., CROVELLA, M., AND DIOT, C. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM 2004* (2004), pp. 219–230.

[65] LAKHINA, A., CROVELLA, M., AND DIOT, C. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM 2005* (2005), pp. 217–228.

[66] LAKHINA, A., PAPAGIANNAKI, K., CROVELLA, M., DIOT, C., KOLACZYK, E. D., AND TAFT, N. Structural analysis of network traffic flows. *SIGMETRICS Perform. Eval. Rev. 32*, 1 (June 2004), 61–72.

[67] LAM, D., COX, D., AND WIDOM, J. Teletraffic modeling for personal communications services. *IEEE Communications Magazine: Special Issues on Teletraffic Modeling Engineering and Management in Wireless and Broadband Networks 35* (February 1997), 79–87.

[68] LI, X., BIAN, F., CROVELLA, M., DIOT, C., GOVINDAN, R., IANNACCONE, G., AND LAKHINA, A. Detection and identification of network anomalies using sketch subspaces. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2006), pp. 147–152.

[69] MALLAT, S. *A Wavelet Tour of Signal Processing*. Academic Press, 1998.

[70] MCCULLAGH, P., AND NELDER, J. A. *Generalized Linear Models*, 2nd ed. Monographs on Statistics and Applied Probability. Chapman and Hall, 1989.

[71] MCFADDEN, D. Modeling the choice of residential location. In *Spatial Interaction Theory and Planning Models*, A. Karlqvist et al., Ed. North Holland, 1978, pp. 75–96.

[72] MEDINA, A., FRALEIGH, C., TAFT, N., BHATTACHARYYA, S., AND DIOT, C. A taxonomy of IP traffic matrices. *Proc. SPIE 4868*, 200–213 (July 2002).

[73] MEDINA, A., TAFT, N., SALMATIAN, K., BHATTACHARYYA, S., AND DIOT, C. Traffic matrix estimation: Existing techniques and new directions. In *ACM SIGCOMM 2002* (2002), pp. 161–174.

[74] MITRA, D., AND WANG, Q. Stochastic traffic engineering for demand uncertainty and risk-aware network revenue management. *IEEE/ACM Trans. Networking 13*, 2 (April 2005), 221–233.

[75] MURPHY, J., HARRIS, R., AND NELSON, R. Traffic engineering using OSPF weights and splitting ratios. In *Proceedings of 6th International Symposium on Communications Interworking of IFIP - Interworking 2002* (2002), pp. 13–16.

[76] MURRAY, G. D., AND CLIFF, A. D. A stochastic model for measles epidemics in a multi-regional setting. *Trans. Institue British Geographers 2* (1977), 158–174.

[77] NLANR. Abilene Trace Data. http://pma.nlanr.net/Special/ipls3.html.

[78] NORROS, I. A storage model with self-similar input. *Queueing Systems 16*, 3-4 (1994), 387–396.

[79] NUCCI, A., BHATTACHARYYA, S., TAFT, N., AND DIOT, C. IGP link weight assignment for operational Tier-1 backbones. *IEEE/ACM Trans. Networking 15*, 4 (August 2007), 789–802.

[80] NUCCI, A., CRUZ, R., TAFT, N., AND DIOT, C. Design of IGP link weight changes for estimation of traffic matrices. In *INFOCOM 2004* (March 2004), vol. 4, pp. 2341–2351.

[81] NUCCI, A., SRIDHARAN, A., AND TAFT, N. The problem of synthetically generating IP traffic matrices: Initial recommendations. *SIGCOMM Comput. Commun. Rev. 35* (July 2005), 19–32.

[82] ODLYZKO, A. M. Internet traffic growth: Sources and implications. In *Optical Transmission Systems and Equipment for WDM Networking II*, B. B. Dingel, W. Weiershausen, A. K. Dutta, and K.-I. Sato, Eds., vol. 5247. Proc. SPIE, 2003, pp. 1–15.

[83] PAPAGIANNAKI, K., TAFT, N., ZHANG, Z.-L., AND DIOT, C. Long-term forecasting of Internet backbone traffic. *IEEE Trans. Neural Netw. 16*, 5 (September 2005), 1110–1124.

[84] PATRIKAKIS, C., MASIKOS, M., AND ZOURARAKI, O. Distributed denial of service attacks. *The Internet Protocol Journal 7*, 4 (December 2004), 13–35.

[85] PAXSON, V. End-to-End routing behavior in the Internet. *IEEE/ACM Trans. Networking 5*, 5 (October 1997), 601–615.

[86] POTTS, R. B., AND OLIVER, R. M. *Flows in Transportation Networks*. Academic Press, 1972.

[87] PÖYHÖNEN, P. A tentative model for the volume of trade between countries. *Weltwirtschaftliches Archive 90* (1963), 93–100.

[88] QUOITIN, B., AND UHLIG, S. Modeling the routing of an autonomous system with C-BGP. *IEEE Network 19*, 6 (2005), 12–19.

[89] RECHT, B., FAZEL, M., AND PARRILO, P. A. Guaranteed minimum–rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review 52*, 3 (2010), 471–501.

[90] REKHTER, Y., AND LI, T. A Border Gateway Protocol 4 (BGP-4). *RFC 1771* (1995). http://www.ietf.org/rfc/rfc1771.txt.

[91] REXFORD, J. Route optimization in IP networks. In *Handbook of Optimization in Telecommunications, Springer Science and Business* (2006), Kluwer Academic Publishers.

[92] REYNOLDS, R. B. A test of the law of retail gravitation. *The Journal of Marketing 17*, 3 (January 1953), 273–277.

[93] RINCÓN, D., ROUGHAN, M., AND WILLINGER, W. Towards a meaningful MRA of traffic matrices. In *ACM SIGCOMM Internet Measurement Conference (IMC'08)* (2008), pp. 331–336.

[94] RINGBERG, H., SOULE, A., REXFORD, J., AND DIOT, C. Sensitivity of PCA for traffic anomaly detection. In *Proceedings of the 2007 ACM SIGMETRICS* (2007), pp. 109–120.

[95] RISSANEN, J. A universal prior for integers and estimation by minimum description length. *Ann. Statis. 11*, 2 (June 1983), 416–431.

[96] ROUGHAN, M. Simplifying the synthesis of Internet traffic matrices. *SIGCOMM Comput. Commun. Rev. 35*, 5 (2005), 93–96.

[97] ROUGHAN, M. A case-study of the accuracy of SNMP measurements. *Journal of Electrical and Computer Engineering 2010* (2010). Article ID 812979.

[98] ROUGHAN, M. Robust network planning. In *The Guide to Reliable Internet Services and Applications*, C. R. Kalmanek, S. Misra, and R. Yang, Eds. Springer, 2010, ch. 5, pp. 137–177.

[99] ROUGHAN, M., GREENBERG, A., KALMANEK, C., RUMSEWICZ, M., YATES, J., AND ZHANG, Y. Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning. In *ACM SIGCOMM Internet Measurement Workshop* (2002), pp. 91–92.

[100] ROUGHAN, M., THORUP, M., AND ZHANG, Y. Traffic engineering with estimated traffic matrices. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (October 2003), pp. 248–258.

[101] SCHERRER, A., LARRIEU, N., OWEZARSKI, P., BORGNAT, P., AND ABRY, P. Non-Gaussian and long memory statistical characterizations for Internet traffic with anomalies. *IEEE Trans. Depend. Secure Computing 4*, 1 (January–March 2007), 56–70.

[102] SCHWARZ, G. Estimating the dimension of a model. *Ann. Statist. 6*, 2 (1978), 461–464.

[103] SEN, A. K., AND SMITH, T. E. *Gravity models of spatial interaction behavior*. Springer, 1995.

[104] SHAIKH, A., ISETT, C., GREENBERG, A., ROUGHAN, M., AND GOTTLIEB, J. A case study of OSPF behavior in a large enterprise network. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement* (2002), IMW '02, pp. 217–230.

[105] SOMMERS, J., BOWDEN, R. A., ERIKSSON, B., BARFORD, P., ROUGHAN, M., AND DUFFIELD, N. G. Efficient network-wide flow record generation. In *IEEE Infocom* (2011), pp. 2363–2371.

[106] SOULE, A., NUCCI, A., CRUZ, R., LEONARDI, E., AND TAFT, N. How to identify and estimate the largest traffic matrix elements in a dynamic environment. *SIGMETRICS Perform. Eval. Rev. 32*, 1 (June 2004), 73–84.

[107] SOULE, A., NUCCI, A., CRUZ, R. L., LEONARDI, E., AND TAFT, N. Estimating dynamic traffic matrices by using viable routing changes. *IEEE/ACM Transactions on Networking 15*, 3 (2007), 485–498.

[108] STEWART, G. W. *Matrix algorithms Volume 1: Basic decompositions*. SIAM, 1998.

[109] STEWART, J. Q. Demographic gravitation: Evidence and applications. *Sociometry 11*, 1/2 (February 1948), 31–58.

[110] SWAIT, J. D. Probabilistic choice set information in transportation demand. Tech. rep., Department of Civil and Environmental Engineering, MIT, June 1984.

[111] TEBALDI, C., AND WEST, M. Bayesian inference on network traffic using link count data. *J. Am. Statist. Assoc. 93*, 442 (June 1998).

[112] TEIXEIRA, R., DUFFIELD, N., REXFORD, J., AND ROUGHAN, M. Traffic matrix reloaded: Impact of routing changes. In *Proc. Passive and Active Measurement Workshop* (April 2005), pp. 251–264.

[113] TEIXEIRA, R., SHAIKH, A., GRIFFIN, T., AND REXFORD, J. Dynamics of hot-potato routing in IP networks. *SIGMETRICS Perform. Eval. Rev. 32* (June 2004), 307–319.

[114] TINBERGEN, J. Shaping the world economy: Suggestions for an international economic policy. *The Twentieth Century Fund* (1962).

[115] UHLIG, S., BONAVENTURE, O., MAGNIN, V., RAPIER, C., AND DERI, L. Implications of the topological properties of internet traffic on traffic engineering. In *19th ACM Symposium on Applied Computing, Special Track on Computer Networks* (March 2004), pp. 339–346.

[116] UHLIG, S., QUOITIN, B., LEPROPRE, J., AND BALON, S. Providing public intradomain traffic matrices to the research community. *Computer Communication Review 36*, 1 (January 2006), 83–86.

[117] VARDI, Y. Network Tomography: Estimating source-destination traffic intensities from link data. *J. Am. Statist. Assoc. 91* (1996), 365–377.

[118] VARGHESE, G., AND ESTAN, C. The measurement manifesto. *SIGCOMM Comput. Commun. Rev. 34*, 1 (January 2004), 9–14.

[119] VEITCH, D., AND ABRY, P. Wavelet analysis of long-range dependent network traffic. In *Proceedings of the 9th INFORMS Applied Probability Conference* (Cambridge, Massachusetts, 30 June - 1 July 1997), INFORMS Technical Section on Applied Probability, p. 57. http://appliedprob.society.informs.org/.

[120] WALLACE, C. S., AND BOULTON, D. M. An information measure for classification. *Computer Journal 11*, 2 (August 1968), 185–194.

[121] WANG, J., AND RABBAT, M. Wavelet-based traffic matrix modelling. In *Network Measurement and Mapping Conference* (2010).

[122] XIA, Y. C., BJORNSTAD, O. N., AND GRENFELL, B. T. Measles metapopulation dynamics: A gravity model for epidemiological coupling and dynamics. *American Naturalist 164*, 3 (2004), 267–281.

[123] ZHANG, Y., GE, Z., DIGGAVI, S., MAO, Z., ROUGHAN, M., VAISHAMPAYAN, V., AND WILLINGER, W. *Markov Processes and Related Topics: A Festschrift for Thomas G. Kurtz*, vol. 4 of *IMS Collections*. Institute for Mathematical Statistics, 2009, ch. Internet Traffic and Multiresolution Analysis, pp. 215–234. Stewart N. Ethier, Jin Feng, Richard H. Stockbridge, Editors.

[124] ZHANG, Y., GE, Z., GREENBERG, A., AND ROUGHAN, M. Network anomography. In *ACM SIGCOMM Internet Measurement Conference (IMC)* (2005), pp. 317–330.

[125] ZHANG, Y., ROUGHAN, M., DUFFIELD, N., AND GREENBERG, A. Fast accurate computation of large-scale IP traffic matrices from link loads. In *ACM SIGMETRICS 2003* (2003), pp. 206–217.

[126] ZHANG, Y., ROUGHAN, M., LUND, C., AND DONOHO, D. An information-theoretic approach to traffic matrix estimation. In *ACM SIGCOMM 2003* (2003), pp. 301–312.

[127] ZHANG, Y., ROUGHAN, M., LUND, C., AND DONOHO, D. Estimating Point-to-Point and Point-to-Multipoint traffic matrices: An information-theoretic approach. *IEEE/ACM Trans. Netw. 13*, 5 (October 2005), 947–960.

[128] ZHANG, Y., ROUGHAN, M., WILLINGER, W., AND QIU, L. Spatio-Temporal compressive sensing and Internet traffic matrices. In *ACM SIGCOMM 2009* (August 2009), pp. 267–278.

Optimizing and Modeling Dynamics in Networks

Ibrahim Matta

1 Introduction

The Internet has grown very large. No one knows exactly how large, but rough estimates indicate billions of users (around 1.8B in 2009, according to eTForecasts [4]), hundreds of millions of web sites (over 200M in February 2009, according to Netcraft [19]), and hundreds of billions of web pages (over 240B, according to the Internet archive [1]). The Internet is also very dynamic — users log in and out, new services get added, routing policies change, normal traffic gets mixed with denial-of-service (DoS) attack traffic, etc.

An important question is: *How do we manage such a huge and highly dynamic structure like the Internet?* As a corollary, how can we build a network of the future unless we understand the *steady-state* and *dynamics* of what we build?

In this chapter, we resort to two mathematical frameworks: *optimization theory* to study optimal steady states of networks, and *control theory* to study the dynamic behavior of networks as they evolve toward steady state. Our emphasis will be on *congestion control* using the notion of *prices* to model the level of congestion, such as delays and losses, observed by users or traffic sources.

Expected Background: We assume basic background in calculus and algebra. We also assume basic knowledge of systems modeling, optimization theory, Laplace transforms, and control theory — Keshav's textbook [13] provides an excellent source for these mathematical foundations, in particular, chapters 4, 5, and 8. Basic knowledge of Internet's Transmission Control Protocol (TCP) [11], namely Reno and Vegas [15] versions, as well as queue management schemes, namely Random Early Drop (RED) [6], should be helpful. This chapter briefly covers needed background material to serve as a refresher or quick reference. The material of this chapter has been used at Boston University in a second (advanced) networking course taken by senior undergraduate and graduate students.

Contribution and Outline: The purpose of this chapter is to make the application of optimization and control theory to congestion control more accessible through intuitive explanations and simple control applications, using examples from Internet's protocols. This chapter has been largely influenced by the work of Frank Kelly [12], which introduces the notion of "prices" and "user utility", the work by R. Srikant [24], which discusses the dynamics of user (traffic source) and network adaptations, and control theory texts and notes (e.g., [20, 16]). The exposition here attempts to tie these various mathematical models and techniques through simple running examples and illustrations, modeling the dynamics of both flow control and routing.

We start by motivating the network control problem using analogy to the problem of producing, pricing, and consuming gas/oil (Section 2). We introduce several examples of optimally allocating resources (link bandwidth) to users (traffic sources), resulting in different notions of fairness. We then introduce dynamic equations that model source and link adaptation algorithms (Section 3). Since these are generally non-linear equations, we review the technique of linearization and how classical (linear) control theory can be used to

study stability and transient performance (Section 4). We use as a running example, a Vegas-like system controlled using different types of well-established controllers. Using linear approximation around a certain operating point, we can only study so-called *local* stability.

To study general (*global*) stability of non-linear models, we introduce the control-theoretic Lyapunov method (Section 5). We also show how the control-theoretic Nyquist stability method can be applied to the linearized model to study the impact of delay in feedback (i.e., measurements of the current state of the system). The material on the Nyquist method is a bit more advanced and can be skipped on a first reading. We generalize the notion of stability by introducing the concept of contractive mapping, and extend its application to routing dynamics (Section 6).

Finally, we provide two case studies that apply control-theoretic techniques introduced in this chapter: the first study investigates stability under class-based scheduling of rate-adaptive traffic flows (Section 7), and the second study investigates stability of data transfer over a dynamic number of rate-adaptive transport connections (Section 8). These case studies can be skipped on a first reading.

The chapter concludes with a set of exercises (Section 9) and their solutions (Section 10).

2   Network Control as an Optimization Problem

In this section, we describe Frank Kelly's optimization framework [12], which models users' expectations (requirements) with utility functions, and network congestion signals (e.g., loss, delay) as prices. The network is shown to allocate transmission rates (throughputs) to users (flows) in such a way as to meet some *fairness* objective.

The objective of a user, or what makes the user happy, can be mathematically modeled as a *utility function*. For example, drivers observe the"price" of transportation and make one of many possible decisions: drive, take the subway instead, walk, bike, or stay home. The decision may involve several factors like the price of gas, convenience, travel time, etc. For example, if it rains, you might decide to drive to work, or you might decide to walk to work to save money and can then afford to go to the movies later in the week. Of course, how much driving a person does, is affected by all sorts of factors and user priorities are unknown to the system of gas stations and oil companies. But, each driver has her own utility!

Figure 1 illustrates with a block diagram the closed-loop relationship between drivers (users), gas stations (where gas is sold to and consumed by users), and the market (which represents OPEC[1], the government, and oil companies that collectively produce gas and set market prices based on user demand). Drivers set the total demand by observing gas prices. Notice that the gas price includes at-the-pump gas price, and possibly other "exogenous" prices like tips for full service, fees for credit card payment, or additional local taxes. Observe also that prices observed by users are *delayed* and do not typically represent the exact current state of the market given inherent delays in gas production, refinement, transportation, etc.

This kind of block diagram is typical of many closed-loop (feedback) control systems where the system is said to reach *equilibrium* if the demand (for gas by drivers) matches the supply (of gas in the market). In data networks, users drive the demand on the network and have different utilities (expectations) when downloading music, playing games, making skype (voice/video) calls, or denying others service by launching a denial-of-service (DoS) attack! In turn, the network observes that user demand and sets "prices", where the price could be real money, or it could be some measure (indication) of congestion (e.g., delay, loss), or it could represent additional resources that need to be allocated to avoid congestion.

An important question is: *What is the goal of network design?* Is it to make users happy? You hope so! Then, mathematically, we say the goal of the network is to maximize the sum of utilities for all its

---

[1]OPEC is the Organization of the Petroleum Exporting Countries.
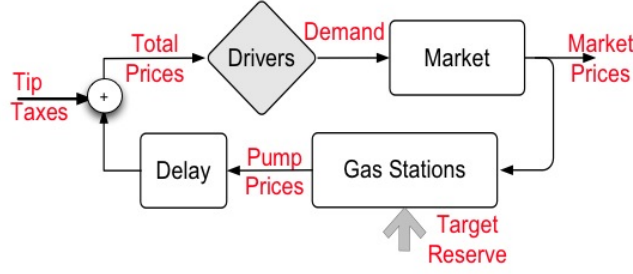
Figure 1: The gas control loop

users [23].[2] Figure 2 illustrates the data network equivalent of the gas control loop shown in Figure 1. We next consider the modeling of user utility and network behavior (resource allocation), before introducing the optimization framework to study the (optimal) steady state for the users and network.
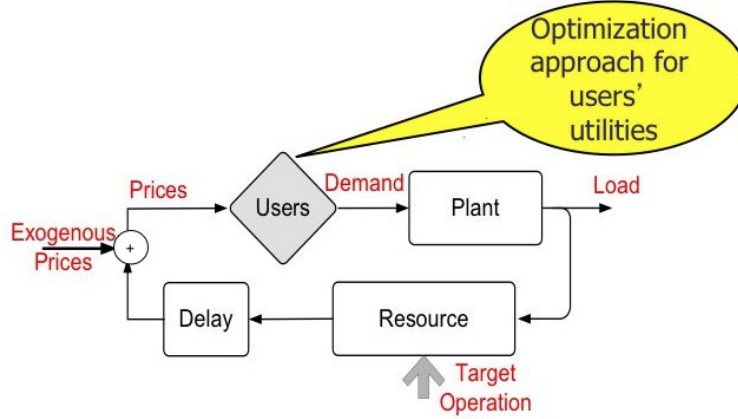
Figure 2: The network control loop

2.1 Modeling the User

Users typically have different utilities, i.e. different applications may perform differently based on the level of service (e.g., loss, delay) they get from the network. But, generally speaking, an application should perform better, the higher the rate (throughput) it is able to send at over the network. It is also generally the case that the gain (level of "happiness") from higher throughput (i.e., *marginal utility*) diminishes as the throughput increases.

Figure 3 shows such a utility function that is typical of what is called *elastic traffic* [23]. Formally, user r has utility $U_r(x_r)$ when allocated rate $x_r > 0$. $U_r(x_r)$ is an increasing, strictly concave function of x_r

---

[2]We note that maximizing the sum of utilities involves interpersonal utility comparison, which needs to be ameliorated by linking to a common good, such as money.

(see Figure 4). And the derivative $\dot{U}_r(x_r) \to \infty$ as $x_r \to 0$, and $\dot{U}_r(x_r) \to 0$ as $x_r \to \infty$. Throughout this chapter we assume strictly concave utilities.
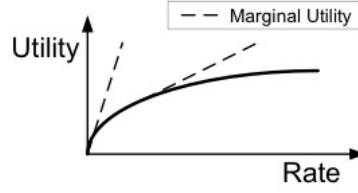
Figure 3: Concave utility function
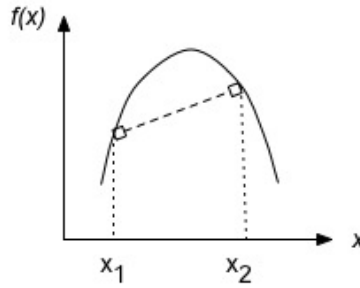
Figure 4: Concave function. A function $f(.)$ is said to be concave if $f(\alpha x_1 + (1 - \alpha)x_2) \geq \alpha f(x_1) + (1 - \alpha)f(x_2)$, i.e., for any two points x_1 and x_2, the straight line that connects $f(x_1)$ and $f(x_2)$ is always below or equal to the function $f(.)$ itself. Note that a differentiable concave function has a maximum value at some point x_{max}, and that the derivative $\dot{f}(x_{max}) = 0$. A strictly concave function would have a strict inequality, whereas a convex function has a cup-like shape and has a minimum instead.

2.2   Modeling the Network

We consider a network of J resources, e.g., transmission links as they are typically considered the bottleneck. We denote by R the set of all possible routes, and we assume that each user (source-destination traffic flow) is assigned to exactly one route r (i.e., static single-path routing)[3]. We then define a 0-1 routing matrix A such that:

$$a_{jr} = \left\{ \begin{array}{ll} 1 & \text{if resource } j \text{ is on route } r \\ 0 & \text{otherwise} \end{array} \right.$$

Figure 5 shows an example with three users over a network of seven links: the first user ("blue" flow) uses the route made of links 1, 4, and 6; the second user ("red" flow) uses the route made of links 2, 5, and 7; and the third user ("green" flow) uses the route made of links 1, 4, and 7. So the routing matrix has seven rows and three columns.

---

[3]Given each user has one flow over a single path, we use the terms "user", "flow", and "route" interchangeably.

167
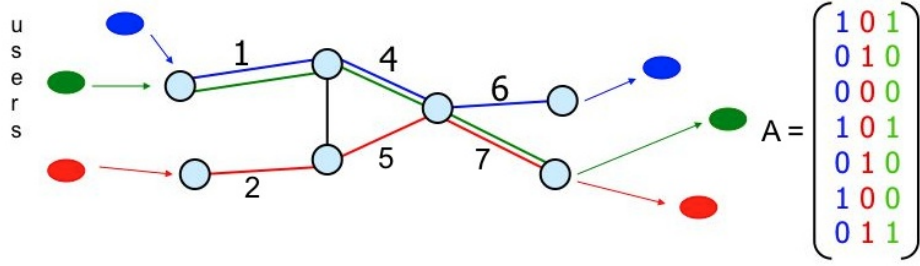
Figure 5: A network model

2.3 The Optimization Problem

Now we are ready to formulate a (centralized) optimization problem that allows the network to allocate rates to users so that the sum of their utilities is maximized [12]. We refer to this problem as $SYSTEM(U, A, C)$ where the inputs are the user utility functions $U_r(.)$, the routing matrix A, and the (column) vector of link capacities C, and the output is the (column) vector of allocated rates x.

$SYSTEM(U, A, C):$

$$\max \sum_{r \in R} U_r(x_r)$$
$$\text{subject to } Ax \leq C$$
$$\text{over } x \geq 0$$

For such an optimization problem, it is known that there exists a unique solution. This is the case because the function to optimize is strictly concave and the link capacity inequality constraints $Ax \leq C$ form a so-called *convex set* (see Figure 6.)
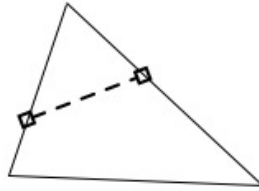
Figure 6: A convex set. A convex set intuitively means that any linear combination of any two points located on the boundary of the region, which is formed by the linear inequalities, lies within the region itself.

The practical challenge in solving this problem however is that the network does not know the utilities of its users, let alone its centralized nature makes it computationally expensive to solve!

To address these challenges, we start by decomposing the problem into R problems, one for each user $r \in R$, and one problem for the network (we will later decompose this network problem further into individual

resource problems). The network will present each user with a "price" λ_r (\$/bit). Through these prices, the network attempts to infer user utilities. Specifically, observing λ_r, user r will then choose an amount to pay w_r (\$/second) for the service (that maximizes the user's utility), which in turn determines how much rate x_r (bits/second) the user would get ($x_r = w_r/\lambda_r$). The network sets its prices λ_r based on the load x_r, $\forall r$.

2.4 Introducing Prices

The decomposed optimization problem can then be stated in terms of the following user optimization problem, and network optimization problem.

$USER_r(U_r, \lambda_r):$

$$\max U_r(\frac{w_r}{\lambda_r})$$
$$\text{over } w_r \geq 0$$

where $\frac{w_r}{\lambda_r} = x_r$. Given the network price λ_r and its own *private* utility function U_r, user r determines how much it is willing to pay w_r so as to maximize her own utility.

Knowing the vector $W = \{w_r, \forall r\}$, its routing and capacity matrices, the network allocates user rates x_r by optimizing some network function $f(x, W)$. Once x_r's are obtained, prices are obtained as $\lambda_r = \frac{w_r}{x_r}$.

$NETWORK(A, C, W):$

$$\max \sum_{r \in R} f(x_r, w_r)$$
$$\text{subject to } Ax \leq C$$
$$\text{over } x \geq 0$$

2.5 Network Optimization

The choice of the network function $f(x, W)$ determines how the capacity of the network gets allocated to users, and so how *fair* we might consider this allocation to be! For example, consider the following function:

$$f = \sum_{r \in R} w_r \, x_r$$

Maximizing this function results in maximizing the total weighted throughput for all users. As a special case, for unit weights, the network optimization problem maximizes the total throughput through the network. This might seem to fly in the face of what we think is fair! Consider the following simple example (see Figure 7): given both links have capacities of 6 units, the total throughput allocated to all users is the total network capacity of 12 units. This can be achieved by allocating 6 units of capacity to each of the 1-link flows (users): the "red" user and the "blue" user, leaving the 2-link ("green") flow with no capacity allocated to its user. That does not seem "fair"! A different function f would allocate rates to users differently and so it would provide a different notion of fairness.

But, the big question is: *how do (should) we define fairness?* The research literature introduces many notions of fairness, most notably the so-called *max-min fairness*.
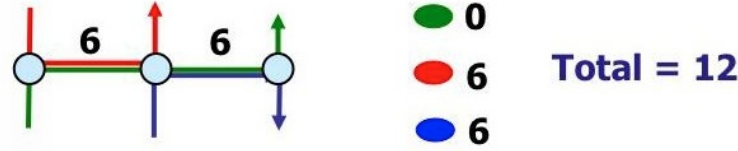
Figure 7: Greedy network allocation

2.5.1 Max-min Fairness

Intuitively, max-min fairness means that resources (link capacities) are allocated to users (flows) so that the allocation is:

1. *fair:* all users get equal share of a link, as long as users have demand that would fully consume their share, and

2. *efficient:* each link is utilized to the maximum load possible.

In other words, if a user cannot consume its equal share of a link, then the excess capacity must be (recursively) allocated equally among high-demanding users. So, the final outcome is that low-demanding users get exactly what they need, while high-demanding users get equal allocations. Consider the following multi-link network example (see Figure 8): all links have capacities of 150 units and we assume elastic traffic
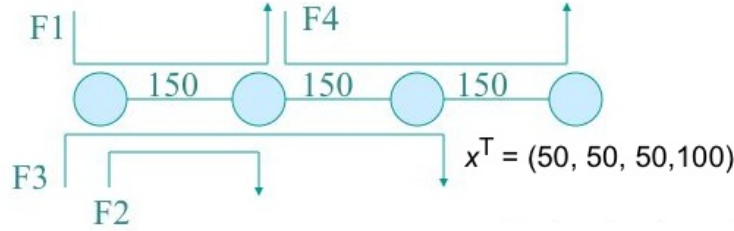
Figure 8: Max-min fair capacity allocation

sources, i.e., sources that would consume all what they can get. We start with the first (left-most) link since it is used by most users so it is the most loaded one. Each flow using that link gets allocated an equal share of $150/3 = 50$ units. Proceeding to the next loaded link, the middle one, each of its two flows should get an equal share of 75, however flow F_3 is limited by its first link to 50 units of throughput. Thus, flow F_4 gets the left-over from F_3 to a total allocation of $75 + 25 = 100$. The right-most link, at capacity of 150, does not limit the throughput of F_4, which ends up using only 100 units of that link, leaving 50 unused. At the end of this process, we say that the max-min fair allocation vector is $x^{\mathrm{T}} = (50,\ 50,\ 50,\ 100)$.

Mathematically, max-min fairness is achieved when the network maximizes the following function:

$$f = \min_{r \in R} x_r$$

Intuitively, maximizing the minimum of allocated rates results in equalizing these rates, as long as users have enough demand that will consume these rates over the network.

170

2.5.2 Proportional Fairness

Another equally popular fairness definition is the so-called *(weighted) proportional fairness*. This notion of fairness is achieved when the network maximizes the following function:

$$f = \sum_{r \in R} w_r \log(x_r)$$

Note that the log function is a concave, and strictly increasing function. Thus, given optimal rate allocation solution x^*, that is feasible, i.e., $x^* \geq 0$ and $A\, x^* \leq C$, any other feasible solution x will cause the aggregate proportional change $\sum_{r \in R} w_r \frac{x_r - x_r^*}{x_r^*}$ to be less than or equal zero. To show this, for simplicity, assume one user and unit weight, so $f(x) = \log(x)$. Expanding $f(x)$ into its first-order (linear) Taylor's approximation around x^*, we obtain:

$$f(x) \approx f(x^*) + (x - x^*)\dot{f}(x^*)$$

Given the derivative $\dot{f}(x^*) = \frac{1}{x^*}$, we have:

$$f(x) \approx f(x^*) + \frac{(x - x^*)}{x^*}$$

Since f is maximized at x^*, $f(x^*) \geq f(x)$ and so the proportional fairness condition must hold:

$$\frac{x - x^*}{x^*} \leq 0$$

Note that the presence of weight w_r intuitively means that user (flow) r is equivalent to w_r users with unit weight each.

2.5.3 General Parameterized Utility

If the network function $f(x)$ is a function of the utilities of its users $U(x)$, then the network is in fact maximizing a function of user utilities. Assuming each user r has unit weight w_r, $U_r(x_r)$ can be generalized as [18]:

$$U_r(x_r) = \frac{x_r^{1-\alpha}}{1 - \alpha}$$

where α is a parameter that determines the fairness criterion of the network. More specifically, if $\alpha \to 0$, then a user's utility is linear in its allocated rate and the network is effectively maximizing the sum of user utilities $\sum_{r \in R} U_r(x_r) = \sum_{r \in R} x_r$, which in turn yields a greedy allocation that maximizes the total throughput over the network.

On the other hand, if $\alpha \to 1$, then this is equivalent to a log utility, yielding proportional fair allocation. To see this, let's take the derivative of $U_r(x_r)$:

$$\dot{U}_r(x_r) = \frac{(1-\alpha)x_r^{-\alpha}}{1 - \alpha} \to \frac{1}{x_r} \text{ as } \alpha \to 1$$

By integrating $\dot{U}_r(x_r)$, we get back $U_r(x_r) = \log(x_r)$.

Similarly, it can be shown that $\alpha \to \infty$ corresponds to a minimum utility, yielding a max-min fair allocation.

2.6 Solution to Optimization Problem

Consider the case where the network is maximizing the weighted sum of the log of user rates, i.e., the network is trying to solve the following optimization problem that would yield a weighted proportional fairness allocation:

$NETWORK(A, C, W)$:

$$\max \sum_{r \in R} w_r \, \log(x_r)$$
$$\text{subject to } Ax \leq C$$
$$\text{over } x \geq 0$$

We can solve this problem using the theory of constrained convex optimization using the Lagrangian technique. Specifically, we move the constraints into the objective function that we want to optimize, thus making the optimization problem effectively unconstrained. We do so by introducing so-called "Lagrangian multipliers" into the new objective (Lagrangian) function L:

$$\max L = \sum_{r \in R} w_r \, \log(x_r) + \lambda^T (C - Ax)$$

The (row) vector λ^T is a Lagrangian vector with a variable λ_j for each link j in the network. Note that L is a strictly concave function, thus a solution exists at which the derivatives of L with respect to each x_r and each λ_j are equal to zero:

$$\frac{\partial L}{\partial x_r} \;\; = \;\; \frac{w_r}{x_r} - \sum_{j \in r} \lambda_j \tag{1}$$

$$\frac{\partial L}{\partial \lambda_j} \;\; = \;\; (C_j - \sum_{r \in j} x_r) \tag{2}$$

The notation $j \in r$ indicates all links j used by user (flow/route) r, whereas $r \in j$ denotes all flows r using link j, i.e. the total load on link j.

By equating the first set of equations (1) to zero, we obtain the (weighted proportionally fair) solution:

$$x_r = \frac{w_r}{\sum_{j \in r} \lambda_j} \tag{3}$$

We obtain λ_j by also equating the second set of equations (2) to zero. Note that λ_j and $(C_j - \sum_{r \in j} x_r)$ must be greater than or equal to zero since negative values do not maximize the objective function L! Furthermore, $(C_j - \sum_{r \in j} x_r) \geq 0$ ensures that the link capacity constraints $\sum_{r \in j} x_r \leq C_j$ are automatically satisfied. If $(C_j - \sum_{r \in j} x_r) = 0$ then λ_j can be greater than zero. On the other hand, if $\lambda_j = 0$, then the associated link may not be fully utilized, i.e. $\sum_{r \in j} x_r < C_j$. Intuitively, λ_j represents the "cost" associated with link j, so it is zero if the link is under-utilized, and positive if the link is allocated to capacity.

Example: Consider the example in Figure 7 but now assume the network's objective of proportionally allocating its capacity, i.e.,

$$\max f = \log(x_0) + \log(x_1) + \log(x_2)$$

subject to:

$$x_0 + x_1 \leq 6$$
$$x_0 + x_2 \leq 6$$
$$x_0, x_1, x_2 \geq 0$$

where x_0, x_1, and x_2 are the rates allocated to the two-link flow (user), the first-link flow, and the second-link flow, respectively. [4]

Using the Lagrangian's solution method, we obtain:

$$\max L = \log(x_0) + \log(x_1) + \log(x_2) + \lambda_1(6 - (x_0 + x_1)) + \lambda_2(6 - (x_0 + x_2))$$

Taking derivatives, we obtain:

$$\frac{\partial L}{\partial x_0} = \frac{1}{x_0} - (\lambda_1 + \lambda_2)$$

$$\frac{\partial L}{\partial x_1} = \frac{1}{x_1} - \lambda_1$$

$$\frac{\partial L}{\partial x_2} = \frac{1}{x_2} - \lambda_2$$

$$\frac{\partial L}{\partial \lambda_1} = 6 - (x_0 + x_1)$$

$$\frac{\partial L}{\partial \lambda_2} = 6 - (x_0 + x_2)$$

Equating these derivatives to zero, the last two equations show full utilization of the link capacities and that $x_1 = x_2$, while the first three equations give the following values of x_i's:

$$x_1 = x_2 = \frac{1}{\lambda_1} = \frac{1}{\lambda_2} = \frac{1}{\lambda}$$

$$x_0 = \frac{1}{2\lambda}$$

Substituting in the capacity equations, we obtain the price of each link λ:

$$\frac{1}{2\lambda} + \frac{1}{\lambda} = 6$$

Thus, $\lambda = \frac{1}{4}$, and so $x_0 = 2$, and $x_1 = x_2 = 4$. Note that in this optimal case, each link is fully utilized to capacity, and the flow that traverses two links is charged twice for each link it traverses and so it gets allocated a lower rate.[5] **End Example.**

---

[4]Note that since the objective (log) function is strictly increasing, then the x_i's should be as large as possible to consume the total capacity of the links, so the two inequalities on link capacities could be turned into equalities.

[5]As we will later see, this proportional rate allocation is what TCP Vegas [15] provides.

If the utility of each user r is a log function in its allocated rate x_r, then the (weighted proportionally fair) network solution $x_r = \frac{w_r}{\sum_{j \in r} \lambda_j}$ is in fact, a solution to the whole system optimization problem that includes the network, as well as all users possibly trying to independently (in a distributed way) maximize their own log utilities. However, in a distributed setting, as noted earlier, even if the network knows the user utility functions, the network allocates user rates based on their willingness to pay, w_r, which might be unknown to the network. This lack of knowledge can be overcome by observing the demand behavior of the user x_r and the price $\lambda_r = \sum_{j \in r} \lambda_j$, and so w_r is computed as $w_r = x_r \ \lambda_r$. Otherwise, the network can just assign some weights w_r to users based on some preference policy.

The moral of the story is that in practice, there is no central network controller that knows W and can then allocate rates to users. Each user and each resource (link) might have its own individual controller that will operate independently and so we need to study the collective behavior of such composite system and answer questions such as: Would the system converge (stabilize) to a solution in the long term (i.e., reaching steady state)? If so, is this solution unique and how far is it from the target (optimal) operating point? In general, if the system gets perturbed, is it stable, i.e. does it converge back to steady state, and how long does it take to converge and how smooth or rough was that? In control-theoretic terminology, we refer to the response to such perturbation until steady state is reached as the *transient response* of the system. We refer to how far the system is from being unstable, or the magnitude of perturbation that renders the system unstable, as *stability margin*.

To formally address these questions, we will resort to the modeling of user and network dynamic behaviors, in the form of differential (or difference) equations, then use well-known control-theoretic techniques to study the overall transient and steady-state behavior of the system.

3   The Control Problem

The basic control problem is to control the output of a system given a certain input. For example, we want to control the user demand (sending rate) given the observed network price (e.g., packet loss or delay). Similarly, we want to control the price advertised by a network resource given the demand (rates) of its users.

There is basically two kinds of control: open-loop control, and closed-loop (feedback) control. In open-loop control systems, there is no feedback about the state of the system and the output of the system is controlled directly by the input signal. This type of control is thus simple, but not as common as closed-loop control. An example of open-loop control system is a microwave that heats food for the input (specified) duration.

Feedback (closed-loop) control is more interesting and multiple controllers may be present in the same control loop. See Figure 2 where a user controller is present to control demand based on price, and a resource controller is also present to control price based on demand. Feedback control makes it possible to control the system well even if we can't observe or know everything, or if we make errors in our estimation (modeling) of the current state of the system, or if things change. This is because we can continually measure and correct (adapt) to what we observe (i.e., feedback signal). For example, in a congestion control system, we do not need to *exactly* know the number of users, the arrival rate of connections, or the service rate of the bottleneck resource, since each user would adapt its demand based on its own observed (measured, fed back) price, which reflects the current overall congestion of the bottleneck resource.

Associated with feedback control is a delay to observe the feedback (measured) signal, which is referred to as *feedback delay*. More precisely, feedback delay refers to the time taken from the generation of a control signal (e.g., updated user demand) until the process/system reacts to it (e.g., demand is routed over

the network), this reaction takes effect at each resource (e.g., load is observed on each link), and this effect is fed back to the controller (e.g., price is observed by the user).

3.1 System Models

Models of controlled systems can be classified along four dimensions:

- *Deterministic versus stochastic models.* The latter models capture stochastic effects like noise and uncertainties.

- *Time-invariant versus time-varying models.* The latter models contain system parameters that change over time.

- *Continuous-time versus discrete-time models.* In the latter models, time is divided into discrete-time steps.

- *Linear versus non-linear models.* The latter models contain non-linear dynamics.

In most of our treatment, we consider the simplest kind of models that are deterministic, time-invariant, continuous-time, and linear. In modeling a controlled system, we characterize the relationships among system variables as a function of time, i.e., dynamic equations. See Figure 9 where functions f and h are generally non-linear functions. The function f models the evolution of the state of the system x as a function of the current state and system's input u. The function h yields system's output y as a function of the current state and input values. As we will see later, for mathematical tractability, we often linearize dynamic non-linear models or we only consider operation in a linear regime, for example, we ignore non-linearity when the buffer goes empty or hits full capacity.

$$\dot{x} = f(x, u)$$
$$y = h(x, u)$$
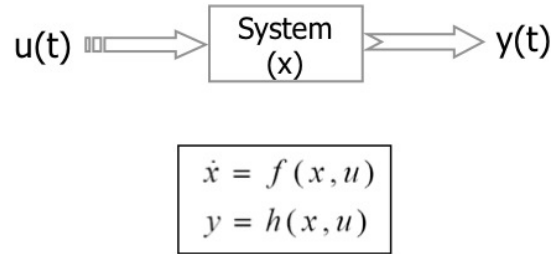
Figure 9: Typical system model

3.2 Modeling Source and Network Dynamics

Consider a source r with log utility, i.e. $U_r(t) = w_r \log(x_r)$, and a network that allocates rates in a weighted proportional fashion. We saw earlier (cf. Section 2.6) that in steady state, the (optimal) solution (Equation 3) is:

$$x_r = \frac{w_r}{\lambda_r} \tag{4}$$

175

This can be re-written as $w_r - x_r \lambda_r = 0$. Also, we saw that the optimal solution ensures that each link l is fully utilized, i.e. the load (total input rate) on link l, denoted by $y_l = \sum_{s:l \in s} x_s$, equals the link capacity c_l.

The dynamics of the sources and links can then be modeled such that these steady-state user rates and link loads are achieved. Specifically, we can write the dynamic (time-dependent) source algorithm as:

$$\dot{x}_r(t) = k[w_r - x_r(t)\lambda_r(t)] \tag{5}$$

where k is a proportionality factor. Note that w_r represents how much user r is willing to pay, whereas $x_r(t)\lambda_r(t)$ represents the cost (price) of sending at that rate. Intuitively, the user sending rate increases (decreases) when the difference between these two quantities is positive (negative). And in steady state, $\dot{x}_r(\infty) \to 0$, and so we obtain the steady-state solution $x_r = \frac{w_r}{\lambda_r}$ (as expected).

Given that the derivative of $U_r(t)$, $\dot{U}_r(t) = \frac{w_r}{x_r}$, the source rate adaptation algorithm can be re-written as:

$$\dot{x}_r(t) = kx_r(t)[\dot{U}_r(t) - \lambda_r(t)]$$

$$\dot{x}_r(t) = K(t)[\dot{U}_r(t) - \lambda_r(t)] \tag{6}$$

Intuitively, the user increases its sending rate if the marginal utility (satisfaction) is higher than the price that the user will pay, otherwise the user decreases its sending rate.

We can also write a dynamic equation for the adaptation in the link price $\lambda^l(t)$, called the link pricing algorithm:

$$\dot{\lambda}^l(t) = h(y_l(t) - c_l) \tag{7}$$

where h is a proportionality factor, and the total price, $\lambda_r(t)$, for user r, is the sum of the link prices along the user's route, i.e. $\lambda_r(t) = \sum_{l:l \in r} \lambda^l(t)$. Intuitively, the link price increases if the link is over-utilized (i.e. $y_l(t) > c_l$), otherwise the link price decreases. Note that at steady state, $\dot{\lambda}^l(\infty) \to 0$, and we obtain the steady-state optimal solution $y_l = c_l$ (as expected).

It turns out that the source and link algorithms, Equations 6 and 7, represent general user and resource adaptation algorithms that collectively determine the transient and steady behavior of the whole system. In what follows, we use the form of Equation 6 to reserve engineer different versions of TCP and deduce the utility function that the TCP source tries to maximize.

3.3 TCP and RED

Many analytical studies considered the network system composed of TCP sources over a network of queues that employ a certain queue management policy. Examples of TCP variants include Reno, SACK [5], NewReno, Vegas [15], FAST [25], etc. Examples of queue management policies include Drop Tail, RED [6], REM [2], PI [10], etc. One of the most widely studied instantiation is that of TCP sources over a RED bottleneck queue — see Figure 10. We start by modeling the dynamic behavior of a TCP source, i.e., the time-dependent relationship between its transmission window (or sending rate) and its observed loss rate or delay (price). We do so for both TCP Reno and Vegas versions, and also deduce their utilities. Then, we model the buffering process inside the network (more precisely, the bottleneck queue), assuming a linear regime (i.e., ignoring non-linearities due to the buffer becoming empty or full). Using the average buffer length, we model the dynamic behavior of RED and how it generates packet losses (or markings) as indication of congestion (price). This overall model represents the closed-loop feedback system shown in the block diagram of Figure 10, which can then be analyzed using control-theoretic techniques.
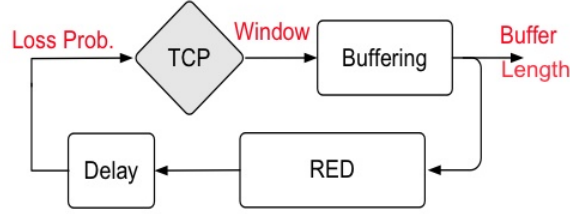
Figure 10: TCP Reno over RED feedback control system

Modeling TCP Reno: First, consider the modeling of TCP Reno, where the congestion window *cwnd* is increased by $1/cwnd$ for every acknowledged TCP segment / non-loss, i.e., it is (roughly) increased by 1 every round-trip time, and *cwnd* is decreased by half for every loss. Thus, we can write the following equation for changes in the congestion window of a single TCP flow, where p is the segment loss probability:

$$\Delta cwnd = \frac{1}{cwnd}(1 - p) - \frac{cwnd}{2}p$$

Let x denote the sending rate, and T the round-trip time, thus $x = \frac{cwnd}{T}$. Assuming acknowledgments (ACKs) come equally spaced, the time between ACKs (or lack thereof) is given by $\frac{T}{cwnd}$. Thus, we can re-write the above equation in terms of change in *rate* as:

$$\frac{d}{dt}cwnd(t) = \frac{\frac{1}{cwnd(t)}(1 - p(t)) - \frac{cwnd(t)}{2}p(t)}{\frac{T}{cwnd(t)}}$$

Dividing both sides by T, we get:

$$\frac{d}{dt}x(t) = \frac{\frac{1}{x(t)T^2}(1 - p(t)) - \frac{x(t)}{2}p(t)}{\frac{1}{x(t)}}$$

$$\frac{d}{dt}x(t) = \frac{1}{T^2}(1 - p(t)) - \frac{x(t)^2}{2}p(t) \tag{8}$$

Let's denote the loss probability $p(t)$ of TCP connection r as $p_r(t)$. $p_r(t)$ depends on the current load on path r, and can be approximated by the sum of loss probabilities experienced on individual links $j \in r$ along the connection's path. More specifically,

$$p_r(t) = \sum_{j \in r} p_j(\sum_{s:j \in s} x_s(t))$$

Assuming small p such that $(1 - p) \approx 1$, we can re-write Equation 8 as follows:

$$\frac{d}{dt}x(t) = \frac{1}{T^2} - \frac{x(t)^2}{2}p(t)$$

177

$$\frac{d}{dt}x(t) = \frac{x(t)^2}{2}\left[\frac{2}{T^2 x(t)^2} - p(t)\right] \qquad (9)$$

Comparing Equation 9 with Equation 6, we can deduce the utility function of a TCP Reno source:

$$\dot{U}(x) = \frac{2}{T^2 x^2}$$

Integrating $\dot{U}(x)$ we get:

$$U(x) = \frac{-2}{T^2 x}$$

Observe that maximizing Reno's utility results in minimizing the quantity $\frac{1}{x}$, which can be viewed as the "potential delay" as it is inversely proportional to the allocated rate x. Thus, a network allocation based on such utility is referred to as *minimum potential delay fair allocation*.

Example: Revisting the example in Figure 7 but now assume the network's objective is to allocate its capacity according to the minimum potential delay fair allocation, i.e.,

$$\max f = \frac{-1}{x_0} + \frac{-1}{x_1} + \frac{-1}{x_2}$$

subject to:

$$x_0 + x_1 \leq 6$$
$$x_0 + x_2 \leq 6$$
$$x_0, x_1, x_2 \geq 0$$

where x_0, x_1, and x_2 are the rates allocated to the two-link flow (user), the first-link flow, and the second-link flow, respectively.

Using the Lagrangian's solution method, we obtain:

$$\max L = \frac{-1}{x_0} + \frac{-1}{x_1} + \frac{-1}{x_2} + \lambda_1(6 - (x_0 + x_1)) + \lambda_2(6 - (x_0 + x_2))$$

Taking derivatives, we obtain:

$$\frac{\partial L}{\partial x_0} = \frac{1}{x_0^2} - (\lambda_1 + \lambda_2)$$

$$\frac{\partial L}{\partial x_1} = \frac{1}{x_1^2} - \lambda_1$$

$$\frac{\partial L}{\partial x_2} = \frac{1}{x_2^2} - \lambda_2$$

$$\frac{\partial L}{\partial \lambda_1} = 6 - (x_0 + x_1)$$

$$\frac{\partial L}{\partial \lambda_2} = 6 - (x_0 + x_2)$$

178

Equating these derivatives to zero, the last two equations show full utilization of the link capacities and that $x_1 = x_2$, while the first three equations give the following values of x_i's:

$$x_1 = x_2 = \frac{1}{\sqrt{\lambda_1}} = \frac{1}{\sqrt{\lambda_2}} = \frac{1}{\sqrt{\lambda}}$$

$$x_0 = \frac{1}{\sqrt{2\lambda}}$$

Substituting in the capacity equations, we obtain the price of each link $\lambda = 0.08$, and so $x_0 \approx 2.5$, and $x_1 = x_2 \approx 3.5$.

Note that in this optimal case, each link is fully utilized to capacity, and the rate allocated to a flow is inversely proportional to the square-root of the price it observes along its path.

Note also that this captures the well-known *steady-state* relationship between the throughput of a TCP Reno source and the inverse of the square-root of the *loss probability* observed by the TCP source [21]. A TCP Reno source adapting based on Equation 8 would converge to such steady-state throughput value. **End Example.**

Modeling TCP Vegas: Now, let us consider the modeling of another version of TCP — TCP Vegas [15]. This version, unlike Reno, tries to *avoid* congestion, rather than induce loss and and then adapt the transmission (congestion) window to it. The basic idea behind Vegas is to calculate the *actual* throughput of the connection as $\frac{w(t)}{T(t)}$, where $w(t)$ is the current window size, and $T(t)$ is the measured round-trip time (RTT) over the connection's path. This RTT includes queueing delay, as well as propagation delay D. Ideally, with no congestion, the ideal throughput can be computed by the source as $\frac{w(t)}{D}$, where D is estimated using the minimum RTT recently observed by the source. To ensure high utilization of the network, we want some queueing, i.e. the actual throughput is lower than the ideal one, but not too low to start causing congestion (i.e. buffer overflow at the bottleneck link resulting in losses). Vegas then adapts $w(t)$ based on some target difference, α, between the actual throughput and the ideal one. More specifically, the window increases if $(\frac{w(t)}{D} - \frac{w(t)}{T(t)}) < \alpha$, decreases if $(\frac{w(t)}{D} - \frac{w(t)}{T(t)}) > \alpha$, and stays the same otherwise. This dynamic source behavior, i.e. change in window over time, can be modeled as:

$$\frac{dw(t)}{dt} = k[\alpha - (\frac{w(t)}{D} - \frac{w(t)}{T(t)})]$$

This can be re-written as:

$$\frac{dw(t)}{dt} = \frac{k}{D}[\alpha D - (w(t) - \frac{w(t)}{T(t)}D)]$$

Denoting the sending rate (throughput) by $x(t) = \frac{w(t)}{T(t)}$, and $\gamma = \frac{k}{D}$, we have:

$$\frac{dw(t)}{dt} = \gamma[\alpha D - (w(t) - x(t)D)]$$

At steady state, as $\dot{w}(\infty) \to 0$, we have:

$$w - xD = \alpha D$$

179

Observe that the left-hand side represents the difference between the window size of packets injected by the source, and the number of packets in flight / propagating along the path (represented by the product of throughput and propagation delay). Thus, the left-hand side represents the number of packets in the bottleneck queue, and αD denotes the *target* queue occupancy of the bottleneck link. Intuitively, Vegas tries to maintain a small number of αD packets (i.e., 1-2 packets) in the bottleneck queue to maintain both small delay and high (100%) utilization. Section 4 uses control theory to analyze a Vegas-like transmission model.

Given that $x = w/T$, we get:

$$xT - xD = \alpha D$$

Denoting the queueing delay by Q, we have $T = Q + D$, and so:

$$xQ = \alpha D$$

$$x = \frac{\alpha D}{Q}$$

Comparing with Equation 4, we can deduce that the willingness to pay w_r for a Vegas user r is αD and that the price λ_r experienced by the user is the queueing delay Q.

Now, to deduce the utility function that a Vegas user tries to maximize, let us write its rate adaptation equation following Equation 5:

$$\dot{x}_r(t) = k[\alpha D - x_r(t)Q(t)]$$

$$\dot{x}_r(t) = K(t)[\frac{\alpha D}{x_r(t)} - Q(t)]$$

Thus, comparing with Equation 6, we deduce:

$$\dot{U}_r(t) = \frac{\alpha D}{x_r(t)}$$

Integrating, we obtain:

$$U_r(t) = \alpha D \log(x_r(t))$$

Recall that maximizing such user utilities results in a weighted proportional fair allocation.

Modeling RED: Let us now consider the modeling of the buffer and associated RED queue management algorithm [6]. Figure 11 shows how RED tries to avoid congestion by dropping (or marking) packets with probability p_c as a (non-linear) function of the average queue length v. First, we model the evolution of the queue length $b(t)$ as a function of the total input rate, $y(t) = \sum x_s(t)$, and (bottleneck) link capacity, C:

$$\dot{b}(t) = y(t) - C$$

Denoting by $v(t)$, the Exponential Weighted Moving Average (EWMA) of the queue length:

$$v(t + \delta) = (1 - \alpha)v(t) + \alpha b(t)$$
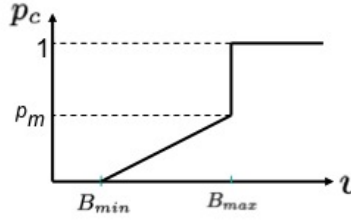
Figure 11: RED dropping (or marking) function

$$v(t + \delta) - v(t) = \alpha(b(t) - v(t))$$

Given $v(t)$ gets updated at the link rate, i.e. $\delta = \frac{1}{C}$, and $\dot{v}(t) \approx \frac{v(t+\delta)-v(t)}{\delta}$, we have:

$$\dot{v}(t) = \alpha C \ (b(t) - v(t))$$

This last equation represents the dynamic model of RED averaging, which in turn determines the price $p_c(t)$ that users experience.

To simplify the model and gain insight, let us ignore the (hard) non-linearities of the RED function and consider only the linear region:

$$p_c(t) = \sigma v(t) + \eta = \sigma \int \dot{v}(t)dt + \eta = \sigma \int \alpha C(b(t) - v(t))dt + \eta$$

where $\sigma = p_m/(B_{max} - B_{min})$, and $\eta = -p_m \ B_{min}/(B_{max} - B_{min})$.

To gain more insight, let us further ignore the RED averaging, assuming that the price is set in proportion to the actual queue length, $B_{min} = 0$ and $p_m = 1$, then we have:

$$p_c(t) = \frac{1}{B_{max}} \ b(t)$$

Differentiating both sides, we obtain:

$$\dot{p}_c(t) = h \ \dot{b}(t) = h(y(t) - C)$$

where $h = \frac{1}{B_{max}}$. Comparing with Equation 7, the packet dropping (congestion marking) probability, $p_c(t)$, represents the "price", i.e. Lagrangian multiplier, observed by users of this buffer. Note that at steady state, $\dot{p}_c(\infty) \to 0$, and so $y = C$, i.e. the link is fully utilized at steady state.

3.4 Solving the Feedback Control System

We have developed dynamic (time-dependent) models for users (sources), e.g. TCP, and the network (links), e.g. RED, and the interaction between them through prices. The next step is to solve for the transient and steady-state performance of such system. Solving such systems is challenging because of inherent non-linearities, e.g. the "hard" non-linearities (discontinuities) in the RED pricing function, or the "soft" non-linearity of TCP where the sending rate changes quadratically in the current rate. Non-linear control theory becomes a useful tool as it deals directly with non-linear differential equations. Specifically, a method

181

called *Lyapunov* [20] allows one to study convergence (stability) by showing that the value of some positive function of the state of the system continuously decreases as the system evolves over time. Finding such a Lyapunov function can be challenging, and transient performance can often only be obtained by solving the system equations numerically.

To this end, a technique called *linearization* can prove more tractable where the non-linear system is approximated by a set of *linear* equations around a single operating point (state). See Figure 12. With

$$
\begin{aligned}
\dot{x} &= f(x,u) \\
y &= h(x,u)
\end{aligned}
\quad\xrightarrow[\text{around } x_0]{\text{Linearization}}\quad
\begin{aligned}
\dot{x} &= Ax+Bu \\
y &= Cx+Du
\end{aligned}
$$

Figure 12: Linearization

linearization, we become concerned with *local stability* and study perturbations around the operating point using standard (linear) control theory. By local stability, we mean that if the system is perturbed within a small region around the operating point then the system will converge and stabilize back to that point. This is in contrast to *global stability* where the original (non-linear) system is shown to converge from *any* starting state. To linearize the non-linear system around an operating point, the basic idea is to expand the non-linear differential equation into a Taylor series around that point and then ignore high-order terms.

In what follows, we briefly review some basics of classical control theory for linear systems, then we introduce non-linear control theory. We also show examples of control theoretic analysis for the dynamic models introduced above. For more detailed background on control theory, we refer the reader to [20, 13, 16].

4 Linear Control Theory

In linear control theory, we transform differential equations in the *time domain* to algebraic equations in the so-called *frequency* or *Laplace* domains. Once this Laplace transformation is done, we use simple algebra to study the performance of the system without the need for going back to the (complicated) time domain. Specifically, we can transform a function $f(t)$ to an algebraic function $F(s)$, referred to as the Laplace transform of $f(t)$, as follows:

$$
F(s) = \int_0^\infty f(t)e^{-st}dt
$$

where s is a complex variable: $s = \sigma + j\omega$, σ is the real part of s, denoted by $Re(s)$, and ω is the imaginary part of s, denoted by $Im(s)$.

Example (Unit step function): The Laplace transform of a unit step function $u(t)$, where $u(t) = 1$ if $t > 0$, and $u(t) = 0$ otherwise, is given by:

$$
U(s) = \int_0^\infty 1.e^{-st}dt = \frac{1}{s}
$$

Example (Impulse function): The Laplace transform of a unit impulse function $\delta(t)$, where $\delta(t) = 1$ if $t = 0$, and $\delta(t) = 0$ otherwise, is given by:

$$U(s) = \int_0^\infty 1.e^{-st} dt = e^0 = 1$$

Table 1 lists basic Laplace transforms.

Table 1: Basic Laplace transforms

Impulse input: $f(t) = \delta(t)$	$F(s) = 1$
Step input: $f(t) = a.1(t)$	$F(s) = a/s$
Ramp input: $f(t) = a.t$	$F(s) = a/s^2$
Exponential: $f(t) = e^{at}$	$F(s) = 1/(s - a)$
Sinusoid input: $f(t) = \sin(at)$	$F(s) = a/(s^2 + a^2)$

Table 2 lists basic composition rules, where $\mathcal{L}[f(t)]$ denotes the Laplace transform of $f(t)$, i.e. $F(s)$.

Table 2: Composition rules

Linearity: $\mathcal{L}[a\ f(t) + b\ g(t)] = aF(s) + bG(s)$
Differentiation: $\mathcal{L}[df(t)/dt] = sF(s) - f(0) = sF(s)$ if $f(0) = 0$
Integration: $\mathcal{L}[\int f(\tau)d\tau] = F(s)/s$
Convolution: $y(t) = g(t) * u(t) = \int_0^t g(t - \tau)u(\tau)d\tau \Rightarrow Y(s) = G(s)U(s)$

Example: Consider the following second-order linear, time-invariant differential equation, where $y(t)$ represents the output of a system, and $u(t)$ represents the input:

$$a_2\ddot{y}(t) + a_1\dot{y}(t) + a_0 y(t) = b_1\dot{u}(t) + b_0 u(t)$$

In the time domain, if we represent the system by $g(t)$, then $y(t)$ can be obtained by convolving $u(t)$ with $g(t)$, i.e. $y(t) = g(t) * u(t)$. This involves a complicated integration over the system responses, $g(t - \tau)$, to impulse inputs of magnitude $u(\tau)$, for all $0 < \tau < t$.

Assuming $y(0) = u(0) = 0$, taking the Laplace transform of both sides, we obtain:

$$a_2 s^2 Y(s) + a_1 s Y(s) + a_0 Y(s) = b_1 s U(s) + b_0 U(s)$$

$$Y(s) = \frac{(b_1 s + b_0)}{(a_2 s^2 + a_1 s + a_0)} U(s) = G(s)U(s)$$

Thus, in the Laplace domain, the output $Y(s)$ can be obtained by simply multiplying $G(s)$, called the *transfer function* of the system, with $U(s)$. We can then take the inverse Laplace transform, $\mathcal{L}^{-1}[Y(s)]$, to obtain $y(t)$, or as we will later see, we can simply analyze the stability of the system by examining the roots of the denominator of the transfer function $G(s)$ and their location in the complex s-domain.

Note that because $Y(s) = G(s)$ for an impulse input, i.e. $U(s) = 1$, the transfer function $G(s)$ is also called *impulse response function*.

4.1 Modeling a Vegas-like System

Consider the system in Figure 13 where a controller G_c is used to match the queue length $b(t)$ to a target B_r by controlling the input window or sending rate $x(t)$. The output rate from the queue is denoted by $c(t)$. The goal is to first write down the differential equations that model the different components of the system,
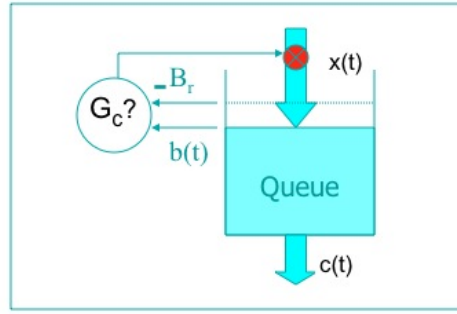
Figure 13: Vegas-like system

then instead of solving the equations in the time domain, we will transform them to the Laplace domain and analyze the stability of the system algebraically.

We start by describing the buffer evolution as:

$$\frac{d}{dt} b(t) = x(t) - c(t)$$

Then, $x(t)$ is the output of convolving the error $e(t) = B_r - b(t)$ with the controller function $G_c(t)$, i.e.

$$x(t) = G_c(t) * e(t)$$

Now, taking the Laplace transforms, we obtain:

$$sB(s) = X(s) - C(s) \Rightarrow B(s) = \frac{X(s) - C(s)}{s}$$

$$X(s) = G_c(s)E(s) = G_c(s)(B_r(s) - B(s))$$

Figure 14 shows the system using its transfer functions and their input/output flows, where $G_0 = \frac{1}{s}$. This is called the *block diagram* and provides a powerful pictorial tool. From the block diagram, one can write
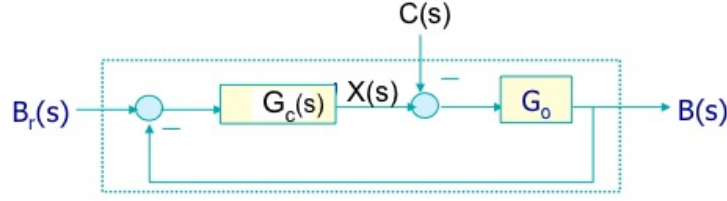
Figure 14: Block diagram of Vegas-like system

the algebraic equation of the output in terms of the input(s). Dropping the "s" parameter for convenience:

$$\frac{(B_r - B)G_c - C}{s} = B$$

Rearranging, we get:

$$B = \frac{G_c}{s + G_c}B_r - \frac{1}{s + G_c}C \tag{10}$$

Note that the system has two inputs: $B_r(s)$ and $C(s)$, subjected to two transfer functions, $\frac{G_c(s)}{s+G_c(s)}$ and $\frac{-1}{s+G_c(s)}$, respectively, and adding their responses we obtain the output $B(s)$.

4.2   Proportional Control and Stability of Vegas-like System

One basic controller G_c is referred to as *Proportional (P)* controller, where the controlled variable $x(t)$ is simply set in proportion to the error signal, i.e. $x(t) = K_p \, e(t)$. In this case, $G_c(s)$ is simply the constant K_p.

Substituting in Equation 10, we have:

$$B(s) = \frac{K_p}{s + K_p}B_r(s) - \frac{1}{s + K_p}C(s) \tag{11}$$

An important question is: *does the P-controller make the system stable?* More precisely, if we subject the system to impulse input(s), does the system converge back to a quiescent state? Control theory gives a systematic way to answer such stability question by examining the roots of the denominator of the system's transfer function, called the *characteristic equation*. In this case, the characteristic equation is:

$$s + K_p = 0 \Rightarrow s = -K_p$$

The system is stable if the roots (also called *poles*) lie in the left-hand side of the complex s-plane. Thus this system is stable if $-K_p < 0 \Rightarrow K_p > 0$.

Note that we did not need to go back to the time domain to analyze the stability of the system. But let's do that here to understand why poles on the left-hand side of the s-plane makes the system stable. Taking the inverse Laplace transform of Equation 11, and assuming impulse inputs, i.e. $B_r(s) = C(s) = 1$, we get:

$$b(t) = K_p e^{-K_p t} - e^{-K_p t}$$

185

We can then see that $b(t)$ decays exponentially over time, starting from $b(0) = (K_p - 1)$. We say that the system is stable or exhibits *overdamped response*.

We can also analyze transient performance by noting that $b(0) = (K_p - 1)$ represents an *overshoot* response to the impulse input, and that this overshoot is lower for lower K_p. See Figure 15. So by controlling K_p, referred to as the *controller gain*, we can also control the system's transient response.
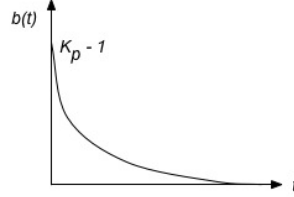
Figure 15: Overdamped response

4.3 Proportional Integral Control and Stability of Vegas-like System

Another type of controller is known as *Proportional Integral (PI)* controller where the controlled variable $x(t)$ is set in proportion to the integral of the error signal, i.e. $x(t) = K_i \int e(t)\ dt$. In this case, taking the Laplace transform, $G_c(s) = \frac{K_i}{s}$. Note that the integration means that the history of the error is used to control $x(t)$.

Substituting in Equation 10, we have:

$$B(s) = \frac{K_i}{s^2 + K_i} B_r(s) - \frac{s}{s^2 + K_i} C(s) \tag{12}$$

To analyze stability, we again examine the poles of the characteristic equation:

$$s^2 + K_i = 0 \Rightarrow s = \overset{+}{-} j\sqrt{K_i}$$

Given $K_i > 0$, the two imaginary conjugate poles lie in the left-hand side of the complex s-plane, and so the system is stable, though *critically* stable as we explain next.

To convince ourselves, let us go back to the time domain by taking the inverse Laplace transform:

$$\mathcal{L}^{-1}[\frac{K_i}{s^2 + K_i}] = \mathcal{L}^{-1}[\frac{K_i}{(s - j\sqrt{K_i})(s + j\sqrt{K_i})}] = \mathcal{L}^{-1}[\frac{A}{(s - j\sqrt{K_i})} + \frac{B}{(s + j\sqrt{K_i})}]$$

And for some values of A and B, this yields:

$$Ae^{j\sqrt{K_i}t} + Be^{-j\sqrt{K_i}t}$$

Given the fact that $e^{j\theta} = \cos\theta + j\sin\theta$, the function in the time domain oscillates in a sinusoidal fashion. Although the time function does not decay over time, it does *not* diverge, i.e. it is not unstable! So, we consider such a system to have bounded oscillations in response to impulse input and we say that it is *critically (or marginally) stable* or the system exhibits *undamped oscillatory response*. Note that a higher value of K_i results in more oscillatory behavior. See Figure 16.

186
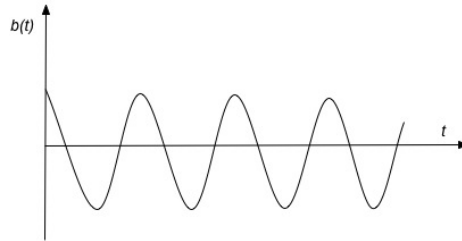
Figure 16: Undamped response

4.4 Stability and Pole Placement

More formally, a linear time-invariant system is stable if all poles of its transfer function are in the left-hand side of the s-plane, i.e. the real part of all poles is negative. Figure 17 shows the time response given the location of the poles.

Note that if the poles are complex conjugates and strictly in the left-hand side of the s-plane, the system is stable as oscillations in response to impulse input decay over time, and we say that the system exhibits *underdamped response*.
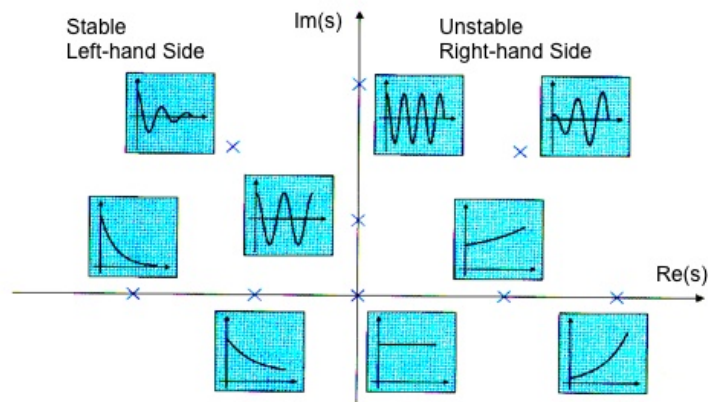
Figure 17: Time response depending on pole location

4.5 Transient Performance and Steady-state Error

Besides stability, there are other performance metrics of interest that characterize the transient performance of the system and the quality of the steady state. Figure 18 shows several of these metrics, including the time for the controlled variable to reach its peak (maximum) value, the time to reach the target, the maximum overshoot over the steady-state value, and the error that remains at steady state when the system stabilizes away from the desired target value.

187
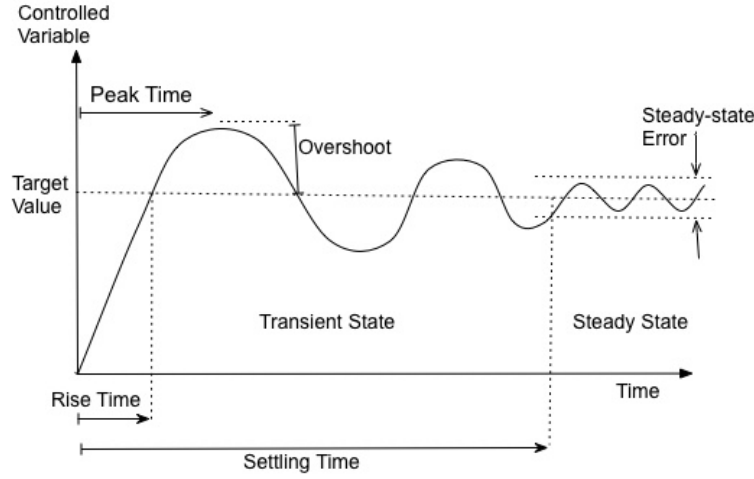
Figure 18: Performance Metrics

For our Vegas-like system, the controlled variable is the window size, i.e. number of packets allowed into the system. The response is the queue length, which we measure and compare to the target buffer size. A "good" system is one that converges quickly to the desired target with minimum oscillations (i.e., overshoots and undershoots) and with almost zero steady-state error.

4.6 Steady-state Error

In control theory, the steady-state error of a stable system is defined as:

$$e_{ss} = \lim_{t \to \infty} e(t) = \lim_{t \to \infty} (r(t) - b(t))$$

where $r(t)$ is the reference input, and $b(t)$ is the system output (response). This error reflects how accurately the system can achieve the desired target, which is chosen to be a step input.

We state without proof the Final Value Theorem [20]:

$$e_{ss} = \lim_{t \to \infty} e(t) = \lim_{s \to 0} s\, E(s)$$

This theorem allows us to calculate e_{ss} algebraically in the Laplace domain.

Example (P-control of Vegas-like system):

$$E(s) = B_r(s) - B(s)$$

Substituting for $B(s)$ from Equation 11 and using the Final Value Theorem, we obtain:

$$e_{ss} = \lim_{s \to 0} s \left[B_r(s) - \frac{K_p}{s + K_p} B_r(s) + \frac{1}{s + K_p} C(s) \right]$$

188

Assuming step inputs, i.e. $B_r(s) = \frac{B_r}{s}$ and $D(s) = \frac{D}{s}$, we have:

$$e_{ss} = \lim_{s \to 0} [B_r - \frac{K_p}{s + K_p} B_r + \frac{1}{s + K_p} C] = \frac{C}{K_p}$$

Recall that under the P-controller, the system is (overdamped) stable, i.e. $b(t)$ approaches the target B_r without oscillations, however, at steady state, $b(t)$ misses the target by $\frac{C}{K_p}$ and stabilizes at a value lower than B_r. Notice that the higher the service capacity C is, the larger the steady-state error. So, to decrease the steady-state error, the controller gain K_p could be increased. However, increasing K_p increases the overshoot. A tradeoff clearly exists between transient performance and steady-state performance, and one has to choose K_p to balance the two and meet desired operation goals. **End Example.**

Example (PI-control of Vegas-like system):

$$E(s) = B_r(s) - B(s)$$

Substituting for $B(s)$ from Equation 12 and using the Final Value Theorem, we obtain:

$$e_{ss} = \lim_{s \to 0} s \left[B_r(s) - \frac{K_i}{s^2 + K_i} B_r(s) + \frac{s}{s^2 + K_i} C(s) \right]$$

Assuming step inputs, i.e. $B_r(s) = \frac{B_r}{s}$ and $C(s) = \frac{C}{s}$, we have:

$$e_{ss} = \lim_{s \to 0} [B_r - \frac{K_i}{s^2 + K_i} B_r + \frac{s}{s^2 + K_i} C] = 0$$

Although the steady-state error is zero under the PI-controller, recall that the system is critically stable, i.e. it converges to the target while oscillating. Decreasing the controller gain K_i decreases these oscillations, however at the expense of longer time to reach steady state. This illustrates again the inherent tradeoff between transient performance and the quality of the steady state.

5   Analyzing the Stability of Non-linear Systems

As we have seen, linear control theory can be applied to non-linear systems if we assume a small range of operation around which the system behavior is linear or approximately linear. This linear analysis is simple to use, and the system, if stable, has a unique equilibrium point.

On the other hand, most control systems are non-linear, and operate over a wide range of parameters, and multiple equilibrium points may exist. In this case, non-linear control theory could be more complex to use.

In what follows, we first consider a non-linear model of the adaptation of sources and network, and use a non-linear control-theoretic stability analysis method, called *Lyapunov method* [20]. Then, we linearize the system and illustrate the application of linear control-theoretic analysis.

5.1   Solving Non-linear Differential Equations

Recall Vegas-like source adaptations from Equation 5:

$$\frac{dx_r(t)}{dt} = k[w_r - x_r(t)p_r(t)]$$

where $p_r(t)$ represents the total price observed by user r along its path. Note that this differential equation is non-linear since $p_r(t)$ is a function of the rates $x_s(t)$:

$$p_r(t) = \sum_{\text{link } l \in \text{route } r} p_l(t) = \sum_{l \in r} p_l(\sum_{s:l \in s} x_s(t))$$

We assume that the pricing function $p_l(y)$ is monotonically increasing in the load y.

At steady state, if the system stabilizes, setting the derivatives to 0, we obtain the steady-state solution:

$$k[w_r - x_r \, p_r] = 0 \Rightarrow x_r = \frac{w_r}{p_r} = \frac{w_r}{\sum_{l \in r} p_l}$$

To prove stability, we use the non-linear method of Lyapunov. The basic idea is to find a positive scalar function $V(x(t))$, we call the Lyapunov function, and show that the function monotonically increases (or decreases) over time, approaching the steady-state solution.

Define $V(x)$ as follows:

$$V(x) = \sum_{r \in R} w_r \, \log(x_r) - \sum_{j \in J} \int_0^{\sum_{s:j \in s} x_s} p_j(y) dy$$

Finding a suitable Lyapunov function that shows stability is tricky and more of an art! If you can't find one, it does *not* mean that the system is not stable. Note that this $V(x)$ has some special meaning: the first term represents the utility gain from making users happy, while the second term represents the cost in terms of price. So $V(x)$ represents the net gain. Also, note that since the first term is concave because of the log function, and the second term is assumed to be monotonically increasing, then the resulting $V(x)$ is concave, i.e. it has a maximum value.

To show that $V(x(t))$ is strictly convergent, we want to show that $\frac{dV(x(t))}{dt} > 0$, which implies that $V(x(t))$ strictly increases (i.e. the net gain keeps increasing over time), until the system stabilizes and reaches steady state when $\frac{dV(x(t))}{dt} = 0$ (i.e. the net gain $V(x)$ reaches its maximum value).

First, we note:

$$\frac{\partial V(x)}{\partial x_r} = \frac{w_r}{x_r} - \sum_{j \in r} p_j(\sum_{s:j \in s} x_s)$$

Then:

$$\frac{V(x(t))}{dt} = \sum_{r \in R} \frac{\partial V(x(t))}{\partial x_r} \frac{dx_r(t)}{dt}$$

$$\frac{V(x(t))}{dt} = \sum_{r \in R} [\frac{w_r}{x_r} - \sum_{j \in r} p_j(\sum_{s:j \in s} x_s(t))] \, k[w_r - x_r(t) p_r(t)]$$

$$\frac{V(x(t))}{dt} = \sum_{r \in R} k \, x_r(t) [\frac{w_r}{x_r} - \sum_{j \in r} p_j(\sum_{s:j \in s} x_s(t))]^2 > 0$$

Observe that this non-linear stability analysis shows that the system is stable, no matter what the initial state $x(0)$ is. This property is referred to as *global stability*, which is in contract to *local stability* that we prove when the system is linearized locally around a certain operating point as we will see next.

190

5.2 Linearizing and Solving Linear Differential Equations

As noted earlier, finding Lyapunov functions to prove global stability of non-linear control systems, even for simple models, is challenging. For example, consider more sophisticated models with feedback delay, different regions of TCP operation (e.g., timeouts, slow-start), queue management with different operating regions (e.g., RED), and challenging or adversarial environments (e.g., exogenous losses over wireless links or due to DoS attacks).

Using linearization, we can separately study simpler linear models around the different points (regions) of operation. More specifically, we linearize the system around a single operating point x^* and study perturbations around x^*, i.e. if we perturb the system away from x^* to a point $x(0)$ such that the initial perturbation $\delta x(0) = x(0) - x^*$, we want to show that $\delta x(t)$ diminishes over time and the system returns to its original state x^*, i.e. $\delta x(t) \to 0$. In this case, we say that the system is *locally stable* around x^*.

Let's consider again the Vegas-like source adaptation and assume, for simplicity, a single user over a single resource:

$$\frac{dx(t)}{dt} = k[w - x(t)p(x(t))]$$

Define the perturbation $\delta x(t) = x(t) - x^*$. Then we can write:

$$\frac{d(\delta x(t) + x^*)}{dt} = k[w - (\delta x(t) + x^*)p((\delta x(t) + x^*))]$$

Expanding the non-linear term $p((\delta x(t) + x^*))$ into its first-order Taylor series:

$$p((\delta x(t) + x^*)) \approx p(x^*) + \dot{p}(x^*)\delta x(t)$$

Substituting with this linear approximation, we get:

$$\frac{d\delta x(t)}{dt} = k[w - (\delta x(t) + x^*)\{p(x^*) + \dot{p}(x^*)\delta x(t)\}]$$

$$\frac{d\delta x(t)}{dt} = k[w - \delta x(t)p(x^*) - x^*p(x^*) - \dot{p}(x^*)\delta^2 x(t) - \dot{p}(x^*)x^*\delta x(t)]$$

If x^* is the optimal steady-state point, we know that $w - x^*p(x^*) = 0$ (cf. Equation 4). Also, given small perturbation $\delta x(t)$, $\delta^2 x(t) \approx 0$. Then, we have:

$$\frac{d\delta x(t)}{dt} = k[-\delta x(t)p(x^*) - \dot{p}(x^*)x^*\delta x(t)]$$

$$\frac{d\delta x(t)}{dt} = -k[p(x^*) + x^*\dot{p}(x^*)]\,\delta x(t)$$

Let $k[p(x^*) + x^*\dot{p}(x^*)] = \gamma$, we have:

$$\frac{d\delta x(t)}{dt} = -\gamma\,\delta x(t) \tag{13}$$

This is now a linear differential equation, which unlike the original non-linear differential equation, we can easily study using linear control-theoretic techniques, or in this simple case, solve by straightforward integration:

$$\int_0^t \frac{d\,\delta x(t)}{\delta x(t)} = -\gamma \int_0^t dt$$

$$\log(\delta x(t)) - \log(\delta x(0)) = -\gamma t$$

$$\log(\frac{\delta x(t)}{\delta x(0)}) = -\gamma t$$

$$\delta x(t) = \delta x(0)\,e^{-\gamma t}$$

Note that from this time-domain analysis, the system is shown to be stable, i.e. the perturbation vanishes over time and the system returns to its original state x^*. We also observe that the system response decays exponentially from its original perturbation $\delta x(0)$, i.e. without oscillations, and so the response is classified as *overdamped*.

If the linearized differential equation modeling the system were more complicated, it is much easier to transform it into the Laplace domain and analyze the system algebraically. Denoting $\delta x(t)$ by $u(t)$, the Laplace transform of $\delta x(t)$ by $U(s)$, and taking the Laplace transform of Equation 13, we get:

$$s\,U(s) - u(0) = -\gamma\,U(s)$$

$$U(s) = \frac{u(0)}{s + \gamma}$$

For stability analysis, we examine the location of the poles (roots) of the characteristic equation $s + \gamma = 0$, yielding the pole $s = -\gamma$. Since the pole is strictly in the left-side of the s-plane, given $\gamma > 0$, the system is stable and its response is overdamped.

To evaluate the steady-state error, we define the error as $e(t) = u(0) - u(t)$, and applying the Final Value Theorem with an impulse perturbation of magnitude $u(0)$, i.e. $U(0) = u(0)$, we obtain:

$$e_{ss} = \lim_{s \to 0} sE(s) = \lim_{s \to 0} s[u(0) - \frac{u(0)}{s + \gamma}] = 0$$

So, there is no steady-state error.

5.2.1 Effect of Feedback Delay and Nyquist Stability Criterion

As we just noted above, the power of solving the linearized model in the Laplace domain comes when the model is even slightly more complicated. For example, let us consider a feedback delay T such that Equation 13 looks like:

$$\frac{du(t)}{dt} = -\gamma\,u(t - T)$$

Taking the Laplace transform, and noting that the Laplace transform of a delayed signal $u(t - T)$ is $e^{-sT}U(s)$, we obtain:

$$sU(s) - u(0) = -\gamma\, e^{-sT}U(s)$$

$$U(s) = \frac{u(0)}{s + \gamma\, e^{-sT}}$$

Then, the characteristic equation is:

$$s + \gamma\, e^{-sT} = 0 \tag{14}$$

which we need to solve to locate the poles and determine the stability of the system.

To solve such characteristic equation, we resort to another control-theoretic method called the *Nyquist stability criterion* [20]. To this end, we introduce, without proof, the *Cauchy's principle* [20], which states that given $F(s)$, and we plot $F(s)$ as we vary s along a certain contour (trajectory) in the s-plane — see Figure 19 — and denote the following:

- Z: the number of *zeros* of $F(s)$, i.e. the roots of the numerator of $F(s)$, inside the contour.

- P: the number of *poles* of $F(s)$, i.e. the roots of the denominator of $F(s)$, inside the contour.

- N: the number of times that the plot of $F(s)$ encircles the origin in the $F(s)$-plane, such that an encirclement is negative if it is in the opposite direction of the s-contour.[6]

Then the following relationship holds:

$$Z = P + N$$
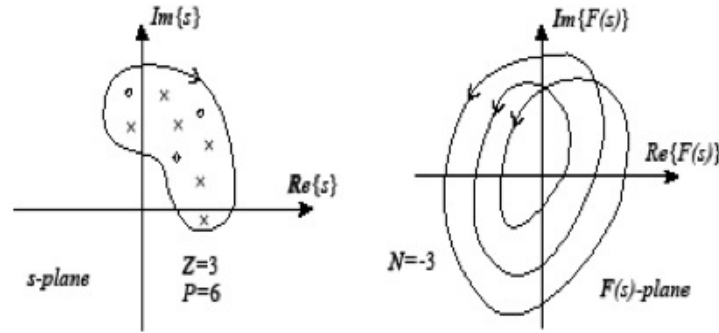
Figure 19: Cauchy's principle

---

[6] The example in Figure 19 shows that $N = -3$. The value of N is negative because the s-contour that is mapped to $F(s)$ is clockwise, whereas the resulting $F(s)$ plot is counter-clockwise. Moreover, the $F(s)$ path encircles the origin in the $F(s)$-plane three times – drawing a line emanating from the origin intersects the $F(s)$ path three times.

The Nyquist method applies the Cauchy's principle as follows. Let's say we want to analyze the stability of a closed-loop control system whose forward transfer function is $G(s)$ and its feedback transfer function is $H(s)$—see Figure 20. Then, the closed-loop transfer function is given by $\frac{G(s)}{1+G(s)H(s)}$, where $G(s)H(s)$ is referred to as the *open-loop* transfer function. The characteristic equation is given by: $F(s) = 1 + G(s)H(s) = 0$. Observe that the zeros of $F(s)$ are the closed-loop poles, and the poles of $F(s)$ are the poles of $G(s)H(s)$ (so-called "open-loop" poles).
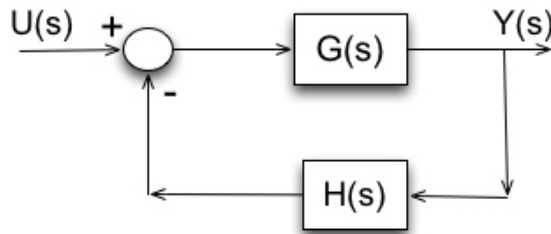
Figure 20: Typical closed-loop control system

By taking the s-contour to be around the right-side (i.e. unstable side) of the s-plane (see Figure 21), and noting the number of *unstable* open-loop poles P and the number of encirclements N around the origin in the $F(s)$-plane, we determine the number of *unstable* zeros Z of $F(s)$, i.e. number of unstable closed-loop poles, using Cauchy's relationship: $Z = P + N$. If $P = 0$ and $N = 0$, then $Z = 0$ implies that there are no *unstable* closed-loop poles and so the closed-loop system is stable.[7]
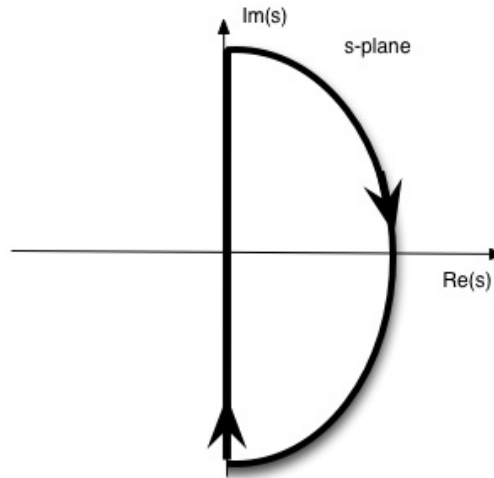
Figure 21: Contour around the unstable right-side of the s-plane

---

[7] Note that a pole on the imaginary axis is not considered unstable and the s-contour avoids such a pole and we show it as a small circle around it.

This process can be slightly simplified if instead of plotting $F(s)$, we instead plot the open-loop transfer function: $G(s)H(s)$ and observe its encirclements of the $(-1, j0)$ point in the $G(s)H(s)$-plane, instead of the origin $(0, j0)$ in the $F(s)$-plane. Given there are no poles of $G(s)H(s)$ in the right-side of the s-plane, i.e. $P = 0$, in order for the closed-loop system to be stable, the plot of $G(s)H(s)$ should not encircle -1 as we vary s along the contour enclosing the right-side of the s-plane. We are mostly interested in varying s along the imaginary axis, i.e. $s = j\omega$ where ω varies from 0 to ∞. This is because the plot for ω from $-\infty$ to 0 is symmetric, and the plot for the semi-circle as $s \to \infty$ maps to the origin in the $G(s)H(s)$-plane. Thus, we are interested in plotting $G(j\omega)H(j\omega)$ as ω varies from 0 to ∞.

Example: Let's go back to the characteristic equation in Equation 14:

$$s + \gamma\, e^{-sT} = 0 \Rightarrow F(s) = 1 + \frac{\gamma}{s}e^{-sT} \Rightarrow G(s)H(s) = \frac{\gamma}{s}e^{-sT}$$

Note that $G(s)H(s)$ does not have any unstable poles, i.e. $P = 0$. In particular, $s = 0$ is considered a (critically) stable pole.

Ignoring the constant factor γ for now, we want to plot:

$$\frac{e^{-j\omega T}}{j\omega} \qquad \omega : 0 \to \infty$$

Noting that $e^{j\theta} = \cos\theta + j\sin\theta$, we have:

$$e^{-j\omega T} = \cos(\omega T) - j\sin(\omega T)$$

Then,

$$\frac{e^{-j\omega T}}{j\omega} = -j\frac{\cos(\omega T)}{\omega} - \frac{\sin(\omega T)}{\omega}$$

Since we are interested in determining intercepts with the real-axis of $G(j\omega)H(j\omega)$ and whether they occur to the right or left of -1 (see Figure 22), we want to determine the values of ω for which the imaginary part of $G(j\omega)H(j\omega)$, i.e. $-\frac{\cos(\omega T)}{\omega}$, is zero. Such intercepts occur when $\omega T = \frac{\pi}{2}, \frac{3\pi}{2}, \cdots$, when the cosine value is zero.

Now, at these values of ωT, we can determine the points of interception along the real-axis, i.e. the magnitude $|G(j\omega)H(j\omega)|$ when the plot intercepts the real-axis:

$$-\frac{\sin(\omega T)}{\omega} = -\frac{1}{\frac{\pi}{2T}}, +\frac{1}{\frac{3\pi}{2T}}, \cdots$$

For the system to be stable, $|G(j\omega)H(j\omega)|$ at these intercepts must be less than 1, so the $G(s)H(s)$ plot does not encircle -1. This is the case if after restoring the constant factor γ we initially ignored, the following condition holds:

$$\gamma\frac{2T}{\pi} < 1$$

End Example.

Observe that T is the feedback delay, so as T gets larger, it becomes harder to satisfy the stability condition. Intuitively, this makes sense since a larger feedback delay results in outdated feedback (measurements) and it becomes impossible to stabilize the system. This is the fundamental reason why TCP over long-delay paths does not work, and architecturally, control has to be broken up into smaller control loops.
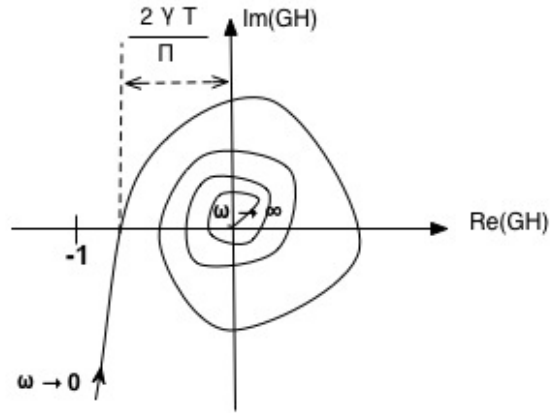
Figure 22: Example showing the effect of feedback delay

6 Routing Dynamics

So far, we assumed routes taken by flows to be static. In general, routes may also be adapted based on feedback on link prices (reflecting load, delay, etc.), albeit over a longer timescale of minutes, hours or even days compared to that of milliseconds for sending rate adaptation. Figure 23 shows a block diagram that includes both route and sending rate adaptation.
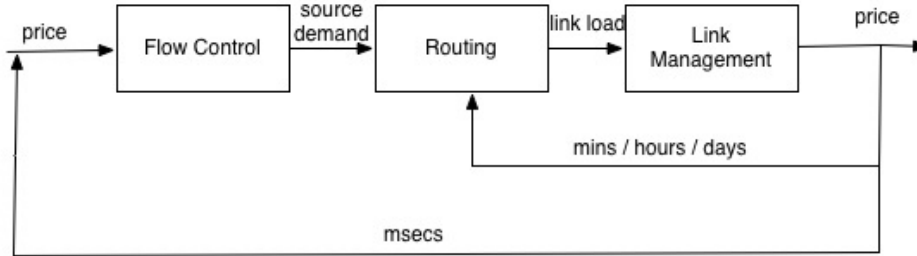
Figure 23: Block diagram with both flow and routing control

Figure 24 illustrates the general process of adaptation. Flow or routing control determines the amount of load directed to a particular link based on the link's observed price — relative to that of other possible links on alternate routes in the case of routing. We call this mapping from link price λ to link load x, the *response function $G(\lambda)$*. Given link load, a certain price is observed for the link. We call such load-to-price mapping, the *pricing (feedback) function $H(x)$*. The process of adaptation is then an iterative process:

$$\lambda = H(x)$$
$$x = G(\lambda)$$
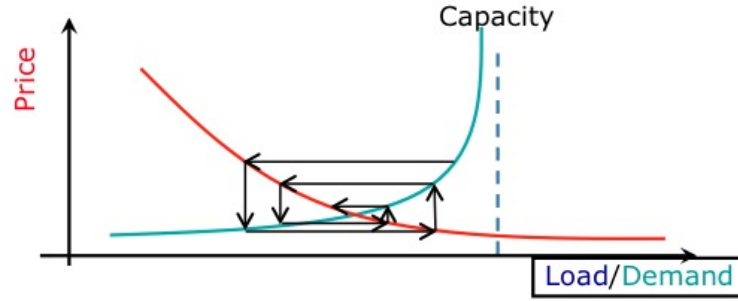
Figure 24: Convergence

We can then write:

$$\lambda = H(G(\lambda)) = F(\lambda)$$

where $F(\lambda)$ is an *iterative* function whose stable (fixed) point λ^* is the intersection of the response function and the pricing function. Figure 25 illustrates convergence to a fixed point. Starting from an initial λ_0, we find $F(\lambda_0)$, then projecting on the 45° line we obtain $\lambda_1 = F(\lambda_0)$, which we use to find $F(\lambda_1)$, and this iterative process continues until we reach the fixed point.
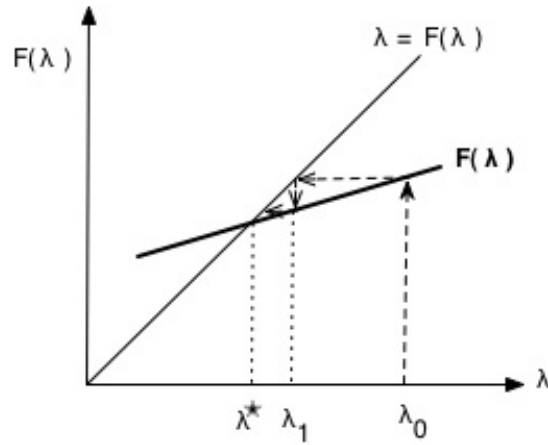
Figure 25: Contractive mapping

In order to converge to that fixed point, $F(\lambda)$ must be a so-called *contractive mapping*. Intuitively, $F(\lambda)$ is contractive iff its slope is less than 1, i.e. $|F(\lambda_2) - F(\lambda_1)| < \alpha|\lambda_2 - \lambda_1|$, $\alpha < 1$. Figure 26 illustrates a mapping that results in divergence.

Intuitively, the use of Lyapunov functions to prove convergence tests whether the iterative process describing the evolution of the system over time is a contractive mapping, i.e. the distance to the fixed point keeps shrinking at every iteration.

197
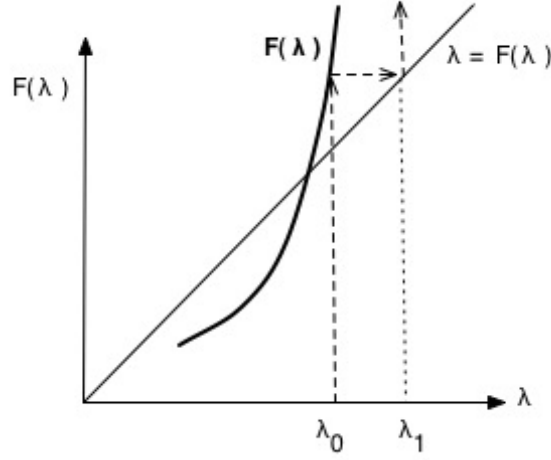
Figure 26: Divergent mapping

Example: Consider the adaptive routing of $N > 0$ unit-rate flows over two possible paths whose prices are given by *monotonically increasing* functions $p_1(x)$ and $p_2(N - x)$, where x represents the number of flows (or load) routed on the first path. Note that x completely defines the state of the system. Also, assume that routing to the least-loaded path is done gradually, to avoid wild oscillations, where $0 < \alpha < 1$ of the flows are re-routed. Using a discrete-time model where routes are adapted at discrete-time steps, we can write the following difference equations:

$$x(t + 1) = \begin{cases} x(t) + \alpha \left[N - x(t)\right] & \text{if } p_1(x(t)) \leq p_2(N - x(t)) \\ x(t) - \alpha \, x(t) & \text{otherwise} \end{cases}$$

At steady state, this system might converge to one of two possible stable (fixed) points. One possibility is obtained when substituting with $x(t) \to x^*$ in the first difference equation: $x(t) \to x^* \Rightarrow x^* = x^* + \alpha[N - x^*] \Rightarrow x^* = N$, so all traffic will end up getting routed on the first path. A *necessary* condition to reach that $x^* = N$ fixed point is that $p_1(N) \leq p_2(0)$, i.e. the first path is least loaded (priced) even when all N flows are on it.

Another possibility is obtained when substituting with $x(t) \to x^*$ in the second difference equation: $x(t) \to x^* \Rightarrow x^* = x^* - \alpha x^* \Rightarrow x^* = 0$, so all traffic will end up getting routed on the second path. A *necessary* condition to reach that $x^* = 0$ fixed point is that $p_1(0) > p_2(N)$, i.e. the second path is least loaded (priced) even when all N flows are on it.

We can show convergence to one of these fixed points depending on which necessary condition holds: $p_1(N) \leq p_2(0)$ or $p_1(0) > p_2(N)$.

Let's assume $p_1(N) \leq p_2(0)$ holds. We want to define a Lyapunov function $V(x) \geq 0$ and show that $V(x(t + 1)) \leq V(x(t))$ for some or all starting state $x(0)$, i.e. $V(x)$ monotonically decreases toward the $x^* = N$ fixed point where equality holds. If there are only certain values of the starting state $x(0)$ for which the system converges then such conditions must hold, in addition to the necessary condition, for convergence to happen. In this case, we say that the necessary condition by itself is not sufficient for convergence.

Define $V(x) = N - x$. Note that $V(x) \geq 0$ because $0 \leq x \leq N$, and $V(x) = 0$ when $x = N$, i.e. at the fixed point. So, under convergence, we expect $V(x)$ to monotonically decrease toward zero. Substituting

198

for $x(t+1)$ in $V(x)$, we obtain:

$$V(x(t+1)) = N - x(t+1)$$

Given the pricing functions are monotonically increasing with load, $p_1(N) \leq p_2(0) \Rightarrow p_1(x(t)) \leq p_2(N - x(t))$, $\forall x(t)$, and we can only use the first difference equation to substitute for $x(t+1)$:

$$V(x(t+1)) = N - (x(t) + \alpha[N - x(t)]) = (1 - \alpha)(N - x(t))$$

$$V(x(t+1)) = (1 - \alpha)V(x(t)) \leq V(x(t))$$

So, we can conclude that the system is convergent regardless of the starting state $x(0)$ as long as $0 < \alpha < 1$.

Thus, $0 < \alpha < 1$, along with the necessary condition $p_1(N) \leq p_2(0)$, represent necessary and sufficient conditions for convergence.

A similar convergence proof can be obtained if on the other hand, the necessary condition $p_1(0) > p_2(N)$ holds. **End Example.**

7 Case Study: Class-based Scheduling of Elastic Flows

In this and the following section, we consider the modeling and control-theoretic analysis of two traffic control case studies. This first case study [17, 9] concerns the performance of elastic flows, i.e., rate-adaptive flows similar to TCP. The goal is to investigate the effect of class-based scheduling that isolates elastic flows into two classes (service queues) based on different characteristics, for example based on their lifetime (transfer size), or burstiness of their arrivals/departures and sending rate (window) dynamics. We want to show the benefits of isolation, in terms of better predictability and fairness, over traditional shared queueing systems.

We formulate two control models. In the first model (Section 7.1), each flow controls its input traffic rate based on the *aggregate* state of the network due to all N flows. In the second model (Section 7.2), each flow (or class of homogeneous flows) controls its rate based on its own *individual* state within the network. We assume that the flows use PI control for adapting their sending rate.

In the aggregate control model, the packet sending rate of flow i, denoted by $x_i(t)$, is adapted based on the difference between a target *total* buffer space, denoted by B, and the current *total* number of outstanding packets, denoted by $q(t)$. In the individual control model, $x_i(t)$ is adapted based on flow (or class) i's target, denoted by B_i, and its current number of outstanding packets, denoted by $q_i(t)$. We denote by $c(t)$ the total packet service rate, and by $c_i(t)$ the packet service rate of flow/class i. In what follows, for each control model, we determine conditions under which the system stabilizes. We then solve for the values of the state variables at equilibrium, and show whether fairness (or a form of weighted resource sharing) can be achieved. Table 3 lists all system variables along with their meanings.

7.1 Aggregate Control or Sharing

Under aggregate PI control, the evolution of the system state is described by the following differential equations:

$$\begin{aligned}
\dot{x}_i(t) &= \alpha_i(B - q(t)) \\
\dot{q}(t) &= \sum_{j=1}^{N} x_j(t) - c(t)
\end{aligned} \tag{15}$$

199

Table 3: Table defining system variables

Variable	Meaning
N	total number of flows (or classes of homogeneous flows)
$x_i(t)$	packet sending rate of flow/class i
$q_i(t)$	number of outstanding packets of flow/class i
$c_i(t)$	packet service rate of flow/class i
$q(t)$	total number of outstanding packets
$c(t)$	total packet service rate
B	target total buffer space
B_i	target buffer space allocated to flow/class i
α_i	parameter controlling increase and decrease rate of $x_i(t)$

Stability Condition: Without loss of generality, assume a constant packet service rate (i.e. $c(t) = C$ for all t), all flows start with the same initial input state (i.e. $x_i(0)$ is the same for all i), and that all flows adapt at the same rate (i.e. $\alpha_i = \alpha$ for all i). Then, Equations (15) can be re-written as:

$$
\begin{aligned}
\dot{x}_i(t) &= \alpha(B - q(t)) \\
\dot{q}(t) &= \sum_{j=1}^{N} x_j(t) - C
\end{aligned}
\tag{16}
$$

Since flows adapt their $x_i(t)$ at the same rate, then $x_i(t) = \frac{\sum_{j=1}^{N} x_j(t)}{N}$ for all i. Denote by $e(t)$ the error at time t, i.e. $e(t) = B - q(t)$, and let $y(t) = \sum_{j=1}^{N} x_j(t) - C$. Equations (16) can then be re-written as:

$$
\begin{aligned}
\frac{\dot{y}(t)}{N} &= \alpha\, e(t) \\
\dot{q}(t) &= y(t)
\end{aligned}
\tag{17}
$$

Taking the Laplace transform of Equations (17), and assuming the buffer is initially empty (i.e. $q(0) = 0$), we get:

$$
\begin{aligned}
\frac{1}{N}(sY(s) - y(0)) &= \alpha\, E(s) \\
s\, Q(s) &= Y(s) \\
E(s) &= B - Q(s)
\end{aligned}
\tag{18}
$$

Solving Equations (18), we obtain the closed-loop system's characteristic equation (see Figure 27 for the system's block diagram):

$$
s^2 + \alpha\, N = 0 \Rightarrow s = \overset{+}{-} j\sqrt{\alpha\, N}
\tag{19}
$$

For $\alpha > 0$, this system is marginally stable. However, the magnitude of oscillations increases for higher α and/or higher N.

This indicates that the existence of flows that rapidly change their sending rates through high values of α_i can cause the system to have high oscillations. This suggests that elastic flows that aggressively change
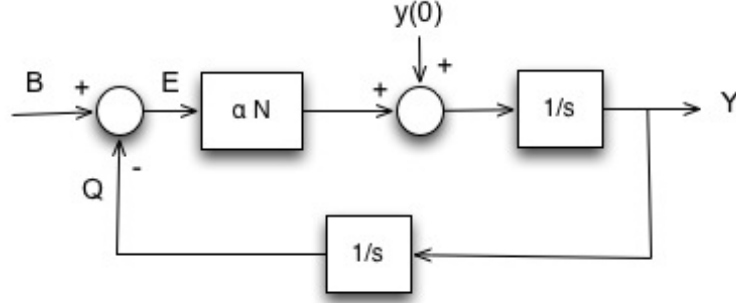
Figure 27: Block diagram of the link sharing model

their sending rates, may affect the stability of other flows that change their sending rates cautiously, in a system that mixes both kinds of flows. Furthermore, in such a system, the value of N may be high so as to cause high oscillations.

We now derive the values of the state variables at equilibrium. Denote by x_i and q the steady-state values of $x_i(t)$ and $q(t)$, respectively. Then, at equilibrium, we have from Equations (16):

$$
\begin{aligned}
0 &= \alpha(B - q) \\
0 &= \sum_{j=1}^{N} x_j - C
\end{aligned}
$$

Thus, at equilibrium, $q = B$ and $\sum_{j=1}^{N} x_j = C$. In other words, the system converges to a state where the total input rate matches the total service rate, and the target total buffer space is met.

Note that if $\alpha_i = \alpha$ for all i, then $x_i(t)$ changes at the same rate for every flow i. Consequently, if we start the evolution of the system with $x_i(0)$ being the same for all flows, only then we have equal sharing of the network at steady-state, i.e. $x_i = \frac{C}{N}$, regardless of the initial value $q(0)$. However, in general, when $x_i(0)$ are not equal for all flows, the system converges to an *unfair* state, more precisely, to a state where

$$
x_i = x_i(0) + \frac{C - \sum_{j=1}^{N} x_j(0)}{N}
$$

To summarize, controlling several flows by observing the resulting aggregate state of the network may lead to high oscillations due to either the existence of flows which are rapidly adjusting their sending rates, or a high number of flows competing for the same *shared* resource. Furthermore, the system is highly likely to converge to an unfair state where flows receive unequal shares of the resource.

7.2 Individual Control or Isolation

Under individual PI control, the evolution of the system state is described by the following differential equations:

$$
\begin{aligned}
\dot{x}_i(t) &= \alpha_i(B_i - q_i(t)) \\
\dot{q}_i(t) &= x_i(t) - c_i(t)
\end{aligned} \tag{20}
$$

Recall that under individual control, flow/class i regulates its input, $x_i(t)$, based on its *own* number of outstanding packets. For simplicity, assume a constant packet service rate, i.e. $c_i(t) = C_i$ for all t. Following the same stability analysis as aggregate control, it is easy to see that flow/class i stabilizes and the poles of the closed-loop system are:[8]

$$s = \overset{+}{-} j\sqrt{\alpha_i}$$

Observe that, unlike aggregate control, flows/classes are *isolated* from each other. Therefore, the existence of flows/classes that rapidly change their sending rates through high values of α_i, does not affect the stability of other flows. This isolation can be implemented using, for example, a class-based queueing (CBQ) discipline [7]. In such CBQ system, each class of homogeneous flows can be allocated its own buffer space and service capacity.

We now derive the values of the state variables of flow/class i at equilibrium. Denote by x_i and q_i the steady-state values of $x_i(t)$ and $q_i(t)$, respectively. Then, at equilibrium, we have from Equations (20):

$$
\begin{aligned}
0 &= \alpha_i(B_i - q_i) \\
0 &= x_i - C_i
\end{aligned}
$$

Thus, at equilibrium, $q_i = B_i$ and $x_i = C_i$. In other words, each flow/class i converges to a state where its input rate matches its allocated service rate, and its target buffer space is met. We note that if the allocated buffers B_i and service capacities C_i are equal, then every flow receives an equal share of the resources, *regardless of* the initial values $x_i(0)$ and $q_i(0)$. One can also achieve a weighted resource sharing by assigning different B_i and C_i allocations. Thus, a flow/class with higher priority (e.g., short interactive TCP flows operating aggressively in slow start) can be allocated more resources, so as to receive better throughput/delay service than other flows (e.g., long TCP flows operating cautiously in congestion avoidance).

To summarize, controlling each flow (or class of homogeneous flows) by observing its own individual state within the network provides isolation between them. Thus, stability can be achieved for a flow/class regardless of the behavior and number of other flows/classes. Furthermore, the system can converge to a fair state where flows/classes receive a weighted share of the resources.

8   Case Study: Elastic Transport Tunnel

Consider n regular user connections between sending and receiving end-hosts, all passing through two "gateways" — let's call them a source gateway and a destination gateway. Our main goal is to provide a "soft bandwidth-guaranteed" tunnel [8] for these user flows over an Internet path of bottleneck capacity C, which is also shared by another set of x flows, representing cross traffic (see Figure 28). By "soft" guarantee we mean that there is no explicit resource reservation. Consider that user and cross-traffic connections are all rate-adaptive connections (similar to TCP). These x cross-traffic connections present a challenge: as x keeps changing, the bandwidth allocation for the n user flows keeps changing in tandem. So an important question is whether it is possible to "counter" the change in x so as to ensure that the n user flows are able to maintain a desirable bandwidth.

Clearly without the intervention of the two gateways, the answer to the above question is *no*. When different flows share a link, the effect of each individual flow (or an aggregate of flows) affects the rest since all are competing for a fixed amount of resources. However, if the gateways dynamically maintain a number
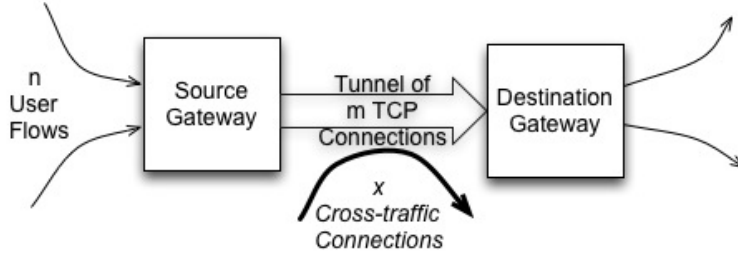
---

[8] We set $N = 1$ in Equation (19).

Figure 28: Soft bandwidth-guaranteed tunnel

m of open rate-adaptive (e.g., TCP) connections between them, they can increase m to provide a positive pressure that would equalize the pressure caused by the cross-traffic connections, if the latter occurs. Since m will be changing over time, we describe the gateway-to-gateway tunnel, made of the m connections, as *elastic*. Note that the source gateway can decide to reduce m (i.e. relieve pressure) if x goes down— the reason is that as long as the tunnel is achieving its target bandwidth, releasing extra bandwidth should improve the performance of cross-traffic connections, which is in the spirit of best-effort networking.

To illustrate this scenario and the issues involved, consider a gateway-to-gateway tunnel going through a single bottleneck link. Assuming long-lived TCP-like load, the behavior of the bottleneck can be approximated by Generalized Processor Sharing (GPS) [22], i.e. each connection receives the same fair share of resources [3]. Thus, each connection ends up with $\frac{C}{m+x}$ bandwidth. This, in turn, gives the m gateway-to-gateway rate-adaptive flows, or collectively the elastic gateway-to-gateway tunnel, a bandwidth of $\frac{Cm}{m+x}$. As the source gateway increases m by opening more rate-adaptive connections to the destination gateway, the tunnel can grab more bandwidth. If x increases, and the gateways measure a tunnel's bandwidth below a target value (say B^*), then m is increased to push back cross-traffic connections. If x decreases, and the gateways measure a tunnel's bandwidth above B^*, then m is decreased for the good of cross-traffic connections. It is important to note that the source gateway should refrain from unnecessarily increasing m, thus achieving a tunnel's bandwidth above B^*, since an unnecessary increase in the total number of competing rate-adaptive flows reduces the share of each connection and may cause flows to timeout leading to inefficiency and unfairness. The source gateway also has the responsibility of scheduling user packets coming on the n user connections over the tunnel, i.e. the m gateway-to-gateway connections. In this case study, we do not focus on scheduling but the control theoretic modeling and analysis of the tunnel's bandwidth adaptation. We study the effect of different types of controllers employed at the source gateway. Such controller determines the degree of elasticity of the gateway-to-gateway rate-adaptive tunnel, thus it determines the transient and steady-state behavior of the soft bandwidth-guaranteed service.

Naïve Control: This naïve controller measures the bandwidth b' grabbed by the current m' gateway-rate-adaptive connections. Then, it directly computes the quiescent number $\hat{m}$ of gateway-rate-adaptive connections that should be open as:

$$\hat{m} \quad = \quad \frac{B^*}{b'}m'$$

Clearly, this controller naïvely relies on the previously measured bandwidth b' and adapts without regard to delays in measurements and possible changes in network conditions, e.g. changes in the amount of cross traffic. We thus investigate general well-known controllers which judiciously zoom-in toward the target

bandwidth value. To that end, let us develop a flow-level model of the system dynamics. The change in the bandwidth grabbed $b(t)$ by the $m(t)$ gateway-rate-adaptive flows (constituting the elastic gateway-to-gateway tunnel) can be described as:

$$\dot{b}(t) = \alpha[(C - B^*)m(t) - B^*x(t)]$$

This dynamic equation captures what we want to model: $b(t)$ increases with $m(t)$, and decreases as the number of cross-connections $x(t)$ increases. α is a constant that represents the degree of multiplexing of flows and we choose it here to be the steady-state connection's fair share ratio of the bottleneck capacity. At steady-state, $\dot{b}(t)$ equals zero, which yields (as expected):

$$B^* = \frac{C\hat{m}}{(\hat{x} + \hat{m})}$$

where $\hat{m}$ and $\hat{x}$ represent the steady-state values for the number of gateway-rate-adaptive flows and cross-traffic flows, respectively. Based of the current bandwidth allocation $b(t)$ and the target bandwidth B^*, an error signal $e(t)$ can be obtained as:

$$e(t) = B^* - b(t)$$

P and PI Control: A controller would adjust $m(t)$ based on the value of $e(t)$. For a simple Proportional controller (P-type), such adjustment is described by:

$$m(t) = K_p e(t) \tag{21}$$

Recall that P-type controllers are known to result in a non-zero steady-state error. To exactly achieve the target B^* (i.e. with zero steady-state error), a Proportional-Integral (PI-type) controller can be used:

$$m(t) = K_p e(t) + K_i \int e(t)\, dt \tag{22}$$

Figure 29 shows the block diagram of this elastic-tunnel model. In the Laplace domain, denoting the controller transfer function by $G_c(s)$, the output $B(s)$ is given by:

$$B(s) = \frac{G_c(s)G_1(s)}{1 + G_c(s)G_1(s)} B^*(s) + \frac{G_2(s)}{1 + G_c(s)G_1(s)} X(s) \tag{23}$$

where $G_1(s)$ is given by:

$$G_1(s) = \frac{\beta}{s}$$

where $\beta = \alpha(C - B^*)$. $G_2(s)$ is given by:

$$G_2(s) = \frac{\gamma}{s}$$

where $\gamma = -\alpha B^*$. For the P-controller, from Equation (21), $G_c(s)$ is simply K_p. For the PI-controller, from Equation (22), $C(s)$ equals $K_p + \frac{K_i}{s}$. Thus, the transfer function $\frac{B(s)}{B^*}$ in the presence of a P-controller is given by:

$$\frac{B(s)}{B^*} = \frac{K_p \beta}{s + K_p \beta}$$

204

The system with P-controller is always stable since the root of the characteristic equation (i.e. the denominator of the transfer function) is negative, given by $-K_p\beta$ for $\beta > 0$ and $B^* < C$. In the presence of a PI-controller, the transfer function $\frac{B(s)}{B^*}$ is given by:

$$\frac{B(s)}{B^*} = \frac{K_p\beta s + K_i\beta}{s^2 + K_p\beta s + K_i\beta}$$

One can choose the PI-controller parameters K_p and K_i to achieve a certain convergence behavior to the target bandwidth B^*. We next define transient performance measures to assess such convergence behavior.
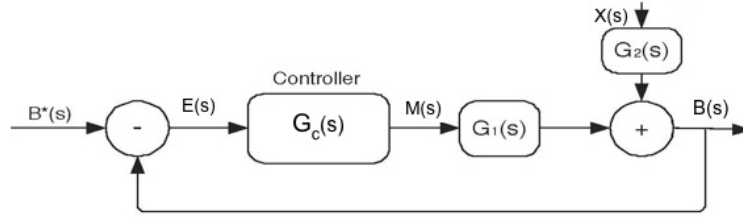
Figure 29: Block diagram of the elastic-tunnel model

8.1 Transient Performance

Transient behavior represents the system's response which decays with time. In the design of reliable systems, it is extremely important that transient response meets certain requirements, e.g. reasonable settling time and overshoot (cf. Section 4.5). Often times, the transient response is obtained by subjecting the system to an impulse or a step input and observing the output(s). One has to guarantee that the response of the system to specific inputs does not render the system unstable or pushes it away from its intended target.

Figure 30 shows the step response of the transfer function given in Equation (23). The left column shows the response to a unit step change in the target bandwidth, while the right column shows the response to a unit step change in the cross-traffic. Figure 30(a), for the P-controller, shows that while the response could be acceptable due to a step change in the reference bandwidth (i.e., the response $b(t)$ achieves the unit target), it fails to remove the steady-state error (i.e., non-zero $b(t)$) obtained from the step change in the cross-traffic. Figures 30(b) and (c) show the response due to the PI-controller. One can see that through a careful choice of K_p and K_i, the transient response can be adjusted. Notice that with a PI-controller, the elastic-tunneling system can reach the target bandwidth with zero steady-state error in response to a step change in cross-traffic.

8.2 Effect of Feedback Delay

So far in our analysis, we have ignored the feedback delay which is inherent in the design of any control system that tries to adjust its signal through a delayed feedback loop.

Figure 31 augments the block diagram of Figure 29 with feedback delay denoted by $H(s)$. This feedback delay arises either due to delayed mesurements of bandwidth and/or delayed response of the system as a result of applying new control. For example, when a new gateway-rate-adaptive connection is opened, it
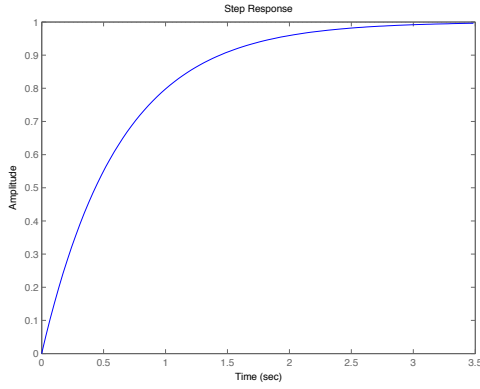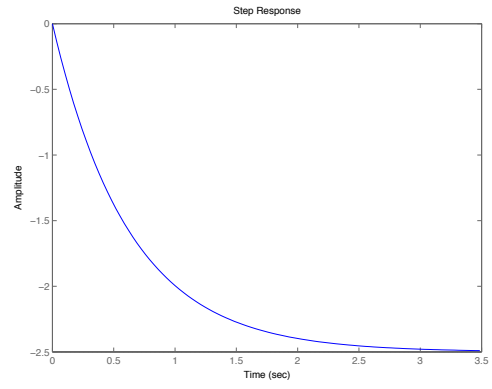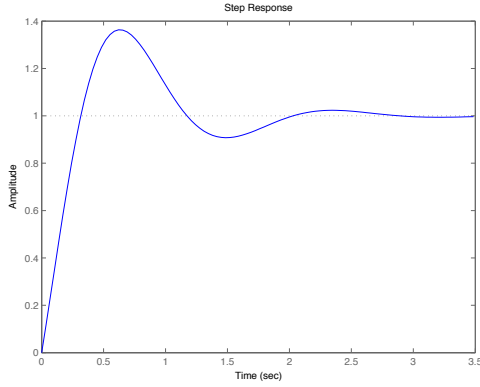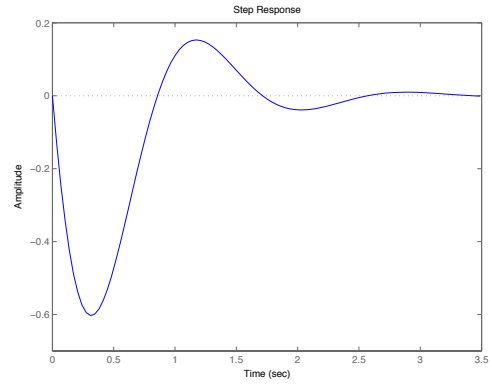
Target Bandwidth Step Response   Cross-traffic Step Response

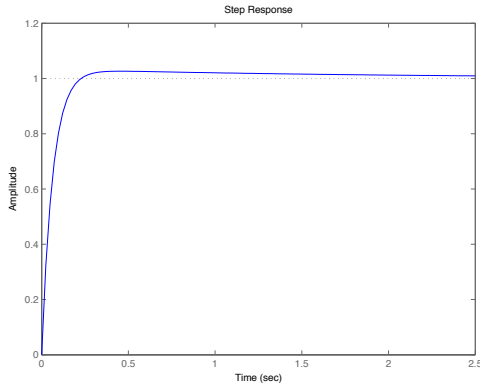(a) Proportional controller with $K_p = 0.1$

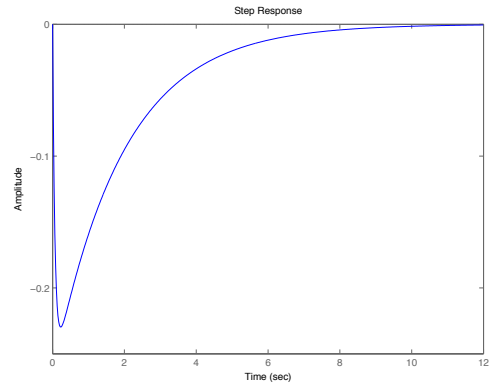Target Bandwidth Step Response   Cross-traffic Step Response

(b) Proportional Integral controller with $K_p = 0.2$ and $K_i = 1$

Target Bandwidth Step Response   Cross-traffic Step Response

(c) Proportional Integral controller with $K_p = 1$ and $K_i = 0.5$

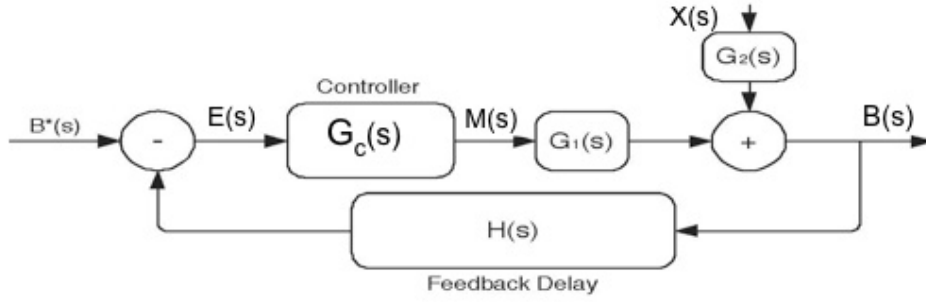Figure 30: Transient analysis of the elastic-tunnel model

Figure 31: Elastic-tunnel model (with feedback delay)

doesn't get its steady-state throughput instantaneously, rather after some delay (say τ). Thus, $H(s)$ is given by:

$$H(s) \quad = \quad e^{-\tau s}$$

where τ represents the feedback time delay. For small τ, the above equation can be approximated by:

$$H(s) \quad \approx \quad 1 - \tau s$$

If we are using a PI-controller, the characteristic equation in the presence of feedback delay becomes:

$$s^2(1 - \beta\tau K_p) + s(K_p\beta - \beta\tau K_i) + \beta K_i = 0$$

Figure 32 shows the response of the PI-controller to a unit step change in the target bandwidth. As the feedback delay τ increases, the system may not converge to the target bandwidth.

Target Bandwidth Step Response   Response with Feedback Delay

Proportional Integral controller with $K_p = 0.01$ and $K_i = 0.02$
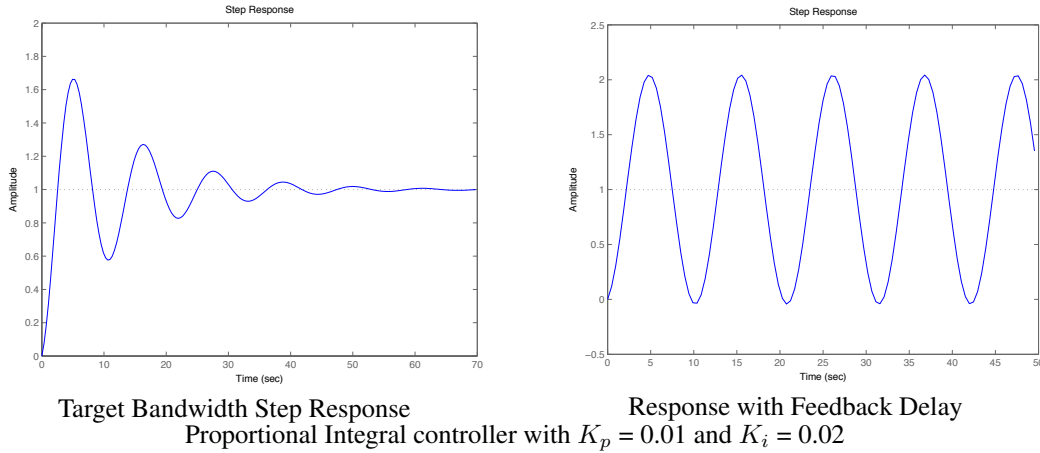
Figure 32: Transient analysis of the elastic-tunnel model (with feedback delay)

9 Exercises

1. Let r be a max-min fair rate vector corresponding to a given network and set of flows. This max-min fair allocation maximizes the allocation of each flow i subject to the constraint that an incremental increase in i's allocation does not cause a decrease in some other flow's allocation that is already as small as i's or smaller.

   (a) Suppose that some of the flows are eliminated and let $\bar{r}$ be a corresponding max-min fair rate vector. Show by example that we may have $\bar{r}_p < r_p$ for some of the remaining flows p.

   (b) Suppose that some of the link capacities are increased and let $\bar{r}$ be a corresponding max-min fair rate vector. Show by example that we may have $\bar{r}_p < r_p$ for some flows p.

2. Consider two network links in tandem (one after the other) of capacities 6 Mbps each. For two different sets of elastic flows (i.e. the utility function of each flow/user is a log function of its allocated rate), you are asked to write down the corresponding network optimization problem, where the network tries to maximize the sum of flow utilities subject to link capacity constraints. For each set of flows, described in parts (a) and (b) below, rewrite the constrained optimization problem as an unconstrained optimization problem: write down the Lagrangian function and corresponding equations to solve for the optimal rate allocation. What are the optimal rates allocated to different flows for each one of these two settings?

   (a) Consider two flows: one flow using the first link only and another flow using both links.

   (b) Consider the same two flows from part (a), as well as a third flow using the second link only.

3. Consider two network links in tandem (one after the other) of capacities 6 Mbps each. Assume three flows: one flow using the first link only, another flow using the second link only, and a third flow using both links. Assume the utility function of each flow/user is a log function of its allocated rate, and that the two-link flow is given a weight of 2, while the two one-link flows are given a weight of 1.

   You are asked to write down the corresponding network optimization problem, where the network tries to maximize the sum of the weighted flow utilities subject to link capacity constraints. Rewrite the constrained optimization problem as an unconstrained optimization problem: write down the Lagrangian function and corresponding equations to solve for the optimal rate allocation. What are the optimal rates allocated to each flow?

4. Consider a source adapting its sending rate $x(t)$ so the buffer size of its path's bottleneck $b(t)$ stays at a certain target value T. Denote the error signal by $e(t) = T - b(t)$. The sending rate is adapted according to one of the following three controllers:

   (a) $x(t) = K_p e(t)$

   (b) $x(t) = K_p e(t) + K_i \int_0^t e(t) dt$

   (c) $x(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{d}{dt} e(t)$

   where K_p, K_i, K_d are constant parameters of the rate controllers.

   (a) Assume $c(t)$ is the capacity available to the source at time t, write down the differential equation for $b(t)$.

(b) By transforming the system to the Laplace domain, determine the conditions under which the system is stable for each type of controller, i.e. does $b(t)$ converge to a given T for certain values of K_p, K_i, K_d. Do this by examining the roots (poles) of the characteristic equation of the system's transfer function. Draw the block diagram of the system that shows the relationships between the system variables in the (Laplace) s-domain for each type of controller.

(c) Again by examining the roots (poles) of the characteristic equation of the system's transfer function and using the Final Value Theorem, compare the transient and steady-state performance under each type of controller.

(d) Support your answers above by numerically solving the system's equations over time for each type of controller. Assume a small time step Δ, say $\Delta = 1$, and solve the discretized version of the equations at these time steps – you can then approximate the differentiation $\frac{d}{dt}b(t)$ by $b(t) - b(t-1)$.

5. Consider an adaptation of a transmission window w whose goal is to reach a target window T as follows:

$$w_{(k+1)} = w_k + \alpha(T - w_k)$$

where $0 < \alpha < 1$.

(a) Derive the necessary condition (if any) for convergence to the target window size.

(b) Use the Lyapunov method to show whether the system converges regardless of the initial window size.

6. Given a system with the following adaptation rules: $x(t)$, representing a sending rate, is adapted using an AIMD policy, whereas $p(t)$, representing the price, is adapted in proportion to how far $x(t)$ is from a target capacity c:

$$\frac{dx(t)}{dt} = 1 - x(t)p(x(t))$$

$$p(x(t)) = \alpha(x(t) - c)$$

(a) Why is that system non-linear?

(b) Linearize the system around a certain operating point x_0.

(c) By transforming the linearized system to the Laplace domain, obtain the condition on x_0 under which the linearized system is stable.

7. This question is based on the paper *The Revised ARPANET Routing Metric, by Zinky and Khanna* [14]. A unified way to model adaptive resource management—whether it is TCP adaptive to RED or routing adaptive to changing link costs or other examples—is through two functions: a feedback (pricing) function such as that of RED or link utilization metric, and an adaptation function such as that of TCP or utilization-based routing. Consider a resource that generates prices p and users that adapt their load λ based on the currently reported p. Assume the following pricing and load adaptation functions:

$$\lambda = 1 - p$$

$$p = \begin{cases} 0.1 & \text{if } 0 \le \lambda < 0.4 \\ 0.2 & \text{if } 0.4 \le \lambda \le 0.8 \\ 1.0 & \text{if } 0.8 < \lambda \le 1.0 \end{cases}$$

Show the two functions on the (λ, p) plane. Trace one trajectory showing convergence to a fixed point, and another trajectory showing oscillations. *Hint:* consider initial values of $\lambda = 0.2$ and 0.6.

10 Solutions to Exercises

1. (a) Refer to Figure 8. Using max-min fairness the original flow assignment is: (F1=50, F2=50, F3=50, F4=100). Assume F2 is removed. The new assignment is: (F1=75, F2=0, F3=75, F4=75). F1 and F3 will share the extra capacity, while F4 will decrease its rate. This is only possible because F4's original assigned rate was not less than or equal to F3's rate, before F2 was removed, thus allowing F3 to increase its rate.

   (b) Refer to Figure 8. Using max-min fairness the original flow assignment is: (F1=50, F2=50, F3=50, F4=100). If the capacity of link 1 is increased to 225 then the new allocated rates will be: (F1=75, F2=75, F3=75, F4=75). The extra capacity on link 1 will be shared by F1, F2 and F3. Again, this is only possible because F4's original assigned rate is not less than or equal to F3's rate, before the capacity of link 1 was increased, thus allowing F3 to increase its rate.

2. Note that earlier in these notes, we did not explicitly cover the technique of "slack", which we use here to denote residual link capacity when solving the dual Lagrangian problem. Intuitively, if the link is fully utilized, i.e. it has zero slack, then its price (Lagrangian multiplier) is non-zero. On the other hand, a non-bottleneck link will have a non-zero slack and so a price of zero.

   (a) Let z_1^2 and z_2^2 denote the (non-negative) slack on links 1 and 2, respectively. Let x_1 and x_2 denote the rates assigned to flows 1 and 2, respectively. Let $f(x_1, x_2)$ denote the function to be maximized.

$$f(x_1, x_2) = \log x_1 + \log x_2$$

Constraints on links 1 and 2, respectively, are as follows:

$$x_1 + x_2 + z_1^2 - c_1 = 0$$

$$x_2 + z_2^2 - c_2 = 0$$

Replacing c_1 and c_2 with their values we get:

$$x_1 + x_2 + z_1^2 - 6 = 0$$

$$x_2 + z_2^2 - 6 = 0$$

210

The Lagrangian function to be differentiated is as follows:

$$F(x_1, x_2) = \log x_1 + \log x_2 - \lambda(x_1 + x_2 + z_1^2 - 6) - \mu(x_2 + z_2^2 - 6)$$

Taking the partial derivative with respect to each variable we get the following equations:

$$\frac{\partial F}{\partial x_1} = \frac{1}{x_1} - \lambda = 0$$

$$\frac{\partial F}{\partial x_2} = \frac{1}{x_2} - \lambda - \mu = 0$$

$$\frac{\partial F}{\partial \lambda} = -(x_1 + x_2 + z_1^2 - 6) = 0$$

$$\frac{\partial F}{\partial \mu} = -(x_2 + z_2^2 - 6) = 0$$

$$\frac{\partial F}{\partial z_1} = -2z_1\lambda = 0$$

$$\frac{\partial F}{\partial z_2} = -2z_2\mu = 0$$

Since F1 can utilize any left over capacity (slack) on link 1, $z_1 = 0$, which implies that $\lambda \neq 0$. Conversely, since F2 is limited by link 1, $z_2 \neq 0$, which implies that $\mu = 0$.

Replacing $z_1 = 0$ and $\mu = 0$ into the derived equations, we get:

$$\frac{1}{x_1} - \lambda = 0$$

$$\frac{1}{x_2} - \lambda - 0 = 0$$

$$x_1 + x_2 + 0 - 6 = 0$$

Solving for x_1 we get $x_1 = \frac{1}{\lambda}$
Solving for x_2 we get $x_2 = \frac{1}{\lambda}$. Hence, $x_1 = x_2$.
Replacing x_2 by x_1 in the capacity equation we get:

$$x_1 + x_1 = 6$$

Hence, $x_1 = x_2 = 3$.

(b) Let z_1^2 and z_2^2 denote the slack on links 1 and 2, respectively. Let x_1, x_2 and x_3 denote the rates assigned to flows 1, 2 and 3, respectively. Let $f(x_1, x_2, x_3)$ denote the function to be maximized.

$$f(x_1, x_2, x_3) = \log x_1 + \log x_2 + \log x_3$$

Constraints on links 1 and 2, respectively, are as follows:

$$x_1 + x_2 + z_1^2 - c_1 = 0$$

$$x_3 + x_2 + z_2^2 - c_2 = 0$$

Replacing c_1 and c_2 with their values we get:

$$x_1 + x_2 + z_1^2 - 6 = 0$$

$$x_3 + x_2 + z_2^2 - 6 = 0$$

The Lagrangian function to be differentiated is as follows:

$$F(x_1, x_2, x_3) = \log x_1 + \log x_2 + \log x_3 - \lambda(x_1 + x_2 + z_1^2 - 6) - \mu(x_3 + x_2 + z_2^2 - 6)$$

Taking the partial derivative with respect to each variable we get the following equations:

$$\frac{\partial F}{\partial x_1} = \frac{1}{x_1} - \lambda = 0$$

$$\frac{\partial F}{\partial x_2} = \frac{1}{x_2} - \lambda - \mu = 0$$

$$\frac{\partial F}{\partial x_3} = \frac{1}{x_3} - \mu = 0$$

$$\frac{\partial F}{\partial \lambda} = -(x_1 + x_2 + z_1^2 - 6) = 0$$

$$\frac{\partial F}{\partial \mu} = -(x_3 + x_2 + z_2^2 - 6) = 0$$

$$\frac{\partial F}{\partial z_1} = -2z_1\lambda = 0$$

$$\frac{\partial F}{\partial z_2} = -2z_2\mu = 0$$

From the equations above we can deduce the following:

$$x_1 = \frac{1}{\lambda}$$

$$x_2 = \frac{1}{\lambda + \mu}$$

$$x_3 = \frac{1}{\mu}$$

Since F1 and F3 can utilize any left over capacity (slack) on links 1 and 2, $z_1 = z_2 = 0$. This implies that $\lambda \neq 0$ and $\mu \neq 0$.

Replacing $z_1 = 0$ and $z_2 = 0$ into the derived equations we get:

$$x_1 + x_2 = 6$$

$$x_2 + x_3 = 6$$

Replacing x_1 by $\frac{1}{\lambda}$, x_2 by $\frac{1}{\lambda + \mu}$ and x_3 by $\frac{1}{\mu}$ into the equations above, we get:

$$\frac{1}{\lambda} + \frac{1}{\lambda + \mu} = 6$$

$$\frac{1}{\lambda + \mu} + \frac{1}{\mu} = 6$$

Multiplying the first equation by (-1) and adding it to second equation, results in the following:

$$\frac{1}{\mu} - \frac{1}{\lambda} = 0$$

Hence, $\lambda = \mu$

Thus, $x_1 = x_3 = \frac{1}{\lambda}$ and $x_2 = \frac{1}{2\lambda} = \frac{x_1}{2} = \frac{x_3}{2}$
Finally we get:

$$x_1 + \frac{x_1}{2} = 6$$

Hence, $x_1 = x_3 = 4$ and $x_2 = 2$.

213

3. The objective function we want to maximize is:

$F(x) = \log x_1 + \log x_2 + 2 \log x_3$

subject to the capacity constraints:

$x_1 + x_3 \leq 6$
$x_2 + x_3 \leq 6$
$x_1, x_2, x_3 \geq 0$

The Lagrangian (unconstrained) function that we want to maximize is:

$L(x) = \log x_1 + \log x_2 + 2 \log x_3 - \lambda_1(x_1 + x_3 - 6) - \lambda_2(x_2 + x_3 - 6)$

Note that we do not explicitly include the slacks for the link capacities here, since both links should be fully utilized as the one-link flows are not limited by any other link, so the slack values are zero.

Taking the partial derivatives of $L(.)$ we obtain:

$\frac{\partial L}{\partial x_1} = \frac{1}{x_1} - \lambda_1 = 0 \Rightarrow x_1 = \frac{1}{\lambda_1}$

$\frac{\partial L}{\partial x_2} = \frac{1}{x_2} - \lambda_2 = 0 \Rightarrow x_2 = \frac{1}{\lambda_2}$

$\frac{\partial L}{\partial x_3} = \frac{2}{x_3} - (\lambda_1 + \lambda_2) = 0 \Rightarrow x_3 = \frac{2}{\lambda_1 + \lambda_2}$

$\frac{\partial L}{\partial \lambda_1} = 0 \Rightarrow x_1 + x_3 - 6 = 0 \Rightarrow x_1 + x_3 = 6$

$\frac{\partial L}{\partial \lambda_2} = 0 \Rightarrow x_2 + x_3 - 6 = 0 \Rightarrow x_2 + x_3 = 6$

The last two equations yield $x_1 = x_2 \Rightarrow \lambda_1 = \lambda_2 = \lambda$

From the capacity equation, we have:

$x_1 + x_3 = \frac{1}{\lambda} + \frac{2}{2\lambda} = 6 \Rightarrow \lambda = \frac{1}{3}$, thus

$x_1 = x_2 = 3$, and $x_3 = 6 - 3 = 3$.

4. We use the following notation:

In the time domain we have:
$b(t)$, buffer size at time t
$x(t)$, sending rate at time t
$c(t)$, service rate at time t
T, target buffer size
$e(t)$, error (difference between current buffer size and the target) at time t

In the frequency domain we have:
$B(s)$, current buffer size
$X(s)$, sending rate
$C(s)$, service rate
$T(s)$, target buffer size
$E(s)$, error signal
$D(s)$, controller (based on error signal computes sending rate)

We will assume that the target buffer size and the buffer's service rate are constant values.

(a) $e(t) = T - b(t)$
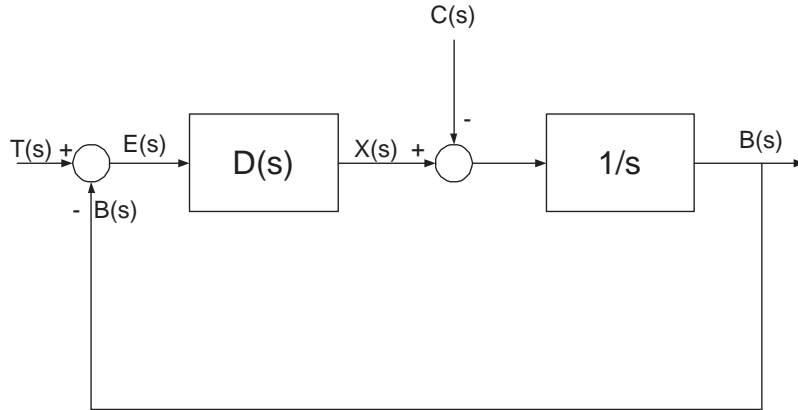
$\frac{d}{dt}b(t) = x(t) - c(t)$

Figure 33: System block diagram

(b) The system's block diagram is depicted in Figure 33. Using the block diagram, we can formulate the following equation:

$$(\, [\, (T(s) - B(s)) \, D(s) \,] - C(s) \,) \, \left(\tfrac{1}{s}\right) = B(s)$$

$$(\, [T(s)D(s) - B(s)D(s)] - C(s) \,) \, \left(\tfrac{1}{s}\right) = B(s)$$

$$(T(s)D(s) - B(s)D(s) - C(s)) \, \left(\tfrac{1}{s}\right) = B(s)$$

$$[\tfrac{T(s)D(s)}{s} - \tfrac{B(s)D(s))}{s} - \tfrac{C(s)}{s}] = B(s)$$

$$B(s) + \tfrac{B(s)D(s)}{s} = \tfrac{T(s)D(s)}{s} - \tfrac{C(s)}{s}$$

$$B(s)[1 + \tfrac{D(s)}{s}] = \tfrac{T(s)D(s)}{s} - \tfrac{C(s)}{s}$$

$$B(s)[\tfrac{s+D(s)}{s}] = \tfrac{T(s)D(s)}{s} - \tfrac{C(s)}{s}$$

$$B(s) = \tfrac{T(s)D(s)}{s+D(s)} - \tfrac{C(s)}{s+D(s)}$$

P-Controller:

$x(t) = K_p e(t)$

By taking the Laplace transform we get:

$X(s) = K_p E(s) \Rightarrow D(s) = \frac{X(s)}{E(s)} = K_p$

Replacing $D(s)$ into the equation for $B(s)$ we get:

$B(s) = \frac{T(s)K_p}{s+K_p} - \frac{C(s)}{s+K_p}$

Roots of the system's characteristic equation are:

$s + K_p = 0 \Rightarrow s = -K_p < 0$
Thus, system is stable for all values of $K_p > 0$

PI-Controller:

$x(t) = K_p e(t) + K_i \int_0^t e(t)dt$

By taking the Laplace transform we get:

$X(s) = K_p E(s) + \frac{K_i}{s} E(s) \Rightarrow D(s) = \frac{X(s)}{E(s)} = K_p + \frac{K_i}{s}$

Replacing $D(s)$ into the equation for $B(s)$ we get:

$B(s) = \frac{(K_p + \frac{K_i}{s})T(s)}{s+(K_p+\frac{K_i}{s})} - \frac{C(s)}{s+(K_p+\frac{K_i}{s})}$

Roots of the system's characteristic equation are:

$s + K_p + \frac{K_i}{s} = 0$
$s^2 + K_p s + K_i = 0$
$a = 1, b = K_p, c = K_i$
$\Delta = b^2 - 4ac = K_p^2 - 4(1)(K_i) = K_p^2 - 4K_i$

$s = \frac{-b \pm \sqrt{\Delta}}{2a}$

$s = \frac{-K_p \pm \sqrt{K_p^2 - 4K_i}}{2}$

$s = -\frac{1}{2}K_p \pm \frac{1}{2}\sqrt{K_p^2 - 4K_i}$

System will be stable with two real roots (i.e., overdamped) if the following conditions are true:
$K_p > 0$
$K_p^2 - 4K_i > 0 \Rightarrow K_p^2 > 4K_i$
$\sqrt{K_p^2 - 4K_i} < K_p \Rightarrow K_p^2 - 4K_i < K_p^2 \Rightarrow K_i > 0$

System will be stable with two complex roots (i.e., underdamped) if the following conditions are true:
$K_p > 0$
$K_p^2 - 4K_i < 0 \Rightarrow K_p^2 < 4K_i$

PID-Controller:

$$x(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t)$$

By taking the Laplace transform we get:

$$X(s) = K_p E(s) + \frac{K_i}{s} E(s) + K_d s E(s) \Rightarrow D(s) = \frac{X(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s$$

Replacing $D(s)$ into the equation for $B(s)$ we get:

$$B(s) = \frac{(K_p + \frac{K_i}{s} + K_d s)T(s)}{s + (K_p + \frac{K_i}{s} + K_d s)} - \frac{C(s)}{s + (K_p + \frac{K_i}{s} + K_d s)}$$

Roots of the system's characteristic equation are:

$$s + K_p + \frac{K_i}{s} + K_d s = 0$$
$$s^2 + K_p s + K_i + K_d s^2 = 0$$
$$(1 + K_d)s^2 + K_p s + K_i = 0$$
$$a = 1 + K_d, b = K_p, c = K_i$$
$$\Delta = b^2 - 4ac = K_p^2 - 4(1 + K_d)(K_i) = K_p^2 - 4K_i(1 + K_d)$$

$$s = \frac{-b \pm \sqrt{\Delta}}{2a}$$

$$s = \frac{-K_p \pm \sqrt{K_p^2 - 4K_i(1 + K_d)}}{2(1 + K_d)}$$

$$s = -\frac{K_p}{2(1 + K_d)} \pm \frac{1}{2(1 + K_d)} \sqrt{K_p^2 - 4K_i(1 + K_d)}$$

System will be stable with two real roots if the following conditions are true:
$$\frac{K_p}{1 + K_d} > 0$$
$$K_p^2 - 4K_i(1 + K_d) > 0 \Rightarrow K_p^2 > 4K_i(1 + K_d)$$
$$\sqrt{K_p^2 - 4K_i(1 + K_d)} < K_p \Rightarrow K_p^2 - 4K_i(1 + K_d) < K_p^2 \Rightarrow 4K_i(1 + K_d) > 0$$

System will be stable with two complex roots if the following conditions are true:
$$\frac{K_p}{1 + K_d} > 0$$
$$K_p^2 - 4K_i(1 + K_d) < 0 \Rightarrow K_p^2 < 4K_i(1 + K_d)$$

(c) Final Value Theorem
$$\lim_{t \to \infty} f(t) = \lim_{s \to 0} sF(s)$$

Thus, we have:
$$\lim_{t \to \infty} b(t) = \lim_{s \to 0} sB(s)$$

Note that,
$$B(s) = \frac{T(s)D(s)}{s + D(s)} - \frac{C(s)}{s + D(s)}$$

Let the service rate be constant and equal to C. We therefore have, $C(s) = \frac{C}{s}$. Similarly, let the target buffer size be a constant and equal to T. We therefore have, $T(s) = \frac{T}{s}$. Replacing these two values into the equation for $B(s)$, we get:

$$B(s) = \frac{\frac{T}{s}D(s)}{s+D(s)} - \frac{\frac{C}{s}}{s+D(s)}$$

Multiplying by s to get $sB(s)$, we get:

$$sB(s) = \frac{TD(s)}{s+D(s)} - \frac{C}{s+D(s)}$$

P-Controller:
$$D(s) = K_p$$

$$sB(s) = \frac{TK_p}{s+K_p} - \frac{C}{s+K_p}$$

$$\lim_{s\to 0} sB(s) = \lim_{s\to 0}\left[\frac{TK_p}{s+K_p} - \frac{C}{s+K_p}\right] = T - \frac{C}{K_p}$$

PI-Controller:
$$D(s) = K_p + \frac{K_i}{s}$$

$$sB(s) = \frac{T(K_p+\frac{K_i}{s})}{s+(K_p+\frac{K_i}{s})} - \frac{C}{s+(K_p+\frac{K_i}{s})} = \left[\frac{T(K_ps+K_i)}{s^2+K_ps+K_i} - \frac{Cs}{s^2+K_ps+K_i}\right]$$

$$\lim_{s\to 0} sB(s) = \lim_{s\to 0}\left[\frac{T(K_ps+K_i)}{s^2+K_ps+K_i} - \frac{Cs}{s^2+K_ps+K_i}\right] = T$$

PID-Controller:
$$D(s) = K_p + \frac{K_i}{s} + K_ds$$

$$sB(s) = \frac{T(K_p+\frac{K_i}{s}+K_ds)}{s+(K_p+\frac{K_i}{s}+K_ds)} - \frac{C}{s+(K_p+\frac{K_i}{s}+K_ds)} = \frac{T(K_ps+K_i+K_ds^2)}{s^2+K_ps+K_i+K_ds^2} - \frac{Cs}{s^2+K_ps+K_i+K_ds^2}$$

$$sB(s) = \frac{T(K_ps+K_i+K_ds^2)}{(1+K_d)s^2+K_ps+K_i} - \frac{Cs}{(1+K_d)s^2+K_ps+K_i}$$

$$\lim_{s\to 0} sB(s) = \lim_{s\to 0}\left[\frac{T(K_ps+K_i+K_ds^2)}{(1+K_d)s^2+K_ps+K_i} - \frac{Cs}{(1+K_d)s^2+K_ps+K_i}\right] = T$$

(d) We leave it to the reader to show the plots for $b(t)$.

5. (a) At steady state, $w_k \to w^*$. This implies that $w^* = w^* + \alpha(T - w^*)$. Thus, $w^* = T$. This means that there is no necessary condition, since the system will always converge to the target window size.

(b) We have $w_{k+1} = w_k + \alpha(T - w_k)$. Re-writing this equation we get, $w_{k+1} = w_k(1 - \alpha) + \alpha T$. Assume $w_0 < T$, let the Lyapunov function be $L(w) = T - w > 0$.

$$L(w_{k+1}) = T - w_{k+1}$$
$$L(w_{k+1}) = T - [w_k + \alpha(T - w_k)]$$
$$L(w_{k+1}) = T - w_k - \alpha T + \alpha w_k$$
$$L(w_{k+1}) = (1 - \alpha)T - (1 - \alpha)w_k$$
$$L(w_{k+1}) = (1 - \alpha)(T - w_k)$$
$$L(w_{k+1}) = (1 - \alpha)L(w_k) < L(w_k)$$

This is true since $(1 - \alpha) < 0$. In other words, $L(w)$ is indeed a decreasing function.

If instead we assume that $w_0 > T$, then let us define the Lyapunov function as $L(w) = w - T > 0$. Similarly we have,

$$L(w_{k+1}) = w_{k+1} - T$$
$$L(w_{k+1}) = [w_k(1 - \alpha) + \alpha T] - T$$
$$L(w_{k+1}) = w_k(1 - \alpha) - (T - \alpha T)$$
$$L(w_{k+1}) = w_k(1 - \alpha) - T(1 - \alpha)$$
$$L(w_{k+1}) = (1 - \alpha)(w_k - T)$$
$$L(w_{k+1}) = (1 - \alpha)L(w_k) < L(w_k)$$

This is true since $(1 - \alpha) < 0$. In other words, $L(w)$ is indeed a decreasing function.

6. (a) $\frac{d}{dt}x(t) = 1 - x(t)p(x(t))$
$p(x(t)) = \alpha(x(t) - c)$
$\frac{d}{dt}x(t) = 1 - x(t)[\alpha(x(t) - c)]$
$\frac{d}{dt}x(t) = 1 - x(t)[\alpha x(t) - \alpha c)]$
$\frac{d}{dt}x(t) = 1 - \alpha x^2(t) + \alpha c x(t)$

The equation above is non-linear because it has an $x^2(t)$ term.

(b) Let $f(x) = 1 - \alpha x^2(t) + \alpha c x(t)$
$f'(x_0) = -2\alpha x_0 + \alpha c$
(Note that one may choose to linearize the x^2 term only. Here we choose to linearize the whole $f(x)$.)
$\Delta f = f'(x_0)\Delta x$
$\frac{d}{dt}\Delta x = f'(x_0)\Delta x$
Let $y(t) = \Delta x$
$\frac{d}{dt}y(t) = f'(x_0)y(t)$
$\frac{d}{dt}y(t) = [-2\alpha x_0 + \alpha c]y(t)$
Let $\beta = [-2\alpha x_0 + \alpha c]$
$\frac{d}{dt}y(t) = \beta y(t)$

(c) Transforming this equation to the s-domain, we get:
$sY(s) - y(0) = \beta Y(s)$
$sY(s) - \beta Y(s) = y(0)$
$Y(s)[s - \beta] = y(0)$
$Y(s) = \frac{y(0)}{s - \beta}$

219

The root of the characteristic equation $s - \beta$ is $s = \beta$. For the system to be stable, β must be less than zero. Assuming $\alpha > 0$, we need the following to be true:

$\beta < 0$
$-2\alpha x_0 + \alpha c < 0$
$-2\alpha x_0 < -\alpha c$
$2\alpha x_0 > \alpha c$
$x_0 > \frac{c}{2}$

7. We leave it to the reader to produce the plot. Notice that the fixed point is $(\lambda = 0.8, p = 0.2)$ where the pricing curve and the load curve intersect. Also the system diverges for starting $\lambda = 0.2$, whereas it converges for $\lambda = 0.6$. Specifically, in the latter case, $\lambda = 0.6$ yields $p = 0.2$, which in turn yields $\lambda = 1 - p = 0.8$ and the system stabilizes at the fixed point.

References

[1] ARCHIVE, I. http://archive.org/web/web.php. [Online; accessed June-11-2013].

[2] ATHURALIYA, S., LOW, S. H., LI, V. H., AND YIN, Q. REM: Active Queue Management. *Netwrk. Mag. of Global Internetwkg. 15*, 3 (May 2001), 48–53.

[3] CHIU, D.-M., AND JAIN, R. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Comput. Netw. ISDN Syst. 17*, 1 (June 1989), 1–14.

[4] ETFORECASTS. http://www.etforecasts.com/. [Online; accessed June-11-2013].

[5] FALL, K., AND FLOYD, S. Simulation-based Comparisons of Tahoe, Reno and SACK TCP. *SIGCOMM Comput. Commun. Rev. 26*, 3 (July 1996), 5–21.

[6] FLOYD, S., AND JACOBSON, V. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Trans. Netw. 1*, 4 (Aug. 1993), 397–413.

[7] FLOYD, S., AND JACOBSON, V. Link-sharing and Resource Management Models for Packet Networks. *IEEE/ACM Trans. Netw. 3*, 4 (Aug. 1995), 365–386.

[8] GUIRGUIS, M., BESTAVROS, A., MATTA, I., RIGA, N., DIAMANT, G., AND ZHANG, Y. Providing Soft Bandwidth Guarantees Using Elastic TCP-based Tunnels. In *Proceedings of ISCC '2004: The Ninth IEEE Symposium on Computers and Communications* (Alexandria, Egypt, June 2004).

[9] GUO, L., AND MATTA, I. The War between Mice and Elephants. In *Proceedings of ICNP'2001: The 9th IEEE International Conference on Network Protocols* (Riverside, CA, November 2001).

[10] HOLLOT, C., MISRA, V., TOWSLEY, D., AND GONG, W.-B. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (2001), vol. 3, pp. 1726–1734 vol.3.

[11] JACOBSON, V. Congestion Avoidance and Control. In *Proc. ACM SIGCOMM* (Stanford, CA, August 1988).

[12] KELLY, F. Mathematical Modelling of the Internet. In *Mathematics Unlimited - 2001 and Beyond*. Springer-Verlag, 2001, pp. 685–702.

[13] KESHAV, S. *Mathematical Foundations of Computer Networking*. Addison-Wesley, 2012.

[14] KHANNA, A., AND ZINKY, J. The Revised ARPANET Routing Metric. In *Symposium proceedings on Communications architectures & protocols* (New York, NY, USA, 1989), SIGCOMM '89, ACM, pp. 45–56.

[15] LAWRENCE S. BRAKMO AND LARRY L. PETERSON. TCP Vegas: End to End Congestion Avoidance on a Global Internet. *IEEE Journal on Selected Areas in Communications 13*, 8 (1995), 1465–1480.

[16] LU, C. Feedback Control Theory: A Computer System's Perspective. http://www.cs.virginia.edu/~cl7v/cs851-talks/control_tutorial.ppt, 2001. [Online; accessed June-11-2013].

[17] MATTA, I., AND GUO, L. Differentiated Predictive Fair Service for TCP Flows. In *Proceedings of ICNP'2000: The 8th IEEE International Conference on Network Protocols* (Osaka, Japan, October 2000).

[18] MO, J., AND WALRAND, J. Fair End-to-end Window-based Congestion Control. *IEEE/ACM Trans. Netw. 8*, 5 (Oct. 2000), 556–567.

[19] NETCRAFT. http://www.netcraft.com/. [Online; accessed June-11-2013].

[20] OGATA, K. *Modern Control Engineering*. Prentice Hall, 2010.

[21] PADHYE, J., FIROIU, V., TOWSLEY, D. F., AND KUROSE, J. F. Modeling TCP Reno Performance: a Simple Model and its Empirical Validation. *IEEE/ACM Trans. Netw. 8*, 2 (Apr. 2000), 133–145.

[22] PAREKH, A. K., AND GALLAGHER, R. G. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Multiple Node Case. *IEEE/ACM Trans. Netw. 2*, 2 (Apr. 1994), 137–150.

[23] SHENKER, S. Fundamental Design Issues for the Future Internet. *IEEE J.Sel. A. Commun. 13*, 7 (Sept. 2006), 1176–1188.

[24] SRIKANT, R. *The Mathematics of Internet Congestion Control*. Birkhauser, 2004.

[25] WEI, D. X., JIN, C., LOW, S. H., AND HEGDE, S. FAST TCP: Motivation, Architecture, Algorithms, Performance. *IEEE/ACM Trans. Netw. 14*, 6 (Dec. 2006), 1246–1259.

Smart Data Pricing (SDP):
Economic Solutions to Network Congestion

Soumya Sen, Carlee Joe-Wong, Sangtae Ha, Mung Chiang

1 Introduction

Advances in Internet technologies have resulted in an unprecedented growth in demand for data. In particular, demand in the mobile Internet sector is doubling every year [25]. Given the limited wireless spectrum availability, the rate of growth in the supply of wireless capacity (per dollar of investment) is unlikely to match the rate of growth in demand in the long run. Internet Service Providers (ISPs) are therefore turning to new pricing and penalty schemes in an effort to manage the demand on their network, while also matching their prices to cost. But changes in pricing and accounting mechanisms, if not done carefully, can have significant consequences for the entire network ecosystem. Multiple stakeholders in this ecosystem, including operators, consumers, regulators, content providers, hardware and software developers, and architects of network technologies, have all been tackling these issues of charging and allocating limited network resources. Even back in 1974, while writing about the future challenges of computer communication networks, Leonard Kleinrock [71] noted:

> *[H]ow does one introduce an equitable charging and accounting scheme in such a mixed network system? In fact, the general question of accounting, privacy, security and resource control and allocation are really unsolved questions which require a sophisticated set of tools.*

While much progress has been made on developing technical solutions, methods, and tools to address these issues, continued growth of the network ecosystem requires developing a better understanding of the underlying economic and policy perspectives. The broader area of *network economics*, which deals with the interplay between technological and economic factors of networks, is therefore receiving more attention from engineers and researchers today. Economic factors like pricing, costs, incentive mechanisms and externalities[1] affect the adoption outcomes (i.e., success or failure of network technologies) and stability [62,65,110], influence network design choices [108,109], and impact service innovation [138]. Conversely, technological limitations and regulatory constraints determine which kind of economic models are most suited to analyze a particular network scenario. This interplay between technology, economics, and regulatory issues is perhaps most easily observed in the case of broadband access pricing, for example, in evaluating the merits of "flat-rate" versus "usage-based" pricing or the neutrality of "volume-based" versus "app-based" accounting, etc. In this chapter we discuss the current trends in access pricing among service operators, factors that affect these decisions, analytical models and related considerations. In particular, we observe that *Smart Data*

---

[1]Network externality is the notion that the cost or value of being a part of a network for an individual user depends on the number of other users using that network. For example, the value of a network grows as more users adopt and positive externalities are realized from being able to communicate with other users on the network. Similarly, when many users start to content for limited resources of a bottleneck link of a network, negative externalities from congestion diminish a user's utility from accessing the network.

Pricing[2] (SDP) is likely to emerge as an effective way to cope with increased network congestion. These smarter ways to count and treat data traffic illustrate three shifts in the principles of network management:

1. ***Pricing for end-user Quality of experience (QoE) and not just byte-counting***: Simple policies like usage-based pricing (byte-counting) (a) force users to pay the same amount per unit of bandwidth consumed irrespective of the congestion levels on the network,[3] and (b) fail to account for the fact that different applications have different bandwidth requirements to attain a certain QoE for the user. SDP should try to match the cost of delivering application-specific desired QoE requirements of the user to the ISP's congestion cost at the time of delivery.

2. ***Application layer control to impact physical layer resource management***: Today's smart devices with their easy to use graphical user interfaces can potentially enable consumer-specified choice for access quality. Whether done manually or in an automated mode, users' specifications of their willingness to pay for their desired QoE of different applications can be taken in as inputs at the APP layer and used to control PHY layer resource allocation and media selection (e.g., WiFi offloading versus 3G). But enabling this requires consumer trials to understand how to design incentives and create interfaces that can be effective in modifying end-user behavior.

3. ***Incorporating edge devices as a part of network management system***: Instead of managing traffic only in the network core, SDP explores ways to make edge devices (e.g., smart mobile devices and customer-premise equipments like gateways) a part of the network resource allocation and management system. For example, instead of throttling traffic in the network core using the policy charging and rules function (PCRF), the edge devices (e.g., home gateways) themselves could locally regulate demand based on a user's budget, QoE requirements, and network load or available prices. Such measures to push control from the network core out to the end-users, while preserving the end-to-end principles of the Internet, have been gaining attention among networking research groups [9].

But before delving deeper into pricing ideas, let us pause to address some common misconceptions often encountered in public discourse. First, many believe that the Internet's development cost was borne by the United States Government, and hence that taxpayers have already paid for it. In reality, by 1994 the National Science Foundation supported less than 10% of the Internet and by 1996 huge commercial investments were being made worldwide [85].

Second, users often do not realize that the Internet is not free [21, 85] and think its cost structure is the same as that of information goods. In contrast to *information goods*, which tend to have zero marginal costs,[4] Internet operators incur considerable network management operation and billing costs. MacKie-Mason and Varian [81] have shown that while the marginal cost of some Internet traffic can be zero because of *statistical multiplexing*, congestion costs can be quite significant. In regard to delivery of bits, it is worthwhile to note that there are some important factors at play:

(a) There is a large and growing variance in the QoE requirements of the different types of applications that consumers are using today, and

---

[2]SDP is the broad set of ideas and principles that goes beyond the traditional flat-rate or byte-counting models and instead considers pricing as a network management solution. See http://www.smartdatapricing.org.

[3]In 1997, David Clark wrote [27] that "The fundamental problem with simple usage fees is that they impose usage costs on users regardless of whether the network is congested or not."

[4]Marginal cost is the change in the total cost that arises when the quantity produced changes by one unit, e.g., the cost of adding one more unit of bandwidth.

(b) The network operator's cost of delivery per bit for a given QoE level also has significant variance, ranging from essentially zero marginal cost in uncongested times to very high in congested times.

(c) There is also a variance in user's willingness to pay for different types of traffic and QoE levels.

So why not match the right pairs? Most SDP ideas aim to do exactly that, i.e., match the operator's cost of delivering bits to the consumer's QoE needs for different application types at the amount they are willing to spend.

Third, there is a popular misconception that network costs are high because billing costs account for 50% of telephony costs. Although true for running costs, it is only 4-6% when depreciation of sunk costs is added [9]. Another important cost for wireless operators today is the cost of acquiring new spectrum to support the growing bandwidth needs of the customers. However, spectrum is limited and expensive, and even auction-based spectrum reallocation schemes are projected to fall short of the demand for spectrum.

Fourth, the belief that better technologies like 4G and offloading mechanisms will solve the problems is already being questioned – "The reasons are two-fold: The amount of spectrum made available to U.S. wireless companies is limited, but the carriers have also been sluggish in buying up enough backhaul to support their capacity requirements. There is only so much data that can be crammed into wireless spectrum – and only so much spectrum available to wireless networks. Thanks to rising mobile data demands, a current wireless spectrum surplus of 225 MHz will become a deficit of 275 MHz by 2014, according to the FCC [44]."

Fifth, users fear that changes in pricing policy will increase their access fees. This need not always be the case, as one can design incentive mechanisms that reward good behavior (e.g., price discounts in off-peak hours to incentivize shifting of usage demand from peak times). In other words, smarter pricing mechanisms can *increase consumer choices* by empowering users to take better control of how they spend their monthly budget. For example, under time-dependent usage-based pricing [48, 113], users have better control over their monthly bills by choosing not only *how much* they want to consume, but also *when* they do so). Smart data pricing also has to be smart in its implementation and in its user interface design, with careful study of user psychology and human-computer interaction aspects, as we will illustrate in later sections and case studies.

Lastly, we also need to remember that pricing is related to the market competition and user population density. For the interested reader, an overview of access fees in different parts of the world is provided in Section A.

The following questions provide a useful way to think about SDP:

(I) Why do we need SDP? Isn't network pricing an untouchable legacy?

Section 2 provides an overview of the driving factors behind network congestion, and the challenges that it poses to various stakeholders of the network ecosystem are discussed in Section 3. We also discuss the rapid evolution in pricing among network operators and highlight in Section 4 how Smart Data Pricing ideas will be useful in finding solutions that can work in today's networks.

(II) Haven't other fields already used pricing innovations? What are the key SDP ideas relevant to communication networks?

We provide an overview of Internet pricing ideas in the existing literature in Section 5, including some pricing plans from the electricity and transportation industries that can be applied to broadband pricing. Section 6 provides an overview of a few examples and analytical models of known pricing mechanisms to illustrate key economic concepts relevant to the SDP literature. We also highlight many crucial differences between SDP in communication or data networks and pricing innovations in other industries.

(III) Isn't SDP too complex to implement in the real world?

Section 9 provides a case study of a field trial of "day-ahead time-dependent pricing" and discusses the model, system design, and user interface design considerations for realizing this plan. It serves to demonstrate both the feasibility of creating such SDP plans for real deployment while also pointing out the design issues that should be kept in mind. The discussion highlights the end-to-end nature of an SDP deployment, which requires developing pricing algorithms, understanding user psychology, designing an effective interface for communicating those prices to users, and implementing an effective system to communicate between the users and ISPs.

(IV) What are the outstanding problems in enabling SDP for the Internet?

SDP is an active area of research in the network economics community and a set of 20 questions and future directions are provided in Section 10 for researchers and graduate students to explore. Many of these research questions have been discussed at various industry-academia forums and workshops on SDP [99, 107].

2 Driving Factors of Network Congestion

With mobile devices becoming smarter, smaller, and ubiquitous, consumers are embracing the technology and driving up the demand for mobile data. According to Cisco's VNI [26], in 2012, global mobile data traffic grew more than 70 percent year over year, to 855 petabytes a month. The growth rate varied by regions, with 44% growth in Western Europe, and about 101% in the Middle East and Africa and a 95% growth rate in Asia Pacific. This section identifies some of the key factors that are expected to drive this growth in demand for mobile data (ref. Figure 1).
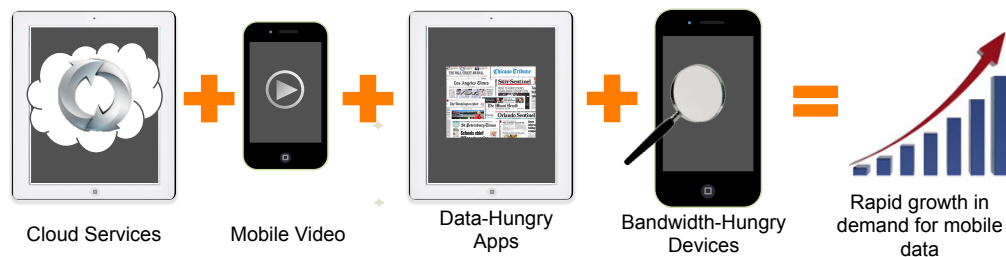

Figure 1: Factors driving the demand for mobile data.

Cloud Services and M2M Applications: Cloud-based services that synchronize data across multiple mobile devices, such as iCloud, Dropbox, and Amazons Cloud Drive, can be a significant factor in traffic growth for ISPs [90]. Similarly, machine-to-machine (M2M) applications that generate data intermittently (e.g., sensors and actuators, smart meters) or continuously (e.g., video surveillance) often load the network with large signaling overhead [36]. However, these traffic types also have some intrinsic time elasticities that create opportunities for intelligently shifting them to low-congestion times through pricing incentives.

Mobile Videos: Video has also been a major contributor to mobile data traffic growth, accounting for 51 percent of global mobile data traffic at the end of 2012. It is expected to account for 66 percent of global mobile data traffic by 2017 [26]. A study by Gartner [129] states that the worldwide mobile video market had 429 million mobile video users in 2011, projected to grow exponentially to 2.4 billion users by 2016. Smartphones and tablet sales will contribute 440 million new mobile video users during the forecast period. The report also forecasts that the worldwide share of mobile video connections on 3G/4G will increase from

18% in 2011 to 43% in 2015 [29]. These growth rates are being further fueled by mobile video content delivery via mobile-optimized websites and video advertisements.

Capacity-Hungry Applications: The popularity of handheld devices has led to rapid growth in the development of other bandwidth-hungry applications for social networking, music, personalized online magazines, etc. in addition to file downloads and video streaming. Virgin Media Business reports that the average smartphone software uses 10.7 MB per hour, with the highest-usage app, Tap Zoo, consuming up to 115 MB/hour. In the current ecosystem, app developers do not have enough incentives to account for network conditions, and consequently many smartphone apps are not optimized for bandwidth consumption.

Bandwidth-Hungry Devices: The widespread adoption of handheld devices, equipped with powerful processors, high-resolution cameras, and larger displays, has made it convenient for users to stream high-quality videos and exchange large volumes of data. Data from laptops with 3G dongles and netbooks with wireless high-speed data access contributes the most to wireless network congestion [36]. As for smartphones, Cisco projects that the average monthly data usage will rise from 150 MB in 2011 to 2.6 GB in 2016 [26]. New features like Siri on the iPhone 4S, which has doubled Apple users' data consumption, are driving this growth [10].

The key takeaways from the above discussion of the various factors contributing to network congestion are:

- Different types of applications and services have different levels of time elasticity of demand (e.g., cloud backup versus financial applications).

- There is a great variance in the rate of growth of different types of applications (e.g., demand for mobile videos is growing fast).

- Different applications consume bandwidth at different rates and have a large variance in QoE requirements (e.g., many apps are not optimized for bandwidth while some well-designed apps can adapt to available network QoS).

- There is a growing variance in the bandwidth requirements of different smart devices (e.g., iPad versus iPhone versus feature phones).

These factors contribute to the need for smarter data plans that can account for the variances across users' QoE needs, time elasticity of demand, application traffic characteristics and their willingness to pay for the service.

Before delving deeper into SDP's promise in addressing congestion issues [99], in the next section we first explore how these trends are impacting the various stakeholders of the network ecosystem, i.e., network operators, consumers, app developers, and content providers.

3 Impact on the Network Ecosystem

3.1 ISPs' Traffic Growth

By 2016, ISPs are expected to carry 18.1 petabytes per month in managed IP traffic.[5] But this growth is causing concern among ISPs, as seen during Comcast's initiative to cap their wired network users to 300 GB per month [30]. Even back in 2008, Comcast made headlines with their decision (since reversed) to throttle

---

[5]Cisco's definition of "managed IP" includes traffic from both corporate IP wide area networks and IP transport of television and video-on-demand.

Netflix as a way to curb network congestion [69]. Video streaming from services like Netflix, Youtube, and Hulu, are a major contributor to wired network traffic. In fact Cisco predicts that by 2016 fixed IPs will generate 40.5 petabytes of Internet video per month [25].

Rural local exchange carriers (RLECs) are also facing congestion in their wired networks due to the persistence of the middle-mile problem for RLECs. Although the cost of middle mile bandwidth has declined over the years (because of an increase in the DSL demand needed to fill the middle mile), the bandwidth requirements of home users have increased quite sharply [43]. Still, the average speed provided to rural customers today fails to meet the Federal Communications Commission's (FCC) broadband target rate of 4 Mbps downstream speed for home users. The cost of middle mile upgrades to meet this target speed will be substantial and is a barrier to digital expansion in the rural areas [43]. Research on access pricing as a mechanism to bring down middle mile investment costs by reducing the RLEC's peak capacity and over-provisioning needs can therefore also help in bridging the digital divide.

3.2   Consumers' Cost Increase

Network operators have begun to pass some of their network costs to consumers through various penalty mechanisms (e.g., overage fees) and increasing the cost of Internet subscriptions. For instance, when Verizon announced in July 2012 that they were offering shared data plans for all new consumers and discontinuing their old plans, many consumers ended up with higher monthly bills [17]. To remain within monthly data caps, consumers are increasingly relying on usage-tracking and data compression apps (e.g., Onavo, WatchDogPro, DataWiz) [101] that help to avoid overage fees. Such trends are common in many parts of the world; in South Africa, for instance, consumers use ISP-provided usage-tracking tools [19] to stay within the data caps. Similarly in the U.S., research on in-home Internet usage has shown that many users are concerned about their wired Internet bills and would welcome applications for tracking their data usage and controlling bandwidth rates on in-home wired networks [21, 77]. Empowering users to monitor their data usage and control their spending has led to a new area of research that considers economic incentives and human-computer interaction (HCI) aspects in a holistic manner [112].

3.3   Application Developers' Perspective

Introducing pricing schemes that create a feedback-control loop between the client side device and network backend devices requires new mobile applications that will support such functionalities. However, most mobile platforms in use today (e.g., iOS, Android, and Windows) have different levels of platform openness. The iOS platform for iPhones and iPads has several restrictions: it strictly specifies what kind of applications can run in the background and further prevents any access other than the standard application programming interfaces (APIs). For example, obtaining an individual application's usage and running a pricing app in the background are prohibited. By contrast, the Android and Windows platforms allow these features, e.g., introducing an API to report individual applications' usage to third-party apps.

An interesting direction to overcome these limitations is to initiate the creation of open APIs between user devices and an ISP's billing systems. For example, this can allow the user devices connected to the ISP's network to easily fetch current pricing, billing, and usage information from the network operator, while also allowing the ISP to easily test and deploy new pricing schemes through the standardized interface. Such an API would foster innovations in pricing for both consumers and providers.

Additionally, new pricing plans create an opportunity for developers to optimize their app according to changing pricing conditions. For instance, some apps that require preloading content, such as magazine apps, might time these preloading downloads so as to coincide with lower-price times, thus saving users

money [112]. This sensitivity to price might even improve users' experience, as lower prices generally occur during times of lower congestion and higher throughput. Shifting usage so as to save money could be especially significant for video apps, as these tend to have higher usage volumes.[6] Such adaptation would also require an API allowing apps to access the network prices in real time.

3.4 Software/Hardware Limitations

Wireless ISPs' current billing systems (including 2G, 3G, and 4G) heavily depend on the RADIUS (Remote Authentication Dial In User Service) protocol, which supports centralized Authentication, Authorization, and Accounting (AAA) for users or devices to use a network service [102]. In particular, RADIUS accounting [120] is well suited to support usage-based pricing, since it can keep track of the usage of individual sessions belonging to each user. Yet individual session lengths are often quite long, making it difficult to retrieve usage at the smaller timescales needed for dynamic pricing.

RADIUS account sessions are initiated by the Network Access Server (NAS) when a user first attempts to connect to the network: the NAS sends a user's login credentials to the RADIUS server, which compares the credentials to a secure database. The RADIUS server then *authenticates* the session and *authorizes* it to access different functionalities on the network. Once this session has been initiated, a start record is created in the RADIUS logs. Interim *accounting* request messages can be sent periodically to update the usage information. When the end user terminates the connection, the NAS sends a stop message to the RADIUS server and a stop record is created that stores the total usage volume of that session. Since these RADIUS sessions can have very long durations, up to several hours, RADIUS logs cannot be used to calculate usage at smaller timescales.[7] Moreover, the RADIUS log has no information on the type of application(s) corresponding to each session. While one session may encompass usage from multiple apps used in parallel, in some cases individual apps initiate new sessions; thus, the concept of a "session" cannot be definitively tied to an individual app [120].

Changing the RADIUS protocol to deliver usage estimates at a smaller time granularity would require significant overhead in both control signaling and storing RADIUS records. A perhaps easier alternative would be to record usage at the network edge, i.e., on client devices–such functionality already exists, but this approach would be vulnerable to users' deliberately falsifying the usage recorded on their device. Similarly, RADIUS logs do not contain any information on per-application usage, but client devices can easily obtain this information. Thus, application-specific pricing could also benefit from usage tracking functionalities on the end user devices. Some verification procedures could be implemented to guard against user tampering, e.g., comparing the total monthly usage measured by RADIUS servers and client devices, but would require careful design and might not be completely secure.

3.5 Content Delivery Issues

Any change in access pricing has to be studied in the larger context of Internet's net-neutrality and openness. These discussions center around the issues of (a) who should pay the price of congestion (i.e., content providers or consumers) and (b) how such pricing schemes should be implemented (i.e., time-of-day, app-based bundles, etc.). The major concern with policy change is the possibility of paid prioritization of certain content providers' traffic, price discrimination across consumers, and promoting anti-competitive behavior in

---

[6]Some video apps cannot shift usage due to legal restrictions on caching content. However, many apps like YouTube own the rights to their video content.

[7]Note that interim update messages are sent periodically when a session joins the system, and hence, the time interval for interim updates should be kept low to support sending time-of-day usage, which may introduce significant control overhead.

bundled offerings of access plus content. While such developments can indeed hurt the network ecosystem, one aspect that should receive more attention is the threat to data usage even under simple usage-based or tiered data plans. As Internet users become more cautious about their data consumption [121], content providers are providing new options to downgrade the quality of experience (QoE) for their users to help them save money. For instance, Netflix has started allowing "*users to dial down the quality of streaming videos to avoid hitting bandwidth caps*" [89]. Additionally, it is "*giving its iPhone customers the option of turning off cellular access to Netflix completely and instead relying on old-fashioned Wi-Fi to deliver their movies and TV shows*" [39]. Thus, the ecosystem today is being driven by an attitude of penalizing demand and lessening consumption through content quality degradation.

Network researchers are investigating these issues broadly along two lines of work: (i) opportunistic content caching, forwarding, and scheduling, and (ii) budget-aware online video adaptation. Opportunistic content delivery involves the smart utilization of unused resources to deliver higher QoE; for example, to alleviate the high cost of bulk data transfers, Marcon et al. [84] proposed utilizing excess bandwidth (e.g., at times of low network traffic) to transmit low-priority data. Since this data transmission does not require additional investment from ISPs, they can offer this service at a discount, relieving data transfer costs for clients. While utilizing excess bandwidth introduces some technical issues (e.g., the potential for resource fluctuations), a prototype implementation has shown that they are not insurmountable [73]. The second stream of works on online video adaptation systems, such as Quota Aware Video Adaptation (QAVA) [18], have focused on sustaining a user's QoE over time by predicting her usage behavior and leveraging the compressibility of videos to keep the user within the available data quota or her monthly budget. The basic idea here is that the video quality can be degraded by non-noticeable amounts from the beginning of a billing cycle based on the user's predicted usage so as to avoid a sudden drop in QoE due to throttling or overage penalties when the monthly quota is exceeded. This relates to the SDP theme of enabling self-censorship of usage and QoE on the client side device through user-specified choices.

3.6 Regulatory Concerns

Pricing in data networks has remained a politically charged issue, particularly for pricing mechanisms that could potentially create incentives for price discrimination, non-neutrality, and other anti-competitive behavior through app-based pricing or bundling of access and content. Academics have already cautioned that the ongoing debate on network neutrality in the U.S. often overlooks service providers' need for flexibility in exploring different pricing regimes [137]:

> *Restricting network providers' ability to experiment with different protocols may also reduce innovation by foreclosing applications and content that depend on a different network architecture and by dampening the price signals needed to stimulate investment in new applications and content.*

But faced with the growing problem of network congestion, there has been a monumental shift in the regulatory perspective in the US and other parts of the world. This sentiment was highlighted in FCC Chairman J. Genachowski's 1 December 2010 statement [106], which recognizes *"the importance of business innovation to promote network investment and efficient use of networks, including measures to match price to cost."*
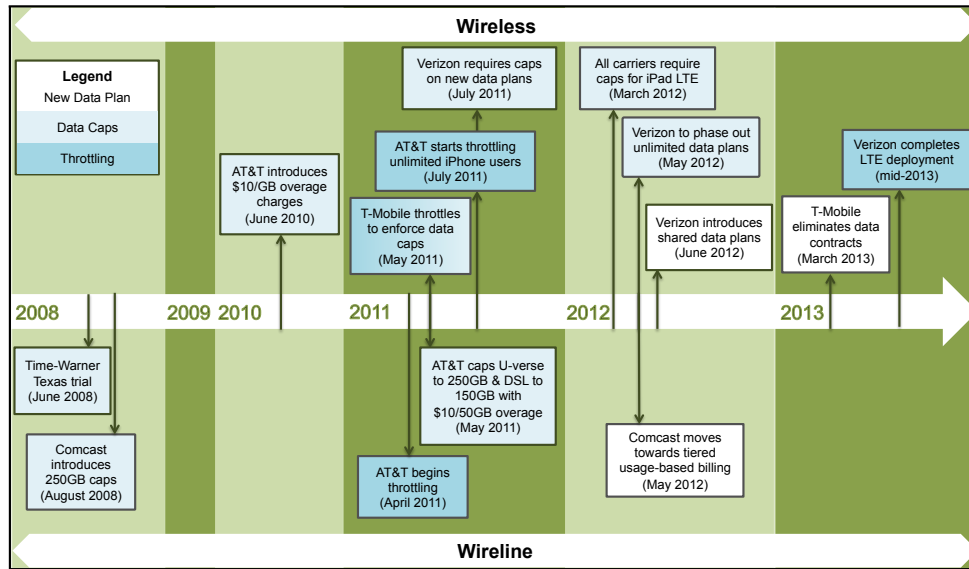
Figure 2: Broadband pricing plans offered by major U.S. ISPs, 2008 - 2013.

4 Smart Data Pricing

Broadband access pricing and demand control practices have rapidly evolved among U.S. ISPs since 2008, as seen in Figure 2. Over the past few years, ISPs around the world have started to offer innovative pricing plans, including usage-based and app-based pricing to tackle the problem of network congestion [114]. Smart Data Pricing (SDP) [107] is an umbrella term for a suite of pricing and policy practices that have been proposed in the past or are being explored as access pricing options by operators instead of the traditional flat-rate model. Such SDP models can include any or of the following mechanisms or a combination, which will be discussed later in the chapter: (a) Usage-based pricing/metering/throttling/capping, (b) Time/location/congestion-dependent pricing, (c) App based pricing/sponsored access, (d) Paris metro pricing, (e) Quota-aware content distribution, (f) Reverse billing or Sponsored Content. SDP does not even need to be an explicit pricing mechanism; it can be another form of innovative congestion management like WiFi offloading or "fair-throttling"[8].

The basic ideas of congestion pricing have received much attention as a research topic both in computer networks and information systems literature, and are once again getting a fresh look from academics in recent years. Given the change in the economic and regulatory environment of Internet pricing, it is likely that some of the ideas will be realized in future data plans. However, research in the design of such smart data pricing plans should account for some new factors: (i) the growth in traffic with high time-elasticity of demand (e.g., downloads, P2P, cloud backup, M2M) and the ability to schedule such traffic to a less congested time without user-intervention, (ii) revisiting the issue of dividing the elements of a congestion control-feedback loop between the network backend and the smart end-user devices, (iii) development of new system architectures to deploy these pricing ideas and demonstration of their potential benefits through field trials. In other words, it requires understanding both the *economic theory* of pricing models as well as the *systems engineering* and *human-computer interaction* aspects of realizing such data plans. These require

---

[8]Fair throttling involves accounting for user's usage history of contributing to congestion in determining what share of available bandwidth the user should receive in a congested time.
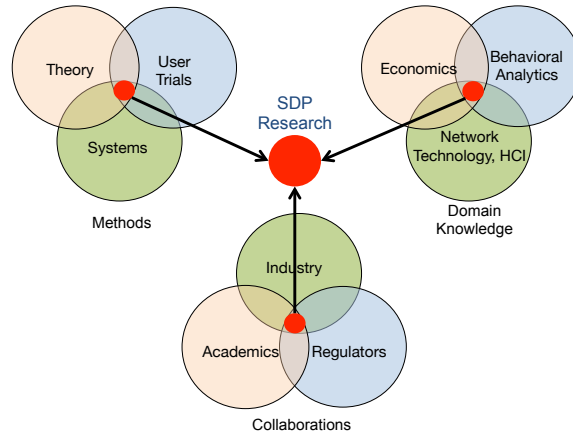
Figure 3: Smart Data Pricing research components

a multi-disciplinary approach in SDP research that bridges theory, systems, and user trials by drawing on economic theory, network engineering and user behavioral studies in a collaborative environment, as shown in Figure 3.

5   A Review of Smart Data Pricing

Smart data pricing encompasses a wide variety of different pricing algorithms and proposals. In this section, we briefly discuss some of these ideas, following the taxonomy given in Figure 4. We include a brief overview of related pricing plans in the electricity and transportation industries, which can help yield insights into the feasibility of various forms of SDP for data, as well as ideas for new pricing plans. Other, more thorough reviews may be found in [33, 115, 118].

A primary goal of SDP is to create the right incentives (or price points) for users to modify their usage behavior so as to help ISPs with better resource allocation and utilization. But creating these incentives requires ISPs to account for users' responses to the prices offered. Of particular relevance is the timescale associated with the pricing mechanism – do the prices continually change as the network load changes? If so, how frequently and by how much? How to balance the trade-offs between users' reluctance to real time dynamic pricing and the inability of static pricing to exploit the time elasticity of demand of different applications in congested times? How to balance the trade-offs between the users' need for transparency and control over her usage and the need for automation in dynamic pricing scenarios?

Static pricing plans are those that change prices on a relatively longer timescale, e.g., months or years: the offered prices do not vary with immediate changes in the network congestion level. The popularity of these plans arises from the certainty they provide to a user's expected monthly bill. For instance, tiered data plans with pre-specified rates are prevalent in the United States, and several European and Asian ISPs offer usage-based pricing in which users are charged in proportion to their usage volume. But such usage-based pricing leaves a timescale mismatch: ISP revenue is based on monthly usage, but peak-hour congestion dominates its cost structure (e.g., network provisioning costs increase with the peak-hour traffic). Another well-known pricing plan is time-of-day (ToD) pricing, in which users are charged higher prices during certain "peak" hours of the day. But even with ToD pricing, the hours deemed as "peak" are fixed, which results in two challenges. First, traffic peaks arise in different parts of the network at different times, which can be hard
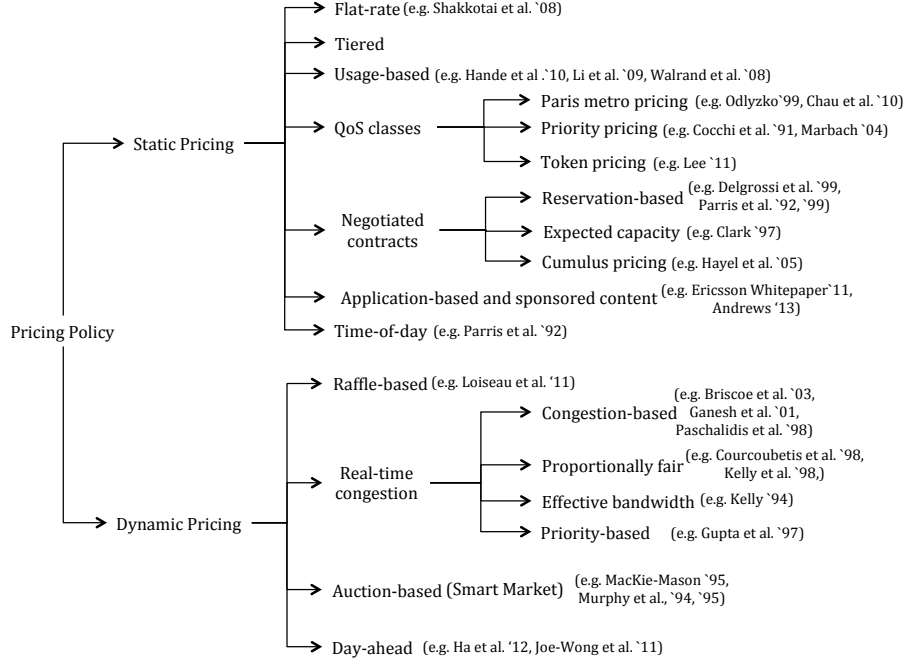
Figure 4: Examples of broadband pricing plans proposed in the research literature.

to predict in advance and could end up creating two peaks during the day–one during peak periods, for traffic that cannot wait for several hours for lower-price periods, and another peak during discounted "off-peak" periods for time-insensitive traffic. Such patterns have been observed in dynamic pricing for voice calls in operational networks [125]. We discuss several of these existing static pricing plans and proposals in greater detail in Section 5.1.

Dynamic pricing takes the ToD idea further in that it does not pre-classify peak and off-peak periods, instead adjusting prices at a finer timescale or by location in response to the network congestion. However, prices that vary depending on the current network load can be sometimes inconvenient for users. Hence, dynamic pricing variants for SDP, such as automated "smart market" [80, 88], raffle-based pricing [78], and day-ahead pricing [48], have been proposed to guarantee the prices a day in advance to give users some certainty about the future prices on offer. Each day, new prices are computed for different times (e.g., hours) of the next day, based on predicted congestion levels. A detailed discussion on these dynamic pricing proposals will be provided in Section 5.2.

5.1 Static Pricing

Due to the fixed nature of their prices, static data plans do not generally allow ISPs to adapt to real-time congestion conditions. In particular, the ISP cannot prevent or alleviate network congestion at peak times by manipulating the prices. On the other hand, static pricing tends to be more acceptable to users, as it offers more certainty and is simpler than dynamically changing prices. Indeed, the most basic form of static pricing, *flat pricing*, is also the most simple for users, though it does not impose any sort of usage incentives [116]. Some other important examples of static pricing include the following:

Usage-based: In its purest form, usage-based pricing charges users in proportion to the amount of data that they consume, without regard to the type of data (e.g., application) or time of consumption. The principal advantage of such a pricing plan lies in its relative simplicity: it imposes a monetary penalty on heavy (i.e., high-usage) users to reduce congestion [50, 76], but also penalizes users even when the network is lightly loaded. Moreover, usage-based pricing requires users to keep close track of their usage in order to determine how much they have spent on data [132].

Tiered: A more common variant of pure usage-based pricing is tiered pricing, in which users pay a fixed amount of money for a monthly *data cap* (e.g., $30 for 3GB). This fixed fee covers usage up to the cap, after which users may pay another fixed fee to increase the cap by a discrete amount, e.g., $10 per extra GB. Thus, tiered or capped pricing can be viewed as a discretization of usage-based pricing. Many ISPs have adopted such a pricing plan or another variant in which the data cap is shared across several devices (i.e., a *shared data plan*). Like usage-based pricing, tiered pricing is simple for users to understand and penalizes heavy usage.

Quality of Service (QoS) classes: Some static pricing plans offer multiple traffic classes with different qualities of service (QoS). A simple differentiated pricing plan is *Paris metro pricing* (PMP), which is named after an actual pricing practice on the Paris metro in the 1900s [92]. In Paris metro pricing, the ISP separates data traffic into different logical traffic classes and charges different prices for logically separate traffic classes (i.e., each class is identical to the others in their treatment of data packets). Only users willing to pay a higher price will adopt this traffic class, which leads to a better QoS due to fewer users. But one of the key issue of academic debate related to PMP has been its viability, namely, whether it is a mechanism to increase the profits for service providers, or whether it achieves higher social welfare. As pointed out in [16], the conclusions of this debate depend on how users react to the congestion externality of the underlying system. Other researchers have investigated more direct forms of QoS pricing, in which users can indicate their desired QoS in their packets and are charged a higher per-byte fee for higher QoS [9, 28, 83].

Another form of QoS pricing is *token pricing*, in which users receive tokens at a fixed rate (e.g., 1 per minute) [74]. Users can then spend these tokens to send some of their traffic at a premium QoS; users can choose the timing of these premium sessions, e.g., to coincide with their individual priorities and preferences.

Negotiated contracts: In these types of pricing schemes, users pre-negotiate contracts with the ISP regarding the price of sending traffic over the network. The main research question for such contracts is then characterizing this user-ISP interaction and both parties' optimal decisions. For instance, in *reservation-based pricing*, users specify a monthly budget for data; the ISP can then accept or reject users' connections based on users' remaining budget and the real-time network congestion [34, 94, 95].

In *expected capacity pricing*, Clark proposed a mechanism in which users similarly negotiate a price in advance based on an "expected" quality of service (e.g., file transfer time), so that at congested times the ISP can freely allocate network resources based on whether a given packet lies "within" a user's purchased traffic profile [27]. The goal of this pricing scheme is to "*provide additional explicit mechanisms to allow users to specify different service needs, with the presumption that they will be differentially priced* [27]." Expected capacity pricing allows users to explicitly specify their service expectation (e.g., file transfer time), while accounting for differences in applications' data volume and delay tolerance. The idea is that by entering into profile contracts for expected capacity with the operator, different users should receive different shares of network resources when the network gets congested [118]. One specific proposal to realize this service involved traffic flagging (i.e., each packet is marked as being *in* or *out* of the user's purchased profile, irrespective of network congestion level) by a traffic meter at access points where the user's traffic enters the network. This is followed by congestion management at the switches and routers where packets marked as *out* are preferentially dropped during congested periods, but are treated in an equal best-effort manner at all other times. The expected capacity is thus not a capacity guarantee from the network to the user, but rather

a notion of the capacity that a user expects to be available and a set of mechanisms that allow the user to obtain a different share of the resource at congested times.

An ISP offers similar contracts under *cumulus pricing*, but users can re-negotiate the price after passing "cumulus" usage points [52]. Cumulus pricing consist of three stages: specification, monitoring, and negotiation. A service provider initially offers a flat-rate contract to the user for a specified period based on the user's estimate of resource requirements. During this time the provider monitors the user's actual usage and provides periodic feedback to the user (by reporting on "cumulus points" accumulated from their usage) to indicate whether the user has exceeded the specified resource requirements. Once the cumulative score of a user exceeds a predefined threshold, the contract is renegotiated.

App-based and sponsored content: Different applications consume different amounts of data traffic (e.g., streaming video consumes much more data than retrieving emails). Some researchers have thus proposed app-based pricing, in which users are charged different rates for different apps [38]. Such pricing plans also include "zero-rated" apps, whose traffic is free for the user. A variant of such pricing schemes is "sponsored content", in which a third-party (advertiser, content provider, or the ISP itself) "sponsors" some part of the traffic in return for accessing specific content or using data at less congested times.

App-based plans have been offered in Europe, largely on a promotional basis. However, app-based pricing presents technical challenges for ISPs– ISPs need to identify and track how much data each user consumes on specific applications, which may raise privacy concerns. Moreover, some apps open links in separate apps (e.g., links in Flipboard may open a separate Internet browser), creating confusion among users as to the app to which some traffic belongs, and whether this traffic counts towards the sponsored volume or not. Even in academia, sponsored content research is relatively sparse, though a few initial models have been developed [5, 49].

Time-of-day (ToD): ToD pricing charges users different usage-based rates at different times of the day (e.g., peak and off-peak hours) [94]. The free nighttime minutes offered for voice calls by most US ISPs before 2013 are one simple form of ToD pricing. However, as the peak times and rates are fixed in advance, ToD pricing can end up creating two peaks, one during the "peak" period and one in the "off-peak" period; indeed, this phenomenon was observed in Africa when MTN Uganda offered discounted prices for voice calls made at night.

Some ISPs offer two-period ToD pricing plans with different charging rates at day and night times. For example, BSNL in India offers unlimited night time (2-8 am) downloads on monthly data plans of Rs 500 ($10) and above. Other variations of ToD pricing are offered elsewhere; for instance, the European operator Orange has a "Dolphin Plan" for £15 ($23.50 USD) per month that allows unlimited web access during a "happy hour" corresponding to users' morning commute (8-9 am), lunch break (12-1 pm), late afternoon break (4-5 pm), or late night (10-11 pm). The underlying idea is to allow consumers to *self-select* themselves into "time-buckets" with QoE guarantees, with the hope of exploiting the variance in consumers' time-of-day preference to spread out demand more evenly over the day.

5.2 Dynamic Pricing

Dynamic pricing allows prices to be changed in (near) real-time, which unlike static pricing allows an ISP to adjust its prices in response to observed network congestion. However, in doing so the ISP significantly complicates its pricing, making it much harder for users to understand. Thus, implementing and offering dynamic pricing plans requires ISPs to account for human factors that can make real-time changes in price more amenable to users. Some of the proposed dynamic pricing plans are discussed below:

Real-time congestion: If ISPs can monitor their network for real-time signs of congestion, they can increase prices when congestion is observed, and decrease them when the traffic load is relatively light. Thus,

there is a *feedback loop* between ISPs offering prices and users correspondingly adjusting their usage [41,96]. This *responsive pricing* sets prices so as to keep user demand under a certain level; if an ISP further chooses the prices so as to optimize a proportional fairness criterion on the amount of bandwidth allocated to different users, we obtain *proportional fairness pricing* [32,42,68]. Many variants of responsive pricing have been proposed in the literature, principally as a congestion control mechanism; in practice, it would be impractical for users to manually respond to the prices offered for each Internet connection. Hence, automation of client devices (or agents) to intelligently adapt their data consumption will be necessary to realize such real-time pricing. But recent HCI studies [20,112] have revealed complex patterns of household politics and user opinion regarding such decision-making about bandwidth consumption. In particular, there is a reluctance among users to delegate such bidding or scheduling to automated agents that stems from a conflict between the psychological assurance of manual control and the convenience of automation, which in turn depends on the perceived trust-worthiness of the underlying system. Many of the findings reported later in this chapter on user behavior and user interface design may serve as guidelines in designing user-friendly client-side agents to enable such pricing plans.

Another form of congestion pricing, *effective bandwidth pricing*, incorporates a form of QoS by charging users based on their connection's peak and mean rates [66]. One can also explicitly incorporate different QoS by using *priority pricing*, in which users can pay less by accepting a longer delay at congested times [47]. If the prices are chosen correctly, the system reaches an equilibrium, in which each user's packets are processed within the delay paid for.

Auction-based: One disadvantage of real-time congestion pricing is that in practice, the ISP must set the prices (just) before observing user behavior. Since user demand can change with time, the ISP may end up setting non-optimal prices due to outdated assumptions of user demand. "Smart market" pricing addresses this slight delay with an auction-like scheme, in which users attach a bid to their packets that signifies their willingness to pay [80,88]. ISPs then admit a limited number of packets in descending order of the bids so as to limit network congestion. Users are charged the lowest bid admitted, which represents the "cost of congestion." While smart market pricing allows true real-time pricing, it also requires automated agents on user devices to make bids as necessary and keep track of the final amount charged.

Raffle-based: This is a variation of dynamic time-dependent pricing inspired by lottery reward mechanism. Under *raffle-based pricing*, the exact price that users pay is determined after-the-fact, i.e., in a probabilistic manner that depends on the amount of data consumed by a user [78]. Users have a chance to receive a monetary reward during congested times if they agree to shift their demand to less-congested times. They are entered into a lottery for a fixed reward, where the probability of winning the lottery depends on the user's contribution to the total amount of traffic shifted. While such a pricing plan is attractive to ISPs in that the total reward offered is fixed, users may be less willing to shift their traffic because of the uncertainty in winning the lottery and the reward amount, which depend on external factors like the behavior of other users.

Day-ahead time-dependent: In an effort to increase user certainty of the prices, ISPs can guarantee their time-dependent prices one day in advance, and continue to compute new prices to maintain this sliding one-day window of known prices [48,61]. Users can then plan their usage in advance, while ISPs can adapt their estimates of user behavior and usage volume in calculating the prices for subsequent days. Day-ahead pricing thus strikes a balance between user convenience and ISP adaptability. This has been a successful pricing mechanism in the electricity market, and hence can be adapted to broadband networks with careful consideration. A schematic of the resulting feedback loop is shown in Figure 5. In the next section, we examine a prototype of day-ahead pricing for mobile data in order to illustrate the "end-to-end" nature of an SDP deployment.

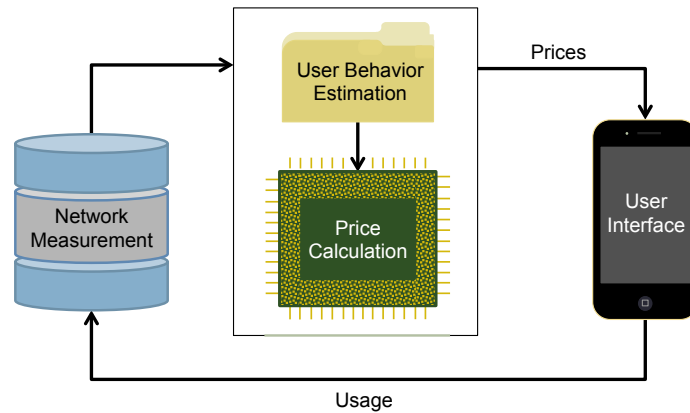We pause to briefly compare day-ahead TDP with the other types of dynamic pricing discussed above.

Figure 5: Feedback loop schematic of day-ahead pricing.

Real-time pricing and Smart Market mechanisms require users to delegate some control and operate in an automated mode as the time-scale is too short for user-mediated choices. On the other hand, simple 2-period (day & night) time-of-day pricing has time scale that is too long to take advantage of any spare capacity availability and time elasticity of demand which vary at much shorter time-scale. Auction-based mechanisms will require modifications to the network equipments (e.g., agents to recognize bid amounts and perform admission control) and client side agents for automated bidding. Raffle-based pricing creates uncertainty in rewards and unless the time-varying prices are known in advance, users may be reluctant to adopt such data plans. A day-ahead dynamic time-dependent pricing plans solves many of these issues by providing guarantees on the future prices in advance, takes advantage of demand elasticities at shorter time-scales, and provides ISPs with a mechanism to optimize the prices they offer.

But what are the challenges of realizing dynamic day-ahead time-dependent pricing?

- How to develop an economic model for dynamic day-ahead TDP which computes optimized prices that accounts for users' time elasticity of demand in maximizing the total revenue of the network provider? The price computation needs to consider (a) the cost incurred in offering price discounts, (b) savings from shifting some traffic from peak to off-peak hours, (c) the increase in baseline demand in discounted periods due to potential "sales day" effect.

- How to engineer a system that enables this pricing by developing both provider and client-side modules (in particular, the user interfaces needed for users to react to the offered prices)?

- How can researchers carry out field trials of such pricing plans by interposing themselves as a "bandwidth reseller" between the network providers and its real consumers? We will address these questions in Sections 6.4, 7, 8, and 9.

5.3 Comparison with other Markets: Similarities and Differences

Let us now take a look at what forms of time-dependent pricing have been already field tested and exist in the real world in networks that suffer from congestion problems to identify differences and opportunities for innovating TDP plans for broadband networks. Much like today's data networks, the electricity and transportation markets have both experienced a capacity shortage over the past decade and have developed new pricing plans to cope with the resulting shortfall. By comparing electricity usage and road traffic to data

traffic, we see that these industries are quite similar to data networks, and that their pricing plans may inform SDP for mobile data. Indeed, both industries observe a highly variable demand throughout the day, allowing for both static and dynamic pricing plans. In particular, time-of-day road tolls have been offered in many transportation networks, and many electricity utilities have both trial-ed and deployed time-of-day pricing. We give an overview of such pricing plans in this section, with the aim of highlighting the unique challenges posed by refining such pricing plans to accommodate broadband data networks. Figure 6 gives an overview of the analogies between pricing plans proposed for the transportation, broadband, and electricity industries.

The similarities and differences between these pricing plans reflect the different industries for which they are designed. In particular, we observe the following distinctions:

1. ***Real-time communication:*** User devices on data networks, e.g., smartphones, are capable of real-time communication with the ISP network, for instance if the prices change in real time. But such real-time feedback for price (toll) changes in road networks is harder to realize and will require additional infrastructure support. In electricity markets, new smart grid interfaces have been developed that can display real-time prices, but individual devices, e.g., air conditioners or vacuum cleaners, generally cannot interact directly with the provider smart grid and require a smart energy controller to schedule their energy consumption.

2. ***Elasticity of demand:*** Smartphones' ability to communicate with the ISP network in real time is complemented by users' ability to easily control their usage on individual devices and applications. For instance, a user could simply stop streaming a video if the price increases; such measures could also be automated within the device. The users' decisions will reflect the large variance in the demand elasticity of different types of applications (some of which, such as software downloading, P2P, file backup may not even require user participation and can be completed in small chunks whenever low prices are available). In contrast, devices on electricity networks typically consume energy constantly as long as they are active. There is little opportunity for many devices (e.g., washer, dryer, lights) to complete their activities in an intermittent manner without requiring active user engagement. In road networks, the contrast is even more stark; users in the network (e.g., already driving) cannot easily exit or postpone their activity.

3. ***Long-term volatility:*** Most people do not have a concrete idea of how much data they consume each month, partly because most data plans charged a flat fee for unlimited access until recently. Moreover, an individual's data usage can vary greatly from day to day, as relatively casual actions such as streaming a video can have a large impact on total data consumption. In contrast, most people have a relatively good idea of how much they drive per day, and the distance traveled, and road toll fees. Thus, people may be more able to plan ahead by buying permits (e.g., EZ pass) or carpooling during congested hours. In electricity markets, household demand similarly does not vary much from day to day. Consumption of electricity is largely driven by user *needs*, rather than the more volatile *preferences* that drive demand for Internet data.

5.3.1 Static Pricing

Traditional road pricing has been simple flat-rate cordon pricing, analogous to flat pricing of data. Pricing by vehicle type, analogous to app-based pricing for data, has also been proposed, e.g., charging trucks more than passenger vehicles [127]. Forms of flat-rate priority pricing have also been implemented, most obviously in the Paris metro's pricing scheme from which data networks' Paris metro pricing takes its name. High-occupancy vehicle or "carpool" lanes can also been seen as analogous to priority pricing, in that users
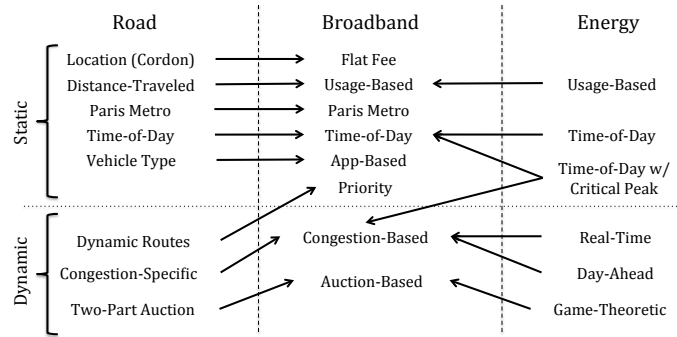
237

Figure 6: Comparison of pricing plans in the transportation, broadband, and electricity industries.

can self-select to take advantage of less-congested HOV lanes by paying the higher "price" of carpooling with other passengers.

In a common variation on flat-rate tolls in road networks, the flat-rate toll can vary depending on the time of the day [45], for a pricing plan analogous to time-of-day pricing. However, such charges are still flat-rate, i.e., they do not depend on the distance traveled over the road network. Distance-traveled pricing, analogous to usage-based pricing in broadband networks in that users' charge is proportional to the distance traveled, has also been proposed for transportation networks, and has been offered in Taiwan and the U.S. [56, 134]. In fact, the Taiwanese implementation varies the distance-traveled price depending on the time of the day; it is thus a form of time-of-day pricing.

Time-of-day pricing is the major form of static pricing practiced in the electricity industry. Most trials of time-of-day pricing for electricity markets have focused on peak/off-peak pricing, as electricity demand generally follows a less variable pattern than data demand, with extremely low demand at night and higher demand during the day. For instance, one major source of electricity consumption is air conditioning in the summer, which follows a fairly regular pattern of being on during the day and off at night. Indeed, many trials have shown time-of-day pricing to be effective in reducing excess demand during peak hours. One popular variant that has also been trial-ed is *critical peak pricing*, in which certain days are designated as "critical," e.g., especially hot days during the summer. On these critical days, the peak price goes up to increase users' incentives to reduce demand. Some studies with California consumers have shown that critical-peak pricing is much more effective than simple peak/off-peak pricing [15, 55]. In this trial, users with "smart devices" that automatically reduce energy consumption reduced their usage almost twice as much as other users, indicating that user interfaces for interacting with prices are critical to the success of dynamic or time-of-day pricing plans.

5.3.2 Dynamic Pricing

Congestion-based pricing has been proposed in both the transportation and electricity industries. One form of congestion pricing in road networks charges users at a price-per-mile rate that is based on their average speed. However, though considered in Cambridge, U.K., this pricing plan was never implemented [45]. A more complex pricing plan proposed using several dynamic origin-destination models to compute effective route costs depending on real-time congestion conditions in the road network [64]. Drivers would then be able to take shorter routes for higher prices; however, computing these prices is highly non-trivial, and it would be difficult to communicate the prices of different routes to drivers in the network.

One variation on dynamic pricing for road networks involves a *secondary market*, in which governments can sell permits to pass through congested areas. Users can then form a market to sell these permits [119]. However, similar pricing schemes have not yet been proposed for data networks, likely due to the difficulty in setting up a secondary market among users. Moreover, the increasingly ubiquitous nature of data connectivity has made it more impractical to ask users to completely refrain from consuming data at congested times.

Some electricity pricing researchers have argued that dynamic pricing can lead to significant gains over simple ToD pricing [7]. Both congestion pricing and auction pricing have been proposed for electricity markets; however, such works often have a more consumer-focused outlook than do pricing proposals for data. In an auction-based electricity market, electricity distributors can make dynamic offers to users (i.e., households) who respond with real-time electricity purchases. Auction schemes have been proposed that take into account varying electricity capacity, which can significantly improve market efficiency [131].

Many papers have studied responsive dynamic pricing from a user's perspective of predicting future prices and scheduling devices accordingly. A game-theoretic framework can be used to model users' scheduling of energy usage as a cooperative game; if users cooperate, the total demand on a network can then be reduced, enhancing efficiency [12]. Other works propose algorithms to predict prices in advance [35, 86] and schedule user devices accordingly; users thus try to anticipate electricity providers' real-time pricing. This price prediction is not necessary with day-ahead pricing, though day-ahead pricing offers electricity providers less flexibility [63]. However, such prediction and scheduling algorithms, which have received relatively little attention for data usage, might help make dynamic congestion pricing for data more amenable to users.

Other papers consider users' actions in conjunction with the provider's price determination [8]. Such approaches can facilitate a study of social welfare, and may incorporate uncertainty in supply and demand [11, 14, 105]. One may also consider a feedback loop between users and an electricity provider, which can yield real-time pricing algorithms analogous to those for dynamic congestion control in data networks [103]. Some works have also considered appliance-specific models of user demand, analogous to different applications having different demands for data [75]. A unique feature of these models is the ability to store electricity, e.g., in batteries, for use in later congested periods. Thus, from the provider's perspective, the user can effectively shift his or her energy consumption to less congested times, even though from the user's perspective nothing has been shifted.

6 Economics of SDP

Given the wide variety of SDP pricing algorithms presented in Section 5, a thorough discussion of the theory behind each one is impractical for a book chapter. In this section, we instead select four representative scenarios to illustrate some of the key economic principles often used in formulating different types of pricing algorithms. We first consider static pricing on a single link, and then consider both real-time dynamic pricing and day-ahead time-dependent pricing. Readers familiar with network economics may wish to skip this section.

6.1 Usage-Based Pricing: A Single Link Example

An operator generally sets its mobile data prices so as to achieve a certain objective, e.g., maximizing profit. In this section, we review some standard economic concepts that are often used in formulating such objective
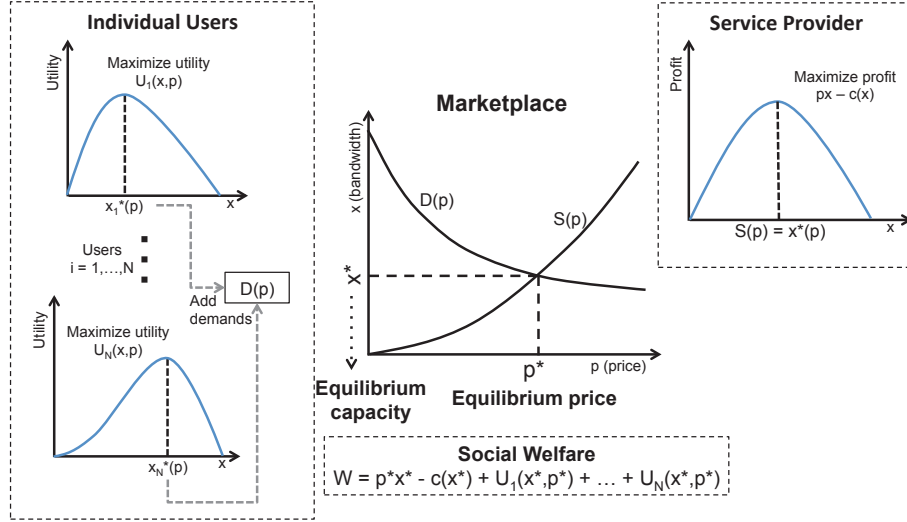
Figure 7: User-ISP interaction in a mobile data marketplace.

functions. We consider two agents: end users and ISPs.[9] For simplicity, we consider only one ISP with a given set of customers, and we suppose that the ISP wishes to build a last-mile access link in its network. The ISP wishes to determine both the *capacity to provision* on this link, as well as the *price per unit bandwidth* to charge its users on the link. This is a standard monopolist profit maximization that we discuss below. We denote the capacity with the variable x, and the price by the variable p. The ISP-user interaction is summarized in Figure 7.

We first consider users' decisions to purchase certain amounts of bandwidth on the ISP's new access link. In modeling this user behavior, we suppose that each user acts so as to maximize his or her *consumer surplus function*, denoted by $U_j(x_j, p)$ for each customer $j = 1, 2, \ldots, J$. The function U_j is the net benefit to a consumer from the utility received in purchasing x_j amount of bandwidth for a price p per unit bandwidth.[10] Thus, given a price p, if $U_j(y_j, p) > U_j(x_j, p)$, user j prefers to purchase y_j units of bandwidth, rather than x_j units. Since the ISP chooses the value of p, each user j takes the price as given and chooses the quantity of bandwidth to purchase (x_j) so as to maximize the utility $U_j(x_j, p)$. We denote this utility-maximizing quantity as $x_j^*(p)$.[11] These functions $x_j^*(p)$ are called users' *demand functions*; adding them up, we obtain the *aggregate demand function*, $D(p) = \sum_j x_j^*(p)$.

We now consider the ISP's problem of choosing a link capacity x and price p. Given a price p and assuming full utilization of the link capacity, the ISP chooses x so as to maximize its utility function. Usually, the ISP's utility is simply its *profit*, but other functions can be used. We write the ISP profit as $px - c(x)$, where px is the ISP revenue and the function $c(x)$ denotes the cost of building a link of capacity x. Given p, the ISP can then find $x^*(p)$, the optimal link capacity as a function of the price p. We use $S(p) = x^*(p)$ to denote this supply side function.

---

[9]Sponsored content and app-based pricing models may also include content providers as a separate type of agent.

[10]This function may be additively decomposed into the form $U_j(x_j, p) = V_j(x_j) - px_j$, i.e., a utility term V_j and the price paid px_j. In this scenario, $V_j(x_j)$ is often called the utility, and $U_j(x_j, p)$ the net benefit received by the user. Additively incorporating the price can also be interpreted as incorporating user budget constraints through Lagrange multipliers; more details can be found in Section 6.3.

[11]The argument p emphasizes the fact that this optimal bandwidth x_j^* depends on the price p offered by the ISP.

When the user and ISP are at a *market equilibrium*, supply equals demand: $D(p) = S(p)$. At such a price p^* satisfying this relation, each user maximizes his or her own utility by purchasing $x_j^*(p^*)$ amount of bandwidth, and the ISP maximizes its utility by providing just enough capacity $x^*(p^*) = \sum_j x_j^*(p^*)$ to support those users' demands. One often-analyzed property of this equilibrium is the *social welfare*, defined as the sum of the utility received by all users j and the ISP:

$$\sum_j U_j\left(x_j^*, p^*\right) + p^* \sum_j x_j^* - c\left(\sum_j x_j^*\right),$$

where x_j^* is understood to be evaluated at the equilibrium price p^*. This social welfare can be divided into two portions: the *user surplus*, or the sum of user utilities, and the *ISP surplus*, or the utility (here, profit) obtained by the ISP. Depending on the utility functions U_j and the cost function c, the total social welfare may change, and the users and ISP may receive different portions of the overall social welfare.

Before moving on, we pause to discuss some of the more common extensions of the simple problem above. One is to introduce *budget constraints* on each user's utility maximization problem: the user may not want to spend more than a certain amount B_j, in which case each user j maximizes the utility $U_j(x_j, p)$ subject to the constraint $px_j \le B_j$. We may also consider a situation in which users impose *externalities* on each other, i.e., a given user j's utility is affected by the capacity allocated to other users $i \neq j$. For instance, there may be a positive externality in which user j's utility increases as other users send traffic over the link in order to interact with user j. On the other hand, one could also observe negative externalities, in which congestion from other users' traffic diminishes a particular user's utility, e.g., by increasing delay.

When solving for the market equilibrium above, we initially took the price p as fixed for the end users and ISP, and then found the equilibrium market price p^*. In fact, one can obtain this equilibrium price by only examining the optimal behavior of end users and ISPs, i.e., without explicitly considering market equilibrium. Suppose that the ISP, knowing users' demand functions $x_j^*(p)$, calculates its revenue as a function of price to be $p \sum x_j^*(p)$ (the price, multiplied by the user demand as a function of price). The ISP can then choose both p and x so as to maximize its profit $p \sum_j x_j^*(p) - c(x)$, subject to the constraint that the link capacity be able to accommodate users' total demand $\sum_j x_j^*$, i.e., that $x \ge \sum_j x_j^*$. It is easy to see that (assuming the cost $c(x)$ is increasing in the capacity x), at the optimum, $x = \sum_j x_j^*$. The ISP then chooses the optimal price p so as to maximize $p \sum_j x_j^*(p) - c\left(\sum_j x_j^*(p)\right)$. One can show that the resulting optimal price, which we will call $\bar{p}$, is the same equilibrium price p^* obtained above: at $\bar{p}$, each user j demands $x_j^*(\bar{p})$, and the ISP chooses its optimal capacity $x^*(\bar{p})$. This is exactly the point at which the supply and demand curves intersect, i.e., p^*.

The above reasoning, in which an ISP chooses a price to offer subject to users' behavior as a function of the price chosen, is a simple example of a *game* between users and ISPs. In such a game, several players interact with each other, and each player acts to maximize his or her own utility, which may be influenced by other players' decisions. For instance, in this scenario, users interact with the ISP by utilizing the access link in its network and paying some price. Their decisions on how much capacity to utilize (i.e., choosing x_j^*) are influenced by the ISP's choice of the price p. This interpretation of the single-link example leads us to next consider some basic principles of *game theory* in relation to SDP.

6.2 Incentive Compatibility: Game-Theoretic Principles

To illustrate some of the basics of game theory, we again consider the single link example above. The user-ISP interaction in such a scenario is an example of a *Stackelberg game*, in which one player, the "leader,"

makes a decision (e.g., the ISP sets a unit price p for link capacity) and the remaining players, or "followers," then make their own decisions based on the leader's actions. In SDP, this framework reflects the need to consider users' and ISPs' optimal actions when choosing pricing policies that incentivize particular types of resource allocations. In this example, users choose their demands $x_j^*(p)$, given the ISP's price p. Stackelberg games, which often arise in user-ISP interactions, may be solved using *backwards induction*: first, one computes the followers' actions as a function of the leader's decision (in our example, we compute the functions $x_j^*(p)$). The followers' actions are sometimes called a *best response* to the leader. The leader then takes these actions into account and makes his or her own decision (given that users' demands are $x_j^*(p)$, the ISP chooses the optimal price p). This decision is then the best response to the followers.

The backwards induction process leads to a *subgame perfect equilibrium* in the Stackelberg game: at this equilibrium, each player is maximizing his or her own utility, and no player has an incentive to change his or her behavior. To formalize this definition, we will need to first explain the concept of a *Nash equilibrium*. Consider a general game with n users, each of whom can take an action, e.g., by choosing the value of a variable y_j; $j = 1, 2, \ldots, n$; and suppose that each user j's utility V_j is a function of all of the y_j variables, i.e., $V_j = V_j(y_1, y_2, \ldots, y_n)$. Then a set of actions $z_1, \ldots, z_n$ is a Nash equilibrium if $V_j(z_1, \ldots, z_j, \ldots, z_n) \geq V_j(z_1, \ldots, y_j, \ldots, z_n)$ for any $y_j \neq z_j$. In other words, assuming that all the other players take actions z_i, player j's action z_j optimizes its utility V_j.

We may generalize the concept of a Nash equilibrium to a Stackelberg game's subgame-perfect equilibrium by considering *subgames* of the Stackelberg game. We do not give the general definition of a subgame here, but it may be understood by envisioning the Stackelberg game as a dynamic game with different levels defined by the time of decision: on the first level, users make their decisions, and on the second, ISPs make their decisions. A subgame encompasses a group of players who mutually interact, but do not directly interact with other players at their level. In our scenario, a subgame would be a subset of users and the ISP. A subgame-perfect equilibrium of the full Stackelberg game is then a set of actions that comprise a Nash equilibrium in each subgame of the full game. It can be shown that any equilibrium found from backwards induction is a subgame-perfect equilibrium; one can easily check that this is the case in our example scenario. Nash and subgame-perfect equilibria are considered stable in that once they have been achieved, no user has an incentive to change their behavior. (Unfortunately, one cannot in general guarantee that such an equilibrium will be achieved in the first place, and a game may have multiple Nash equilibria.)

Another type of game that often arises in SDP is that of competing service providers. For instance, we may have an *oligopoly* of a few companies who dominate the market for mobile data, e.g., AT&T and Verizon in the United States are the dominant market players. Each of these companies then competes for customers (i.e., market share) and revenue with the others. This competition defines their interactions, and each company can try to make strategic decisions that optimize its market share. Given a mathematical model of the companies' actions, one can then try to study the corresponding game, e.g., by computing possible Nash equilibria.

While certainly useful for explicit pricing problems like that considered above, game theory can also be applied to more general resource allocation problems, just as SDP allows ISPs to incentivize users to consume data so as to realize particular resource allocations. To illustrate these uses, we again consider the single link example, but we now suppose that the link's capacity is fixed and that the ISP wishes to allocate this fixed amount of capacity x among its n users.

If users selfishly maximize their individual utilities (i.e., choose demands $x_j^*(p)$), then the ISP can set a virtual price p to force an allocation in which $\sum_j x_j^*(p) = x$, i.e., all of the available capacity is utilized, and each user maximizes his or her utility. This price serves as a signal through which the ISP can control users' demands. However, such an allocation may be unfair: very price-sensitive users may be able to afford significantly less capacity than others. Since revenue is no longer involved, the ISP can afford to care about

other objectives like fairness. Indeed, a vast literature exists on just such a problem; we will not go into fairness theory here, but we will present one approach inspired by game theory.

In the Stackelberg game discussed above, users did not cooperate: each user maximized only his or her own utility, subject to the ISP's offered price. Yet if users do cooperate, they may reach a better decision. We can study this problem by first defining individual users' utilities $U_j(y_j)$; given a capacity amount y_j, each user j derives utility $U_j(y_j)$. For instance, users could jointly choose their demands y_j, subject to the capacity constraint $\sum_j y_j \leq x$, so as to maximize an overall utility function $U(U_1(y_1), \ldots, U_n(y_n))$. Depending on the choice of U, of course, one would obtain different allocations y_j^*. We use y_j^* to denote the y_j that jointly maximize U. Nash proposed that the y_j^* satisfy the following four axioms:

1. *Invariant to affine transformations:* For each user j, define the utility function $V_j(y_j) = \alpha_j U_j(y_j) + \beta_j$ for some constants $\alpha_j > 0, \beta_j$. Then the allocation $\{z_j^*\}$ maximizing $U(V_1(z_1), \ldots, V_n(z_j))$ satisfies $V_j(z_j^*) = \alpha_j U_j(y_j^*) + \beta_j$ for each user j, where the allocation $\{y_j^*\}$ maximizes $U(U_1(y_1), \ldots, U_n(y_n))$. An affine transformation of the utility functions U_j does not change the utility received at the optimal allocation.

2. *Pareto-optimality:* An allocation $\{y_1^*, \ldots, y_n^*\}$ is Pareto-optimal if for any user j, any feasible allocation $\{z_1, \ldots, z_n\}$ with $U_j(z_j) > U_j(y_j^*)$ satisfies $U_i(z_i) < U_i(y_i^*)$ for some user i. In other words, no user can be made better off without making another worse off.

3. *Independence of irrelevant alternatives:* Suppose that $U(U_1(y_1), \ldots, U_n(y_n)) > U(U_1(z_1), \ldots, U_n(z_n))$ for two feasible allocations $\{y_j\}$ and $\{z_j\}$. Then if the problem constraints are relaxed to allow new feasible allocations, we still have $U(U_1(y_1), \ldots, U_n(y_n)) > U(U_1(z_1), \ldots, U_n(z_n))$.

4. *Symmetry:* Suppose that $\{y_1, \ldots, y_n\}$ and $\{z_1, \ldots, z_n\}$ are feasible capacity allocations with $U_{j_1}(y_{j_1}) = U_{j_2}(z_{j_2})$ for some users j_1 and j_2, $U_{j_2}(y_{j_2}) = U_{j_1}(z_{j_1})$, and $U_j(y_j) = U_j(z_j)$ for all $j \neq j_1, j_2$. Then $U(U_1(y_1), \ldots, U_n(y_n)) = U(U_1(z_1), \ldots, U_n(z_n))$. In other words, switching the order of the utilities received does not change the overall utility U.

An allocation satisfying these four axioms is said to be a *Nash bargaining solution*. One can show that if U is taken to be $\prod_j U_j(y_j)$, then the resulting y_j^* is a Nash bargaining solution. Taking the logarithm, we see that this is equivalent to maximizing $\sum_j \log(U_j(y_j))$. In other words, users choose their demands to maximize the sum of the *logarithms* of their utilities U_j. Since the logarithm is sub-linear for large $U_j(y_j)$, the optimal allocation $\{y_j^*\}$ will penalize large values of U_j relative to smaller ones, yielding a "more equal" allocation $U_j(y_j^*)$ than simply maximizing the sum of utilities $\sum_j U_j(y_j)$.

6.3 Real-Time Dynamic Pricing

So far, we have focused on pricing and bandwidth allocation of a single access link. However, in reality an ISP's network does not consist of single bandwidth links: it is, in fact, a network, with multiple nodes and links between them. Data traffic between two nodes, e.g., between a user and a content provider, flows across a subset of the network links. Since different links may experience different types of congestion at different times, an ISP may want to adjust the prices charged based on how much congestion is experienced by a particular user at a given time. It is this philosophy that lies behind dynamic pricing for congestion control.

To illustrate the basic concepts of congestion control, we consider a relatively simple example given in Kelly et al.'s seminal paper on the subject [68]. Consider a set of nodes, indexed by $n = 1, 2, \ldots, N$, and a set of links indexed by $l = 1, 2, \ldots, L$ that connect different nodes together. We suppose that each node

n wishes to communicate with another node, and we use R_n to denote the subset of links traversed by node n's traffic.[12] The ISP's goal is then to set a traffic rate x_n for each node n, such that 1) the total amount of traffic on any link l lies below link l's capacity c_l, and 2) all users are as satisfied as possible. To accomplish this, each link can set a unit *congestion price* for traffic on the link. By prescribing the evolution of these prices in time, the ISP can satisfy its two objectives in a distributed manner.

We first define a *routing matrix* to summarize the routes taken by different nodes' traffic over the network: let R be an $L \times N$ matrix, and set $R_{ln} = 1$ if $l \in R_n$, i.e., node n's traffic travels over link l, and $R_{ln} = 0$ otherwise. If we concatenate nodes' traffic rates x_n into an $N \times 1$ vector $\vec{x}$, we see that $\vec{y} = R\vec{x}$ yields a vector of length L. Each entry y_l of $\vec{y}$ equals the total volume of traffic on link l. Letting $\vec{c}$ be an $L \times 1$ vector of the capacities of each link l, we then have the capacity constraint $R\vec{x} \leq \vec{c}$: the total amount of traffic on each link l cannot exceed the link's capacity.[13] This constraint ensures that the ISP's first objective is satisfied.

The ISP's second objective is that each user be "as satisfied as possible." We define satisfaction by defining utility functions $U_n(x_n)$ for each node n; the ISP is then assumed to assign source rates x_n so as to maximize the total sum of utilities, $\sum_n U_n(x_n)$, subject to the constraint $R\vec{x} \leq \vec{c}$. To solve this problem, we next make the assumption that each utility function U_n is concave. Such an assumption is consistent with the economic principle of *diminishing marginal utility*, i.e., that the extra utility received from an additional unit of bandwidth decreases as the user receives more and more bandwidth. Under this assumption, the ISP's objective function $\sum_n U_n(x_n)$ is concave. Since the constraints $R\vec{x} \leq \vec{c}$ are linear, the overall optimization problem is a convex optimization.

We now follow standard optimization theory and introduce a $L \times 1$ vector of Lagrange multipliers $\vec{p}$, with each p_l corresponding to link l's capacity constraint in the component-wise inequality $R\vec{x} \leq \vec{c}$. These multipliers $\vec{p}$ will eventually become the congestion prices set by the links l. The ISP's optimization problem is then equivalent to solving

$$\min_{\vec{p} \geq 0} \max_{\vec{x}} \left(\sum_{n=1}^{N} U_n(x_n) + \vec{p}^T \left(\vec{c} - R\vec{x} \right) \right). \tag{1}$$

Solving (1) centrally can be done relatively easily; it is not hard to see that we have the solution

$$x_n^* = U_n'^{-1}(q_n), \quad p_l^* = \begin{cases} 0 & \text{if } c_l - y_l > 0 \\ > 0 & \text{if } c_l - y_l = 0 \end{cases},$$

where we define $\vec{y} = R\vec{x}$ and the nth entry of the $N \times 1$ vector $\vec{q} = R^T \vec{p}$ equals the sum of the congestion prices for the links traversed by node n's prices; q_n thus represents the total price paid by user n. However, our goal is to develop a *distributed* solution, in which nodes adjust their rates x_n and links adjust their prices p_l so as to converge to the optimal solution. The ISP can drive these dynamics with the link prices–i.e., links change their prices p_l, and nodes respond by adjusting their rates x_n according to the solution above. An example of a price-driven algorithm is [79]

$$p_l(t+1) = [p_l(t) - \gamma(y_l(t) - c_l)]^+, \quad x_n(t+1) = U_n'^{-1}(q_n(t+1)), \tag{2}$$

where the argument t denotes the value of a variable at time t, and we consider discretized times $t = 1, 2, \ldots$ The constant $\gamma > 0$ is a stepsize parameter, and the $+$ superscript $[\alpha]^+$ denotes the maximum of a quantity α

---

[12]Choosing the optimal routes for each node n is a non-trivial problem in itself; for simplicity, we assume here that all of the routes R_n are fixed.

[13]This inequality is to be interpreted component-wise, i.e., each component of the left-hand and right-hand side vectors should satisfy it.

and $0.$[14] We note that each link l evolves its price p_l using only the traffic on that link $y_l(t)$ and its capacity c_l, both of which are known without communicating with other nodes or links. Similarly, each node n adjusts its rate x_n based only its price q_n, a quantity that can be carried with node n's traffic and is known without node n communicating with other nodes or links. One can show that these dynamics converge to the optimal prices and rates if the stepsize γ is sufficiently small. Moreover, as user utilities U_j or link capacities c_l change over time, following the dynamics (2) will reposition the rates and prices to their new optimal values. Thus, real-time dynamic pricing can adapt to network congestion levels and keep ISP traffic from exceeding the network capacity. We explore some variations on this dynamic pricing model in the next section.

6.4 Dynamic Day-ahead Time-Dependent Pricing

One limitation of real-time dynamic pricing is that it requires users to respond to price changes by adjusting their demands in real time. Yet to consciously involve users in adapting their demand to price changes, a longer timescale is preferable.[15] In this section, we again consider the case of a single access link for longer timescale time-dependent pricing. Our goal is to develop an alternative model that is simple enough to be practical as a real pricing plan, yet also helps to alleviate congestion on ISP networks. We aim to offer prices to users in advance, so that users can have more certainty in planning changes to their usage behavior. Indeed, previous studies showed that prices which change in real time can discourage changes in usage, as users are not sure whether the price will decrease in the future [117]. Additionally, in computing the optimized prices (discounts) that maximize network provider's profit, the model needs to consider (a) the cost incurred from offering price discounts, (b) the savings from shifting some traffic from peak to off-peak hours, and (c) the increase in baseline demand in discounted periods due to potential "sales day" effect.

We divide one day into T time periods–e.g., $T = 24$ periods of one hour each–and suppose that the ISP can offer a different price at each time $t = 1, 2, \ldots, T$. We suppose that the ISP has an existing link of capacity C, and that it may accommodate additional demand at a marginal rate γ. This additional cost may represent the increased cost of handling customer complaints once usage exceeds capacity C, or an additional investment cost necessary for increasing the network capacity. We let X_t, $t = 1, 2, \ldots, T$, denote user demand at each time t. Then the ISP's cost of accommodating these demands is

$$\sum_{t=1}^{T} \gamma \max\left(X_t - C, 0\right). \tag{3}$$

Given that accommodating demand in the peak periods is more expensive than demand during less-congested periods in which $X_t < C$, the ISP has an incentive to offer lower prices in less-congested periods, thus inducing users to shift some demand into those periods. This is the core idea of time-dependent pricing: by offering lower prices during less congested times, the ISP can even out its demand over the day, resulting in lower capacity costs. Our treatment here follows that in [48].

To formalize this argument, we next need to develop a mathematical model for users' shifting of traffic in response to the prices offered. We let p_t denote the price offered by the ISP at each time t. We suppose that the ISP can offer a maximum price that is normalized to 1, e.g., if the ISP currently offers a usage-based price of 1 without time-dependent pricing. We can then define the discount offered at each time t as $d_t = 1 - p_t$. Users' willingness to shift some traffic from one period t_1 to another period t_2 then depends

---

[14]This modification ensures that the prices are nonnegative.

[15]There is of course a tradeoff in choosing the timescale: while longer timescales are simpler for users to understand, shorter timescales allow prices to more accurately reflect congestion.

on the additional discount offered in period t_2, i.e., $d_{t_2} - d_{t_1}$. If $d_{t_2} >> d_{t_1}$, then the user will be able to save more money and thus will be more willing to shift his or her traffic.[16] However, users' willingness to shift their usage does not just depend on the discounts offered: it also depends on the time shifted $t_2 - t_1$. For instance, a user may be very willing to shift some traffic by half an hour, but much less willing to shift his or her usage by more than an hour, even with the same discounts.[17]

The discounts offered and time shifted are not the only factors determining how willing users are to shift their data traffic: the *type of traffic* also matters. Software downloads, for instance, may be readily delayed by several hours, but users are often much less willing to delay urgent apps like email retrieval. For simplicity, in the following discussion we assume only one type of traffic, but our model can be easily extended to multiple traffic types by adding up the amount of traffic shifted for each traffic class. Considering multiple traffic classes is necessary for ISPs to accurately take user behavior into account: since each user will have some traffic that can be delayed more than other traffic, the fact that we consider the simultaneous behavior of multiple users need not mitigate the need for multiple traffic classes.[18] We use $w(d, t)$ to denote users' probability (i.e., willingness) to delay their traffic by an amount of time t, in exchange for an additional discount d. We suppose that w is increasing in the discount d and decreasing in time t, and that the value of w lies between 0 and 1, so that it may be interpreted as a probability. Many functions w meet these criteria, e.g., the functions

$$w(d, t) = \frac{\max(d, 0)}{\Gamma(t+1)^\beta},$$

where Γ is a normalizing factor and $\beta \geq 0$ is a model parameter that controls the rate at which willingness to shift decreases with the time shifted t. For larger values of β, the willingness to shift decays faster with time, denoting increased impatience.

The general idea behind our model is to use the $w(d, t)$ functions to calculate how much traffic is shifted to different times of the day, relative to a baseline traffic pattern without time-dependent discounts. Our model thus attempts to reflect users' thought processes in looking at a set of future prices and deciding whether to delay their traffic or not. While a more traditional model would use utility functions (cf. Section 6.1) to describe users' behavior, user utility functions are difficult to derive from user behavior, since "utility" attempts to quantify the relatively vague idea of "user benefit." The probability of shifting traffic, on the other hand, can be calculated directly from observing the amount of traffic shifted in response to discounts offered to users.

Given the functions w, the expected amount of traffic shifted from time t_1 to time t_2 is

$$w\left(d_{t_2} - d_{t_1}, |t_2 - t_1|_T\right),$$

where $|t_2 - t_1|_T$ denotes the number of periods between time t_2 and period t_1, modulo the number of periods T (e.g., if $t_2 < t_1$, then $|t_2 - t_1|_T$ is the number of periods between period t_1 and period t_2 on the next day). Given an initial amount of traffic Y_t in each period t, we calculate that $Y_{t_1} w\left(d_{t_2} - d_{t_1}, |t_2 - t_1|_T\right)$ amount of traffic is shifted from time t_1 to time t_2. The ISP then loses $\left(d_{t_2} - d_{t_1}\right) Y_{t_1} w\left(d_{t_2} - d_{t_1}, |t_2 - t_1|_T\right)$ amount of revenue due to the traffic shifted from time t_1 to t_2. Some additional revenue is lost from the

---

[16]One could also consider functions such as the ratio of discounts in the two periods, but $d_{t_2} - d_{t_1}$ has a reasonable interpretation as the amount of money a user can save by shifting traffic.

[17]In this section, we suppose that users only delay their traffic, i.e., traffic is shifted only to later, not earlier, times. In practice both types of shifting can occur, but we make the reasonable assumption that it is more natural for users to delay some traffic than it is to anticipate future usage.

[18]The amount of traffic corresponding to each traffic class can be treated as a model parameter; along with the other model parameters, it can be estimated from observed aggregate usage data.

unshifted traffic in each period, for a total revenue loss of

$$\sum_{t=1}^{T} \left[Y_t d_t + \sum_{s \neq t} Y_s \left(d_t - d_s \right) w \left(d_t - d_s, |t - s|_T \right) \right].$$ (4)

In addition to this revenue loss, offering discounts at some times may induce users to increase their overall usage during those time periods. This increase is independent of any usage shifted from more expensive times and constitutes a psychological "sales day effect," which was observed during time-dependent pricing trials [48, 112]. It reflects the qualitative insight that user demand increases as the price charged decreases: users see that they are saving money, relative to usage during high-price periods, and are thus encouraged to spend and use more. The larger the discount offered d_t, the larger this increase will be. We can model this increase with a power law: given an initial amount of traffic Y_t in period t, the amount of traffic after a discount d_t is offered (neglecting any traffic shifted to other periods) is $Y_t \left(1 + d_t \right)^{\alpha}$ for some positive model parameter α. We then have the desirable property that demand does not change if no discount is offered ($d_t = 0$); if $\alpha = 0$, the total demand does not depend on the discount at all. The ISP thus earns additional revenue

$$Y_t \left(\left(1 + d_t \right)^{\alpha} - 1 \right) \left(1 - d_t \right)$$ (5)

due to this additional demand in period t. We can then add the ISP's revenue loss from discounts offered (4), less the revenue gain (5) from additional traffic, to the ISP's cost of capacity (3) to obtain the objective function

$$\sum_{t=1}^{T} \left[Y_t d_t - Y_t \left(\left(1 + d_t \right)^{\alpha} - 1 \right) \left(1 - d_t \right) + \sum_{s \neq t} Y_s \left(d_t - d_s \right) w \left(d_t - d_s, |t - s|_T \right) + \gamma \max \left(X_t - C, 0 \right) \right],$$

where the total traffic at each time t is

$$X_t = Y_t \left(1 + d_t \right)^{\alpha} + \sum_{s \neq t} Y_s w \left(d_t - d_s, |t - s|_T \right) - \sum_{s \neq t} Y_t w \left(d_s - d_t, |s - t|_T \right).$$

The first term represents the increase in traffic due to the discount, while the second is the amount of traffic deferred into period t, and the third term the amount of traffic deferred out of period t. The ISP then chooses the discounts d_t (equivalently, the prices $p_t = 1 - d_t$) to minimize its objective function. Under reasonable conditions on α, γ, and w, this is a convex optimization problem, which may be rapidly solved.

By solving this optimization problem, the ISP can obtain a set of prices for one day; it can then offer day-ahead pricing by running an optimization in each period that determines the optimal discount to offer one day from the current time. Moreover, the ISP can observe the traffic consumed in each period once these discounts are offered. Comparing this data to the traffic observed without discounts, the ISP can estimate the parameters of its user behavior models (i.e., α and w above) from the observed changes in usage, given the prices offered. These estimates can be periodically updated and used to calculate new prices, completing a feedback loop (cf. Figure 5) between users and the ISP [48].

7 Psychological Aspects of SDP

Arguably, the foremost factor in realizing a successful data plan is its adoptability by end users. But this depends not only on good economic models and system capabilities, but also on understanding consumer

behavior and psychological aspects, and in particular, how client-side interfaces need to be designed to make them effective. These concerns require the networking researchers to interact and work closely with the ongoing efforts in the human-computer interaction (HCI) community. This section covers some of the basic themes emerging from the HCI research on psychology of Internet users.

As early as 2005, HCI researchers introduced networking, particularly user behavior considerations in home networks, as an important aspect of HCI studies [46]. In the context of broadband networks, few HCI works have developed human-facing systems to manage network usage. The *Eden* system [136] modified a home router to provide users with an intuitive interface for managing their "home network experience." The main focus of such home-networking tools has been on designing GUIs to help users understand the physical location of different devices in their home and be able to perform basic administrative functionalities like perform membership management, access control, network monitoring, etc. In a similar vein, the *Homework project* [87] modifies the handling of protocols and services at the home router to monitor data usage, prioritize different devices, and monitor other users data consumption (usually in the context of parental control) in order to reflect the interactive needs of the home. Other studies have focused on understanding the impact of monitoring and sharing bandwidth speed in a wired home network [21]. The authors carried out a field trial with 10 households using a visual network probe designed to help educate and empower consumers by making broadband speeds and sources of slow-downs more visible. In a separate work, Chetty et al. [20] also carried out a field trial of their *Home Watcher* bandwidth management tool to study the effect of viewing others bandwidth usage on social dynamics in the household. They showed that when resource contention amongst different household members is made visible, users have better understanding of bandwidth their usage and allocation, revealing new dimensions in household politics. Yet while these works addressed several crucial elements of network intervention (e.g., throttling, capping, parental control) and its related visualization tools, they have been limited to either modifying the network stack within the OS [20] or deploying a custom-built access point [87]. In contrast, the need today is to focus on understanding the following issues: (a) how economic incentives (i.e., pricing) can impact and modify mobile user behavior, (b) how researchers can carry out field trials (almost seamlessly from an end-user perspective) by interposing between the ISP and its real customers (i.e., participants).

In the context of mobile networks, [104] considers the need for mechanisms to help users track their spending on mobile data. This line of research on user interface designs needs more attention to realize more dynamic pricing plans and study user response to such plans. However, realizing such data plans requires assumptions of rational behavior to be realized in practice, i.e., that people perceive the pricing signals and change their behavior. The Berkeley INDEX project [128] in the 1990s for wired Internet suggested that users should be able to view prices and select desired QoS levels (i.e., bandwidth speed), with similar results reported by the M3I project [9]. Similar field trials on the HCI aspects of time-dependent pricing for mobile data have been reported in [112], and the main themes emerging from these studies will be discussed below.

Recently, there have been calls to investigate HCI aspects of time-dependent pricing in the context of ecological sustainability and energy consumption [98]. Even without economic incentives, good UI designs of energy monitors have been shown to be effective in changing consumer behavior [40, 98]. These investigations have ranged from a large-scale media art installation visualizing energy consumption in an office building, to power strips that change color to show the energy used by individual electrical sockets [54, 57]. One of the usual tradeoffs in visualizing energy usage is the use of pictorial versus numerical usage amounts. For instance, the iPhone application *WattBot* allows users to monitor their home energy usage, with colors indicating usage amounts [97]. While the colors enabled users to quickly grasp their qualitative energy usage, users also wanted to track their evolving usage behavior by viewing their usage history [22]. In addition to the "manner" of presenting usage data, users expressed concern over the "convenience" of checking their usage. For instance, researchers testing a desktop widget that showed computer energy efficiency found that

users appreciated the inconspicuous, easy-access nature of the widget [70]. Many of these design insights were incorporated in the client-side UI design of the TDP trial for mobile users reported in [112]. Another key consideration in designing such mobile applications for smarter data plans is that the GUIs need to be account for the form-factor, presentation, and convenience of use on mobile devices and platforms. Incorporating features like parental control and usage history have been found to be desirable to users, even in the TDP paradigm [101, 112].

The main themes emerging from these research on HCI aspects of networking are that:

1. Consumers are very concerned about the increasing cost of data plans but are not fully aware of their monthly usage. Therefore mechanisms that help them to better monitor and control their own usage and allocate bandwidth among household members (or in shared data plans for mobile) are perceived to be useful by users. Moreover, given the right economic incentives, many are willing to wait for some form of a feedback signal from the network on the congestion levels and change their behavior, e.g., choose higher QoS service for critical applications in smart market scenarios [9], or wait for discounted periods for non-critical applications in case of time-dependent data plans [112]. This is therefore a promising direction in the evolution of Internet pricing, and follows similar trends in other markets like electricity and transport networks.

2. The key factor in enabling smart data plans is designing the user interfaces that are effective and understandable to users. Previous research has shown that well-designed user-side applications can not only help users make decisions on deferring usage, but also can become a tool that helps them to self-educate, monitor, and control their usage and spending. Among the features found to be effective as means of communicating useful information to users were color-coding[19] of price/discounts, information indicators (e.g., current price) on the home screen of devices, usage history, recommendations on usage deferrals, etc.

3. Past research also revealed interesting insights on the user psychology of controlling their usage. First, trial participants viewed parental control at the granularity of apps as being useful in managing their usage. Second, they showed a desire to control their usage manually instead of delegating control to an autopilot mode. When coupled with the desire for parental control, we find users want to take charge of their consumption behaviors, for themselves and their families, in ways that require transparency and flexibility. This tradeoff between users' desire for transparency and control will be an important design considerations for enabling smarter data plans.

4. Consumers prefer a certain degree of certainty in the prices, which explains the popularity of flat-rate plans. Therefore for dynamic and smarter data plans to succeed, the time granularity of changes in prices have to be carefully accounted for. Users want to know and have certainty about these prices/discounts in advance, and therefore naive dynamic pricing plans may not work. In particular, real-time pricing that provides price signals based on current congestion levels may demand a higher degree of user engagement (or automation) than users are willing to adopt. Plans that have been explored in energy markets, such as "day-ahead" pricing [48, 112], therefore may be more acceptable to users.

5. As both networking and HCI address increasingly complex socio-technical ecosystems, e.g., mobile Internet, cloud computing, smart grids, etc., incorporating economic analysis as a part of user behavior studies is important. There is also a need to understand how researchers can create new frameworks

---

[19]While color-coding has been used and appeared effective in several scenarios, care must be taken to provide a secondary signal for persons suffering from color-blindness.

for realistic experiments and field trials on the HCI aspects of network economics. Various research ideas have taken different systems approaches, from modifying the network stack within the OS [20] to deploying custom-built access points [87], to the TUBE project [48] which developed a system for researchers to act as a resale ISP by interposing between ISPs and their customers, and provided these two sides with client-side mobile applications and ISP-side incentive computation modules.

8 Engineering Aspects of SDP

Following the works of congestion pricing by Kelly, Gibbens, and others [42, 66–68], researchers have focused on developing the basic idea of pricing bandwidth resources according to congestion marks (i.e., price "pollution" instead of just volume). Congestion marks on packets can be used as a mechanism to indicate to the end-users or end-systems (or "agents") that they may need to pay more to avail a certain level of QoS when the network gets congested. A team of researchers, led by Briscoe,[20] today are developing these ideas on the accounting architecture, feedback aspects of congestion signaling, incentive compliance, security etc. in conjunction with working groups like WG CONEX in the IETF[21] that focus on implementation aspects. One such EU collaborative project, the *M3I* (Market Managed Multi-service Internet) has proposed an architecture for market management of the Internet's QoS and demonstrated new business models that it can enable. The core of the QoS problem tackled by the M3I project is to solve the fast control problem to avoid QoS degradation during short-term congestion. In doing so, the M3I architecture requires network providers to deploy ECN (Explicit Congestion Notification) on all routers so that the congestion experienced field in the IP packet header can be set with a probability related to the current network load, allowing prices to adapt to the congestion conditions in the network. The M3I solution can be realized in different ways and in different scenarios, which have been extensively discussed in [9]. One solution uses these pricing feedback signals to encourage self-admission control. In effect this is real time pricing (RTP) version of time-of-day pricing. A second solution synthesizes admission control at the network edge from a dynamically priced wholesale service.[22] The M3I technology has also implemented several business models over different QoS mechanisms as discussed in [4]. In a similar vein, [133] has used game-theoretic model to explore the uplink pricing as a way to provide differential pricing to P2P and regular users in a competitive market.

Like M3I, much SDP research realizes the vision of pushing control out of the network into the hands of the end-users. However, giving users more usage control has to be introduced carefully so that the users are not overwhelmed with choices and network providers do not lose revenue. Users typically prefer to have some certainty about their monthly bills and hence as opposed to real-time dynamic pricing, some SDP researchers have looked at day-ahead time-dependent pricing. But day-ahead pricing requires network providers to be able to optimize for the future prices they are willing to offer based on past usage patterns and their prediction of user elasticity of demand for various types of traffic. On the user-side, it requires developing interfaces through which users can provide their feedback to these prices.

The key subsystems required in realizing the basic feedback-control architecture envisioned in M3I [9] and SDP are:

1. *Tariff Communication* is used to distribute pricing policy to other subsystems. The Price Communication Protocol (aka Tariff Distribution Protocol [53]) is a flexible protocol that can use different transport mechanisms like UDP multicast, HTTP and RSVP to distribute tariffs between the ISP's management systems and to customers.

---

[20]http://bobbriscoe.net/pubs.html
[21]http://datatracker.ietf.org/wg/conex/charter/
[22]http://bobbriscoe.net/projects/m3i/dissem/cc02/m31_cc02.pdf

2. Charging, Authentication, and Accounting module applies the chosen tariff plan to measured usage data for billing purposes.

3. Price Computation calculates optimal prices to offer to the user. In M3I, this price adapts to the real time network load while in SDP the prices are computed in advance to provide guarantees to the user's expenses.

4. Optimization and Prediction Modules are used by the provider to predict the future congestion levels and estimate the users' patience across different traffic classes at different types of traffic based on the history of traffic volumes and the observed changes to it in reaction to offered prices. The optimization module then computes the optimal prices (or discounts) to offer to the end-users in the SDP architecture.

5. Data Gathering/Usage Metering is used by the provider as inputs to the charging and accounting systems and for price calculation.

6. Mediation is used to aggregate gathered data and do format conversions as needed.

7. Client Scheduler is used in the SDP framework to create an "autopilot" mode of operation for the end-user which schedules the various types of application traffic based on user's specified delay elasticities and offered future prices so as to minimize user expenses.

8. Charge Reaction allows customers to react to the offered prices which gets fed back into the feedback-control loop between the ISP and its customers. In some scenarios such function is performed by a software agent on the end-user device (e.g., "autopilot" mode in SDP or the Dynamic Price Handler in M3I), while in others, as in day-ahead TDP, user interfaces need to be provided for users to react to the offered prices.

Next we discuss a case study of a particular realization of SDP –dynamic day-ahead time-dependent pricing; the model considerations for it, system architecture, HCI design aspects and results from a field trail.

9   Case Study of SDP: A Trial of Day-ahead TDP

While pricing algorithms are essential to SDP research, in practice such algorithms must be able to function within an ISP network. In this section, we discuss the design and results of a trial of Section 6.4's day-ahead time-dependent pricing (TDP) to highlight some ways in which implementability concerns can influence the development of pricing algorithms. We examine some important system and user interface design principles that were used in developing the prototype of this system, called TUBE, and finally present some trial results that illustrate how these elements can come together in practice. While some SDP trials have been conducted in the past, e.g., the Berkeley INDEX project in the 1990s [128], the design of this TUBE pilot trial illustrates the way new factors such as smartphones' computing capabilities affect SDP's feasibility.

9.1   Model Considerations

Most forms of dynamic pricing, in which prices must be determined in (near) real-time, require the prices to adapt based on users' behavior. For instance, users' perception of the prices offered may change over time, and demographically distinct user populations may react differently to the same set of prices (e.g., teenagers

versus businessmen). Offering dynamic pricing thus requires that the ISP first estimate its users' behavior and then use this to inform its choice of prices. In the case of time-dependent pricing, such estimates are particularly important. The basic philosophy of TDP is that by offering lower prices at less congested times, an ISP incentivizes users to shift some of their usage from more expensive, congested times to less congested times. Users' demand over the day is thus even-ed out, with peak usage decreasing; this decrease in peak usage then reduces ISPs' need to over-provision capacity for their peak demand. While lower prices would effectively encourage users to shift their demand, thus reducing costs, ISPs would also lose a large amount of revenue if the prices were too low. Moreover, users might shift their usage too much, and end up creating a new peak period during the discounted times.

In practice, these estimates of user behavior must take into account the available information that the ISP can collect from its users. For instance, TUBE's TDP algorithms, discussed in Section 6.4, use only *aggregate usage data*, that is, the total usage volume on the network at different times, in order to estimate user behavior and calculate the prices. This approach has the following benefits:

- *Scalability:* Since only aggregate usage is recorded and used in the algorithms [48], we can scale up the user behavior and price computations to multiple users and multiple applications. The algorithm complexity does not increase with the number of users contributing to the aggregate usage totals.

- *User privacy:* The amount of traffic that an individual consumes for different applications can be sensitive information (e.g., unusually large amounts of streaming video might reveal a movie buff). The TUBE algorithm does not consider application-specific usage, so the ISP need not receive or record such sensitive information.

- *Utility function estimation:* Utility function estimation is usually a hard problem. When temporal considerations are involved, it can potentially become even more complicated, as the utility of consuming data at any given time depends on the prices offered at all times of the day.

- *Empirical observations:* Instead of using utility functions, we can model users' willingness to shift their demand from one period to another, depending on the time elapsed between these periods and the price difference. Such usage shifts can be directly observed by comparing the amount used at different times and prices, and the model can then adapt as these observed shifts change over time.

By following these principles, we develop a scalable price calculation and user behavior estimation algorithm (see Section 6.4 and [48] for details) that can be feasibly deployed in a real system.

9.2 System Design

A core feature of SDP, and time-dependent pricing in particular, is that it involves both end users and ISPs. Thus, the system design must have components both on ISP servers and on user devices. Figure 8 shows this division of functionality and the requisite communication channels between the user device and ISP server. In order to make the system practical, we follow three basic principles:

- *Functionality separation:* Users and ISPs have different roles in an SDP system: while users respond to the prices offered, an ISP must set the prices. TUBE utilizes individual user devices to facilitate not only displaying prices to users, but also helping them respond to the prices offered via an *autopilot mode* that automatically schedules apps to lower price periods. Since such computations need not involve the ISP, this functionality is located on users' devices.
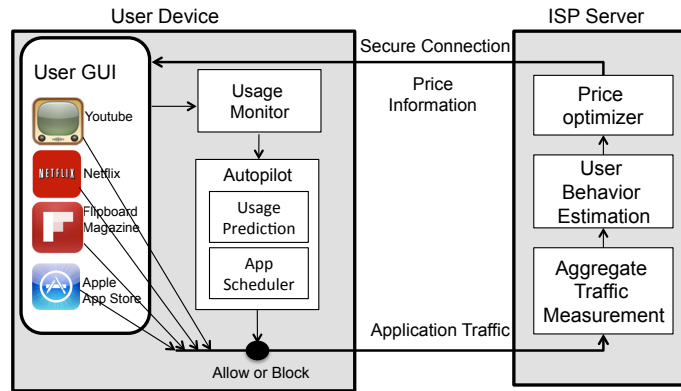
Figure 8: User-ISP functionality division and feedback communication in time-dependent pricing.

- *A feedback loop:* In order to successfully adapt prices to user behavior, the ISP needs to monitor usage in its network.[23] Thus, users must periodically send their usage to the ISP server. Similarly, the ISP must periodically update the prices displayed on users' devices as new prices are calculated. This mutual communication forms a feedback loop.

- *An open API:* An ISP's users may have many different devices with different operating systems–for instance, iOS, Android, and Windows phones and tablets. Each of these devices must therefore communicate with the ISP server. To ensure consistency across different device types, TUBE offers an open API for transmission of usage and prices between the ISP and users.

9.3 User Interfaces

SDP depends not just on pricing algorithms and system design, but ultimately on *whether users respond to the prices offered*. Thus, careful user interface design is necessary to ensure that users understand the prices being offered and to encourage them to respond accordingly. In some cases, interface design goes beyond displaying prices; users' devices can automatically adjust data usage based on the prices offered and user preferences. TUBE's user interface components can be grouped into three different categories:

- *Information displays:* Since TUBE offers day-ahead TDP, the prices for the next day should be displayed to users. But users may also find it helpful to track their spending by viewing how much usage they have consumed in the past. TUBE thus shows users both the price and usage for several past hours, so that users can understand how they usually respond to the prices offered and how this affects their spending on data. TUBE also shows the amount used for the five apps with the highest data usage, so that users can see which applications consume more data. Figure 9abc shows some sample screenshots of these features.

- *Out-of-app indicators:* Most users find checking a mobile application too onerous for keeping track of current or future prices. A more convenient way to display the prices is to show a color-coded price indicator on the device home screen to (qualitatively) signal the current price to the user, without

---

[23]Such usage monitoring also allows the ISP to calculate the amount spent by individual users.
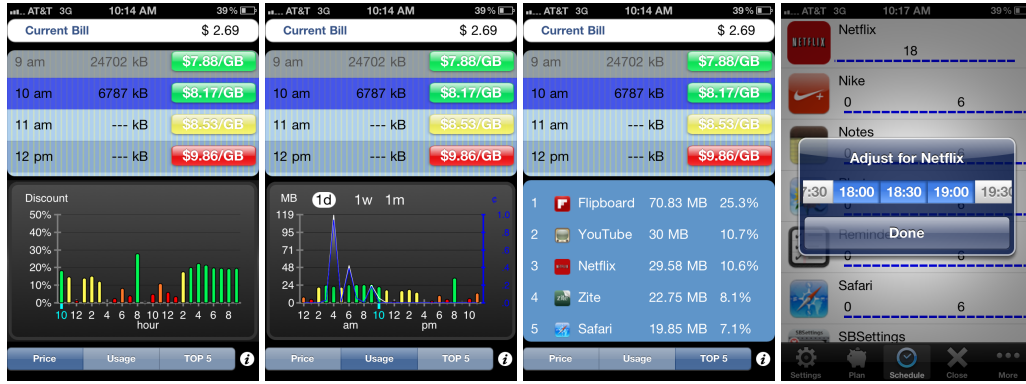
(a) Price display.   (b) Price and usage.   (c) Top 5 applications.   (d) App scheduling.

Figure 9: Screenshots of user interfaces for time-dependent pricing. Users can (a) check the prices for next 24 hours, (b) view their price and usage history, (c) identify the top 5 apps by bandwidth usage, and (d) schedule their apps at different times of the day.

requiring any special action on the user's part. Such color-coding can also be helpful for visualizing the future prices within the app, so that users can quickly decide whether to wait for lower prices.

- *Automation:* Many users prefer not to manually schedule different applications due to the complications involved in tracking future prices. TUBE thus offers an *autopilot mode* that takes into account users' delay sensitivity for different applications and monthly budget to automatically schedule some apps to lower-price times. The autopilot mode utilizes users' past spending to forecast how much the user will spend over a month. If this amount exceeds a user's monthly budget, delay-tolerant apps can be scheduled to lower-price periods; as users' spending further exceeds their budget, apps with lower delay tolerances will be scheduled to lower-price periods. However, such algorithms need to be optimized so as to be as non-intrusive as possible; in user interviews after the TUBE trial, many trial participants expressed concern over an automated algorithm controlling their data usage [112]. One way to accommodate these concerns is to allow users to override the autopilot scheduling and to configure algorithm parameters, e.g., changing the delay tolerances of different apps (Figure 9d).

9.4   Trial Results

Recently, the authors of the present work developed a prototype of the above pricing algorithms, system components, and user interfaces and trial-ed it with 50 end users [48]. We here present some results from this TUBE trial, which illustrate both the importance of user interface design and the effectiveness of optimized TDP.

An initial phase of the TUBE trial offered alternating high (10% discount) and low (40% discount) time-dependent prices in different hours. After two weeks of following the high-low-low price pattern, the prices changed, repeating the pattern of a 9% discount at midnight, followed by 28%, 30%, 28%, 9%, and 30% discounts in subsequent hours. The home screen price indicator was green for discounts over 30%, orange for 10–29% discounts, and red for discounts below 10%.

Usage in different hours with these pricing patterns can be compared to assess the effect of the indicator color and numerical discount: hours deemed as Type 1 periods offered a 10% discount in the first stage of the

254

Table 1: Period types in the color experiment.

Type	Periods	First Stage		Second Stage	
		Color	Disc.	Color	Disc.
1	2, 8, 14, 20	Orange	10%	Orange	28%
2	3, 6, …, 24	Orange	10%	Green	30%
3	5, 11, 17, 23	Orange	10%	Orange	9%

(a) Period types 1 and 3.   (b) Period types 1 and 2.
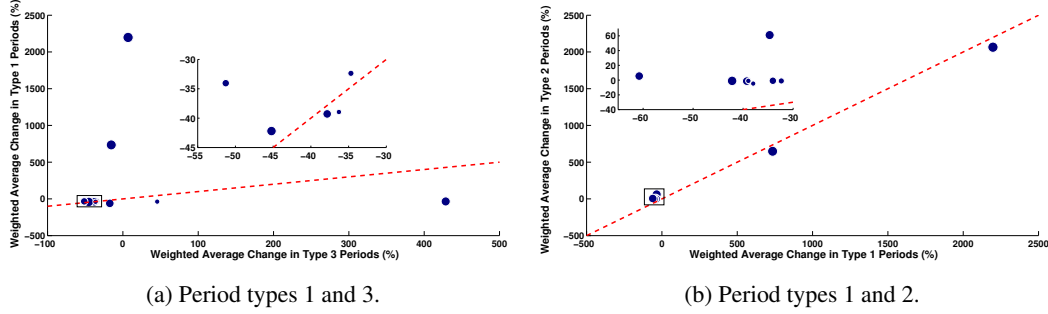
Figure 10: Average percent changes in usage for the period types in Table 1. Users' usage behavior is (a) not affected by the prices when only the numerical discounts, but not the indicator color changes. When (b) both the color and numerical discount change, users increase their usage behavior more in low-price periods.

experiment and 28% discount in the second stage, with the indicator remaining orange despite this increase in the discount. Type 2 periods offered a 10% (orange) discount in the first stage and 30% (green) discount in the second stage, while Type 3 periods offered a 10% discount in the first and 9% discount in the second stage of the experiment (the indicator was orange in both periods). Table 1 summarizes the combinations of discounts and colors used in the two stages that characterize each type of period.

To analyze the trial results, the percentage changes in usage for each type of period were computed, relative to usage without time-dependent prices. These changes showed that *users responded more to changes in the price indicator color than changes in the numeric value of the TDP discounts.* In post-trial interviews, nearly all of the trial participants indicated that they relied on the price indicator colors to know the current prices, rather than opening the TUBE app.

Figure 10 compares the usage changes observed in different period types. Each data point represents one user's average change in each period type, with the size of the data point indicating the volume of usage in the second stage of the experiment. The reference line represents equal changes in both period types considered. Figure 10a shows the average change in usage for each user in Type 1 periods versus Type 3 periods. For both period types, the color did not change, but the discount in Type 1 periods increased significantly. Thus, if users had reacted to the numerical prices, usage should increase in Type 1 and decrease in Type 3 periods: users' data points should lie above the reference line. Figure 10a shows that this is the case with only half of the users. Since the indicator color did not change, users were mostly agnostic to the numerical values of the discounts. Figure 10b plots the average change in usage in Type 2 versus Type 1 periods. The discounts in both periods increased by comparable amounts, but the indicator color changed from orange to green only in Type 2 periods. Most users' data points lie above the reference line, indicating that usage increased more (or decreased less) in Type 2 as compared to Type 1 periods. Thus, users responded to the indicator color despite the comparable numerical discounts. In fact, 80% of our participants admitted to this behavior when asked in post-trial interviews whether they paid attention to the indicator color, numerical discounts, or both.

(a) PARs.   (b) Average daily traffic.
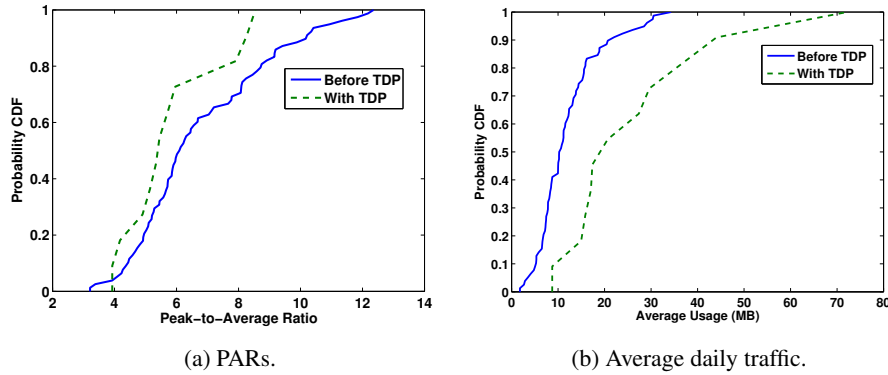
Figure 11: Usage statistics in the TIP (time-independent pricing) and optimized TDP phases of the TUBE trial. When optimized TDP is offered, (a) the ISP's peak-to-average ratio generally decreases, while (b) the average daily traffic per user increases.

The final stages of the trial offered optimized time-dependent prices, with initial user behavior estimates based on the usage observed in previous stages of the trial with non-optimized prices. The reduction in peak traffic was measured by the *peak-to-average ratio* (PAR), i.e., the ratio of usage in the peak period to average per-period usage, for each day. Comparing the PARs from before and after optimized TDP reveals that *optimized time-dependent prices reduce the peak-to-average ratio* from usage before time-dependent prices were offered (time-independent pricing, or TIP). Moreover, *overall usage significantly increased* after TDP was introduced, partially because people used more in the discounted valley periods.

Figure 11a shows the distribution of daily PARs both before and after TDP was introduced. The maximum PAR decreases by 30% with TDP, and approximately 20% of the PARs before TDP are larger than the maximum PAR with TDP. Thus, TDP significantly reduced the peak-to-average ratio, flattening demand over the day. Moreover, this decrease in PAR is not due to a net loss of traffic. Figure 11b shows the average per-user daily usage observed before and after TDP. The overall volume of usage after TDP is greater than that before TDP; in fact, across all users, the average change in usage from TIP to TDP is a 130% increase. Part of this increase may be due to the time of year–TIP usage was measured from July to September, and the TDP usage in January. TDP, however, is likely a major factor: the discounts during off-peak periods allowed users to consume more data while still spending less money and decreasing the PAR. In fact, in post-trial interviews 30% of the trial participants admitted to consciously using more data in the heavily discounted periods, with one explicitly comparing the situation to shopping at a clothing sale in department stores.

10   20 Open Questions and Future Directions

Current trends and future directions in smart pricing practices aim to make proposed pricing plans economically viable. For instance, substantial research has been done on day-ahead pricing, including the development of carefully designed user interfaces to display price and usage data as shown in Figure 9. Incorporating human factors into the engineering and design phase along with economic models can provide a holistic approach in solving the challenges of network congestion.

In addition to the pricing plans proposed above, new pricing plans have recently been proposed that have been rarely studied in the academic literature. Some promising directions include the following:

(a) Home screen.   (b) Cellular usage per app.   (c) Usage locations.   (d) Shared usage.

(e) Individual usage.   (f) Usage over time.   (g) Usage distribution.   (h) Usage of devices.

Figure 12: Screenshots of the DataWiz app with shared data plans. A user can add other users or devices to her shared data plan and can keep track of the shared data usage as well as individual usage. When a user is requested to be a part of the shared data plan, a push notification is sent to the user and the user can either accept or reject this invitation.

Shared data plans: AT&T and Verizon in the United States recently introduced shared data plans, in which several devices share a common data cap. While some studies of shared data plans have been made [111], the effects of such plans on user behavior, and how such a data cap can be shared fairly and efficiently among users, remain to be studied in detail.

One practical challenge for users on shared data plans is keeping track of their usage across multiple devices–while many apps exist to track usage on individual devices; e.g., Onavo Count, My Data Manager, and DataWiz; none track the usage of multiple devices. In fact, tracking usage on multiple devices requires not only modifications to user interface design but also integration with a common server that can send usage information of other user devices to each device on a shared plan. A login and agreement process is also necessary to ensure that users can see each others' usage only once both users have given permission.

Screenshots of a prototype shared data plan app based on the DataWiz usage monitoring application are shown in Figure 12. The app lets users view their monthly, weekly, and daily usage as compared to their monthly data cap (Figure 12a), as well as per-app and per-location usage (Figures 12b and 12c). The small pie chart indicator on the top left corner of the phone allows users to quickly see their monthly usage; the chart is filled according to how much of his or her monthly cap the user has consumed. In addition to these basic features, users can add and remove other users and devices to their data plan; if a user is added to someone else's data plan, the invitation recipient receives a push notification and can accept or reject the invitation. Upon acceptance, both users may view each other's usage over time (Figures 12e and 12f) and on a pie chart (Figure 12g). Users may add multiple devices to their accounts; each user's total usage is then the combined usage from all of these devices (Figure 12h). A push notification is sent if a user is removed from a shared data plan.

Fair throttling: Instead of merely charging users more over a certain cap, ISPs may forcibly limit usage by throttling users to a limited bandwidth rate. However, researchers have still not thoroughly examined how these bandwidth limits should be set, how they should vary over time, and what their implications are in terms of fairness across different users.

Heterogeneous networks: Many ISPs are turning to supplementary networks such as WiFi and femtocells to offload traffic from congested cellular networks. While access to such networks is often free, in the future they may wish to implement more systematic access pricing to influence the adoption of such technologies and distribution of the user population across complementary networks like WiFi and 3G [59, 62].

Sponsored content: A major question in pricing is "whom to charge" for delivering traffic. In a two-sided pricing model (like in the credit card business) of the Internet, the price of connectivity is shared between content providers (CPs) and end users (EUs). ISPs become the middle man (or platform) proving the connectivity between CPs and EUs. This *clearing house* or data traffic exchange market would extend the existing 1-800 model of phone-call services in the USA, which charges the callee rather than the caller. The tradeoffs in the resulting economic benefit between CPs and EUs remains to be quantified. Intuitively, end-users' access prices are subsidized by third party sponsors (e.g., advertisers, content providers etc.) and the ISPs have an additional source of revenue. Perhaps more surprisingly, content providers may also stand to gain from two-sided pricing if subsidizing connectivity to end-users translates into a net revenue gain through a larger amount of consumption. However, the gain to content providers depends on the extent to which content-provider payment translates into end-users' subsidy, and on the demand elasticities of the consumers. The precise gains to the three

entities will therefore depend on the respective bargaining powers stemming from their contributions and price sensitivities.

Another special case of sponsored content includes *zero-rating* and *1-800 reverse billing* policies for data traffic. Under zero-rating, an ISP makes certain types of application traffic available to the users for free. This kind of policy, although contentious from a net neutrality viewpoint, is a major step in app-based pricing and has been practiced in some parts of Europe (e.g., Mobistar introduced a 'zero-rated' plan for Facebook, Twitter, and Netlog). Understanding the impact of such pricing plans on the network ecosystem and its neutrality are important active research directions in the area of network economics.

10.1 Static Pricing

Usage-based static pricing has traditionally been offered by ISPs around the world, and is in some sense the simplest and least controversial form of SDP. Yet even simple caps on monthly usage require a means to communicate those caps to users and, on the ISP side, accounting infrastructure to keep track of users' remaining quotas. Pricing plans like token bucket pricing or negotiated contracts require even more interaction with end users, leading to questions that include:

1. How can users use their quota efficiently and keep track of a monthly usage quota?

2. If users choose different QoS levels or times to receive better QoS, e.g., in Paris metro or token bucket pricing, how can they do so without much technical knowledge of what "QoS" means? How can the ISP's infrastructure keep track of users' choices and offer the appropriate QoS?

3. Without such technical knowledge, how can users negotiate contracts (e.g., cumulus pricing) with ISPs? How can ISPs enforce these contracts?

4. If the ISP offered some form of personalized (e.g., app-based) pricing, how would it measure the usage of different applications for each user? Where in the network should such measurements take place (i.e., client devices or the network core)? From a regulatory perspective, does this violate privacy or network neutrality concerns?

5. How will users share the monthly data quotas imposed by shared data plans among different devices?

10.2 Dynamic Pricing

While static pricing offers some challenges in communicating between ISPs and end users, dynamic pricing introduces even more complications as the user must be informed of changes in price. Deployment questions unique to dynamic pricing include:

6. How often should the prices change? Should they change with the network congestion, or should they change only after a fixed time interval (e.g., one hour)?

7. Should users be told the prices in advance? Will they accept or respond to prices that change in real time?

8. The answer to the previous question can be more broadly phrased as follows–how can users be appropriately informed of the changing prices (e.g., with an app on their mobile devices)? What kind of design is optimal for such an app? Going further, what mechanisms can be developed to help users adjust their behavior in response to the prices?

9. In the context of mobile data, network bottlenecks are generally highly location-dependent. Should the prices vary by location as well as by time? How will this affect users who move from one location to another?

10. How can the prices be computed efficiently? Should this computation be done online or offline? What usage monitoring must take place, and how real time does it need to be?

11. In addition to efficient usage monitoring, how can the ISP anticipate user reactions to the prices so as to set the "optimal" prices? How can these change over time? Does the measurement process adequately protect user privacy?

12. Should dynamic pricing be coupled to QoS? If so, how?

10.3 Sponsored Content

Sponsored content pricing, in which content providers and advertisers subsidize users' spending on data, has not been widely deployed, partially due to the network neutrality implications of content provider subsidies. As a relatively new type of pricing, many questions remain to be answered:

13. What is the preferred mode of "sponsoring" in sponsored content/access? Should it be based on increasing the user's data cap, monetary discounts, or improved speed (e.g., less throttling)?

14. Will content providers sponsor content on a per-transaction basis? If so, how should these transactions be metered, and how much should they charge?

15. How can ISPs measure the cost of each transaction and develop accounting systems to keep track of content providers' sponsorship?

16. Does the idea of "sponsored content" violate network neutrality? Or can it be structured in a net-neutral way, e.g., sponsoring some data usage but not specifying the application?

10.4 Fair Throttling and Heterogeneous Networks

Other solutions to network congestion that do not explicitly use SDP include fair throttling and deployment of heterogeneous networks to offload traffic. Fair throttling has not been widely deployed in practice–while many ISPs do throttle users who exceed a certain usage cap, such measures are fairly crude and do not take into account users' full profiles. More sophisticated throttling, e.g., Comcast's throttling of Netflix traffic in 2007, has been controversial. In contrast, many ISPs have begun offering WiFi hotspots, but it remains unclear how effective they are in relieving congestion on mobile networks. Thus, interesting theoretical and implementation questions remain for both these types of pricing, including the following:

17. What criteria should the ISP consider when performing "fair" throttling? Does measuring these criteria violate user privacy or network neutrality (e.g., throttling based on the usage of specific application types)?

18. Should users be directly involved in prioritizing different types of traffic? How can their preferences be incorporated into the throttling algorithm without the act of declaring such preferences becoming onerous to the user?

19. How much traffic can be offloaded to other heterogeneous networks (e.g., 4G traffic to WiFi)? How cost effective is deploying such networks as a solution to network congestion? How can ISPs estimate the monetary and spectral benefits achieved through such traffic offloading or demand shifting?

20. If ISPs were to charge for bundled access to supplementary networks like WiFi hotspots, how would such pricing plans affect users' adoption and the overall network congestion?

These 20 questions are only some of the key questions that arise in deploying SDP and can help researchers identify interesting topics for exploration. In the coming years, as the Internet evolves further, answering these questions and others that emerge will help determine how we access (and pay for) the Internet in a highly connected, data-driven world.

11 Exercises

1. *Nash Bargaining*

   Consider a single access link and suppose that its bandwidth capacity C is shared by n users. Let x_i denote the amount of bandwidth received by each user $i = 1, 2, \ldots, n$, and suppose for simplicity that each user receives utility x_i from x_i amount of bandwidth. Suppose that the ISP allocates bandwidth to users so as to maximize

$$\prod_{i=1}^{n} x_i \quad \text{s.t.} \sum_{i=1}^{n} x_i \leq C. \tag{6}$$

   Show that the resulting allocation $\{x_i^*\}$ satisfies the four Nash bargaining axioms presented in Section 6.2.

2. *Time-of-Day Smart Grid Pricing*

   Taken from M. Chiang, *Networked Life: 20 Questions and Answers*, Cambridge University Press, 2012.

   Smart grid electricity providers often set time-dependent prices for energy usage. This problem considers a simplified example with two periods, the day-time and the night-time. The provider can set different prices for the two periods, and wishes to shift some night usage to the day. The energy provider always offers the full price during the night, and offers a reward of \$$p$/kWh during the day.

   Suppose that with uniform (not time-dependent) prices, customers vacuum at night, using 0.2 kWh, and also watch TV, using 0.5 kWh, and do laundry, using 2 kWh. During the day, customers use 1 kWh. The probability of users shifting vacuum usage from the night to the day is

$$1 - \exp\left(-\frac{p}{p_V}\right), \tag{7}$$

   where $p_V = 2$, and the probability of shifting laundry to the daytime is

$$1 - \exp\left(-\frac{p}{p_L}\right), \tag{8}$$

   where $p_L = 3$. Users never shift their TV watching from the night to the day.

   Suppose that the provider has a capacity of 2 kWh during the night and 1.5 kWh during the day. The marginal cost of exceeding this capacity is \$1/kWh. Assume that energy costs nothing to produce until the capacity is exceeded.

Figure 13: Illustration of equilibrium in PMP.

(a) Compute the expected amount of vacuum and laundry energy usage (in kWh) that is shifted from the night to the day, as a function of p.

(b) Find (to the nearest cent) the reward p which maximizes the energy provider's profit.

(c) Suppose that if vacuum or laundry usage is shifted from the night to the day, it is shifted by 12 hours. Compute the expected time shifted of vacuum and laundry using $p = p^*$, the optimal reward found above.

3. *Paris Metro Pricing*

   Taken from M. Chiang, *Networked Life: 20 Questions and Answers*, Cambridge University Press, 2012.

   Consider a metro system where two kinds of services are provided: service class 1 and service class 2. Let p_1, p_2 be the one-off fees charged per user when accessing service classes 1 and 2 respectively. Suppose each user is characterized by a valuation parameter $\theta \in [0, 1]$ such that its utility of using service class i is

   $$U_\theta(i) = (V - \theta K(Q_i, C_i)) - p_i,$$

   where V is the maximum utility of accessing the service, $K(Q_i, C_i)$ measures the amount of congestion of service class i, given $Q_i \geq 0$ as the proportion of users accessing service class i (with $\sum_i Q_i = 1$), and $C_i \geq 0$ is the proportion of capacity allocated to service class i (with $\sum_i C_i = 1$).

   At the equilibrium, i.e., no user switches from his selection, $U_\theta(i)$ is merely a linear function of θ. Suppose the equilibrium is illustrated as in Figure 13.

   Let θ_1 be the θ of the user who is indifferent to joining the first service class or opting out of all the services, let θ_2 be that of the user who is indifferent to joining the first service class or the second service class, and let $F(\theta)$ be the cumulative distribution function of θ.

(a) Show that

$$Q_1 = F(\theta_1) - F(\theta_2),$$
$$Q_2 = F(\theta_2),$$
$$V - p_1 = \theta_1 K(Q_1, C_1),$$
$$p_1 - p_2 = \theta_2(K(Q_2, C_2) - K(Q_1, C_1)).$$

(b) Assume that θ is uniformly distributed, i.e., $F(\theta) = \theta$, and that the congestion function is defined as

$$K(Q, C) = \frac{Q}{C}.$$

Solve for θ_1 and θ_2 as functions of V, p_1, and p_2.

(Hint: Try $\frac{p_1 - p_2}{V - p_1}$.)

(For details, see C. K. Chau, Q. Wang, and D. M. Chiu, "On the Viability of Paris Metro Pricing for Communication and Service Networks," *Proc. IEEE INFOCOM*, 2010.)

4. *Two-Sided Pricing*

Taken from M. Chiang, *Networked Life: 20 Questions and Answers*, Cambridge University Press, 2012.

Suppose an ISP charges a content provider (CP) the usage price h_{CP} and flat price g_{CP} and charges an end user (EU) the usage price h_{EU} and flat price g_{EU}. For simplicity, we assume zero flat prices ($g_{CP} = g_{EU} = 0$). Let μ be the unit cost of provisioning capacity. The demand functions of the CP and EU, denoted as D_{CP} and D_{EU} respectively, are given as follows:

$$D_{EU}(h_{EU}) = \begin{cases} x_{EU,max}(1 - \frac{h_{EU}}{h_{EU,max}}) & \text{, if } 0 \leq h_{EU} \leq h_{EU,max} \\ 0, & \text{, if } h_{EU} > h_{EU,max} \end{cases}$$

$$D_{CP}(h_{CP}) = \begin{cases} x_{CP,max}(1 - \frac{h_{CP}}{h_{CP,max}}) & \text{, if } 0 \leq h_{CP} \leq h_{CP,max} \\ 0, & \text{, if } h_{CP} > h_{CP,max}. \end{cases}$$

The parameters are specified as follows:

$$h_{CP,max} = 2.0\mu,$$
$$h_{EU,max} = 1.5\mu,$$
$$x_{CP,max} = 1.0,$$
$$x_{EU,max} = 2.0.$$

263

The ISP maximizes its profit by solving the following maximization problem

$$
\begin{array}{ll}
\text{maximize} & (h_{CP} + h_{EU} - \mu)x \\
\text{subject to} & x \leq \min\{D_{CP}(h_{CP}), D_{EU}(h_{EU})\} \\
\text{variables} & x \geq 0, h_{CP} \geq 0, h_{EU} \geq 0.
\end{array}
\tag{12}
$$

Find the optimal $x^\star, h_{CP}^\star, h_{EU}^\star$.

5. *Monitoring Mobile Data Usage*

   Many commercial mobile applications have been developed to help users keep track of their mobile data usage. Some examples include 3GWatchdog Pro (Android), DataWiz (iOS and Android), My-DataManager (Android), and Onavo Count (Android and iOS).

   (a) Visit two or three app websites and list their features (e.g., showing usage by application, forecasting future usage, alerts when you approach your monthly data quota). Are there any significant differences between the apps? Can you identify any consistent differences between iOS and Android apps?

   (b) What visual elements are used in the app designs? For instance, do the apps use bars or pie charts to represent usage? How are these displays different on different apps?

   (c) Based on your answers to the above questions, try to design your own app for tracking mobile data usage. What screens would you implement? What features would you offer?

12 Supplementary Materials

As a part of the supplementary reading materials for the chapter, students are encouraged to refer to the following materials:

- Slides on Time-dependent Usage-based pricing engineering (TUBE) and shared data plans.

- Chapter 12 from M. Chiang's book
  (http://www.amazon.com/Networked-Life-20-Questions-Answers/dp/1107024943) [23].

- Lecture notes, slides and homework questions on SDP from ELE 381
  (http://scenic.princeton.edu/network20q/).

- Course videos on SDP (Coursera material available at https://www.coursera.org/course/friendsmoneybytes;
  videos available at http://www.youtube.com/watch?v=MMl_fZypX0w,
  http://www.youtube.com/watch?v=N2oM0ISs0nY, http://www.youtube.com/watch?v=v_uHP4SNKGo,
  http://www.youtube.com/watch?v=21KlcErIiHc) [2].

- Demo videos related to the DataMi project (http://scenic.princeton.edu/datami/) [100].

- Free DataWiz iPhone and Android app for usage monitoring
  (download links at http://scenic.princeton.edu/datawiz/) [101].

- Research papers, surveys, and white papers from SDP workshops
  (available at http://scenic.princeton.edu/SDP2012/program.html) [99].
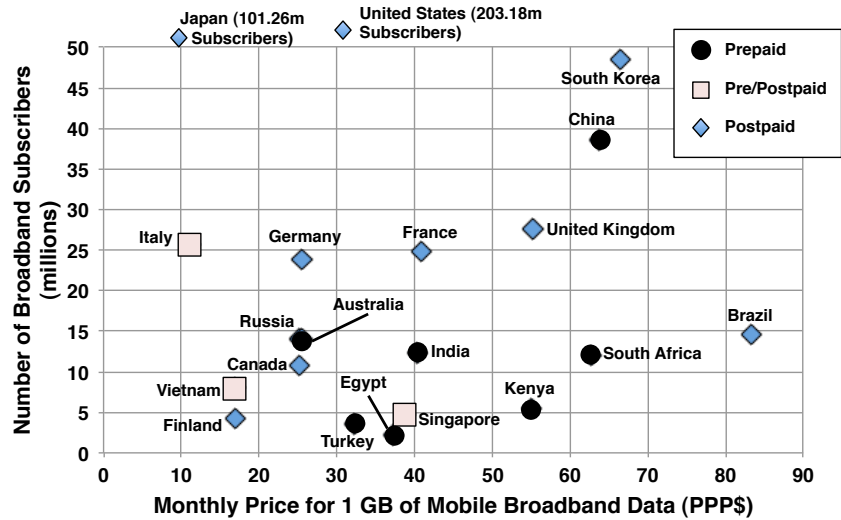
Figure 14: Prepaid and postpaid mobile broadband data plans around the world (data from 2011). Data were taken from operator websites as well as [3, 6, 24, 31, 37, 58, 60, 72, 82, 91, 93, 122–124, 126, 130, 135].

A   Data Plans around the World

In most countries, the dominant forms of mobile data pricing are still usage-based or tiered. However, the actual cost of mobile data can vary greatly from country to country, depending on such factors as the country's network infrastructure, market competition, and population density. One common distinction is between pre- and postpaid plans.[24] Castells et al. [13] found a strong correlation between the availability of prepaid plans for voice calls and adoption rates of mobile telephone subscribers, which is corroborated by Hauge et al. [51]. In this Appendix, we provide an overview of prepaid and postpaid plan adoption in different parts of the world and its relationship to consumer market demographics.

Figure 14 shows whether prepaid or postpaid plans are the dominant form of data plans in several countries of the world, as of 2011. To determine this data, we examined the standard mobile subscriptions for the largest mobile operator in each country and plotted the monthly cost (in $ after accounting for purchasing power parity) of 1GB of data usage. We note that the data plans in different countries have different broadband caps, which are not reflected in the graph, but may be found in [60]. The cost of 1GB of data/month has been plotted against the number of mobile broadband subscribers in each country; we observe that there is a large variation in cost. Additionally, we see that the countries of Africa (e.g., Kenya, South Africa), the Middle East (e.g., Turkey, Egypt), India and China tend to prefer prepaid plans, while postpaid plans dominate in Europe and the Americas. There is a weakly positive correlation between the price for mobile data and the number of broadband subscribers in the country.

Figure 15 more explicitly shows the correlation between prepaid mobile broadband data plans and lower per capita gross national incomes (GNI): we see that countries with a lower GNI per capita tend to have prepaid plans as the dominant subscription mode, with the exception of Australia. Moreover, two of the three countries offering both pre- and postpaid plans (Italy and Vietnam) have GNI per capita below the

---

[24]Prepaid plans are those in which a consumer pays for his/her data usage beforehand, while for postpaid plans, a consumer is billed for his/her monthly usage at the end of the billing cycle.

265
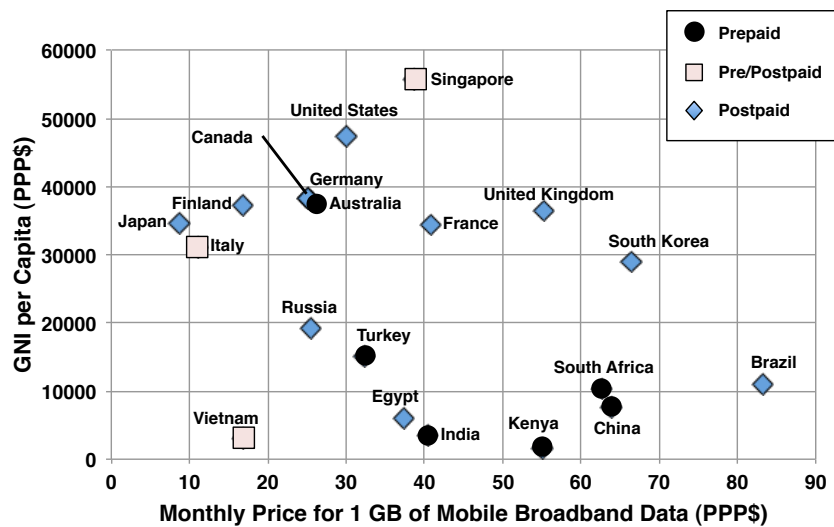
Figure 15: GNI per capita vs. monthly cost of 1GB of mobile data (data sources as in Figure 14).
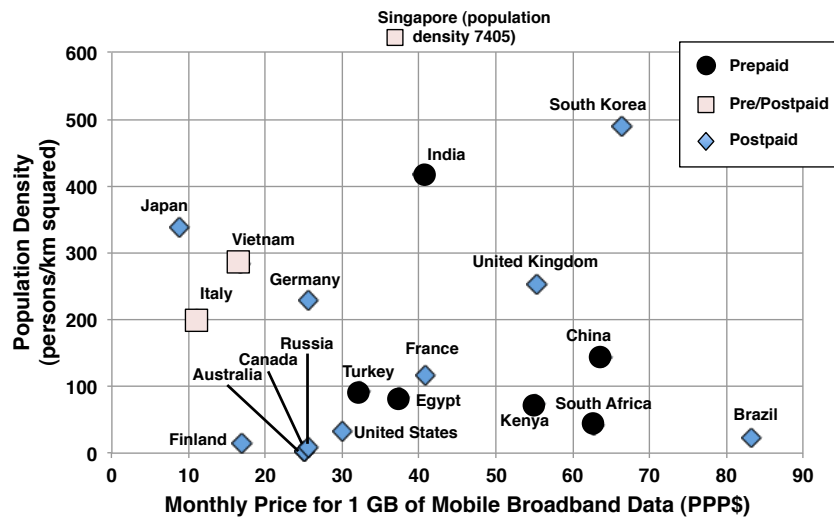
Figure 16: Population density vs. monthly cost of 1GB of mobile data (data sources as in Figure 14).

wealthiest 9 countries. Wealthier countries tend to have lower prices for mobile data, even accounting for purchasing power parity. In lower-GNI countries, it is possible that only a wealthy subset who can afford higher prices generally purchase mobile data plans; another explanation is that these countries have lower investment costs due to their relative wealth.

Infrastructure development is also related to population density: in denser countries, networks need to cover a smaller geographical area to reach the same number of people, resulting in lower investment costs. In the rural United States, small carriers often cite wireline backhaul over long distances as a major source of wireless network costs [1]. Figure 16 shows a negative correlation between population density and the price for 1GB of mobile data, consistent with our hypothesis that denser countries have lower infrastructure costs. Prepaid data plans are somewhat more popular in less dense countries, with India a notable exception: 83% of countries with prepaid plans have lower population density than 36% of countries with high population density and postpaid data plans. These countries also tended to have lower GNIs, which along with their population density suggest that their network infrastructure is less comprehensive than that of denser, wealthier countries.

References

[1] TRENDS: A report on rural telecom technology. National Exchange Carrier Association, 2010. https://www.neca.org/Trends_Report.aspx.

[2] Networks: Friends, Money, and Bytes, September-December 2012. https://www.coursera.org/course/friendsmoneybytes.

[3] 86CALLCHINA. China GPRS/EDGE data plans, 2011. http://www.86callchina.com/gprs-cdma-data-cards.htm.

[4] ANDREASSEN, R. M3i; requirements specifications; reference model, deliverable 1, m3i eu vth framework project, 2000. http://www.m3i.org/.

[5] ANDREWS, M., ÖZEN, U., REIMAN, M. I., AND WANG, Q. Economic models of sponsored content in wireless networks with uncertain demand. In *Proceedings of the Second Smart Data Pricing Workshop* (2013), IEEE.

[6] BBC. Africa's mobile phone industry 'booming', November 2011. November, http://www.bbc.co.uk/news/world-africa-15659983.

[7] BORENSTEIN, S. The long-run efficiency of real-time electricity pricing. *The Energy Journal 26*, 3 (2005), 93–116.

[8] BORENSTEIN, S., JASKE, M., AND ROSENFELD, A. Dynamic pricing, advanced metering, and demand response in electricity markets. Tech. rep., Center for the Study of Energy Markets, 2002. Working Paper.

[9] BRISCOE, B., DARLAGIANNIS, V., HECKMAN, O., OLIVER, H., SIRIS, V., SONGHURST, D., AND STILLER, B. A market managed multi-service internet, 2003. http://www.m3iproject.org/.

[10] BROWNING, J. Apple's Siri doubles iPhone data volumes. Bloomberg News, January 6 2012. http://www.bloomberg.com/news/2012-01-06/apple-s-voice-recognition-siri-doubles-iphone-data-volumes.html.

[11] BRUNEKREEFT, G. Price capping and peak-load pricing in network industries. *Discussion Papers 73* (2000), 1–9. University of Freiburg, Institute for Transport Economics and Regional Policy.

[12] CARON, S., AND KESIDIS, G. Incentive-based energy consumption scheduling algorithms for the smart grid. In *First IEEE International Conference on Smart Grid Communications* (2010), IEEE, pp. 391–396.

[13] CASTELLS, M., QIU, M., AND LINCHUAN, J. *Mobile Communication and Society: a Global Perspective : a Project of the Annenberg Research Network on International Communication*. The information revolution and global politics. MIT Press, Cambridge, MA, 2007.

[14] CHAO, H. Peak load pricing and capacity planning with demand and supply uncertainty. *The Bell Journal of Economics 14*, 1 (1983), 179–190.

[15] CHARLES RIVER ASSOCIATES. Impact evaluation of the California statewide pricing pilot. Tech. rep., Charles River Associates, 2005.

[16] CHAU, C.-K., WANG, Q., AND CHIU, D.-M. On the viability of paris metro pricing for communication and service networks. In *Proceedings of IEEE INFOCOM* (2010), IEEE, pp. 1–9.

[17] CHEN, B. X. Shared mobile data plans: Who benefits? New York Times, 2012. July 19, Bits Blog.

[18] CHEN, J., GHOSH, A., MAGUTT, J., AND CHIANG, M. QAVA: Quota aware video adaptation. In *Proceedings of ACM CoNEXT* (December 2012), ACM, pp. 121–132.

[19] CHETTY, M., BANKS, R., BRUSH, A., DONNER, J., AND GRINTER, R. You're capped: Understanding the effects of bandwidth caps on broadband use in the home. In *Proceedings of ACM CHI* (2012), ACM, pp. 3021–3030.

[20] CHETTY, M., BANKS, R., HARPER, R., REGAN, T., SELLEN, A., GKANTSIDIS, C., KARAGIANNIS, T., AND KEY, P. Who's hogging the bandwidth: The consequences of revealing the invisible in the home. In *Proceedings of ACM CHI* (2010), ACM, pp. 659–668.

[21] CHETTY, M., HASLEM, D., BAIRD, A., OFOHA, U., SUMNER, B., AND GRINTER, R. Why is my Internet slow?: Making network speeds visible. In *Proceedings of ACM CHI* (2011), ACM, pp. 1889–1898.

[22] CHETTY, M., TRAN, D., AND GRINTER, R. Getting to green: Understanding resource consumption in the home. In *Proceedings of ACM UbiComp* (2008), ACM, pp. 242–251.

[23] CHIANG, M. *Networked Life: 20 Questions and Answers*. Cambridge University Press, 2012.

[24] CIA. Country comparison: Population. CIA World Factbook, 2011. https://www.cia.gov/library/publications/the-world-factbook/rankorder/2119rank.html.

[25] CISCO SYSTEMS. Cisco visual networking index: Forecast and methodology, 2011-2016, May 30 2012. http://tinyurl.com/VNI2012.

[26] CISCO SYSTEMS. Cisco visual networking index: Global mobile data traffic forecast update, 2012-2017, February 6 2013. February 6, http://tinyurl.com/VNI2013-mobile.

[27] CLARK, D. D. Internet cost allocation and pricing. In *Internet Economics*, L. W. McKnight and J. P. Bailey, Eds. The MIT Press, Cambridge, MA, 1997, pp. 215–252.

[28] COCCHI, R., ESTRIN, D., SHENKER, S., AND ZHANG, L. A study of priority pricing in multiple service class networks. *ACM SIGCOMM Computer Communication Review 21*, 4 (1991), 123–130.

[29] COLOMBUS, L. How Google is driving mobile video market growth. Forbes, August 8 2012. http://www.forbes.com/sites/louiscolumbus/2012/08/27/how-google-is-driving-mobile-video-market-growth/.

[30] COMCAST. About excessive use of data, October 2012. September, http://customer.comcast.com/help-and-support/internet/data-usage-what-are-the-different-plans-launching.

[31] COMMUNICATIONS COMMISSION OF KENYA. Quarterly sector statistics report, 2011. http://www.cck.go.ke/resc/downloads/SECTOR_STATISTICS_REPORT_Q1_11-12.pdf.

[32] COURCOUBETIS, C., STAMOULIS, G. D., MANOLAKIS, C., AND KELLY, F. P. An intelligent agent for optimizing QoS-for-money in priced ABR connections. Tech. rep., Institute of Computer Science–Foundation for Research and Technology Hellas and Statistical Laboratory, University of Cambridge, 1998. Preprint.

[33] COURCOUBETIS, C., AND WEBER, R. *Pricing Communication Networks: Economics, Technology, and Modelling*. Wiley, 2003.

[34] DELGROSSI, L., AND FERRARI, D. Charging schemes for reservation-based networks. *Telecommunication Systems 11*, 1 (1999), 127–137.

[35] DU, P., AND LU, N. Appliance commitment for household load scheduling. *IEEE Transactions on Smart Grid 2*, 2 (June 2011), 411–419.

[36] EL-SAYED, M., MUKHOPADHYAY, A., URRUTIA-VALDÉS, C., AND ZHAO, Z. J. Mobile data explosion: Monetizing the opportunity through dynamic policies and QoS pipes. *Bell Labs Technical Journal 16*, 2 (2011), 79–100.

[37] EMISR. National broadband plan, July 2011. http://www.tra.gov.eg/emisr/Presentations/Plan_En.pdf.

[38] ERICSSON. Differentiated mobile broadband, January 2011. White Paper, http://www.ericsson.com/res/docs/whitepapers/differentiated_mobile_broadband.pdf.

[39] FITCHARD, K. New Netflix iOS app capitulates to bandwidth caps. GigaOm, May 31 2012. May 31, http://gigaom.com/mobile/new-netflix-ios-app-capitulates-to-bandwidth-caps/.

[40] FROELICH, J., FINDLATER, L., AND LANDAY, J. The design of eco-feedback technology. In *Proceedings of ACM SIGCHI* (2010), ACM, pp. 1999–2008.

[41] GANESH, A., LAEVENS, K., AND STEINBERG, R. Congestion pricing and user adaptation. In *Proceedings of IEEE INFOCOM* (2001), vol. 2, IEEE, pp. 959–965.

[42] GIBBENS, R. J., AND KELLY, F. P. Resource pricing and the evolution of congestion control. *Automatica 35*, 12 (1999), 1969–1985.

[43] GLASS, V., PRINZIVALLI, J., AND STEFANOVA, S. Persistence of middle mile problems for rural exchanges local carriers. Smart Data Pricing Workshop Talk, July 2012. http://scenic.princeton.edu/SDP2012/Talks-VictorGlass.pdf.

[44] GOLDMAN, D. 4G won't solve 3G's problems, March. 2011. http://money.cnn.com/2011/03/29/technology/4g_lte/index.htm.

[45] GOMEZ-IBANEZ, J. A., AND SMALL, K. A. *Road pricing for congestion management: a survey of international practice*. Transportation Research Board, Washington, DC, 1994. National Cooperative Highway Research Program.

[46] GRINTER, R., EDWARDS, K. W., AND NEWMAN, M. W. The work to make a home network work. In *Proceedings of ECSCW* (2005), Springer, pp. 469–488.

[47] GUPTA, A., STAHL, D., AND WHINSTON, A. Priority pricing of integrated services networks. In *Internet Economics*, L. W. McKnight and J. P. Bailey, Eds. The MIT Press, Cambridge, MA, 1997, pp. 323–352.

[48] HA, S., SEN, S., JOE-WONG, C., IM, Y., AND CHIANG, M. TUBE: Time-dependent pricing for mobile data. In *Proceedings of ACM SIGCOMM* (2012), ACM, pp. 247–258.

[49] HANDE, P., CHIANG, M., CALDERBANK, R., AND RANGAN, S. Network pricing and rate allocation with content provider participation. In *Proceedings of IEEE INFOCOM* (2009), IEEE, pp. 990–998.

[50] HANDE, P., CHIANG, M., CALDERBANK, R., AND ZHANG, J. Pricing under constraints in access networks: Revenue maximization and congestion management. In *Proceedings of IEEE INFOCOM* (2010), IEEE, pp. 1–9.

[51] HAUGE, J., CHIANG, E., AND JAMISON, M. Whose call is it? Targeting universal service programs to low-income households' telecommunications preferences. *Telecommunications Policy 33*, 3-4 (2009), 129–145.

[52] HAYEL, Y., AND TUFFIN, B. A mathematical analysis of the cumulus pricing scheme. *Computer Networks 47* (April 2005), 907–921.

[53] HECKMAN, O., DARLAGIANNIS, V., KARSTEN, M., AND BRISCOE, B. Tariff dissemination protocol, internet draft, ietf, 2002. http://www.m3i.org/papers/draft-ietf-tsvwg-ecn-ip-00.txt/.

[54] HELLER, F., AND BORCHERS, J. PowerSocket: Towards on-outlet power consumption visualization. In *Proceedings of ACM SIGCHI (extended abstracts)* (2011), ACM, pp. 1981–1986.

[55] HERTER, K. Residential implementation of critical-peak pricing of electricity. *Energy Policy 35*, 4 (2007), 2121–2130.

[56] HOLGUIN-VERAS, J., CETIN, M., AND XIA, S. A comparative analysis of us toll policy. *Transportation Research Part A: Policy and Practice 40*, 10 (2006), 852–871.

[57] HOLMES, T. Eco-visualization: Combining art and technology to reduce energy consumption. In *Proceedings of C&C* (2007), ACM, pp. 153–162.

[58] IDA SINGAPORE. Telecommunications, 2011. http://www.ida.gov.sg/Publications/20070822130650.aspx.

[59] IM, Y., JOE-WONG, Y., HA, S., SEN, S., KWON, T., AND CHIANG, M. AMUSE: Empowering Users for Cost-Aware Offloading with Throughput-Delay Tradeoffs. In *Proceedings of IEEE INFO-COM (mini-conference* (2013).

[60] ITU. Measuring the information society, 2011. http://www.itu.int/ITU-D/ict/publications/idi/2011/Material/MIS_2011_without_annex_5.pdf.

[61] JOE-WONG, C., HA, S., AND CHIANG, M. Time-dependent broadband pricing: Feasibility and benefits. In *Proceedings of IEEE ICDCS* (2011), IEEE, pp. 288–298.

[62] JOE-WONG, C., SEN, S., AND HA, S. Offering supplementary wireless technologies: Adoption behavior and offloading benefits. In *Proceedings of IEEE INFOCOM* (2013).

[63] JOE-WONG, C., SEN, S., HA, S., AND CHIANG, M. Optimized day-ahead pricing for smart grids with device-specific scheduling flexibility. *IEEE Journal on Selected Areas in Communications 30*, 6 (2012), 1075–1085.

[64] JOKSIMOVIC, D., BLIEMER, M. C. J., AND BOVY, P. H. L. Dynamic road pricing optimization with heterogeneous users. In *Proceedings of 45th Congress of the European Regional Science Association* (August 2005), European Regional Science Association.

[65] JOSEPH, D., SHETTY, N., CHUANG, J., AND STOICA, I. Modeling the adoption of new network architectures. In *Proceedings of ACM CoNEXT* (2007), pp. 5:1–5:12.

[66] KELLY, F. On tariffs, policing, and admissions control for multiservice networks. *Operations Research Letters 15* (1994), 1–9.

[67] KELLY, F. Models for a self managed internet. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences 358*, 1773 (2000), 2335–2348.

[68] KELLY, F. P., MAULLOO, A. K., AND TAN, D. K. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society 49*, 3 (1998), 237–252.

[69] KEY, P. Comcast, Level 3, Netflix, the FCC: Busy week for neutrality debate. Philadelphia Business Journal, 2010. December 1.

[70] KIM, T., HONG, H., AND MAGERKO, B. Design requirements for ambient display that support sustainable lifestyle. In *Proceedings of DIS* (2010), ACM, pp. 103–112.

[71] KLEINROCK, L. Research areas in computer communications. *Computer Communication Review 4*, 3 (July 1974).

[72] LANCASTER, H. Italy–telecoms, IP networks, digital media and forecasts. BuddeComm, 2011. http://www.budde.com.au/Research/Italy-Telecoms-IP-Networks-Digital-Media-and-Forecasts.html.

[73] LAOUTARIS, N., SIRIVIANOS, M., YANG, X., AND RODRIGUEZ, P. Inter-datacenter bulk transfers with netstitcher. *ACM SIGCOMM Computer Communication Review 41*, 4 (2011), 74–85.

[74] LEE, D., MO, J., WALRAND, J., AND PARK, J. A token pricing scheme for Internet services. In *Proceedings of the Seventh ICQT* (2011).

[75] LI, N., CHEN, L., AND LOW, S. Optimal demand response based on utility maximization in power networks. In *IEEE Power and Energy Society General Meeting* (2011), IEEE, pp. 1–8.

[76] LI, S., HUANG, J., AND LI, S. Y. R. Revenue maximization for communication networks with usage-based pricing. In *Proceedings of IEEE GLOBECOM* (2009), IEEE, pp. 1–6.

[77] LOCKTIME SOFTWARE. Netlimiter website, 2012. http://www.netlimiter.com/.

[78] LOISEAU, P., SCHWARTZ, G., MUSACCHIO, J., AND AMIN, S. Incentive schemes for Internet congestion management: Raffles versus time-of-day pricing. In *Proceedings of the Allerton Conference* (2011), IEEE, pp. 103–110.

[79] LOW, S. H., AND LAPSLEY, D. E. Optimization flow control-i: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking 7*, 6 (1999), 861–874.

[80] MACKIE-MASON, J. K., MURPHY, L., AND MURPHY, J. Responsive pricing in the Internet. In *Internet Economics*, L. W. McKnight and J. P. Bailey, Eds. The MIT Press, Cambridge, MA, 1997, pp. 279–303.

[81] MACKIE-MASON, J. K., AND VARIAN, H. R. Pricing the Internet. Computational economics, EconWPA, Jan. 1994.

[82] MALIK, O. In India, broadband means a 3G connection. GigaOm, July 2011. July 22, http://gigaom.com/broadband/in-india-broadband-means-a-3g-connection/.

[83] MARBACH, P. Analysis of a static pricing scheme for priority services. *IEEE/ACM Transactions on Networking 12* (March 2004), 312–325.

[84] MARCON, M., SANTOS, N., GUMMADI, K. P., LAOUTARIS, N., RODRIGUEZ, P., AND VAHDAT, A. NetEx: Cost-effective bulk data transfers for cloud computing. Tech. rep., Max Planck Institute for Software Systems, 2012.

[85] MCKNIGHT, L., AND BAILEY, J. *Internet Economics*. MIT Press, 1998.

[86] MOHSENIAN-RAD, A. H., AND LEON-GARCIA, A. Optimal residential load control with price prediction in real-time electricity pricing environments. *IEEE Transactions on Smart Grid 1*, 2 (2010), 120–133.

[87] MORTIER, R., RODDEN, T., TOLMIE, P., LODGE, T., SPENCER, R., CRABTREE, A., KOLIOUSIS, A., AND SVENTEK, J. Homework: Putting interaction into the infrastructure. In *Proceedings of ACM UIST* (2012), ACM, pp. 197–206.

[88] MURPHY, J., AND MURPHY, L. Bandwidth allocation by pricing in ATM networks. *IFIP Transactions C: Communications Systems C-24* (1994), 333–351.

[89] NEWMAN, J. Netflix has bandwidth cap sufferers covered. PC World, June 23 2011. June 23, http://www.pcworld.com/article/230982/Netflix_Has_Bandwidth_Cap_Sufferers_Covered.html.

[90] NGO, D. iCloud: The hidden cost for the magic, and how to avoid it. Cnet News, November 7 2011. http://www.cnet.com/8301-17918_1-57319231-85/icloud-the-hidden-cost-for-the-magic-and-how-to-avoid-it/.

[91] NTT DOCOMO. DOCOMO to introduce two prepaid data billing plans, 2011. http://www.nttdocomo.com/pr/2011/001543.html.

[92] ODLYZKO, A. Paris metro pricing for the Internet. In *Proceedings of the 1st ACM Conference on Electronic Commerce* (1999), ACM, pp. 140–147.

[93] OECD. OECD broadband portal, June 2011. June, http://tinyurl.com/oecd-mobile-data.

[94] PARRIS, C., AND FERRARI, D. A resource based pricing policy for real-time channels in a packet-switching network. Tech. rep., Tenet Group, ICSI, UC Berkeley, 1992. TR-92-018.

[95] PARRIS, C., KESHAV, S., AND FERRARI, D. A framework for the study of pricing in integrated networks. Tech. rep., Tenet Group, ICSI, UC Berkeley, 1992. TR-92-016.

[96] PASCHALIDIS, I. C., AND TSITSIKLIS, J. N. Congestion-dependent pricing of network services. *IEEE/ACM Transactions on Networking 8* (1998), 171–184.

[97] PETERSEN, D., STEELE, J., AND WILKINSON, J. WattBot: A residential electricity monitoring and feedback system. In *Proceedings of ACM SIGCHI (extended abstracts)* (2009), ACM, pp. 2847–2852.

[98] PIERCE, J., AND PAULOS, E. Beyond energy monitors: Interaction, energy, and emerging energy systems. In *Proceedings of ACM SIGCHI* (2012), ACM, pp. 665–674.

[99] PRINCETON EDGE LAB. Smart Data Pricing Forum website, July 31 2012. http://scenic.princeton.edu/SDP2012.

[100] PRINCETON EDGE LAB. DataMi website, 2013. http://scenic.princeton.edu/datami.

[101] PRINCETON EDGE LAB. DataWiz website, 2013. http://www.datawizapp.com.

[102] RIGNEY, C., WILLENS, S., RUBENS, A., AND SIMPSON, W. Remote Authentication Dial In User Service (RADIUS). RFC 2865 (Draft Standard), June 2000. Updated by RFCs 2868, 3575, 5080.

[103] ROOZBEHANI, M., DAHLEH, M., AND MITTER, S. Dynamic pricing and stabilization of supply and demand in modern electric power grids. In *First IEEE International Conference on Smart Grid Communications* (2010), IEEE, pp. 543–548.

[104] ROTO, V., GEISLER, R., KAIKKONEN, A., POPESCU, A., AND VARTIAINEN, E. Data traffic costs and mobile browsing user experience. In *Proceedings of the Workshop on Empowering the Mobile Web* (2006).

[105] SAMADI, P., MOHSENIAN-RAD, A., SCHOBER, R., WONG, V. W. S., AND JATSKEVICH, J. Optimal real-time pricing algorithm based on utility maximization for smart grid. In *First IEEE International Conference on Smart Grid Communications* (2010), IEEE, pp. 415–420.

[106] SCHATZ, A., AND ANTE, S. E. FCC chief backs usage-based broadband pricing. Wall Street Journal, 2010. December 2.

[107] SDP. Smart Data Pricing Workshop, IEEE INFOCOM, April 19 2013.

[108] SEN, S., GUÉRIN, R., AND HOSANAGAR, K. Shared versus separate networks: the impact of reprovisioning. In *Proceedings of the 2009 workshop on Re-architecting the internet* (2009), ReArch '09, pp. 73–78.

[109] SEN, S., GUERIN, R., AND HOSANAGAR, K. Functionality-rich versus minimalist platforms: a two-sided market analysis. *SIGCOMM Computer Communications Review 41*, 5 (Oct. 2011), 36–43.

[110] SEN, S., JIN, Y., GUÉRIN, R., AND HOSANAGAR, K. Modeling the dynamics of network technology adoption and the role of converters. *IEEE/ACM Transactions on Networking 18*, 6 (Dec. 2010), 1793–1805.

[111] SEN, S., JOE-WONG, C., AND HA, S. The economics of shared data plans. In *Proceedings of 22nd Annual Workshop on Information Technologies and Systems (WITS)* (2012).

[112] SEN, S., JOE-WONG, C., HA, S., BAWA, J., AND CHIANG, M. When the price is right: Enabling time-dependent pricing of mobile data. In *Proceedings of ACM SIGCHI* (2013), ACM.

[113] SEN, S., JOE-WONG, C., HA, S., AND CHIANG, M. Incentivizing time-shifting of data: A survey of time-dependent pricing for internet access. *IEEE Communications Magazine* (November 2012).

[114] SEN, S., JOE-WONG, C., HA, S., AND CHIANG, M. A survey of broadband data pricing: Past proposals, current plans, and future trends. *arXiv* (2012).

[115] SEN, S., JOE-WONG, C., HA, S., AND CHIANG, M. A survey of smart data pricing: Past proposals, current plans, and future trends. *ACM Computing Surveys* (2013).

[116] SHAKKOTAI, S., SRIKANT, R., OZDAGLAR, A., AND ACEMOGLU, D. The price of simplicity. *IEEE Journal on Selected Areas in Communication 26*, 7 (2008), 1269–1276.

[117] SHIH, J. S., KATZ, R. H., AND JOSEPH, A. D. Pricing experiments for a computer-telephony-service usage allocation. In *Proceedings of IEEE GLOBECOM* (2001), vol. 4, IEEE, pp. 2450–2454.

[118] SONGHURST, D. *Charging communication networks: From theory to practice*. Elsevier, 1999.

[119] STARKIE, D. Efficient and politic congestion tolls. *Transportation Research Part A: General 20*, 2 (1986), 169–173.

[120] TAYLOR, T. Megaco Errata. RFC 2886 (Historic), Aug. 2000. Obsoleted by RFC 3015.

[121] TEITELL, B. Cellphone overcharges putting a strain on many families. Boston Globe, 2012. September 22.

[122] TELECOMANDINTERNET. China's 3G mobile broadband subscribers: China Telecom, China Mobile and China Unicorn, Nov. 2010. http://tinyurl.com/7sgsk9w.

[123] TELECOMPAPER. Russia targets 60-80% broadband penetration in 5 yrs, December 2010. http://www.telecompaper.com/news/russia-targets-60-80-broadband-penetration-in-5-yrs.

[124] TELEGEOGRAPHY. Fixed and mobile broadband base increases 71% in 2010 to 34.2m (Brazil), 2010. http://tinyurl.com/brazil-3g.

[125] THE ECONOMIST. The mother of invention: Network operators in the poor world are cutting costs and increasing access in innovative ways, September 24 2009. Special Report, September 24.

[126] TWA NETWERK. Dark outlook of unlimited data plan for Korean smartphones, 2011. http://www.twanetwerk.nl/upl_documents/smart%20phone%20data%20plan%20ZK.pdf.

[127] U.S. OFFICE OF HIGHWAY POLICY INFORMATION. Toll facilities in the United States, July 2011. Publication No: FHWA-PL-11-032.

[128] VARAIYA, P. P., EDELL, R. J., AND CHAND, H. INDEX Project proposal, 1996.

[129] VERMA, S. Market trends: Worldwide, the state of mobile video, 2012. Gartner, 2012. February 10.

[130] VIETNAMNET BRIDGE. Vietnam telecom market remains unsustainable, May 2011. http://tinyurl.com/vietnam-3g.

[131] VYTELINGUM, P., RAMCHURN, S. D., VOICE, T. D., ROGERS, A., AND JENNINGS, N. R. Trading agents for the smart electricity grid. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1* (2010), International Foundation for Autonomous Agents and Multiagent Systems, pp. 897–904.

[132] WALRAND, J. Economic models of communication networks. In *Performance Modeling and Engineering*, Z. Liu and C. H. Xia, Eds. Springer Publishing Company, New York, 2008, ch. 3, pp. 57–90.

[133] WANG, Q., CHIU, D.-M., AND LUI, J.-S. Isp uplink pricing in a competitive market. In *Telecommunications, 2008. ICT 2008. International Conference on* (2008), pp. 1–6.

[134] WEN, C., AND TSAI, C. Traveler response to electronic tolls by distance traveled and time-of-day. *Journal of the Eastern Asia Society for Transportation Studies 6* (2005), 1804–1817.

[135] WORLD BANK. Population density, 2011. http://data.worldbank.org/indicator/EN.POP.DNST.

[136] YANG, J., EDWARDS, W. K., AND HASLEM, D. Eden: Supporting home network management through interactive visual tools. In *Proceedings of ACM UIST* (2010), ACM, pp. 109–118.

[137] YOO, C. S. Network neutrality, consumers, and innovation. *University of Chicago Legal Forum 25* (2009), 179. U of Penn Law School, Public Law Research Paper No. 08-40.

[138] ZHANG, Z.-L., NABIPAY, P., ODLYZKO, A., AND GUERIN, R. Interactions, competition and innovation in a service-oriented internet: an economic model. In *Proceedings of IEEE INFOCOM* (2010), pp. 46–50.

MPLS Virtual Private Networks

Luca Cittadini   Giuseppe Di Battista   Maurizio Patrignani

Summary

This chapter is devoted to Virtual Private Networks (VPNs) designed with Multi Protocol Label Switching (MPLS) [14, 15, 1], one of the most elusive protocols of the network stack. Saying that MPLS is "elusive" is not overemphasizing: starting from its arduous fitting within the ISO/OSI protocol stack, continuing with its entangled relationships with several other routing and forwarding protocols (IP, OSPF, MP-BGP, just to name a few), and ending with the complex technicalities involved in its configuration, MPLS defies classifications and challenges easy descriptions.

On the other hand, and in a seemingly contradictory way, the configuration of VPNs with MPLS is rather simple and elegant, despite the complexity of the underlying architecture. Also, MPLS flexibility and maintenance ease make it a powerful tool, and account for its ubiquity in Internet Service Providers' networks.

The chapter is organized as follows. Section 1 gives a brief introduction and motivation behind the concept of Virtual Private Network and explains why Layer 3 MPLS VPNs are by far the most popular widespread kind of VPNs deployed today.

In Section 2 we introduce the reader to basic concept and terminology about Label Switching (also known as Label Swapping) and Virtual Private Networks.

Section 3 gives a high-level step-by-step description of an MPLS VPN. This is based on three main ingredients: an any-to-any IP connectivity inside the network, a signalling mechanism to announce customer IP prefixes, and an encapsulation mechanism, based on MPLS, to transport packets across the network.

Section 4 explores in detail the complex interplay between IP and MPLS that is at the basis of MPLS VPNs.

More technical details about dynamic routing and connecting to the Internet, advanced usage of routing, and preserving IP-specific per-hop behavior are provided in Section 5.

Strengths and limitations of MPLS VPNs are discussed in Section 6. The same section proposes further readings on the subject.

The reader who is interested in getting only a high-level understanding on how MPLS VPNs work can read Sections 1, 2, and 3. An indepth view of MPLS VPNs can be gained by reading Sections 4 and 5.

1   Virtual Private Networks

After giving a brief introduction and motivation behind the concept of Virtual Private Network, this section explains why Layer 3 MPLS VPNs are by far the most popular widespread kind of VPNs deployed today.

1.1 The Need for Virtual Private Networks

The concept of Virtual Private Networks (VPNs) is essential in today's networks and will probably become paramount in tomorrow's networks, yet it is sometimes considered too advanced to be covered in a networking course. This apparently contrasts with the simplicity of the concept of a VPN: in its most generic form, a VPN is a closed ("Private") group of nodes that want to be connected in a network ("Network") and are willing to use virtual connections, or *pseudowires* ("Virtual") instead of physical connections.

Such a definition captures the essence of a VPN from the perspective of the customer. A network provider has a slightly different abstraction about a VPN, mostly because she has a different interpretation of the keyword "Network": within the graph that represents her own network, she needs to provide connectivity to a subset of the nodes. Despite being seemingly very easy, each of the other two keywords that appear in the definition hides a fair amount of complexity that is not obvious at first glance.

Virtual Where in the ISO/OSI stack does virtualisation happen?

Private Is there any authentication mechanism? Does the VPN need to preserve confidentiality of the messages?

Each of these questions has many possible answers, which is the reason why there are so many different types of VPNs in today's networks. For example, a peer-to-peer network can be seen as a VPN where pseudowires are transport sessions, there is no authentication amongst nodes and no traffic encryption, and the topology of the network is defined by a dynamic algorithm. At the opposite side of the spectrum we have optical networks, which can be seen as VPNs where pseudowires are light paths through optical switching devices, there is no authentication and no encryption, and the network topology is defined by simply configuring arbitrary pseudowires among the nodes.

In the context of VPNs, the term "virtualisation" indicates the technology that is used to multiplex traffic from multiple VPNs on the same underlying network topology. The most important feature of a VPN technology is what multiplexing technique is used and at which layer of the protocol stack. In general, pushing the multiplexing component down to the lower layers of the protocol stack (e.g., the physical or data-link layer) implies a higher implementation cost compared to the higher layers (e.g., the transport or application layer). For example, deploying an optical network to be able to run arbitrary pseudowires between two computers is several orders of magnitude more expensive than connecting those two computers to the Internet and writing a software that establishes a tunnel between them. On the other hand, multiplexing is transparent to upper layer protocols: for this reason, multiplexing at lower layers in the stack allows us to support a wider fraction of the protocol stack.

The most common layers where multiplexing happens are layer 2 and layer 3. A layer 2 VPN (L2VPN) transports packets of a specific layer 2 protocol and hence, thanks to the layered architecture of the protocol stack, is capable of supporting any kind of layer 3 protocol. L2VPN technologies join the nodes belonging to the same VPN within the same broadcast domain. For example, with a L2VPN, all nodes in the VPN could participate in the same VLAN and exchange Ethernet packets. We refer the reader to [3] for a detailed discussion of requirements for L2VPNs, and to [2] for a reference model of L2VPNs and a discussion of the main functional components. Analogously, a layer 3 VPN (L3VPN) transports packets of a specific layer 3 protocol and hence is capable of supporting any kind of layer 4 protocol. Nodes belonging to the same L3VPN can exchange IP packets that are routed through a provider network. We refer the reader to [8] for a detailed discussion of requirements for L3VPNs, and to [7] for a reference model of L3VPNs and a discussion of the main functional components.

277

1.2   Layer 3 VPNs and MPLS

Layer 3 VPNs are by far the most popular of VPNs deployed today. One reason is that layer 3 offers a good trade-off between deployment cost and transparency to end hosts. Another, perhaps stronger reason is that, as the Internet converged towards today's everything-over-IP scheme, it seemed natural to place the multiplexing component at the highest layer that supports transporting IP packets[1].

Despite a variety of technologies to realize virtual layer 3 services, most L3VPNs are based on the Multi Protocol Label Switching protocol (MPLS). The popularity of an L3VPN technology strongly depends on its ability to meet the demands of customers, providers, and vendors:

Customers' needs: Typical VPN customers (e.g., private companies, public administrations, etc.) have several geographically distributed sites and would like to have a unique IP network connecting all of them. Besides mere connectivity, they have other requirements: (i) they want to keep their own IP addressing plan for all the sites; (ii) they want their traffic to be logically separated from the traffic of other customers that happen to use the same shared infrastructure; and (iii) they want guaranteed quality of service.

Providers' targets: Providers have invested lots of resources in building their own network backbone. Since they have an existing infrastructure with many distributed PoPs (Points of Presence) connected to the backbone, they would prefer to sell pseudowires rather than physical connections to their customers. Among multiple techniques to implement pseudowires, providers prefer those that involve lower configuration efforts, which usually implies lower maintenance costs. Moreover, they want the implementation to be scalable with respect to the number of customers: the amount of state to keep in the core of the network should only depend on the network topology, not on the number of customer VPNs.

Vendors' strategies: No network technology can be easily deployed without meeting the strategies of network device producers and vendors, whose immediate aim is to sell many machines (possibly expensive carrier-grade routers) and, in the long run, to drive the shift from a variety of old technologies for VPNs (e.g., ATM or Frame Relay) to new technologies that are simpler to manage and hence have the potential to grow the vendor's market share.

After having introduced MPLS terminology and having given an overview of its main building blocks, in Section 6 we discuss the extent to which MPLS is able to meet the above requirements.

Throughout this chapter we will refer to a very simple scenario (see Fig. 1) where a provider has a network infrastructure with three PoPs (in Turin, Milan, and Rome) and offers connectivity to two customers. Customer 1 has two sites and has an IP addressing plan that allocates the $200.1.6.0/24$ to its site in Milan and the $10.0.0.0/24$ to its site in Rome. Customer 2 has two sites too and has an IP addressing plan that allocates the $10.0.0.0/24$ to its site in Turin and the $193.1.1.0/24$ to its site in Rome. Observe that the two customers have overlapping IP address space.

We will use the sample scenario to illustrate most of the concepts introduced in this chapter. The text that describes and refers to the sample scenario will be framed into shaded boxes like the one that encloses this paragraph. The configuration language is that of a leading router vendor and references can be found in [9].

---

[1]We are recently observing a similar convergence trend at layer 2 with Ethernet: consequently, in the last few years there has been a significant increase in the demand of virtual layer 2 services.
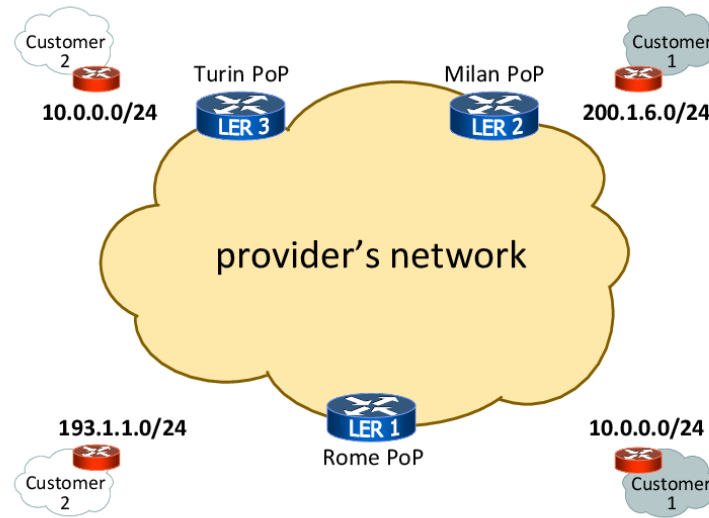
Figure 1: The sample network used throughout this chapter.

2 Background and Terminology

In this section we introduce the reader to basic concept and terminology about Label Switching (also known as Label Swapping) and Virtual Private Networks.

Throughout the chapter, we extensively refer to two tightly related yet distinct concepts: forwarding and routing. *Forwarding* is the process of receiving a packet from a network interface and deciding on which interface that packet should be sent. Usually, in order to minimize the latency of traversing a router, the decision about where to forward a packet is taken based on some pre-computed data structure. This is usually referred to as the *forwarding table* because a table is the simplest logical structure to accommodate forwarding information. A table can be used as a physical data structure if addresses can be matched exactly. However, IP addresses must be forwarded based on the longest matching prefix [10]. This implies that efficient IP lookups need a more sophisticated data structure than a table. We refer the reader to [18] for a discussion on various data structures and algorithms to speed up prefix-match lookups, which exceeds the scope of this chapter. In the following, we simply refer to the logical forwarding table, irrespective of the actual physical data structure.

Routing is the process by which each router builds its forwarding table and adapts it as the network topology changes over time.

Correspondingly, we have *forwarding and routing protocols*, where the formers describe the formatting rules for network packets and the conventions that routers and hosts have to follow in order to exchange them, while the latter describe packet formats and conventions used to exchange routing information among routers. The information that routing protocols provide is used by each router to populate its forwarding table.

Finally, standard network terminology distinguishes between the corresponding router's software layers. Namely, the layer where the forwarding process takes place is called *data plane* or *forwarding plane*, while the layer where the routing process is managed is called *control plane*.

279

Destination address	Egress interface
10.100.100.0/24	en2
10.100.200.128/25	en1
22.30.100.0/24	en2

Table 1: Structure of the forwarding table in the "forwarding by network address" approach.
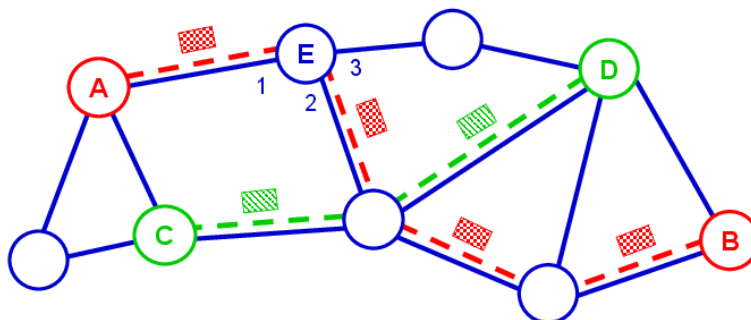
Figure 2: A label switching network (where labels are not swapped at each hop).

2.1 Label Switching

In this section we introduce the concept of label switching as a forwarding paradigm. After having described the fundamentals characteristics of label switching in general, we move to MPLS-specific details in the following sections. Traditionally, there are two different approaches to packet forwarding, each mapping to a specific structure of the forwarding table. They are called *forwarding by network address* and *label switching*.

The most intuitive approach is *forwarding by network address*, that is the approach of IP. When a packet arrives at a router, the router parses the destination address from the packet header and looks it up in its forwarding table. The forwarding table has a simple 2-column structure where each row maps a destination address to the egress interface that the packet should be forwarded to (see Table 1). For scalability and efficiency reasons, it is possible to aggregate several destination prefixes into a single row, provided that they can be numerically aggregated and that they share the same egress interface.

An alternative approach is known as *label switching*. Essentially, while forwarding by network address requires that the egress interface be chosen based on the *destination* of the packet, label switching requires that such an interface be chosen based on the *flow* the packet belongs to, where a flow corresponds to an instance of transmission, i.e., a set of packets, from a source to a destination and is identified by a tag (called *label*) attached to each packet of the flow.

As an example, Fig. 2 shows a label switching network where each flow has an associated label (labels

Incoming interface	Incoming label	Egress interface	Egress label
en2	101	en5	218

Table 2: Structure of the forwarding table in the "forwarding by label swapping" approach with per-interface label scope.
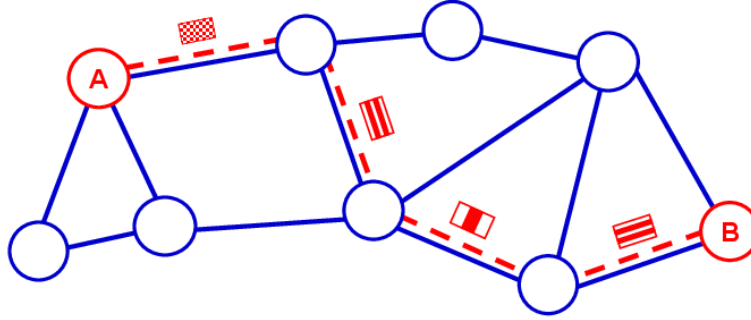
Figure 3: A label switching network (where labels are swapped at each hop).

Incoming label	Egress interface	Egress label
101	en5	218

Table 3: Structure of the forwarding table in the "forwarding by label swapping" approach with per-router label scope.

are represented with colors). Packets with a red label belong to the flow from router A to router B of Fig. 2. If a packet with a red label enters interface 1 of router E, it will exit from interface 2 with the same label.

If the label switching technologies followed the approach of Fig. 2 they would have the advantage that labels do not have to be changed at each hop. On the other hand, if they did they would have the big drawback of requiring a centralized control of the assigned labels, as labels should be unique for the entire network. Fig. 3 shows what actually happens in label switching networks, where labels are swapped at each hop. This choice requires that labels are unique for each router or for each interface only and does not need a centralized control. Fig. 4 illustrates the forwarding table of a router. Independent of whether labels are swapped at each hop or not, the forwarding paths towards the same destination node typically form a tree rooted at the destination node itself, even though this is not mandatory.

More formally, the operations performed by a label switching router can be summarized as follows. When the packet arrives at the router, the router extracts (*pops*) the label from the header, looks the label value up in its forwarding table, and finds (i) the egress interface the packet should be forwarded to, and (ii) a new label to apply (*push*) to the packet.

A forwarding process based on labels rather than destination addresses poses challenges to the corresponding routing protocols. In fact, the instances of flow traversing the network might be much more volatile than the addressing scheme used to identify their destinations. Before transmitting a new flow, a route from its source to its destination has to be computed and a new label has to be assigned to each leg of the route. As observed before, in order to facilitate the task of picking a new, unused, label, labels are not required to be unique for the entire network but are required to be unique for each router or for each interface only. This is why they have to be changed at each hop. Depending on whether labels have a per-interface or per-router scope, the forwarding table is structured as in Table 2 or Table 3, respectively. Observe that such a simple structure for forwarding tables allows efficient lookups (e.g., by using a hash table or a direct access table).

Label switching is not a unique feature of MPLS and it is not necessarily implemented at the network level of the protocol stack: other protocols, notably ATM and Frame Relay, traditionally adopt the same
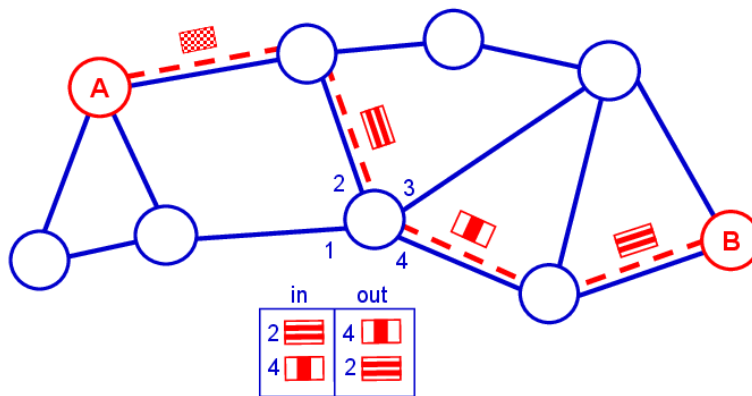
Figure 4: A label switching network where the forwarding table of a router is shown.

Layer 3	IP
Layer 2.5	MPLS
Layer 2	Ethernet, Frame relay, ATM, PPP, etc
Layer 1	Physical layer

Figure 5: MPLS and ISO/OSI network layers.

forwarding mechanism. Initially, the reason to prefer label switching was performance: looking up a label value in the forwarding table was much faster than looking up an IP address. Besides the fact that labels can take values in a much smaller range than IP addresses, label values can be looked up exactly, while IP addresses need to be looked up by the longest matching prefix. However, modern routers use extremely specialized hardware (e.g., content-addressable memories) and very efficient data structures (e.g., tries) to implement their forwarding tables, in such a way that the performance gain of label switching over forwarding by destination address is now believed to be no longer an argument.

2.2 MPLS header and terminology

The MPLS protocol brings the label switching forwarding paradigm to the extreme, managing, instead of a single label, a whole stack of labels, where the external one determines the egress interface. It does not fit the ISO/OSI model very well. The MPLS header is transported over L2 packets and can encapsulate L3 packets as well as L2 packets. Since MPLS does not fit the definition of either L2 protocols nor L3 protocols, it is frequently referred to as a "layer 2.5" protocol, emphasizing the fact that it requires L2 connectivity and can encapsulate IP packets.

When an IP packet from a router needs to be transported over an MPLS backbone, the first MPLS-enabled router in the network pushes an MPLS header in between the Ethernet header and the IP header. The resulting packet layout is depicted in Fig. 5.

The MPLS header consists of a *stack* of 4-byte records where each record has the following structure (depicted in Fig. 6):

- a **label** field (20 bits), which carries the label value;
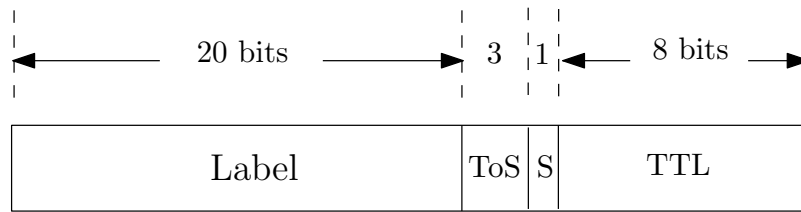
Figure 6: Structure of a record in an MPLS header.
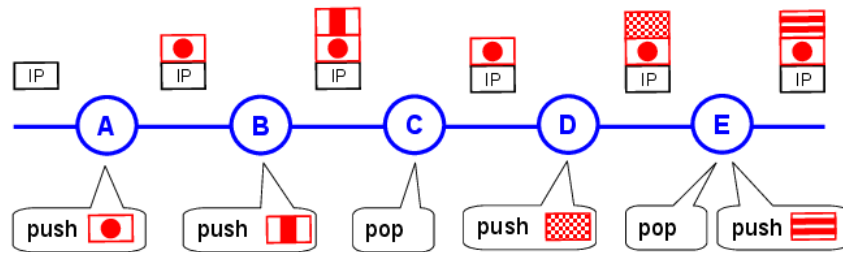
Figure 7: The evolution of the MPLS label stack as a packet traverses several routers that perfom random push and pop operations.

- a **ToS** field (3 bits) which is used to discriminate different levels of quality of service (QoS) and to carry explicit congestion notifications (ECN);

- a **bottom-of-stack** field (1 bit) which is set to 1 when the record is the last record in the stack; and

- a **TTL** field (8 bits) which is decremented at each hop, similarly to the TTL field in the IP header.

When an MPLS-enabled router receives a packet, it can perform three different operations: (i) *push* a label onto a (possibly empty) stack, (ii) *pop* a label from the stack (possibly resulting in an empty stack), or (iii) *swap* the top label of the stack, which can be seen as a pop operation followed by a push operation. Figure 7 shows the evolution of the MPLS label stack as a packet traverses several routers that perform random push and pop operations.

MPLS-VPN terminology uses specific names to distinguish routers that do not understand labels at all, routers that push (or pop) labels, and routers that simply swap labels. Routers belonging to the first group are called *customer edge* (CE) routers because they are not MPLS-enabled. Typically those are the customer's routers that need to be interconnected via an L3VPN. CE routers can only handle IP packets and are not aware of the MPLS layer which is used to implement the VPN.

Routers belonging to the second group are called *provider edge* (PE) routers, or *label edge routers* (LERs). They are placed at the edge of the MPLS backbone of the provider, have direct connectivity to the CE routers, and act as the access point for the customer to the VPN. While they need to perform label swap operations because they are part of the backbone, they spend most of their time pushing labels (when an IP packet comes from a CE router) and popping labels (when an MPLS packet needs to be forwarded to a CE router).

Routers belonging to the third group are called *provider* (P) routers, or *label switching routers* (LSRs). They are in the core of the MPLS network. Since they do not interact directly with non-MPLS routers, they
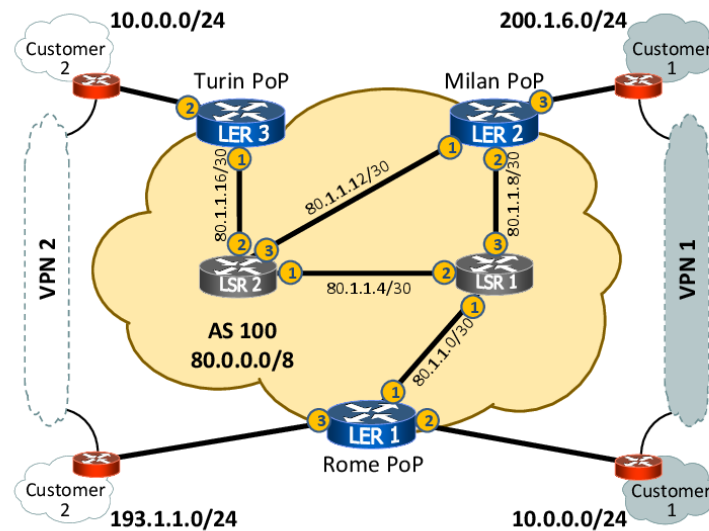
Figure 8: Inside the provider's infrastructure.

mainly perform label swapping operations in order to forward packets to other P or PE routers.

MPLS groups destinations into Forwarding Equivalence Classes (FEC). Packets that need to be forwarded to the same CE using the same path and with the same quality of service belong to the same FEC.

Fig. 8 shows some details of the provider's infrastructure of our scenario. It is both an MPLS network and an IP network (it has an MPLS data plane and an IP data plane).

If we look at it from the MPLS point of view, we can distinguish CE, PE, and P routers. The small, red routers placed at the customer premise in the corners of Fig. 8 are CE routers. CE routers are directly attached to the blue routers at the edge of the provider premise, which are the PE routers (or LERs). Finally, the grey routers in the core of the provider network are the P routers (or LSRs).

Since the provider network is also an IP network an IP address is given to the interfaces. To do this, our provider exploits prefix $80.0.0.0/8$. This prefix will not be announced outside the provider's network. The reason for the presence of label AS100 in the provider network will be explained soon.

The two CE routers serving Customer 1 are connected through a VPN called VPN1 (on the right side of Fig. 8). The two CE routers serving Customer 2 are connected through a VPN called VPN2 (on the left side).

3 Checkmate VPNs in Three Moves

In this section we give a high-level description of an MPLS VPN. Such a description is based on three main ingredients that we call "moves". We claim that a reader that understands these three moves will be able to checkmate this complex matter.

From the perspective of the customer, an MPLS VPN simply routes IP packets among customers CE routers, as if they were connected by a *pseudowire*. Observe that, as customers may have overlapping

address spaces, their packets cannot be simply routed through the provider network. Instead, they need to be encapsulated. It is tempting to implement such a pseudowire using a tunnel (e.g., GRE, IP-in-IP or IPSec) between PE routers where the customer packets travel across the provider network *encapsulated* into IP or IPSec packets. However, as the number of interconnected sites grows, manually managing configured tunnels and maintaining forwarding tables becomes excessively complex. For example, if we were to use tunnels to implement an L3VPN over 5 customer sites, a full-mesh topology would translate to 20 manually configured tunnels. Moreover, if the customer adds a new subnet to one of its sites, we need to update the forwarding tables of all our 5 PE routers. Observe that the complexity of managing tunnels can be eased by automatic setup mechanisms. However, such mechanisms are out of the scope of this chapter, therefore we refer the interested reader to [16, 13, 17].

The intrinsic problem with tunnels is that they rely on a pre-determined endpoint which is configured at tunnel setup time. Ideally, we would like to take advantage of the benefits of encapsulation without dealing with the issue of knowing the tunnel endpoint in advance. Namely, we would like packets to be encapsulated at the ingress PE and decapsulated at egress PE. We can split this goal into three high-level steps that we call moves:

Move 1: Achieve any-to-any IP connectivity among PEs,

Move 2: Define a signalling mechanism to distribute customer prefixes among PEs, and

Move 3: Define an encapsulation mechanism to transport packets from one PE to another across the network.

One of the key benefits of using encapsulation (Move 3) is that the complexity of configuring L3VPNs for customers is confined to PEs. The core of the network (i.e., P routers) does not need to know anything about customer prefixes: it simply needs to know how to transport packets from one PE to another (Move 1). This means that the size of the forwarding table of P routers depends on the number of PE routers rather than on the number of customer prefixes. Finally, if PE routers use a signalling mechanism to dynamically synchronize the list of customer prefixes, the only pieces of information that need to be manually configured at each PE are the L3VPN identifier and the IP address of the CE router.

In the following we elaborate each move in more detail.

3.1 Move 1 – Any-to-any IP connectivity among PEs

The first move is actually quite simple. It is nothing more than what any Internal Gateway Protocol (IGP) is designed to achieve: seamless, redundant and dynamic IP-level any-to-any connectivity. Since PEs are our encapsulation endpoints, we want them to be reachable independent of the availability of specific network interfaces. In other words, we do not want to use the IP address of physical interfaces for PEs, but loopback addresses. A loopback address is an address associated with a virtual interface of the router. Since it is virtual, a loopback interface it is active independent of the status of physical network interfaces. To fulfill Move 1, we simply assign a loopback address to each PE router and use an IGP (e.g. OSPF or IS-IS) to announce these addresses as /32 prefixes in order to ensure any-to-any connectivity among them.

Fig. 9 shows the loopback addresses assigned to the PEs of our example network. Also, we assume that routers use OSPF to propagate reachability information of loopbacks of routers.

Configuring routers to fulfill Move 1 is straightforward. In our sample network, the configuration of LER1 for Move 1 is as follows.
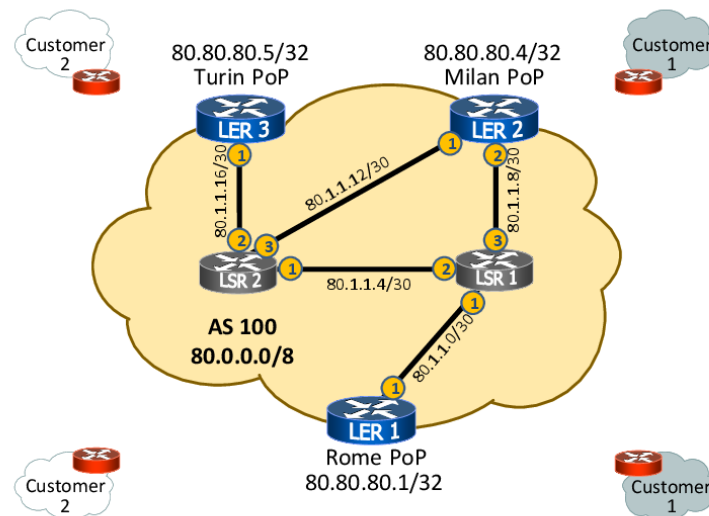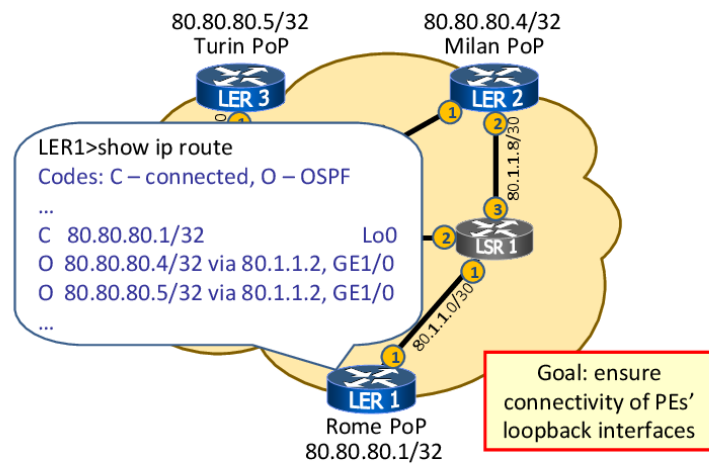
Figure 9: Loopbacks of PEs.
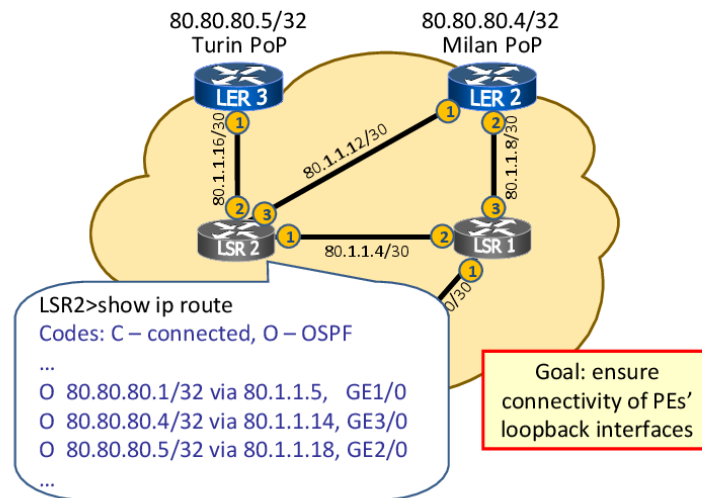

Figure 10: IP connectivity for LER1.

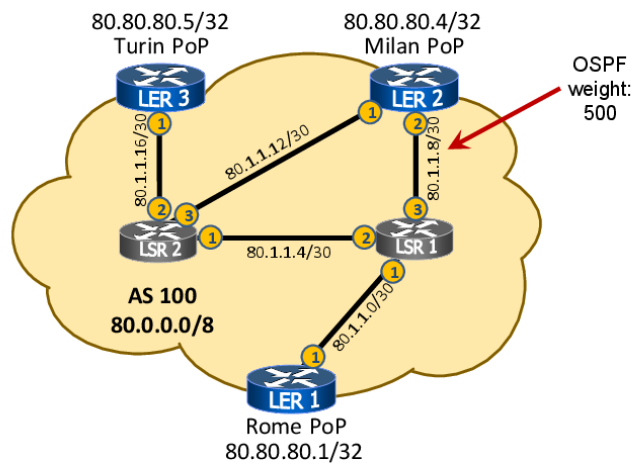Figure 11: IP connectivity for LSR2.

Figure 12: The OSPF weight of a link.

```
interface Loopback0
  ip address 80.80.80.1 255.255.255.255
interface GigabitEthernet1/0
  ip address 80.1.1.1 255.255.255.252
router ospf 10
  network 80.0.0.0 0.255.255.255 area 0
```

The first two lines assign an IP address to interface `loopback0`. The second pair of lines assign an IP address to interface `GigabitEthernet1/0` that connects LER1 with LSR1. The last two lines activate OSPF protocol.

Fig. 10 shows the result of command `show ip route` performed on router LER1. Fig. 11 shows the result of command `show ip route` performed on router LSR2. Command `show ip route` has the effect of showing the control plane routing table of routers.

In order to force a more interesting routing in the following part of the example, we set OSPF weight 500 for a specific link, discouraging the use of that link by the IGP routing protocol, as shown in Fig. 12.

3.2 Move 2 – Use BGP to distribute customer prefixes

In order to distribute reachability information about customer prefixes, MPLS relies on a variant of BGP called Multi-Protocol BGP (MP-BGP)[5]. Whereas BGP advertises reachability information for IPv4 addresses only, MP-BGP supports multiple *address families* (e.g., IPv4 and IPv6). Since advertising VPN addresses implies exchanging not only IPv4 prefixes, but also additional information to identify the VPN, MP-BGP treats VPNs as a separate address family. PE routers establish a full-mesh of iBGP peerings and each PE announces to all the other PEs the customer prefixes that it can reach via the CE router it is connected to. The Multi-Protocol extension to BGP is needed to introduce the concept of the "customer" (i.e., the "L3VPN identifier") which does not exist in plain BGP.

Compared with any ad-hoc signalling mechanism that could have been designed specifically for MPLS, the choice of using BGP has the advantage of relying on a well-known protocol and thus making the learning curve smoother for practitioners. Moreover, BGP has built-in mechanisms (e.g., route reflection) to scale as the number of PE routers increases.

Fig. 13 shows a high-level illustration of how the BGP peerings with LER3 and LER2 can be used by LER1 to announce customer prefixes.

Configuring   MP-BGP   peerings   is   very   similar   to   configuring   plain   iBGP   peerings.   Consider   the   following   snippet   from   the   configuration   of   router   LER1:
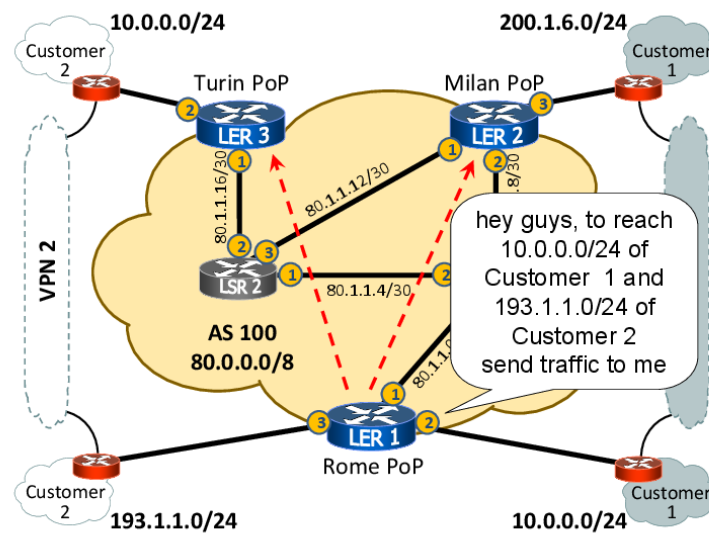
288

Figure 13: Use of BGP to distribute customer prefixes.

```
router bgp 100
  neighbor 80.80.80.4 remote-as 100
  neighbor 80.80.80.4 update-source Loopback0
  neighbor 80.80.80.5 remote-as 100
  neighbor 80.80.80.5 update-source Loopback0
  !
  address-family vpnv4
    neighbor 80.80.80.4 activate
    neighbor 80.80.80.5 activate
  exit-address-family
```

The first line starts the BGP configuration and states that the router belongs to AS100. Observe that all the routers are supposed to belong to Autonomous System (AS) 100. This AS number will not be necessarily propagated outside the provider's network and is only needed to establish peerings between PEs.

The following lines specify the BGP peerings. The presence of the "vpnv4" address family identifies LER2 and LER3 as MP-BGP neighbors of LER1.

3.3 Move 3 – Use MPLS encapsulation among PEs

Having performed Move 1 and Move 2, a PE router r is able to select the PE router r' that is connected to a given customer prefix (by Move 2). Also, r is able to forward IP packets to r' (by Move 1). The only piece missing is an encapsulation mechanism to transport IP packets from r to r'. One such encapsulation mechanism is MPLS: the PE router r encapsulates the IP packet by pushing two MPLS labels. The label at

the top of the stack (*outer* label) is switched by P routers in order to deliver the packet to router r'. The label at the bottom of the stack (*inner* label) is left untouched and it is used by the egress PE r' to identify the correct L3VPN. Observe that the inner label is necessary because r and r' could serve a variety of customers, and address spaces might be overlapping. For instance, routers r and r' could be serving two distinct VPNs for two customers, both using addresses in the RFC 1918 space.

Let us briefly recap how a packet is delivered across an MPLS cloud. When PE router r receives a packet from a CE router, it picks the VPN identifier and the destination address and, based on information contained in its MP-BGP Routing Information Base (RIB), it finds the PE router the packet should be delivered to. The MP-BGP RIB also contains the inner label that should be used. In our running example, the egress PE router is r'. Then, r pushes the inner MPLS label and an outer label which is guaranteed to deliver the packet to r'.

How does r pick this outer label? The outer label that maps to router r' is determined by the Label Forwarding Information Base (LFIB) of r, which is the forwarding table for MPLS.

The task of distributing labels and maintaining the LFIB of label switch routers is performed by the Label Distribution Protocol (LDP)[1][2]. LDP is able to setup a Label Switch Path (LSP) from one PE to another. In its simplest form, LDP maps an address prefix FEC (i.e., a FEC which represents an IP prefix) to a label. Each router receives LDP mappings from all its neighbors. In order to populate the LFIB, each router inspects its own forwarding table to determine the IP nexthop for the FEC, and picks the corresponding label. This way, LDP effectively creates a forwarding tree rooted at the egress point of the FEC, by simply importing the nexthop from the IP data plane (remember Move 1) at each intermediate hop.

It is extremely simple to configure a router to fulfill Move 3, because the LDP protocol can be safely run in the default configuration, and enabling MPLS encapsulation on specific interfaces is a single command. The configuration of LER1 for Move 3 is as simple as the following.

```
mpls label protocol ldp
interface GigabitEthernet1/0
  ip address 80.1.1.1 255.255.255.252
  mpls ip
```

4   An In-Depth View of MPLS VPNs

We have seen that the architecture of MPLS VPNs builds upon three building blocks: a working IP data plane that is capable of interconnecting the loopback addresses of PE routers, a BGP-based control plane to distribute reachability information about customer prefixes, and MPLS encapsulation among PEs.

While the first building blocks might seem straightforward at a first glance, there are a number of details which complicate the big picture but nevertheless are important in order to grasp the internals of MPLS VPNs.

4.1   IP Data Plane

IP connectivity between PE routers is easy to achieve using any suitable routing protocol. However, PE routers might be attached to a number of CEs of different customers, and must ensure that each CE is

---

[2]Alternative protocols such as RSVP and BGP can also serve the same purpose, but are out of the scope of this chapter.

VRF id	Ingress interface	Destination address	Egress interface
VRF-3	en5	10.100.200.32	en2

Table 4: Structure of the forwarding table in the "forwarding by network address" approach with Virtual Routing and Forwarding (VRF).

mapped to the correct VPN. A traditional IP data plane is unfit for this purpose since the IP address space of customers can overlap. Hence, a PE router must be able to route packets based on both the IP address and the specific VPN the packet belongs to. To accomplish this task, MPLS VPNs exploit a technique called Virtual Routing and Forwarding (VRF) which allows a router to have multiple (virtual) routing tables, potentially a separate virtual routing table for each network interface (either physical or logical). With this technique, mapping a CE to the correct VPN is as easy as configuring the corresponding interface within a specific VRF table. An MPLS inner label actually identifies a VRF instance.

One way to implement VRF while still maintaining a single forwarding table is using the ingress interface as an additional input parameter in the forwarding table. In such an implementation, the organization of the forwarding table of a router would be the one illustrated in Table 4. Observe that only the PE router needs to support VRF. The CE router is configured with a plain eBGP configuration and is completely unaware of the VRF implemented on the provider side.

Assigning an interface to a specific VRF instance is straightforward. In our sample network, we configure router LER1 as follows.

```
interface GigabitEthernet2/0
  ip vrf forwarding VPN1
  ip address 10.0.0.1 255.255.255.0
interface GigabitEthernet3/0
  ip vrf forwarding VPN2
  ip address 193.1.1.1 255.255.255.0
```

Address 10.0.0.1 is the address assigned to the interface that connects LER1 to Customer 1, while address 193.1.1.1 is assigned to the interface that connects LER1 to Customer 2.

4.2   MP-BGP Control Plane

Multi-protocol extensions to BGP allow us to segregate each L3VPN in a different *namespace*, identified by a proper VPN identifier. Separating namespaces is important because an IP prefix is not guaranteed to be unique across multiple VPNs: in practice, customers might want to use their own private IP address spaces, possibly overlapping with the address space of other customers. MP-BGP solves this issue by introducing the concept of VPN-IP addresses, that is, IP addresses tagged with a 8-byte VPN identifier which is called *route distinguisher* (RD). A VPN-IP address is nothing more than the concatenation of the RD and the IP prefix. By imposing that different VPNs be assigned distinct RD values, the uniqueness of VPN-IP addresses is guaranteed even in the presence of overlapping IP address space among customers. A special RD value consisting of 8 NULL bytes represent the default VPN, which allows MP-BGP to distribute information regarding pure IP routes alongside information about VPN-IP prefixes.
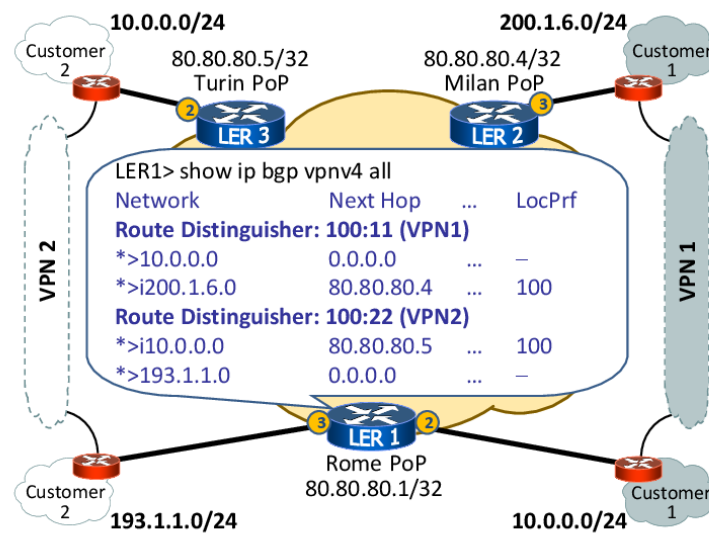
Figure 14: How MP-BGP can distribute per-VPN reachability information. Observe that the Local Preference attribute is not meaningful for locally originated prefixes, which are automatically preferred by the BGP decision process.

Observe that, while a VPN-IP prefix uniquely identifies a destination, it provides no information about the reachability of that destination. For this reason, MP-BGP messages associate VPN-IP prefixes with the MPLS labels that should be used for forwarding.

It is easy to assign an RD value to a single VRF instance:

```
ip vrf VPN1
   rd 100:11
ip vrf VPN2
   rd 100:22
```

Fig. 14 shows the output of command `show ip bgp vpnv4 all` on router LER1. This command has the same effect of `show ip bgp` but it shows the routing entries related to IPv4 VPNs. In this case the output highlights that LER1, in addition to its locally originated prefixes 10.0.0.0/24 with Route Distinguisher 100:11 and 193.1.1.0/24 with Route Distinguisher 100:22, knows two remote prefixes. Namely, it knows 200.1.6.0/24 with Route Distinguisher 100:11 and 10.0.0.0/24 with Route Distinguisher 100:22.

Tagging IP prefixes with a VPN identifier is an easy solution, but it is suboptimal in a specific use case which has seen increasing popularity recently: the so-called *extranets*. In its simple definition, an extranet is simply a connection between two different VPNs that are guaranteed to have non-overlapping IP address spaces. A realistic example might be a specific site of one customer that needs to connect to another specific site of another customer. A naive implementation of extranets would define an ad-hoc VPN and assign it a

new RD value. However, this solution is undesirable because it creates multiple VPNs that have duplicate entries, yielding a waste of router memory (to store the entries) and a waste of router's CPU time (to process update messages that are identical but for the RD value).

In order to overcome such limitations, MPLS decouples the concept of route distinguisher, which is used to segregate the address space in multiple namespaces, from the concept of *route target* (RT) which is another tag that is used to control which routes are imported in a given VPN and, similarly, which routes are exported from a given VPN. The route target is transported by MP-BGP using extended communities. More precisely, by exporting a route from a VPN we attach a user-defined RT community to all VPN-IP prefixes belonging to that VPN. On the other hand, by importing a given RT into a VPN we accept that every route having that RT value will be visible from the devices in that VPN.

Each VRF instance can be configured to import or export routes labelled with a specific Route Target value. In our simple example, assuming that no extranet connectivity is required between Customer 1 and Customer 2, each VRF instance can simply import a single RT value, as the following configuration snippet of LER1 shows:

```
ip vrf VPN1
  rd 100:11
  route-target export 100:1000
  route-target import 100:1000
ip vrf VPN2
  rd 100:22
  route-target export 100:2000
  route-target import 100:2000
```

This means that all the prefixes of VPN1 announced via MP-BGP by LER1 to any other PE are tagged with RT 100:1000. Also, any prefix that is tagged 100:1000 and is announced to LER1 via MP-BGP is imported into the VRF of VPN1. The configuration for VPN2 is similar.

Route Targets provide network operators with the flexibility of leaking specific routes into specific VRF instances, easing the deployment of extranets. Route Targets are transported in MP-BGP messages as extended BGP communities. For this reason, the configuration of MP-BGP peers needs to specify that the peer supports extended communities (which are disabled by default).

```
router bgp 100
  address-family vpnv4
    neighbor 80.80.80.4 activate
    neighbor 80.80.80.4 send-community both
    neighbor 80.80.80.5 activate
    neighbor 80.80.80.5 send-community both
  exit-address-family
```
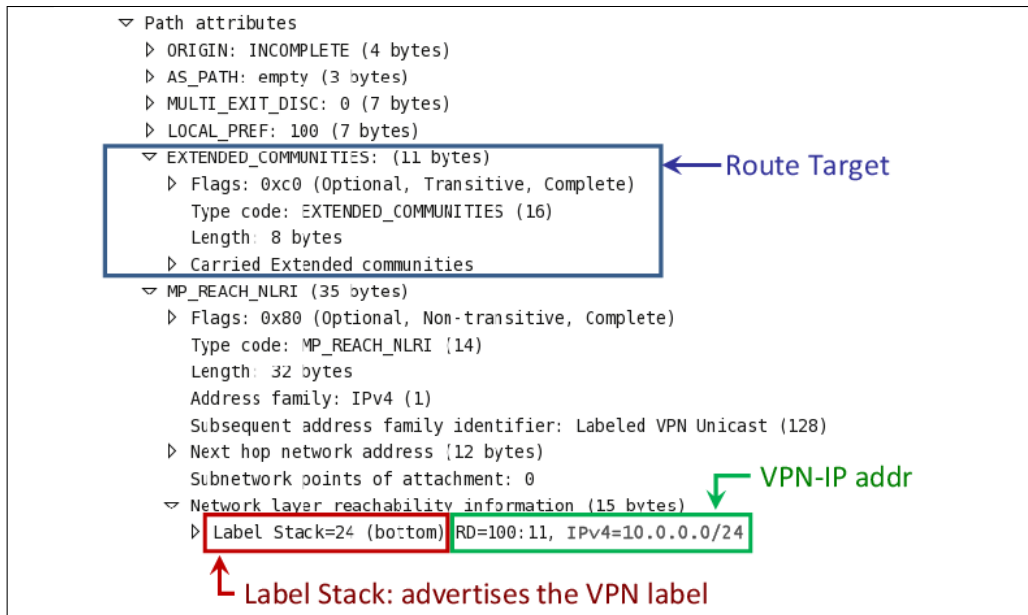
293

Figure 15: An MP-BGP signaling packet captured over the network.

To better understand the interplay between MP-BGP and the Route Targets, let us look at the content of an MP-BGP packet captured in our network (see Fig. 15). Observe how the route target in the blue frame is contained in the extended communities.

The announcements tells to the MP-BGP peer receiving it that the packets that will be received with the inner MPLS label 24 (red frame in the picture) will refer to the specified route target and the specified route distinguisher (green frame in the picture).

4.3   MPLS Control and Data Plane

The task of MPLS control plane is simply to establish Label Switched Paths (LSPs) between the loopback addresses of PE routers. LDP is in charge of populating and maintaining routers' LFIBs that implement the LSPs. In its most popular distribution mode, called *unsolicited downstream*, LDP works in the following way. Each router creates a label for locally originated prefixes (e.g., the loopback address). The binding between a label and a locally originated prefix is called a *local* binding. By contrast, a *remote* binding is a binding between a label and a remotely originated prefix. Each router advertises its local bindings to its neighbors. When a neighboring router receives a binding for prefix p_1 and label l_1 on interface i_1, it looks up its IP forwarding table to check whether the advertised prefix is routed on interface i_1. If this is the case, it picks another label l_2 and starts announcing a binding for p_1 and l_2. Meanwhile, it updates its LFIB with the tuple $\langle l_2, l_1, p_1, i_1 \rangle$. This means that when a packet arrives that is labelled l_2, the LFIB will swap l_2 with
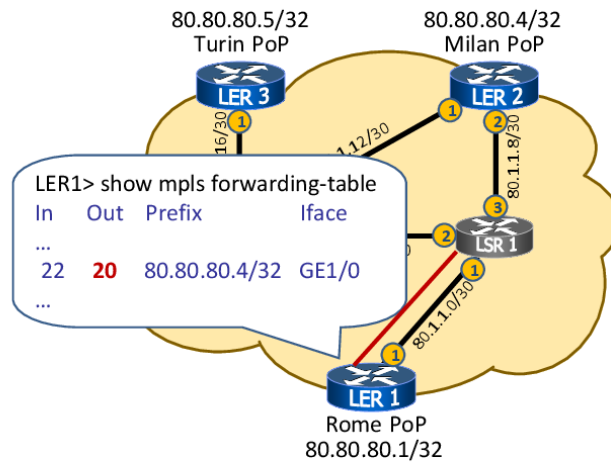
Figure 16: The MPLS forwarding table of LER1.

l_1 and deliver it via interface i_1[3].

Regarding MPLS data plane, we have already seen that the ingress PE router looks up its MP-BGP RIB to find the loopback address of the egress PE router, looks up its LFIB to select the outer label, and then encapsulates the received IP packet by pushing the inner and the outer MPLS labels. The packet is then label-switched across the MPLS network to the egress PE router using the bindings found in the LFIB of each router. As an optimization, the penultimate router, i.e., the router that receives a local binding from the egress PE, can pop the outer label, in such a way that the egress PE router only receives the inner label and therefore performs a single lookup in its LFIB.

Figs. 16, 17, and 18 show the MPLS forwarding tables of some routers of our network.

Figs. 19–24 illustrate the travel of a packet through our network.

First, Fig. 19 shows what happens when an IP packet originated by the Rome site of Customer 1 reaches the PE called LER1. Namely, it is encapsulated into an MPLS packet with two labels and then sent to LSR1. The inner label (yellow) identifies the VRF while the outer label (red) is the label used for the forwarding process.

Second (Fig. 20), the MPLS packet reaches LSR1, its outer red label is replaced with a blue label, and the packet is forwarded to LSR2.

Third (Figs. 21 and 22), the MPLS packet reaches LSR2. LDP makes LSR2 aware that it is the penultimate hop in the LSP. For this reason, LSR2 simply pops the outer label and forwards the packet to LER2.

---

[3]This explanation assumes per-router label scope, which is the default for most router vendors. An alternative is per-interface label scope, when the router can advertise different labels for each interface, which is used mostly on ATM interfaces.
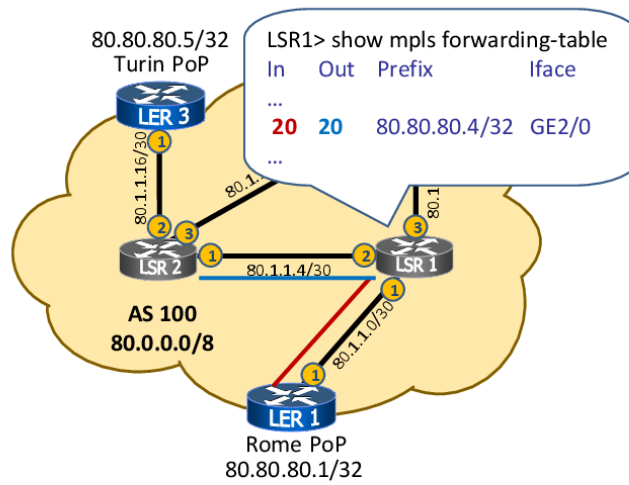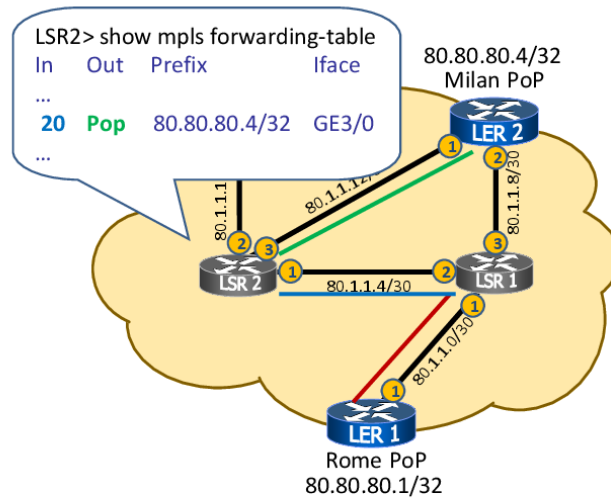
Figure 17: The MPLS forwarding table of LSR1.

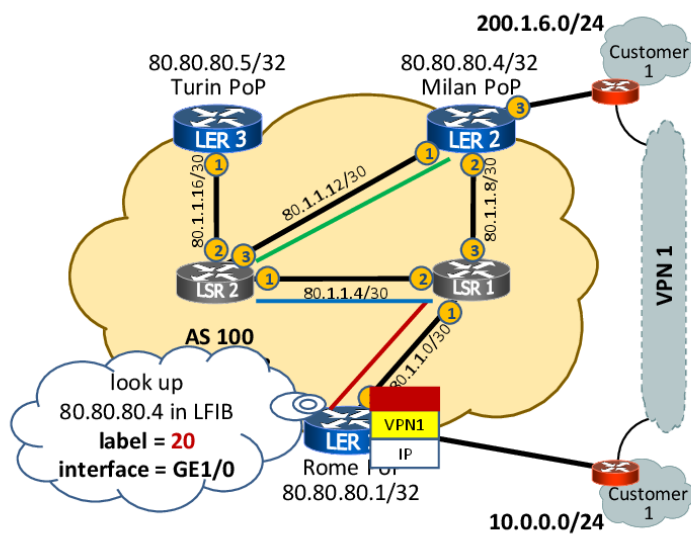Figure 18: The MPLS forwarding table of LSR2.

Figure 19: An IP packet originated by the Rome site of Customer 1 reaches PE router LER1.
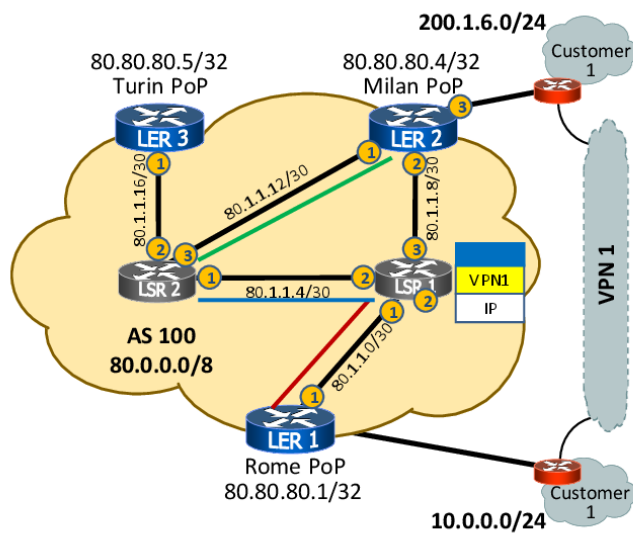

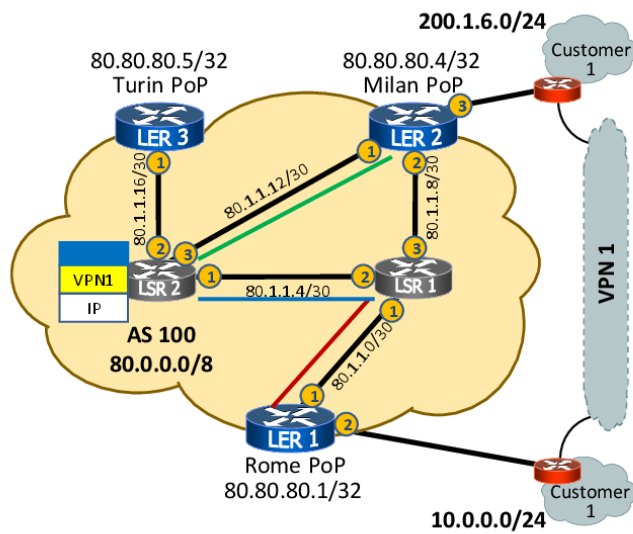Figure 20: An MPLS packet reaches P router LSR1.
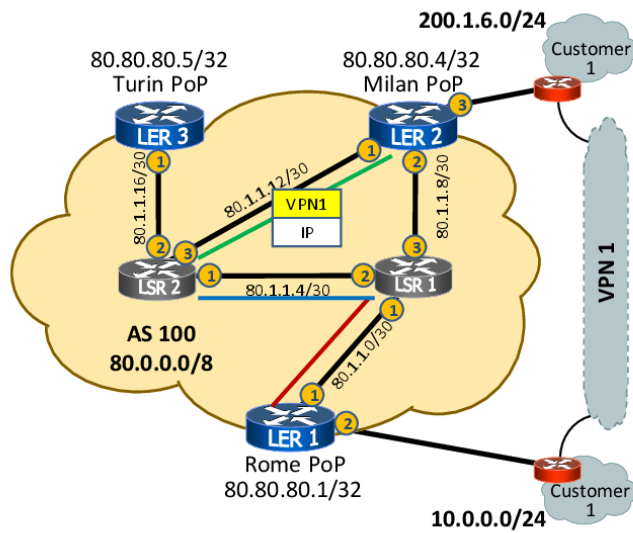
Figure 21: An MPLS packet reaches P router LSR2.

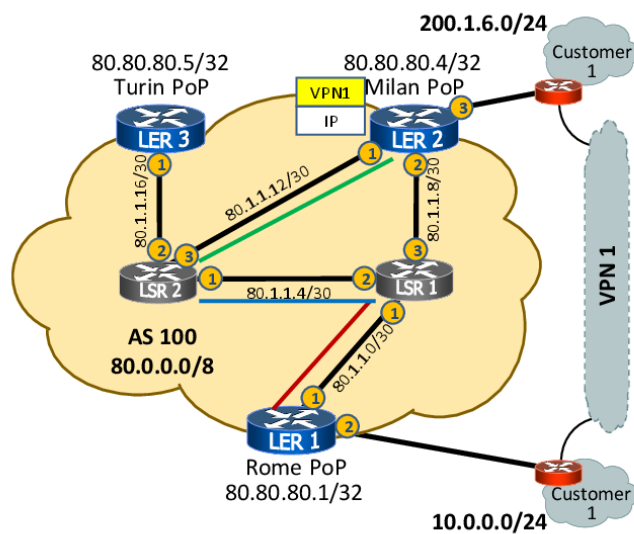Figure 22: An MPLS packet traveling to PE router LER2.

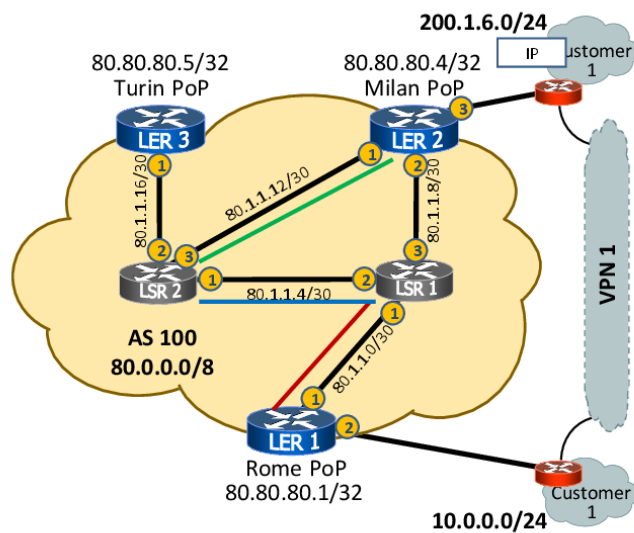Figure 23: An MPLS packet reaches PE router LER2.

Figure 24: An IP packet for Customer 1.

5   Advanced Topics

In this section we give more technical details about dynamic routing and connecting to the Internet, creating complex VPN topologies, and dealing with IP-specific features (e.g., MTU) that need extra care when encapsulation is involved.

5.1   Dynamic Routing and Connecting to the Internet

So far we have not yet discussed how the PE router can learn the prefixes that are served by its directly attached CE router. Of course, it is trivial to configure static routes on the PE, however this creates an undesirable coupling between the provider and the customer: whenever the customer wants to add a different IP subnet, it has to bother the provider to configure static routes before that IP subnet is reachable from other customer sites in the same VPN.

The solution is to have the CE and the PE establish an eBGP peering where the CE announces its local networks, while the PE announces all the networks that it learns in the same VPN. Observe that the, contrary to the MP-BGP peerings among PEs, peerings between CEs and PEs are pure eBGP peering: the CE does not know anything about VPNs and route distinguishers. It is the MP-BGP process on the PE router that takes care of processing the reachability information learned from the CE and updating the VPN reachability information accordingly.

Setting up an eBGP session in order to use the MPLS-VPN service may discourage those customers who do not have a strong BGP expertise. In such cases, usually the providers also offer to their customers CE management and configuration.

A BGP peering between the CE and the PE also allows a CE in a VPN to announce a default route, causing all other sites in the same VPN to route Internet traffic via that CE router. This might be advantageous if the customer's policy forces Internet traffic to pass through a centralized checkpoint (e.g., a firewall or a proxy). However, this is not the only way to connect a VPN to the Internet. For example, a PE might be configured to forward natively all the packets from a CE which do not match any VPN route. Alternatively, the default route might be given its own route target, and whenever a VPN site needs Internet access the PE simply imports that route target in the corresponding VPN routing table. We refer the reader to [14] for a discussion of alternatives to get Internet access within a VPN.
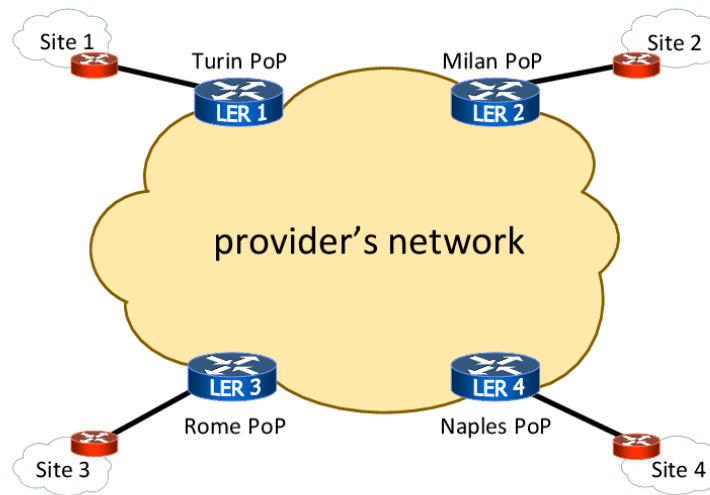
Figure 25: A configuration where a single customer has four sites: Site 2, 3, and 4 are only allowed to exchange traffic with Site 1 in Rome.

```
router bgp 100
  address-family ipv4 vrf VPN1
  neighbor 10.0.0.2 remote-as 65001
  exit-address-family
  !
  address-family ipv4 vrf VPN2
  neighbor 193.1.1.2 remote-as 65002
  exit-address-family
```

5.2 Designing Complex VPNs

So far we have assumed any-to-any connectivity within a VPN, i.e., all sites of a VPN communicate with all other sites. Sometimes more sophisticated configurations are needed. For example we might have a VPN where not all pairs of sites are allowed to exchange packets. A typical situation is the so called hub-and-spoke configuration, where a customer has a main site and several peripheral sites and the peripheral sites can communicate only through the main site.

How to do this is illustrated in the following example.

A suitable use of Route Distinguishers and Route Targets allows sophisticated configurations like the one shown in Fig. 25 where Rome is the main site and Turin, Milan, and Naples are the peripheral sites.

We can choose Route Distinguisher `100:1` for all four sites and split the customer VPN into three VPNs. VPN1 is used to connect Turin with Rome, VPN2 is used to connect Milan with Rome, and VPN3 is used to connect Naples with Rome.

For each VPN we define a distinct Route Target:

VPN1: `100:1000`

VPN2: `100:2000`

VPN3: `100:3000`

The configuration of peripheral sites, like for example Turin, is as follows:

```
ip vrf siteTurin
        rd 100:1
        route-target import 100:1000
        route target export 100:1000
```

Rome's PE configuration (the hub) is as follows:

```
ip vrf siteRome
        rd 100:1
        route-target import 100:1000
        route target export 100:1000
        route-target import 100:2000
        route target export 100:2000
        route-target import 100:3000
        route target export 100:3000
```

In this way Rome imports all the Route Targets and exports all the Route Targets and is hence able to communicate with all sites. On the other hand a peripheral site like Turin imports and exports Route targets only with respect to Rome and hence is able to communicate with Rome only.

5.3   ToS, TTL, and MTU

Whenever encapsulation of IP packets happens, there are three main questions that arise:

1. what happens to the ToS / DSCP information in the IP header that the customer might have set in order to properly prioritize traffic?

2. what happens to the TTL field in the IP header and how does encapsulation cope with forwarding loops?

3. how does encapsulation affect MTU for upper layer protocols?

Luckily, MPLS has an easy answer for the first two questions. Recall from Fig. 6 that MPLS has dedicated fields for ToS and TTL. When the ingress PE router receives an IP packet from the CE router, it simply

copies the values of ToS and of TTL in the MPLS header. More precisely, a push operation implies copying ToS and TTL from the IP header to the MPLS header. Conversely, a pop operation implies copying the TTL value from the MPLS header back to the IP header. This way, the TTL continues to serve as a hop count[4] even within the MPLS network, and P routers can honor the quality of service parameters related to the ToS field.

Regarding the third question, since an MPLS label takes 4 bytes and the PE router pushes two of them, the MTU within the MPLS network should be at least 8 bytes larger than the MTU that the CE is aware of. Given that modern OSes tend to perform path MTU discovery by default, MTU is becoming less of an issue for MPLS deployments. Rewriting the Maximum Segment Size TCP options at the PE router is also a common solution, even though it does not support UDP traffic.

6 Summary

6.1 Strengths of MPLS VPNs

After having described the details of MPLS VPNs, we are able to discuss the extent to which the goals that we stated in Section 1.2 are met.

By using Route Distinguishers and label stacks within the provider cloud, customers can retain their IP address plan and the traffic belonging to different customers is properly segregated. Moreover, the configuration of CE routers is completely unaware of MPLS-specific details. Since MPLS transports QoS information by copying the ToS field from the IP header, MPLS VPNs can in principle provide different forwarding treatment to different packets. However, the architecture of MPLS VPNs does not inherently support QoS, because LSPs are simply built from the underlying IP plane. Other mechanisms (e.g., [4]) can be employed to compute LSPs based on QoS features.

Providers are able to keep the configuration in the core of the network extremely simple and scalable: in fact, the configuration of P routers does not depend on the number of deployed VPNs. Since the backbone is only concerned with transporting packets from a PE to another, the sizeof the forwarding table of P routers only depends on the number of PEs and does not depend on the number of prefixes of VPNs. Configuring a new VPN implies modifying the configuration of the PE routers that are directly connected to the customer's sites. Moreover, such a configuration boils down to assigning a unique RD and RT and establishing eBGP peerings with the CE routers.

6.2 Limitations of MPLS VPNs

Virtual Private Networks designed with MPLS have also some known limitations. At least the following should be mentioned.

- BGP know-how is needed to configure the customer CE. As discussed in Section 5.1, this sometimes forces the carrier to provide also CE management.

- Customer CEs need to support BGP if eBGP peering are established with the PEs. This may not be the case for cheap entry-level router models. As an alternative, BGP can be replaced by static routes

---

[4]Observe that when the TTL in the MPLS header reaches 0 (e.g. in a traceroute), a P router does not know how to send the corresponding ICMP error back to the sender, because it lacks information about VPNs. A naive yet effective solution is to generate the ICMP packet and label-switch it to the egress PE anyway. The egress PE (which has information about VPNs) will then send the ICMP packet back to the sender.

configured on PEs, sacrificing part of the flexibility that a dynamic routing protocol between CEs and PEs provides.

- The P in the VPN acronym stands for "Private". However, MPLS VPNs are only private at routing level: no authentication, confidentiality, or integrity is provided by the architecture. For instance, the provider can inspect all customers' traffic in plaintext. Even worse, since the separation is enforced at the routing level, it turns out that the ability of guaranteeing isolation within the same VPN actually depends on the provider's topology [6].

- The basic MPLS architecture lacks support for quality of service. QoS can usually be offered on top of MPLS, e.g., by establishing LSPs which reflect traffic engineering policies. However, adding traffic engineering and quality of service support comes at the cost of keeping more state in the network, hence posing scalability concerns [12, 19].

- In most configurations, the customer and the provider share the job of maintaining a network, which potentially complicates debugging routing and connectivity problems.

6.3   Further Readings

The interested reader could refer to the classical books about MPLS authored by Minei and Lucek [11] and by De Ghein [9].

Most of the technologies regarding MPLS are defined by RFCs. These include the main MPLS VPNs architecture [15, 7], label distribution via LDP [1], Layer 2 VPNs [2], BGP variants [5, 14, 13], and RSVP-TE [4].

Considerations about about MPLS VPNs integrity and scalability can be found in [6] and [12, 19], respectively.

Acknowledgments

References

[1] ANDERSSON, L., MINEI, I., AND THOMAS, B. LDP Specification. RFC 5036, 2007.

[2] ANDERSSON, L., AND ROSEN, E. Framework for Layer 2 Virtual Private Networks (L2VPNs). RFC 4664, 2006.

[3] AUGUSTYN, W., AND SERBEST, Y. Service Requirements for Layer 2 Provider-Provisioned Virtual Private Networks. RFC 4665, 2006.

[4] AWDUCHE, D., BERGER, L., GAN, D., LI, T., SRINIVASAN, V., AND SWALLOW, G. RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC 3209, Dec. 2001.

[5] BATES, T., CHANDRA, R., KATZ, D., AND REKHTER, Y. Multiprotocol Extensions for BGP-4. RFC 4760, 2007.

[6] BUSH, R., AND GRIFFIN, T. G. Integrity for virtual private routed networks. In *In Proc. IEEE INFOCOM* (2003).

[7] CALLON, R., AND SUZUKI, M. A Framework for Layer 3 Provider-Provisioned Virtual Private Networks. RFC 4110, 2005.

[8] CARUGI, M., AND MCDYSAN, D. Service Requirements for Layer 3 Provider-Provisioned Virtual Private Networks. RFC 4031, 2005.

[9] DE GHEIN, L. *MPLS Fundamentals*. Cisco Press, Dec. 2006.

[10] FULLER, V., AND LI, T. Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan. RFC 4632, 2006.

[11] MINEI, I., AND LUCEK, J. *MPLS-Enabled Applications: Emerging Developments and New Technologies*. Wiley, Oct. 2005.

[12] MINEY, I. Scaling considerations in MPLS networks. Nanog 35, 2005.

[13] MOHAPATRA, P., AND ROSEN, E. The BGP Encapsulation Subsequent Address Family Identifier (SAFI) and the BGP Tunnel Encapsulation Attribute. RFC 5512, 2009.

[14] ROSEN, E., AND REKHTER, Y. BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4364, 2006.

[15] ROSEN, E., VISWANATHAN, A., AND CALLON, R. Multiprotocol Label Switching Architecture. RFC 3031, 2001.

[16] T. WORSTER, Y. R., AND ROSEN, E. Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE). RFC 4023, 2005.

[17] TOWNSLEY, M. MPLS over various IP tunnels. Nanog 30, 2004.

[18] VARGHESE, G. *Network Algorithmics: An Interdisciplinary Approach to Designing Fast Networked Devices (The Morgan Kaufmann Series in Networking)*. Morgan Kaufmann, Dec. 2004.

[19] YASUKAWA, S., FARREL, A., AND KOMOLAFE, O. An Analysis of Scaling Issues in MPLS-TE Core Networks. RFC 5439, Feb. 2009.

Collaboration Opportunities for
Content Delivery and Network Infrastructures

Benjamin Frank, Ingmar Poese, Georgios Smaragdakis,
Anja Feldmann, Bruce M. Maggs, Steve Uhlig,
Vinay Aggarwal, and Fabian Schneider

1   Motivation

The Internet is a hugely successful man-made artifact that has changed society fundamentally. Imagine the effect a prolonged outage of the Internet would have: (1) Youngsters wouldn't know how to interact with their peers and how to spend their leisure time as they increasingly rely on social networks, online games, YouTube, and other online entertainment offerings. (2) Manufacturing would hit a roadblock as the communication paths within and between companies increasingly rely on the Internet. (3) Control of critical infrastructures would be hampered as it increasingly relies on the Internet for gathering input data and propagating control information.

In becoming a hugely successful infrastructure, the usage of the Internet and thus its structure has also undergone continuous changes. Usage has changed from dominance by email and FTP in the early days, to the World Wide Web (WWW) from 1995 to 2000, to peer-to-peer applications (P2P) from 2000 to 2007, back to the WWW since 2007. These changes are driven in part by the Internet users' interests as well as how content, including user generated content, is made available.

When considering the current application mix and traffic streams in the Internet, the latest buzz is that "Content is King" just as Bill Gates [28] predicted in his essay from 1996. Hereby, the term content has to be seen very broadly and encompasses everything from commercially prepared content, e.g., broadcast and interactive TV, news, and software, to user-generated content, e.g., videos uploaded to YouTube, and photos uploaded to Flickr, to interactive activities, e.g., online games. Or to quote Bronfman [56], the head of a major music producer and distributor: "What would the Internet be without 'content'? It would be a valueless collection of silent machines with gray screens. It would be the electronic equivalent of a marine desert—lovely elements, nice colors, no life. It would be nothing."

The idea of content delivery being the fundamental operation around which to design future Internet architecture for comes as no surprise. In fact, the idea of Content-Centric Networking (CCN) [92] is guided by this principle. Change, however, takes time, and when hundreds of million of devices are involved, change can only be made slowly. Before such a novel and radically different architecture such as CCN is available or potentially deployable, the Internet in its current state must cope with the challenge of delivering ever-increasing amounts of content to Internet users.

Accordingly, it appears that solely providing connectivity to end users is no longer sufficient for Internet Service Providers (ISPs). Yet, connectivity is a crucial ingredient and some authors, e.g., Andrew Odlyzko [136] have opined that enabling communication is the main task of the Internet network infrastructure. In his paper "Content is not king" he claims that "Content will have a place on the Internet, possibly a

substantial place. However, its place will likely be subordinate to that of business and personal communication".

At this point it is crucial to realize that the producers of content are usually not the operators of today's Internet infrastructure. Nonetheless, both content producers and network operators depend on each other. In fact, neither the Internet infrastructure operators nor the content producers can be successful without the other. After all, the content producers want to ensure that their content gets to Internet users with reasonable performance for which they need to rely on the network infrastructure. On the other hand, the network infrastructure providers have to transport the content and manage the infrastructure to satisfy the demand for content from their subscribers. It is this symbiosis between the two parties that motivates our work collaboration between content producers and network operators in delivering content.

Outline: We start this chapter with a short introduction in Section 2. Then, in Section 3, we set the stage by providing an overview of today's Internet network infrastructure, discussing how Internet Service Providers (ISPs) perform traffic engineering, and reviewing the Domain Name System (DNS), an essential component of any Web-based content-delivery architecture. Next, we review current trends in Internet traffic and the application mix as well as traffic dynamics in Sections 4 and 5.

We finish the overview with a brief summary on the background of content delivery in Section 6. Here, we assume that the reader is familiar with the basic architecture of the Web. There are excellent text books on this topic, e.g., [103]. Given that there are several approaches to content delivery, we provide a general high level description of how different Content Delivery Infrastructures work. Since there are also many peer-to-peer based content delivery systems we provide a short review of the basic P2P architectures as well. For additional background on P2P we refer the reader to, e.g., [34, 164].

An overview of the current content delivery spectrum is presented in Section 7. Here we discuss various types of Content Delivery Infrastructures (CDIs) which range from Web-based Content Distribution Networks (CDNs) to Hybrid CDNs to peer-to-peer (P2P) systems. Furthermore, in Section 8 we turn to the challenges that each party involved in Internet content delivery faces separately today.

Finally, we turn to the state of the art of collaboration between networks and content providers. We start by outlining the collaboration incentives for each member of the content delivery landscape in Section 9. Next we review the collaboration schemes that have been discussed in research as well as at the Internet Engineering Task Force (IETF) in Section 10. We briefly introduce the well-known approaches and summarize their key functions. We then pick two collaboration schemes, namely the P2P Oracle and the Provider-aided Distance Information System (PaDIS) for a case study. In Section 11.1 we discuss the P2P Oracle with regards to its effect on the P2P system as well as on network operations. Likewise, the second case study discusses the model of the Provider-aided Distance Information System in Section 11.2, including a large scale analysis based on real traffic traces. Section 12 outlines a possible future direction for collaboration between content providers and network operators. We conclude this part of the chapter in Section 13.

Summary: This chapter builds upon the student's basic knowledge of how the Internet infrastructure operates, i.e., as a network of networks. After reading this chapter the student should have a fundamental understanding about how content distribution via the Internet works today, what the challenges are, and which opportunities lie ahead. Moreover, the chapter points out how all parties—including end users—can benefit from the collaboration between ISPs and content providers. Indeed, simple, almost intuitive, means will enable such collaboration.

307

2 Introduction

Recent traffic studies [79, 107, 145] show that a large fraction of Internet traffic is due to content delivery and is originated by a small number of Content Delivery Infrastructures (CDIs). Major CDIs include highly popular rich-media sites like YouTube and Netflix, One-Click Hosters (OCHs), e.g., RapidShare [23], Content Delivery Networks (CDNs) such as Akamai and Limelight, and hyper-giants, e.g., Google, Yahoo!, and Microsoft. Gerber and Doverspike [79] report that a few CDIs account for more than half of the traffic of a US-based Tier-1 carrier. Poese et al. [145] report a similar observation from the traffic of a European Tier-1 carrier. Labovitz et al. [107] infer that more than 10% of the total Internet inter-domain traffic originates from Google, and Akamai claims to deliver more than 20% of the total Internet Web traffic [135]. Netflix alone, a company that offers a high definition video video-on-demand streaming service, is responsible for a significant fraction of the traffic in North America ISPs during peak hours [154, 69].

To cope with the increasing demand for content, CDIs have deployed massively distributed server infrastructures to replicate content and make it accessible from different locations on the Internet [172]. These infrastructures have multiple choices as to how and where to place their servers. As described in [112], the main approaches are (1) centralized hosting, (2) data center-based CDIs, (3) edge-cache-based CDIs, and (4) peer-to-peer (P2P) networks. Approaches 2 and 3 scale content delivery by distributing the content onto dedicated infrastructures. These infrastructures can be composed of a few large data centers, a large number of edge caches, or any combination thereof.

To complicate matters further, some of these infrastructures are entangled with the very infrastructures that provide network connectivity to end-users. For example, one of the largest players in content delivery, Akamai, operates more than 120,000 servers in more than 2,000 locations across nearly 1,150 ISP networks [135, 13]. Google is reported to operate tens of data centers and front-end server clusters worldwide [105, 170, 83]. Microsoft has deployed its content delivery infrastructure in 24 locations around the world [125]. Amazon maintains at least 5 large data centers and caches in at least 21 locations around the world [19]. Limelight operates thousands of servers in more than 22 delivery centers and connects directly to 600 networks worldwide [114]. Last but not least, P2P networks rely on a huge number of end users to store, replicate, and distribute content.

Despite the significant entanglement between the infrastructures that deliver content and the network connectivity fabric, our knowledge of their interactions is largely through the literature on network interconnections, e.g., see the recent book by W. Norton [134]. Given the nature of network interconnections, previous work has studied the interactions from an economic perspective [123, 24, 111]. The limited knowledge available about the settlements between networks have led researchers to try to reason about why peering choices are made [38] and what drives the evolution of the Internet [50].

Most of the literature has considered the interactions between content and the network indirectly, e.g., through peerings and traffic measurements, despite recent changes in Internet traffic [79, 107] that have shown the importance of content and applications. The observed changes in traffic, either through direct traffic measurements [63, 65, 173, 107, 2], or through inference [124, 186, 185, 87, 142, 166] have repeatedly shown how volatile traffic can be. With the rise of user-generated content and large shifts in content popularity, traffic volatility has become especially relevant.

Handling changes in traffic has traditionally been done through traffic engineering (TE). Initially, traffic engineering was used by large network operators to optimize the utilization of their networks [27]. The vast majority of the traffic engineering literature has therefore focused on traffic engineering inside a single network [61, 70, 181, 26, 101, 71]. In reality, most of the traffic in the Internet is exchanged between different networks [107], and especially directly between data centers and residential ISPs [2]. Organizations that originate a lot of content, e.g., Google, connect directly to a large number of other networks [107], and need

to optimize how content leaves their networks. Organizations that provide Internet access to broadband or mobile users typically wish to optimize how traffic enters their networks, as most users still download more content than they upload. In between, transit ISPs try to balance the load of the traffic exchanged between the networks they connect.

Traditional traffic engineering aims at reducing the likelihood that bottlenecks arise inside a given network due to mismatches between network provisioning and expected demand. Changes in network provisioning are slow, taking place over time scales of weeks or months. Popular content, on the other hand, generates bursts in demand over much smaller time scales, e.g., hours or minutes. Today's Internet requires much more reactive network control techniques than those we have today, and these techniques must take content delivery into consideration. A few steps have been made in this direction. Indeed, collaborative approaches [53, 117, 76] have been proposed to help deal with the traffic generated by content delivery infrastructures. Even in the case of P2P, portals have been proposed to allow P2P applications and users to communicate with ISPs to receive updated views of their networks [182]. In broad terms, all information CDIs are missing today for optimizing their operations is available to ISPs. Combined with the already proposed schemes for collaboration, it is surprising how little real collaboration is performed in today's Internet between these parties.

In this chapter, we analyze the operation of CDIs as well as network operators. The analysis demonstrates the potential for fruitful collaboration. We argue that for collaboration to become more common, it is important for every party in the content delivery landscape, i.e., the content delivery infrastructures, the network operators, and the end users, to benefit. Finally, we present, in depth, two systems that have incentives for every party and that can readily be used today.

3   Internet Network Infrastructure

The Internet Network Infrastructure is provided by a set of Internet Service Providers (ISPs). An ISP is, in general terms, an organization that provides access to the Internet for its customers. The Internet is structured by the interconnection of multiple individual networks run by ISPs. However, control of an individual network remains solely with the ISP operating it. Figure 1 shows how the Internet is structured today [107]. Here, the ISPs run their own networks. This forces a clear distinction between the individual network that an ISP runs and the global Internet as a network of networks. Also, from this, it can be deduced that nobody has control over the Internet, but instead each ISP has only control over its own network and the direct connections to other networks.

To be able to interconnect with other networks, an ISP needs to operate an autonomous system (AS). An AS is an administrating entity, generally under the control of one administrative domain. On the technical side, each AS is usually managed by an Interior Gateway Protocol (IGP), e.g., OSPF [128] or ISIS [138] while the Border Gateway Protocol (BGP [152]) is the de-facto standard for interconnecting different ASes. For more information and additional details about the Internet topology, we'd like to refer the reader to Chapter 7 of this book [179].

3.1   Traffic Engineering in an AS

The greatest challenge for an ISP is to keep its infrastructure operating efficiently. This is especially hard, since the ISP itself controls neither the behavior, nor the source nor destination of the majority of the traffic it carries. The destination of the traffic is determined by the end-users the ISP sells services to, while the source is usually operated by a Content Delivery Infrastructure (CDI). The behavior is dictated through end-users requesting content, and by the operational choices of the CDI. ISPs today tackle the problem of network
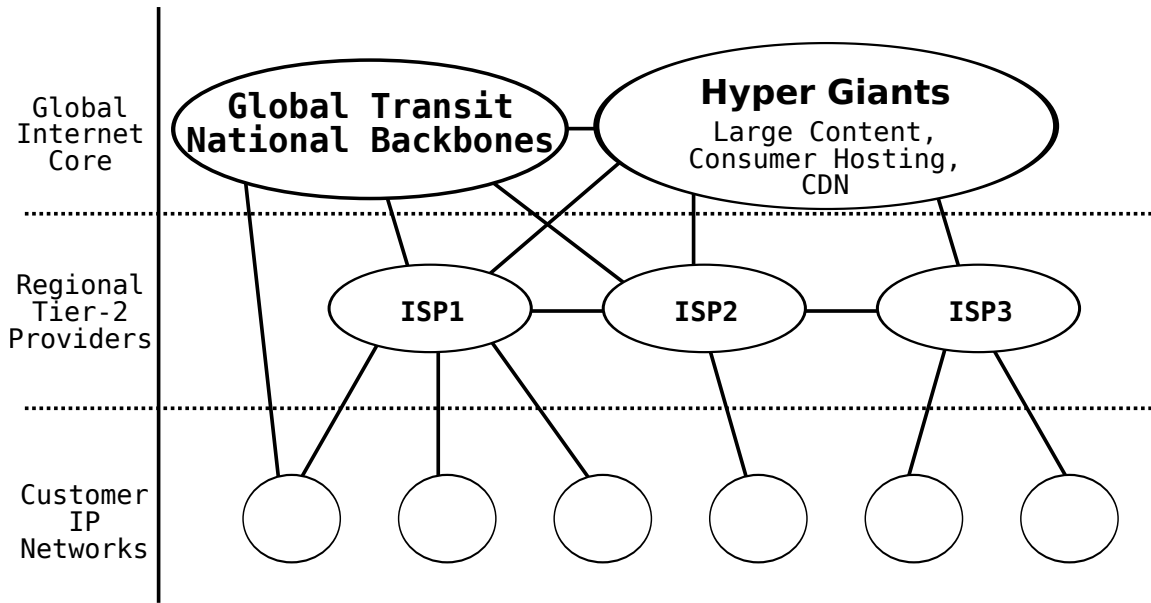
Figure 1: Layout of the Internet Structure (Reprinted from [145], ©ACM, 2010. Included here by permission
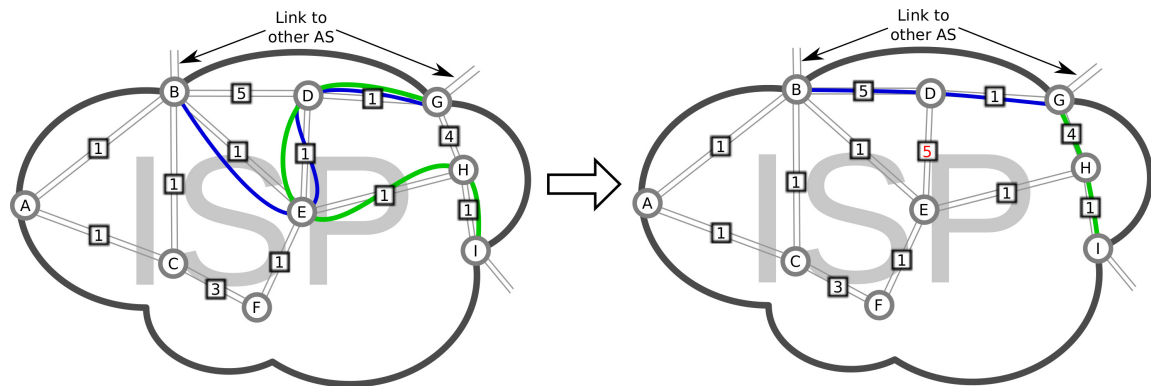sion

Figure 2: IGP based Traffic Management Example

operation efficiency by performing Traffic Engineering (TE). In its broadest sense, today's TE encompasses
the application of technology and scientific principles to the measurement, characterization, modeling, and
control of Internet traffic [27]. Today, traffic engineering reduces to controlling and optimizing the routing
function and to steering traffic on an Origin-Destination (OD) flow basis through the network in the most
effective way.

Traffic Engineering encompasses multiple steps in order to be performed successfully. First, an ISP needs

310

to record its traffic volume in terms of Origin-Destination flows. This means keeping traffic statistics of how much traffic flows from one router in the network to another. Once the OD flows have been successfully recorded, TE uses this information to simulate the network behavior with different IGP configurations. The goal of these simulations is to find an IGP configuration that spreads the network load as evenly as possible.

Figure 2 shows an example of how an IGP configuration can be used to engineer traffic. The labeled circles represent routers, while the numbers in the squares represent the IGP-weight for the link. For ease of presentation, the weights for each link are set to the same value for both directions. An OD flow, which starts at one router and finishes at another, takes the path through the network that yields the smallest sum over all weights along the path. For example, in the starting configuration of the network (Figure 2 (left)) the flow IG does not take the direct path $I \rightarrow H \rightarrow G$ two, since according to the IGP weights, a more effective path exists. In fact, the path $I \rightarrow H \rightarrow E \rightarrow D \rightarrow G$ has an accumulated weight of 4 instead of 5 (green path). All traffic at router I destined for router G takes this path. Similarly, all traffic that originates from B and goes to G follows the path $B \rightarrow E \rightarrow D \rightarrow G$ (blue path). Also, both paths share links, leading to a possible overload situation. In order to solve this problem, we choose to modify the link weight between the routers D and E. By increasing the weight from 1 to 5 (marked red in the right network), the blue as well as the green paths are shifted to the direct path. The change is shown in Figure 2 (right).

This simple diagram allows for illustrating multiple caveats that IGP based traffic engineering introduces. First, IGP-based traffic engineering affects traffic on an OD-flow basis only. This means that the path from one router to another can be changed, but the traffic on the OD flow cannot be split onto multiple paths. Secondly, the change of one weight can affect multiple OD-flows at the same time. Thus, the weights have to be changed very carefully. In the worst case, it might not be possible to fully separate some OD-flows due to the network layout.

One caveat is not immediately obvious but needs to be taken into account when performing traffic engineering. While the link weights are usually known to all routers, they are propagated by messages that routers exchange. This propagation takes time, which can lead to short-term inconsistencies in the view of a network. We again use Figure 2 for illustrating this. When the link weight is changed as described in the example explained before, routers D and E update their routing. This has an immediate effect on the traffic from B to G. With the update, the shortest path from router E to G is now $E \rightarrow H \rightarrow G$. In accordance, E configures its routing to send all traffic for G through H. However, H has not converged at this point and still uses the old path ($H \rightarrow E \rightarrow D \rightarrow G$). Thus, H still sends all traffic for G towards E. As long as H uses the outdated IGP weight information, all traffic for G that reaches either E or H is sent back and forth between the two routers. This forwarding, on the one hand, likely overloads the link. On the other hand, most traffic that is affected by this will be dropped due to its time-to-live (TTL) running out.

The work of Francois et al. [72] shows that it is possible to gradually change IGP weights by sequentially ordering changes. Accordingly, routing loops like those in the example are avoided. However, these changes still require time during which the network can be in a transient state with overloaded links. Besides the challenges induced by optimizing the IGP, this approach also assumes that traffic is predictable and stable over time. By running simulations based on past traffic aggregates to engineer the routing for the future, it is implicitly assumed that traffic patterns remain similar over a longer period of time.

With the emergence of CDIs, however, traffic has become volatile in terms of its origin. In fact, CDIs can shift massive amounts of traffic in a matter of seconds from one server cluster to another. While this behavior is needed and propagated by CDIs to cope with volatile demand surges, it is in stark contrast to the ISP's traffic engineering, which assumes traffic behavior to be stable for days, weeks or sometimes months.
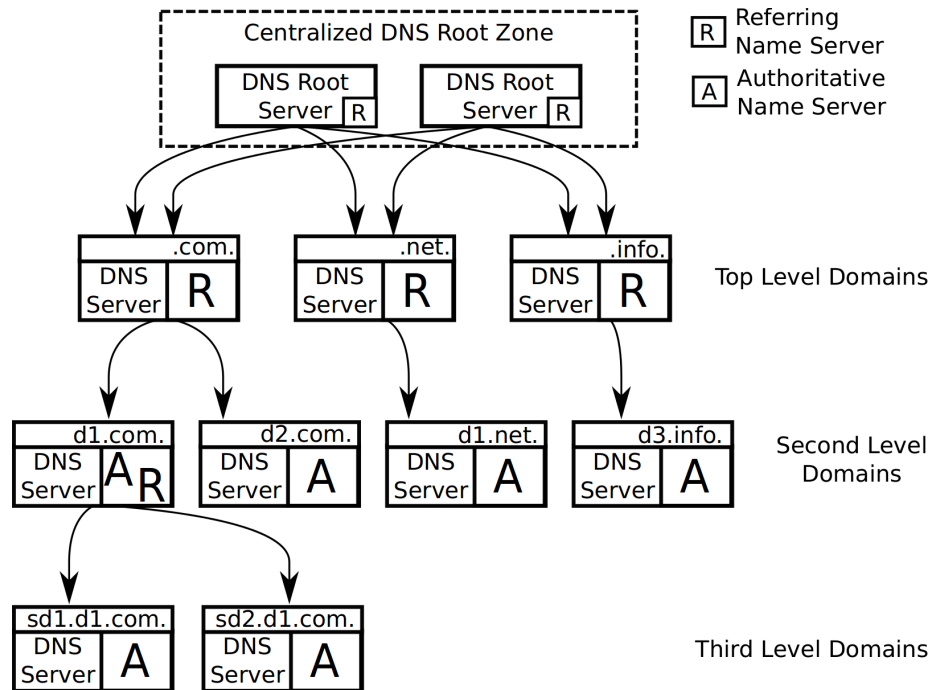
Figure 3: Example DNS hierarchy

3.2 Domain Name System Basics

The Domain Name System (DNS) plays a major role in today's Internet architecture and is an essential component of any Web based content delivery architecture. DNS relies on a distributed database with a hierarchical structure. The root zone of the DNS system is centrally administered and serves its *zone* information via a collection of *root servers*. The root servers delegate responsibility for specific parts (zones) of the hierarchy to other *name servers*, which may in turn delegate the responsibility to other name servers. At the end, each site is responsible for its own *domain* and maintains its own database containing its information and operates an *authoritative* name server.

The whole DNS database is usually queried by end-hosts using a local name server called *caching resolver*. If this name server receives a query for a domain that it does not know about, it fetches this information from another name server. If the server does not know how to contact the authoritative server for a zone, it will query a root server[1]. The root server will *refer* the resolver to another server that is authoritative for the domain that is immediately below the root and of which the zone is a part. The resolver will then query this server, and so forth, stepping down the tree from the root to the desired zone.

To illustrate this process, Figure 3 show a sample DNS hierarchy. In this case, the root of the DNS name space, denoted with a '.', is hosted on two *DNS root servers*. Both servers are under one administrative control, and both can refer a request to any of the top level domain servers. Here, three domains exist, i.e., *.com*, *.net* and *.info*. Again, these name servers refer to the second level domains. Since the domain name are

---

[1]The first query can go to some authoritative server below the root if there exists cached information.

concatenated together as the hierarchy is traversed, the domains that are now possible are *d1.com.*, *d2.com.*, *d1.net.* and *d3.info.*. At this point, the second level domains d1.net. and d3.info have reached their authoritative resolver. For example, a query to the name server of *.d3* for *www.d3.info* is answered authoritatively from there. Note that the name servers for the second level domains are operated by independent entities that know nothing of each other. Thus, the database is distributed, while each party is responsible for its own zone. Finally, the name server of *.d1.com.* has a dual role. While it is referring the subdomains *.sd1.d1.com.* and *.sd2.d1.com.* to other name servers, it also answers queries for other names in its name space authoritatively. This means that a query for *www.d1.com.* is directly answered, while a query for *www.sd1.d1.com* is referred to the name server responsible for *.sd1.d1.com*.

For efficiency reasons DNS relies heavily on caching [96, 3]. All information that a name server delivers to a resolver is cached for a duration specified in the time-to-live (TTL) field of the *resource records* (RR). Caching today is usually also performed on end-hosts by the operating system's *stub resolver*, as well as applications, e.g., web browsers.

DNS Today. When DNS was introduced in 1983, its sole purpose was to resolve host names into IP addresses in a more scalable fashion than the until then used `hosts` file. Since then a number of features and new uses have found their way into the now omnipresent DNS. In addition to the increasing complexity within the DNS protocol itself [176], new and oftentimes unforeseen (ab)uses have been established. Paul Vixie gives an overview in [177]. The most important points of critique are as follows:

CDI load balancing: Content delivery infrastructures set short TTLs on their DNS answers to allow for short reaction times to shifting loads. Short TTLs impede on cacheability and therefore increase the load on the whole DNS system. In addition, CDIs tailor their reply for the IP address of the requesting resolver using the assumption that the DNS resolver is close to the client originating the request. It has been shown in the past that this assumption is quite often wrong [119, 141, 3, 47].

NXDOMAIN catcher: Some ISPs and third party DNS providers mangle a negative reply with the NXDOMAIN status code into a positive one with the IP address of a search website under the control of the ISP. By hosting advertisements along the search results it is easily possible to increase the profit margin. While this may work to some degree for web browsing, applications relying on proper delivery of NXDOMAIN records, e.g., email, are inevitably hampered.

A third-party ecosystem around DNS has evolved over the last couple of years. Players like OpenDNS, AdvantageDNS, UltraDNS, and most recently Google offer open resolvers to anyone with different feature sets. OpenDNS Basic does NXDOMAIN catching but offers phishing and botnet protection for free. Furthermore, OpenDNS increases the service level for payment between 5 dollars a month up to several thousand dollars per year for business customers. When Google Public DNS entered the market, their highest-valued goals were to "speed up your browsing experience" and to "improve your security". To achieve both targets Google advertises an impressive list of optimizations and fine tuning [85], e.g., prefetching, load balancing with shared cache, validity checking, and nonce prepending. Google Public DNS also refrains from utilizing NXDOMAIN to make profit. From an implementation perspective, most if not all of the third-party resolvers host their DNS servers on multiple sites around the globe and use anycast to guide DNS clients to the nearest resolver.

In this open market space a user annoyed by his ISP's DNS can easily choose for cost-free third-party service. Tools such as namebench [130] might help him in choosing a well-performing one. The irony however is that a user, by choosing a different DNS than the one assigned by his ISP, will most likely undermine the traffic matrix optimizations performed by CDIs and ISPs, and can potentially even lower his quality of experience due to longer download times [3].

4 Traffic Trends: Overall

Before delving into the details of the collaborating opportunities for content providers and infrastructures we embark on giving an overview of typical characteristics of Internet traffic. We start by summarizing previous studies on how Internet traffic looks like. We consider four aspects: *(i)* The composition of the application mix, *(ii)* popular content-types, *(iii)* the distribution of traffic over the course of a day, and *(iv)* the distribution of connection sizes.

4.1 Application Mix

One constant in the Internet during the last 10 years has been its steady growth by more than 50 % each year [137, 80]. Initially, protocols such as FTP, SMTP, and NNTP were popular. Then, in about 1994, HTTP entered into the picture. Until 2000, P2P protocols such as Napster and Gnutella became popular but were later overtaken by eDonkey and BitTorrent. However, the traffic mix has undergone substantial changes. Therefore, we now revisit previously reported results regarding the application mix of Internet traffic. For this purpose we rely on various studies that report on the application mix between 2007 and 2009 from different vantage points:

- The study by Maier et al. [118], which is based on a subset of the traces studied in Section 11.2.5. It was presented at IMC '09.
- Two studies by ipoque [157], which report on different regions in the world (Germany and Middle East). These studies are available for download after registration via a Web form.
- The Arbor report [107] on the ATLAS Internet Observatory presented at a recent NANOG[2] meeting.
- The Sandvine report on "Global Broadband Phenomena" [154].

In order to compare the results we have to summarize and unify the traffic categories as each study uses their own nomenclature (see Figure 4). For this purpose we use the following seven categories:

Web. All HTTP traffic including One-Click-Hosters (OCHs or Direct Download Providers) but excluding video and audio streaming over HTTP (i.e., Flash-Video).

Streaming. All types of streaming in the Internet including streaming over HTTP, RTP, RTSP, RTMP, ShoutCast, etc.

Usenet. The article reading and posting system that evolved from UUnet and which uses NNTP as protocol.

BitTorrent/P2P. The popular P2P-protocol BitTorrent and all other P2P traffic that is not eDonkey. Note, that the P2P traffic that is not BitTorrent or eDonkey only adds a tiny fraction. Moreover, this category represents all P2P traffic if the study no further subdivides P2P traffic. This is the case for Arbor [107] and Sandvine [154]. Note as well, that the Arbor study [107] reports a table with traffic shares, stating 0.95 % for P2P. This table is annotated with the comment that P2P is more likely to account for 18 % based on payload inspection of a limited data subset.

eDonkey. Another P2P protocol, if reported.

Other/known. Other identified traffic, for details we refer to the corresponding studies.

Unclassified. Traffic that has not been classified. Note, that the Sandvine [154] study does not mention unclassified traffic, which either implies a minute fraction or that it is missing in the plot.
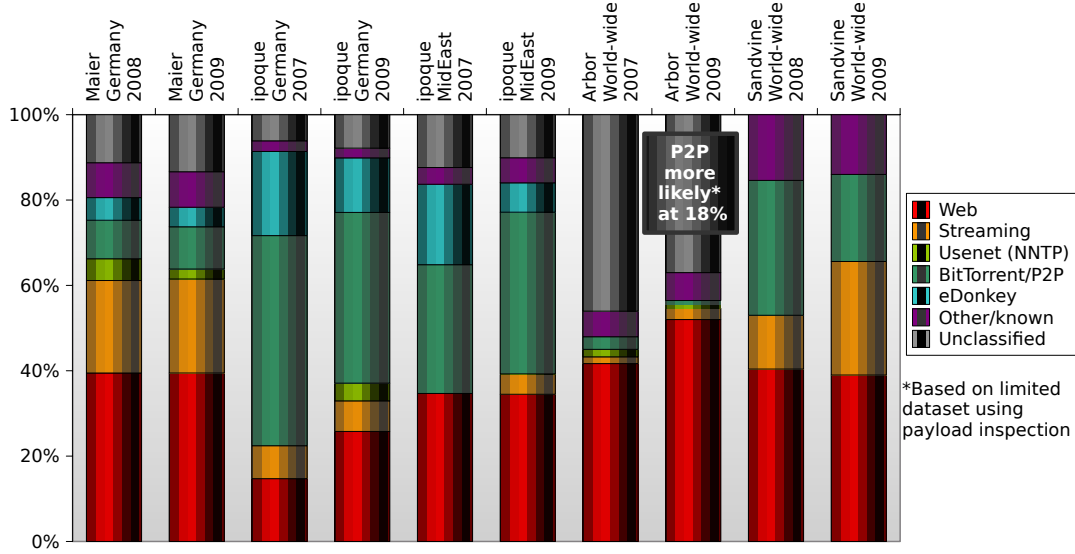
Figure 4: Barplot of the application mix in the Internet (unified categories) for different years, different regions according to several sources [118, 157, 107, 154]. (BitTorrent/P2P contains all P2P except eDonkey.)

Looking at these statistics we find that all studies report a significant fraction of Web traffic. Indeed, Web is dominant (> 50 %) in most studies, followed by P2P and streaming. It is noteworthy that Usenet is responsible for a non-negligible fraction in several studies. This is surprising and a good example for the importance of revisiting the application mix periodically in order to identify new trends.

In terms of P2P protocol distribution Figure 4 shows that BitTorrent is dominating and the shares of eDonkey are decreasing. Thus, we note that the results of Plissonneau et al. [144] who observed 91 % of the P2P traffic is due to eDonkey in 2004 are no longer applicable. Indeed, the popularity among P2P protocols swapped in favor of BitTorrent. We can also see a general trend: P2P is declining according to all studies. This is also supported by the results of Anderson [21]. He points out that this decline comes with an increase in video streaming. Moreover, most of the studies pointed out that currently One-Click-Hoster (e.g., Rapidshare or MegaUpload) are as important for file-sharing as P2P systems.

Of course there are also trends that do not impact the application mix, for example Online Social Networks (OSNs) such as Facebook. This is due to the fact that OSNs use HTTP and they do not transport large videos, but profile elements. Nevertheless, OSNs are not unimportant given the huge number of OSN users world-wide.

4.2 Content-types in the Internet

Next, we turn to the popularity of content-types in the Internet. Again, we leverage several data sources, namely Maier et al. [118], ipoque [157], and Erman et al. [58]. Once more we unify the categories and present results for contents transferred via BitTorrent, eDonkey, and HTTP. See Figure 5 for a summary.

We see that videos are the most popular content in P2P systems (BitTorrent and eDonkey). Even in HTTP videos account for more traffic than any other category. Although HTTP was designed to transfer

---

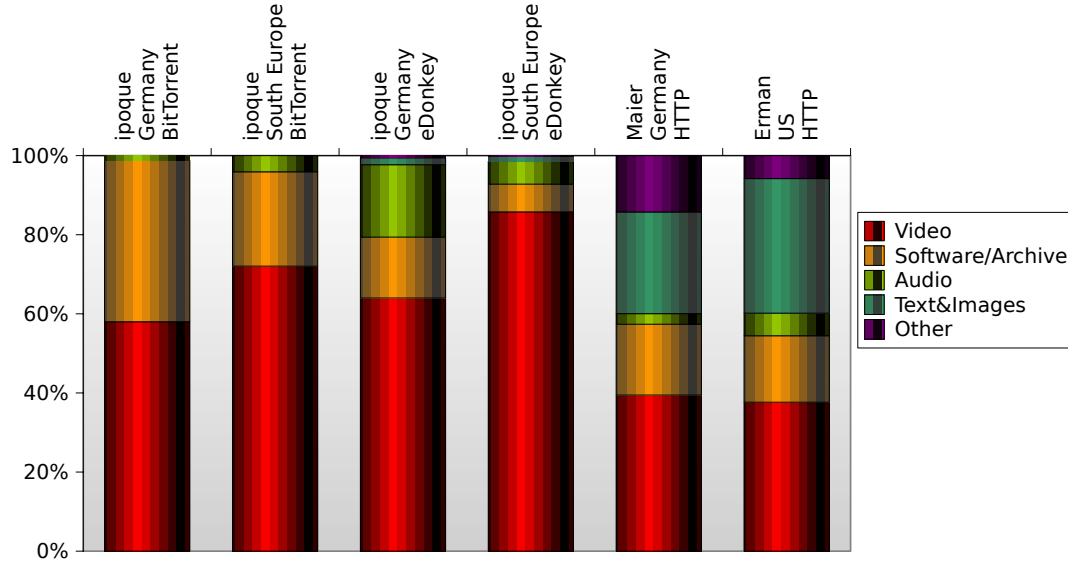[2]NANOG is the North American Network Operators Group.

Figure 5: Barplot of content-type popularity in the Internet (unified categories) for different protocols, different regions according to several sources [118, 157, 58].

Web pages (text, e.g., HTML, XML, CSS, JavaScript, and image files) these contribute less than a third of the total HTTP volume.

Overall, a significant fraction of software and archives is noticeable. According to Maier et al. [118] almost all videos are in flash-video format and are served by video portals such as YouTube. Similarly, almost all archives are served by One-Click-Hosters. This is confirmed by the results of Erman et al. [58].

Shifts in the popularity of content-types can be another indicator of new trends. For example, there have been almost no flash-videos before the breakthrough of YouTube.

4.3 Time-of-day Effects

In order to understand when people are active in the Internet we show time-of-day usage plots of link utilization from Maier et al. [118] in Figure 6, and aggregated traffic volume Sandvine [154] in Figure 7.

In general, we observe a peak utilization at prime-time around 8 pm and a daily low between 2 am and 4 am. As the data sets of all these studies are primarily collected from residential networks, it not surprising that they all show similar characteristics. The peak usage in the evening hours can easily be explained by the fact that people are usually not at home during business hours. Rising demands just before lunch and in the afternoon may be due to children returning home from school.

5 Traffic Trends: Content Server Diversity

So far we have highlighted that the Web and P2P protocols are responsible for a major share of the Internet traffic. However, we have not yet explored if all content is equally popular or if a few content providers dominate. This is the goal of this section.
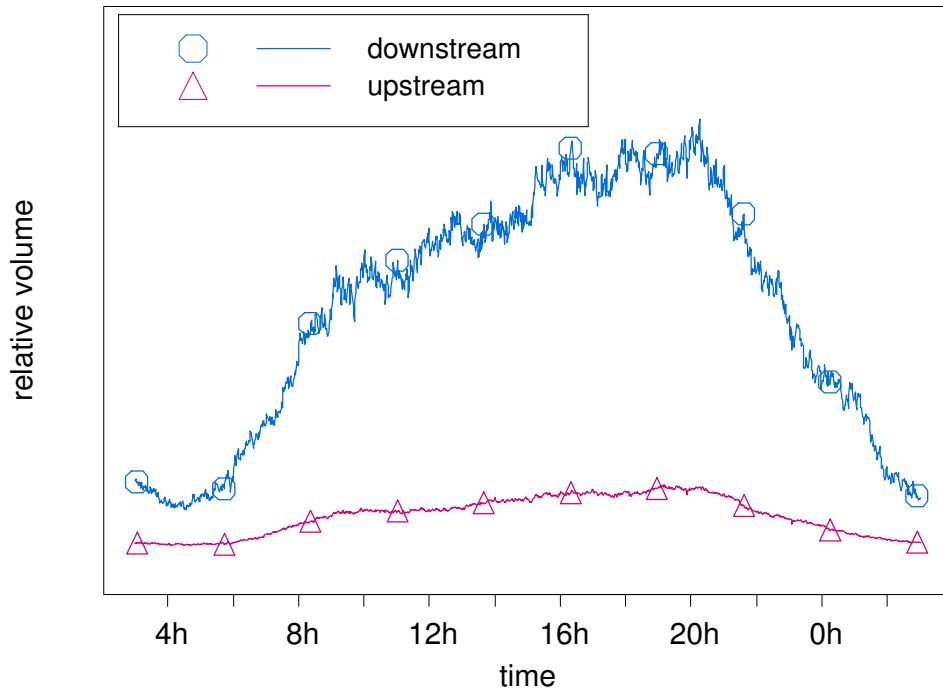
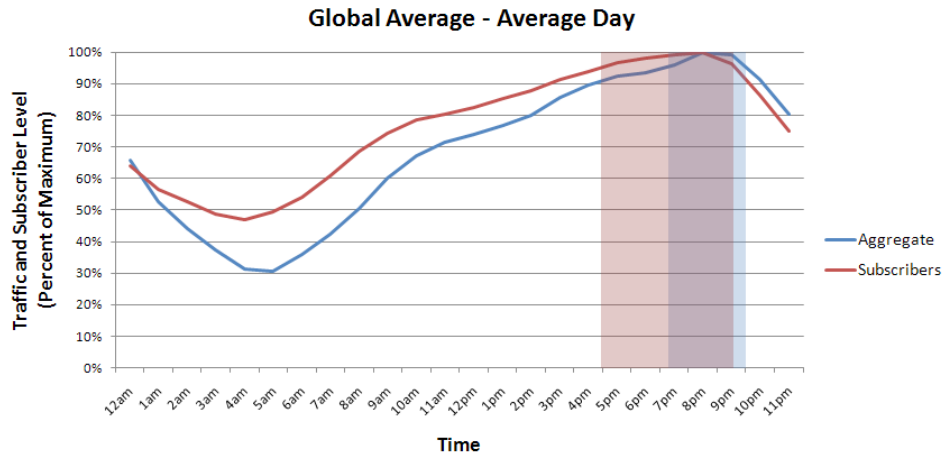Figure 6: Timeseries of link utilization from Maier et al. [118]

Figure 7: Timeseries of traffic volume from Sandvine [154]

Our evaluation methodology relies on packet level traces from a large European ISP. We analyze them towards identifying CDI infrastructures and their behavior as seen by an ISP. Here, we find that CDIs rely on

Name	Type	Start date	Dur.	Size	Application Volume
MAR10	packet	04 Mar'10 2am	24 h	>5 TB	> 3 TB HTTP, > 5 GB DNS
HTTP-14d	log file	09 Sep'09 3am	14 d	> 200 GB	corresponds to > 40 TB HTTP
DNS-5d	packet	24 Feb'10 4pm	5 d	>25 GB	> 25 GB DNS

Table 1: Summaries of anonymized traces from a European ISP

the domain Name System (DNS) for their operation. Thus, we focus our analysis on the DNS infrastructure in order to find the server deployment, mapping and operational behavior of CDIs. Based on these observations, we develop classification methods to detect CDI infrastructures and perform a first potential analysis on the impact of CDI operation when basic ISP knowledge is available.

5.1 Residential ISP Traces

We base our study on three sets of anonymized packet-level observations of residential DSL connections collected at aggregation points within a large European ISP. Our monitor, using Endace monitoring cards, allows us to observe the traffic of more than 20,000 DSL lines to the Internet. The data anonymization, classification, as well as application protocol specific header extraction and anonymization is performed immediately on the secured measurement infrastructure using the Bro NIDS [143] with dynamic protocol detection (DPD) [55].

We use an anonymized 24 h packet trace collected in March 2010 (MAR10) for detailed analysis of the protocol behavior. For studying longer term trends, we used Bro's online analysis capabilities to collect an anonymized protocol specific trace summary (HTTP-14d) spanning 2 weeks. Additionally, we collected an anonymized 5 day DNS trace (DNS-5d) in February 2010 to achieve a better understanding of how host-names are resolved by different sites. Due to the amount of traffic at our vantage point and the resource intensive analysis, we gathered the online trace summaries one at a time. Table 1 summarizes the characteristics of the traces, including their start, duration, size, and protocol volume. It is not possible to determine the exact application mix for the protocol specific traces, as we only focus on the specific protocol. However, we use full traces to cross check the general application mix evolution.

With regards to the application mix, recall Section 4, Maier et al. [118] find that HTTP, BitTorrent, and eDonkey each contribute a significant amount of traffic, see Table 1. In MAR10 HTTP alone contributes almost 60 % of the overall traffic at our vantage point, BitTorrent and eDonkey contribute more than 10 %. Recall that similar protocol distributions have been observed at different times and at other locations of the same ISP, see Figure 4 summarizes the results. Note that almost all streaming is done via the Web on top of HTTP. Therefore, we conclude that currently HTTP is the dominant service and P2P is still responsible for at least 15% of the traffic.

Analyzing HTTP-14d, we find more than 1.2 billion HTTP requests, or 89 million requests per day on average. This is consistent with 95 million requests in 24 hours in MAR10. The advantage of using click stream data from a large set of residential users is their completeness. We are, e.g., not biased by the content offered *(i)* by a web service, *(ii)* whether sufficient users installed measurement tools such as the alexa.com toolbar, or *(iii)* whether users actually use some kind of Web proxy.

To identify the most popular web services, we focus on the most popular hosts. As expected, the distribution of host popularity by volume as well as by number of requests is highly skewed and is consistent with a Zipf-like distribution as observed in other studies [118]. The top 10,000 hosts by volume and the top 10,000 hosts by number of requests together result in roughly 17,500 hosts. This indicates that on the one hand, some hosts that are popular by volume may not be popular by number of requests and vice versa. On

the other hand, there are some hosts that are popular according to both metrics. The total activity by these hosts accounts for 88.5 % of the overall HTTP volume and more than 84 % of the HTTP requests. Assuming that the HTTP traffic volume accounts for roughly 60 % of the total traffic, similar to the observations made in September 2009 [118, 5] and in MAR10, more than 50 % of the trace's total traffic is captured by these hosts.
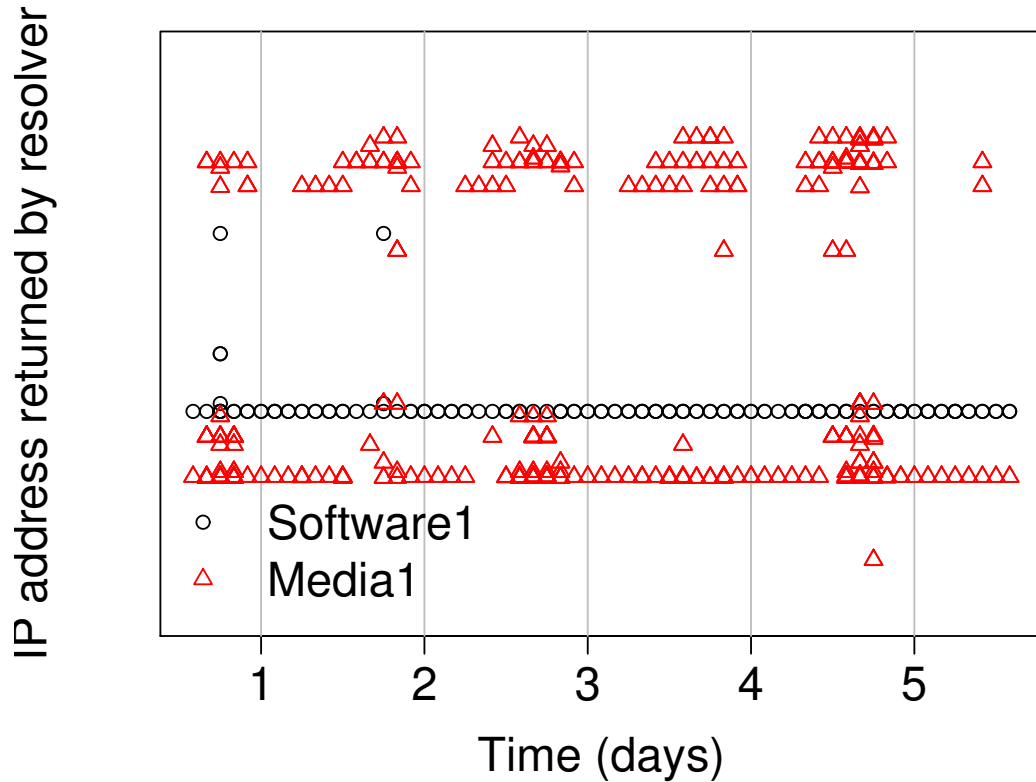

Figure 8: DNS replies for two different sites hosted on a CDI, in two-hour bins. Reprinted from [145], ©ACM, 2010. Included here by permission.

5.2 Server Diversity and DNS Load Balancing

To better understand how HTTP requests are handled and assigned to servers, we use DNS-5d to analyze the 20 most heavily queried DNS names to identify typical usage patterns. We consider only the most heavily used resolver. Figure 8 shows two of the typical patterns for two of the DNS names. It also shows how the resolved IP addresses change (y-axis) across time (x-axis) for two hostnames; respectively a software site, labeled Software1, and a media site, labeled Media1. The vertical lines annotate midnight. If two IP addresses are plotted close to each other, this indicates that the longest common prefix of the two addresses is close. We note that the hostname of Software1 is mainly resolved to a single subnet, excepting a few special cases. However, Media1 is load balanced across approximately 16 different sites. For Media1, there

appears to be one main site which is almost always available, while the remaining 15 are predominantly used during afternoon and evening peak usage hours.

These results are promising, and show that individual sites do expose a certain degree of server diversity to their users. While our trace (HTTP-14d) includes the queried hostnames, it does not include the resolved IP address, as a HTTP request header contains the hostname but not the IP address of a server. To verify the above behavior and get an up-to-date view of the DNS replies for the hostnames of our trace, we used 3 hosts within the ISP to issue DNS queries to the ISP's DNS resolver for all 17,500 hostnames repeatedly over a fourteen day measurement period starting on Tue Apr 13th 2010. During these two weeks, we received more than 16 million replies. Unless otherwise mentioned, we rely on our active DNS measurements, with augmented statistics concerning volume and requests from HTTP-14d.

5.3 Server Location Diversity

Our analysis of hostnames and their assignment to servers in section 5.2 has shown that content can be served by multiple servers in different locations. In fact, many domains use the service of a *Content Delivery Infrastructure* (CDI), which can be seen during the DNS resolution progress: The original domain name is mapped to the domain of a CDI, which then answers requests on behalf of the requested domain name from one of its caches [169]. Almost all CDIs rely on a distributed infrastructure to handle the expected load, load spikes, flash crowds, and special events. Additionally, this introduces needed redundancy and fail over configurations in their services. Among the most studied CDI' are Content Distribution Networks (CDNs), such as Akamai [112, 169, 90], and Content Delivery Platforms (CDPs), such as Google [105] and their YouTube service [37].

The DNS server can choose to return one or more server IP addresses based on the domain name in the request and the IP address of the requesting DNS resolver. For example, it may use a geo-location database [121] to localize the region of the DNS resolver, utilize BGP data to identify the ISP, create a topology map derived via traceroutes, or any combination of these and other topological and geographic localization techniques. A DNS server has, in principle, two methods for load balancing across multiple servers:

MultQuery: Can return multiple IP addresses within a single DNS response
CrossQuery: Can return different IP addresses for repeated queries and thus perform DNS redirection.

In our active DNS measurements, we found that often a mixture of MultQuery and CrossQuery is being used in practice. Furthermore, we used the measurement results to *(i)* map hostnames to sets of IP addresses and *(ii)* check the IP address diversity of these sets for a better understanding of server diversity and their location. We achieved this by aggregating the returned IP addresses into subnets based on BGP information obtained from within the ISP. This allows for detailed information about the different locations within the ISP, while giving an aggregated view of subnets reachable via peering links.

Another issue stems from the fact that the IP address returned by the CDI depends on the IP address of the ISP DNS resolver [3, 141, 169]. Due to this, we used the DNS resolver of the ISP of our vantage point as well as external DNS resolvers (see section 5.3). The former reflects the experience of most of the clients at our vantage point[3]. The latter lets us discover additional diversity as well as understand the preference of the CDI for this specific ISP.

---

[3]We verify using the traces that more than 95 % of the clients use the ISP's DNS resolver as their default one.

Prevalence of MultQuery. We start our analysis by checking the prevalence of the first form of DNS based load balancing, MultQuery. Figure 9 shows a CCDF plot of the average number of IP addresses (top) and subnets (bottom) per DNS reply. In addition, we included the same data normalized by traffic volume and number of requests.

A first observation is that the number of returned IP addresses per request is rather small. The median is 1, the average is 1.3 and even the 0.9 percentile is 2. We note that even when an answer yields multiple IP addresses, the majority of them are from the same subnet. Therefore, the diversity decreases even further if we aggregate to subnets. From a network perspective, this implies that there is not much choice, neither for the ISP nor for the user, regarding where to download the content from. Both are limited to the information provided by the DNS server. However, when we normalize the hosts by their respective popularity, we see a significant improvement. More than 29% of the volume and 19% of requests have a choice among at least 2 IP addresses.

Prevalence of CrossQuery. Next, we check how prevalent CrossQuery, the second form of DNS based load balancing is. Since CrossQuery returns different IP addresses for repeated queries, its potential contribution to server diversity can only be studied by aggregating across time. The lines labeled `Full Domain Name` in Figures 10 and 11 capture this case.

We find that more than 50 % of the volume or requests can be served by more than one IP address. Similarly, there is choice between at least two subnets over 40 % of the time across both metrics, see Figure 11. This indicates that there is significant potential for the ISP to bias the location preference of the CDI.
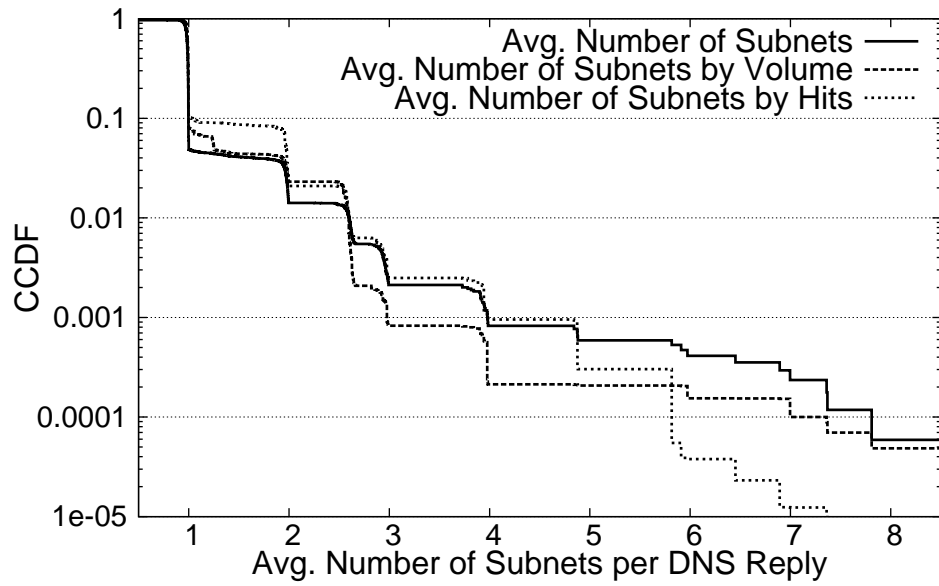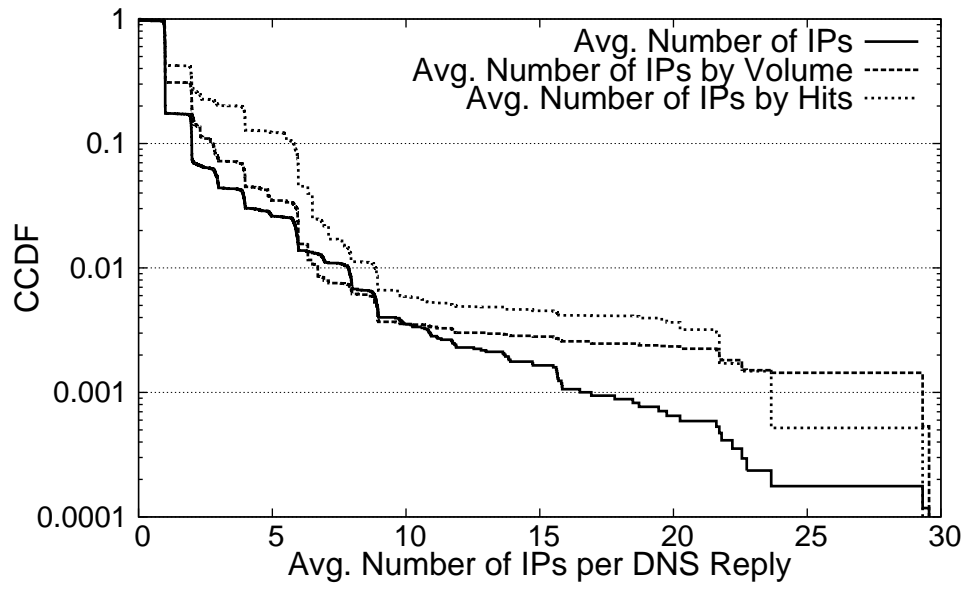
Figure 9: CCDF of mean # of IPs (top) and subnets (bottom) per DNS reply for the ISPs DNS resolver. Reprinted from [145], ©ACM, 2010. Included here by permission.
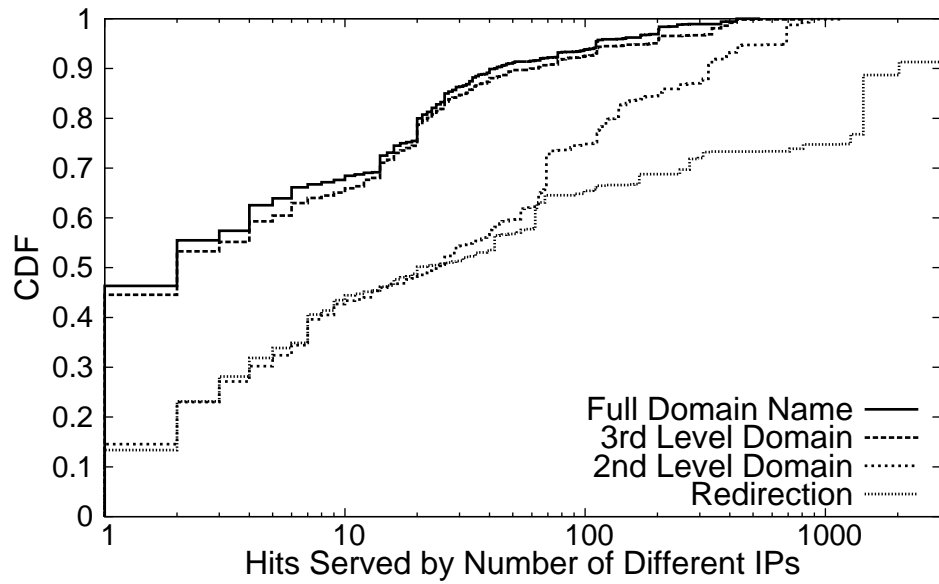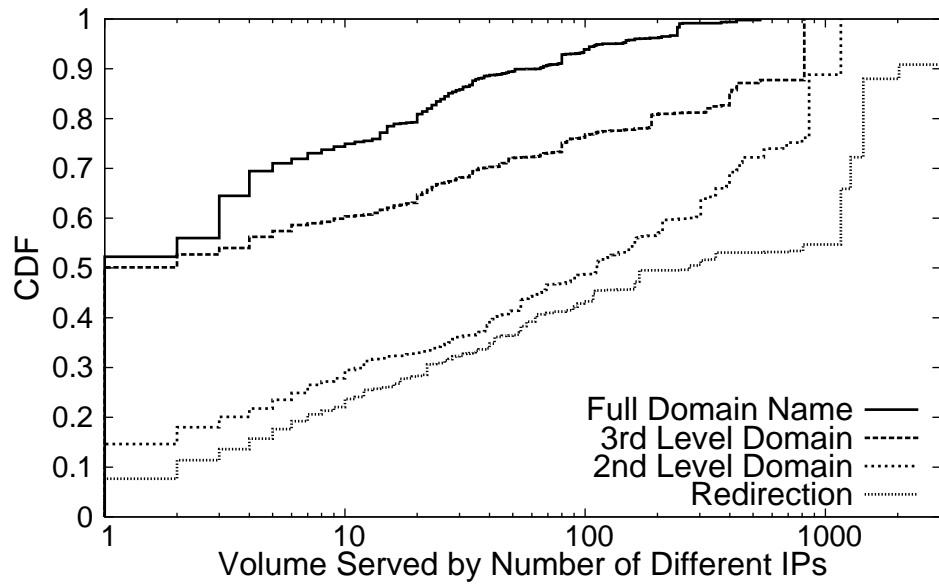
Figure 10: CDF of # of IPs for the ISP DNS resolver normalized by traffic volume (top) and requests (bottom) including aggregation on domain levels. (Logarithmic x-axis.) Reprinted from [145], ©ACM, 2010. Included here by permission.

Figure 11: CDF of # of subnets for ISP DNS resolver normalized by traffic volume (top) and by requests (bottom) including aggregation on domain levels. (Logarithmic x-axis.) Reprinted from [145], ©ACM, 2010. Included here by permission.
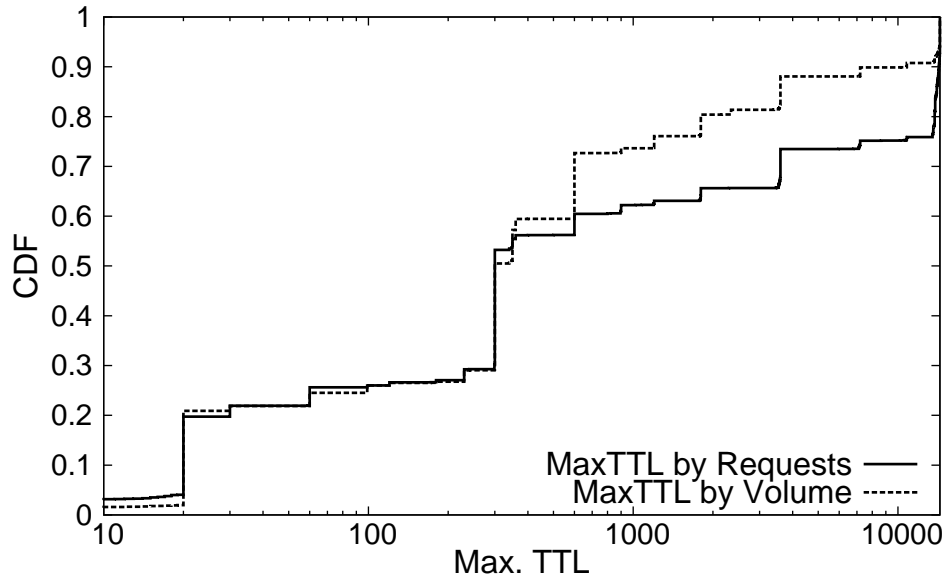
Figure 12: CDF of DNS TTL value by traffic volume and by number of requests. Reprinted from [145], ©ACM, 2010. Included here by permission.

Subdomain Aggregation. Since some CDIs only use subdomains as hints about the context of the requested URLs or the requested services, we accumulate the answers further regarding the 2nd and 3rd part of the domain names 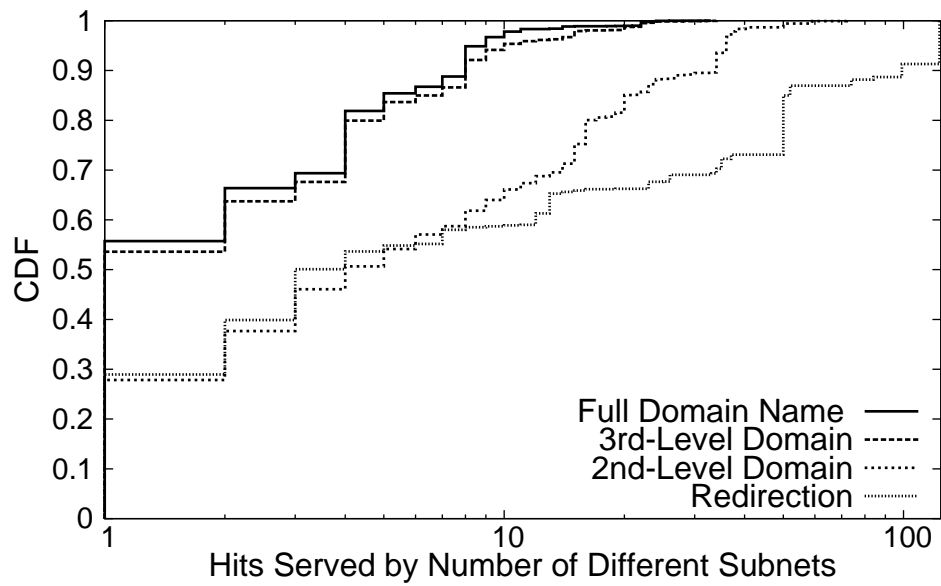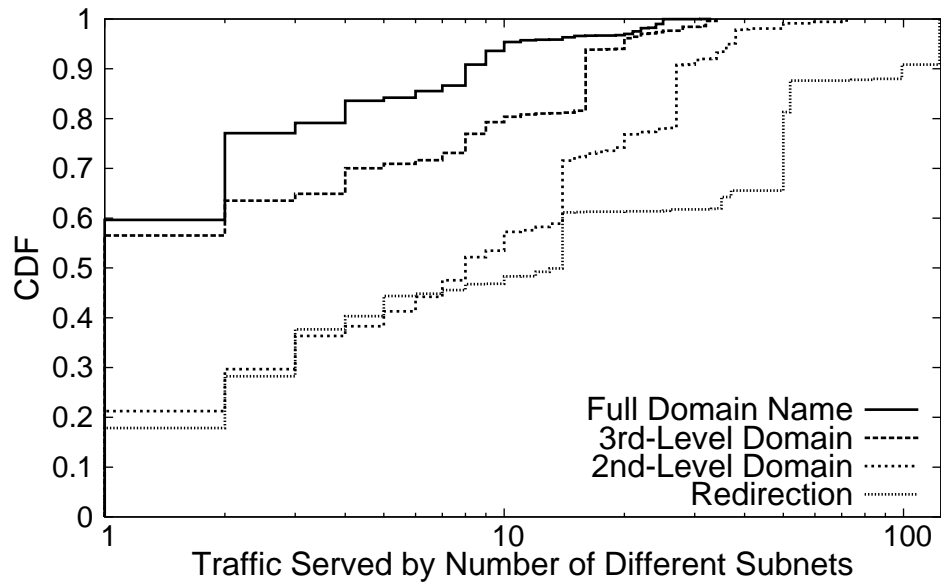of the hosts, see Figures 10 and 11 at the respective data series called `3rd Level Domain` and `2nd Level Domain`. For example, we might accumulate the IP addresses from DNS replies for dl1.example.org and dl2.example.org for the statistics on the 2nd level domain, but not the third level domain.

This is a feasible approach, since many hosts respond to all requests that belong to a subset of the subnets returned when accumulating by the second-level domain of DNS resolver answer, including recursive requests and redirections. This behavior was verified with active measurements in [145]. We find that at least two major CDIs, a streaming provider and a One-Click Hoster, serve requested content from servers that match in their second level domain.

We note that the accumulation by third-level domain, and especially by second level domain significantly increases the number of observed subnets per request both normalized by requests as well as by volume. The number of returned subnets further increases when accumulating to the second-level domain of DNS resolver answer. Studying our traces in more detail, we find that this is due to the substantial traffic volume and number of requests that are served by CDIs, some of which are highly distributed within ISPs or located

325

Metric	ISP DNS		OpenDNS		GoogleDNS	
	observed	potential	observed	potential	observed	potential
IPs	12.3 %	24.2 %	5.8 %	16.0 %	6.0 %	9.7 %
requests	14.9 %	33.2 %	4.7 %	18.8 %	4.8 %	6.4 %
volume	23.4 %	50.0 %	12.0 %	27.7 %	12.3 %	13.4 %

Table 2: Traffic localization within the network by different DNS resolvers normalized by number of requests and traffic volume together with the potentially available fraction of localized traffic.

in multihomed datacenters or peer-exchange points.

Infrastructure Redirection Aggregation. Taking a closer look at the DNS replies [127], we find that some CDIs use CNAME records to map queried hostname to an A record. These A records show the same pattern as the hostnames in the previous section: the second level domain is identical. Similar to the previous approach, we can aggregate by these A records.

Turning our attention to the implications of the proposed aggregation schemes, we notice the available diversity increases tremendously. More than 50% of the hits and 70% of the bytes can be served by more than 20 servers. With regards to subnets, the diversity decreases slightly. Nevertheless, more than 5 subnets are available for 45 % of the hits and 55% of the bytes.

If we consider aggregation periods in the order of tens of minutes, the numbers do not decrease by much. The reason that most of the diversity is observable even over these short aggregation time periods, is that the typical TTL, see Figure 12, is rather short with a mean of $2,100$ seconds and an median of 300 seconds normalized by volume. When weighted by requests, the mean/median is $4,100/300$ seconds.

Alternative DNS Resolvers. So far we have only considered the effect of content diversity when the ISP DNS resolver is used. To understand how much the DNS load balancing deployed by a CDI is biased by the queried DNS resolver, we repeat the experiment from Section 5.2 using two other DNS resolvers. In particular, we pick the next most popular DNS resolvers found in our traces: GoogleDNS and OpenDNS. Both are third-party resolvers with a global footprint and utilize DNS anycast.

Comparing the results, we find that we attain more IP address diversity and subnet diversity when using the ISP DNS resolver. This is mainly due to the fact that CDIs select the supplied caches based on the source IP address of the querying DNS resolver. Since the CDIs are no longer able to map the request to the AS it originates from, but rather to AS the DNS resolver belongs to, the server selection by the CDI cannot optimize for the location of the DNS client.

A possible solution to the problem is the EDNS-Client-Subnet extension [46], an extension that utilizes the EDNS0 option field that is used today by DNS Security Extensions (DNSSEC). A recent study [140] showed that the user-to-server allocation can be significantly improved as well as the end-to-end performance for the client. On the other hand, this requires that all the involved resolvers and authoritative servers in ISPs, CDNs, third parties that maintain resolvers and authoritative servers, e.g., GoogleDNS, OpenDNS, have to support EDNS-Client-Subnet extension.

5.4 Impact on Traffic Localization

Analyzing the three active DNS measurements from the ISP, OpenDNS as well as Google DNS resolver, we find that a significant part of the requests that could have been in principle served by sources within the ISP are directed towards servers that are outside of the ISP. However, before tackling this issue, we need to understand what fraction of the traffic may be served by IP addresses within the ISP's network and what fraction is served by IP addresses outside of the AS. To this end, we analyze each of the three active DNS traces separately. For each trace, we start by classifying all DNS replies regarding the `redirection` aggregation described in Section 5.3 and account the volume (or hits) evenly to each of the IP addresses. Next, we classify the IP addresses in two groups - inside and outside of the ISP network. Table 2 summarizes the results of this aggregation regarding the traffic and hits that were kept inside the ISP's network in the columns labeled `observed`.

Turning to the results, we find that there is hardly any difference between those clients that use the external DNS resolvers, i.e., GoogleDNS or OpenDNS. Of the returned IP addresses, less than 6 % are within the AS. When weighted by number of requests, this does not change much. However, when normalizing by volume, about 12 % of the traffic stays within the AS. In contrast, clients that use the ISP's DNS resolver fare better: almost a quarter of the traffic volume is served from servers within the AS. Normalized by requests, we see a three fold increase, and normalized by hits or volume, roughly a two fold increase over using external DNS resolvers. Among the reasons for the "bad" performance of external DNS resolvers is that some CDIs may always return IP addresses outside the ISP, despite the fact that many of its servers are deployed within the ISP. The reason behind this is that the CDIs cannot map the DNS resolver to the AS anymore, and thus are unaware of the origin of the request. This explains the substantial difference and highlights on the one hand the effectiveness of the CDI optimization, but also points out its limits. As such, it is not surprising that there are efforts under way within the IETF to include the source IP addresses of the DNS client in the DNS requests [47].

However, one can ask if the CDI utilizes the full potential of traffic localization on an AS level. For this, we check the potential of traffic localization, by changing the volume (or hit) distribution from even to greedy. Thus, as soon as we observe at least one IP address inside the ISP's network, we count all traffic for the entire aggregation to be internal. Table 2 shows the results in the columns labeled `potential` for all three DNS traces. Note the substantial differences. Our results indicate that a gain of more than a factor of two can be achieved. Furthermore, up to 50 % of the traffic can be delivered from servers within the ISP rather than only 23.4 %. This may not only in itself result in a substantial reduction of costs for the ISP, but it also points out the potential of collaboration between CDIs and ISPs. While the increase is noticeable it is nowhere near that of the ISP's DNS resolver. The potential benefit when relying on GoogleDNS is rather small. A deeper study on our results unveils that content served by highly distributed and redundant infrastructures can be localized the most.

5.5 Summary

We find that HTTP is again the dominant traffic source, while the prevalence of P2P traffic decreases. Since most CDIs rely on distributed infrastructure, we not only observe significant server location diversity but also significant path diversity for accessing HTTP based content. Indeed, there is the potential to bias roughly half of the overall traffic by redirecting queries to different content servers.

More precisely, we estimate that around 70 % of the HTTP traffic in a big European ISP can be redirected when taking advantage of the diversity due to MultQuery, CrossQuery and hostname aggregation. Furthermore, we show that current CDI optimizations that approximate the location of end-users based on the location of the local DNS resolvers are more effective than those based on the location of third-party

resolvers. Finally, we show that the traffic localization potential within the above mentioned ISP is very high especially when the ISP DNS resolver is utilized.

6 Content Delivery: An Overview

While content may be seen as king not all content is equally popular among users. Indeed, content popularity often follows "Zipf's law". If the popularity of elements as function of the rank is consistent with a power-law distribution it is referred to as Zipf's-like (see [187, 126] and references therein). The rank is determined by the number of occurrence of an element, where a low rank index refers to a popular element. Not surprisingly Zipf's law does not only apply to the popularity of content but also quite a number of different quantities in Internet traffic, including the popularity of Web pages [33, 160], traffic demands [59, 62, 184, 178, 44], as well as interdomain Web traffic demands [65]. Thus, while some content can be served by a single server most content, namely the popular content, can only be served if it is highly replicated across multiple servers. Thus, one of the main challenges in content delivery is **server selection**. Server selection means identifying a specific server from which the request for content by a user is satisfied.

Content delivery and the network infrastructure interact mostly through content source selection, often called server selection. Here, it does not matter whether the source is a server pushing content through HTTP or from a peer in a P2P network. In the case of HTTP, the domain name system (DNS) is the preferred mechanism for performing server selection. In the case of P2P, peer selection strategies drive where the content is obtained from and how, e.g., when the content is cut into chunks.

To direct users to appropriate servers, CDIs rely extensively on the Domain Name System (DNS). We describe this and other server selection mechanisms in detail later in this section. The CDI chooses a server based on several metrics. Criteria for server selection include the IP address of the end-user's DNS resolver, the availability of the server, the proximity of the server to the resolver, and the monetary cost of delivering the content. Note that the server selection does not know the client IP address or network location, it only knows the IP address of the DNS resolver the end-user contacted. A recent study [3] showed that sometimes the end-user is not close to the resolver. To improve the mapping of end-users to servers, the client-IP eDNS extension [47] has been recently proposed.

In P2P systems peers can choose among all other peers to download content from but only if the have the desired content. Thus the problem of getting content in a P2P system is actually two-fold: first the user needs to find the content and once it knows of possible peers it can download the content from, it needs to connect to some of them to get the desired content. In P2P systems the content lookup is realized in many different ways. Some P2P network, called structured P2P, implement a distributed lookup system most often referred to as distributed hash table (DHT). Other P2P systems, called unstructured P2P, like Gnutella, flood search request into the network. Some systems rely on a partial centralized infrastructure to obtain content information. We discuss the different approaches in P2P systems in more detail in section 6.2.

Before we can discuss all the various options on how content delivery can be improved in the current Internet we give a short overview how a typical Content Distribution Network operates.

6.1 Content Delivery Networks

Recall content is king in the current Internet and content is typically first placed on the Web site of the content producer, the original Web servers. Content Delivery Infrastructures (CDIs) (see, e.g., [91, 52, 78, 29, 95, 104, 155]) are designed to reduce the load on origin servers and at the same time improve performance for the user. Most CDIs have a large set of servers deployed throughout the Internet and cache the content of the original publisher at these servers. Therefore another view of CDIs is that they provide reverse proxy
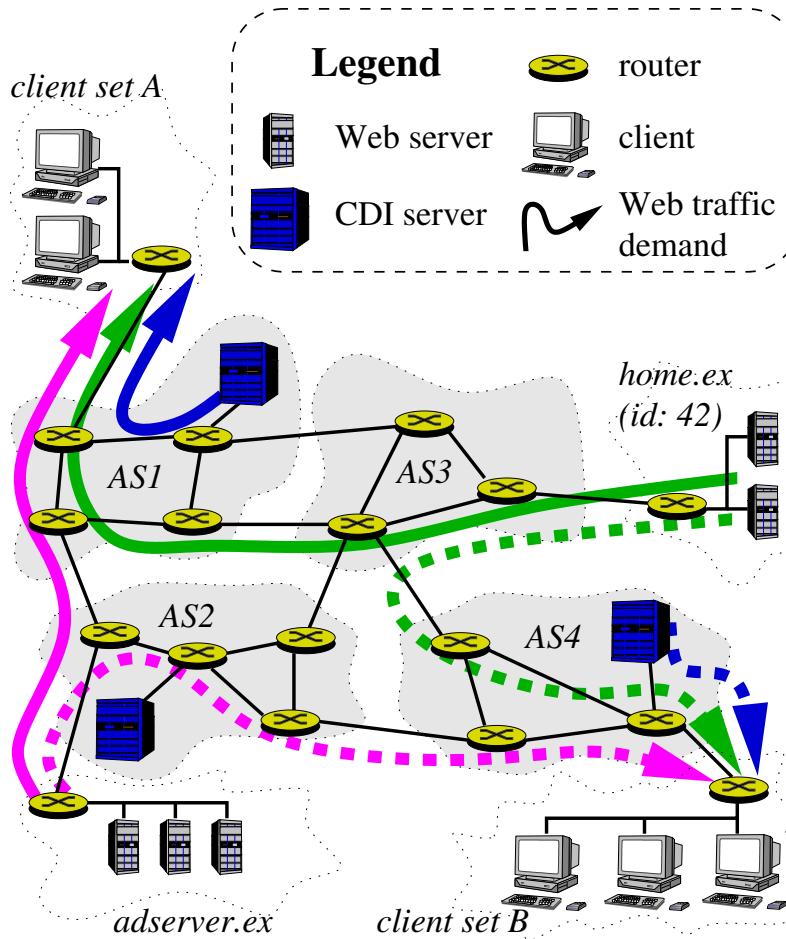
Figure 13: Example of CDI deployment and traffic flows (Web traffic demands). Reprinted from [64], ©ACM, 2004. Included here by permission.

services for content providers, the publishers. In order to take advantage of their distributed infrastructure, requests for data are redirected to the "closest" cache server. Intelligent redirection can reduce network latency and load (and therefore network congestion) improving response time. CDIs differ in their approach to redirecting traffic. Some (such as Akamai [135]), use DNS to translate the hostname of a page request into the IP address of an appropriate server. This translation may consider the location of the client, the location of the server, the connectivity of the client to the server, the load on the server, and other performance and cost based criteria.

An example that shows how the CDI infrastructure is embedded in the Internet architecture is shown in Figure 13. Recall, the Internet is divided into a collection of autonomous systems (ASes). Each AS is managed by an Internet Service Provider (ISP), who operates a backbone network that provides connectivity to clients and to other ISPs. Figure 13 shows four ASes, numbered 1–4, whose backbones consist of three

http://home.ex/index.htm

URL: cdi .ex/ex1.gif
Referrer: home.ex/index.htm

**This is only
an example**

URL: home.ex/ex2.gif
Referrer: home.ex/index.htm

URL: adserver.ex/ex3.gif
Referrer: home.ex/index.htm

URL: cdi .ex/ex4.jpg
Referrer: home.ex/index.htm
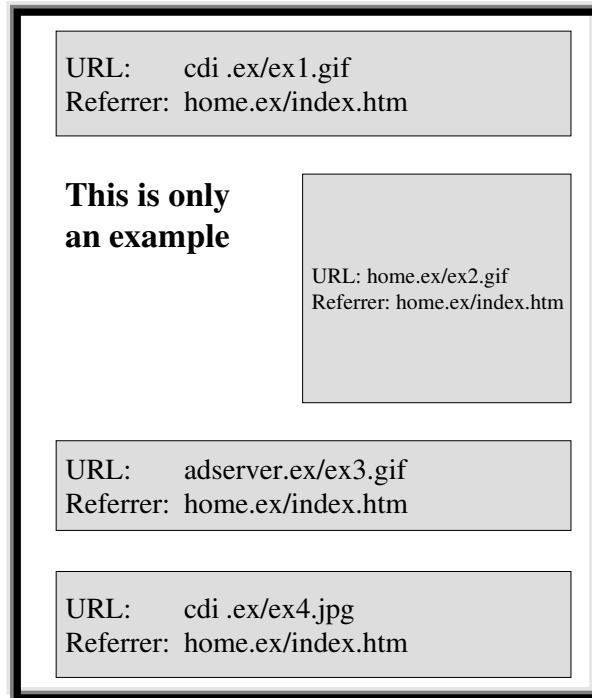
Figure 14: Example Web page with some CDI content. Reprinted from [64], ©ACM, 2004. Included here by permission.

routers each, two Web site publishers, home.ex and adserver.ex, and two sets of clients. The publisher home.ex is connected to AS 3 while the publisher adserver.ex is connected to AS 2. A set of clients is connected to AS 1, another to AS 4.

The location of the CDI's servers differ from CDI to CDI and depends on contractual agreements between the CDI and the individual ISPs. In some instances, the CDI servers are deployed within the data centers of the ISP and therefore belong to the same AS, like AS 1, 2, 4 in Figure 13. Clients of the ISP (end-users) are typically served by these servers in the same AS. With other ISPs, the CDI may have a private peering agreement that allows the CDI to serve requests from the ISPs clients via a direct connection between the CDI and the AS. The CDI may also co-locate servers with the ISP's clients, e.g., on university campuses. With other ISPs there may be no relationship with the CDI, and the traffic to the ISP's clients is routed via another AS.

330

Let us consider the steps that are necessary to download the Web page shown in Figure 14. This page consists of one main page located at home.ex/index.htm and four embedded objects. The publisher responsible for home.ex has decided to use the services of a CDI, cdi.ex. One object (ex2.gif) of the sample page is located on the same server as the page itself (index.htm); another object (ex3.gif) is served by a company providing dynamic advertisements, adserver.ex; and objects ex1.gif and ex4.jpg are hosted by the CDI.

If a specific client from client set A in Figure 13 accesses the Web page, publisher home.ex serves the bytes for the main page and one embedded object, publisher adserver.ex serves the bytes for the object located on its servers, and the "nearest" CDI server serves the two CDI-located objects—in this case, they will be served from AS 1. In contrast, if a specific client from client set B accesses the page, the two CDI objects are delivered from a different CDI server, namely the one in AS 4. Keep in mind that it is the objective of the CDI to direct the client to a CDI server that is close to the client.

To complete the picture one question remains. How does the CDI choose the "nearest" server to deliver the content from? Today's CDI landscape relies mainly on three techniques to assign end-users to servers.

1. IP-Anycast
2. DNS based redirection
3. HTTP redirection

While all techniques help the CDIs to assign end-users to their servers, all of them have different drawbacks. In the following we will explain how the different techniques work and what those drawbacks are:

IP-Anycast. IP Anycast is a routing technique used to send IP packets to the topologically closest member of a group of potential CDI servers. IP Anycast is usually realized by announcing the destination address from multiple locations in a network or on the Internet. Since the same IP address is available at multiple locations, the routing process selects the shortest route for the destination according to its configuration. Simply speaking, each router in a network selects one of the locations the Anycasted IP is announced from based on the used routing metrics (e.g., path length or routing weights) and configures a route towards it. Note that, if a network learns of an Anycasted IP address from different sources, it does not necessarily direct all its traffic to one of its locations. Its routing can decide to send packets from region A in the network to location A' while region B gets a route to location B'. This means that the entire server selection of a CDI becomes trivial as it is now a part of the routing process. This means that the CDI loses control of how the users are mapped to the server because the network calculates the routing based on its own metrics. Another issue is that the routing in a network is optimized based on the ISPs criteria which might not be the same as the CDIs or even contrary. Thus the "nearest" server might not be the best one the CDI could offer.

DNS-based redirection. Today most CDIs rely on the Domain Name System (DNS) to direct users to appropriate servers. When requesting content, the end-user typically asks a DNS resolver, e.g., the resolver of its ISP, for the resolution of a domain name. The resolver then asks the authoritative server for the domain. This can be the CDI's authoritative server, or the the content provider's authoritative server, which then delegates to the CDI's authoritative server. At this point the CDI selects the server for this request based on where the request comes from. But the request does not come directly from the end-user but from its DNS resolver! Thus the CDI can only select a server based on the IP address of the end-user's DNS resolver. To improve the mapping of end-users to servers, the client-IP eDNS extension [47] has been recently proposed. Criteria for server selection include the availability of the server, the proximity of the server to the resolver, and the monetary cost of delivering the content. For proximity estimations the CDIs rely heavily on network measurements [135] and geolocation information [121] to figure out which of their servers is close by and has the best network path performance. A recent study [3] showed that sometimes the end user is not close

to the resolver and another study points out that geolocation databases can not be relied upon [147]. Thus the proximity estimations for the "nearest" CDI server highly depend on the quality and precision of network measurements and a proper DNS deployment of the ISPs. For an excellent survey on DNS-based Server Selections in CDNs, we refer the reader to [141].

HTTP redirection. The Hypertext Transfer Protocol (HTTP) is today's de-facto standard to transport content in the Internet (see section 5). The protocol incorporates a mechanism to redirect users at the application level at least since it was standardized as version 1.0 in 1996 [30]. By sending an appropriate HTTP status code (HTTP status codes 3XX) the web server can tell the connected user that a requested object is available from another URL, which can also point to another server. This allows a CDI to redirect an end-user to another server. Reasons for this might include limited server capacities, poor transfer performance or when another server is closer to the end-user, e.g., a client from the US connecting to a server in Europe although the CDI has servers in the US. The HTTP redirection mechanism has some important benefits over the DNS based approach. First, the CDI directly communicates with the end-user and thus knows the exact destination it sends the traffic to (opposed to the assumption that the DNS resolver is "close"). Yet it still has to estimate the proximity of the end-user using the same methodologies as described in the DNS based case. Second, the CDI already knows which object the end-user requests and can use this information for its decision. It allows a CDI to direct a user towards a server where the content object is already available to improve its cache hit rate. Other important informations includes the size and type of the object. This allows the CDI to optimize the server selection based on the requirements to transfer the object e.g., for delay sensitive ones like streaming video or more throughput oriented ones like huge software patches. Yet this improvement comes at a price as the user has to establish a new connection to another server. This includes another DNS lookup to get the servers IP address as well as the whole TCP setup including performance critical phases like slow start. This can repeat itself multiple times before an appropriate server is found, which delays the object delivery even further.

6.2 Peer-to-Peer Networks

Peer-to-peer (P2P) is a distributed system architecture in which all participants, the so called peers, are equally privileged users of the system. A P2P system forms an overlay network on top of existing communication networks (e.g., the Internet). All participating peers of the P2P system are the nodes of the overlay network graph, while the connections between them are the edges. It is possible to extend this definition of edges in the overlay network graph to all known peers, in contrast to all connected peers. Based on how peers connect to each other and thus build the overlay network, we can classify P2P systems into two basic categories:

Unstructured: The P2P system does not impose any structure on the overlay network. The peers connect to each other in an arbitrary fashion. Most often peers are chosen randomly. Content lookups are flooded to the network (e.g., Gnutella), resulting in limited scalability, or not offered at all (e.g., plain BitTorrent).

Structured: Peers organize themselves following certain criteria and algorithms. The resulting overlay network graphs have specific topologies and properties that usually offer better scalability and faster lookups than unstructured P2P systems (e.g., Kademlia, BitTorrent DHT).

The overlay network is mainly used for indexing content and peer discovery while the actual content is transferred directly between peers. Thus the connection between the individual peers has significant impact on both the direct content transfers as well as the performance of the resulting overlay network. This has been shown in previous studies and multiple solutions have been proposed [182, 39, 169, 11, 18, 133] which are described in detail in section 10.

Applications of P2P systems in content delivery range from time insensitive applications like file sharing, software delivery or patch distribution to very time sensitive ones like streaming TV or on demand video delivery.

Peer-to-Peer systems To construct an overlay topology, unstructured P2P networks usually employ an arbitrary neighbor selection procedure [167]. This can result in a situation where a node in Frankfurt downloads a large content file from a node in Sydney, while the same information may be available at a node in Berlin. While structured P2P systems follow certain rules and algorithms, the information available to them either has to be inferred by measurements [151] or rely on publicly available information such as routing information [153]. Both options are much less precise and up-to-date compared to the information information an ISP has readily at hand. It has been shown that P2P traffic often crosses network boundaries multiple times [8, 98]. This is not necessarily optimal as most network bottlenecks in the Internet are assumed to be either in the access network or on the links between ISPs, but rarely in the backbones of the ISPs [17]. Besides, studies have shown that the desired content is often available "in the proximity" of interested users [98, 150]. This is due to content language and geographical regions of interest. P2P networks benefit from increasing their traffic locality, as shown by Bindal et. al [31] for the case of BitTorrent.

P2P systems usually implement their own routing [20] in the overlay topology. Routing on such an overlay topology is no longer done on a per-prefix basis, but rather on a query or key basis. In unstructured P2P networks, queries are disseminated, e.g., via flooding [81] or random walks, while structured P2P networks often use DHT-based routing systems to locate data [167]. Answers can either be sent directly using the underlay routing [167] or through the overlay network by retracing the query path [81]. By routing through the overlay of P2P nodes, P2P systems hope to use paths with better performance than those available via the Internet native routing [20, 156]. However, the benefits of redirecting traffic on an alternative path, e.g., one with larger available bandwidth or lower delay, are not necessarily obvious. While the performance of the P2P system may temporarily improve, the available bandwidth of the newly chosen path may deteriorate due to the traffic added to this path. The ISP has then to redirect some traffic so that other applications using this path can receive enough bandwidth. In other words, P2P systems reinvent and re-implement a routing system whose dynamics should be able to explicitly interact with the dynamics of native Internet routing [99, 158]. While a routing underlay as proposed by Nakao et al. [129] can reduce the work duplication, it cannot by itself overcome the problems created by the interaction. Consider a situation where a P2P system imposes a lot of traffic load on an ISP network. This may cause the ISP to change some routing metrics and therefore some paths (at the native routing layer) in order to improve its network utilization. This can however cause a change of routes at the application layer by the P2P system, which may again trigger a response by the ISP, and so on.

P2P today. The P2P paradigm has been very successful in delivering content to end-users. BitTorrent [45] is the prime example, used mainly for file sharing. Other examples include more time sensitive applications such as video streaming [54, 116, 106]. Despite the varying (and perhaps declining) share of P2P traffic in different regions of the world [118], P2P traffic still constitutes a significant fraction of the total Internet traffic. P2P systems have been shown to scale application capacity well during flash crowds [183]. However, the strength of P2P systems, i.e., anybody can share anything over this technology, also turns out to be a weakness when it comes to content availability. In fact, mostly popular content is available on P2P networks, while older content disappears as users' interest in it declines. In the example of BitTorrent, this leads to torrents missing pieces, in which case a download can never be completed. In case of video streaming, the video might simply no longer be available or the number of available peers is too low to sustain the required video bit-rate, resulting in gaps or stuttering of the video stream.
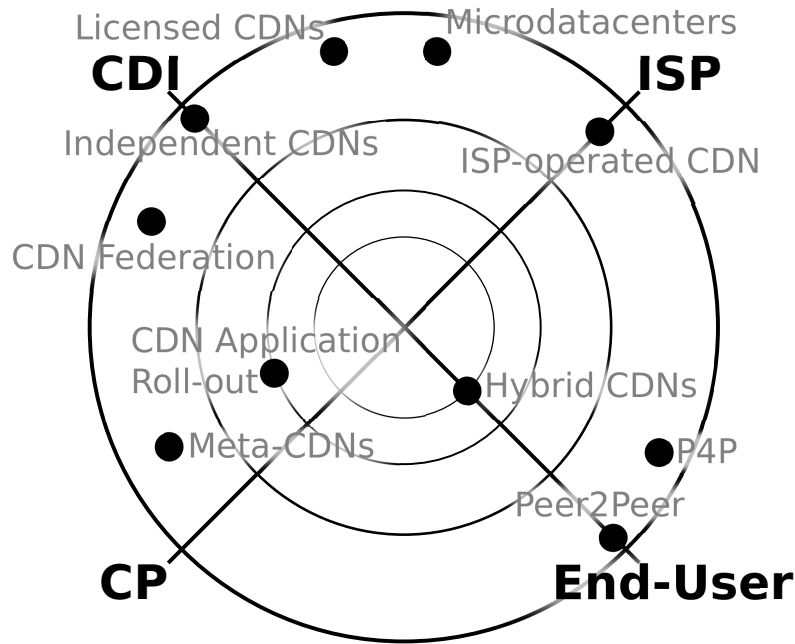
Figure 15: Spectrum of content delivery solutions and involvement of stake-holders in the content delivery. Reprinted from [73]. Included here by permission.

7 Content Delivery: The Landscape

Internet traffic grows at a rate of approximately 30% per year [43] and is dominated by the delivery of content to end users [2, 79, 107, 145]. To cope with the increasing demand for content, and to support the level of reliability and scalability required by commercial-grade applications, Content Distribution Infrastructures (CDIs) have emerged. In general terms, CDIs are overlays built on top of existing network infrastructures that aim to accelerate the delivery of content to end-users. CDIs include, but are not limited to, Content Distribution Networks (CDNs), such as Akamai and Google, Video Streaming Portals (VSP) such as YouTube, One-Click-Hosters (OCH) like Rapidshare and MegaUpload. However, a CDI does not necessarily produce the content that it delivers. Thus, we define a Content Producer (CP) as the entity that generates content. In some cases, e.g., Google and YouTube, the CP and CDI can be the same entity. In other instances, for example Akamai and Limelight, the CDI only delivers what a CP pays for.

But not all CDIs are built upon the same philosophy, designs and technology. For example, a CDI can be operated independently by deploying caches in different networks, by renting space in datacenters or by building its own datacenters. Furthermore, some CDIs are operated by ISPs, by Content Producers, or in the case of Peer-to-Peer networks, by self-organized end-users. To summarize the spectrum of CDI solutions, Figure 15 provides an overview of different CDI solutions. They are aligned by their architectures according to which parties are involved.

7.1 Independent Content Distribution

Independent CDIs are usually referred to as Content Delivery Networks (CDNs). They have a strong customer base of content producers and are responsible for delivering the content of their customers to end-users around the world. Today, they are, by traffic volume as well as hosted content, the largest players on the Internet. In general, there are four main components to independent CDN architectures: a server deployment, a strategy for replicating content on servers, a mechanism for directing users to servers, and a system for collecting and processing server logs.

For server deployment, three main approaches exist [112]: centralized, datacenter based and distributed infrastructures:

Central Location: This approach is used by small CDNs, One-Click Hosters, and applications running in public clouds. Centralized hosting takes advantage of (a) the economies of scale that a single location offers [25], (b) the flexibility that multihoming offers [82], and (c) the connectivity opportunities that IXPs offer [2]. The disadvantages of centralized hosting are the potential for a single point of failure, and the limited ability to ensure low latency to users located in different networks around the world [113].

Datacenter Based: This approach deploys in several large data centers. It again leverages economies of scale while improving reliability and creating a larger footprint with further reach. However, by utilizing multiple datacenters, new challenges regarding the content distribution, synchronization and delivery arise. For example, the datacenter delivering content to an end-user cannot be statically configured anymore, but the selection needs to take the location of the end-user into account. This approach is used by CDNs such as Limelight, EdgeCast and BitGravity. Many cloud providers also use this approach, including Amazon CloudFront and Microsoft Azure.

Distributed Infrastructures: This approach consists of a highly distributed infrastructure deployment, potentially deep inside third-party networks. Here, the large number of servers scattered across numerous networks offer high availability and replication of content while being very close to end-users. Furthermore, this type can balance traffic across locations, best react to flash crowds by dynamic server assignments, and deliver content with improved latency. However, with the highly distributed infrastructures, the challenges of assigning users to the right server location increase many-fold. Also, with deep deployment datacenters are usually not available anymore, leading to the question where to deploy how many servers. Today, Akamai is only one independent CDN that uses this approach on a global scale.

CDNs with more than one location typically follow a pull strategy [135] for content distribution and replication. Thus, content requests can be directed to servers that do not have the required object cached. When a requested object is not at the selected server, neighboring servers in the same cluster or region are asked. If the object is not available at neighboring servers, the origin or root server responsible for the object is contacted to retrieve the content. A requested object that is fetched from a remote server is saved locally and then delivered to the end user. To keep the copies of the object fresh, a TTL value is assigned to it. When the TTL value expires, the object is removed. For scalability reasons, any server of the CDN or within a region can respond to the request of an end user [172].

A special case of the independent CDI category are free CDNs such as Coral [77], which follow a similar architectural design. In these CDNs, server resources are offered by end-users or non-profit organizations.

7.2 ISP-operated CDIs

The potential for generating revenue from content delivery has motivated a number of ISPs to build and operate their own Content Distribution Infrastructures. For example, large ISPs such as AT&T and Verizon have built their own CDNs along the same general architectural principles as independent CDIs. However, due

to the limitations arising from being restricted to one network, these CDNs are not deployed in a distributed fashion across multiple networks and thus are not globally operating solutions. To overcome this issue, the CDNi group at the IETF [133] is discussing how to interconnect these CDNs to boost their efficiency and coverage. The content provider are third parties, applications and services offered by the ISP. Other ISPs with large footprints, such as Level3 and Telefonica [109, 110], have also built CDNs in order to efficiently transfer content across the globe and offer improved services to their end users.

7.3 Emerging Trends in CDI Architectures

Economics, especially cost reduction, is the key driving force behind emerging CDI architectures. The content delivery market has become highly competitive. While the demand for content delivery services is rising and the cost of bandwidth is decreasing, the profit margins of storage and processing [25] are dwindling, increasing the pressure on CDIs to reduce costs. At the same time, more parties are entering the market in new ways, looking to capture a slice of the revenue. However, today's traditional CDI deployments lack agility to combat these effects. Contracts for server deployments last for months or years and the available locations are typically limited to datacenters. The time required to install a new server today is in the order of weeks or months. Such timescales are too large to react to changes in demand. CDIs are therefore looking for new ways to expand or shrink their capacity, on demand, and especially at low cost.

7.3.1 Hybrid Content Distribution

In a hybrid CDI, end-users download client software that assists with content distribution. As in P2P file-sharing systems, the content is broken into pieces and offered by both other users who have installed the client software as well as by the CDI's servers. The client software contacts dedicated CDI servers, called control plane servers, which schedule which parts of the content are to be downloaded from what peers. Criteria for selecting peers include AS-level proximity as well as the availability of the content. If no close peers are found, or if the download process from other peers significantly slows the content delivery process, the traditional CDI servers take over the content delivery job entirely. Akamai already offers NetSession [1], a hybrid CDI solution for delivering very large files such as software updates at lower cost to its customers. Xunlei [51], an application aggregator with high penetration in China, follows a similar paradigm. It is used to download various types of files including videos, executables, and even emails, and supports popular protocols such as HTTP, FTP, and RTSP. Xunlei maintains its own trackers and servers. A study of hybrid CDIs [89] showed that up to 80% of content delivery traffic can be outsourced from server-based delivery to end-users, without significant degradation in total download time.

7.3.2 Licensed CDNs

Licensed CDNs have been proposed to combine the benefits of the large content-provider customer base of an independent CDI with the end-user base of an ISP [168]. A licensed CDN is a partnership between an independent CDI and an ISP. The CDI licenses the content delivery software that runs on servers to the ISP while the ISP owns and operates the servers. The servers deliver content to the end-users and report logging information back to the CDI. The revenue derived from content producers is then shared between the two parties. Thus, a CDI can expand its footprint deep inside an ISP network without investing in hardware, incurring lower operating costs. The ISP benefits from not having to invest in developing the software for a reliable and scalable content distribution. More importantly, a licensed CDN also alleviates the ISP's need to negotiate directly with content producers, which might be challenging, given an ISPs limited footprint.

7.3.3 Application-based CDIs

Recently, large application and content producers have rolled out their own CDIs, hosted in multiple large data centers. Some popular applications generate so much traffic that the content producers can better amortize delivery costs by doing content distribution themselves. Google is one such example. It has deployed a number of data centers and interconnected them with high speed backbone networks. Google connects its datacenters to a large number of ISPs via IXPs and also via private peering. Google has also launched the Google Global Cache (GGC) [84], which can be installed inside ISP networks. The GGC reduces the transit cost of small ISPs and those that are located in areas with limited connectivity, e.g., Africa. The GGC servers are given for free to the ISPs which install and maintain them. GGC also allows an ISP to advertise through BGP the prefixes of users that each GGC server should serve. As another example, Netflix, which is responsible for around 30% of the traffic in North America at certain times, is also rolling out its own CDI. The Netflix system is called Open Connect Network [131]. Netflix offers an interface where ISPs can advertise, via BGP, their preferences as to which subnets are served by which Open Connect Network servers.

7.3.4 Meta-CDIs

Today, content producers contract with multiple CDIs to deliver their content. To optimize for cost and performance [115], meta-CDIs act as brokers to help with CDI selection. These brokers collect performance metrics from a number of end-users and try to estimate the best CDI, based on the server that a user is assigned. To this end, the brokers place a small file on a number of CDIs. Then they embed the request for this file in popular websites' source code, in the form of a javascript. When users visit these sites, they report back statistics based on the servers that each CDI assigned the users. The broker then recommends CDIs for a given source of demand taking also into consideration the cost of delivery. Cedexis is one of these brokers for web browsing. Another broker for video streaming is Conviva [54]. These brokers may compensate when a CDI does not assign a user to the optimal server (which a recent study [145] has shown sometimes occurs) by selecting a different CDI.

7.3.5 CDI Federations

To avoid the cost of providing a global footprint and perhaps to allow for a single negotiating unit with content providers, federations of CDIs have been proposed. In this architecture, smaller CDIs, perhaps operated by ISPs, join together to form a larger federated CDI. A CDI belonging to the federation can replicate content to a partner CDI in a location where it has no footprint. The CDI reduces its transit costs because it only has to send the object once to satisfy the demand for users in that location. Overall, cost may be reduced due to distance-based pricing [174]. The IETF CDNi working group [133] works on CDI federation.

8 Challenges in Content Delivery

The challenges that CDIs and P2P systems are faced with are based on the fact that they are unaware of the underlying network infrastructure and its conditions. In the best case, they can try to detect and infer the topology and state of the ISP's network through measurements, but even with large scale measurements, it is a difficult task, especially if accuracy is necessary. Furthermore, when it comes to short-term congestion and/or avoiding network bottlenecks, measurements are of no use. In the following we describe the

challenges those systems face in more detail.

8.1 Content Delivery Infrastructures (CDIs)

From the viewpoint of the end-users and ISPs, the redirection schemes employed by existing CDIs have three major limitations:

Network Bottlenecks. Despite the traffic flow optimization performed by CDIs, the assignment of end-user requests to servers by CDIs may still result in sub-optimal content delivery performance for the end-users. This is a consequence of the limited information CDIs have about the network conditions between the end-user and their servers. Tracking the ever changing conditions in networks, i.e., through active measurements and end-user reports, incurs an overhead for the CDI without a guarantee of performance improvements for the end-user. Without sufficient information about the network paths between the CDI servers and the end-user, any assignment performed by the CDI may lead to additional load on existing network bottlenecks, or to the creation of new bottlenecks.

User Mis-location. DNS requests received by the CDI DNS servers originate from the DNS resolver of the end-user, not from the end-user itself. The assignment is therefore based on the assumption that end-users are close to their DNS resolvers. Recent studies have shown that in many cases this assumption does not hold [119, 3]. As a result, the end-user is mis-located and the server assignment is not optimal. As a response to this issue, DNS extensions have been proposed to include the end-user IP information [47].

Content Delivery Cost Finally, CDIs strive to minimize the overall cost of delivering huge amounts of content to end-users. To that end, their assignment strategy is mainly driven by economic aspects. While a CDI will always try to assign users in such a way that the server can deliver reasonable performance, this can again result in end-users not being directed to the server able to deliver best performance.

8.2 Peer-to-Peer Networks (P2P)

P2P traffic often starves other applications like Web traffic of bandwidth [163]. This is because most P2P systems rely on application layer routing based on an overlay topology on top of the Internet, which is largely independent of the Internet routing and topology [8]. This can result in a situation where a node in Frankfurt downloads a large content file from a node in Sydney, while the same information may be available at a node in Berlin. As a result P2P systems use more network resources due to traffic crossing the underlying network multiple times. For more details and information on P2P systems, see Section 6.2.

8.3 Internet Service Providers (ISPs)

ISPs face several challenges regarding the operation of their network infrastructure. With the emergence of Content, and especially distributed content delivery, be it from CDIs or P2P networks, these operational challenges have increased manifold.

Network Provisioning. Provisioning and operation a network means running the infrastructure at its highest efficiency. To ensure this, new cables as well as the peering points with other networks need to be

established and/or upgraded. However, with the emergence of CDIs and P2P networks, the network provisioning has become more complicated, since the network loads tend to shift depending on the content that is currently transported while the direct peering might not be effective anymore.

Volatile Content Traffic. CDIs and P2P networks strive to optimize their own operational overhead, possibly at the expense of the underlying infrastructure. In terms of CDIs, this means that a CDI chooses the best server based on its own criteria, not knowing what parts of the networks infrastructure is being used. Especially with globally deployed CDIs it becomes increasingly difficult for ISPs to predict what CDI is causing what traffic from where based on past behavior. This has a direct implication on the traffic engineering of the network, as this is usually based on traffic predictions from past network traffic patterns.

Customer Satisfaction. Regardless of the increased difficulty with network provisioning and traffic engineering, end-users are demanding more and larger content. This, coupled with the dominant form of flat rates for customer subscriptions, increases the pressure on ISPs to delay network upgrades as long as possible to keep prices competitive. But letting links run full increases packet loss. This, in turn, drastically reduces the quality of experience of the end-users. This, in turn, encourages end-users to switch their subscriptions.

8.4 Summary

In summary, we identify the following challenges in todays content delivery:

- The ISP has limited ability to manage its traffic and therefore incurs potentially increased costs, e.g., for its interdomain traffic, as well as for its inability to do traffic engineering on its internal network while having to offer competitive subscriptions to its end-users.
- The P2P system has limited ability to pick an optimal overlay topology and therefore provide optimal performance to its users, as it has no prior knowledge of the underlying Internet topology. It therefore has to either disregard or reverse engineer it.
- The CDI has limited ability to pick the optimal server and therefore provide optimal performance to its users, as it has to infer the network topology as well as the dynamic network conditions. Moreover, it has limited knowledge about the location of the user as it only knows the IP address of the DNS resolver.
- The different systems try to measure the path performance independently.

9 Incentives for Collaboration

ISPs are in a unique position to help CDIs and P2P systems to improve content delivery. Specifically, ISPs have the knowledge about the state of the underlying network topology and the status of individual links that CDIs are lacking. This information not only helps CDIs in their user-to-server mapping, but also reduces the need for CDIs to perform large-scale active measurements and topology discovery [16]. It also enables CDIs to better amortize their existing infrastructure, offer better quality of experience to their users, and plan their infrastructure expansion more efficiently. On the other side, ISPs are not just selflessly giving up their network information. Offering their intimate knowledge of the network to CDIs puts ISPs in the position that they can also actively guide the CDIs. This allows ISPs to gain unprecedented influence on CDI traffic.

The opportunity for ISPs to coordinate with CDIs is technically possible thanks to the decoupling of server selection from content delivery. In general, any end-user requesting content from a CDI first does a

mapping request, usually through the Domain Name System (DNS). During this request, the CDI needs to locate the network position of the end-user and assign a server capable of delivering the content, preferably close to the end-user. ISPs have this information ready at their fingertips, but are currently not able to communicate their knowledge to CDIs. Furthermore, ISPs solve the challenge of predicting CDI traffic, which is very difficult due to the lack of information on the CDI mapping strategy regarding the end-users to servers assignment. In order to reap the benefits of the other's knowledge, both parties require incentives to work together.

9.1   Incentives for CDIs

The CDIs' market requires them to enable new applications while reducing their operational costs and improve end-user experience [135]. By cooperating with an ISP, a CDI improves the mapping of end-users to servers, improves in the end-user experience, has accurate and up-to-date knowledge of the networks and thus gains a competitive advantage. This is particularly important for CDIs in light of the commoditization of the content delivery market and the selection offered to end-users, for example through meta-CDNs [54]. The improved mapping also yields better infrastructure amortization and, thanks to cooperation with ISPs, CDIs will no longer have to perform and analyze voluminous measurements in order to infer the network conditions or end-user locations.

To stimulate cooperation, ISPs can operate and provide their network knowledge as a free service to CDIs or even offer discounts on peering or hosting prices, e.g., for early adopters and CDIs willing to cooperate. The loss of peering or hosting revenue is amortized with the benefits of a lower network utilization, reduced investments in network capacity expansion and by taking back some control over traffic within the network. Ma et al. [117] have developed a methodology to estimate the prices in such a cooperative scheme by utilizing the Shapley settlement mechanism. Cooperation can also act as an enabler for CDIs and ISPs to jointly launch new applications in a cost-effective way, for example traffic-intensive applications such as the delivery of high definition video on-demand, or real-time applications such as online games.

9.2   Incentives for ISPs

ISPs are interested in reducing their operational and infrastructure upgrade costs, offering broadband services at competitive prices, and delivering the best end-user experience possible. Due to network congestion during peak hour, ISPs in North America have recently revisited the flat pricing model and some have announced data caps to broadband services. A better management of traffic in their networks allows them to offer higher data caps or even alleviate the need to introduce them. From an ISP perspective, cooperation with a CDI offers the possibility to do global traffic and peering management through an improved awareness of traffic across the whole network. For example, peering agreements with CDIs can offer cooperation in exchange for reduced costs to CDIs. This can be an incentive for CDIs to peer with ISPs, and an additional revenue for an ISP, as such reduced prices can attract additional peering customers. Furthermore, collaboration with CDIs has the potential to reduce the significant overhead due to the handling of customer complaints that often do not stem from the operation of the ISP but the operation of CDIs [40]. Through this, ISPs can identify and mitigate congestion in content delivery, and react to short disturbances caused by an increased demand of content from CDIs by communicating these incidents back directly to the source.

9.3 Effect on End-users

Collaboration between ISPs and CDIs in content delivery empowers end-users to obtain the best possible quality of experience. As such, this creates an incentive for end-users to support the adoption of collaboration by both ISPs and CDIs. For example, an ISP can offer more attractive products, i.e., higher bandwidth or lower prices, since it is able to better manage the traffic inside its network. Also, thanks to better traffic engineering, ISPs can increase data caps on their broadband offers, making the ISP more attractive to end-users. Moreover, CDIs that are willing to jointly deliver content can offer better quality of experience to end-users. This can even be done through premium services offered by the CDI to its customers. For example, CDIs delivering streaming services can offer higher quality videos to end-users thanks to better server assignment and network engineering.

10 Opportunities for Collaboration

As pointed out ISPs are in a unique position to help CDIs and P2P systems to improve content delivery since they have the knowledge about the state of the underlying network topology, the status of individual links, as well as the location of the user. In this section we first describe the high level concept all existing solutions have in common and then continue by illustrating where and why they differ in certain aspects.

The presented solutions include the original Oracle concept proposed by Aggarwal et al [11], P4P proposed by Xie et al. [182], Ono proposed by Choffnes and Bustamante [39] and PaDIS proposed by Poese et al. [146]. We also give an overview of the activities within the IETF which have been fueled to some extend by the proposed systems discussed in this section, namely ALTO and CDNi.

10.1 Conceptual Design

To overcome the challenges in Content Delivery, recall section 8, various solutions have been proposed by the research community. While they all differ in certain aspects, their basic idea is the same: utilize available information about the network to make an educated selection prior connecting to a service. Following this idea, all of the proposed solution employ the same basic conceptual design: the *management plane* is responsible for collecting up-to-date information about the network while the *control plane* acts as an interface to this information for the application.

Management Plane: The Network Map. The systems management plane is responsible to collect up-to-date state network information, such as network topology, routing information, link utilization and other important metrics. This information is used to maintain an internal map of the network representing the current state of the real network. One important aspect of this component is how the information about the network is retrieved. The different implementations range from active measurements over passive measurements to active participation in network management systems (such as BGP). Another important aspect is the frequency in which the information is collected. For certain information such as topology or routing an immediate update is necessary to guarantee correct functioning of the system, while others, such as link utilization or packet loss rates, only degrade the quality of the system. Still other information, such as link capacities or transmission delays, can be considered (semi-)static. Last but not least the systems differ in what information is necessary to be operational and if additional information sources can be used to improve accuracy.

Control Plane: The Information Interface. The control plane of the system is responsible for providing an interface to the information of the management plane so that clients can make use of the information. This can basically be seen as an interface or API that clients can query to get information about the current network state. The various proposed solutions differ mainly in which fashion and at which granularity the information can be retrieved. There are two main competing approaches: abstracted network maps and preference lists. The first one transforms the available information from the management plane into an annotated representation of nodes and edges. The big difference to the actual data of the management plane is the aggregation level and the specific annotations. Clients can then query the system to get an up-to-date abstract network map, which they can use to decide which of the possible destination to connect to by calculating the best candidates by themselves using their own optimization target. The second one uses the information of the management plane to create a ranked list of possible service destinations (read: IP addresses). The required input includes the source, possible destinations and (if the system supports multiple collaboration objectives) an optimization goal, e.g., minimal delay. The output consists of a re-ordered list of the possible destinations in regard to the optimization goal, the first being the most and the last being the least desirable destination.

Note that in both cases the client is in the position to select the final destination, allowing to completely ignore the additional information. Another important fact is that the client is not necessarily the end-user but might be a service provider themselves. For instance a company providing content delivery service (CDN) could make use of this service to improve its user-to-server mapping accuracy or in case of the BitTorrent P2P system the tracker could query the service prior returning an initial peer list to a connected client. While not strictly necessary, the two components are usually implemented as separate entities within the system to allow better scalability, information aggregation and/or anonymization without loosing precision or multiple collaboration objectives. In addition to that, all systems table important issues for any collaboration approach, such as privacy information leakage or targeted objective(s).

The presentation of the following solutions will outline the specific implementation and thus highlights the differences between the solutions.

10.2   P2P Oracle Service

Aggarwal et al. [11] describe an *oracle* service to solve the mismatch between the overlay network and underlay routing network in P2P content delivery. Instead of the P2P node choosing neighbors independently, the ISP can offer a service, the *oracle*, that ranks the potential neighbors according to certain metrics: a client supplied peer list is re-ordered based on coarse-grained distance metrics, e.g., the number of AS hops [88], the peer being inside/outside the AS or the distance to the edge of the AS. This ranking can be seen as the ISP expressing preference for certain P2P neighbors. For peers inside the network additional information can be used, such as access bandwidth, expected delay or link congestion to further improve the traffic management.

10.3   Proactive Network Provider Participation for P2P (P4P)

The "Proactive Network Provider Participation for P2P" is another approach to enable cooperative network information sharing between the network provider and applications. The P4P architecture [182] introduces iTrackers as portals operated by network providers that divides the traffic control responsibilities between providers and applications. Each iTracker maintains an internal representation of the network in the form of nodes and annotated edges. A node represents a set of clients that can be aggregated at different levels, e.g., certain locations (PoP) or network state (similar level of congestion). Clients can query the iTracker

to obtain the "virtual" cost for possible peer candidates. This "virtual" cost allows the network operators to express any kind of preferences and may be based on the provider's choice of metrics, including utilization, transit costs, or geography. It also enables the client to compare and choose the most suited peers to connect to.

10.4 Ono - Travelocity-based Path Selection

The Ono system [39] by Choffnes and Bustamante is based on "techniques for inferring and exploiting the network measurements performed by CDNs for the purpose of locating and utilizing quality Internet paths without performing extensive path probing or monitoring" proposed by Su et al. in [169]. Based on their observations that CDN redirection is driven primarily by latency [169], they formulate the following hypothesis: Peers that exhibit similar redirection behavior of the same CDN are most likely close to each other, probably even in the same AS. For this each peer performs periodic DNS lookups on popular CDN names and calculates how close other peers are by determining the cosine similarity with their lookups. To share the lookup among the peers they use either direct communication between Ono enabled peers or via distributed storage solutions e.g., DHT-based. On the downside Ono relies on the precision of the measurements that the CDNs perform and that their assignment strategy is actually based mainly on delay. Should the CDNs change their strategy in that regard Ono might yield wrong input for the biased peer selection the authors envision.

When considering our design concept described above, Ono is a bit harder to fit into the picture: Ono distributes the functionality of the management and control planes among all participating peers. Also, Ono does not try to measure the network state directly, but infers it by observing Akamai's user-to-server mapping behavior on a large scale and relies on Akamai doing the actual measurements [135], Thus the management plane of Ono consists of recently resolved hostnames from many P2P clients. The quality of other peers can then be assessed by the number of hostnames that resolve to the same destination. The control plane in Ono's case is a DHT, which allows decentralized reads and writes of key-value pairs in a distributed manner, thus giving access to the data of the management plane.

10.5 Provider-aided Distance Information System (PaDIS)

In [145, 146] Poese et al. propose a "Provider-aided Information Systems (PaDIS)", a system to enable collaboration between network operators and content delivery systems. The system enhances concept of the P2P Oracle to include server based content delivery systems (e.g., CDNs), to maintain an up-to-date annotated map of the ISP network and its properties as well as the state of ISP-operated servers that are open for rent. In addition, it provides recommendations on possible locations for servers to better satisfy the demand by the CDN and ISP traffic engineering goals. In the management plane, it gathers detailed information about the network topology, i.e., routers and links, annotations such as link utilization, router load as well as topological changes. An Interior Gateway Protocol (IGP) listener provides up-to-date information about routers and links. Additional information, e.g., link utilization and other metrics can be retrieved via SNMP. A Border Gateway Protocol (BGP) listener collects routing information to calculate the paths that traffic takes through the network, including egress traffic. Ingress points of traffic can be found by utilizing Netflow data. This allows for complete forward and reverse path mapping inside the ISP and enables a complete path map between any two points in the ISP network. While PaDIS builds an anotated map of the ISP network, it keeps the information acquired from other components in separate data structures. This separation ensures that changes in prefix assignments do not directly affect the routing in the annotated network map. Pre-calculating path properties for all paths, allow for constant lookup speed independent of path

length and network topology. On the control plane, PaDIS makes use of the prefrence lists known from the P2P Oracle, but supports multiple, individual optimization targets. Apart from basic default optimizations (e.g., low delay, high throughput), additional optimizations can be negotiated between the network operator and the content delivery system. For CDN-ISP collaboration opportunities when the ISP operates both the network and the CDN we refer the reader to [94, 53, 162, 60].

10.6 Application-Layer Traffic Optimization (ALTO)

The research into P2P traffic localization has led the IETF to form a working group for "Application Layer Traffic Optimization (ALTO)" [120]. The goal of the ALTO WG is to develop Internet standards that offer "better-than-random" peer selection by providing information about the underlying network and to design a query-response protocol that the applications can query for an optimized peer selection strategy [18]. On the control plane, ALTO offers multiple services to the Applications querying it, most notably are the Endpoint Cost Service and the Map service. The Endpoint Cost Service allows the Application the query the ALTO server for costs and rankings based on endpoints (usually IP subnets) and use that information for an optimized peer selection process or to pick the most suitable server of a CDI. The Network Map service makes use of the fact that most endpoints are in fact rather close to each other and thus can be aggregated into a single entity. The resulting set of entities is then called an ALTO Network Map. The definition of proximity in that case depends on the aggregation level, in one Map endpoints in the same IP subnet may be considered close while in another all subnets attached to the same Point of Presence (PoP) are close. In contrast to the Endpoint Cost Service the ALTO Network Map is suitable when more Endpoints need to be considered and offers better scalability, especially when coupled with caching techniques. Although the ALTO WG statement is more P2P centric, the service is also suitable to improve the connection to CDN servers.

11 Collaboration Use Cases: P2P and TE

The growth of demand for content is motivating collaboration between ISPs and applications. In this chapter we review to use cases: P2P and Traffic Engineering.

11.1 Use Case: P2P

Recall, P2P systems are self-organizing systems of autonomous entities, called peers, that cooperate for common goals. These common goals range from sharing of resources, e.g., music and video files, processing power, or storage space [167] to collaborative routing as in Skype and P2P-TV. A fundamental characteristic of these systems is their distributed designs and resources.

Advantages of P2P systems include elimination of bottlenecks and single points-of-failure within the system, increased processing power, high availability/redundancy, and little or no dependence on any particular central entity. However, P2P systems are plagued by some fundamental issues, such as overlay/underlay topological and routing mismatch [158], inefficiencies in locating and retrieving resources, and scalability and performance issues caused by uncontrolled traffic swamps [167].

Several of these drawbacks can be addressed by collaboration between the P2P overlay and the Internet routing underlay. To overcome these limits each ISP can offer the "oracle" service as introduced in Section 10.2 to the P2P users which explicitly helps P2P users to choose "good" neighbors. The P2P user can supply its ISP's oracle with a list of possible P2P neighbors, during bootstrapping and/or content exchange. The ISP's oracle then returns a ranked list to the querying user, according to its preference (e.g., AS-hop

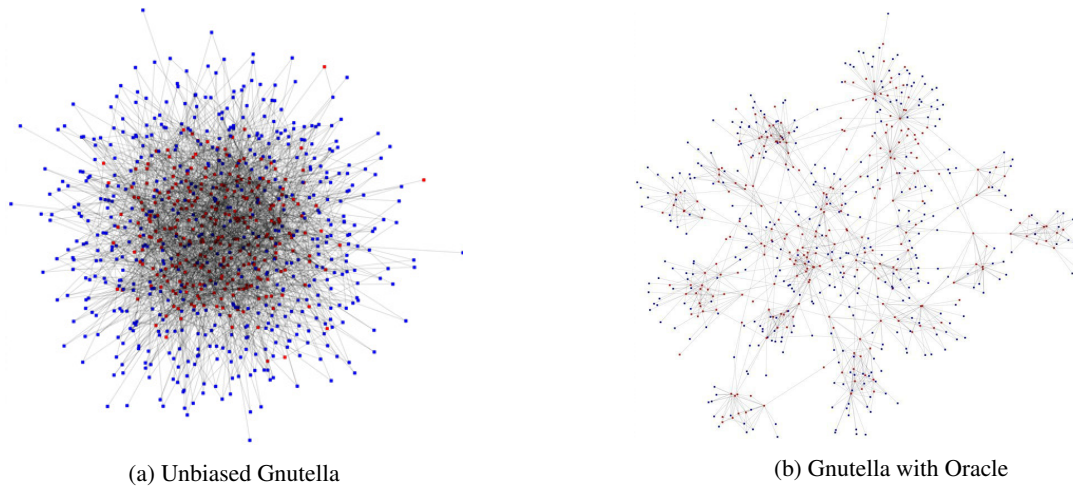(a) Unbiased Gnutella   (b) Gnutella with Oracle

Figure 16: Visualization of Gnutella overlay topology. Reprinted from [11]. Included here by permission.

distance) and knowledge of the ISP topology and traffic volume, while at the same time keeping the interest of the P2P user in mind. We show that in principle, P2P systems as well as the ISPs profit from the use of the oracle even when only considering the AS-distance for ranking nodes [11], because the overlay topology is now localized and respects the underlying Internet topology, and the P2P user profits from the ISP's knowledge.

To study the impact of biased neighbor selection on a real P2P network that implements its own routing, we run extensive simulations of the Gnutella protocol. We show that in contrast to the unmodified P2P system, the ISP-aided localized P2P system shows consistent improvements in the observed end-user experience, measured in terms of content download times, network locality of query responses and desired content, and quality of query responses. A significantly large portion of P2P traffic remains local to the ISP network, and ISPs notice a substantial reduction in overall P2P traffic. This can lead to immense cost savings for the ISPs [35]. The oracle consistently shows performance gains even across different topologies under a broad range of user behavior scenarios. For a more detailed analysis of the P2P oracle service, see [11, 9, 10].

11.1.1 Influence on P2P Topology

To explore the influence of consulting the oracle on the network topology we visualize, in Figure 16 [171], the Gnutella overlay topology. At a particular instant in time, we sample the Gnutella overlay topology, display all the online nodes in the graph, and join two nodes with an edge if there exists a Gnutella peering between them at this point of time. The resulting graph structures are displayed in Figure 16. For our simulations we consider 5 different topologies: Germany, USA, World1, World2 and World3, each modeled after their respective AS topologies (World1-3 differ in the size of the ASes). We can easily observe that the Gnutella topology in the biased case is well correlated with the Internet AS topology, where the nodes within an AS form a dense cluster, with only a few connections going to nodes in other ASes. This is in stark contrast to the unbiased Gnutella graph, where no such property can be observed. Multiple runs of the above experiments, using the different topologies yield similar results.
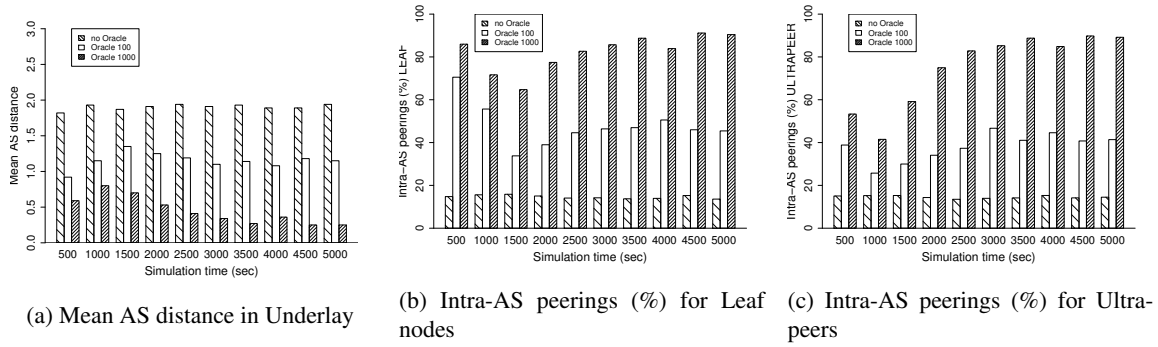
(a) Mean AS distance in Underlay

(b) Intra-AS peerings (%) for Leaf nodes

(c) Intra-AS peerings (%) for Ultra-peers

Figure 17: Metrics for Gnutella simulations

11.1.2 Benefits of Collaboration

Mean AS distance:
The benefits of using an oracle for biasing the neighborhood in Gnutella are visible in Figure 17a, which shows the average AS distance (in the underlay) between any two connected Gnutella nodes. The AS distance is obtained as follows. We map each Gnutella node's IP address to its parent AS, and for each overlay edge, we find the network distance in AS hops between the two end-nodes. We observe that the least amount of decrease in the average AS distance occurs from 1.93 to 0.8 at 1000 seconds, and the maximum decrease from 1.94 to 0.25 happens at 5000 seconds. Given that the AS diameter remains constant at 4 hops, the average decrease of 1.45 in the AS distance is significant. Besides, as the average AS distance in the case of oracle list size of 1000 is 0.45, a value less than 1, it implies that most of the Gnutella peerings are indeed within the ASes, i.e., they are not crossing AS boundaries. This can be a major relief for ISPs, as they do not incur any additional cost for traffic within their domains. Also traffic that does not leave the network is easier to manage. Moreover, P2P traffic will not encounter inter-ISP bottlenecks.

Intra-AS P2P connections:
The above observations on AS distance are even better understood from the plots in Figures 17b and 17c, where we show the total number of intra-AS P2P connections in the Gnutella network as a percentage of the total number of intra- and inter-AS P2P connections, for both leafs and ultrapeers.

In Figure 17b, we observe that in the case of leaf nodes, taking the average over the 10 time points, the percentage of intra-AS P2P connections increases from 14.6% in unbiased case to 47.88% in the case of oracle with list size 100. For oracle with list size 1000, we note an average of 82.22% intra-AS P2P connections.

In Figure 17c, we observe similar results for ultrapeers. The percentage of intra-AS P2P connections increases from an average value of 14.54% in the unbiased case to 38.04% in the case of oracle with list size 100, and further to 74.95% in case of oracle with list size 1000.

The percentage increase in intra-AS P2P connections is larger for leaf nodes as compared to ultrapeers, a welcome development. One needs a certain number of inter-AS connections, to maintain network connectivity and to be able to search for file content that may not be available within an AS. However, as leaf nodes typically have poor connectivity to the Internet, and have lower uptimes, it is reasonable to have leaf nodes keep most of their peerings within their AS, while allowing the ultrapeers to have slightly more connections outside their ASes.

For the impact of Oracle on download time under different topologies we refer the reader to [7]. For the impact of Oracle-like localization techniques on the inter-AS traffic flow of the BitTorrent P2P system we refer the reader to [49, 32, 139, 159].

11.2 Use Case: Traffic Engineering

The growth of demand for content and the resulting deployment of content delivery infrastructures pose new challenges to CDIs and to ISPs. For CDIs, the cost of deploying and maintaining such a massive infrastructure has significantly increased during the last years [148] and the revenue from delivering traffic to end-users has decreased due to the intense competition. Furthermore, CDIs struggle to engineer and manage their infrastructures, replicate content based on end-user demand, and assign users to appropriate servers.

The latter is challenging as end-user to server assignment is based on inaccurate end-user location information [119, 47], and inferring the network conditions within an ISP without direct information from the network is difficult. Moreover, due to highly distributed server deployment and adaptive server assignment, the traffic injected by CDIs is volatile. For example, if one of its locations is overloaded, a CDI will re-assign end-users to other locations, resulting in large traffic shifts in the ISP network within minutes. Current traffic engineering by ISP networks adapts the routing and operates on time scales of several hours, and is therefore too slow to react to rapid traffic changes caused by CDIs.

The pressure for cost reduction and customer satisfaction that both CDIs and ISPs are confronted with, coupled with the opportunity that distributed server infrastructures offer, motivate us to propose a new tool in the traffic engineering landscape. We introduce *Content-aware Traffic Engineering* (CaTE). CaTE leverages the location diversity offered by CDIs and, through this, it allows to adapt to traffic demand shifts. In fact, CaTE relies on the observation that by selecting an appropriate server among those available to deliver the content, the path of the traffic in the network can be influenced in a desired way. Figure 18 illustrates the basic concept of CaTE. The content requested by the client is in principle available from three servers (A, B, and C) in the network. However, the client only connects to one of the network locations. Today, the decision of where the client will connect to is solely done by the CDI and is partially based on measurements and/or inference of network information and end-user location. With CaTE the decision on end-user to server assignment can be done jointly between the CDI and ISP.

11.2.1 The CaTE Approach

CaTE complements existing traffic engineering solutions [18, 53, 94, 161, 180, 182] by focusing on traffic demands rather than routing. Let $\mathbf{y}$ be the vector of traffic counts on links and $\mathbf{x}$ the vector of traffic counts in origin-destination (OD) flows in the ISP network. Then $\mathbf{y}=A\mathbf{x}$, where A is the routing matrix. $A_{ij} = 1$ if the OD flow i traverses link j and 0 otherwise. Traditional traffic engineering is the process of adjusting A, given the OD flows $\mathbf{x}$, so as to influence the link traffic $\mathbf{y}$ in a desirable way. In CaTE, we revisit traffic engineering by focusing on traffic demands rather than routing changes. Content-aware Traffic Engineering (CaTE) is thus the process of adjusting the traffic demand vector $\mathbf{x}$, without changing the routing matrix A, so as to change the link traffic $\mathbf{y}$ in a desirable way.

CaTE offers additional traffic engineering capabilities to both ISPs and CDNs to better manage the volatility of content demand in small time scales. Traditional traffic engineering [18, 53, 94, 161, 180, 182] relies on changes of routing weights that take place in the time scale of hours [71]. On the contrary, in CaTE, the redirection of end-users to servers can take place per request or within the TTL of a DNS query that is typically tens of seconds in large CDNs [145]. Thanks to the online recommendations by ISP
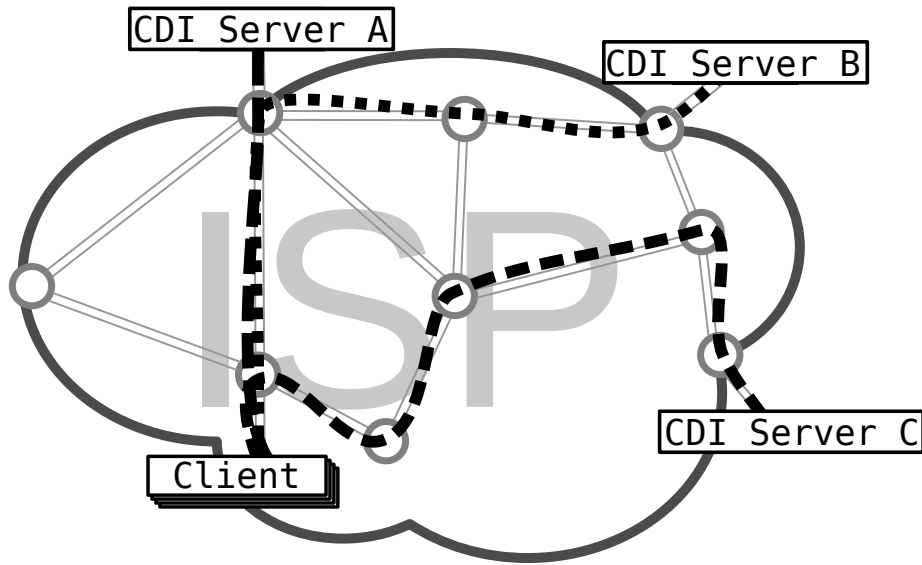
Figure 18: By choosing a CDI server for a client with the help of CaTE, traffic engineering goals and accurate end-user server assignment become possible. Reprinted from [76]. Included here by permission.

networks, CDNs gain the ability to better assign end-users to servers and better amortize the deployment and maintenance cost of their infrastructure. Network bottlenecks are also circumvented and thus the ISP operation is improved. Furthermore, the burden of measuring and inferring network topology, and the state of the network, both challenging problems, is removed from the CDNs. Moreover, in [74, Sections 4 and 5] we show that the online CaTE decisions on the end-user to server assignment leads to optimal traffic assignment within the network under a number of different metrics. The advantage is that now the problem of assigning traffic to links reduces to a fractional solution (on the contrary, assigning routing weights to links is NP-hard). In short, all involved parties, including the end-users, benefit from CaTE, creating a win-win situation for everyone.

11.2.2 A Prototype to Support CaTE

CaTE relies on a close collaboration between CDN and ISP in small time scales (seconds or per request). To achieve this goal, network information has to be collected and processed by the ISP. Candidate CDN servers have to be communicated to the ISP and ranked based on a commonly agreed criteria, e.g., to optimize the delay between the end-user and the CDN server. Today, there is no system to support the above operations. This motivate us to design, implement and evaluate a novel and scalable system that can support CaTE. In this section we describe the architecture and deployment of our working prototype to enable CaTE. We start by presenting our prototype in Section 11.2.2. We then comment on its operation and deployment within the ISP, its interaction with a CDN, and its performance that is beyond the state-of-the-art [18].

Architecture:
The CaTE system is installed in an ISP and interacts with the existing CDN server selector. The main tasks of the CaTE system are to: (1) maintain an up-to-date annotated map of the ISP network and its properties, (2) produce preference rankings based on the paths between end-users and candidate servers, and (3) communicate with the CDN server selection system to influence the assignment of end-user to servers. To
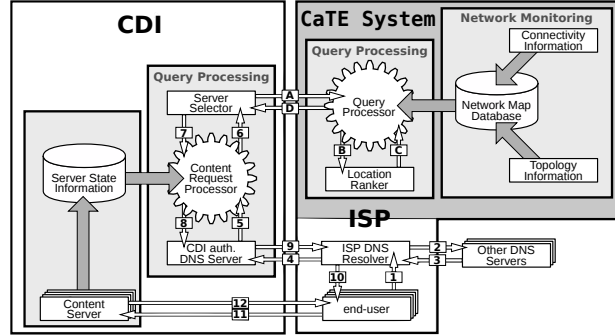
348

Figure 19: CaTE System architecture and flow of messages. Reprinted from [76]. Included here by permission.

this end, we propose an architecture that comprises a *Network Monitoring* component, a *Query Processing* component and a *communication interface* between an ISP and a CDN. For an overview of the architecture see Figure 19.

Network Monitoring:
The network monitoring component gathers information about the topology and the state of the network from several sources to maintain an up-to-date view of the network. The network monitoring component consists of the following subcomponents:

The **Topology Information** component gathers detailed information about the basic network topology, i.e., routers and links, as well as annotations such as link utilization, router load, and topological changes. An Interior Gateway Protocol (IGP) listener provides up-to-date link-state (i.e., IS-IS, OSPF) information. Information about routers and links is retrieved, thus, the network topology can be extracted. The nominal link delay, i.e., the latency on a link without queuing, can be found through the link length and physical technology. The link utilization and other metrics can be retrieved via SNMP from the routers or an SNMP aggregator.

The **Connectivity Information** component uses routing information to calculate the paths that traffic takes through the network. Finding the path of egress traffic can be done by using a Border Gateway Protocol (BGP) listener. Ingress points of traffic into the ISP network can be found by utilizing Netflow data. This allows for complete forward and reverse path mapping inside the ISP. Furthermore, the system can map customers as well as CDN infrastructures into the network map by finding the routers that announce the address space associated with them. In total, this allows for a complete path map between any two points in the ISP network. Finally, our system has access to an uplink database that provides information about the connectivity statistics of end-users.

The **Network Map Database** component processes the information collected by the *Topology* and *Connectivity Information* components to build an annotated network map of the ISP network tailored towards fast lookup on path properties. It uses a layer of indirection to keep the more volatile information learned from BGP separate from the slower changing topological information. This allows address space to be quickly reassigned without any reprocessing of routing or path information. It also enables pre-calculation of path properties for all paths that yields a constant database lookup complexity independent of path length and network architecture. If topology changes, e.g., IGP weights change or a link fails, the *Topology Information* component immediately updates the database which only recalculates the properties of the affected paths. Having ISP-centric information ready for fast access in a database ensures timely responses and high query throughput.

349

Query Processing:

The **Query Processing** component receives a description of a request for content from the CDN, which specifies the end-user making the request and a list of candidate CDN servers. It then uses information from the *Network Map Database* and a selected ranking function to rank the candidate servers. This component consists of the following subcomponents:

The **Query Processor** receives the query from the CDN. First, the query processor maps each source-destination (server to end-user) pair to a path in the network. In most cases, the end-user is seen as the ISP DNS resolver, unless both ISP and CDN support the client IP eDNS extension [47]. Once the path is found, the properties of the path are retrieved. Next, the pairs are run individually through the location ranker subcomponent (see below) to get a preference value. Finally, the list is sorted by preference values, the values are stripped from the list, and it is sent back to the CDN.

The **Location Ranker** component computes the preference value for individual source-destination pairs based on the source-destination path properties and an appropriate function. Which function to use depends on (a) the CDN, (b) what metrics the CDN asked for and (c) the optimization goal of the ISP. The preference value for each source-destination pair is then handed back to the Query Processor. Multiple such optimization functions being defined upon the collaboration agreement and subsequently selected individually in each ranking request. For example, a function might be the minimization of end-user and server delay. In [74] we evaluate CaTE with multiple ranking functions for different optimization goals.

Communication Interfaces:

When a CDN receives a content request, the *Server Selector* needs to choose a content server to fulfill this request. We propose that the server selector sends the list of eligible content servers along with the source of the query and an optimization goal to the ISP's CaTE system to obtain additional guidance about the underlying network. If the guidance is at granularity of a single DNS request, we propose a DNS-like protocol using UDP to prevent extra overhead for connection management. If the granularity is at a coarser level, i.e., seconds or even minutes, we rely on TCP.

11.2.3 Privacy and Performance

During the exchange of messages, none of the parties is revealing any sensitive operational information. CDNs only reveal the candidate servers that can respond to a given request without any additional operational information (e.g., CDN server load, cost of delivery or any reason why a server is chosen). The set of candidate servers can be updated per request or within a TTL that is typically in the order of a tens of seconds in popular CDNs [145]. On the other side, the ISP does not reveal any operational information or the preference weights it uses for the ranking. In fact, the ISP only re-orders a list of candidate servers provided by the CDN. This approach differs significantly from [18, 182], where partial or complete ISP network information, routing weights, or ranking scores are publicly available. We argue that an important aspect to improve content delivery is to rely on up-to-date information during server selection of the CDN. This also eliminates the need of CDNs to perform active measurements to infer the conditions within the ISP that can add overhead to CDN operation and may be inaccurate. With CaTE, the final decision is still made by the CDN, yet it is augmented with up-to-date network guidance from the ISP.

To improve the performance of our system, we do not rely on XML-based network maps as proposed in [18], but on light protocols that are close to DNS in design. This design choice is important as topology information in large networks (in the order of multiple MBytes). Transferring this information periodically to many end-users is likely to be challenging. In a single instance of our system, we manage to reply to up

to $90,000$ queries/sec when 50 candidate servers supplied by the CDN. At this level, the performance of our system is comparable to that of current DNS servers, such as BIND. However, the number of replies drops to around $15,000$ per second when considering 350 candidate servers. The additional response time when our system is used is around 1 ms when the number of candidate servers is 50 and around 4 ms when considering 350 candidate servers. This overhead is small compared to the DNS resolution time [3]. The performance was achieved on a commodity dual-quad core server with 32 GB of RAM and 1GBit Ethernet interfaces. Furthermore, running additional servers does not require any synchronization between them. Thus, multiple servers can be located in different places inside the network.

Deployment:
Deploying the system inside the ISP network does not require any change in the network configuration or ISP DNS operation. Our system solely relies on protocol listeners and access to ISP network information. Moreover, no installation of special software is required by end-users. The CaTE system adds minimal overhead to ISPs and CDNs. It only requires the installation of a server in both sides to facilitate communication between them.

Typically, an ISP operates a number of DNS resolvers to better balance the load of DNS requests and to locate DNS servers closer to end-users. To this end, we envision that the ISP's CaTE servers can be co-located with DNS resolvers in order to scale in the same fashion as DNS. CaTE servers can also be located close to peering points in order to reduce the latency between the CDN and an instance of the system. Synchronization of multiple CaTE instances is not necessary as they are aware of the state of the same network. We concluded that this is the best deployment strategy, other possible deployment strategies we have considered are presented in [74].

Operation:
We now describe the operation of our working prototype and its interaction with the CDN. In Figure 19 we illustrate the basic system architecture to support CaTE including the flow of information when the CaTE system is used. When a DNS request is submitted by an end-user to the ISP DNS resolvers *(1)* there are a number of recursive steps *(2)* until the authoritative DNS server is found *(3)*. Then, the ISP DNS resolver contacts the authoritative DNS server *(4)*. There, the request is handed to the content request processor operated by the CDN query processing component *(5)*. The content request processor has access to full information about the status of the CDN. Based on the operational status of the CDN servers, the server selection system [135] is responsible for choosing eligible content servers *(6)*. In the end, a preference list of content servers is generated. At this point, the CDN server selector sends the list of eligible content servers *(A)* along with user information, such as the IP of the DNS resolvers or client and an optimization metric to ISP. The query processor of the ISP system ranks the list using the location ranker *(B)*. After all the elements have been processed, the query processor has an annotated list with preferences for the ISP *(C)*. The query processor sorts the list by the preference values, strips the values and sends the list back to the CDN *(D)*. The CDN server selector incorporates the feedback, selects the best content server(s) and hand them back to the content request processor *(7)*. Then, the answer travels the path back to the client, i.e. from the CDN's authoritative DNS server *(8)* via the ISP DNS resolver *(9)* to the end-user *(10)*. Finally, the end-user contacts the selected server *(11)* and downloads the content *(12)*.

11.2.4 Modeling CaTE

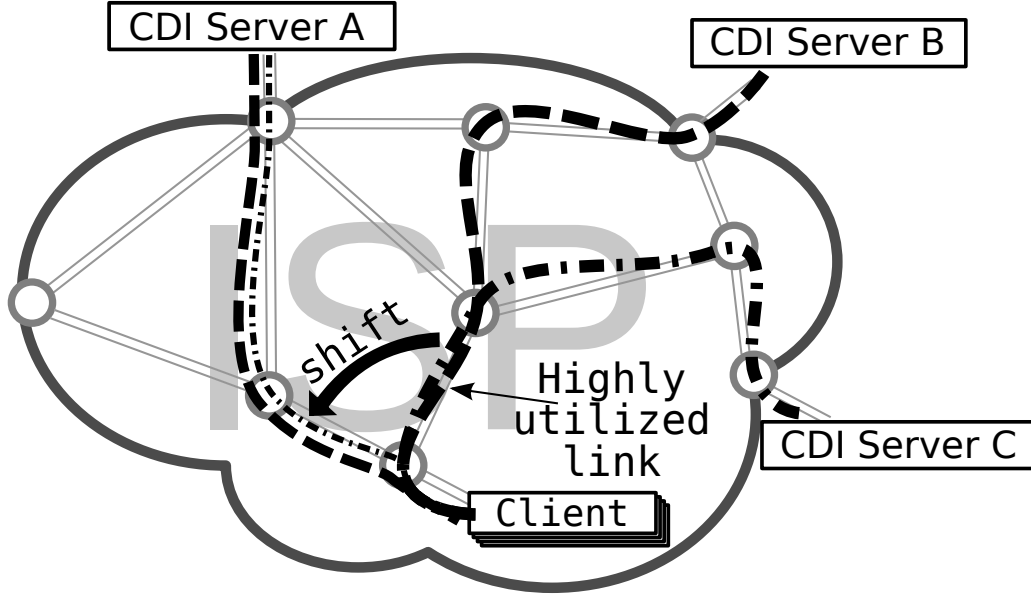Next, we formalize CaTE and discuss how it relates to traditional traffic engineering and multipath routing.

Figure 20: Content-aware Traffic Engineering Process. Reprinted from [76]. Included here by permission.

Architecture:
We model the network as a directed graph $G(V, E)$ where V is the set of nodes and E is the set of links. An origin-destination (OD) flow f_{od} consists of all traffic entering the network at a given point $o \in V$ (origin) and exiting the network at some point $d \in V$ (destination). The traffic on a link is the superposition of all OD flows that traverse the link.

The relationship between link and OD flow traffic is expressed by the routing matrix A. The matrix A has size $|E| \times |V|^2$. Each element of matrix A has a boolean value. $A_{ml} = 1$ if OD flow m traverses link l, and 0 otherwise. The routing matrix A can be derived from routing protocols, e.g., OSPF, ISIS, BGP. Typically, A is very sparse since each OD flow traverses only a very small number of links. Let **y** be a vector of size $|E|$ with traffic counts on links and **x** a vector of size $|V|^2$ with traffic counts in OD flows, then **y**=A**x**. Note, **x** is the vector representation of the traffic matrix.

Traditional Traffic Engineering: In its broadest sense, traffic engineering encompasses the application of technology and scientific principles to the measurement, characterization, modeling, and control of Internet traffic [27]. Traditionally, traffic engineering reduces to controlling and optimizing the routing function and to steering traffic through the network in the most effective way. Translated into the above matrix form, traffic engineering is the process of adjusting A, given the OD flows **x**, so as to influence the link traffic **y** in a desirable way, as coined in [108]. The above definition assumes that the OD flow vector **x** is known. For instance, direct observations can be obtained, e.g., with Netflow data [42, 63].

Terminology: We denote as *flow* an OD flow between two routers in the network. We call a flow *splittable* if arbitrarily small pieces of the flow can be assigned to other flows. This is not to be confused with end-to-end sessions, i.e., TCP connections, which are *un-splittable*. The assumption that flows are splittable is reasonable, as the percentage of traffic of a single end-to-end session is small compared to that of a flow between routers. Let C be the set of nominal capacities of the links in the network G. We denote as *link utilization* the fraction of the link capacity that is used by flows. We denote as *flow utilization* the maximum link utilization among all links that a flow traverses. We introduce the terms of *traffic consumer*

and *traffic producer* which refer to the aggregated demand of users attached to a router, and the CDIs that are responsible for the traffic respectively. We refer to the different alternatives from which content can be supplied by a given CDI as *network locations* that host servers.

Definition of CaTE:
We revisit traffic engineering by focusing on the traffic demands rather than changing the routing.
Definition 1: Content-aware Traffic Engineering(CaTE) is the process of adjusting the traffic demand vector $\mathbf{x}$, given a routing matrix A, so as to change the link traffic $\mathbf{y}$.

Not all the traffic can be adjusted arbitrarily. Only traffic for which location diversity is available can be adjusted by CaTE. Therefore, $\mathbf{x}=\mathbf{x}_r+\mathbf{x}_s$ where $\mathbf{x}_r$ denotes the content demands that can be adjusted and $\mathbf{x}_s$ denotes the content demands that can not be adjusted as there is only a single location in the network where the content can be downloaded from. The amount of traffic that can be adjusted depends on the diversity of locations from which the content can be obtained. We can rewrite the relation between traffic counts on links and traffic counts in flows as follows: $\mathbf{y}=A(\mathbf{x}_s + \mathbf{x}_r)$. CaTE adjusts the traffic on each link of the network by adjusting the content demands $\mathbf{x}_r$: $\mathbf{y}_r=A\mathbf{x}_r$. Applying CaTE means adjusting the content demand to satisfy a traffic engineering goal.
Definition 2: Optimal Traffic Matrix is the new traffic matrix, $\mathbf{x}^*$, after applying CaTE, given a network topology G, a routing matrix A and an initial traffic matrix $\mathbf{x}$.

Figure 20 illustrates the CaTE process. A content consumer requests content that three different servers can deliver. Let us assume that, without CaTE, the CDI redirects the clients to servers B and C. Unfortunately, the resulting traffic crosses a highly-utilized link. With CaTE, content can also be downloaded from server A, thus, the traffic within the network is better balanced as the highly utilized link is circumvented.

Minimizing the maximum utilization across all links in a network is a popular traffic engineering goal [70, 71, 112]. It potentially improves the quality of experience and postpones the need for capacity increase. CaTE mitigates bottlenecks and minimizes the maximum link utilization by re-assigning parts of the traffic traversing heavily loaded paths. Thus it redirects traffic to other, less utilized paths. Later in this chapter, we will elaborate in Section 11.2.5, different metrics such as path length or network delay can also be used in CaTE.

CaTE and Traditional TE:
CaTE is complementary to routing-based traffic engineering as it does not modify the routing. Routing-based traffic engineering adjusts routing weights to adapt to traffic matrix changes. To avoid micro-loops during IGP convergence [72], it is common practice to only adjust a small number of routing weights [71]. To limit the number of changes in routing weights, routing-based traffic engineering relies on traffic matrices computed over long time periods and offline estimation of the routing weights. Therefore, routing-based traffic engineering operates on time scales of hours, which can be too slow to react to rapid change of traffic demands. CaTE complements routing-based traffic engineering and can influence flows at shorter time scales by assigning clients to servers on a per request basis. Thus, CaTE influences the traffic within a network online in a fine-grained fashion.

CaTE and Multipath Routing:
Multipath routing helps end-hosts to increase and control their upload capacity [100]. It can be used to minimize transit costs [82]. Multipath also enables ASes to dynamically distribute the load inside networks in the presence of volatile and hard to predict traffic demand changes [63, 57, 97, 66]. This is a significant advantage, as routing-based traffic engineering can be too slow to react to phenomena such as flash crowds. Multipath takes advantage of the diversity of paths to better distribute traffic.

CaTE also leverages the path diversity, and can be advantageously combined with multipath to further improve traffic engineering and end-user performance. One of the advantages of CaTE is its limited investments in hardware deployed within an ISP. It can be realized with no change to routers, contrary to some of the previous multipath proposals [97, 57, 66]. The overhead of CaTE is also limited as no state about individual TCP connections needs to be maintained, contrary to multipath [97, 57, 66]. In contrast to [57, 97], CaTE is not restricted to MPLS-like solutions and is easily deployable in today's networks.

CaTE and Oscillations:
Theoretical results [68, 67] have shown that load balancing algorithms can take advantage of multipath while provably avoiding traffic oscillations. In addition, their convergence is fast. Building on these theoretical results, Fischer et al. proposed REPLEX [66], a dynamic traffic engineering algorithm that exploits the fact that there are multiple paths to a destination. It dynamically changes the traffic load routed on each path. Extensive simulations show that REPLEX leads to fast convergence, without oscillations, even when there is lag between consecutive updates about the state of the network. CaTE is derived from the same principles and thus inherits all the above-mentioned desired properties.

11.2.5 Potential of Collaboration

In this section, we quantify the potential benefits of CaTE when deployed within an European Tier-1 ISP using operational data.
Experimental Setting: To evaluate CaTE, an understanding of the studied ISP network is necessary, including its topological properties and their implications on the flow of traffic. Indeed, the topological properties of the ISP network influence the availability of disjoint paths, which are key to benefit from the load-balancing ability of CaTE. Because CaTE influences traffic aggregates inside the ISP network at the granularity of requests directed to CDIs, fine-grained traffic statistics are necessary. Traffic counts per-OD flow, often used in the literature, are too coarse an input for CaTE.
Data from a Large European ISP: To build fine-grained traffic demands, we rely on anonymized packet-level traces of residential DSL connections from a large European Tier-1 ISP, henceforth called *ISP1*. For ISP1, we have the complete annotated router-level topology including the router locations as well as all public and private peerings. ISP1 contains more than 650 routers and 30 peering points all over the world. Using the same monitoring infrastructure as in Section 5.1, we collect a 10 days long trace of HTTP and DNS traffic starting on May 7, 2010. We observe 720 million DNS messages as well as more than 1 billion HTTP requests involving about 1.4 million unique hostnames, representing more than 35 TBytes of data. We note that more than 65% of the traffic volume is due to HTTP.

A large fraction of the traffic in the Internet is due to large CDIs, including CDNs, hyper-giants, and OCHs, as reported in earlier studies [79, 107, 145]. In Figure 21, we plot the cumulative fraction of HTTP traffic volume as a function of the CDIs that originate the traffic. For this, we regard a CDI as a organizational unit where all servers from the distributed infrastructure serve the same content, such as Akamai or Google. We rank the CDIs by decreasing traffic volume observed in our trace. Note that the x-axis uses a logarithmic scale. The top 10 CDIs are responsible for around 40% of the HTTP traffic volume and the top 100 CDIs for close to 70% of the HTTP traffic volume. The marginal increase of traffic is diminishing when increasing the number of CDIs. This shows that collaborating directly with a small number of large CDIs, can yield significant savings.

In Figure 22 we plot the traffic of the top 1, 10, 100 CDIs by volume as well as the total traffic over time normalized to the peak traffic in our dataset. For illustrative purposes, we show the evolution across the first 60 hours of our trace. A strong diurnal pattern of traffic activity is observed. We again observe that a small
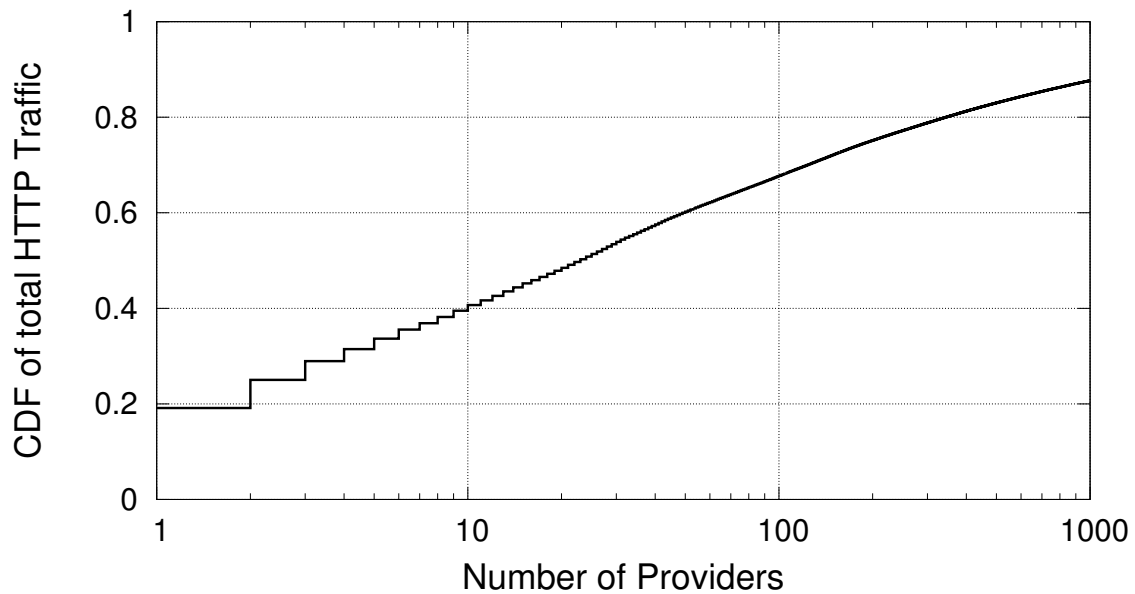
Figure 21: CDF of traffic volume of CDIs in ISP1. Reprinted from [76]. Included here by permission.
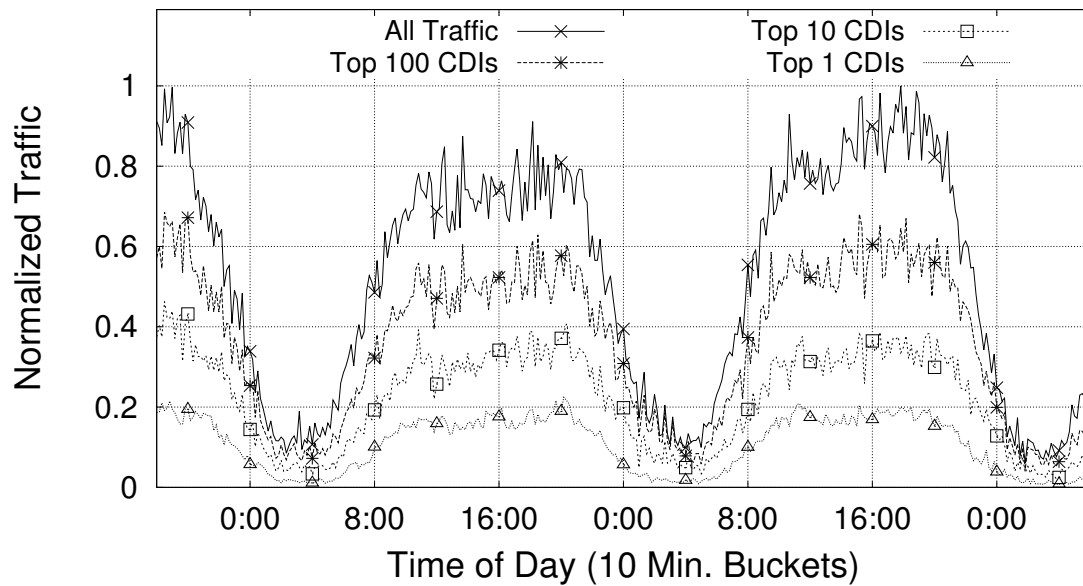

Figure 22: Normalized traffic for top CDIs by volume in ISP1. Reprinted from [76]. Included here by permission.

number of CDIs are responsible for about half of the traffic. Similar observations are made for the rest of the trace.

Understanding the Location Diversity of CDIs: To achieve traffic engineering goals, it is crucial to also
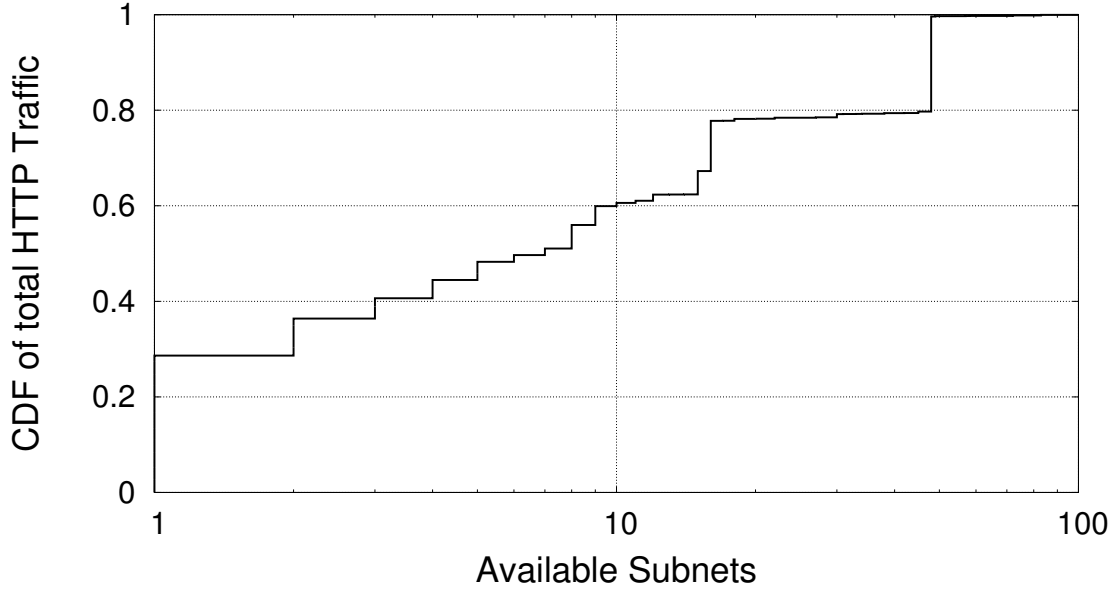
Figure 23: Subnet diversity from which content is available.

understand the location diversity of the top CDIs, as CaTE relies on the fact that the same content is available at multiple locations. Traffic originated from multiple network locations by a given CDI is seen by CaTE as a single atomic traffic aggregate to be engineered. Furthermore, as routing in the Internet works per prefix, we assume that the granularity of subnets is the finest at which CaTE should engineer the traffic demand. Thus, we differentiate candidate locations of CDIs by their subnets and quantify the location diversity of CDIs through the number of subnets from which content can be obtained.

We examine the amount of location diversity offered by CDIs based on traces from ISP1. To identify the subnets of individual CDIs, we rely on a similar methodology to the one from Poese et al. [145]. Our granularity is comparable to their "infrastructure redirection aggregation". Figure 23 shows the cumulative fraction of HTTP traffic as a function of the number of subnets (logarithmic scale) from which a given content can be obtained, over the entire 10 days of the trace. We observe that more than 50% of the HTTP traffic can be delivered from at least 8 different subnets, and more than 60% of the HTTP traffic from more than 3 locations. These results confirm the observations made in [145].

Dynamics in Location Diversity: So far the location diversity of CDIs has been evaluated irrespective of time. To complement the finding, we turn our attention to the location diversity exposed by CDIs at small time-scales, i.e., in the order of minutes. To this end, we split the original trace into 10 minutes bins. Figure 24 shows the evolution of the number of exposed subnets of five of the top 10 CDIs by volume. Note that the diversity exposed by some CDIs exhibits explicit time of day patterns, while others do not. This can be due to the structural setup or the type of content served by the CDI. The exposed location diversity patterns, i.e., flat or diurnal, are representative for all CDIs with a major traffic share in our trace. We conclude that a significant location diversity is exposed by popular CDIs at any point in time, and is quite extensive during the peak hour.

Content Demand Generation: The location diversity is not a mere observation about CDIs deployment. It requires to revisit the mapping between a given content demand and the realized traffic matrix. Given the
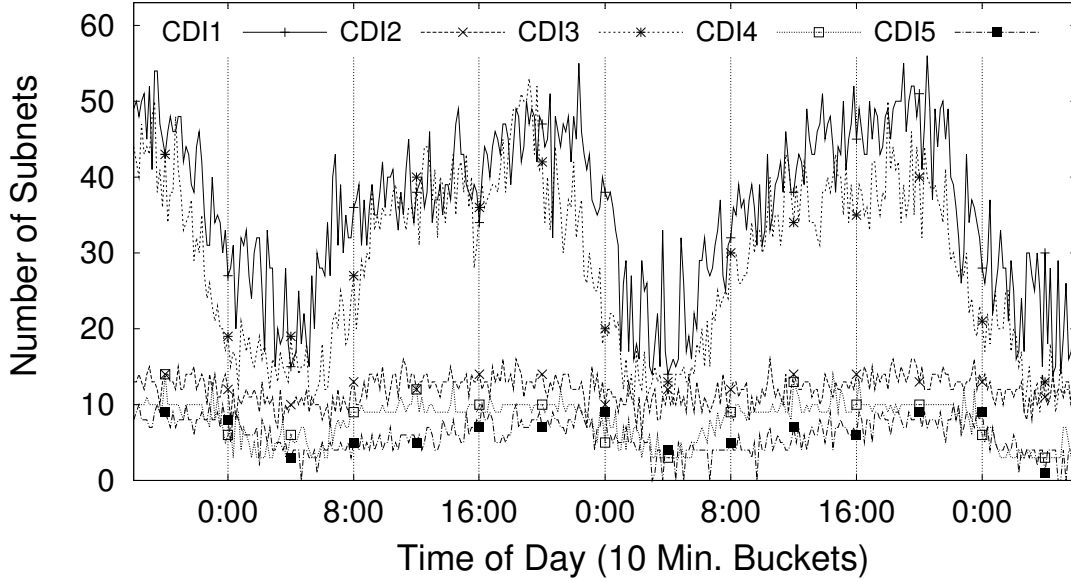
356

Figure 24: Evolution over time of number of subnets for selected CDIs in the top 10 CDIs. Reprinted from [76]. Included here by permission.

location diversity for content, multiple traffic matrices can be realized from a given content demand. The standard view of the OD flows therefore provides an incomplete picture of the options available for CaTE.

As an input for CaTE, we introduce an abstraction of the demand that reflects the available location diversity. We rely on the notion of *potential vectors*, that were denoted as x_r in Section 11.2.4. To generate the potential vector for a given CDI, the amount of traffic this CDI originates as well as the potential ingress points need to be known. Combining all potential vectors and x_s, we synthesize a network-wide content demand matrix for each time bin, by scaling the traffic demand to match the network utilization of ISP1. For our evaluation, we use the series of content demand matrices over a period of 10 days. The content demands are based exclusively on the HTTP traffic of our trace.

11.3 Summary

In this section we presented the potential of the *Oracle* and *Content-aware Traffic Engineering* (CaTE), two collaborative approach to improve content delivery while achieving traffic engineering goals. Both leverage location diversity offered by P2P systems or CDIs. Moreover, CaTE enables dynamic adaption to traffic demand shifts. With CaTE/the oracle the decision on end-user to server/peer assignment can be done jointly between the CDI/P2P system and the ISP. Our analysis of operational data from a European Tier-1 ISP has shown ample opportunities for CaTE to improve content delivery as it is done today.

Through extensive experiments, we show that both P2P users and ISPs benefit from ISP-aided P2P locality, measured in terms of improved content download times, increased network locality of query responses and desired content, and overall reduction in P2P traffic. For a more detailed analysis of the possible improvements and additional background information on the parameter set and resulting improvements, we refer the reader to [11, 6].

In [76, 74, 75] we quantify the benefits of CaTE and consider one of the most popular traffic engineering

goals, namely minimizing the maximum utilization of the links in the network [70, 71]. Our evaluation shows that CaTE yields encouraging results, even when only a few large CDIs are collaborating with an ISP. In fact, even metrics that are not directly related to the optimization function of CaTE are improved. Besides significant improvements for the operation of ISP networks, the end-users to also benefit from these gains. This can be attributed to the decrease of delay as well as the reduced link utilization. In [74] we also consider other network metrics such as path length or path delay and the effect of other network topologies. We also outline how CaTE can aid in the deployment of popular large scale applications, e.g., NetFlix, by selecting strategic locations for caches and specific optimization goals to support their operation. With this we conclude the section on collaborative traffic engineering and continue to elaborate on our idea for "in-network server deployment" in the next chapter.

12   Future of Collaboration

PaDIS and CaTE are designed to enable cooperation between CDI and ISPs for the already deployed servers. Recent advances in virtualization offer CDIs additional degree of freedom to scale-up or shrink the footprint on demand. This can be done either by jointly deploying and operating new servers with the ISPs. In this section we formally introduce the design of on-demand services motivated by the recent announcement of major ISPs to support generic hardware-network appliances, also referred to as microdatacenters, and offer them to application, service, and content providers. We also provide the design and implementation of NetPaaS, a system to orchestrate the deployment of on-demand services inside microdatacenters, by utilizing the view of the ISP about the network and additional computation and storage resources inside the network.

12.1   The New Cloud: Microdatacenters Deep Inside the Network

Applications are increasingly relying on direct interactions with end-users and are very sensitive to delay [112]. Indeed, transaction delay is critical for online businesses [102]. Network delay and loss are important contributors. Today, large-scale service deployments are restricted by limited locations in the network, e.g., datacenters, peering locations, or IXPs. These locations are not necessarily ideal [113]. We point out that *selection of service location is critical and currently not flexible enough*. Services should be located close enough to, in terms of network distance, the clients. Since client demands are volatile and change across time, CDIs need agility [41]. They can improve their service quality by quickly allocating, de-allocating, and migrating resources on-demand where and when they are needed. Indeed, since delay and packet loss are among the critical metrics, the service may need to be deployed deep inside the network, as many ISPs do for IPTV services. This option is not yet available for non-ISP content delivery Infrastructures, e.g., for cloud services.

Currently, most services and networks are run by independent entities with different and often conflicting objectives. Lack of information about the other entity leads to suboptimal performance and resource allocation for both the CDI and the ISP. For example, CDIs implement sophisticated methods to infer network conditions to improve perceived end-user experience [135], e.g., active measurements within the ISPs. Yet, the information gleaned from these measurements is already available with far greater precision to the ISP. On the other hand, ISPs continuously upgrade their infrastructures without being able to efficiently engineer the CDI traffic flows [145]. Today, cooperation and/or partnership between providers is limited to, e.g., peering or lately direct interconnections with content delivery Infrastructures. This level of cooperation is too narrow to reduce operational costs, improve end-user experience, circumvent bottlenecks, handle flash crowds, and adapt to changing network conditions and user demands. This has led to initial discussions on
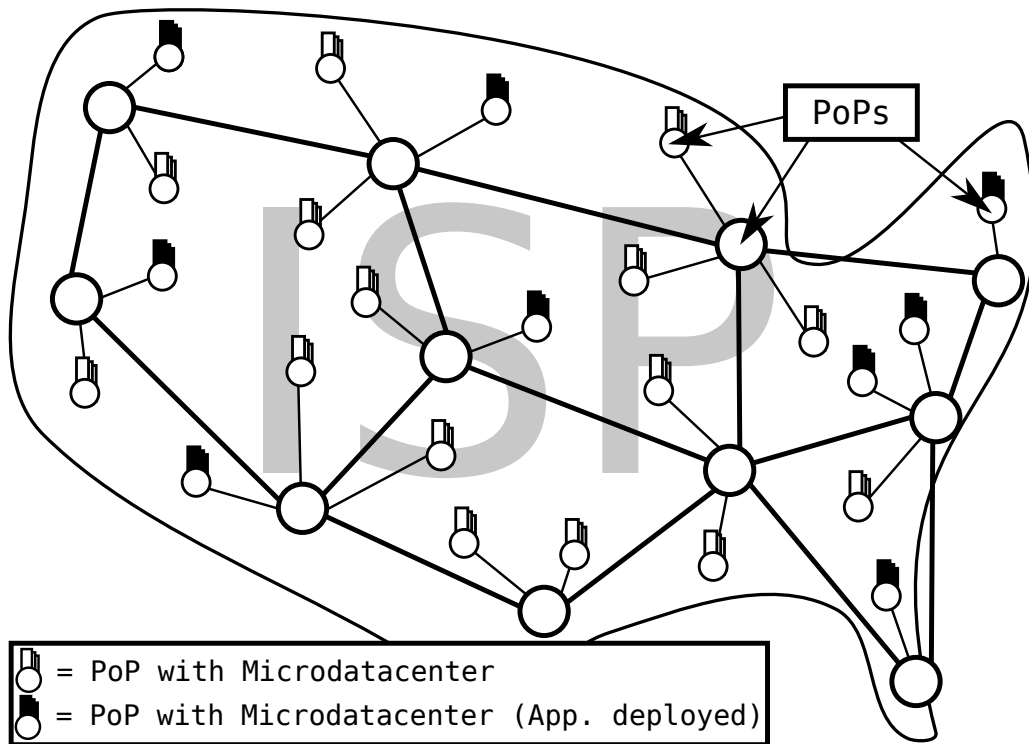
Figure 25: Microdatacenters in an ISP with NetPaaS enabled

how to improve communication between the various entities, e.g., within the IETF ALTO and CDNi working groups.

12.1.1 The ISPs Proposal

To overcome the above mentioned obstacles in service deployment and operation, major ISPs, including AT&T, Verizon, Deutsche Telekom, Telefonica, NTT, have proposed the use of cloud resources consisting of general purpose appliances that are co-located at network aggregation points inside the ISP. With the convergence of computing, storage, and communications, the acceptance of cloud services, and the ever increasing demand for popular services, ISPs are moving towards deploying general-purpose computing and storage infrastructures in their points of presences (PoPs). Henceforth, we refer to these as *microdatacenters*. The description of the functionality of these microdatacenters is provided in a white paper [132] that appeared in the SDN and OpenFlow World Congress in October 2012 and signed by 13 of the largest ISPs. Microdatacenters can be also the technical solution needed to materialize recent alliances of major CDIs, such as Akamai with large ISPs in the area of content delivery [12, 14, 15]. We notice that Software Defined Networks (SDNs) is another alternative to redirect traffic or perform traffic engineering when applied within an ISP or between and ISP and a CDN in cooperation. The comparison of the two approaches, NFV and SND, is out of the scope of this chapter and we refer the users to the related literature on SDN e.g., [36, 86, 122, 149, 93].

Figure 25 illustrates the basic idea. The ISP can offer *slices* within its microdatacenters, that can be leased by the CDIs—using our proposed mechanism—based on their needs. This approach leverages recent advances in virtualization technology, and flexible billing models, such as pay-as-you-go, to provide cost-efficient and scalable service deployment, enabling unprecedented flexibility. Moreover, the diversity of available service locations within the network can be used to improve end-user experience and makes it possible to launch even more demanding applications, such as interactive ones. On-demand service enables CDIs to rely on a fixed infrastructure deployment for their baseline operation and then scale it up by dynamically allocating resources closer to end-users. It also lowers the burden of entrance in the service market for smaller CDIs who might rely exclusively on the on-demand service at first.

12.1.2 Microdatacenter Specifications

Microdatacenters consist of one or more racks of off-the-shelf hardware deployed in general purpose rack space at network aggregation points. State-of-the-art solutions have been proposed by the VMware/Cisco/EMC VCE consortium [175], and are also offered by other vendors, such as NetApp and Dell. These solutions are general-purpose and provide a shared infrastructure for a large range of applications. They typically consist of two basic components: hardware and management software.

Hardware: Typical microdatacenters include *storage*, *computing*, *memory*, and *network access* components. Storage consists of tens of Terabytes with an ultra-fast controller providing I/O throughput in the order of hundreds of Gbps. The storage component is connected to the Internet through multi-Gbps interfaces and to the computing component with Gigabit Ethernet switches. Typically, a rack includes up to 40 physical multi-core blade servers as well as two routers and two switches in mesh configuration, for redundancy and load balancing.

Management Software: Each vendor offers a set of management tools not only for administering the components but also to create resource slices and to delegate the operation of the slices to external entities. This can be done per-server or via hardware supported virtualization[4]. The management software is also responsible for storage allocation and handling network resources, including IP address space. In addition, the management tools come with a monitoring interface that allows the ISP to monitor the utilization of the overall microdatacenter as well as the information for each slice that can be shared with the external entity.

An ISP can allocate resource slices consisting of computing, storage, memory, and network access in a microdatacenter and then delegate the operation of the slice to a CDI. This is what we refer to as the *ISPs cloud service* which is realized via resource slices in microdatacenters throughout the ISPs infrastructure.

12.1.3 Microdatacenter Network Footprint

Most ISPs' networks consist of an access network to provide Internet access to DSL and/or cable customers, as well as an aggregation network for business and/or VPN customers. Routers at this level are often referred to as *edge routers*. The access and aggregation networks are then connected to the ISP's backbone which consists of *core routers*. *Border routers* are core routers that are used to connect either to other networks or to co-location centers. Opportunities to deploy microdatacenters exist at each level: edge, core, or border router locations.

---

[4] For example, para-virtualization [48] presents the VM with an abstraction that is similar but not identical to the underlying hardware.

The advantage of deploying service infrastructure only at the core router locations is that there are a few large and well established locations. This is also a disadvantage as location diversity might be limited. Location diversity is highest at the edge router locations. However, it might not always be possible to deploy a microdatacenter, i.e., due to limited space and/or power at the facilities, or due to cost. These locations, however, minimize the distance to the customers. Border router locations are often a subset of core routers, hence they inherit the same advantages and disadvantages.

The advantage of using an ISP cloud service vs. a public cloud service for a CDI is the chance to minimize the distance to the end-user. on-demand service deployment allows the CDI to control the location of the slices and ensures that there are no major network bottlenecks.

12.2 On-Demand Service Design

An *on-demand service* is a service of the ISP (see Figure 25) that enables CDIs to use a hosting infrastructure that scales according to end-user demands, so as to minimize its capital expenditures and operating costs, as well as the distance between its hosting infrastructure and the source of the demand. Moreover, it offers an interface that enables the CDI to map user requests to appropriate slices in order to maximize slice utilization and minimize the distance between the end-user and the slices.

Definition 1: ISP On-Demand Service. The ISP on-demand service is a service offered by the ISP and uses as its base unit of resource allocation the notion of a microdatacenter *slice*. It is the ISP's task to allocate/de-allocate the slices since it operates the microdatacenter. The CDI requests slices based on its clients demand. When the slice is allocated to the CDI, the service can be installed on the slice. From that point on, the CDI fully controls the operation of the service installed in the microdatacenter. Negotiation about slices are done via the *on-demand service interface* through which CDI demands are matched to the ISPs resources. How to map demands to resources in an efficient manner is the task of the ISP and part of the *on-demand service realization*. In addition, the interface allows for access to the billing information. Moreover, the *on-demand service interface* enables the mapping of user requests to appropriate slices.

The above mentioned use of microdatacenters is in-line with the available primitives of private and public clouds operated in large-scale datacenters, e.g., [19, 125].

12.2.1 Microdatacenter Slice

Based on our description of microdatacenters in the previous sections, we define a slice as follows.

Definition 2: Slice. The slice of a microdatacenter is a set of physical or virtualized resources of a specific capacity, for each of the resources of the microdatacenter. The slice is delegated to the service provider that can install and operate its service using the resources of the slice.

For example, a slice can be a 1-core server with 2 GB RAM, 30 GB storage, a 1 Gbps Internet access bandwidth, 2 public IPs—an actual physical resource. Alternatively, it can be a VServer with 2GB and 1 Gbps Internet access bandwidth, 1 public IP, and a pre-installed OS—a virtual machine of a specific type. With the current management and virtualization tools available from microdatacenter vendors, it is possible to allocate/deallocate slices on-demand in with unprecedented degree of freedom, e.g., [25] and references within.

12.2.2 On-Demand Service Realization

Based on the above specification of on-demand service, the ISP has to implement two functions to offer its *on-demand service*: *mapping of service provider demands to slices* and *assigning users to slices*.

361

Note, the time scales at which these two services are expected to be used differ significantly. The first one allows the service provider to flexibly allocate and de-allocate its slices based on its forecast of demands, in those locations where it wants them. We foresee that requests for slices are not issued individually but rather collectively on a time scale of tens of minutes or hours.

The CDI provides the ISP with a set of demands for slice resources, predicted demand locations, desired slice locations, as well as optimization criteria. The ISP then has to map the demands to its microdatacenter resources. We expect that the major degree of freedom that the ISP uses to jointly optimize performance is the desired slice location. We refer to this optimization problem as the `slice location` problem. If the desired slice locations are fully specified or the predicted demand locations are missing, the `slice location` problem becomes trivial and the ISP only grants or denies the request.

At the second time scale, the ISP can help the CDI in assigning users to slices. Since the service is offered at multiple locations, a good assignment of users to slices impacts not only the load on the network but also the network delay and packet loss, which are key contributors to the user experience. Jointly optimizing this mapping is therefore of interest to both the ISP and the CDI. The CDI can query the ISP for each request on where to map it, based on the current set of slice assignments and service loads. The ISP then uses its network information to propose possible slices. We refer to this problem as the `user-slice assignment` problem, see [76].

Another degree of freedom on-demand service offers to the CDI is auto-scaling. While it is quite feasible to dimension applications, flash-crowds or device failures are hard to predict. To this end, a CDI may allow on-demand service to create replicas if its monitoring indicates that the capacity of the service at a given location is or will be exceeded. To realize this service, the ISP needs to constantly monitor resource availability and if necessary migrate or suggest the creation of additional slices. Moreover, it has to allow the CDI to monitor the utilization of its slices.

12.2.3 Service Interfaces

The ISP offers four interfaces to the content delivery Infrastructures:

Resource discovery: Using this interface the CDI requests information about resources, e.g., about available locations for slices and if in principle slices are available at those locations at what price.

Slice allocation: Using this interface the CDI requests slice allocation within a certain cost limit.

User-slice assignment: Using this interface the CDI requests recommendations for user demand to slice mapping.

Monitoring and billing: Using this interface the CDI monitors the status and cost of its slices.

In Section 12.3 we give specific examples of how these service interfaces can be used by a CDI and ISP to cooperate in order to improve their services.

12.2.4 Billing

It is important for the CDI to minimize and track the cost of its use of on-demand service. Depending on the scale of the services, the service provider has to pay the usual price or negotiate bilateral agreements with the ISP. Using the resource discovery interface, it estimates the cost of slice allocation at possible locations. Using the slice allocation interface, it can bound the total cost of the request.

We expect that the billing of a slice allocated via on-demand service follows that of large-scale datacenters. This means that there is an installation cost and a usage cost. The installation cost applies to a single slice in a microdatacenter and is charged only once or over long time intervals, e.g., hours, and is fixed.

The installation cost typically increases if additional licenses have to be leased, e.g., software licenses. The installation cost can depend on the location of the microdatacenter that hosts the slice or the time-of-day.

The usage cost follows a pay-as-you-go billing model and charges for the usage of different resources assigned to a slice. The billing among different resources in the same slice can be quite diverse. The slice can use expensive resources such as bandwidth or cheaper ones such as CPU.

For example, a slice may have a $0.01 per hour installation cost and a usage cost that depends on its use of various resources, e.g., $0.02 per real CPU usage per hour, $0.001 per GByte stored per hour, and $0.001 per Gbps outgoing traffic per hour. If the slice is idle, then only the installation cost is charged. Note, that if the slice is used for a short period within the allocation time, e.g., a few minutes, then the charge may apply to the minimum billing granularity.

To minimize the cost of deploying an on-demand service, the CDI can change its total slice demands as well as its slice specifications dynamically. Moreover, it can relax the slice specifications to reduce overall cost of its service deployment.

12.3   Network Platform as a Service (NetPaaS)

Next, we discuss the prototype system that has been proposed to materialize the On-demand service, Network Platform as a Service (NetPaaS). NetPaaS leverages the view of PaDIS and also utilize the knowledge about the status of the microdatacenters within the network. NetPaaS is also able to map the requests of CDIs to available microdatacenters to better match the demand with the resources inside the network. The granularity at which they are exchanged via the service interface. We also outline several possible protocols for the service interfaces. We focus on resource discovery, slice allocation, and user-slice assignment. We do not discuss monitoring and billing because they can be realized today using techniques similar to those in use by current public clouds, e.g., [25]. Due to space limitations, we refer the reader to [73] for a formalization of NetPaaS, as well as an evaluation on a CDI use case.

Recall that our assumption that the time scales at which the two principle components of on-demand service operate are different. On the one hand, resource discovery and slice allocation are expected to be done on time scales of tens of minutes, hours, or even days. On the other hand, user-slice assignment potentially happens on a per user request basis. Accordingly, the protocols differ. We propose to use out-of-band protocols for the first two service interfaces and in-band protocols for the third one.

12.3.1   Resource Discovery

The purpose of resource discovery is to provide the CDI with the ability to gather information about the resources offered by the ISP. Accordingly, we have two message types: CDI_Discovery_Request and ISP_Discovery_Response.

CDI_Discovery_Request: Is issued either without and with arguments. In the first case the response is the set of resources that are offered. In the second case the responds contains details about the resources named in the argument.

ISP_Discovery_Response: List of available resource or details about the resources specified in the argument.

So far we have not outlined at what granularity and specificity the resources are requested. This depends on the agreements between the CDI and the ISP. For example, the ISP may have no problem revealing its microdatacenter locations to a major CDI. However, it may not want to share this information with an
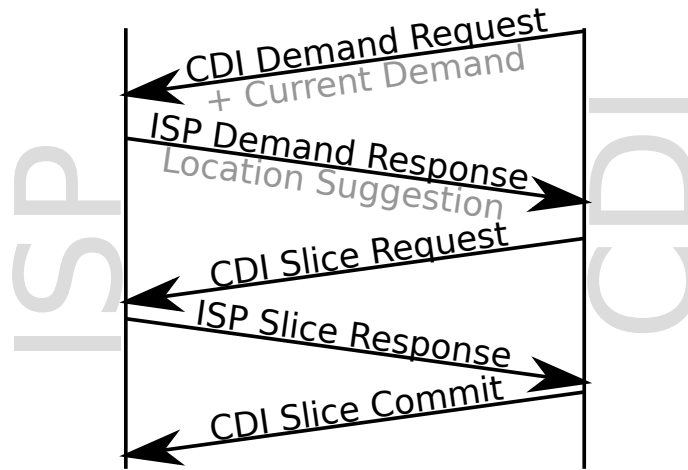
Figure 26: Slice allocation message exchange.

untrusted CDI that wants to run a single slice. For the latter, the region in which the microdatacenter is located might well suffice.

With regards to granularity, the ISP can specify which type of servers it is offering in each microdatacenter region, as is common for public cloud services [19], unless another agreement is in place that enables access to more specific information. With regards to the availability and/or price, the ISP can either return a base price, including installation and usage cost, to indicate that resources are available or offer an auction-based system. In the latter case, the discovery request returns information about past prices.

12.3.2   Slice Allocation

Slice allocation enables the CDI and ISP to cooperate for allocating slices in microdatacenters close to the end-user that are able to satisfy the demands. We envision five message types: CDI_Demand_Request, ISP_Demand_Response, CDI_ Slice_Request, ISP_Slice_Response, CDI_Slice_Commit. The first two message types enable the cooperation between the CDI and the ISP to allocate slices at appropriate microdatacenter by utilizing network information, see Figure 26. The last three messages enable a three way handshake between the CDI and the ISP to verify that the ISP is able to provide a specific slice for the CDI.

CDI_Demand_Request: Is submitted by the CDI to the ISP and contains a summary of the hardware resources the CDI wants together with optimization criteria, constraints, and a demand forecast, e.g., per region or per network prefix. Possible optimization criteria are to minimize network distance or the cost. The constrains include: number of locations, minimum resources per slice, etc.

ISP_Demand_Response: The ISP returns a set of proposed slice configurations and their price. It computes these by solving the slice location problem.

CDI_Slice_Request: The CDI either selects, based on its criteria, a set of proposed slices as returned by the ISP_Demand_Response, or it completes a specification of a slice set request using information it retrieved via the resource discovery interface. In addition, the request contains a maximum cost.

ISP_Slice_Response: Upon receiving a CDI_Slice_Request, the ISP checks if it can offer the set of slices at the requested cost. This involves solving another version of the slice location problem. If possible, the ISP returns the price otherwise it declines the request. At this step, the ISP reserves the

requested resources to guarantee their availability. Note, the ISP does not have to return precise slice definitions, e.g., instead of returning that slice x should be located in microdatacenter y attached to router z it only returns slice x should be located in region xyz.

CDI_Slice_Commit: This step confirms CDI_Slice_Requests. Upon receiving the commit from the CDI, the ISP allocates the slices and delegates their control to the CDI.

Now we discuss different ways in which a CDI and an ISP can cooperate using the above messages. These ways differ in which information is shared and with whom.

Minimum information exchange: The CDI uses the information from the ISP_Demand_Response for queries via CDI_Demand_Request with a uniform distributed demand vector. The responses include slice candidates with servers having specified hardware profiles and in specific regions. Then, the CDI scales the suggested slices according to its demand locations and uses the CDI_Slice_Request message to check if the ISP can offer it and at what price. Once it has found a satisfactory configuration it can use the CDI_Slice_Commit message to request the necessary slices.

Partnership 1: The CDI uses CDI_Demand_Request with a scaled demand CDI selects one of these and uses the ISP_Slice_Request message so that the ISP can reserve the resources. Upon successful reservation, the CDI_Slice_Commit message confirms the allocation of the slices.

Partnership 2: The CDI uses the CDI_Demand_Request without a demand vector but with specific resource requests. The ISP response specifies candidate microdatacenters with available resources. Then, the CDI uses its version of the `slice location` problem to find possible slice sets at a subset of these locations. Then, the CDI uses the ISP_Slice_Request message to see if the ISP can offer it and at what price. Once it finds a satisfactory configuration it uses the CDI_Slice_Commit message to allocate the slices.

The first scenario corresponds to the minimum information that has to be exchanged in order to reach a consensus on the locations and specification of the slices. The latter two summarize different forms of possible cooperations that can be agreed upon in bilateral agreements.

So far, we have assumed that there are no preallocated slices. However, this is typically not the case, and the actual task is to augment a preexisting set of slices in such a way as to best serve the predicted demand for the next time period. To enable this, another argument to each message request can be provided, indicating a set of already allocated resources and a penalty value for deallocating slices in one location and allocating them in another. This penalty is needed as part of the optimization problem. Basically, it indicates up to which point it is preferable to keep a suboptimal location because of stability of resource allocation vs. when to migrate the service to a new location. Based on the returned information, the CDI has the option of either moving slices using VM migration or to de-allocate and allocate new slices.

The ISP microdatacenter can offer VM migration and/or consolidation[5] with keeping the IP addresses only within the same microdatacenter location. Across microdatacenters it may only offer migration with tunnels which requires the CDI to temporarily operate two slices at both locations. However, the old one is a good candidate for consolidation so that it is possible to reduce the allocated resources to a minimum within a microdatacenter once all new requests are served by the newly allocated slices. Thus, if an ISP offers service consolidation, one option for CDIs that want to use diverse sets of microdatacenters is to always keep a minimal slice active at each location and expand or shrink it according to the demand.
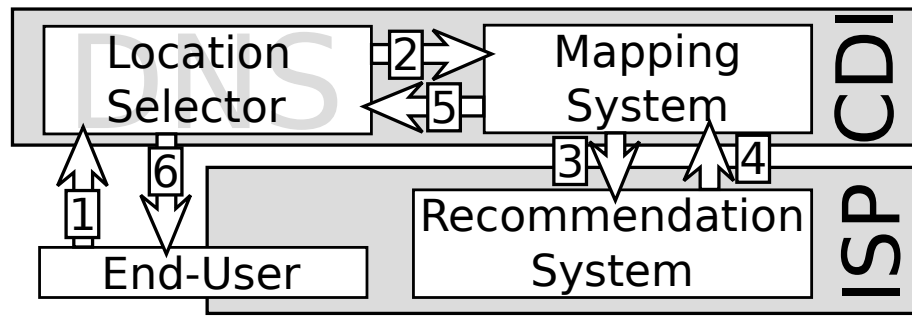
Figure 27: User-slice assignment schematic.

12.3.3 User-Slice Assignment

The purpose of the user-slice assignment interface is to enable small time scale interactions between the CDI and the ISP, to ensure that end-user demand is mapped to the appropriate slice. Therefore, the interface has to be integrated into the process used by the CDI to map user requests to CDI servers. Next, we first review how this is currently realized using DNS. Then, we discuss options on how the user-slice assignment interface can be integrated, see Figure 27.

CDI: User request to CDI server mapping. Before an end-user issues a request for the CDI service, e.g., downloading some content or watching a video, it issues a DNS request to map the hostname to an IP address of the server that hosts the service. This DNS request is sent to the ISPs DNS resolver or an alternative DNS resolver. This resolver contacts the authoritative DNS server of the CDI service, since caching is typically restricted by small TTL's. The authoritative DNS server uses a CDI service, the CDI mapping system, to select a CDI server from which to satisfy the future requests of this end-user. The CDI mapping system performs a CDI specific optimization. This optimization may consider the load on the CDI servers, the network location of the CDI server as well as the requesting DNS server, the price of network access at the CDI server, etc. Note, the CDI's authoritative DNS name servers usually does not have access to the IP address of the end-user as the request is forwarded via the DNS resolver unless the eDNS [47] standard is used. With eDNS, the client IP address or the IP prefix can be added to the DNS request sent by the DNS resolver to the authoritative DNS name server. In addition, the CDI can use HTTP redirection to further load balance within its infrastructure.

User-Slice Assignment: Option 1. Considering the process outlined above, one possible way to use the user-slice assignment interface is within the optimization that the CDI mapping system performs. For this case, we envision two message types: CDI_User-Slice_Assign_Request and ICDI_User-Slice_Assign_Response which correspond to steps 3 and 4 in Figure 27.

CDI_User-Slice_Assign_Request: Issued by the CDI's DNS server to the ISP. It contains the client IP address as well as slice locations within or outside of the ISP.
ISP_User-Slice_Assign_Response: The ISP responses with a ranking of the slice locations.

The previous two messages enable the CDI to consider information from the ISP, conveyed via the ranking, for its mapping. This is equivalent to the functionality and protocols proposed by the IETF ALTO working group. However, we envision a more light weight implementation. We propose to not rely on XML

---

[5]Here, consolidation corresponds to moving multiple VMs with minimal resource requirements to the same physical machine to keep a base service.

for encoding each request if the granularity of the requests is on a per connection level. If the CDI uses a coarser granularity such as subnet or region, the efficiency of the message exchange is even less critical.

User-Slice Assignment: Option 2. Another way to integrate the user-slice assignment interface within the above process is by pro-actively sharing information between the CDI and the ISP. For this case, we again envision two message types: ISP_User-Slice_Assign_Proposal and CDI_User-Slice_Assign_Ack.

ISP_User-Slice_Assign_Proposal: Is sent by the ISP to the CDI mapping system. It contains a set of IP prefixes each with an associated ranking of the different microdatacenter locations.
CDI_User-Slice_Assign_Ack: The CDI either acknowledges or rejects the proposal.

This again enables the CDI to include information from the ISP, conveyed via the ranking, in its mapping. For example, one can use BGP to send such messages—a mechanism Akamai already utilizes to aid in mapping users to its clusters [135].

12.4 Summary

Motivated by the high requirements newly deployed services put on the ISPs networks, we formally introduced the design of on-demand services to relieve both the CDI and the ISP. We also provided the design and implementation of NetPaaS, a system to orchestrate the deployment of on-demand services using the ISPs view of the network and available resources. In [73] we evaluate NetPaaS using operational traces from the biggest commercial CDN and a European Tier-1 ISP. We quantify the benefits of NetPaaS by utilizing different optimization goals e.g., delay reduction or reduced link utilization. Our results show that NetPaaS yields encouraging results for a joint deployment of new services that significantly improves the end-users performance while reducing the network utilization for the ISP and offering agile service deployment to the CDI. For the analysis and performance evaluation of on-demand server placement algorithms in wide-area networks we refer the reader to [165] and [22].

13 Conclusion

People value the Internet for the content and the applications it makes available [92]. For example, the demand for online entertainment and web browsing has exceeded 70% of the peak downstream traffic in the United States [154]. Recent traffic studies [79, 107, 145] show that a large fraction of Internet traffic is originated by a small number of Content Distribution Infrastructures (CDIs). Major CDIs include highly popular rich-media sites such as YouTube and Netflix, One-Click Hosters (OCHs), e.g., RapidShare, Content Delivery Networks (CDNs) such as Akamai, and hyper-giants, e.g., Google and Yahoo!. Gerber and Doverspike [79] report that a few CDIs account for more than half of the traffic in a US-based Tier-1 carrier.

To cope with the increasing demand for content, CDIs deploy massively distributed infrastructures [112] to replicate content and make it accessible from different locations in the Internet [172, 4]. Not all CDIs are built upon the same philosophy, design, or technology. For example, a CDI can be operated independently by deploying caches in different networks, by renting space in datacenters, or by building datacenters. Furthermore, some CDIs are operated by ISPs, some by Content Producers, or in the case of Peer-to-Peer networks, by self-organized end users. Accordingly, we give an overview of the spectrum of CDI solutions.

CDIs often struggle in mapping users to the best server, regardless of whether the best server is the closest, the one with the most capacity, or the one providing the lowest delay. The inability of CDIs to map end users to the right server stems from the fact that CDIs have limited visibility into ISP networks,

i.e., a CDI has incomplete knowledge of an ISP's set up, operation, and current state. Thus, in this book chapter, we propose viewing the challenges that CDIs and ISPs face as an opportunity: *to collaborate*. We point out the opportunities and incentives for all parties—CDIs, ISPs and end users—to get involved. This collaboration may ultimately lead to major changes in the way that content is distributed across the Internet.

Accordingly, we review the proposed enablers and building blocks for collaboration ranging from the P2P oracle service, P4P, Ono, and PaDIS, to the IETF activities [11, 182, 39, 145, 120, 133]. To illustrate the benefits of collaboration between applications and networks, we provide two use-cases: P2P and traffic engineering. The main take away is that substantial benefits for all involved parties are obtainable.

Upcoming trends include virtualization and the Cloud. These trends offer new ways of collaborative deployment of content delivery infrastructure if combined with the proposed enablers for collaboration. Accordingly, we propose Network Platform as a Service (NetPaaS), which allows CDIs and ISPs to cooperate not only on user assignment, but on dynamically deploying and removing servers and thus scaling content delivery infrastructure on demand.

We believe that ISP-CDN collaboration and NetPaaS can play a significant role in the future content delivery ecosystem. Most of the collaboration enablers, however, have not yet been deployed in the wild, and therefore only the future will tell if the Internet takes advantage of these opportunities.

14   Acknowledgments

References

[1] ADITYA, P., ZHAO, M., LIN, Y., HAEBERLEN, A., DRUSCHEL, P., MAGGS, B., AND WISHON, B. Reliable Client Accounting for Hybrid Content-Distribution Networks. In *Proc. NSDI* (2012).

[2] AGER, B., CHATZIS, N., FELDMANN, A., SARRAR, N., UHLIG, S., AND WILLINGER, W. Anatomy of a Large European IXP. In *Proc. SIGCOMM* (2012).

[3] AGER, B., MÜHLBAUER, W., SMARAGDAKIS, G., AND UHLIG, S. Comparing DNS Resolvers in the Wild. In *Proc. IMC* (2010).

[4] AGER, B., MÜHLBAUER, W., SMARAGDAKIS, G., AND UHLIG, S. Web Content Cartography. In *Proc. IMC* (2011).

[5] AGER, B., SCHNEIDER, F., KIM, J., AND FELDMANN, A. Revisiting Cacheability in Times of User Generated Content. In *Proc. IEEE Global Internet* (2010).

[6] AGGARWAL, V. *ISP-Aided Neighbour Selection in Peer-to-Peer Systems*. Doktorarbeit, Technische Universität Berlin, Berlin, Germany, 2009.

[7] AGGARWAL, V., AKONJANG, O., AND FELDMANN, A. Improving User and ISP Experience through ISP-aided P2P Locality. In *Global Internet* (2008).

[8] AGGARWAL, V., BENDER, S., FELDMANN, A., AND WICHMANN, A. Methodology for Estimating Network Distances of Gnutella Neighbors. In *GI Jahrestagung - Informatik 2004* (2004).

[9] AGGARWAL, V., AND FELDMANN, A. ISP-aided Biased Query Search for P2P Systems in a Testlab. In *European Conference on Complex Systems* (2007).

[10] AGGARWAL, V., FELDMANN, A., AND KARRER, R. An Internet Coordinate System to Enable Collaboration between ISPs and P2P Systems. In *ICIN* (2007).

[11] AGGARWAL, V., FELDMANN, A., AND SCHEIDELER, C. Can ISPs and P2P Systems Cooperate for Improved Performance? *ACM CCR 37*, 3 (2007).

[12] AKAMAI. Akamai and AT&T Forge Global Strategic Alliance to Provide Content Delivery Network Solutions. http://www.akamai.com/html/about/press/releases/2012/press_120612.html.

[13] Akamai Facts & Figures. http://www.akamai.com/html/about/facts_figures.html.

[14] AKAMAI. Orange and Akamai form Content Delivery Strategic Alliance. http://www.akamai.com/html/about/press/releases/2012/press_112012_1.html.

[15] AKAMAI. Swisscom and Akamai Enter Into a Strategic Partnership. http://www.akamai.com/html/about/press/releases/2013/press_031413.html.

[16] AKAMAI INC. SureRoute. www.akamai.com/dl/feature_sheets/fs_edgesuite_sureroute.pdf.

[17] AKELLA, A., SESHAN, S., AND SHAIKH, A. An Empirical Evaluation of Wide-Area Internet Bottlenecks. In *Proc. IMC* (2003).

[18] ALIMI, R., PENNO, R., AND YANG, Y. ALTO Protocol. draft-ietf-alto-protocol-15, May 2011.

[19] AMAZON WEB SERVICES. http://aws.amazon.com.

[20] ANDERSEN, D., BALAKRISHNAN, H., KAASHOEK, M., AND MORRIS, R. Resilient Overlay Networks. In *Proc. SOSP* (2001).

[21] ANDERSON, N. P2P traffic drops as streaming video grows in popularity. http://arstechnica.com/old/content/2008/09/p2p-traffic-drops-as-streaming-video-grows-in-popularity.ars, 2008.

[22] ANDREEV, K., GARROD, C., MAGGS, B., AND MEYERSON, A. Simultaneous Source Location. *ACM Trans. on Algorithms 6*, 1 (2009), 1–17.

[23] ANTONIADES, D., MARKATOS, E., AND DOVROLIS, C. One-click Hosting Services: A File-Sharing Hideout. In *Proc. IMC* (2009).

[24] ARLANDIS, A., AND BARANES, E. Interactions between network operators, content producers and internet intermediaries: Empirical implications of network neutrality. *Intereconomics 46*, 2 (2011), 98–105.

[25] ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R. H., KONWINSKI, A., LEE, G., PATTERSON, D. A., RABKIN, A., STOICA, I., AND ZAHARIA, M. Above the Clouds: A Berkeley View of Cloud Computing. UC Berkeley Technical Report EECS-2009-28, 2009.

[26] AUKIA, P., KODIALAM, M., KOPPOL, P., LAKSHMAN, T., SARIN, H., AND SUTER, B. RATES: A server for MPLS traffic engineering. *IEEE Network Magazine* (2000).

[27] AWDUCHE, D., CHIU, A., ELWALID, A., WIDJAJA, I., AND XIAO, X. Overview and Principles of Internet Traffic Engineering. RFC3272.

[28] B. GATES. Content is King. Microsoft Essay, 1/3/96.

[29] BENT, L., AND VOELKER, G. Whole Page Performance. In *Proc. Int. Workshop on Web Content Caching and Distribution* (2002).

[30] BERNERS-LEE, T., FIELDING, R., AND FRYSTYK, H. Hypertext Transfer Protocol – HTTP/1.0. RFC1945, 1996.

[31] BINDAL, R., CAO, P., CHAN, W., MEDVED, J., SUWALA, G., BATES, T., AND ZHANG, A. Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. In *Proc. ICDCS* (2006).

[32] BLOND, S. L., LEGOUT, A., AND DABBOUS, W. Pushing BitTorrent Locality to the Limit. *Com. Networks 22* (2011).

[33] BRESLAU, L., CAO, P., FAN, L., PHILIPS, G., AND SHENKER, S. Web caching and Zipf-like distributions: Evidence and implications. In *Proc. INFOCOM* (1999), pp. 126–134.

[34] BUFORD, J., YU, H., AND LUA, E. K. *P2P Networking and Applications*. Morgan Kaufmann Series in Networking, 2008.

[35] CacheLogic: The True Picture of P2P Filesharing. http://www.cachelogic.com/.

[36] CASADO, M., FREEDMAN, M. J., PETTIT, J., LUO, J., MCKEOWN, N., AND SHENKER, S. Ethane: Taking Control of the Enterprise. In *Proc. SIGCOMM* (2007).

[37] CHA, M., KWAK, H., RODRIGUEZ, P., AHN, Y.-Y., AND MOON, S. Analyzing the Video Popularity Characteristics of Large-scale User Generated Content Systems. *IEEE/ACM Trans. Netw. 17*, 5 (2009), 1357–1370.

[38] CHANG, H., JAMIN, S., AND WILLINGER, W. To Peer ot not to Peer: Modeling the Evolution of the Internet's AS-level Topology. In *Proc. INFOCOM* (2006).

[39] CHOFFNES, D. R., AND BUSTAMANTE, F. E. Taming the Torrent: a Practical Approach to Reducing Cross-ISP Traffic in Peer-to-peer Systems. In *Proc. SIGCOMM* (2008).

[40] CHOW, B., GOLLE, P., JAKOBSSON, M., SHI, E., STADDON, J., MASUOKA, R., AND MOLINA, J. Controlling Data in the Cloud: Outsourcing Computation without Outsourcing Control. In *Proc. ACM workshop on Cloud computing security* (2009).

[41] CHURCH, K., GREENBERG, A., AND HAMILTON, J. On Delivering Embarrasingly Distributed Cloud Services. In *Proc. HotNets* (2008).

[42] CISCO. NetFlow services and applications. White paper: http://www.cisco.com/warp/public/732/netflow.

[43] CISCO GLOBAL VISUAL NETWORKING AND CLOUD INDEX. Forecast and Methodology, 2011-2016. http://www.cisco.com.

[44] CLAFFY, K. C., AND BROWNLEE, N. Understanding Internet traffic streams: Dragonflies and Tortoises. *IEEE Trans. Communications* (2002).

[45] COHEN, B. Incentives Build Robustness in BitTorrent. In *P2PEcon Workshop* (2003).

[46] CONTAVALLI, C., VAN DER GAAST, W., LEACH, S., AND LEWIS, E. Client subnet in DNS requests. draft-vandergaast-edns-client-subnet-01.

[47] CONTAVALLI, C., VAN DER GAAST, W., LEACH, S., AND RODDEN, D. Client IP Information in DNS Requests. IETF draft, work in progress, draft-vandergaast-edns-suspend-ip-01.txt, 2011.

[48] CROSBY, S., AND BROWN, D. The Virtualization Reality. *Queue* (2006).

[49] CUEVAS, R., LAOUTARIS, N., YANG, X., SIGANOS, G., AND RODRIGUEZ, P. Deep Diving into BitTorrent Locality. In *Proc. INFOCOM* (2011).

[50] DHAMDHERE, A., AND DOVROLIS, C. Ten years in the evolution of the Internet ecosystem. In *Proc. IMC* (2008).

[51] DHUNGEL, P., ROSS, K. W., STEINER, M., TIAN, Y., AND HEI, X. Xunlei: Peer-Assisted Download Acceleration on a Massive Scale. In *Proc. PAM* (2012).

[52] DILLEY, J., MAGGS, B., PARIKH, J., PROKOP, H., SITARAMAN, R., AND WEIHL, B. Globally distributed content delivery. *IEEE Internet Computing* (2002).

[53] DIPALANTINO, D., AND JOHARI, R. Traffic Engineering versus Content Distribution: A Game-theoretic Perspective. In *Proc. INFOCOM* (2009).

[54] DOBRIAN, F., AWAN, A., STOICA, I., SEKAR, V., GANJAM, A., JOSEPH, D., ZHAN, J., AND ZHANG, H. Understanding the Impact of Video Quality on User Engagement. In *Proc. SIGCOMM* (2011).

[55] DREGER, H., FELDMANN, A., MAI, M., PAXSON, V., AND SOMMER, R. Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection. In *Proc. USENIX Security Symp.* (2006).

[56] E. BRONFMAN, JR. Remarks as prepared for delivery. Real Conference, 2000.

[57] ELWALID, A., JIN, C., LOW, S., AND WIDJAJA, I. MATE : MPLS adaptive traffic engineering. In *Proc. INFOCOM* (2001).

[58] ERMAN, J., GERBER, A., HAJIAGHAYI, M., PEI, D., AND SPATSCHECK, O. Network-aware forward caching. In *Proc. WWW* (2009).

[59] FANG, W., AND PETERSON, L. Inter-AS Traffic Patterns and their Implications. In *Proc. IEEE Global Internet* (1999).

[60] FELDMANN, A., GLADISCH, A., KIND, M., LANGE, C., SMARAGDAKIS, G., AND WESTPHAL, F. J. Energy Trade-offs among Content Delivery Architectures. In *CTTE* (2010).

[61] FELDMANN, A., GREENBERG, A., LUND, C., REINGOLD, N., AND REXFORD, J. NetScope: Traffic Engineering for IP Networks. *IEEE Network Magazine* (2000).

[62] FELDMANN, A., GREENBERG, A., LUND, C., REINGOLD, N., REXFORD, J., AND TRUE, F. Deriving Traffic Demands for Operational IP Networks: Methodology and Experience. In *Proc. SIGCOMM* (2000).

[63] FELDMANN, A., GREENBERG, A., LUND, C., REINGOLD, N., REXFORD, J., AND TRUE, F. Deriving Traffic Demands for Operational IP Networks: Methodology and Experience. *IEEE/ACM Trans. Netw.* (2001).

[64] FELDMANN, A., KAMMENHUBER, N., MAENNEL, O., MAGGS, B., PRISCO, R. D., AND SUNDARAM, R. A Methodology for Estimating Interdomain Web Traffic Demand. In *Proc. IMC* (2004).

[65] FELDMANN, A., KAMMENHUBER, N., MAENNEL, O., MAGGS, B., PRISCO, R. D., AND SUNDARAM, R. A Methodology for Estimating Interdomain Web Traffic Demands. In *Proc. IMC* (2004).

[66] FISCHER, S., KAMMENHUBER, N., AND FELDMANN, A. REPLEX: Dynamic Traffic Engineering based on Wardrop Routing Policies. In *Proc. CoNEXT* (2006).

[67] FISCHER, S., RÄCKE, H., AND VÖCKING, B. Fast Convergence to Wardrop Equilibria by Adaptive Sampling Methods. In *Proc. STOC* (2006).

[68] FISCHER, S., AND VÖCKING, B. Adaptive Routing with Stale Information. In *Proc. PODC* (2005).

[69] FLORANCE, K. Netflix Content Delivery. CDN Summit, 2013.

[70] FORTZ, B., AND THORUP, M. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proc. INFOCOM* (2000).

[71] FORTZ, B., AND THORUP, M. Optimizing OSPF/IS-IS Weights in a Changing World. *IEEE J. Sel. Areas in Commun.* (2002).

[72] FRANCOIS, P., AND BONAVENTURE, O. Avoiding transient loops during the convergence of link-state routing protocols. *IEEE/ACM Trans. Netw.* (2007).

[73] FRANK, B., POESE, I., LIN, Y., SMARAGDAKIS, G., FELDMANN, A., MAGGS, B., RAKE, J., UHLIG, S., AND WEBER, R. Pushing CDN-ISP Collaboration to the Limit. *ACM CCR 43*, 3 (July 2013).

[74] FRANK, B., POESE, I., SMARAGDAKIS, G., UHLIG, S., AND FELDMANN, A. Content-aware Traffic Engineering. *CoRR arXiv 1202.1464* (2012).

[75] FRANK, B., POESE, I., SMARAGDAKIS, G., UHLIG, S., AND FELDMANN, A. Content-aware Traffic Engineering. In *Proc. SIGMETRICS* (2012).

[76] FRANK, B., POESE, I., SMARAGDAKIS, G., UHLIG, S., FELDMANN, A., AND MAGGS, B. Enabling content-aware traffic engineering. *ACM CCR 42*, 4 (2012), 21–28.

[77] FREEDMAN, M. J. Experiences with CoralCDN: A Five-Year Operational View. In *Proc. NSDI* (2010).

[78] GADDE, S., CHASE, J., AND RABINOVICH, M. Web caching and content distribution: a view from the interior. *Computer Communications* (2001).

[79] GERBER, A., AND DOVERSPIKE, R. Traffic Types and Growth in Backbone Networks. In *OFC/NFOEC* (2011).

[80] GLOBAL INTERNET GEOGRAPHY. TeleGeography research. http://www.telegeography.com/product-info/gb/download/executive-summary.pdf, 2009.

[81] Gnutella v0.6 RFC. http://www.the-gdf.org.

[82] GOLDENBERG, D., QIUY, L., XIE, H., YANG, Y., AND ZHANG, Y. Optimizing Cost and Performance for Multihoming. In *Proc. SIGCOMM* (2004).

[83] Google Datacenters. http://www.google.com/about/datacenters/.

[84] Google Global Cache. http://ggcadmin.google.com/ggc.

[85] GOOGLE PUBLIC DNS. https://developers.google.com/speed/public-dns/.

[86] GUDE, N., KOPONEN, T., PETTIT, J., PFAFF, B., CASADO, M., MCKEOWN, N., AND SHENKER, S. NOX: Towards an Operating System for Networks. *ACM CCR 38*, 3 (2008).

[87] GUNNAR, A., JOHANSSON, M., AND TELKAMP, T. Traffic Matrix Estimation on a Large IP Backbone: A Comparison on Real Data. In *Proc. IMC* (2004).

[88] HALABI, S. *Internet Routing Architectures*. Cisco Press, 2000.

[89] HUANG, C., LI, J., WANG, A., AND ROSS, K. W. Understanding Hybrid CDN-P2P: Why Limelight Needs its Own Red Swoosh. In *NOSSDAV* (2008).

[90] HUANG, C., WANG, A., LI, J., AND ROSS, K. Measuring and Evaluating Large-scale CDNs. In *Proc. IMC* (2008).

[91] HULL, S. *Content Delivery Networks: Web Switching for Security, Availability, and Speed*. McGraw-Hill, 2002.

[92] JACOBSON, V., SMETTERS, D., THORNTON, J., PLASS, M., BRIGGS, N., AND BRAYNARD, R. Networking Named Content. In *Proc. CoNEXT* (2009).

[93] JAIN, S., KUMAR, A., MANDAL, S., ONG, J., POUTIEVSKI, L., SINGH, A., VENKATA, S., WANDERER, J., ZHOU, J., ZHU, M., ZOLLA, J., HOLZLE, U., STUART, S., AND VAHDAT, A. B4: Experience with a Globally-Deployed Software Defined WAN. In *Proc. SIGCOMM* (2013).

[94] JIANG, W., ZHANG-SHEN, R., REXFORD, J., AND CHIANG, M. Cooperative Content Distribution and Traffic Engineering in an ISP Network. In *Proc. SIGMETRICS* (2009).

[95] JOHNSON, K., CARR, J., DAY, M., AND KAASHOEK, M. The Measured Performance of Content Distribution Networks. In *Proc. Int. Workshop on Web Caching and Content Delivery* (2000).

[96] JUNG, J., SIT, E., BALAKRISHNAN, H., AND MORRIS, R. DNS Performance and the Effectiveness of Caching. *IEEE/ACM Trans. Netw. 10*, 5 (2002), 589–603.

[97] KANDULA, S., KATABI, D., DAVIE, B., AND CHARNY, A. Walking the Tightrope: Responsive Yet Stable Traffic Engineering. In *Proc. SIGCOMM* (2005).

[98] KARAGIANNIS, T., RODRIGUEZ, P., AND PAPAGIANNAKI, K. Should ISPs fear Peer-Assisted Content Distribution? In *Proc. IMC* (2005).

[99] KERALAPURA, R., TAFT, N., CHUAH, C., AND IANNACCONE, G. Can ISPs Take the Heat from Overlay Networks? In *Proc. HotNets* (2004).

[100] KEY, P., MASSOULIÉ, L., AND TOWSLEY, D. Path Selection and Multipath Congestion Control. *Commun. of ACM* (2011).

[101] KODIALAM, M., AND LAKSHMAN, T. V. Minimum Interference Routing with Applications to MPLS Traffic Engineering. In *Proc. INFOCOM* (2000).

[102] KOHAVI, R., HENNE, R. M., AND SOMMERFIELD, D. Practical Guide to Controlled Experiments on the Web: Listen to Your Customers not to the HiPPO. In *KDD* (2007).

[103] KRISHNAMURTHY, B., AND REXFORD, J. *Web protocols and practice: HTTP/1.1, Networking protocols, caching, and traffic measurement*. Addison-Wesley, 2001.

[104] KRISHNAMURTHY, B., WILLS, C., AND ZHANG, Y. On the Use and Performance of Content Distribution Networks. In *Proc. ACM IMW* (2001).

[105] KRISHNAN, R., MADHYASTHA, H., SRINIVASAN, S., JAIN, S., KRISHNAMURTHY, A., ANDERSON, T., AND GAO, J. Moving Beyond End-to-end Path Information to Optimize CDN Performance. In *Proc. IMC* (2009).

[106] KRISHNAN, S. S., AND SITARAMAN, R. K. Video Stream Quality Impacts Viewer Behavior: Inferring Causality using Quasi-Experimental Designs. In *Proc. IMC* (2012).

[107] LABOVITZ, C., LEKEL-JOHNSON, S., MCPHERSON, D., OBERHEIDE, J., AND JAHANIAN, F. Internet Inter-Domain Traffic. In *Proc. SIGCOMM* (2010).

[108] LAKHINA, A., PAPAGIANNAKI, K., CROVELLA, M., DIOT, C., KOLACZYK, E., AND TAFT, N. Structural Analysis of Network Traffic Flows. In *Proc. SIGMETRICS* (2004).

[109] LAOUTARIS, N., SIRIVIANOS, M., YANG, X., AND RODRIGUEZ, P. Inter-Datacenter Bulk transfers with NetStitcher. In *Proc. SIGCOMM* (2011).

[110] LAOUTARIS, N., SMARAGDAKIS, G., STANOJEVIC, R., RODRIGUEZ, P., AND SUNDARAM, R. Delay-Tolerant Bulk Data Transfers on the Internet. *IEEE/ACM Trans. Netw.* (2013).

[111] LEE, J., YI, Y., CHONG, S., AND JIN, Y. On the Interaction between Content-oriented Traffic Scheduling and Revenue Sharing among Providers. In *Proc. of IEEE INFOCOM Workshop on Smart Data Pricing* (2013).

[112] LEIGHTON, T. Improving Performance on the Internet. *Commun. of ACM* (2009).

[113] LI, A., YANG, X., KANDULA, S., AND ZHANG, M. CloudCmp: Comparing Public Cloud Providers. In *Proc. IMC* (2010).

[114] LIMELIGHT NETWORKS. http://www.limelight.com/technology/.

[115] LIU, H. H., WANG, Y., YANG, Y., WANG, H., AND TIAN, C. Optimizing Cost and Performance for Content Multihoming. In *Proc. SIGCOMM* (2012).

[116] LIU, X., DOBRIAN, F., MILNER, H., JIANG, J., SEKAR, V., STOICA, I., AND ZHANG, H. A Case for a Coordinated Internet-Scale Video Control Plane. In *Proc. SIGCOMM* (2012).

[117] MA, R. T. B., CHIU, D. M., LUI, J. C. S., MISRA, V., AND RUBENSTEIN, D. On Cooperative Settlement Between Content, Transit, and Eyeball Internet Service Providers. *IEEE/ACM Trans. Netw.* (2011).

[118] MAIER, G., FELDMANN, A., PAXSON, V., AND ALLMAN, M. On Dominant Characteristics of Residential Broadband Internet Traffic. In *Proc. IMC* (2009).

[119] MAO, Z., CRANOR, C., DOUGLIS, F., RABINOVICH, M., SPATSCHECK, O., AND WANG, J. A Precise and Efficient Evaluation of the Proximity Between Web Clients and Their Local DNS Servers. In *Proc. USENIX ATC* (2002).

[120] MAROCCO, E., AND GURBANI, V. Application-Layer Traffic Optimization. http://http://datatracker.ietf.org/wg/alto/charter/, 2008.

[121] MAXMIND LLC. http://www.maxmind.com.

[122] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., PARULKAR, G., PETERSON, L., REXFORD, J., SHENKER, S., AND TURNER, J. OpenFlow: enabling innovation in campus networks. *ACM CCR 38*, 2 (2008).

[123] MCKNIGHT, L. W., AND (EDS), J. P. B. *Internet Economics*. MIT Press, 1998.

[124] MEDINA, A., TAFT, N., SALAMATIAN, K., BHATTACHARYYA, S., AND DIOT, C. Traffic Matrix Estimation: Existing Techniques and New Directions. In *Proc. SIGCOMM* (2002).

[125] MICROSOFT AZURE. http://www.windowsazure.com.

[126] MITZENMACHER, M. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics* (2003).

[127] MOCKAPETRIS, P. Domain Names - Implementation and Specification. RFC 1035, Nov 1987.

[128] MOY, J. OSPF Version 2. RFC 2328, IETF, 1998.

[129] NAKAO, A., PETERSON, L., AND BAVIER, A. A Routing Underlay for Overlay Networks. In *Proc. SIGCOMM* (2003).

[130] NAMEBENCH. http://code.google.com/p/namebench/.

[131] Announcing the Netflix Open Connect Network. http://blog.netflix.com/2012/06/announcing-netflix-open-connect-network.html.

[132] Network Functions Virtualisation. SDN and OpenFlow World Congress, Oct 2012.

[133] NIVEN-JENKINS, B., FAUCHEUR, F. L., AND BITAR, N. Content Distribution Network Interconnection (CDNI) Problem Statement. IETF draft, work in progress, draft-ietf-cdni-problem-statement-03.txt, 2012.

[134] NORTON, W. The Internet Peering Playbook: Connecting to the Core of the Internet. DrPeering Press, 2012.

[135] NYGREN, E., SITARAMAN, R. K., AND SUN, J. The Akamai Network: A Platform for High-performance Internet Applications. *SIGOPS Oper. Syst. Rev.* (2010).

[136] ODLYZKO, A. Content is not king. *First Monday 6*, 2 (2001).

[137] ODLYZKO, A. M. Internet traffic growth: Sources and implications. http://www.dtc.umn.edu/mints/home.php, 2003.

[138] ORAN, D. OSI IS-IS Intra-domain Routing Protocol. RFC 1142, IETF, 1990.

[139] OTTO, J. S., SANCHEZ, M. A., R.CHOFFNES, D., BUSTAMANTE, F., AND SIGANOS, G. On Blind Mice and the Elephant: Understanding the Network Impact of a Large Distributed System. In *Proc. SIGCOMM* (2011).

[140] OTTO, J. S., SÁNCHEZ, M. A., RULA, J. P., AND BUSTAMANTE, F. E. Content Delivery and the Natural Evolution of DNS - Remote DNS Trends, Performance Issues and Alternative Solutions. In *Proc. IMC* (2012).

[141] PAN, J., HOU, Y., AND LI, B. An Overview of DNS-based Server Selections in Content Distribution Networks. *Com. Networks* (2003).

[142] PAPAGIANNAKI, K., TAFT, N., AND LAKHINA, A. A Distributed Approach to Measure IP Traffic Matrices. In *Proc. IMC* (2004).

[143] PAXSON, V. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks 31*, 23-24 (1999), 2435–2463.

[144] PLISSONNEAU, L., COSTEUX, J.-L., AND BROWN, P. Analysis of Peer-to-Peer Traffic on ADSL. In *Proc. PAM* (2005).

[145] POESE, I., FRANK, B., AGER, B., SMARAGDAKIS, G., AND FELDMANN, A. Improving Content Delivery using Provider-Aided Distance Information. In *Proc. IMC* (2010).

[146] POESE, I., FRANK, B., AGER, B., SMARAGDAKIS, G., UHLIG, S., AND FELDMANN, A. Improving Content Delivery with PaDIS. *IEEE Internet Computing* (2012).

[147] POESE, I., UHLIG, S., KAAFAR, M. A., DONNET, B., AND GUEYE, B. IP Geolocation Databases: Unreliable? *ACM CCR 41* (April 2011).

[148] QURESHI, A., WEBER, R., BALAKRISHNAN, H., GUTTAG, J., AND MAGGS, B. Cutting the Electric Bill for Internet-scale Systems. In *Proc. SIGCOMM* (2009).

[149] RAGHAVAN, B., CASADO, M., KOPONEN, T., RATNASAMY, S., GHODSI, A., AND SHENKER, S. Software-Defined Internet Architecture: Decoupling Architecture from Infrastructure. In *Proc. HotNets* (2012).

[150] RASTI, A., STUTZBACH, D., AND REJAIE, R. On the Long-term Evolution of the Two-Tier Gnutella Overlay. In *Global Internet* (2006).

[151] RATNASAMY, S., HANDLEY, M., KARP, R., AND SHENKER, S. Topologically aware overlay construction and server selection. In *Proc. INFOCOM* (2002).

[152] REKHTER, Y., LI, T., AND HARES, S. A Border Gateway Protocol 4 (BGP-4). RFC 4271, IETF, 2006.

[153] University of Oregon Routeviews Project. http://www.routeviews.org/.

[154] SANDVINE INC. Global Broadband Phenomena. Research Report http://www.sandvine.com/news/global_broadband_trends.asp, 2011.

[155] SAROIU, S., GUMMADI, K., DUNN, R., GRIBBLE, S., AND LEVY, H. An analysis of Internet content delivery systems. In *proc. OSDI* (2002).

[156] SAVAGE, S., COLLINS, A., AND HOFFMAN, E. The End-to-End Effects of Internet Path Selection. In *Proc. SIGCOMM* (1999).

[157] SCHULZE, H., AND MOCHALSKI, K. Internet study 2007-2009. http://www.ipoque.com/resources/internet-studies/.

[158] SEETHARAMAN, S., AND AMMAR, M. On the Interaction between Dynamic Routing in the Native and Overlay Layers. In *Proc. INFOCOM* (2006).

[159] SEIBERT, J., TORRES, R., MAFFIANO, M., MELLA, M., NITA-ROTARU, C., AND RAO, S. The Internet-wide Impact of P2P Traffic Localization on ISP Profitability. *IEEE/ACM Trans. Netw. 20* (2012).

[160] SERPANOS, D. N., KARAKOSTAS, G., AND WOLF, W. H. Effective Caching of Web Objects using Zipf's Law. In *ICME* (2000).

[161] SHARMA, A., MISHRA, A., KUMAR, V., AND VENKATARAMANI, A. Beyond MLU: An application-centric comparison of traffic engineering schemes. In *Proc. INFOCOM* (2011).

[162] SHARMA, A., VENKATARAMANI, A., AND SITARAMAN, R. K. Distributing Content Simplifies ISP Traffic Engineering. In *Proc. SIGMETRICS* (2013).

[163] SHEN, G., WANG, Y., XIONG, Y., ZHAO, B. Y., AND ZHANG, Z.-L. HPTP: Relieving the Tension between ISPs and P2P. In *Proc. IPTPS* (2007).

[164] SHEN, X., YU, H., BUFORD, J., AND AKON, M. *Handbook of peer-to-peer networking*, vol. 1. Springer Heidelberg, 2010.

[165] SMARAGDAKIS, G., LAOUTARIS, N., OIKONOMOU, K., STAVRAKAKIS, I., AND BESTAVROS, A. Distributed Server Migration for Scalable Internet Service Deployment. *IEEE/ACM Trans. Networking* (2013).

[166] SOULE, A., LAKHINA, A., TAFT, N., PAPAGIANNAKI, K., SALAMATIAN, K., NUCCI, A., CROV-
ELLA, M., AND DIOT, C. Traffic Matrices: Balancing Measurements, Inference and Modeling. In
Proc. SIGMETRICS (2005).

[167] STEINMETZ, R., AND WEHRLE, K. *P2P Systems and Applications*. Springer Lecture Notes in CS,
2005.

[168] Streamingmedia Blog. http://blog.streamingmedia.com/the_business_of_online_vi/2011/06.

[169] SU, A., CHOFFNES, D., KUZMANOVIC, A., AND BUSTAMANTE, F. Drafting behind Akamai
(travelocity-based detouring). In *Proc. SIGCOMM* (2006).

[170] TARIQ, M., ZEITOUN, A., VALANCIUS, V., FEAMSTER, N., AND AMMAR, M. Answering What-if
Deployment and Configuration Questions with Wise. In *Proc. SIGCOMM* (2008).

[171] TASHEV, R. Experimenting with Neighbour Discovery Schemes for P2P Networks in a Simulation
Framework. In *Master thesis, Dept of CS, TU Munich* (2006).

[172] TRIUKOSE, S., AL-QUDAH, Z., AND RABINOVICH, M. Content Delivery Networks: Protection or
Threat? In *ESORICS* (2009).

[173] UHLIG, S., QUOITIN, B., LEPROPRE, J., AND BALON, S. Providing Public Intradomain Traffic
Matrices to the Research Community. *ACM CCR* (2006).

[174] VALANCIUS, V., LUMEZANU, C., FEAMSTER, N., JOHARI, R., AND VAZIRANI, V. V. How Many
Tiers? Pricing in the Internet Transit Market. In *Proc. SIGCOMM* (2011).

[175] VIRTUAL COMPUTING ENVIRONMENT CONSORTIUM. http://www.vce.com.

[176] VIXIE, P. DNS Complexity. *ACM Queue 5*, 3 (2007), 24–29.

[177] VIXIE, P. What DNS is Not. *Commun. of ACM* (2009).

[178] WALLERICH, J., DREGER, H., FELDMANN, A., KRISHNAMURTHY, B., AND WILLINGER, W. A
methodology for studying persistency aspects of Internet flows. *ACM CCR* (2005).

[179] WILLINGER, W., AND ROUGHAN, M. *Internet Topology Research Redux*. ACM SIGCOMM eBook:
Recent Advances in Networking, 2013.

[180] XIA, P., CHAN, S.-H. G., CHIANG, M., SHUI, G., ZHANG, H., WEN, L., AND YAN, Z. Distributed
Joint Optimization of Traffic Engineering and Server Selection. In *IEEE Packet Video Workshop*
(2010).

[181] XIAO, X., HANNAN, A., BAILEY, B., AND NI, L. Traffic Engineering with MPLS in the Internet.
IEEE Network Magazine (2000).

[182] XIE, H., YANG, Y. R., KRISHNAMURTHY, A., LIU, Y. G., AND SILBERSCHATZ, A. P4P: Provider
Portal for Applications. In *Proc. SIGCOMM* (2008).

[183] YANG, X., AND DE VECIANA, G. Service Capacity of Peer to Peer Networks. In *Proc. INFOCOM*
(2004).

[184] ZHANG, Y., BRESLAU, L., PAXSON, V., AND SHENKER, S. On the Characteristics and Origins of Internet Flow Rates. In *Proc. SIGCOMM* (2002).

[185] ZHANG, Y., ROUGHAN, M., DUFFIELD, N., AND GREENBERG, A. Fast Accurate Computation of Large-scale IP Traffic Matrices from Link Loads. In *Proc. SIGMETRICS* (2003).

[186] ZHANG, Y., ROUGHAN, M., LUND, C., AND DONOHO, D. An Information-theoretic Approach to Traffic Matrix Estimation. In *Proc. SIGCOMM* (2003).

[187] ZIPF, G. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.

The Design Space of Network Mobility

Pamela Zave   Jennifer Rexford

Abstract

While the Internet is increasingly mobile, seamless mobility is difficult to implement at Internet scale. Over the years, standards bodies and the research community have introduced a large and confusing collection of mobility proposals that are difficult to compare. In this tutorial, we present these mobility proposals in a uniform framework, called the geomorphic view of networking. The geomorphic view shows that there are two distinct patterns for implementing mobility, each with its own range of design choices and cost-benefit trade-offs. We use these patterns to classify and explain a representative sample of mobility mechanisms, abstractly yet precisely. The patterns also serve as a basis for evaluating properties of these mechanisms such as resource costs and scalability, and for considering composition of mobility mechanisms.

1   Introduction

The Internet is increasingly mobile. Users access Internet services from mobile devices that move from one wireless access point to another, or switch between WiFi and cellular network connectivity. Ubiquitous computing relies on sensors and actuators attached to vehicles, portable objects, and animals as well as people. Applications provide customers with application-level identities that can be used to reach them at whichever device they are currently using. Increasingly, software runs on virtual machines that can migrate from one physical server, or even one data center, to another.

We define network mobility as the capability that allows a communicating entity to continue to communicate over a network, despite the fact that its location at (or binding to) a lower-level communicating entity is changing. Further, we focus on so-called "seamless" mobility, in which the high-level entity's communication channels are preserved throughout the change.

It is important to note that this definition applies at all conceptual levels. A person can have an identifier as a communicating entity, and can be mobile by moving from one networked device to another. A device such as a cellphone can have an identifier, and can be mobile by moving from one network attachment point to another. An interface on a device, such as an Ethernet interface, can have an identifier and be mobile by moving within or between local area networks.

In recent years, many mechanisms have emerged for supporting mobility. A recent survey [39] cites 22 Internet protocols dating from 1991 to 2009, including most prominently Mobile IPv4, Mobile IPv6, MSM-IP, HIP, MOBIKE, Cellular IP, HAWAII, ILNP, and LISP Mobile Node. In other contexts mobility is supported by:

- Ethernet LANs and VLANs, which allow an interface to retain its IP address as the host moves within the LAN;

- the scalable "flat" routing architectures SEATTLE [18], PortLand [21], VL2 [10], NVP [23], and Rbridges/TRILL [29, 36] that also naturally support routing to an interface that retains its addresses

as the host move;

- injecting the IP address of a mobile machine into existing routing protocols such as OSPF and BGP [1, 16];

- the General Packet Radio Service (GPRS) Tunneling Protocol (GTP), which supports mobility in most cellular networks;

- other research proposals including TCP Migrate [32], Serval [25], and the Internet Indirection Infrastructure [35];

- application-level protocols such as the Session Initiation Protocol (SIP) [31].

Each well-known proposal tends to spawn a family of variants, so the total number is probably in the hundreds and growing.

These various mechanisms operate at different levels, and make different assumptions about naming, routing, session protocols, scale, security, and the cooperation of remote endpoints or multiple administrative domains. Because the community lacks a common framework for describing and comparing mobility mechanisms, their relationships are poorly understood. Comparisons tend to be based on superficial characteristics rather than inherent ones. Quantitative comparison must be based on labor-intensive prototyping and measurement or simulation.

This book chapter has two goals:

- To describe and compare existing proposals for implementing mobility.

- To map out a design space in which new mobility mechanisms can be discovered, evaluated, and exploited.

To achieve these goals, we explain mobility in a new way. We begin by defining a common framework for describing network architectures. This framework is called the *geomorphic view* of networking, and is introduced in Section 2.

The geomorphic view has been developed to be simple, modular, comprehensive, and formalizable. It supports the first goal by providing a precise, unique description of each mobility mechanism that omits inessential detail while exposing subtle differences and important engineering trade-offs.

The common framework supports the second goal in several ways. It allows us to generalize over the implementations of mobility, showing that they are all instances of two major patterns. It also allows us to understand how different instances of the mobility patterns at different places in a network architecture can be composed, generating a potentially large design space to be explored.

Section 3 of this chapter introduces the two major patterns—*dynamic-routing mobility* and *session-location mobility*—for implementing mobility. If they are both implemented within an IP layer, the difference centers on whether a mobile machine retains its IP address when it moves, or changes its IP address and updates its correspondents. Each pattern has a completely different set of subsidiary design decisions and resource costs.

The next sections use the patterns to describe and compare many of the most important proposals for mobility. Sections 4 and 5 compare different ways to implement dynamic-routing mobility, with a non-hierarchical name space (*e.g.,* MAC addresses in a local area network) or a hierarchical name space (*e.g.,* IP addresses in the wide area), respectively. Section 6 compares four prominent protocols for session-location mobility at different stages in the IETF standardization process.

Section 7 of the chapter turns to a more systematic exploration of the design space, first showing that there may be some freedom concerning where in an architecture a particular kind of mobility is handled. The section also discusses composition of mobility mechanisms. As an example, we illustrate how Mobile IPv6 is a composition of both the dynamic-routing mobility and session-location mobility design patterns.

Finally, Section 8 surveys several topics closely related to mobility, including multihoming, anycast services, site mobility, incremental deployment of mobility protocols, and security issues for mobility. The chapter ends with a brief conclusion outlining several more advanced areas of study.

2   The geomorphic view of networking

The geomorphic view of networking was originally inspired by the work of Day [7], although we have made many changes and additions in both content and presentation. In this common framework for describing networks, the module is a *layer*, and a network architecture is a hierarchy of layers.

2.1   Comparison with the Internet and OSI models

Layers may seem familiar and obvious because both the classic Internet architecture [6] and the OSI reference model [14] also describe network architecture as a hierarchy of layers. However, our concept of a layer is very different. As a preview of this section, our layer hierarchies differ from these earlier ones in at least four ways:

- The classic Internet architecture and the OSI reference model both have a fixed number of levels. In a geomorphic layer hierarchy, there can be any number of levels.

- In the earlier models, there is only one layer on each level, so there is no distinction between layer and level. In a geomorphic hierarchy, there can be multiple layers on the same level.

- In the earlier models, each layer has a specific function that is distinct from the functions of other layers. In the geomorphic view each layer is a microcosm of networking, containing all of the basic components and functions in some form. In different layer instances there are different versions of these basic ingredients, used at different levels, with different scopes, and for different purposes.

- Most people interpret the earlier models as describing the data plane of networking only. The control plane is seen as separate and not modularized in the same way. In the geomorphic view, each layer— being a microcosm of networking—has a data plane and a control plane. Layers decompose both planes into modules.

Figure 1 illustrates these differences, and also shows how the "geomorphic" view got its name. The complex arrangement of layers, with overlapping, abutting, and bulging shapes, can resemble the complex arrangement of layers in the earth's crust.

2.2   Components of a layer

A layer has *members*, each of which has a unique and persistent *name* within the layer. For example, Figure 2 is a snapshot of a layer with five members, each having a capital letter as a name. In general a member is a concurrent process, *i.e.,* a locus of state and control with the potential for autonomous action.

The members of a layer communicate with each other through *links*, shown by lines in Figure 2. A link is a communication channel.
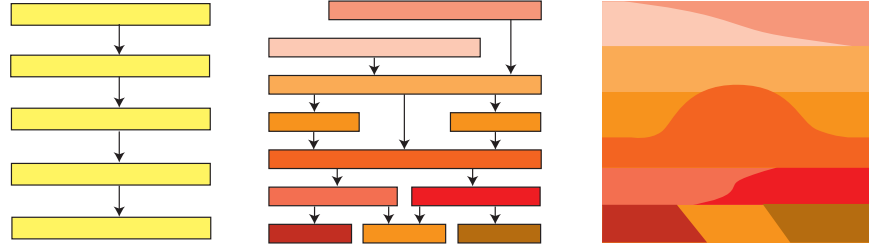
Figure 1: Arrangement of layers in the classic Internet architecture (left), the geomorphic view (middle), and the earth's crust (right).
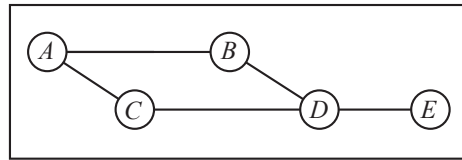

Figure 2: Members and links of a layer.

One of the two primary functions of a layer is to enable members to send messages to each other. This function is accomplished by a *forwarding protocol*, which runs in all members and has operations for sending and receiving messages over the links.

In general, a layer does not have a link between each pair of members. Such a layer needs *routes* indicating how one member can reach another through links and intermediate members. For example, (*A, B, D, E*) is a route from *A* to *E*. If *B* receives a message that is destined for *E*, its forwarding protocol uses the route information to forward the message to *D* on its way to *E*.

The *routes* are shared state of the layer, and a simple geomorphic description need say no more about them. To provide more realistic detail, in a real layer the *routes* information is often distributed over forwarding tables found in the individual members.

The other primary function of a layer is to implement enriched end-to-end communication services on top of its bare message transmission. This function is carried out by a *session protocol*. The forwarding protocol can be unreliable, especially if links are dynamic and the current routes are obsolete. A session protocol can provide services including reliability, FIFO delivery, and quality-of-service guarantees. Figure 3 shows a session between endpoints *a* and *e* of the lower layer.

A *channel* is an instance of a communication service. Both links and sessions are channels. A layer can implement its own links internally, and a layer can implement its sessions for the benefit of its own members.

Most commonly, however, a link in one layer is implemented by a session in another layer, as shown in Figure 3, placing the other layer lower in the *"uses" hierarchy*. If an underlay (lower layer) is implementing a link for an overlay (higher layer), then the basic attributes of the channel must be stored in the states of both layers. In the overlay, the channel object is one of its *links*. In the underlay, the channel object is one of its *sessions*. There must be two names for the sets of channels of interest to a layer, because a typical layer both uses *links* and implements *sessions*.

For a link in an overlay to be implemented by a session in an underlay, both endpoint *machines* must have members in both layers, as shown in Figure 3. The boundary of a *machine* is the boundary of an operating
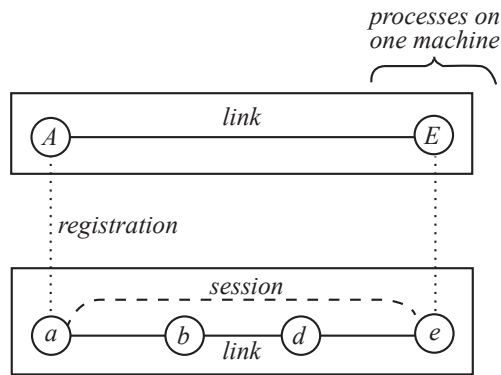
Figure 3: Implementation of a link in an overlay by a session in an underlay.

system that provides fast, reliable communication between members of different layers on the machine. This fast, reliable operating-system communication is the foundation on which networked communication is built.[1]

The relation between an overlay member and an underlay member on the same machine is called *registration*. Registrations must be stored in the state of both layers. In the overlay a registration is recorded as an *attachment*, which says that the overlay member is attached to the network through a particular lower layer. In the underlay a registration is recorded as a *location*, which says that a particular member of a particular overlay is attached to the network at a particular member (its location) of this layer.

The session protocol creates and maintains *sessions* data in its layer, and uses *locations* data. For example, in Figure 3, *A* sent a request to *a* for a session with *E*. To create this session, *a* learned from its layer's *locations* that *E* is currently located at *e*. Messages sent from *A* to *E* through the link in the overlay travel through *a, b, d,* and *e*; the first and last steps uses operating-system communication, while the middle three steps use networked communication.

All the major components of a layer are shown in Figure 4. The forwarding and session protocols perform the two primary functions of the layer. These protocols and their operations are collectively known as the "data plane" of the layer. The network's data plane also includes the inter-layer interfaces through which the endpoints of an implemented link transfer messages to and from the implementing session.

There are six major state components, all of which can be dynamic. We have seen that the session protocol creates and maintains *sessions*; the other five are created and maintained by their own maintenance algorithms. The state and algorithms are collectively known as the "control plane" of the layer. Note that the network's control plane also includes the inter-layer interfaces through which the control algorithms communicate.

2.3 Layers within a network architecture

Figure 5 shows a geomorphic view of the classic Internet architecture. The *scope* of a layer is its set of potential members. For example, at the top level of the hierarchy, there are two application layers. The

---

[1]Although layer members have been described as concurrent processes, they are not usually "processes" as defined by the operating system; processes in an operating system have many more properties and associations than layer members do. A virtual machine can be regarded as a *machine*, in which case communication through the hypervisor and soft switch of the physical machine is regarded as networked communication.

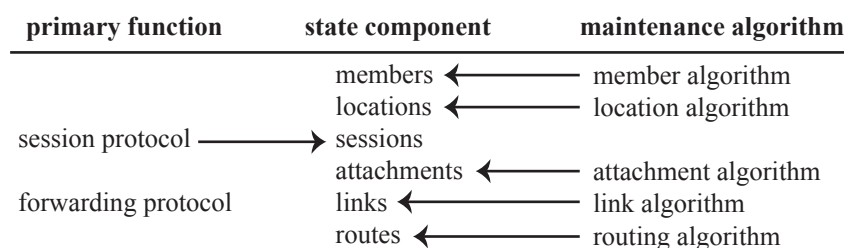| primary function	state component	maintenance algorithm
	members $\longleftarrow$	member algorithm
	locations $\longleftarrow$	location algorithm
session protocol $\longrightarrow$	sessions	
	attachments $\longleftarrow$	attachment algorithm
forwarding protocol	links $\longleftarrow$	link algorithm
	routes $\longleftarrow$	routing algorithm

Figure 4: Major components of a layer. Arrows show which protocol or algorithm writes a state component.

scope of each layer is the set of potential processes running software for that application. These layers are pictured as overlapping because the horizontal dimension is an approximation of geographical space, and both applications can have members world-wide. In particular, the registration lines in the diagram show that each application has a member on one particular machine.

In the middle level of the hierarchy there is a single layer called the "Internet core." Its members are the IP interfaces of networked machines. In this layer, IP (the "network layer" of the classic Internet architecture) is the forwarding protocol, and TCP and UDP (the "transport layer" of the classic Internet architecture) are variants of the session protocol.

At the bottom level of the hierarchy there are local area networks (LANs) with local scopes. Each LAN member is an interface appropriate to the type of LAN. For example, for an Ethernet LAN, the members are the Ethernet interfaces of machines. Figure 5 illustrates the point, made in the preview of this section, that in the geomorphic view there can be multiple layers at one level of the "uses" hierarchy.

Note that every member of the Internet core is attached to a member of a layer at the bottom level. Note especially that for two members of the Internet core layer to be linked, both of those members must be attached to the same layer at a lower level, so that the lower layer can implement the link. This observation is important for understanding mobility. A gateway in the Internet core layer is attached to multiple LANs, so it can forward messages from one LAN to another.

Because layers instantiated at different levels have different purposes, they have different versions of the common components enumerated in Figure 4. For one example, the best-known routing algorithms are found in the Internet core, where their purpose is reachability. Now consider a middleware layer, above the Internet core, offering cloud services and other facilities for enterprise computing. To provide security, this layer might have routing that ensures that all messages to a particular destination pass through a particular filtering server. Thus this layer has its own routing (control plane), separate from Internet routing. One of the major purposes of its routing is enterprise-specific security.

This example illustrates the points, made in the preview of this section, that in the geomorphic view the number of levels is not fixed (the middleware layer need not be present for the Internet to work), and that each layer can contain its own version of any basic function or component of networking (such as routing). In some layers, where a particular function or component is not needed, its presence is vestigial.

For another example of a basic function with different forms in different layers, low-level layers such as Ethernet LANs provide broadcast as a communication service. In geomorphic terms, channels (links and sessions) can be multi-point as well as point-to-point. The main services provided by the Internet core are point-to-point, while an application layer might implement its own multi-party communication service.[2]

---

[2]For simplicity, in the remainder of this chapter, all communication channels are assumed to be point-to-point. This is sufficient for a study of mobility.
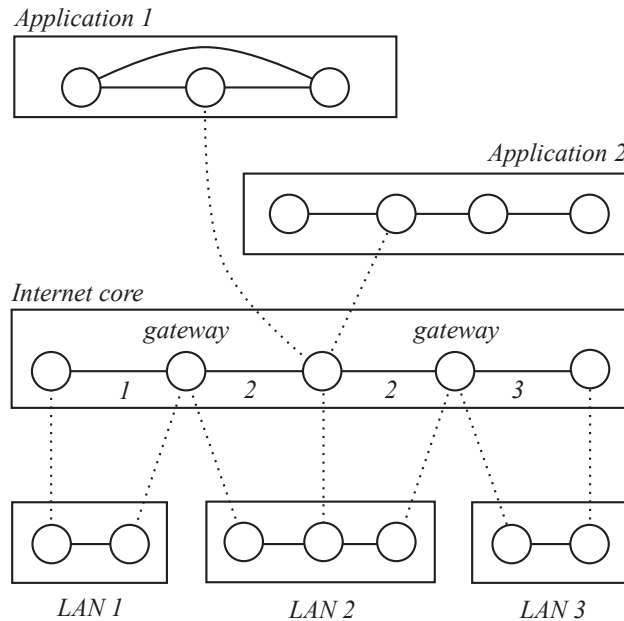
Figure 5: Geomorphic view of the classic Internet architecture. Internet links are labeled with the LAN that implements them.

Today's Internet is host to many customized architectures running simultaneously [30, 33]. Middleware is an important part of the ecosystem, while cloud services and virtual private networks add extra layers to the classic Internet architecture. It is self-evident that fixed layer structures cannot describe these architectures adequately. The geomorphic view is intended not only to describe them, but also to generate a design space including many others not yet explored.

2.4 Layers and mobility

If asked to define network mobility, most people would say something like, "A mobile device continues to have network connectivity as it moves geographically." For a simple Internet example, we can imagine a laptop that detaches from one edge subnetwork, where it has one IP address, and re-attaches to another edge subnetwork, where it has another IP address.

No layering is required to understand this scenario. At the same time, technologically the scenario is indistinguishable from a scenario in which one laptop is tossed into a deep lake and another laptop is purchased new.

Clearly mobility is more than this. As the mobile device detaches and re-attaches, we expect it to retain some identity and credentials, so that it can be reached in some of the same ways as before, and has some of the same rights and capabilities as before. The identity that is preserved is its membership in some layer, which must not change. What does change is the attachment of this identifying process to some process in some lower layer.

The left column of Figure 13 (which appears later, in Section 7.2) shows the two forms that this change of attachment can take. In the top picture, a process *m* changes its registration from one member of a lower

layer to another member of the same layer. The name *m* might be an application name, and *a1* and *a2* might be IP addresses in an Internet core layer. In the bottom picture, *m* changes its registration from a member of one lower layer to a member of another lower layer. Here *m* might be an IP address, and *a1* and *a2* might be members of two different LANs.

This shows that layering is an intrinsic part of the study of mobility, because it explains what stays the same and what changes. It will help us understand how a person can call a friend's cellphone, even though that friend has traveled hundreds of miles since the last call.

Even this is not sufficient to explain, however, how a person can talk to a friend's cellphone *while the friend is traveling hundreds of miles.* To explain this aspect of mobility it is necessary to focus on the communication channel that is being preserved across mobility events. As presented in Section 2.2, a communication channel is most often used in one layer, where it is called a link, and implemented in a lower layer, where it is called a session. Here is another place where layering is intrinsic to the study of mobility, explaining that the layer that benefits from mobility is usually not the layer that has the responsibility of implementing it.

To summarize, there are two relationships on layer pairs that are important in mobility. There is a dynamic registration relationship between an overlay with a mobile member and the underlays to which that member of the overlay is attached over time. There is an implementation relationship between an overlay with a link and the underlay whose session implements that link. Two overlay/underlay pairs—in any given instance of mobility, they must be the same overlay and underlay, right?

Wrong. Section 3 will show that there are two patterns for mobility. In one pattern the layer pairs coincide, and in the other they are different. People are often confused by mobility because it is often over-simplified. Mobility is easy to over-simplify when one is not explicit about the layers involved.

2.5 Mobility in the wild

It might be said that the problem with mobility is not too few proposals, but too many. As mentioned in the introduction, the total number is probably in the hundreds and growing.

In this chapter, the geomorphic view will provide a descriptive framework that imposes some order on this chaotic design space. This works because every mobility proposal has a unique description in terms of layers in the geomorphic view.

Unique description is achieved only because the geomorphic view is precisely defined and precisely used. For one example, in the geomorphic view there is one name space per layer. If any proposal has two different names for the same machine, one higher-level and one lower-level, then those names must be in the name spaces of two different layers. For another example, in the geomorphic view there is no tunneling. Tunneling is evidence that there are two distinct layers: a higher layer in which the "tunnel" is a link, and a lower layer that implements the link.

As a result, a geomorphic description of a network architecture might have more layers than a different description, and some components of some layers might be vestigial. This is a cost, but in return we get many benefits, even beyond the benefits of having a unique and comparable description of each proposal:

- Each layer is simpler, with a minimum of *ad hoc* complications.

- Proposals that might seem very diverse fall into a few recognizable patterns that apply at any level of the network stack.

- We can identify opportunities for re-use of formal models, formal analysis, and implementation code.

- Mobility is not the only networking challenge. If other complex mechanisms are also described in terms of the geomorphic view, we can make sure that they interact correctly.

Furthermore, redundancies in a description or model can be removed by optimization in an implementation phase. The trick is to understand and analyze the model first, then use the analysis to determine which optimizations are safe.

This approach leads to differences from other literature on mobility. As exemplified by [2], it is common for mobility proposals to be classified according to the layer of the classic Internet architecture where they are implemented. In contrast, we emphasize that each specific proposal is an instance of a general pattern, and that the general pattern can be used at any level of a network architecture. Comparisons between ideas are less subjective, because they are based on a common framework that exposes real similarities and differences, even when obscured by incidentals of language and application.

3 Two patterns for implementing mobility

In this section we show that there are two completely different patterns for implementing mobility. They differ in where the change of attachment appears with respect to the implementing layer, in which algorithms and protocols of the implementing layer are involved in implementing mobility, and in which parts of the shared state of the implementing layer are altered. They also differ in their detailed design decisions, and in their cost, performance, and scalability issues.

Not only are these patterns non-overlapping, they also completely cover all implementations of mobility, in the sense that each implementation either follows one pattern or is clearly a composition of the two patterns.

3.1 Dynamic-routing mobility

Figure 6 has two stages depicting the effect of mobility on an inter-layer channel. Recall that the channel is a *link* in the state of the layer that uses it, and a *session* in the state of the layer that implements it; its *higher endpoints* are members in the user layer, while its *lower endpoints* are members in the implementing layer.

The precise site of mobility here is the lower endpoint *A*. In Stage 1 *A* is registered at *a1* in Underlay 1. *a1* and *A* are connected to the rest of their layers through Links 1 and 2, respectively. Link 2 is implemented by Underlay 1.

Between Stage 1 and Stage 2 Link 1 stops working, possibly because the machine on which *A* and *a1* reside has been unplugged from a wired subnetwork, or has moved out of range of a wireless subnetwork. In a cascading sequence of events, Link 1 is destroyed, Link 2 is destroyed, and the registration of *A* at *a1* is destroyed. *A* is now disconnected from the rest of its layer.

Eventually the mobile machine may become plugged into another wired subnetwork or enter the range of another wireless subnetwork, as shown in Stage 2. In a cascading sequence of events, member *a2* (which is the mobile machine's member in the new Underlay 2) connects to the rest of its layer through Link 3, *A* becomes attached to new location *a2*, and new Link 4 is created in the mobility layer and implemented by Underlay 2. Note that *A* is now linked to *C* rather than *B*; this change is necessary because *C* is attached to Underlay 2 and *B* is not.

Between Stages 1 and 2 there may be an interval during which *A* has no connection with the rest of its layer. There may also be an interval in which Stages 1 and 2 overlap, so that *A* is temporarily attached to both underlays.
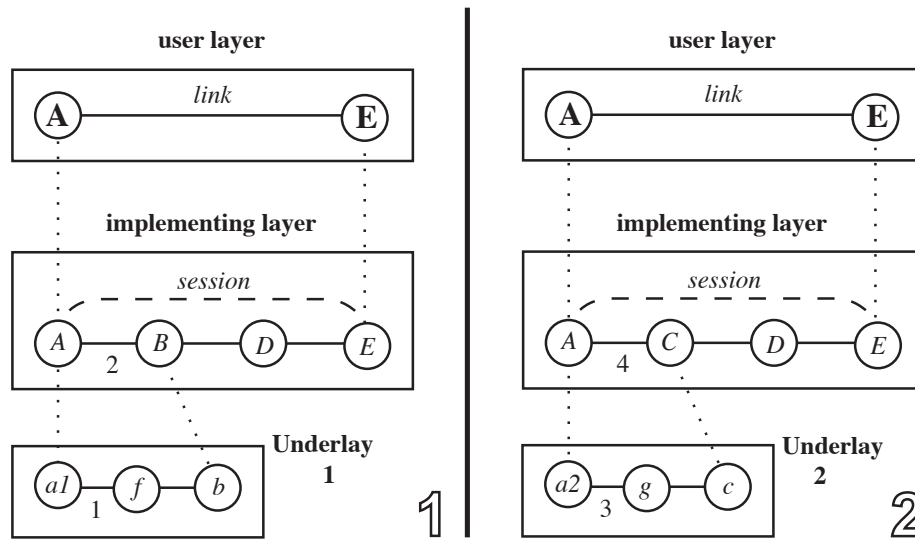
Figure 6: Two stages in an instance of dynamic-routing mobility.

The hard problem to be solved in Figure 6 is that even after *A* is again reachable by other members of its layer such as *D* and *E*, they do not know how to find it because the routes to it are obsolete. *Dynamic-routing mobility* relies on the routing algorithm of the layer, which must learn about new links, recompute routes, and update forwarding tables. After this is accomplished, *D* will know that it can reach *A* by forwarding to *C*.

There are three ways in which actual dynamic-routing mobility can differ from the example in Figure 6. Fortunately, none of them affect what the implementation has to do, so none of them need be discussed separately. First, the new attachment *a2* could be in the same layer as *a1*, rather than in a different layer. Because *a1* and *a2* are different locations, after the move *A* is probably linked to a different member of its own layer, even though the new link is implemented by the same lower layer as before.

Second, in Figure 6 the mobile member *A* has only one attachment and one necessary link. As shown in Figure 5, members such as gateways have multiple simultaneous attachments to different underlays. Because each such attachment is necessary for the gateway's purpose and supports its own link or links, the mobility of each attachment is a separate problem to be solved.

Third, occasionally a layer implements sessions for the benefit of its own members, rather than as a service to a higher user layer. In this case there is no **A** or **E**, and the beneficiaries of the mobility implementation are *A* and *E*.

A *router* is a member of a layer that receives and forwards messages not destined for itself, whether it sends and receives messages on its own behalf or not. A *forwarding table* is a distributed copy of some of the *routes* state component of a layer. Implementations of dynamic-routing mobility incur four kinds of resource cost:

- *storage cost* is the cost of storing routes to mobile members, in the forwarding tables of all the routers that need them;

- *update cost* is the cost of updating the stored routes as mobile members move;
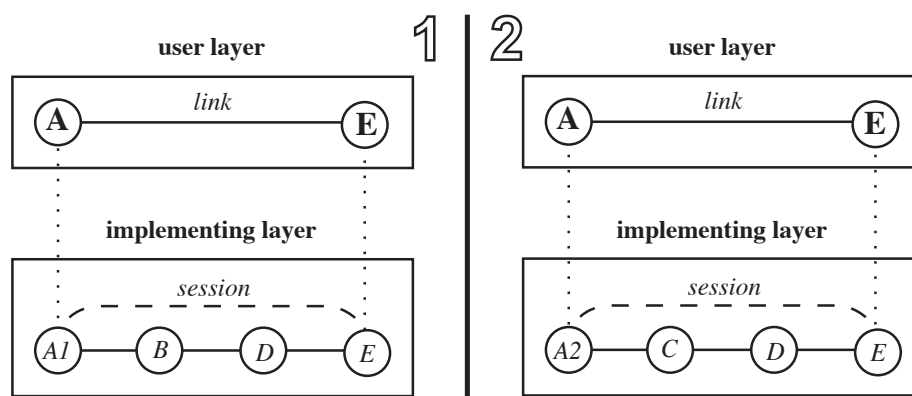
389

Figure 7: Two stages in an instance of session-location mobility.

- *path cost* is the cost of longer or more congested message paths due to mobility;

- *handoff latency* is the message delay caused by a move.

These costs will be discussed further in Section 3.3.

The primary issue in implementing dynamic-routing mobility (DRM) is that large layers such as the classic Internet core achieve scalability through a hierarchical name space. In the Internet core, names (IP addresses) are organized into a hierarchy based on geographical, topological, and administrative factors. A layer member is assigned a name based on its location in this hierarchy. Subtrees in the hierarchy correspond to blocks of names, and routing scales because it operates on aggregated blocks rather than individual names. Mobility violates the rules of this scheme, because a mobile member retains its name as it moves across the boundaries of the hierarchy. If implemented naively, it would require a large number of entries in the forwarding table of each IP router for individual mobile machines.

This issue is so important that the design decisions made to implement DRM are completely different in hierarchical and non-hierarchical layers. For that reason, we have divided examples of DRM into two sections (Sections 4 and 5).

3.2   Session-location mobility

Figure 7 has the same two stages as Figure 6. The most important difference is that **A**'s location in the implementing layer changes from *A1* to *A2*, rather than staying the same as it did in Figure 6. In geomorphic terms, the mobile machine's representative in the implementing layer (with name *A1*) has died, and has been reborn as a member of the implementing layer with name *A2*.

This is a natural occurrence in a layer with a hierarchical name space. It should be familiar from observing what happens when a laptop with an IP address *A1* moves to a new subnetwork of the Internet, and gets a new IP address *A2* from DHCP. The laptop cannot continue to use *A1* in the new subnetwork, because *A1* is not in the subnetwork's address block.

DHCP alone is not sufficient to implement mobility, however. As explained in Section 2.4 and shown in Figure 7, the strongest form of mobility requires preserving the communication channel in the user layer. The bulk of the work of implementing session-location mobility lies in ensuring that **A**'s correspondents

know that it is now located at *A2* rather than *A1*. Each lower endpoint that was participating in a session with *A1* on behalf of **A** must be informed that it should now be corresponding with *A2* instead.

As explained in Section 2.2, when an underlay is implementing a channel for an overlay, the initiating lower endpoint must be able to look up the location of the accepting higher endpoint in the underlay, so that it can send messages to it. This means that there must be a globally accessible copy of the *locations* mapping in the layer. Session-location mobility also requires updating this mapping when a higher endpoint moves.

Generally the fastest handoffs are achieved when a new lower endpoint sends updates directly to all its correspondent lower endpoints (in addition to updating the *locations* mapping). This requires, of course, that the new lower endpoint have the correct name of the lower endpoint at the other end of each session.

Interesting behavior arises if both ends of a session move concurrently. Neither lower endpoint will know the new name of the far endpoint, so neither can send an update to the other. In this simultaneous handoff scenario a mobile endpoint, finding that it cannot reach a far endpoint to update it, will suspect that the far endpoint has moved also. Both endpoints must fall back on lookup from the *locations* mapping to get the new location of the far endpoint.

As with Figure 6, the two stages in Figure 7 might have a gap between them or might overlap. If they overlap, there will be an interval during which *A* has two attachments in the same layer.

In Figure 7 the underlays are not shown, although they probably look similar to those in Figure 6. Most likely there is an underlay member *a1* that is destroyed, and an underlay member *a2* that is created. There is no mobility observable at this level, however, because *A1* is attached to *a1* in Underlay 1 throughout its lifetime, and *A2* is attached to *a2* in Underlay 2 throughout its lifetime. The only mobility that is observable is **A**'s change of attachment from *A1* to *A2*.

Strictly speaking some dynamic routing could be involved in session-location mobility, because *A2* is a new member of the layer and there must be routes to it. In practice this is rarely an issue, because the name *A2* is part of some larger block to which routes already exist.

Like DRM, session-location mobility (SLM) has storage costs, update costs, and handoff latency. The storage costs are the costs of maintaining a scalable implementation of *locations*. The update costs are the costs of updating *locations* and current correspondents when a member moves.

Implementations of SLM vary in a number of ways (see Section 6), although no one variation is as important as the hierarchical *versus* non-hierarchical variation for DRM.

3.3   Major differences between the patterns

There are obvious structural differences between the two patterns:

- In DRM the change of attachment appears between the implementing layer and the level below it, while in SLM the change of attachment appears between the user layer and the implementing layer (see Figures 6 and 7).

- In DRM the bulk of the work is performed by the routing algorithm, while in SLM the bulk of the work is performed by the session protocol and location algorithm (see Figure 4).

- In DRM the major state components that change are *attachments, links,* and *routes* (see Figure 4). In SLM the major state components that change are *locations* and *sessions*.

These structural differences prove that the two patterns are fundamentally different.

In attempting to understand mobility mechanisms, people are sometimes confused by the fact that *routes* (changed by DRM) and *locations* (changed by SLM) are both mappings. The *locations* mapping is usually implemented by a shared global data structure called a *directory*. The *routes* mapping is usually distributed

across the forwarding tables of the routers, but is occasionally implemented as a directory. The result is that directories are sometimes used in both DRM and SLM implementations.

This similarity is superficial because it does not tell us the most important thing about these mappings, which is what they mean in terms of network architecture. The mappings used in DRM and SLM are always fundamentally different, and can always be distinguished from one another. As mentioned in Section 3.1, *routes* is a peer-to-peer or intra-layer mapping: at each router, entries in the forwarding table map each destination name to a member, link, or path *in the same layer*. *Locations*, on the other hand, is always an inter-layer mapping, mapping names in a higher layer to names in a lower layer.

In describing mobility mechanisms, people often focus on the "identifier-locator split." This may be useful intuition, but should be interpreted carefully. In an episode of mobility there is always a layer member that retains its identity (the "identifier"), and two members at a lower level, where the attachment of the identifier moves from one to the other (the "locators"). The identifier-locator split does not distinguish DRM from SLM, although in the two patterns the identifiers and locators appear at different levels. In addition, it is important to remember that these terms are relative, as mobility can occur anywhere in a layer hierarchy.

On the surface, it may seem that DRM should be called "in-network mobility" or the like, while SLM should be called "end-to-end mobility" or the like. This reflects a misunderstanding of how general the patterns are, and how freely they can be applied at different levels. For one example, consider an application layer whose members run only on Internet hosts. The members include user clients and named services. The layer could have its own dynamic, application-specific routing to services, which allows services to be reached even though they move from server to server. This instance of DRM is not "in network" from most peoples' perspective. For another example, an Internet router might itself be mobile, and might have some of its links to other Internet routers preserved as it moves by session-location mobility at a lower level. This instance of SLM does not involve any endpoints according to most peoples' perspective.

Obviously a quantitative comparison between two mobility implementations cannot be made without implementation details and a profile of the expected load. Nevertheless, it is possible to make some general comparisons between the two patterns based on their potential strengths and weaknesses. We say "potential" because any characteristic, whether positive or negative, can be irrelevant in some situations.

The greatest potential weakness of DRM is its storage, update, and path costs. Normally routing information is different in different places, so there is a lot of it, it is spread widely across a layer, and it is expensive to update. Attempts to economize on storage and update costs can lead to high path costs (see Section 5), as messages travel further to be routed successfully. Path costs must be weighted heavily because *every* message that travels on a channel is affected by its path cost, if any.

Locations are very different from routes because the result of a location query is usually the same no matter which member is querying (in contrast to a route, which is different depending on where it is starting from), and because a location query is needed only at the beginning of a session and possibly after a move (in contrast to routes, which are consulted on every hop of every message). As a result, locations can be stored and updated much more cheaply than routes. For example, even a centralized directory would perform adequately in many contexts. And even if lookup of a location is slow, we do not count it as a path cost because the cost is incurred a few times for each channel rather than being built into the cost of transmitting each message on the channel.

The greatest potential weakness of SLM is that it must be implemented with the participation of session endpoints. This means that deployment of an SLM mechanism requires new or upgraded mobile devices that run the SLM protocol for sending and receiving location updates. Full interoperation with legacy endpoints calls for expensive middleboxes. Security is a concern because endpoint devices can initiate updates of the global layer state.
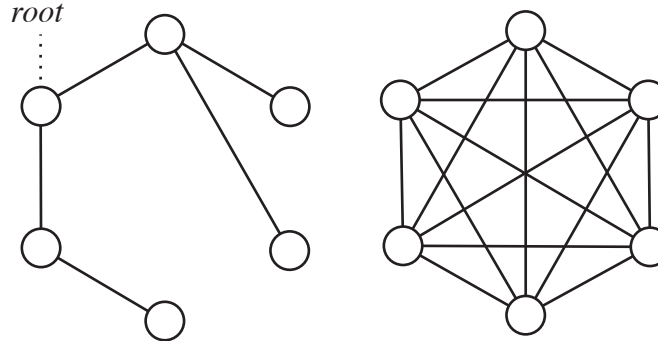
Figure 8: Inter-switch links of an Ethernet LAN layer (left) and an overlay layer (right). The Ethernet links are physical, while the overlay links are virtual.

Concerns such as software upgrading and security have attracted less attention with respect to DRM. This is because a layer can, in principle, be designed so that its members are partitioned into endpoints and routers, and only the routers need be aware of or participate in an implementation of DRM. In reality these concerns are ubiquitous in distributed computing, and can apply to routers as well.

4 Examples of dynamic-routing mobility in non-hierarchical layers

Dynamic-routing mobility is often used in LANs, which have smaller scopes and can function without a hierarchical name space. These LANs handle mobility naturally as part of the normal routing function, since end-points retain their addresses as they move and routing does not rely on location-dependent addressing.

4.1 Wired Ethernet LANs

An Ethernet LAN is a single layer. Its member processes are the Ethernet representatives of hosts (end-points) and switches (routers), and its names are MAC addresses. It has no pre-attachment requirements or configuration for hosts, which makes it "plug and play."

The LAN offers both broadcast and point-to-point services to higher layers. In this brief section we do not consider these communication services further, so there will be no discussion of the layer's sessions or locations. Also, for simplicity, we will not extend the modeling into lower levels, so links in the Ethernet layer are primitives.

An Ethernet layer has two kinds of links. There are point-to-point links between switches, each of which is basically a wire between two machines. There are also shared media or buses. A bus delivers each message to every machine on the bus, and is used to connect a switch to a set of hosts. Either kind of link can be identified at each switch that uses it by the port on the switch's machine to which it is attached.

The inter-switch links of the layer must form a bidirectional spanning tree (see Figure 8). Otherwise, when flooding is used (see below), the network could be overwhelmed by messages traveling on cycles. There are usually more physical links than needed for the spanning tree, but the extras can only be used when other links fail and the spanning tree is recomputed.

Each switch has a forwarding table containing *(MAC address, port)* pairs. The port identifies the link on which the switch should forward messages destined for the MAC address. Each switch's table is sparse and

393

is populated lazily by a routing algorithm called "MAC learning." Upon receiving a message with a source MAC address that is not in its forwarding table, the switch adds to its table the MAC address and the link on which the message was received.

The forwarding algorithm of a switch is similarly simple. Upon receiving a message not destined for itself, the switch looks for the destination MAC address in its own forwarding table. If it finds an entry, it forwards the message on the designated link. If it does not find an entry, it "floods" by forwarding the message on every link except the one on which it was received.

These mechanisms implement dynamic-routing mobility as an aspect of normal operation rather than as a special case. When a host moves within the layer, it changes the link through which it is attached to the layer. As soon as it sends messages, new routes to it begin to propagate through the layer. Obsolete forwarding-table entries are removed when their time-to-live expires. Missing table entries are handled by flooding. Note that an entry might also be removed from a forwarding table because the table is full and space for a newer entry is needed.

4.2 Ethernet overlays

Several recent designs [18, 10, 23] avoid flooding by forming an overlay topology that interconnects all of the edge switches, as shown on the right side of Figure 8. While the inter-switch links of an Ethernet are physical and form a spanning tree, the inter-switch links of an overlay network are virtual and fully connect the switches.

The virtual links are communication services implemented by a second, lower layer. For example, Figure 9 shows the path of a message from host *Hv* to host *Hz* (the lower-case letters stand for their MAC addresses). On each hop, the path is labeled with the source name above and the destination name below. The virtual hop between switches *Sw* and *Sy* in the overlay layer is implemented in the underlay, where the message is encapsulated in a message with source *w* and destination *y*.

How are the virtual links in the overlay implemented by the underlay? The members of the underlay layer are the switches only, not the hosts. Each switch's name is the MAC address of its machine, just as in the overlay, so there is no need for a *locations* state component to map one name to another. The members of the underlay are stable and stationary. Routing is static except for failures, and the forwarding tables are fully populated. Because there is no flooding, there is no need to restrict the links to a spanning tree, and all of the physical links between switches can be fully utilized. The underlay can run an efficient routing protocol, such as a link-state protocol, to compute a shortest path from one edge switch to another.

Routing in the overlay is unusual compared to routing in general, because every edge switch is directly linked to every other edge switch. This means that an inter-switch route to a host can be identified simply by the MAC address of the host's edge switch, and is exactly the same no matter which switch needs the route! Thus inter-switch routing is a mapping that is *global* within the layer. Note that, despite the use of a single global directory, the mapping performed is indeed part of *routing* within the layer (i.e., the *routes* mapping), *not* a *locations* mapping between two layers. The underlay layer in Figure 9 exists to make the *routing* between the edge switches more scalable, not to implement the *link* between the two end-points.

As with Ethernet LANs, each switch has a routing table that is populated lazily (*e.g.*, through MAC learning). The difference lies in what happens when a switch needs a route to an unknown destination. Rather than flooding, it looks the route up in a global routing directory.

When a host moves, the directory is updated with the new route to the host. The exact update mechanism differs from one overlay design to another, depending on whether mobility is planned (*e.g.,* virtual-machine migration in a data center) or unplanned (*e.g.,* a mobile device moving within a campus). In a data center, a central controller that triggers virtual-machine migration can also update the directory with the new
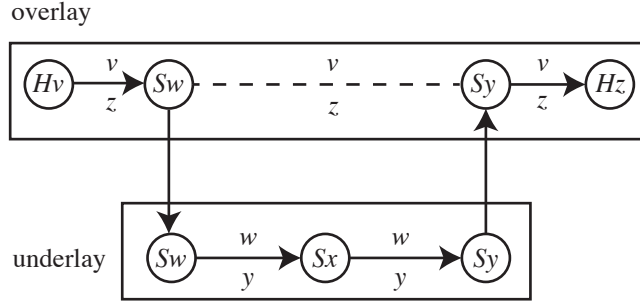
Figure 9: The path of a message through two layers and three switches.

route [10, 23]. If the directory cannot be informed in advance that a host is moving, the new local switch can learn that a new device has connected and subsequently update the directory [18].

The routing directory in an overlay is an important part of its design, and may have many features to make both queries and updates fast and efficient. Different designs have different directory structures. VL2 [10] and NVP [23] run a centralized directory on a collection of server machines. In these designs, if the ingress switch *Sw* does not know the route for host *Hz*, *Sw* queries a directory server to learn the route *Sy*. In contrast, SEATTLE [18] implements the directory as a "one-hop Distributed Hash Table" [11] running directly on the switches. If the ingress switch *Sw* does not know the route for host *Hz*, *Sw* computes the hash of the *Hz*'s MAC address and forwards the message over a single overlay link to the switch responsible for this hashed value. This switch, in turn, forwards the message to *Hz*'s local switch *Sy* and informs switch *Sw* of the route for *Hz* so that future messages flow directly from *Sw* to *Sy*.

To improve the speed of mobile handoff, the ingress switch *Sw* can receive an update when a host moves to a new location. To perform these updates, the directory could maintain information about all ingress switches that recently queried the directory for a route to *Hz*. However, this can require the directory to maintain a large amount of state. Instead, when a host moves, the directory can update the mobile host's old local switch. Upon receiving a message for the mobile host, the old local switch can both forward the message to the mobile host and send an immediate notification about the new route to the sending switch [18]. This reactive invalidation of stale routes obviates the need for the directory to maintain information about which ingress switches sent queries for *Hz*, while still ensuring rapid invalidation of stale routes.

In addition to SEATTLE, VL2, and NVP, several other designs adopt certain aspects of the overlay solution. The early work on Rbridges [29], and the resulting TRILL [36] standard at the IETF, also forms an Ethernet overlay with shortest-path routing in the underlay. However, instead of having an explicit directory service, TRILL relies on flooding to reach hosts with unknown routes. Rather than flooding on all normal overlay links, TRILL floods on a special multicast link in the overlay. This special link is implemented in the underlay by a multicast tree formed on the underlay topology.

Like VL2 and NVP, the PortLand [21] design has a set of directory servers that allow ingress switches to learn the route to a destination host. Instead of encapsulating a message, PortLand assigns each edge switch a block of host "pseudo-MAC addresses" and rewrites the host MAC addresses at the edge. To enable the use of hierarchical pseudo-MAC addresses, PortLand is restricted to the tree topologies common in data-center networks. Table 1 summarizes the structural characteristics of all five designs.

395

Protocol	Routing Directory	Encapsulation
SEATTLE	one-hop DHT on the switches	simple encapsulation
VL2	directory servers	simple encapsulation
NVP	directory servers	simple encapsulation
Rbridges/TRILL	none, flooding on multicast tree	simple encapsulation
PortLand	directory servers	none, MAC rewriting

Table 1: Ethernet overlay designs for dynamic-routing mobility.

4.3 Comparative resource costs

Concerning storage costs, both Ethernet LANs and overlay designs incur the costs of the forwarding tables in switches. These costs are kept moderate by the fact that the tables are sparsely populated. Because there is no aggregation of names or table entries, the costs of densely populated tables would be too great. In addition to the forwarding tables, the overlay designs incur a storage cost for the routing directory, which maintains global state for the layer.

Concerning update costs, both approaches incur negligible costs for populating forwarding tables lazily through MAC learning. The biggest update cost is the cost of Ethernet flooding. The cost of flooding, in terms of bandwidth, grows quadratically with the size of the network—which makes it a potential scalability problem. Whether it becomes an actual problem or or not depends on its frequency, which depends on both the frequency of moves and the number of correspondents that a mobile host tends to have. SEATTLE, VL2, NVP, and PortLand have no flooding cost, though they do have the additional cost of updating the directory.

Mobility in the overlay designs incurs no path cost. The path cost of Ethernet mobility is significant, because the spanning tree (which is necessitated by flooding) forces paths to be longer and forces some physical links to go unused.

We can measure handoff latency from the instant when the mobile host re-attaches to the network and informs its local switch (before that time no mobility mechanism can take effect). The following scenarios assume that a correspondent switch is sending a steady stream of messages to a mobile host. They describe the elapse of time before messages sent by the correspondent switch (CS) are forwarded to the mobile host at its new attachment.

The Ethernet scenario:

1. The time-to-live of the CS's route to the mobile host expires, if it has not already.

2. CS receives the next message from the correspondent host and floods it.

3. After a round trip to the mobile host, CS learns the new route.

After Step 3, messages sent by CS are forwarded to the mobile host at its new attachment.

In the directory-based overlay solutions (*i.e.,* SEATTLE, VL2, NVP, and PortLand):

1. The directory receives an update of the mobile host's new local switch.

2. The directory informs the mobile host's old local switch of the new route.

3. The next message arrives at the mobile host's old local switch, and is forwarded on the new route.

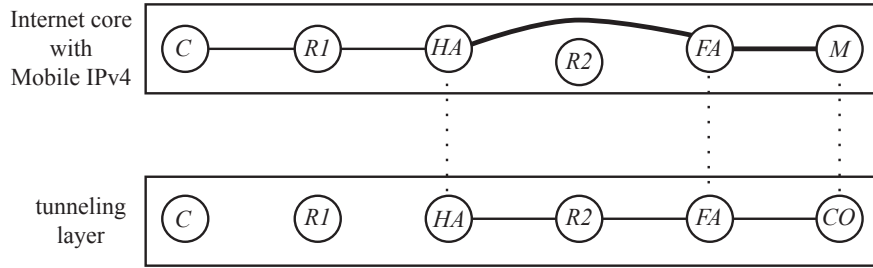4. The mobile host's old switch also informs the CS of the new route.

Figure 10: The path of a message to mobile host *M* with Mobile IPv4. Special links are drawn with heavier lines. Only the links employed in the path are shown.

At Step 3 and after, messages sent by CS are forwarded to the mobile host at its new attachment. If Step 1 of the Ethernet scenario takes time, then the handoff latency of the overlay designs will be smaller than the Ethernet's.

In addition to resource costs, security and privacy are ever-present concerns. In Section 3.3 we noted that DRM usually has minimal security problems because only routers participate in routing. Ethernet flooding is an exception to this rule because it allows hosts to play a role in routing. Malicious hosts can force flooding by filling up the network's forwarding tables. (This would be accomplished by sending many messages from spoofed source MAC addresses.) Severe flooding can cause denial of service. Also, the malicious hosts will receive all the flooded packets, which may contain private information that they wish to see.

5 Examples of dynamic-routing mobility in hierarchical layers

Both of the designs in this section are intended for mobility within the Internet core. For this reason, both must grapple with the problem of a hierarchical name space as explained in Section 3.1. To reduce overhead, both solutions significantly limit the number of routers in the Internet that must store and update state concerning how to reach each mobile node.

5.1 Mobile IPv4

Mobile IPv4 [27, 28] drastically reduces storage and update costs by reducing the number of routers that must have a current route to a particular mobile host to one or two. Also, because each router is responsible for only a limited number of mobile hosts, no router is over-burdened by mobility.

Figure 10 shows the path of a message from correspondent host *C* to mobile host *M* in an Internet core layer with Mobile IPv4. Router *HA* is the *home agent* of *M*, and is supposed to have a route to it at all times. Router *FA* is the *foreign agent* of *M*, meaning that it is local to the subnetwork where *M* is now attached, and currently knows a route to *M* through the subnetwork.

The IP address *M* is in an aggregated routing block such that all messages destined for the block are routed to *HA*. Thus this router need only be the home agent for mobile hosts with IP addresses in its block. The message from *C* arrives at *HA* by means of normal IP routing through router *R1*. The subnetwork of *HA* is *M*'s home subnetwork, so when *M* is at home *HA* has a local route to it.

When *M* is not at home and becomes attached to the subnetwork of *FA*, it gets a local "care-of" IP address *CO* in that subnetwork. *M* informs *FA*, which informs *HA* that it is the current foreign agent of *M*.

To forward messages to *M*, however, *HA* cannot merely forward them toward *FA*. It they were sent out on normal IP links, normal IP routing would send them back to *HA*! Messages to *M* from *HA* and *FA* must be forwarded on special links that are separate from normal IP links.

As shown in Figure 10, the special links in the Internet core are implemented by a tunneling layer below the core layer. The home agents, foreign agents, and mobile hosts of the Internet core are all registered at members of the tunneling layer. Home agents and foreign agents have the same names in both layers, while *M* is attached to member *CO* in the tunneling layer. To forward a message for *M* on its special link, *HA* in the core layer encapsulates the message in another message destined for *FA*, and passes the message to member *HA* in the tunneling layer.

Although the tunneling layer resembles the core layer (see below), its state differs from that of the core layer in several important respects:

- *Routes:* In the core layer, at *HA* messages for *M* are forwarded to *FA* on a special link, at *FA* messages for *M* are forwarded to *M* on a special link, and everywhere else messages for *M* are forwarded to *HA* on a normal link. In the tunneling layer *M* does not exist.

- *Attachments:* Some members of the core layer are attached to members of the tunneling layer.

- *Locations:* The core layer has no *locations* state, at least not related to Mobile IPv4. Although the tunneling layer need not maintain explicit *locations* state for mobile routers because they have the same names in both layers, it must maintain explicit *locations* state for mobile hosts from the core layer. This state, which supplies the current local IP address of a mobile host, is stored in the foreign agent to which it is relevant.

In Mobile IPv4, mobile hosts such as *M* send messages to their correspondents such as *C* though normal IP links. This often creates problems because IP address *M* is not part of the normal routing block of the subnetwork at *FA*. If there is ingress filtering for security in or near this subnet, messages with a source address of *M* will be thrown away. In Section 7 we shall see how Mobile IPv6 eliminates this problem.

Overall this is an interesting architecture because the Internet core layer and the tunneling layer are mostly identical, and share the same implementation. Home agents, foreign agents, and mobile hosts are all aware of the differences between the layers and aware of their dual membership and dual roles. The shared implementation works because none of the other members of the layers need to be self-aware in that way. They always behave the same, without knowing that sometimes their actions contribute to the core layer, while other times their actions contribute to the tunneling layer.

By distinguishing clearly between the two layers, we make it possible to check the correctness of the software for each. It also becomes possible to make further distinctions if advantageous. For instance, implementation of a link between *HA* and *R2* can be shared by both layers, but it might make sense to distinguish links in the two layers for reasons of performance or accounting.

5.2   MSM-IP

MSM-IP [20] is a proposal for using IP multicast to implement mobility. A mobile host gets an IP address *M* in the distinguished multicast block. When the mobile host attaches to a new subnetwork using local IP address *L*, *L* joins the multicast group for *M*, and the previous local address used by *M* resigns from the group.

With IP multicast there is a distinguished set of multicast routers, which are globally distributed and are responsible for routing messages destined for a multicast address to all members of the address's current multicast group. These routers exchange information and forward messages to each other through special

links, exactly as the routers participating in Mobile IP do. The special links are implemented by a tunneling layer, exactly as the special links in Mobile IP are.

With MSM-IP, every subnetwork that supports either mobile hosts or their correspondents must have a multicast router. Messages to mobile hosts (or true multicast groups) are recognized by their distinguished addresses and sent to their local multicast router, where they enter the special multicast routing system.

5.3 Comparative resource costs

The costs of dynamic-routing mobility depend greatly on the number of routers that must have a current route to each mobile host. More routers incur more storage and update costs. Storage and update costs are much greater for MSM-IP than for Mobile IPv4, because an entire network of multicast routers must be updated on each move.

Using fewer routers, on the other hand, incurs more path cost. With MSM-IP path cost is minimal, because a message travels from the multicast router in the sender's subnetwork, along an optimal path through the distributed multicast routers, to the multicast router in the receiver's subnetwork. With Mobile IPv4 path cost can be high, because each message to a mobile host must pass through the home agent, regardless of where the sender is and where the mobile host is. This problem of path cost or "triangular routing" is the reason why the designers of Mobile IPv4 decided to send messages from mobile hosts through normal IP links. They incur no path cost, but they do run afoul of security filtering.[3]

In Section 3.3 we said that dynamic-routing mobility does not in principle require participation of the endpoints. Mobile hosts in Mobile IPv4 and MSM-IP do not have this advantage. The reason that they must have special behavior is that both designs use special routing mechanisms, separate from normal IP routing, to find mobile hosts. Because the routing mechanism is special, it is not necessary to update every IP router when a mobile host moves. But also, because the routing mechanism is special, mobile hosts must also behave differently to interact with it in the correct way.

6 Examples of session-location mobility

In this section we compare four proposals for session-location mobility: the Host Identity Protocol (HIP) [19, 24], the Identifier-Locator Network Protocol (ILNP) [5, 4], the Locator/Identifier Separation Protocol (LISP) Mobile Node [22], and the "route optimization" mechanism of Mobile IPv6 [17, 26] (Section 7.3 will explain how route optimization fits into Mobile IPv6 overall). All but ILNP are IETF standards, and ILNP has resulted in many IETF documents with "experimental" status.

These proposals have many similarities, as they all provide mobility by splitting the Internet core layer (shown in Figure 5) into two layers. These two layers are shown in Figure 11, and also correspond to the two layers in Figure 7. SLM supports the persistence of inter-layer channels that are links in the upper layer and sessions in the lower layer.[4]

In an attempt to use a widely acceptable common terminology, we call the upper layer the *identifier layer*, and the lower layer the *locator layer*. Names in the two layers are referred to as *identifiers* and *locators*, respectively. This common terminology does not necessarily match the terminology typically used to explain each specific protocol.

---

[3]Messages from MSM-IP mobile hosts do not have problems with security filtering because multicast IP addresses are recognizable as belonging to a special category.

[4]Note that the members of the identifier layer are *hosts*, while the members of the locator layer are *interfaces*. This distinction can safely be ignored in this section, but it is important for multihoming as discussed in Section 8.1.
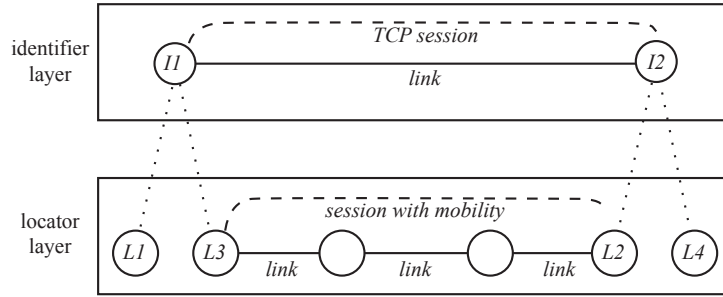
Figure 11: The Internet core layer splits into two layers for well-known examples of session-location mobility. In the identifier layer, session protocols such as TCP run largely unmodified. In the locator layer, the session protocol implements mobility.

Protocol	Identifier	Locator
HIP	(hash of) public key	IPv4 or IPv6 address
ILNPv6	64-bit IPv6 suffix	IPv6 address
LISP Mobile Node	IPv4/IPv6 address (called EID)	IPv4/IPv6 address (called RLoc)
Mobile IPv6	IPv6 address	IPv6 address

Table 2: Comparison of SLM proposals on the basis of names.

Note that Figure 11 is an approximation of the real implementations of these standards, in which the split between layers may be implicit or incomplete. In the geomorphic view, two separate session protocols are employed. In the identifier layer, a largely unmodified TCP implementation provides the usual TCP service as if identifiers were IP addresses. (UDP and other service protocols operate here as well.) In the locator layer, the only purpose of the session protocol is to implement SLM.

6.1 Names

Table 2 compares the four proposals on their choices of names. They differ most on identifiers, which must be globally unique and persistent, but have no other necessary constraints.

HIP places a great emphasis on building in security, so the identifier of a host is the host's public cryptographic key. With the use of keys as identifiers, messages can have self-authenticating source information. Self-authentication provides security within the SLM locator update protocol (see Section 6.3), while the guaranteed presence of a public key makes it easy to protect the channel data with encryption. Identifiers can also be hashes of public keys, which allows for shorter identifiers without sacrificing self-authentication.

ILNPv6 is the IPv6 version of ILNP. Its identifiers are 64 bits long; a host usually chooses a unique identifier for itself by taking the 48-bit MAC address of one of its hardware interfaces and using a standard algorithm to extend it to 64 bits.

The significance of 64 bits is that in IPv6 routing based on hierarchical names and aggregation, the longest possible prefix is 64 bits. This means that at its finest-grained, IPv6 routing examines the first 64 bits of an IPv6 address and points to a subnetwork. The IPv6 address still has a 64-bit suffix to map to an IPv6 interface attached to the subnetwork. In ILNPv6, identifiers are carried in the 64-bit suffixes of IPv6 addresses. In other words, an ILNPv6 locator is derived from an ILNPv6 identifier by prefixing 64 bits that

Protocol	Location Directory
HIP	DNS
ILNP	DNS
LISP Mobile Node	LISP subsystem
Mobile IPv6	home agent for each host has its locator

Table 3: Comparison of SLM proposals on the basis of directories.

indicate a subnetwork where the identified host can be found.[5] This scheme is very efficient in its use of address bits.

By basing identifiers on MAC addresses, which are globally unique, ILNP ensures that hosts have unique identifiers without relying on any administrative authority. ILNP requires that, when a host joins a new subnetwork, it is allowed to choose its own 64-bit address suffix.

LISP Mobile Node and Mobile IPv6 are less interesting. In both cases, identifiers are normal routable IP addresses.

Naming choices have the biggest effect on the deployment opportunities of a design. Mobile IPv6 requires the deployment of IPv6. HIP and ILNP require more changes to TCP because their identifiers are not IP addresses. Deployment of new protocols is usually incremental, which means that upgraded hosts and subnetworks must interoperate with legacy hosts and subnetworks. This raises the following interesting question: if an IP-based SLM protocol is interoperating with ordinary IP, does the ordinary Internet layer coincide with the identifier layer or the locator layer? Interoperation will work best if the ordinary Internet layer coincides with the identifier layer. This composition of layers (making the SLM identifier layer and Internet layer into one) will work best if SLM identifiers look like ordinary IP addresses.

6.2 Directories

An implementation of SLM requires a globally accessible implementation of *locations* in the locator layer, mapping identifiers to locators. Table 3 compares the four standards on their choices of location directory or other mapping implementation.

LISP Mobile Node inherits its directory mechanism from LISP [8], which is an IETF standard designed for a different purpose (multihoming of large-scale enterprise subnetworks), and not originally intended for the support of mobility. The directory mechanism is a special-purpose distributed subsystem of directory servers. While this requires a substantial initial investment, it does give the deployer maximum freedom. For example, different deployments could use almost any name space as the set of identifiers.

Both HIP and ILNP usually make use of the Domain Name System (DNS) as a scalable, highly available directory subsystem. When they do, note that their use of DNS is different from the ordinary use of DNS, which is to map application-level names (domain names) to IP addresses (which, as we have seen, can sometimes be interpreted as identifiers or locators). An IP-oriented SLM mechanism requires a directory or the equivalent to map IP-oriented identifiers to IP-oriented locators, which has nothing inherently to do with application-level names.

DNS relies on the hierarchical structure of domain names both for scalability of lookups and to manage the distributed administration of DNS servers. Consequently, a DNS lookup must begin with a domain name. Thus when HIP and ILNP use DNS as their directory subsystem, every mobile host must have a domain name that serves as a key for finding its current location, even though the domain name is not in the

---

[5]This is different from ILNP terminology, in which the 64-bit prefix is itself the "locator."

name space of either of the relevant layers, and may not be needed for any other reason. For example, in the use of a client-server service, the server usually has a domain name while the client does not. But if the client is mobile with HIP or ILNP, it must have a domain name, known to the server, for this purpose alone.

For both HIP and ILNP it is necessary to add new record types to those stored by DNS servers, because the value being looked up is not always an IP address. Finally, the DNS server with the authoritative copy of a locator must send it out with a small time-to-live, preferably zero. Otherwise other DNS servers will cache the information for longer times, impeding responsiveness to changes of location.

The route optimization (SLM) mechanism of Mobile IPv6 is an adjunct to the Mobile IPv6 DRM implementation (see Sections 5.1 and 7.3). Because the DRM implementation uses home agents, the SLM implementation uses them also. The current locator of an identifier can always be obtained from its home agent. Mobile IPv6 may be less reliable than other designs because a home agents is a single point of failure with respect to its mobile hosts. Home agents do not necessarily have the built-in redundancy and high availability that the directories of the other designs have.

6.3 Locator update protocols

An implementation of SLM must have a protocol through which mobile nodes update the directory and their correspondents after a move. The protocol must have security to prevent updates from unauthorized hosts.

It would take far too much space to report on how each proposal meets these requirements. Also, many standards provide a menu of implementation alternatives, some of them better-documented than others. In lieu of this detail, we will merely touch on a few of the design issues for SLM protocols.

Even without the problem of simultaneous handoff (as introduced in Section 3.2), an update protocol can suffer from lost or re-ordered messages. If a correspondent node or directory receives two different update messages from a mobile host in the wrong order, it could retain an obsolete locator for the mobile node. If a correspondent node or directory determines from sequence numbers that an update message has been lost, it might wait forever for a retransmission that will not come because the mobile node is somewhere else and will not receive the retransmission request. These bugs have been discovered in real SLM protocols [3]. In general, the two techniques to rely on are (1) version numbers rather than sequence numbers, and (2) some form of protocol verification to insure against otherwise-almost-inevitable mistakes.

If an endpoint loses track of the session's other endpoint because of simultaneous handoff, loss of update messages, or a protocol bug, it can always get the current locator by making a new lookup in the directory. In general, an SLM protocol can be made more robust by having mobile nodes report their locators to the directory frequently, and having correspondent nodes refresh their cached locators from the directory frequently. This robustness comes at the cost of increased overhead in the form of message traffic.

HIP uses a different method to solve the problem of simultaneous handoff. When a mobile host moves, its old locator is adopted by a "rendezvous server" that keeps track of its new locator. The rendezvous server intercepts control messages destined for the old locator, and forwards them to the new locator. Even when both endpoints of a session move at the same time, their update messages will reach each other through rendezvous servers. As always, data messages travel directly between hosts.

6.4 Encapsulation

Like Table 1, Table 4 compares the standards on the basis of how they encapsulate overlay messages as they travel through an underlay. Note that there are two versions of the Mobile IPv6 standard which differ in this respect (the newer [26] supersedes the older [17], so this comparison is of academic interest only).

Protocol	Encapsulation
HIP	encapsulation with IPSec Encapsulating Security Payload
ILNPv6	none, identifier is extracted from locator
LISP Mobile Node	simple encapsulation
Mobile IPv6 (RFC 3775)	simple encapsulation
Mobile IPv6 (RFC 6275)	Home Address destination option and Type 2 Header

Table 4: Comparison of SLM proposals on the basis of encapsulation.

Consider a message being sent from left to right in Figure 11, at a time when identifier *I1* has locator *L3* and identifier *I2* has locator *L2*. In the simplest implementation, the message consists of a message with source *I1* and destination *I2*, encapsulated in a message with source *L3* and destination *L2*. There are other possibilities, however, motivated by the desire to conserve space in message headers. This is a serious concern in IPv6, where each of the four address fields is 128 bits long.

LISP Mobile Node and the original version of Mobile IPv6 use simple encapsulation as above. HIP does also, with the additional proviso that the message body containing the identifiers is protected with IPsec. In ILNP each identifier is a suffix of its current locator, so it need not be sent separately.

In the revised Mobile IPv6 standard, there is an optimization apparently based on the observation that, most of the time, only one of the endpoints of a session will be mobile. For a stationary node, the identifier and locator are always the same, and need not be sent twice. So a message from the mobile node needs a source identifier and not a destination identifier, and a message to a mobile node needs a destination identifier and not a source identifier.

Now let us assume that *I1* is mobile and *I2* is stationary. For messages from *I2* to *I1*, a destination identifier is needed. The revised Mobile IPv6 standard uses a special Type 2 header. This is a kind of "source routing" header, allowing the source to provide a list of destination addresses through which the message must be routed. The messages have destination list (*L3; I1*), where the second hop from *L3* to *I1* is internal to the mobile host. For messages from *I1* to *I2*, a source identifier is needed. The extra source address *I1* is inserted using a special "Home Address destination option." This option is an extension to IPv6 allowing an extra field in a message header. In this way, the extra identifiers can be added to messages only when needed.

One might speculate that such a complex optimization would cause trouble in the form of further, cascading complexities, and this is indeed the case. There are elaborate rules in [26] for processing messages so that IPsec works correctly: each message must be processed partially with *I1* in the ordinary source or destination field, and partially with *I1* moved to the Type 2 header or Home Address destination option field. Note that this is an interaction that has been explicitly recognized and accommodated in the standard. No one knows how many problematic interactions with other protocols, caused by this optimization, will be discovered if Mobile IPv6 comes into widespread use.

The SLM implementation of Mobile IPv6 is constrained by the need to compose with the DRM implementation of Mobile IPv6 (see Section 7.3). If there were not so many constraints, the size of headers could be reduced without violating the principle of separation of concerns. For example, Figure 12 revises Figure 11 in an obvious way as suggested by the geomorphic view. The session protocol in the locator layer now performs both TCP and SLM functions.

In the identifier layer, the link between *I1* and *I2* is uniquely identified at one end by *I1, p1*, where *p1* is a port number, and uniquely identified at the other end by *I2, p2*. Note that the identifier layer is more like an application layer than an IP layer in that it has no forwarding—only direct links between pairs of communicating endpoints. Consequently, once the link has been set up, there is absolutely no
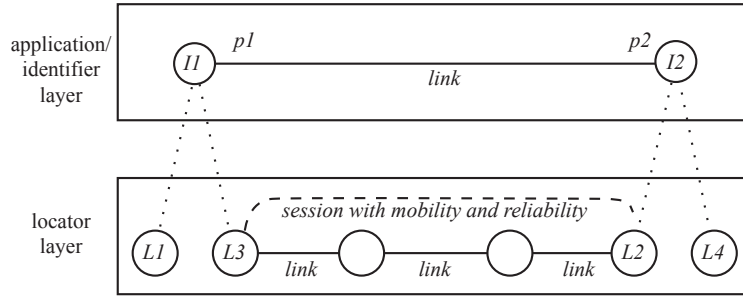
Figure 12: A more efficient version of Figure 11.

need to transmit *I1* and *I2* in data messages. Each endpoint simply uses the port number to pass messages unambiguously across the layer boundary. Figure 12 has many similarities with TCP Migrate [32] and Serval [25, 3], which are other proposals for SLM.

7 Composition of the patterns

7.1 Structural modeling

The geomorphic view of networking is rigorously defined and can be formalized. We have formalized various aspects of the geomorphic view in Alloy, which is the modeling language of the Alloy Analyzer [15], and in Promela, which is the modeling language of the Spin model checker [13]. These are *structural models*.

Networking researchers and practitioners are accustomed to *analytical models*, which are also formal, but quantitative rather than structural. One can assign numbers to some symbols in an analytical model, give the numbers and model to a suitable evaluator, and receive numbers for other symbols in the model.

Structural models are similar, except that evaluation is logical rather than quantitative. We assume that some formulas in a model are true, give the assumptions and the model to a suitable evaluator such as the Alloy Analyzer or Spin, and receive information about the truth of other formulas. We either learn that a formula is true, or get a counterexample showing why it is not true.

Sections 3 through 6 should have made clear why the term we use to describe these models is *structural*. We use them to describe hardware and software structures within networks, and to compare mechanisms based on where their structures are similar and different. We also use them to show how structural decisions constrain other decisions and affect important properties such as scalability and interoperability. In the next subsection, we will mention some additional knowledge gained with the help of structural modeling.

7.2 Generating the design space of mobility

An instance of mobility is an isolated episode in which one layer member changes its attachment from one underlay member to another. One of our goals is to give network architects the freedom to handle any instance of mobility with any mobility pattern at any level of the layer hierarchy. This should enhance efficiency and scalability by allowing solutions that are finely tuned to the characteristics of the problem they are solving.
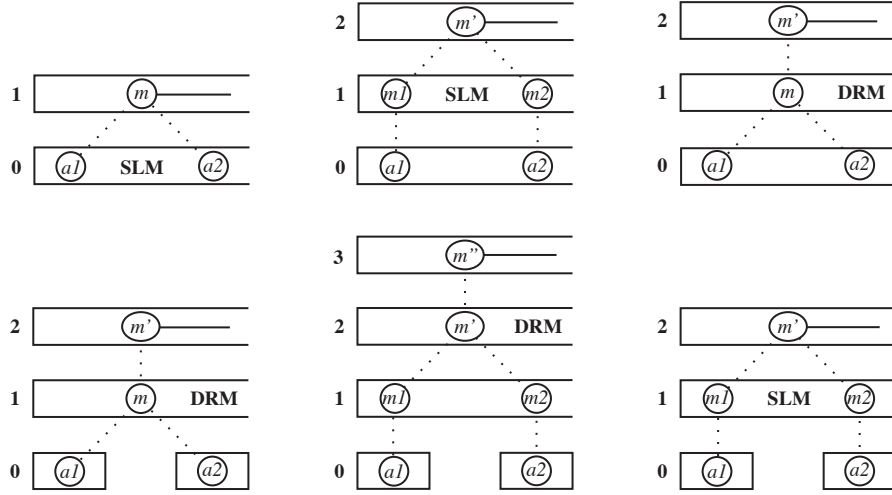
Figure 13: Generating the design space of mobility.

The first step was to identify the two possible implementation patterns and to provide sufficiently abstract versions of them (Section 3). The next step, taken in this section, is to show that any instance of mobility can be implemented with either pattern at almost any level of the layer hierarchy.

In the left column of Figure 13, top half, we see a fundamental instance of mobility in which the old and new locations are in the same layer at Level 0. As notated, the channel at Level 1 can be preserved by session-location mobility (SLM) at Level 0. In the left column, bottom half, we see a fundamental instance of mobility in which the old and new locations are in different layers at Level 0. As notated, a channel at Level 2 can be preserved by dynamic routing mobility (DRM) at Level 1.

The middle column of the figure shows the effects of a "lifting" transformation in which each mobility implementation is moved up a level in the hierarchy. The purpose is to show that mobility can be implemented in many different places, if the current architecture allows it or the designer has control of the content and design of relevant layers. In each case member *m* at Level 1 is replaced by two members *m1* and *m2*. Neither *m1* nor *m2* is mobile, as each has a stationary registration in Level 0 throughout its lifetime. Now member *m'* at Level 2 is mobile. As shown in the figure (top), a channel in Level 2 with *m'* as its higher endpoint can be preserved by SLM at Level 1. Or, as shown at the bottom, a channel in Level 3 with *m''* as its higher endpoint and *m'* as its lower endpoint can be preserved by DRM at Level 2.

The right column of the figure shows where one implementation pattern can be replaced by the other. To replace SLM by DRM (top right), it is necessary to lift the channel up one level. To replace DRM by SLM (bottom right), the channel can stay at the same level, but the mobility must be lifted up a level.

Figure 13 illustrates the crucial point that mobility is about name spaces and member identities in individual layers, and about the mappings between these concepts in adjacent layers. Identity is a fluid concept in software systems, which is why mobility is fluid, and can be pushed around an architecture to appear in different places.

If mobility is fluid, and different implementations are used for different purposes at different levels of an architecture, it follows that any layer could include implementations of either or both mobility patterns. Is this a problem? We have proved that it is not, at least for implementations of the patterns as modeled in Alloy. Verification with the Alloy Analyzer shows that the two patterns can be freely composed, in the
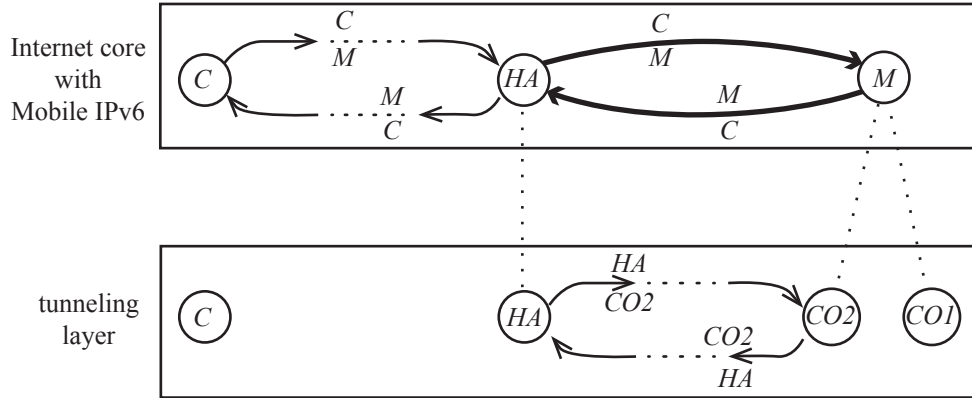
Figure 14: The paths of messages to and from mobile host *M* with the dynamic-routing mobility mechanism of Mobile IPv6. Special links are drawn with heavier lines. Only the links employed in the path are shown.

same layer or different layers of the same hierarchy. They will work together without interference or other undesirable interactions [38].

The limitation of this theoretical result is that, to benefit from proven compositionality, implementations must maintain the minimal separation of concerns inherent in our model of the geomorphic view. If real implementations have state dependencies or interfering actions that are not represented in our model, they are not necessarily compositional even if the theorem says they are. In effect the model is presenting sufficient conditions for compositionality, which can be used as design guidelines for real systems.

7.3 Composition in Mobile IPv6

As we saw in Section 5, Mobile IPv4 is an instance of DRM. Mobile IPv6 [17, 26] uses a similar DRM mechanism, and also composes it with the SLM mechanism described in Section 6. We first consider the DRM mechanism.

Figure 14 is the Mobile IPv6 version of Figure 10. Note that Mobile IPv6 has no foreign agents, as their functions are performed by the mobile hosts themselves. In the figure, both the old attachment of *M* at *CO1* and its new attachment at *CO2* are shown. Normal routers are not shown, being replaced by ellipses in the paths consisting of normal links.

Figure 14 also differs from Figure 10 in showing the source and destination addresses of the messages on every link (source on top, destination below). Thus a message in the core layer with source *C* and destination *M* is forwarded on a special link from *HA* to *M*. The implementation of this special link in the tunneling layer encapsulates the message in another message with source *HA* and destination *CO2*, and sends it through normal IP links and routers.

Figure 14 also differs from Figure 10 in showing the paths of return messages from *M* to *C*. In contrast to MIPv4, return messages from *M* travel on a special link as far as *HA*. At *HA* they enter the realm of normal links and routers. There is no problem with ingress filtering because *M* belongs to the address block of *HA*'s subnetwork, so *M* is a normal source address at that location.

The big problem with Mobile IP is the path cost of routing every message through a mobile host's home agent. Path cost is even worse in Mobile IPv6 than in Mobile IPv4, because it is incurred by messages *from*
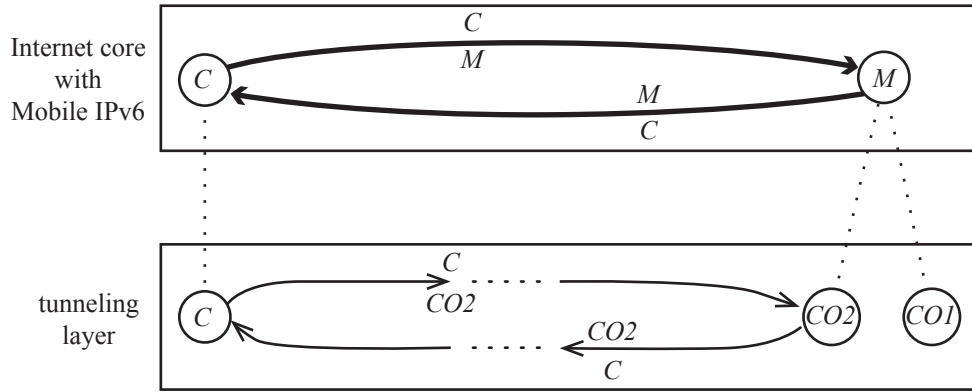
Figure 15: The paths of messages to and from mobile host M with the session-location mobility mechanism of Mobile IPv6. Special links are drawn with heavier lines.

a mobile host as well as messages *to* it. To reduce this problem, Mobile IPv6 standardizes a version of SLM called "route optimization," as already presented in Section 6. SLM is used only after the session between C and M is established, and only if both endpoints have the protocol capability.

Figure 15 shows the SLM mechanism of Mobile IPv6 in the same context as its DRM mechanism in Figure 14. For simplicity, this figure uses simple encapsulation as in [17]. After the endpoints have exchanged messages to set up SLM, they send messages on special links implemented in the tunneling layer. Note that these are *different* special links than those used by DRM (Figure 14). The DRM special links involve HA and will change when M moves. The SLM special links do not involve HA, and will not change from the perspective of the core layer when M moves. With SLM, the only role played by HA in the tunneling layer is to store the directory entry for M. It is not needed originally because when SLM begins the two endpoints are already connected and know each other's locations in the tunneling layer, but it may be needed in case of simultaneous handoff.

Figure 15 also shows the SLM mechanism of Mobile IPv6 in the same context as Figure 11. The "Internet core with Mobile IPv6" layer in Figure 15 is the same as the identifier layer in Figure 11. The tunneling layer in Figure 15 is the same as the locator layer in Figure 11.

So far our discussion of the Internet core/identifier layer has been limited to its links. We have seen that C and M may have a choice of links over which to send their messages. Because the network path associated with each link is different, its performance may be different. It is the job of TCP in the Internet core/identifier layer to smooth over any difficulties caused by diverse paths, for example by ensuring that messages are delivered to an application layer in FIFO order.

The design space generated in Section 7.2 is intended primarily for compositions in which different instances of mobility are managed by different mechanisms or at different levels of the architecture. Composition in Mobile IPv6 is a little different because the exact same instance of mobility—pictured in the figures as M's change of attachment from $CO1$ to $CO2$—is being handled simultaneously by both forms of mobility. The DRM implementation is in the core/identifier layer, while the SLM implementation is in the tunneling/locator layer. DRM is the default mechanism, because SLM can only be used if both endpoints are SLM-enabled.

8   Design considerations related to mobility

Mobility mechanisms lie close to the heart of networking, so they are related to many other communication services and aspects of networking. In this section, we touch briefly on several other topics closely related to mobility.

8.1   Multihoming

Increasingly, mobile devices connect to the Internet via multiple interfaces (*e.g.,* a laptop with WiFi and wired Ethernet interfaces, a smartphone with WiFi and cellular interfaces, or a virtual machine running on a physical server with multiple wired Ethernet interfaces). Since these interfaces usually connect to different administrative domains (*e.g.,* a campus WiFi network and a commercial cellular provider), they must have different IP addresses in different address blocks. There is no name for the Internet host itself, and no way to route to the host itself rather than a specific one of its interfaces.

Sequential multihoming is the use by a host of multiple interfaces, one after the other, during the lifetime of a channel. A mobility mechanism can be used to implement sequential multihoming; in principle either of the mobility patterns can be used to provide it.

If we look at the specific real mobility protocols in Sections 4 through 6, however, we see that the DRM protocols in Sections 4 and 5 work with a single IP address per host, while the SLM protocols in Section 6 work with multiple IP addresses per host. This is an artifact of where these protocols sit in the IP stack rather than an inherent property of the implementation patterns, but it does mean that these SLM implementations are currently a better match for multihoming than these DRM implementations.

Simultaneous multihoming allows a host to use its multiple interfaces to contribute bandwidth to the same channel simultaneously. Simultaneous multihoming is closely related to sequential multihoming and therefore to mobility, but not strictly the same because the multiple interfaces of a host might not be allowed to *change* during the lifetime of a channel. Of course, what we really want is simultaneous *and* sequential multihoming, which is a little more complex than sequential multihoming because it requires protocol extensions so that a layer member can have and use more than one attachment at a time. All of HIP, ILNP, LISP Mobile Node, and Serval already have session protocols capable of simultaneous multihoming.

Because there is such a close association between host multihoming and mobility, the well-known multihoming protocols Multipath TCP [9] and the Stream Control Transmission Protocol (SCTP) [34] are worth studying. They have been used enough to gather experience on the performance aspects of switching from one interface to another, and on how performance affects the buffering and rate control in real transport (session) protocols such as TCP. This experience is as relevant to mobility as it is to multihoming.

8.2   Anycast

Increasingly, the Internet is a platform for users to access services hosted on multiple servers in multiple locations. The appropriate network abstraction for their requirements is *anycast*, in which a service has an *anycast name* that corresponds to a group of servers offering the service. A request for a communication channel to the service can result in a channel to any member of the group of servers.

For simple query-response services like DNS, all server replicas can share a single locator (*i.e.,* an IP address), and rely on IP routing to direct client requests to one of the server replicas. However, IP routing does not guarantee that multiple messages sent to the same IP address would reach the same server replica. In today's Internet, a domain name for a geo-replicated service can map to multiple IP addresses, one for each server replica. This supports communication services with multiple message exchanges between client

and server. It is different from mobility, however, because the higher-level domain name has nothing to do with the channel after the initial DNS lookup.

Alternatively, anycast could be combined with SLM, as it is in Serval [25, 3]. A Serval identifier is an anycast name, and is registered as located at a dynamic group of servers in the locator layer. When a request for a channel to an identifier is handled by the Serval session protocol, the protocol selects the locator of some member of the group. In contrast to the previous paragraph, the identifier remains the name of the channel's higher endpoint throughout its lifetime. The SLM session protocol can thus maintain the channel through both mobility events and changes to the membership of the server group.

8.3 Subnetwork mobility

Mobility proposals typically focus on the movement of a single mobile endpoint, like a mobile device or a virtual machine. However, in some scenarios a subnetwork serving multiple endpoints can move from one location to another. For example, a fast-moving bus, train, or plane may carry a LAN that provides network connectivity to a large collection of passengers.

Dynamic-routing mobility in a hierarchical layer naturally handles subnetwork mobility by updating the routes used to reach the entire aggregated block of names. For example, Boeing had an early in-flight WiFi service that provided seamless mobility for airline passengers by associating each international flight with an IP address block, and announcing the block into the global routing system at different locations as the plane moved [1]. However, this solution required all interdomain routers in the Internet to store and update fine-grained routing information, leading to high overhead.

In our recent work [38], we have shown that subnetwork mobility is merely the mobility of the gateway's attachment to the larger network, and is implemented with the same two patterns as mobility of endpoints. We identified several applications of the design patterns that seem promising for handling combinations of subnetwork mobility and endpoint mobility. These solutions have the property that the mechanism for subnetwork mobility (a bus moves its access point from one roadside LAN to another) is completely independent of the mechanism for endpoint mobility (a user with a laptop gets on and off the bus). Nevertheless, it is not yet clear which solutions for subnetwork mobility would be most viable in practice.

8.4 Incremental deployment and interoperation

Deploying new protocols that span administrative domains is always challenging, since the Internet is a federated infrastructure and cannot easily have everyone upgrade to a new protocol at the same time. Most real deployments are DRM mechanisms that operate within a single administrative domain (*e.g.,* cellular networks, Ethernet LANs, or data-center networks), or require support only from the mobile endpoint and a small number of routers (*e.g.,* Mobile IPv4).

It is not surprising that most real mobility implementations use DRM, because SLM entails many more deployment hurdles. There can be a new set of identifiers, a new global directory service, changes to both endpoints, and even changes to the service interface that are visible to applications. The early SLM protocol TCP Migrate [32] is probably the most deployable, requiring only changes to the operating system at the participating endpoints, but even so it has not had significant deployment.

Nevertheless, as noted in the introduction, the pressure for better network mobility support is mounting. Ubiquitous computing may be a particularly powerful motivator, because an enormous number of sensors and actuators will require network access. This could accelerate the adoption of IPv6, enabling many other changes in its wake.

For incremental deployment, an SLM-enabled host can interoperate with a legacy host in a degraded mode in which mobility does not work but other functions do. For full mobility, an SLM-enabled host can interoperate with a legacy host through a proxy or other middlebox. This raises many new questions concerning how a middlebox is introduced into the path between the hosts, and on the scalability of stateful middleboxes. These new questions must be added to the perennial list of old interoperability questions, such as how to traverse NAT boxes. Identifying effective ways to deploy these protocols incrementally remains an active area of research and standards work.

8.5 Security

All mobility solutions raise important questions about security. In DRM, who is authorized to announce routing changes for an address or address block? In SLM, who is authorized to update the directory service and a mobile endpoint's correspondents? Answering these questions successfully requires unforgeable notions of identity, and secure protocols for sending update messages to routers, directory servers, and endpoints. Can accidental misconfigurations or malicious attacks overload the routers, directory servers, or endpoints? Preventing denial-of-service attacks requires effective ways to limit the work performed before recognizing that messages are unauthorized.

As mentioned in Section 3.3, some people believe that SLM protocols face greater security challenges because arbitrary endpoints can initiate updates to global layer state. On the other hand, SLM protocols provide for persistent identifiers which can be used as the basis for authentication of hosts, a valuable assistance to security. SLM protocols that use DNS as the directory service can update DNS records with an existing secure protocol [37]. Some protocols, like HIP and Serval, embed an endpoint's public key (or a hash of the key) in the identifier; these identifiers support secure communication as well as authentication. Still, security is a rich and important topic warranting a much deeper treatment, especially since new protocols can easily introduce unforeseen vulnerabilities and new threats.

9 Conclusion

In this chapter we have presented an abstract framework for describing, understanding, and comparing approaches to network mobility. As illustrations, we have covered several mobility protocols in some detail. We believe the geomorphic model provides a clear and precise way to understand the considerable similarities between different mobility proposals, allowing discussions to focus on their meaningful distinctions rather than artificial differences in terminology.

We have compared mobility proposals on both qualitative (deployment constraints, security) and quantitative (resource costs, latency) criteria. The basis for making comparisons has been completely *structural*, in the sense of *structural modeling* as defined in Section 7.1. This is important because structural comparisons are vastly easier to obtain than comparisons based on simulation, and should always be the first step in any evaluation project.

In the interest of brevity, our discussions of quantitative criteria have merely suggested trends and trade-offs, rather than providing a more substantive analysis. A true understanding of metrics such as storage cost, update cost, and path cost requires a more detailed characterization of the proposals, including supporting technologies such as routing protocols and directory services. Scalability depends on how these costs grow as the size of a network grows within the expected range.

Equally important, different mobility mechanisms can be composed. Even today it would not be surprising to see dynamic-routing mobility used within an administrative domain, while session-location mobility is used simultaneously across administrative boundaries. The ultimate goal would be to compose performance

models along with the mechanisms they are modeling, so that the performance of a composed solution could be derived from the performance of its components.

Today, many Internet applications that could benefit from mobility use work-arounds instead, satisfying the need for session continuity with *ad hoc* application-specific mechanisms, or simply doing without [12]. This is both an effect and a cause of scant deployment of mobility mechanisms. Most existing mobility protocols operate at fairly low levels in a network architecture, specifically the link, network, and transport levels of the classic Internet stack. At these levels they are expensive, difficult to deploy, or both.

Many of these limitations are unnecessary, as the essence of mobility is simply a dynamic binding of more abstract names to more concrete names. As such, mobility can be easily implemented in middleware as a service to even higher-level application layers. We believe that this is a fruitful avenue for further exploration, particularly because it might be easier to optimize narrowly-targeted implementations of mobility.

More generally, we believe that the geomorphic view promotes common terminology, modularity, separation of concerns, discovery of design patterns, composition, rigorous reasoning, and code reuse in networking. While widely appreciated by software engineers, these concepts have been less central to the study of networking. We believe that the geomorphic view should be extended to understand other important aspects of networking. We also believe that an appreciation of these concepts would be valuable for networking researchers and practitioners alike, far beyond the treatment of any one subject like network mobility.

References

[1] ABARBANEL, B. Implementing global network mobility using BGP. NANOG Presentation, http://www.nanog.org/meetings/nanog31/abstracts.php?pt=NTk1Jm5hbm9nMzE=&nm=nanog31, May 2004.

[2] AKYILDIZ, I. F., XIE, J., AND MOHANTY, S. A survey of mobility management in next-generation all-IP-based wireless systems. *IEEE Wireless Communications 11*, 4 (August 2004), 16–28.

[3] ARYE, M., NORDSTROM, E., KIEFER, R., REXFORD, J., AND FREEDMAN, M. J. A formally-verified migration protocol for mobile, multi-homed hosts. In *Proceedings of the International Conference on Network Protocols* (October/November 2012).

[4] ATKINSON, R., BHATTI, S., AND HAILES, S. ILNP: Mobility, multi-homing, localised addressing and security through naming. *Telecommunication Systems 42*, 3-4 (December 2009), 273–291.

[5] ATKINSON, R., BHATTI, S., AND HAILES, S. Evolving the Internet architecture through naming. *IEEE Journal on Selected Areas in Communication 28*, 8 (October 2010), 1319–1325.

[6] CLARK, D. The design philosophy of the DARPA internet protocols. In *Symposium proceedings on Communications architectures and protocols* (New York, NY, USA, 1988), SIGCOMM '88, ACM, pp. 106–114.

[7] DAY, J. *Patterns in Network Architecture: A Return to Fundamentals*. Prentice Hall, 2008.

[8] FARINACCI, D., FULLER, V., MEYER, D., AND LEWIS, D. The locator/ID separation protocol (LISP). IETF Request for Comments 6830, January 2013.

[9] FORD, A., RAICIU, C., HANDLEY, M., AND BONAVENTURE, O. TCP extensions for multipath operation with multiple addresses. IETF Request For Comments 6824, January 2013.

[10] GREENBERG, A., HAMILTON, J. R., JAIN, N., KANDULA, S., KIM, C., LAHIRI, P., MALTZ, D. A., PATEL, P., AND SENGUPTA, S. VL2: a scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication* (New York, NY, USA, 2009), SIGCOMM '09, ACM, pp. 51–62.

[11] GUPTA, A., LISKOV, B., AND RODRIGUES, R. One hop lookups for peer-to-peer overlays. In *HotOS* (Lihue, HI, May 2003).

[12] HANDLEY, M. Why the Internet only just works. *BT Technology Journal 24*, 3 (July 2006), 119–129.

[13] HOLZMANN, G. J. *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley, 2004.

[14] ITU. Information Technology—Open Systems Interconnection—Basic Reference Model: The basic model. ITU-T Recommendation X.200, 1994.

[15] JACKSON, D. *Software Abstractions: Logic, Language, and Analysis*. MIT Press, 2006, 2012.

[16] JALAPARTI, V., CAESAR, M., LEE, S., PANG, J., AND VAN DER MERWE, K. SMOG: A cloud platform for seamless wide area migration of networked games. In *Proceedings of ACM/IEEE NetGames* (November 2012).

[17] JOHNSON, D., PERKINS, C., AND ARKKO, J. Mobility support in IPv6. IETF Request for Comments 3775, June 2004.

[18] KIM, C., CAESAR, M., AND REXFORD, J. Floodless in SEATTLE: A scalable Ethernet architecture for large enterprises. In *Proceedings of ACM SIGCOMM* (2008).

[19] MOSKOVITZ, R., AND NIKANDER, P. Host identity protocol HIP architecture. IETF Network Working Group Request for Comments 4423, 2006.

[20] MYSORE, J., AND BHARGHAVAN, V. A new multicasting-based architecture for Internet host mobility. In *Proceedings of the 3rd Annual ACM/IEEE International conference on Mobile Computing and Networking* (1997).

[21] MYSORE, R. N., PAMBORIS, A., FARRINGTON, N., HUANG, N., MIRI, P., RADHAKRISHNAN, S., SUBRAMANYA, V., AND VAHDAT, A. PortLand: A scalable fault-tolerant layer 2 data center network fabric. In *Proceedings of ACM SIGCOMM* (August 2009).

[22] NATAL, A. R., JAKAB, L., PORTOLÉS, M., ERMAGAN, V., NATARAJAN, P., MAINO, F., MEYER, D., AND APARICIO, A. C. LISP-MN: Mobile networking through LISP. *Wireless Personal Communications 70*, 1 (May 2013), 253–266.

[23] NICIRA. It's time to virtualize the network. http://nicira.com/en/network-virtualization-platform, 2012.

[24] NIKANDER, P., GURTOV, A., AND HENDERSON, T. R. Host identity protocol (HIP): Connectivity, mobility, multi-homing, security, and privacy over IPv4 and IPv6 networks. *IEEE Communications Surveys and Tutorials 12*, 2 (April 2010), 186–204.

[25] NORDSTRÖM, E., SHUE, D., GOPALAN, P., KIEFER, R., ARYE, M., KO, S., REXFORD, J., AND FREEDMAN, M. J. Serval: An end-host stack for service-centric networking. In *Proceedings of the 9th Symposium on Networked Systems Design and Implementation* (April 2012).

[26] PERKINS, C., JOHNSON, D., AND ARKKO, J. Mobility support in IPv6. IETF Request for Comments 6275, July 2011.

[27] PERKINS, C. E. Mobile IP. *IEEE Communications* (May 1997).

[28] PERKINS, C. E. IP mobility support for IPv4. IETF Network Working Group Request for Comments 3344, 2002.

[29] PERLMAN, R. Rbridges: Transparent routing. In *Proceedings of IEEE INFOCOM* (2004).

[30] ROSCOE, T. The end of Internet architecture. In *Proceedings of the 5th Workshop on Hot Topics in Networks* (2006).

[31] SCHULZRINNE, H., AND WEDLUND, E. Application-layer mobility using SIP. *Mobile Computing and Communications Review 4*, 3 (July 2000), 47–57.

[32] SNOEREN, A. C., AND BALAKRISHNAN, H. An end-to-end approach to host mobility. In *Proceedings of MOBICOM* (2000).

[33] SPATSCHECK, O. Layers of success. *IEEE Internet Computing 17*, 1 (2013), 3–6.

[34] STEWART, R. Stream Control Transmission Protocol. IETF Network Working Group Request for Comments 4960, September 2007.

[35] STOICA, I., ADKINS, D., ZHUANG, S., SHENKER, S., AND SURANA, S. Internet indirection infrastructure. In *Proceedings of ACM SIGCOMM* (August 2002), ACM.

[36] TOUCH, J., AND PERLMAN, R. Transparent Interconnection of Lots of Links (TRILL): Problem and applicability statement. IETF Request For Comments 5556, May 2009.

[37] WELLINGTON, B. Secure domain name system (DNS) dynamic update. IETF Network Working Group Request for Comments 3007, November 2000.

[38] ZAVE, P., AND REXFORD, J. Compositional network mobility. In *Proceedings of the 5th Working Conference on Verified Software: Theories, Tools, and Experiments* (May 2013), Springer-Verlag LNCS to appear.

[39] ZHU, Z., WAKIKAWA, R., AND ZHANG, L. A survey of mobility support in the Internet. IETF Request for Comments 6301, July 2011.

413

Enabling Multihop Communication in Spontaneous Wireless Networks

Juan Antonio Cordero, Jiazi Yi, Thomas Clausen, Emmanuel Baccelli

1 Introduction

Since the end of the 20th century, wireless networking is experiencing explosive growth, driven by the popularity of wireless telephony on one hand, and by the development of wireless computer networks on the other hand. Both trends are currently merging into a single attempt: enabling massive wireless Internet access. This phenomenon was inspired by Norman Abramson's pioneer work on packet radio networks [14] in the 1970s, and made possible by the authorization of wireless spectrum use for civil telecommunication purposes, in the 1980s[1]. At first, this deregulation encouraged the democratization of wireless telephony, in the 1990s, thanks to the availability of cheaper, more efficient hardware stemming from Cold War military industry efforts. Since 2000, the introduction of new wireless communication standards using the spectrum authorized for civil use has also fueled the development of wireless computer networks and wireless Internet access.

1.1 Managed Wireless Networks

Wireless Internet access is nowadays mostly provided via link layer technologies such as Wifi (IEEE 802.11 infrastructure mode standards [11]), WiMAX[2] (IEEE 802.16 [13]), UMTS[3] or LTE[4] (3GPP standards [1]), on user terminals such as smartphones, tablets, laptops, *etc*. Such technologies have in common a communication model that is similar to the local wired network model: user terminals (hereafter denominated *hosts*) access the Internet through a dedicated, authoritative infrastructure device (hereafter denominated *router*). In that sense user terminals are competing "consumers" of the same networking resource, which consists locally in access to the router granting internetwork (Internet) connectivity. Routers, on the other hand, are "providers" of the networking resource, and collaborate with one another to provide this resource, *i.e.* internetwork connectivity. This similarity enables IPv4 and IPv6 protocol suites to run quite naturally over such wireless access networks, although IP protocols were in fact designed for wired networks at a time when massive use of wireless Internet access was not yet envisioned.

The basic mechanisms provided by IEEE 802.11 infrastructure mode, WiMAX, UMTS or LTE thus provide communication capabilities over a single wireless hop, between a user terminal and an infrastructure access point. Some extensions of these basic mechanisms provide direct device-to-device communication

---

[1]ISM (Industrial, Scientific, Medical) bands, released in 1985 by US Federal Communications Commission (FCC) for unlicensed use.

[2]Worldwide Interoperability for Microwave Access.

[3]Universal Mobile Telecommunications System.

[4]Long Term Evolution.

414

(as the Wifi ad hoc mode) or even multi-hop wireless communication through relays planned in advance (*e.g.* with LTE or WiMAX). However, these wireless networks all have in common their *managed* nature: they depend entirely on an infrastructure planned and deployed in advance, controlled by an operator. This chapter does not focus on such networks.

1.2 Spontaneous Wireless Networks

Although so far not as successful as managed wireless networking, an alternative type of wireless networks has also emerged since 2000: *spontaneous wireless networks*. Inspired by the Push-To-Talk concept used in walkie-talkies (portable half-duplex radio transceivers developed during the Second World War), spontaneous wireless networks depart from the traditional distinction between routers and hosts, whereby each user terminal (hereafter, *node*) may behave as a router and a host simultaneously. In spontaneous wireless networks, user terminals are thus "prosumers" (*i.e.* both producers and consumers) of networking resources instead of mere consumers. Terminals self-organize to provide multi-hop wireless communications among themselves, with or without help/control from infrastructure devices. Each node may thus simultaneously originate/receive traffic (role of a host), as well as forward traffic on behalf of other terminals (role of a router).

Popular examples of spontaneous wireless networks include mobile ad hoc networks, wireless mesh networks, wireless sensor or actuator networks, wireless smart meter networks, vehicular networks, opportunistic wireless networks or delay tolerant networks. Spontaneous wireless networks are considered as interesting solutions to extend and offload managed wireless networks hampered by increasingly heavy smartphone data communications [9]. They can also increase the resilience of the network in scenarios where infrastructure is not usable, due to a disaster, to the military situation or to the political situation, for instance [6]. In addition, spontaneous wireless networking is an effective way to extend the reach of wireless Internet access, without costly additional infrastructure deployment [5].

Popular link layer technologies providing device-to-device communication in spontaneous networks include so far IEEE 802.11 ad hoc mode [11] and IEEE 802.15.4 [12]. However, in order to provide multi-hop communication in spontaneous wireless networks, additional techniques have to be employed on top of such link layer technologies, and that is the subject of this chapter. The focus is put on the use of standard IP protocols to enable multi-hop wireless communications in spontaneous wireless networks – in order for these networks to effectively blend in the Internet, where appropriate.

Handling heterogeneity at layer 3 Since the early days of computer networking and the first steps of today's Internet, the diversity of networking technologies has been handled exclusively at the physical and the link layers (layers 1 and 2 OSI). The internetworking layer (layer 3) has been conceived as a "convergence layer" in which a single protocol (the Internet Protocol, IP) runs unchanged on top of heterogeneous interconnected networks, as it can be observed in Figure 1 [65].

The development of wireless technology entails however substantial changes in the way that networks are usually represented and conceived. Characteristics of spontaneous wireless networks cannot be handled exclusively at lower layers of communication, as they challenge some of the key assumptions of the IP-based networking architecture. They need thus to be taken into account at layer 3. As more flexible wireless networks are deployed and get increasingly interconnected and integrated with other networks –or in the Internet–, the use of IP over these networks need thus to be adapted or reconsidered. The **first contribution** of the chapter is a review of these considerations, as it elaborates on how the IP-based network architecture
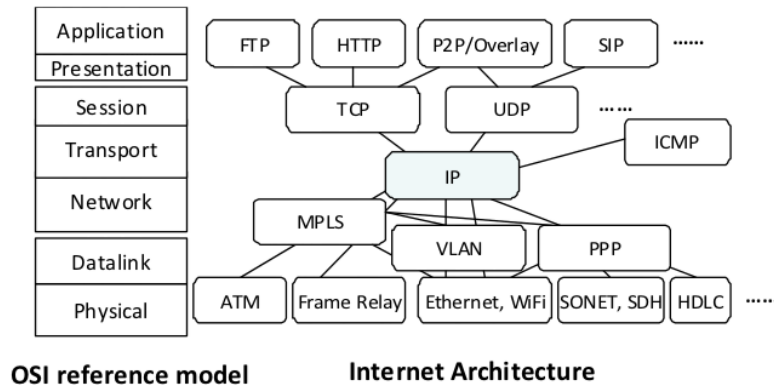
Figure 1: OSI reference model and IP networking architecture [65].

is challenged by spontaneous wireless networks.

1.3 Mobile Ad hoc and Low-Power Lossy Networks

IP protocols are developed, standardized and maintained by the Internet Engineering Task Force (IETF [92]). Most of the IETF's protocol design and standardization activities have so far focused on two categories of spontaneous wireless networks: Mobile Ad hoc Networks (MANETs) and Low-Power Lossy Networks (LLNs).

Mobile Ad hoc Networks (MANETs)   According to the IETF's terminology (defined in RFC 2501 [48]), a MANET consists in a set of "mobile platforms (..) –herein simply referred to as 'nodes'– (..) which are free to move about arbitrarily. The nodes may be located in or on airplanes, ships, trucks, cars, perhaps even on people or very small devices, and there may be multiple hosts per router. A MANET is an autonomous system of mobile nodes. The system may operate in isolation, or may have gateways to and interface with a fixed network" [48]. Note that this definition *allows* router mobility, but it is *not restricted* to mobile networks; the term includes all wireless multi-hop ad hoc networks, regardless of whether they are static or not.

Low-Power Lossy Networks (LLNs)   According to the IETF's terminology (defined in [95]), LLNs are "typically composed of many embedded devices with limited power, memory, and processing resources interconnected by a variety of links, such as IEEE 802.15.4, LowPower WiFi" [95]. LLNs are thus a more specific case of MANETs (as defined in the previous paragraph), in which routers typically operate with constraints on processing power, memory, and energy (battery power). Their interconnections are characterized by high loss rates, low data rates, and link instability. LLNs are comprised of anything from a few dozen to thousands of routers. Supported traffic flows include point-to-point (between devices inside the LLN), point-to-multipoint (from a central control point to a subset of devices inside the LLN), and multipoint-to-point (from devices inside the LLN towards a central control point). Alternative, but similar terminology is employed in `draft-ietf-lwig-terminology` [25], which defines the terms "constrained nodes"

416

and "constrained networks" with various classes of constraints.

Concrete examples of MANETs and LLNs include the following three use cases, selected only for illustrative purposes, to hint at the wide heterogeneity of features, requirements and user expectations that one must address in spontaneous wireless networking.

Vehicular Ad hoc Networks (VANETs)   Communication in VANETs is enabled between moving vehicles in urban scenarios or roadways, (possibly) with fixed devices installed in Roadside Units (RSUs) along the road/street. The combination of vehicles and RSUs forms a mobile, highly dynamic ad hoc network. Devices participating in vehicular networks (either inside vehicles or in RSUs) have neither significant energy constraints nor severe computational limitations, but those installed in vehicles are not, in general, cooperative and willing to dedicate resources to others' communication. Research in these networks has typically focused on safety applications, such as distribution along the highway of information about traffic-related events – *e.g.*, jams or accidents [90]. Other purposes could be also considered, such as dissemination of service availability along the highway (gas stations, tolls, accommodation, etc.). As such, a VANET is a category of MANET.

Community Wireless Mesh Networks   These are cooperative, non-commercial networking projects in which users join and contribute to the deployment of the network, in particular by sharing resources and allowing the use of their devices as networking relays. Several initiatives have flourished in the last years, such as Spain's *Guifi* [3], mostly deployed over the Eastern coast of Spain (Catalonia and Valencia) but also present in many other parts of the country. Other examples include Germany's *Freifunk* [8] and Austria's *Funkfeuer* [2]. Some of them cover large geographical areas and contain thousands of nodes[5]. These networks are typically static, most of the links are wireless links operating in free (unlicensed) frequency bands. Their topology and capacity evolve dynamically, in an unplanned manner, subject to events such as the ingress and egress of users, the subsequent availability of new links and resources or the upgrade of a particular networking region. These networks enable free communication among their users, but they can also provide access to the Internet if there are gateways available. As such, a community wireless mesh networks is also a category of MANET.

Wireless Sensor Networks (WSNs)   WSNs are collections of sensors intended to measure one or several properties of the environment in which they are deployed. Communication facilities required by such networks need to include, at least, the transmission of collected information from the sensors to a gateway or central server that stores and eventually process it, and the transmission of information (*e.g.*, configuration instructions or measurement schedules) from the server to one or more sensors. There is a broad range of information that may be collected and exchanged through WSNs, some examples including climate studies, bird observation, power monitoring in buildings or tracking of patients' health parameters with body sensors. Properties of a WSN may vary depending on the purposes of the sensor deployment, but there are some usual constraints. Sensors are often battery driven, the lifetime of the sensor is limited by the battery lifetime. Protocols for enabling communication within WSNs must therefore be designed with energy consumption and energy-efficiency in mind. As such, a wireless sensor network is a category of LLN.

---

[5]*Guifi.net*, for instance, claims 31865 nodes, 20425 of them being "operating nodes" (last query to `http://www.guifi.net` on April 10th, 2013).

Despite their heterogeneity, these use cases – and other applications of spontaneous wireless networks – have common characteristics, including bandwidth scarcity and need for self-organization. These characteristics both require the use of efficient, highly decentralized routing and flooding mechanisms, able to react quickly to topology changes without overloading the network. This chapter thus focuses more specifically on IP protocols that enable routing and flooding in MANETs and LLNs. While plenty of protocols have been proposed in the literature, only few have been effectively implemented, standardized and used in real-world deployments. The **second contribution** of the chapter consists in:

(1) an analysis of the main implications of wireless mesh characteristics on the task of flooding and routing typically implemented in upper-layer protocols; and

(2) a description and discussion of the key mechanisms and operation of the main protocols deployed so far and standardized at the IETF for routing in MANETs, in LLNs and in heterogeneous wired/wireless inter-networks.

1.4   Reader's Guide

Reader is assumed to be familiar with the main concepts of computer networking and the TCP/IP network reference model, whose terminology is used in this chapter. Interested readers are referred to the book of Tanenbaum *et al.* [91] for details; the glossary at the end of the chapter displays standard definitions of the basic networking concepts. Basic knowledge of the Internet Protocol (IP) operation, addressing model, routing and Internet architecture is also preferable, but not necessary. These elements are briefly overviewed in section 2, in order to better highlight the issues that arise with the traditional IP model in spontaneous wireless networks, addressed in section 3. This section describes the conditions under which wireless communication occurs, and examines their impact on the communication performance and the architecture of spontaneous wireless networks. In particular, the section explains the non-suitability of the conventional IP networking model for spontaneous wireless networks, and discusses an alternative model.

The rest of the chapter focuses on the mechanisms and protocols that have been designed to handle flooding and routing in spontaneous wireless networks, paying a particular attention to the efforts deployed at the IETF. Section 4 motivates and presents the mechanisms, and section 5 describes the routing and flooding protocols that have been specifically designed in the IETF to operate on MANETs and LLNs. Section 6 focuses on the problem of extending legacy Internet routing protocols so that they can efficiently operate on hybrid (wired/wireless) inter-networks. Finally, section 7 concludes the chapter.

2   Fundamentals of IP Networking and Internet Routing

This section introduces the main ideas and concepts that are used as a basis for traditional wired Internet. Subection 2.1 presents the key elements of the IP networking model, including addressing, forwarding and the notion of IP link. Subection 2.2 describes the most relevant routing techniques used in the Internet, and subsection 2.3 overviews the Internet routing architecture, based on the notion of Autonomous System. A certain familiarity with the basics of computer networking is assumed, so no details are provided. This section mainly follows the classic manuals of Tanenbaum *et al.* [91], Comer [42] and Perlman [82]. Interested readers are referred to these resources for further explanations.

2.1 The IP Networking Model

The Internet Protocol (IP) defines the key elements enabling communication in an IP network. This section presents the IP addressing mechanism, the notion of IP link and the routing rule used by router in IP networks – the *longest prefix match* criterion.

Addressing In an IP network, every *network interface* is assigned at least one **IP address** that identifies unambiguously the interface in the network. The IP address format varies depending on the protocol version (32 bits for IPv4, 128 bits for IPv6, see Figure 2), but three elements can be distinguished.

- The **host identifier** is the set of bits that identifies the interface in the network.

- The **network prefix** is the set of bits that identifies the network to which the interface is attached.

- The **network mask** allows to obtain the network prefix and the host identifier from the IP address.

Remark The IP address of a network interface is both an *identifier* and a *locator* of the interface: it indicates *who* is (unambiguosly in the internetwork) the attached interface and *where* is it attached (to which network).

a) IPv4 addressing example: 192.168.0.1/24

Mask (24): $\underbrace{11111111.11111111.11111111}_{\text{netmask (24 bits)}}.00000000$

IP address: $\underbrace{11000000.10101000.00000000}_{\text{network prefix}}.\underbrace{00000001}_{\text{host identifier}}$

b) IPv6 addressing example: $2001 : 0DB8 : 02DE :: 0E13/64$

Mask (64): $\underbrace{FFFF : FFFF : FFFF : FFFF}_{\text{netmask (64 bits)}} : 0000 : 0000 : 0000 : 0E13$

IP address: $\underbrace{2001 : 0DB8 : 02DE : 0000}_{\text{network prefix}} : \underbrace{0000 : 0000 : 0000 : 0E13}_{\text{host identifier}}$
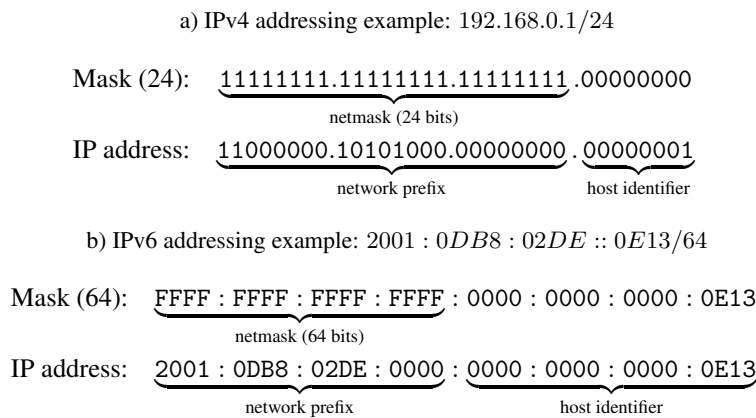
Figure 2: IP address structure, for IPv4 and IPv6.

Based on the information contained in IP addresses from the destination field of the IP header, routers and hosts are able to take decisions upon reception of an IP packet. Trivially, a host receiving an IP packet will accept it only in case that the destination IP address is itself[6] and drop it otherwise. A router receiving an IP packet over an interface will compare the network prefix of the destination IP address with the prefix of its own interface: if it does not match, it may forward it through another interface, according to the IP forwarding rule (see below). In case of forwarding, the router decreases the *Time-To-Live* field (or *hop-limit* for IPv6), to indicate that the corresponding packet has traversed one (more) router in its path to its destination. This leads to the notion of *IP link* (see Figure 3).

---

[6]Or the destination address is a broadcast address or a multicast address to which the host has suscribed.

419

IP Link Two network interfaces, x and y, are connected to the same *IP link* when they can exchange packets in an IP network without requiring that any router forwards them, that is, when packets sent from one interface are received in the other with the same TTL/hop-limit value. This relationship is denoted as $x \sim_{IP} y$.

- In these conditions, communication is performed in a single *IP hop*.

Remark Let a, b and c be network interfaces. The previous definition implies the following properties of IP links:

- *Symmetry*: $a \sim_{IP} b \Longleftrightarrow b \sim_{IP} a$.

- *Transitivity*: $a \sim_{IP} b, b \sim_{IP} c \Longrightarrow a \sim_{IP} c$.

Note that transitivity does *not* hold in terms of routers. The fact that a router R_1 and a router R_2 are connected to the same link, and R_2 and R_3 are connected to the same link, does not imply that R_1 and R_2 have a link in common: R_2 may be attached to two different links (one connecting with R_1 and another with R_3) by way of two different network interfaces.
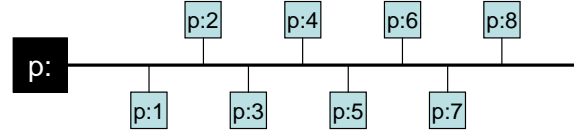

Figure 3: An IP link p: with network prefix p. *IP addresses of nodes in this IP link have the structure* p:i/[p], *for* $0 < i < 2^{[p]}$.

Forwarding rule When source and destination of a packet do not belong to the same IP link, routers receiving the packet compare the IP address of the destination to the prefixes stored in the routing table, and forward the packet through the network interface corresponding to the prefix showing the *longest prefix match*, this is, the prefix in the routing table for which a bigger number of bits are coincident with those from the network prefix of the IP address of the packet destination.

2.2 Main Routing Techniques

Two types of routing techniques currently dominate [82]: *link-state* routing and *distance-vector* routing (with the variant of *path-vector* routing). The main protocols used historically and currently in the Internet are based on these techniques.

Link-State Routing Routers advertise the status of their links (link-state) to the whole network. Link status may include information about the type of link (broadcast, point-to-point...), the link communication capabilities (one-directional, bi-directional, link cost) or the routers to which communication is available through this link. This way, every router in the network receives the link-state of other routers in the network, maintains information about the whole network topology and is therefore able to locally compute network-wide shortest paths, usually by way of Dijkstra's algorithm [52].

- Some examples of this approach are the Open Shortest Path First (OSPF, RFCs 2328 and 5340 [74, 41]) and the Intermediate System to Intermediate System (IS-IS, RFC 1142 [78]) protocols, as well as the Optimized Link State Routing protocol (OLSR, RFC 3626 [39]).

Distance-Vector Routing   A router shares information from its routing table only with its neighbors, indicating *distances* and next hops towards reachable destinations. Neighbor distance is defined according to the current *link metric*, which maps links between routers with estimations of the cost of sending packets through them, represented by scalar values. By receiving the routing tables of all its neighbors, which in turn have been shared with the neighbors of the neighbors, a router is able to identify, for each advertised destination, the neighbor that provides shortest distance and select it as next hop. Distance-vector protocols mostly use the distributed Bellman-Ford algorithm [24, 53] to identify network-wide shortest paths.

- The Routing Information Protocol (RIP, RFCs 1058 [58], 2080 [72] and 2453 [71]) is a prominent example of this family.

Path-vector routing   It is based on the same principle as distance-vector routing, a router advertises to its neighbors the paths to all reachable destinations. Each path is described by indicating the routers that are traversed. This way, local distribution of locally maintained paths enables all routers in the network to build routes to all possible destinations.

- The most prominent example of this family of protocols is the Border Gateway Protocol (BGP, RFC 1771 [85]).

The link-state algorithm requires that every single router has storage and computational capacity to compute locally the shortest-path tree of the network, based on the information received from every other routers, and extract from that tree the next-hop towards every destination in the network. Distance-vector algorithms only require that each router updates the distance-vectors received from their neighbor to infer its own vector of distance vector and select its next-hops.

Due to their computational simplicity, distance-vector protocols were used in the early stages of the Internet. They were gradually replaced by link-state protocols as the ARPANET grew bigger and more complex, due to problems such as the well-known count-to-infinity problem [91] (which appears in the original distance-vector algorithm, but does not appear on path-vector protocols). Poor scalability and slow convergence properties of distance-vector with respect to link-state algorithms were also major reasons to switch from one technique to the other [82].

- The network reaction to a link failure illustrates the differences between link-state and distance-vector algorithms in terms of convergence. In distance-vector algorithms, once a router detects such a failure, it updates the cost of its route towards the lost neighbor and sends the new vector of distances to its neighbors. Neighbors receive this update and *recompute* the cost of the affected route, and then transmit in turn their new vectors. Propagation of topology changes is thus slower than in link-state algorithms, in which a router detecting the failure of the link towards one of its neighbors *floods* an updated topology description which is directly forwarded over the network, without delays caused by route re-computation in intermediate routers [82].

Routing protocols for wired networks used to be *proactive* or table-driven, in which next hop to any possible destination is stored in a table. With the emergence of wireless networks and, more generally, more

dynamic networking architectures coping with more scarce (shared) bandwith, *reactive* routing protocols were then designed and deployed, in which routes were only computed upon request (on-demand).

Proactive routing Routers collect and periodically disseminate topology information over the network; this enables them to maintain proactively (*i.e.*, regardless on whether they are used) routes towards all destinations. This way, routers are able to forward packets at any time to any destination in the network.

Reactive routing A router calculates a route to a destination only when it receives packets addressed to that destination and the routing table does not provide a next hop towards it. In this case, the router triggers a *route discovery* process by disseminating a Route Request (RREQ) packet through the network. The route discovery process terminates when the requested destination or another router knowing a valid route towards the destination reply to the requesting router.

- Dynamic Source Routing (DSR, RFC 4728 [62]) or Ad hoc On-Demand Distance Vector (AODV, RFC 3561 [80]), both used in spontaneous wireless networks, are examples of reactive routing protocols.

Proactive maintenance of next hops to every possible destination in the table requires a constant exchange of control traffic in the network, but enables routers to forward packets immediately after receiving them. Reactive protocols adapt the control traffic to the data traffic requirements: when there is no traffic to route, or the traffic follows known paths, a mostly negligible amount of control traffic (very low or even zero, depending on the protocol) is needed. When a router receives packets to be sent to a destination for which no route is known, the router needs to address a route discovery process over the network – such a discovery process is costly in terms of overhead, and leads to significant delays in the forwarding.

Other routing approaches Some other approaches have been explored for routing over spontaneous wireless networks. In some cases, they rely on additional assumptions about properties and capabilities of the involved devices. If nodes' position is available (for instance, by way of GPS), **geographical routing** approaches are possible: in these protocols, a packet is forwarded to the relay getting closer to the final destination. The Greedy Perimeter Stateless Routing (GPSR) protocol [63] was the first protocol exploring this principle.

2.3 The Internet Routing Architecture

In terms of routing, the Internet is organized as a set of interconnected internetworks, denominated *Autonomous Systems* (see Figure 4). The networks in each Autonomous System are under the same administrative control, and are assumed to perform routing inside the AS *autonomously* from the rest of networks in the Internet. The formal definition of an AS is as follows:

Autonomous System "An *Autonomous System (AS)* is a connected group of one or more IP prefixes [internetwork] run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy" [57], the term "routing policy" denoting the way that routing information is exchanged between (but not within) Autonomous Systems. In the interior of an AS, "routers may use one or more interior routing protocols, and sometimes several sets of metrics" [23].
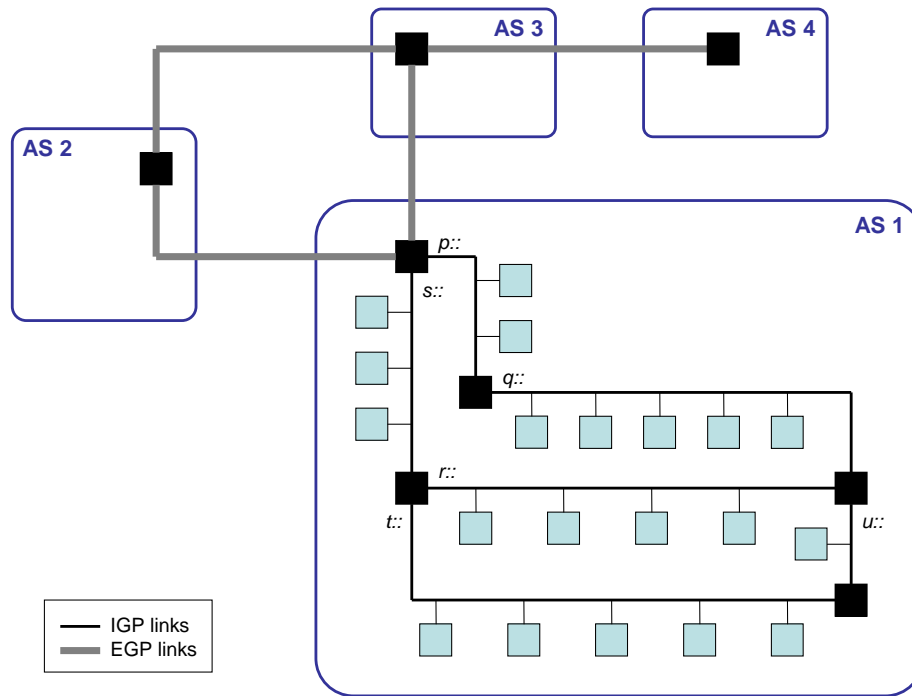
Figure 4: Connection of different Autonomous Systems.

The distinction between routing inside an Autonomous System (intra-AS or intra-domain routing) and routing between different ASes (inter-AS or inter-domain routing) leads to two different types of routing protocols:

(i) *Interior Gateway Protocols* (IGPs), for route discovery and maintenance within an Autonomous System. Intra-domain routing is mostly performed by way of link-state protocols; the most significant link-state routing protocol for TCP/IP networks in the Internet are the Open Shortest Path First protocol (OSPF, [74, 41], described in section 6) and the Integrated IS-IS, an IP variant of the Intermediate Systems to Intermediate Systems (IS-IS) protocol (see [26]).

(ii) *Exterior Gateway Protocols* (EGPs), for route acquisition and information exchange between different Autonomous Systems. The current standard protocol for inter-domain routing is the path-vector Border Gateway Protocol (BGP, [85]).

3 Communication in Spontaneous Wireless Networks

This section describes the basics of communication between wireless devices and presents the main implications for spontaneous wireless networks at layer 3. Physical limitations and derived properties are examined in section 3.1. The implications of these properties in the networking model for spontaneous wireless networks, and in particular the suitability of the IP model, are detailed in section 2.1, is discussed in section
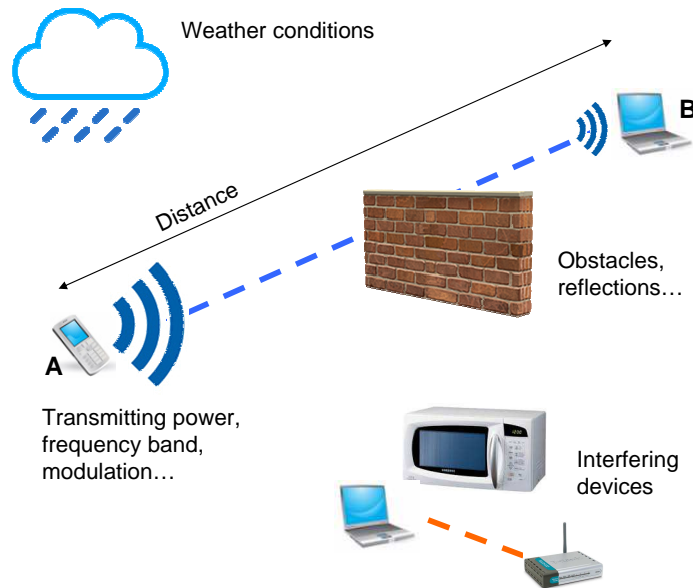
Figure 5: Communication between wireless devices A and B

3.2. Finally, section 3.3 describes an IP-compatible networking model for multi hop communication in spontaneous wireless networks.

3.1   Physical Aspects of Wireless Communication

The fact that two wireless devices in a wireless network are able to communicate to each other depends of several factors (see Figure 5), and some of them are not related to any of the involved devices. The most significant factors include:

  (i) The distance between two devices.

 (ii) The physical properties of the transmitting and receiving antennas: number of transmission/reception antennas, transmission power and antenna directivities.

(iii) Network dynamics: in mobile networks, depending on the relative motion of wireless devices involved in communication, the Doppler frequency shift may have a non-negligible impact.

The modulation and coding schemes used to transmit and receive packets have impact in other physical factors of the transmission, including:

 (iv) The characteristics of the wireless medium: signal frequency band, noise power, effect of weather conditions or *interferences* from other devices transmitting in close frequency bands.

 (v) The physical topology of the *coverage* area: fading caused by obstacles, reflection and absorption causing multi-path interference and signal loss.

Note that, as some of these factors are time-variant and their impact may change rapidly, *e.g.* (iv), some links (or all of them) may have intermittent availability, even if devices keep static.

424

Coverage and Interference   The concepts of *coverage* and *interference* have been mentioned in the previous list, and are some of the key parameters that define the behavior and impact of a wireless interface in a spontaneous wireless network.

Coverage Area   Given a wireless interface A, the *coverage area* of A is the geographical region in which packets transmitted by A can be received and correctly decoded by other interfaces on the same wireless medium as A, when no competing transmission is ongoing. The coverage area of A is denoted by $Cov(A)$.

Interference Area   Given a wireless interface A, the *interference area* of A is the geographical region in which interfaces connected to the same wireless medium as A may be unable to receive or correctly decode other packets when there is an ongoing transmission from A. The interference area of A is denoted by $Intf(A)$.

Remark   Note that, the coverage area of a wireless networking interface is always contained in the interference area of that interface, that is, $Cov(A) \subseteq Intf(A) \forall A$, as shown in the following and represented in Figure 6.

- Let $T > 1$ be the SINR (Signal-and-Interferece-Noise-Ratio) threshold for receiving and decoding correctly packets from a wireless interface. That means that a transmission (*e.g.*, from A) is received and correctly decoded by the receiver, in absence of competing transmissions, if $\text{SINR}|_{I=0} = \text{SNR} = \frac{S}{N} > T$ and discarded otherwise. As received power decreases quadratically with distance from the transmitter, let $S(d) = \frac{P}{d^2}$. Then, the maximum coverage distance is $d_c = \sqrt{\frac{P}{NT}}$. The maximum distance at which there may be interference (from A), d_i, corresponds to the distance to a receiver B such that another transmitter, C, transmitting with the same power P at any distance $d \leq d_c$ from B, would be unable to send successfully a packet in case of concurrent transmission from A. This is, $T = \text{SINR}|_{N \ll I} = \text{SIR} = \frac{P/d_{BC}^2}{P/d_i^2}$. In the worst case, $d_{BC} = d_c$ and $\frac{P/d_{BC}^2}{P/d_i^2} = \frac{d_i^2}{d_c^2} = T$, and therefore $d_i < d_c$.
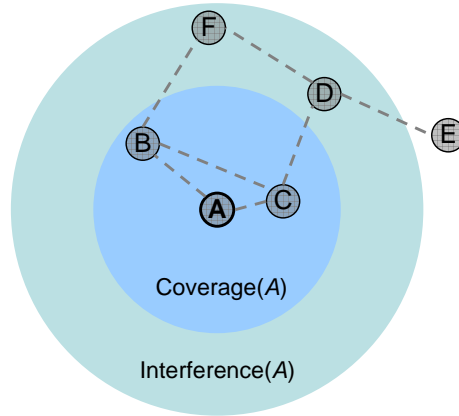

Figure 6: Idealized representation of coverage and interference areas of an interface A.

Due to the variability of factors having impact on wireless communication, coverage and interference areas of an interface are time-variant and in practice their shapes are significantly more irregular than the

circles depicted in Figure 6 [66]. Even within the coverage area at a particular time, when communication is possible, a wireless link is inherently unreliable and prone to transmission errors and packet losses [94], for instance due to interferences from other interfaces in the network or external sources transmitting in the same frequency band.

3.2 IP Model Issues in Spontaneous Wireless Networks

The properties of wireless medium have severe implications for the characteristics of neighbor relationship at layer 3 (L3) in spontaneous wireless networks. In contrast to the case of wired IP links, neighbor relationships between wireless interfaces are not necessarily *symmetric* nor *transitive* [21]. This entails some additional effect that are further illustrated in this section: the *hidden node* problem and the *exposed node* problem.
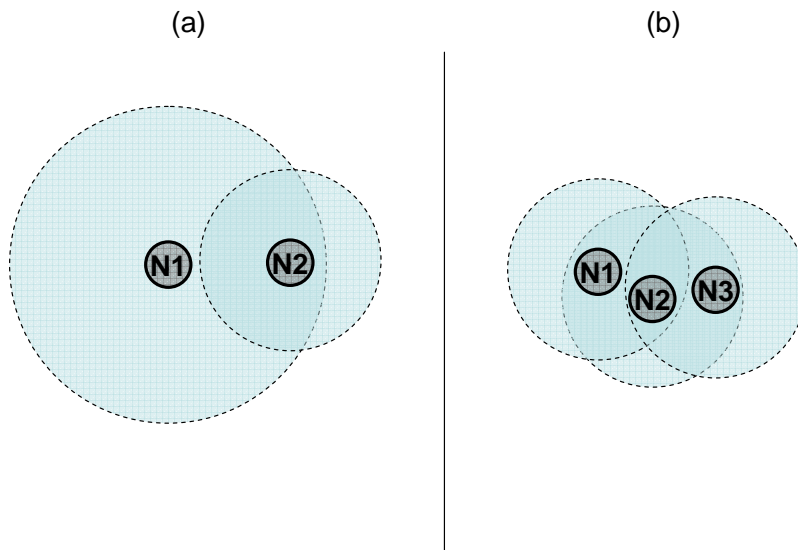

Figure 7: Asymmetry and non-transitivity in neighbor relationships between wireless interfaces.

Non-Symmetric Links   Consider the small wireless network in Figure 7.a: for some reason (powerful transmitter, large antenna, ...) the wireless interface of $N1$ has a large enough coverage area that its transmissions can be received by the wireless interface $N2$. The wireless interface of $N2$, on the other hand, has a much smaller coverage radius, such that transmissions from the wireless interface of $N2$ do not arrive at the wireless interface of $N1$. Thus an asymmetric –or more precisely, an unidirectional– connectivity between the wireless interface of $N1$ and the wireless interface of $N2$ exists: $N2$ sees $N1$ as a neighbor (since the wireless interface $N2$ can receive transmissions from the wireless interface of $N1$), whereas $N1$ does not see $N2$ as a neighbor (since the wireless interface of $N1$ can not receive transmissions from the wireless interface of $N2$). This situation illustrates that neighbor relationships in a wireless network are not necessarily symmetric.

Non-Transitive Links Figure 7.b shows a case of non-transitive links in a 2-hop wireless network. $N1$ and $N2$ are neighbors: the wireless interface of $N1$ is inside the coverage area of $N2$, and therefore $N1$'s transmissions are received at the wireless interface of $N2$ – and viceversa. Observe that the same applies with $N2$ and $N3$: $N2$ and $N3$ are also neighbors. However, direct communication between $N1$ and $N3$ is not possible, as their respective wireless interfaces are outside the coverage area of each other. In a spontaneous wireless network, the fact that $N1$ and $N2$ are neighbors (*i.e.*, can communicate directly) and $N2$ and $N3$ are neighbors as well does not imply that $N1$ and $N3$ are neighbors to each other: neighbor relationship in a spontaneous wireless network is not necessarily transitive.

These two constraints lead to situations that do not occur in traditional IP networks, such as the *hidden node problem* and the *exposed node problem*.
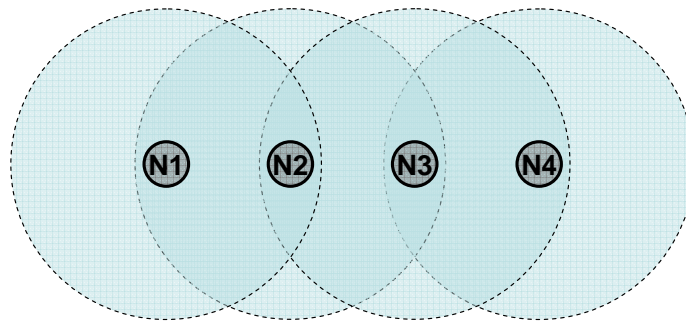

Figure 8: A four-node wireless network, with the (idealized) coverage areas of its nodes.

Hidden Nodes Consider the spontaneous wireless network represented in Figure 8. If $N3$ agrees with its neighbours (N2 and $N4$) that it will, for the moment, have exclusive access to the wireless media via its wireless interface, then $N3$ may go ahead and make a transmission. However, if at the same time $N1$ also transmits over its wireless interface, then the transmissions of the wireless interfaces of $N1$ and $N3$ may appear concurrently at the wireless interface of $N2$ – potentially interfering and causing $N2$ to receive neither of the transmissions. Denoted a *collision*, the possibility and probability of this occurring depends on the L2 (data link layer) mechanisms in place – suffice to observe that such collisions can and do occur when using some common wireless interfaces such as IEEE 802.11. The term *hidden node* originates from the fact that while the node wishing exclusive access to the wireless media may negotiate this with its direct neighbours (in our case $N2$ and $N4$), nodes out of direct radio range (in our case $N1$) are *hidden* to the node requesting media access and cannot thus participate in the negotiation.

Exposed Nodes This can be considered as the dual problem of the *hidden node* situation described above: an *exposed node* is a node that is prevented to transmit due to the transmission of a neighbor, even when the two transmissions would not be interfer. Consider again the network of Figure 8. In the moment in which $N3$ starts a transmission, after having agreed the exclusive use of the wireless channel with neighbors $N2$ and $N4$, $N2$ is an *exposed node* because it is not able to transmit during the transmission of $N3$, in order to avoid collisions. Note however that not all concurrent transmissions from $N2$ would cause collision with the ongoing transmission from $N2$ to $N4$ – in particular, there are no collisions if destinations do not receive several packets at the same time: a packet transmission from $N3$ to $N4$ and from $N2$ to $N1$ would not

cause any collision. The *exposition* of $N2$ to the transmission of $N3$ entails thus a reduction in the available bandwidth.

The hidden and exposed node problems are consequences of the fact that links between wireless interfaces in a spontaneous wireless network are not necessarily symmetric or transitive. These are major differences with the IP networking model (see section 2.1), in which neighbor relationships inside an IP link are assumed to be symmetric and transitive. As these assumptions do not hold necessarily in spontaneous wireless networks, *wireless links in a spontaneous wireless network should not be directly modeled, in general, as IP links* [16].

3.3   An IP-compatible Architectural Model

This section derives from the previously-described observations a general IP-compatible networking model for spontaneous wireless networks [28]. This model enables the compatibility of IP with the characteristics of spontaneous wireless networks, without relying on assumption concerning topology or capabilities of wireless links.

Network View   IP operating on a spontaneous wireless network can be conceived in two separate levels, as represented in Figure 9: the level of traditional IP networking, and the level in which wireless interfaces are connected in a spontaneous wireless network.

- The first level (inner white cloud in Figure 9) contains *wireless interfaces* from routers that communicate with each other by way of *wireless links*, and form a spontaneous wireless network with the non-standard properties described throughout this section. Wireless interfaces in this level present *semibroadcast* communication properties (see paragraph below) and are therefore not required to satisfy the conditions of IP links.

- The second level (outer gray cloud in Figure 9) contains the links between routers and hosts, in which the classic IP link model, as described in section 2.1, applies.

Semibroadcast Interfaces   As mentioned in section 3.1, packets transmitted by a wireless interface A are simultaneously received by the set of wireless interfaces within the coverage area of A, and can be successfully decoded by all those receiving interfaces for which no other transmission causes interference. In a spontaneous wireless network, this set does not contain in general all interfaces in the network. Moreover, as links are not necessarily symmetric in wireless networks and interface coverage areas have a time-variant, irregular arbitrary shape [66], packets from an interface that has received and correctly decoded packets from A are not guaranteed to be received and correctly decoded by A. Wireless **semibroadcast interfaces** are thus "broadcast-capable interfaces that *may* exhibit asymmetric reachability" (as defined in `draft-ietf-autoconf-manetarch` [27]) and *may* not reach all interfaces in the spontaneous wireless network.

Node Morphology   It has been mentioned (see section 1.2) that nodes in a spontaneous wireless network can behave simultaneously as routers and hosts, in contrast to traditional wired computer networks which enforce a clear separation between host and router roles. A first intuition deriving from this observation leads to consider nodes in a spontaneous wireless network as standard hosts with routing capabilities, with an IP subnet prefix assigned to their wireless (semibroadcast) interfaces. This intuition however assumes
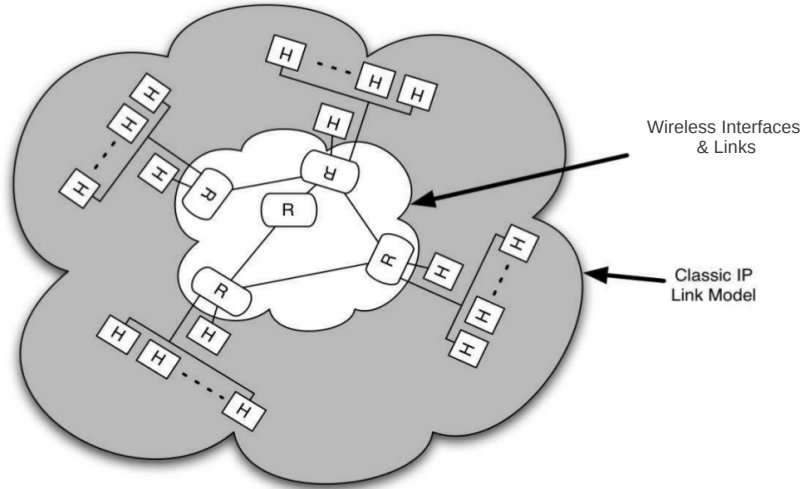
Figure 9: A view of a spontaneous wireless network, following the architecture model described in this section.

implicitly that the semibroadcast interface of the router is attached to the IP link over which the host is configured (and receives its prefix), which is not consistent with the differences between IP links and links between interfaces in spontaneous wireless networks, detailed in section 3.2. Instead, an alternative node model is proposed, which is compatible with the specific characteristics of links between semibroadcast interfaces, and consistent with the two-level network view described above in this subsection. In this model, a node virtually contains *one* router with a wireless interface to interact with the rest of wireless interfaces. As shown throughout this section, links between these wireless interfaces have semibroadcast properties and hence, cannot be configured, in general, as standard IP links in a straightforward manner. A node may also contain one or more hosts: if it does, its hosts belong to the *second level* of the network architecture (see Figure 9). This entails that the links between these hosts and the corresponding router are standard IP links. Figure 10 illustrates the case of a node formed by a router R with a wireless (semibroadcast) interface, and three hosts H_1, H_2 and H_3 connected to R via standard IP links.

This implies that, from the point of view of the hosts, and the applications running on these hosts, connectivity is via a classic IP link. Host applications can thus run unaltered over spontaneous wireless networks, as the specificities of wireless semibroadcast communication have no architectural implications over the links through which hosts are connected: they remain architecturally banished to the *first level* depicted in Figure 9, and handled by wireless interfaces of the routers to which hosts are connected. Characteristics of multi-hop wireless communications can however still impact end-to-end performance experienced by hosts – for instance, TCP may not be able to function as expected [55].

With this model, nodes in spontaneous wireless networks can behave simultaneously as hosts (that is, being source or destination of traffic) and as routers (forwarding other's traffic towards its destination), but hosts and routers interface differently with the rest of the network: hosts are connected to a classic IP link, while
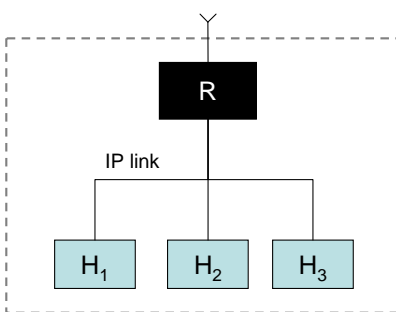
429

Figure 10: A node model for a spontaneous wireless network.

routers are connected to the spontaneous network by way of a semibroadcast interface over links that cannot be assumed symmetric or transitive, for instance.

Impact on IP Addressing   IP addressing model is tied to the notion of IP link, as shown in section 2.1. As the assumptions underlying IP links do not hold in general on links between wireless interfaces, IP links should not be configured by default in spontaneous wireless networks [22]. There are two major implications on not configuring IP links on such a network:

- **Unique Prefixes**. Wireless interfaces must be configured with unique prefixes, *i.e.* such that no two wireless interfaces are configured such that they appear within the same IP subnet. Some common ways to achieve this are:

  – unnumbered interfaces (IPv4) [23];
  – Link-Local Addresses (IPv6);
  – full length prefixes: /128 (IPv6) or /32 (IPv4) prefixes.

  However it is worth noting that prefix lengths shorter than /128 (IPv6) or /32 (IPv4) are possible on the semibroadcast interface, as long as the prefixes are unique to a single wireless interface.

- **Link Local Multicast/Broadcast Scope**. On a wireless interface, a Link Local multicast or broadcast reaches wireless interfaces of neighbor nodes only, regardless of their configured addresses. A Link Local multicast or broadcast on a wireless interface is, thus, a "neighborcast", and is not forwarded nor assumed to be received by all nodes within a spontaneous wireless network.

The principles of the model described in this section have a concrete impact on spontaneous wireless networks operation as described in the remainder of the chapter. On one hand it specifies how IP interfaces should be configured on such networks, and on the other hand it identifies the need for novel protocols and the exact scope of their operation – the *first level* depicted in Figure 9. The following sections will describe techniques and protocols for enabling communication at layer 3 in spontaneous wireless networks, within the scope of the *first level* shown in Figure 9. When needed, assumptions beyond those described in this model will be explicitly detailed in the corresponding protocols.

4 Flooding and Routing in Spontaneous Wireless Networks

As described in section 3, there are important differences in the way that spontaneous wireless networks enable communication between nodes, with respect to the classic fixed/wired networks. These differences have a significant impact in the mechanisms and protocols used in wireless multi-hop scenarios to disseminate information through the network (*flooding*) and find and maintain paths between pairs of computers in the network (*routing*). This section examines several mechanisms that are used in different routing protocols, discusses the issues and problems that these mechanisms have when operating in spontaneous wireless networks, and describes some techniques to fix or overcome these issues.

Section 4.1 explores the use of neighbor discovery procedures in spontaneous wireless networks. Section 4.2 describes techniques to perform efficient flooding over such networks. Finally, section 4.3 presents the problem of estimating link costs and using them to to identify "good routes" over the network.

4.1 Neighborhood Discovery

In many routing and flooding protocols, routers need to be aware of their own neighborhood. This is particularly important (although not only) in spontaneous wireless networks, in which the neighborhood may change frequently during network operation. Routers acquire knowledge about their neighborhood by way of a **neighborhood discovery** mechanism.

Neighborhood Discovery (ND) is the process whereby each router advertises all the routers to which direct communication is possible (*i.e.*, the routers to which there are network links) about its presence in the network. This way, routers receiving such advertisements from other (neighboring) routers gain insight on their own neighborhood.

1-hop and 2-hop neighborhood   Depending on the information advertised by ND messages, receiving routers learn different aspects about their neighborhood. If messages only advertise the presence of the originating router, the receiving router will acquire information about the routers that maintain links to itself. If links are bi-directional (as IP links in standard IP networks), this is sufficient for enabling bi-directional communication between routers: a router receiving a ND message from a neighbor can exchange packets in both directions with it. This is the case of traditional neighbor discovery protocols for Internet, such as the *Neighbor Discovery Protocol* (NDP) for IPv6 [75], which assumes that all the links are bi-directional and is used to actively keep track of which neighbors are reachable.

In spontaneous wireless networks, bi-directional communication availability cannot be inferred from the reception of an ND advertisement, given the fact that asymmetric links are possible (section 3.2). This is taken into account in ND protocols for spontaneous wireless networks. In these protocols, ND advertisements (typically denominated **Hello messages**), contain not only the id of the originating router, but also the list of its current neighbors (*i.e.*, routers from which the originating router has received Hello messages). This enables every router in the network to detect the (1-hop) neighbors with which bi-directional communication is possible, and identify the routers that belong to its 2-hop neighborhood – that is, the set of routers that are 2-hop neighbors or "neighbors of its neighbors".

- **Example** Assume that router A receives a Hello from a neighbor B, in which B indicates to have recently received a Hello from A; then A learns that link A-B is symmetric. As B lists identifiers of all its 1-hop neighbors in its Hello, A learns its 2-hop neighbors through this process.

Together with 1-hop neighbors, additional information may be included in Hello messages – in particular, the cost of the links towards the listed neighbors, when metrics other than hop-count (see section 4.3) is used. Exchange of Hello messages is typically done periodically, although some events may trigger non-periodic Hellos (*e.g.*, changes in the topology).

The *Mobile Ad Hoc Network Neighborhood Discovery Protocol* (NHDP, RFC 6130) [35] is the main ND protocol for spontaneous wireless networks. It is used as auxiliary protocol by other routing protocols that need neighborhood information to take their decisions, such as OLSRv2 (see section 5.1). In NHDP, Hello messages are exchanged periodically and they contain the id of the originating router and the list of its 1-hop neighbors. The IETF also standardized an optimization of NDP for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). This optimization, specified in RFC 6775, adapts the operation of NDP to the lossy conditions of communications and the low-power device constraints of LoWPANs. This is done, for instance, by avoiding unsollicited messages (such as periodic router announcements), reducing the use of multicast for address resolution, limiting the duplicate address detection checks, or enabling better compression algorithms [88].

Other routing protocols include their own Hello mechanism. This is the case of AODV [80] (see section 5.2) or OSPF and its MANET extensions [20, 77, 86] (see section 6.2). In the last case, some approches have been explored in order to avoid redundant notifications and hence reduce control traffic by only reporting changes in the neighborhood occurred since the last Hello transmission: this principle leads to the *incremental Hellos* mechanism used in the Overlapping Relays extension of OSPF (OR/SP [86]) and the *differential Hellos* mechanism used in the MANET Designated Routers extension of OSPF (OSPF-MDR [77]). However, experiments show that the potential benefits (mostly, saved amount of traffic) of these two mechanisms are not significant, in particular when compared with the additional complexity they introduce in the corresponding protocols [19].

4.2 Flooding

Flooding is the process through which information originated in one router is disseminated across the network, so that it can be received by every other router in the network.

The most obvious procedure to perform flooding from a router in a conventional IP network consists of the **pure flooding** procedure:

1. The source router sends the message through all its network interfaces.

2. Every router that receives the message *for the first time* retransmits it over all the network interfaces *except the one over which it was received*.

The fact that each router retransmits only *once* ensures that the process terminates in a finite number of steps. The fact that *all* routers receiving the message retransmit it ensures that the message is received –if there are no packet losses– by every router in the network at least once.

In a spontaneous wireless network, as routers communicate with *all* their wireless neighbors by way of a *single* wireless interface (see section 3), the straightforward usage of this mechanism implies that the source router broadcasts the message to be flooded and the neighboring routers rebroadcast it over the same interface it was received. It is known [76] that such a naive approach is not efficient and does not scale in a wireless multi-hop scenario. Three reasons can be highlighted:

a) excessive retransmissions that reduce the available bandwidth,

b) systematic packet collisions due to concurrent transmissions of wireless interfaces (partly) sharing the same wireless channel, and

c) duplicate packets reception due to the fact that the packet is received and retransmitted over the same interface (and, therefore, transmitted twice in the intersection between the coverage area of the sender and the receiver interface).

Remark Although the three effects are closely inter-related, and all are due to the bandwidth scarcity and the semi-broadcast properties of wireless communication detailed in section 3, it is important to point out that they constitute *different* effects; solving one of them does not necessarily solve the others.

Excessive Number of Retransmissions and Efficient Flooding   In a spontaneous wireless network, a single transmission from a wireless interface is received by the wireless interfaces of all the neighbors within its coverage area. If all routers retransmit the same message as they receive it, this is likely to cause a significant number of *redundant transmissions – i.e.*, transmissions that do not bring new information for *any* of the interfaces receiving them.

Consider the situation in Figure 11, in which node A (in the center) floods a message to all its neighbors, and they in turn retransmit the same message so that it is received by all the 2-hop neighbors of A. In theory, every 2-hop neighbor has received the message – possibly several times. The redundant retransmissions do not bring new information, but increase the probability of collisions. This effect becomes more relevant in a context of bandwidth scarcity and high network router density.

Figure 11: Classical flooding in spontaneous wireless networks.

   Efficient flooding techniques explore different strategies to reduce the number of flooding retransmissions (and therefore, to decrease the amount of traffic overhead involved), while preserving as much as possible the ability of the flooding procedure to reach all (or most of) the routers in the spontaneous wireless network.

For every efficient routing technique, a set of the routers that receive a message (typically, not all of them) are allowed to retransmit it. If efficient flooding reaches all routers in the network, the set of routers allowed to retransmit a message is a *Dominating Set* (DS) in the network graph. As only one router originates and originally sends the message, and every other forwarding router has previously received it via flooding, the set of routers and the wireless links between them usually form a **Connected Dominating Set** (CDS). Given a graph $G = (V, E)$ representing a spontaneous wireless network, where V is the set of vertices (representing network routers) and E is the set of edges (representing network links), a Connected Dominating Set of G is a subset of vertices $D \subseteq V$ with two properties:

1. *Connection.* D induces a connected subgraph of G, that is, any node in D can reach any other node in D by a path that stays entirely within D.

$$\forall x, y \in V, \qquad \exists p = (x, p_1, p_2, ..., p_n, y) \subseteq D \land x\bar{p}_1, p_1\bar{p}_2, ..., p_n\bar{y} \in E$$

2. *Domination.* D is a dominating set of G, meaning that every vertex in G either belongs to D or is adjacent to a vertex in D.

$$\forall v \in V, \qquad v \in D \lor (\exists w \in D : u\bar{w} \in E)$$

Figure 12 displays an example of Connected Dominating Set over a network graph.
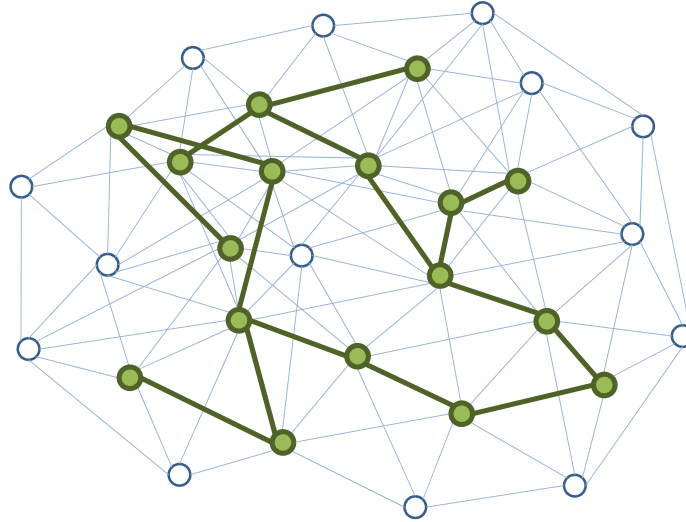

Figure 12: Example of CDS (thick edges) over a network graph of 30 nodes (light edges represent communication between nodes).

The notion of CDS is useful for efficient flooding purposes: several efficient flooding techniques rely on the construction and maintenance of Connected Dominating Sets of forwarding nodes, over which packets are flooded through the network. One of the main techniques based on this principle is the *Multi-Point Relaying* (MPR) technique.

- **Multi-Point Relays** [83] is an algorithm through which a node selects a subset of its 1-hop neighbors (*multi-point relays*) such that each 2-hop neighbor is reachable through (at least) one of the selected

1-hop neighbors (*MPR coverage criterion*). MPR selection requires that the selecting node knows the 2-hop neighbors that will be covered by its MPRs. By using MPR, the retransmission of flooding traffic can be significantly reduced, as shown in Figure 13, compared to classical flooding in Figure 11.
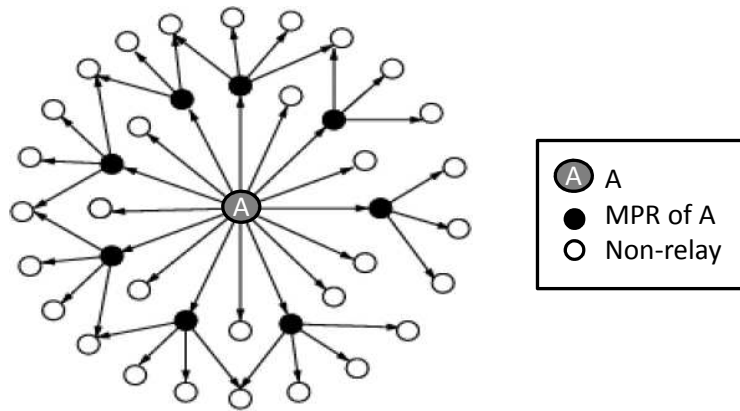

Figure 13: Efficient flooding with Multi-Point Relays.

Remark Note that the MPR coverage criterion does not guarantee by itself that the set of nodes selected as MPRs form a Connected Dominating Set [43]. The set of MPRs is by definition a dominating set, as every node is either a MPR for a neighbor, or is adjacent to its own MPR. As the heuristic for MPR selection is relative to the source, however, the subgraph resulting from MPRs and the MPR links (links connecting nodes with their relays) between them *is not necessarily connected*. This can be easily fixed by adding an arbitrary router and all its links to the subgraph, as proved in Cordero (2010) [43].

Multipoint relays of a router can be used to perform efficient flooding, but the principle can be used for performing other networking operations. The MPR selection algorithm can be slightly modified so that the overlay that includes all links between routers and its (modified) multi-point relays is sufficient to compute shortest paths over the underlying network [43]. This result has been exploited in some extensions of OSPF for MANETs, as shown in section 6.

Systematic Packet Collisions and Jittering Techniques   Consider the spontaneous wireless network of Figure 14, in which router A floods a message through the network. The broadcast transmission of A is received at the same time by B and C, which retransmit the message towards E and D (in the case of B) and D and F (in the case of C). Then, concurrent retransmissions from B and C cause a systematic packet collision from D's perspective.

Remark In this example, the collision could not be detected with any CSMA[7] layer 2 mechanism neither by B nor by E, due to the fact that B and E are not neighbors to each other. B is a *hidden node* for E (and vice versa).

---

[7]Carrier Sense Multiple Access. CSMA is a medium access control (MAC) protocol for wireless networks in which nodes sense the medium before transmitting, and only transmit if the sensed medium is idle, that is, if the node does not detect any ongoing transmission within its reception range. See *e.g.* Tanenbaum *et al.* [91] for reference.
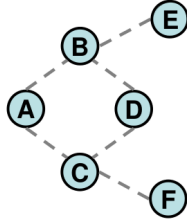
Figure 14: An example of spontaneous wireless network. *(Broken lines denote direct communication)*

Remark Note that this problem cannot be addressed only by way of *efficient flooding* approaches: none of the retransmissions by B and E are redundant, so none of them could be avoided without leaving nodes uncovered (C if B does not retransmit, F if E does not retransmit).

The fact that flooded messages are forwarded simultaneously by wireless interfaces receiving them through the same wireless shared medium may cause packet collisions during the flooding procedure (depending on the network topology at the time of flooding). Unlike other packet transmissions for which the collision probability may vary depending on the traffic pattern, these flooding collisions are *systematic* and will occur, for a given network topology and flooding algorithm, any time that the source node floods a new message (in the example, any time A floods a message).

This effect could be alleviated by allowing routers to wait a random amount of time (denominated **jitter**) before retrasmitting a flooded message, in order to reduce the probability of concurrent transmissions by neighboring wireless interfaces. This technique is known as **jittering**, and has been standardized by the IETF in RFC 5148 [34]. The recommendation from RFC 5148 is that delays are selected following an uniform distribution between 0 and a maximum jitter value, MAX_JITTER. Figure 15 illustrates the effect of jittering techniques in the network example of Figure 14. In the example, node A is flooding a packet to all the other nodes. When node *B* and *C* receive the packet from *A*, instead of retransmitting the packet immediately, they wait a random delay. In this way, simultaneous transmission of B and C (which can cause collision at D in this case) can be avoided.

Although jittering can be theorically implemented at different layers of the protocol stack, it has been shown that its use in layers upper than L3 brings little benefit [54]. As the problem of systematic collisions affects every L3 routing protocol using wireless flooding, jittering techniques can be implemented by different protocols. The addition of random delays in flooded packets impacts differently in proactive and reactive protocols, given the different use of flooding in both routing strategies. Other jittering effects are due to the specifities of the techniques employed in each case. In proactive protocols where jitter is used (as described in RFC 5148 [34], which recommends to introduce random delays and piggyback all pending messages when a transmission is scheduled), such as OLSR or the OSPF extensions for MANETs, jittering leads to longer LSA messages – this may cause additional packet collisions, if jitter values are not configured properly [46, 44]. In reactive protocols such as AODV, where jitter is used for Route Request (RREQ) flooding, the addition of random delays may lead to suboptimal path selections, which can be minimized by adapting the random distribution used for determining jitter values [100, 47].
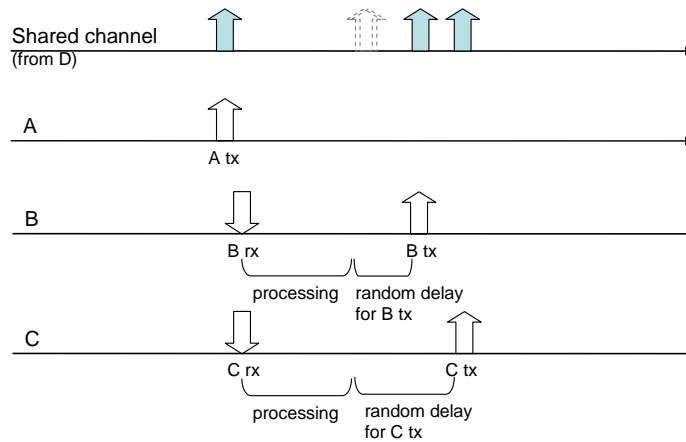
Figure 15: Use of jitter for flooding. Node *A* is flooding a packet in a network. Node *B* and *C* wait a random delay before the packet is retransmitted. The dashed overlapping arrows represent the packet collision that would occur if no jitter were used.

Duplicate Packets and Detection Techniques   The reception of duplicate packets is a common situation in wireless flooding, due to the fact that flooded messages are retransmitted by forwarding nodes over the same wireless interface in which they were received. Consider the situation of Figure 16: router *N2* is retransmitting a broadcast packet received from router *N1* on the same interface as the one over which it was received, so as to ensure receipt also by router *N3*, causing router *N1* to receive the packet a second time.
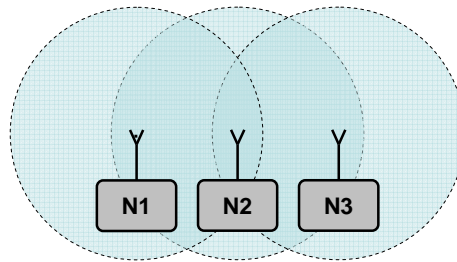

Figure 16: The need for duplicate detection: retransmission over the same interface as a packet was received.

Depending of the protocol and the use it makes of flooding and of flooded packets, the way to detect duplicate packets might be different. In link-state routing protocols such as OLSR or OSPF, for instance, flooded messages are link-state advertisements (LSAs) that are stored locally before being retransmitted, in case they bring fresh topology information; in this case, a duplicate LSA can be easily recognized by checking whether the LSA is already installed in the local Link-State Database (LSDB). If flooded messages are not stored locally, the protocol needs to store state for every forwarded message in order to detect a duplicate – this is, for instance, the strategy of the Simplified Multicast Forwarding (SMF) protocol [70]. In case a received message was already received and forwarded, it is dropped.
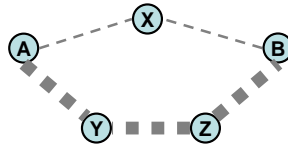
Figure 17: An example of different link metric. $\bar{AX}$, $\bar{XB}$ are unreliable links; $\bar{AY}$, $\bar{YZ}$, $\bar{ZB}$ are reliable links.

4.3 Link Metrics

Metrics are used to evaluate the cost of a link or a path (set of links), so that a routing protocol is able to determine whether a path or a link should be preferred over another.

The simplest link metric is the **hop-count**: a link has metric 1 if it is available, 0 otherwise. Typically, in the early versions of routing protocols for spontaneous wireless networks (e.g., AODV, OLSRv1 for mobile ad hoc networks), only the hop-count metric is used. This way, the metric or cost for a path is equal to the number of hops involved.

When a route between two hosts is being calculated under the hop-count metric, paths with less number of hops are preferred to paths with more number of hops. However, using only minimum hop routes in spontaneous wireless networks may result in suboptimal routing in practice, as the minimum-hop routes are not necessarily the best ones [49, 97].

Figure 17 give an example showing the limit of hop-count metric. The minimum hop route from node A to B is $\{A, X, B\}$. However, the links $\bar{AX}$ and $\bar{XB}$ are poor with high loss rate (but still able to deliver packets), and $\bar{AY}$, $\bar{YZ}$, $\bar{ZB}$ are reliable links. In this case, $\{A, Y, Z, B\}$ is preferred to the route with minimum hop count.

Because of the limitation of hop-count metric, new metrics need to be defined. The link metrics used in spontaneous wireless network are expected to have following properties [51]:

- **Dimensionless**. The metric may correspond to specific physical information, but this knowledge is not used by the routing protocol.

- **Additiveness**. The metric of a route is the sum of the metrics of the links forming that route. It also requires a metric where a low value of a link metric indicates a "good" link a high value of a link metric indicates a "bad" link.

- **Directionality**. The metric from a router A to router B does not need be the same as the metric from B to A. This is a direct consequence of the fact that wireless links are not bi-directional.

The kind of metric used in a network depends on the link/physical layer protocol used, the type of information that is available from lower layers, the application requirements, etc. Some examples of link metric include delay, packet loss probability (Expected Transmission Count, ETX [50], that estimates the average number of transmissions before success over a link), queue length (at the receiver) or data rates (Expected Data Rate, EDR [79]; this metric is not additive, thus a mapping that inverts its ordering must be applied).

5 IETF Routing Protocols for Spontaneous Wireless Networks

Since the late 1990s, in parallel with the emergence and deployment of new and more flexible networking technologies, the IETF has embarked upon a path of designing, developing and standardizing new routing protocols and flooding mechanisms. These protocols and mechanisms are designed for networks with increasingly more fragile and low-capacity links, with less pre-determined connectivity properties and with increasingly constrained router resources.

Most of the IETF protocol design and standardization activity has focused on protocols designed for Mobile Ad hoc Networks (MANETs) and Low-Power Lossy Networks (LLNs), both defined in section 1.3. This section presents the main flooding and routing protocols designed and standardized by the IETF for these types of networks in the last years: OLSRv1 and OLSRv2, RPL, AODV and LOADng.

Routing in MANETs: OLSR and AODV   IETF activities targeting MANETs have converged on the development of two protocols, each one representative of one of the two main routing families (see section 2.2): reactive and proactive routing.

IETF design and standardization work in the reactive routing realm for mobile ad hoc networks first led to the Ad-hoc On-demand Distance Vector protocol (AODV) [80]; the efforts in proactive routing, in turn, led to the Optimized Link State Routing (OLSR) [39]. A distance vector protocol, AODV operates in an *on-demand* fashion, acquiring and maintaining routes only while needed for carrying data, by way of *Route Request-Route Reply* exchanges. A link state protocol, OLSR is based on periodic control messages exchanges, and each router proactively maintaining a routing table with entries for all destinations in the network. OLSR provides low delays in forwarding and has a predictable, constant control overhead – at the expense of requiring memory in each router for maintaining complete network topology. AODV limits the memory required for routing state to that for actively used routes – at the expense of delays for the *Route Request-Route Reply* exchange to take place, and control overhead dependent on data flows.

Based on the operational experience acquired through AODV and OLSRv1, the IETF is currently designing and developing successors for OLSR and AODV. In the first case, the IETF community involved in OLSR has standardized OLSR version 2 (OLSRv2) [37] and its related components (packet format [36] [33], NHDP [35]). Work on AODV version 2 [81] has started, as AODV derivative flourished: IEEE 802.11s [61], which is based on AODV, and the G3-PLC standard [10], published in 2011, which specifies the use of the *6LoW-PAN Ad hoc Routing Protocol* (LOAD, specified in `draft-daniel-6lowpan-adhoc-routing`) [64] at the MAC layer, for providing layer 2 routing for utility (electricity) metering networks.

Routing in LLNs: RPL and LOADng   LLNs can be regarded as a subset of MANETs, but with more stringent constraints in terms of device CPU and memory limitations, and work over more fragile links. Concerning LLNs, two protocols can be highlighted: RPL and LOADng. The IETF explored the problems of routing and adaptation of IPv6 for operation over the IEEE 802.15.4 MAC protocol, accommodating characteristics of that MAC layer, and with a careful eye on resource constrained devices (memory, CPU, energy, ...). Two initial approaches to such routing were explored: *mesh-under* and *route-over*. Both approaches entail different additional assumptions on the (link) characteristics of the addressed spontaneous wireless network, not present in the general networking model described in section 3.3.

1. The *mesh-under* approach performs L2.5 multi-hop routing, that is, provides routing in an adaptation layer between 802.15.4 (MAC layer, L2) and IP (network layer, L3). This L2.5 routing enables the

underlying mesh-routed multi-hop topology to be presented at the network layer as a single broadcast domain.

2. The *route-over* approach, in contrast, exposes the underlying multi-hop topology to the IP layer, whereupon IP routing would build multi-hop connectivity.

The IETF efforts on routing over 802.15.4 initially led to LOAD [64], a derivative of AODV adapted for L2-addresses and mesh-under routing, and with some simplifications over AODV (*e.g.,* removal of intermediate node replies and sequence numbers). However, 6LoWPAN was addressing other issues regarding adapting IPv6 for IEEE 802.15.4, such as IP packet header compression, and efforts to solve routing issues were suspended. In parallel with these efforts, the IETF has also specified the "Routing Protocol for Low-power lossy networks" (RPL), designed to support 6LoWPAN networks in a route-over configuration [98, 96].

However, reasons for using a simplified reactive approach instead of RPL have emerged, including better support for bi-directional data flows such as a request/reply of a meter reading [60], as well as algorithmic and code complexity reasons [38]. These observations led on one hand to a renewed interest in AODV-derived protocols for specific scenarios, resulting in LOADng [32] [30] and AODVv2 [81], while on the other hand leading to the development of an extension of RPL to support reactive path discovery (P2P-RPL [56]).

5.1 Optimized Link State Routing Protocol (OLSR)

OLSR is developed for mobile ad hoc networks, and operates as a table driven, proactive protocol, i.e., it exchanges topology information with other routers in the network regularly. The key concept used in the protocol is that of multipoint relays (MPRs, described in section 4.2), selected nodes which forward broadcast messages during the flooding process. This efficient flooding technique substantially reduces the message overhead as compared to a classical flooding mechanism.

OLSR version 1 was standardized in RFC 3626 [39]. The work continues as OLSR version 2 (OLSRv2 [37]), which retains the same basic algorithms as its predecessor, however offers various improvements, *e.g.* a modular and flexible architecture allowing extensions, such as security, to be developed as add-ons to the basic protocol.

Every router running OLSR in the network generates two types of messages: *Hellos* and *Topology Control* (TC) messages. Information collected through exchange of these messages allows routers to perform the three basic processes of OLSR: Neighborhood Discovery, Link State Advertisements and Routing Set Calculation. Because OLSR (version 1) and OLSRv2 shares the same basic mechanisms, the text below applies to both protocols.

Neighborhood Discovery   OLSR routers discover their neighborhood by exchanging Hello messages with their 1-hop neighbors, as explained in section 4.1. These Hello messages can be generated proactively at a regular interval or as a response to a change in the router itself. In OLSR, a Hello message contains the local interface address(es), and its 1-hop neighbor addresses. With the broadcast of Hello messages to the router's 1-hop neighbor, the router is able to get the topology information in two hops.

Link State Advertisements   Link State Advertisement is the process where by the determined link state information is advertised through the network. For OLSR, this process is optimized by MPR flooding. MPR

selection is encoded in outgoing Hellos.

Routers may express, in their Hello messages, their "willingness" (integer between 1 "will never" and 7 "will always") to be selected as MPR, which is taken into consideration for the MPR calculation. This is useful, for example, when an OLSRv2 network is *managed*, meaning that its topology is known or predictable. The set of routers having selected a given router as MPR is the MPR-selector-set of that router. Each router must advertise, at least, all links between itself and its MPR-selector-set, in order to allow all routers to calculate shortest paths.

Such link state advertisements are carried in Topology Control (TC) messages. TC messages are broadcast by each node to the whole network to build the intra-forwarding database needed for routing packets. A TC message is sent by a node in the network to declare a set of links, which must include at least the links to all nodes of its MPR Selector set, *i.e.*, the neighbors which have selected the sender node as a MPR. TC messages are received by all nodes in the network, by way of the MPR flooding process described above. With the broadcast of TC messages to the whole network, the node is able to get the topology information that is more than two hops away. TCs are sent periodically, however certain events may trigger non-periodic TCs.

Routing Set Calculation   The Routing Set of a router is populated with Routing Tuples that represent paths from that router to all destinations in the network. These paths are calculated based on the Network Topology Graph, which is constructed from information in the Information Bases, obtained via Hello and TC message exchange.

Changes to the Routing Set do not require any messages to be transmitted. The state of the Routing Set should, however, be reflected in the IP routing table by adding and removing entries from that routing table as appropriate. Only appropriate Routing Tuples (in particular only those that represent local links or paths to routable addresses) need to be reflected in the IP routing table.

OLSR does not mandate which algorithm to be used for path calculation, as along as the shortest paths for all destinations from all local OLSR interfaces can be obtained using Network Topology Graph. One example is Dijkstra's algorithm [52].

5.2   Ad Hoc On-Demand Distance-Vector Protocol (AODV)

The Ad hoc On-Demand Distance Vector (AODV) [80] protocol enables dynamic, self-starting, multi-hop routing between participating mobile routers wishing to establish and maintain an ad hoc network. AODV allows mobile nodes to obtain routes quickly for new destinations, and does not require nodes to maintain routes to destinations that are not in active communication.

Compared to pro-active protocols like OLSR, AODV is more suitable under following constraints:

- Few concurrent traffic flows in the network (i.e., traffic flows only between few sources and destinations);

- Little data traffic overall, and therefore the traffic load from periodic signaling (for proactive protocols) is greater than the traffic load from flooding RREQs (for reactive protocols);

- State requirements on the router are very stringent, i.e., it is beneficial to store only few routes on a router.

AODV was initially standardized as an experimental RFC in 2003 [80]. Derivatives of AODV include 802.11s (with HWMP [61]) and LOADng [32]. In the following of this section, the basic mechanisms of AODV, *Route Discovery* and *Route Maintenance* are explained. Then a main derivative work of AODV, called LOADng, is also introduced. Note that at the time of writing, work on AODVv2 [81] has just started, and thus we will not elaborate further on AODVv2 in this chapter.

Route Discovery   The route discovery process is initiated when a source router needs a route to a destination router and it does not have a route in its routing table. The source router floods the network with a RREQ packet specifying the destination for which the route is requested. When the destination router, or an intermediate router with sufficiently up-to-date information about the requested destination, receive the RREQ packet, they generate a Route Reply (RREP) packet, which is sent back to the source along the reverse path. Each router along the reverse path sets up a forward pointer to the router it received the RREP from. This sets up a forward path from the source to the destination.

Route Maintenance   When a router detects a broken link while attempting to forward a packet to the next hop, it generates a RERR packet that is sent to all sources using the broken link. The RERR packet erases all routes using the link along the way. If a source receives a RERR packet and a route to the destination is still required, it initiates a new route discovery process.

5.2.1   Lightweight On-demand Ad hoc Distance-Vector (LOADng)

The *Lightweight On-demand Ad hoc Distance-vector Routing Protocol - Next Generation* (LOADng) [32] is derived from AODV. Compared to AODV [80], it has more concise and flexible message format, and simplified message processing, which makes it more adapted to networks with constrained devices, such as sensor networks. It is also used for ITU Standard G. 9956 [10].

Compared to AODV, LOADng has both simplifications and extensions to be more suitable to LLNs:

- Only the destination is permitted to respond to an RREQ; intermediate LOADng Routers are explicitly prohibited from responding to RREQs, even if they may have active routes to the sought destination. This also eliminates Gratuitous RREPs while ensuring loop freedom, so that the protocol complexity can be greatly reduced.

- A LOADng Router does not maintain a precursor list, thus when forwarding of a data packet to the recorded next hop on the route to the destination fails, an RERR is sent only to the originator of that data packet. The rationale for this simplification is an assumption that few overlapping routes are in use concurrently in a given network.

- Optimized flooding is supported, reducing the overhead incurred by RREQ generation and flooding. If no optimized flooding operation is specified for a given deployment, classical flooding is used by default.

- Different address lengths are supported – from full 16 bytes IPv6 addresses over 6 bytes MAC addresses and 4 bytes IPv4 addresses to shorter 1 and 2 bytes addresses such as RFC 4944 [73]. The only requirement is, that within a given routing domain, all addresses are of the same address length.

- Control messages are carried by way of the Generalized MANET Packet/Message Format [36].

- Using RFC 5444 [36], control messages can include TLV (Type-Length-Value) elements, permitting protocol extensions to be developed.

- LOADng supports routing using arbitrary additive metrics, which can be specified as extensions to this protocol.

5.3 Routing Protocol for LLNs (RPL)

RPL – the *Routing Protocol for Low Power and Lossy Networks* [98] – is an IPv6 routing protocol designed and standardized by the ROLL Working Group in the IETF. It is is intended to be the IPv6 protocol for LLNs and sensor networks, applicable in all kinds of deployments and applications of LLNs.

DODAG Construction   The basic construct in RPL is a "Destination Oriented Directed Acyclic Graph" (DODAG), depicted in Figure 18 . In a converged LLN, each RPL router has identified a stable set of parents, each of which is a potential next-hop on a path towards the "root" of the DODAG, as well as a preferred parent. Each router, which is part of a DODAG (i.e. has selected parents) will emit DODAG Information Object (DIO) messages, using link-local multicast, indicating its respective rank in the DODAG (i.e. distance to the DODAG root according to some metric(s), in the simplest form hop-count). Upon having received a (number of such) DIO messages, a router will calculate its own rank such that it is greater than the rank of each of its parents, select a preferred parent and then itself start emitting DIO messages. The emission of DIO message is controlled by *Trickle* Algorithm [68], to reduce the flooding overhead.

The DODAG formation thus starts at the DODAG root (initially, the only router which is part of a DODAG),
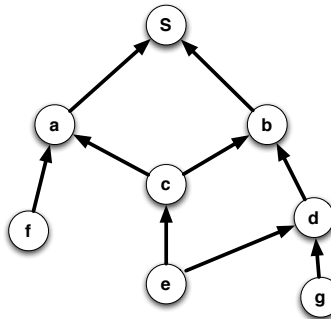

Figure 18: RPL Basic Construct: DODAGs

and spreads gradually to cover the whole LLN as DIOs are received, parents and preferred parents are selected and further routers participate in the DODAG. The DODAG root also includes, in DIO messages, a DODAG Configuration Object, describing common configuration attributes for all RPL routers in that network - including their mode of operation, timer characteristics etc. RPL routers in a DODAG include a verbatim copy of the last received DODAG Configuration Object in their DIO messages, permitting also such configuration parameters propagating through the network.

A Distance Vector protocol, RPL restricts the ability for a router to change rank. A router can freely assume a smaller rank than previously advertised (i.e. logically move closer to the root) if it discovers a parent advertising a lower rank, and must then disregard all previous parents of higher ranks. The ability for a router to assume a greater rank (i.e. logically move farther from the root) than previously advertised is restricted, to avoid count-to-infinity problems. The root can trigger "global recalculation" of the DODAG by increasing a sequence number, DODAG version, in DIO messages.

The DODAG so constructed is used for installing routes: the "preferred parent" of an RPL router can serve as a default route towards the root, or the root can embed in its DIO messages the destination prefixes, included by DIOs generated by RPL routers through the LLN, to which connectivity is provided by the root. Thus, RPL by way of DIO generation provides "upward routes" or "multipoint-to-point routes" from the sensors inside the LLN and towards the root.

"Downward routes", i.e., the routes from root to sensor nodes, are enabled by having sensors issue Destination Advertisement Object (DAO) messages, propagating as unicast via parents towards the DODAG root. These describe which prefixes belong to, and can be reached via, which RPL router. In a network, all RPL routers must operate in either of storing-mode or non-storing-mode, specified by way of a "Mode of Operation" (MOP) flag in the DODAG Configuration Object from the root. Depending on the MOP, DAO messages are forwarded differently towards the root:

- In *non-storing-mode*, an RPL router originates DAO messages, advertising one or more of its parents, and unicast it to the DODAG root. Once the root has received DAOs from an RPL router, and from all routers on the path between it and the root, it can use source routing for reaching advertised destinations inside the LLN.

- In *storing-mode*, each RPL router on the path between the originator of a DAO and the root records a route to the prefixes advertised in the DAO, as well as the next-hop towards these (the router, from which the DAO was received), then forwards the DAO to its preferred parent.

"Point-to-point routes", for communication between devices inside the LLN and where neither of the communicating devices are the DODAG root, are as default supported by having the source sensor transmit via its default route to the DODAG root (i.e., using the upward routes) which will then, depending on the "Mode of Operation" for the DODAG, either add a source-route to the received data for reaching the destination sensor (downward routes in non-storing-mode) or simply use hop-by-hop routing (downward routes in storing-mode). In the case of storing-mode, if the source and the destination for a point-to-point communication share a common ancestor other than the DODAG root, a downward route may be available (and used) before reaching the DODAG root. Both of these modes stretch the route by important factors, and lead to significantly longer paths compared to the shortest P2P paths available in the network [99]. To address this issue, an extension of RPL called RPL-P2P [56] is currently developed by the IETF. P2P-RPL defines a new mode of operation which provides RPL with a reactive approach to discover better paths on demand between an arbitrary source and destination, without having to go through the root or the first common ancestor of this source and destination.

While RPL has been specified as *Proposed Standard* in IETF, its applicability and performance in LLNs are not yet fully understood [31]. The following lists some limitations and concerns that have emerged concerning basic RPL mechanisms.

Requirement of DODAG Root   In RPL, the DODAG Root has both a special responsibility and is subject to special requirements. The DODAG Root is responsible for determining and maintaining the configuration parameters for the DODAG, and for initiating DIO emissions. The DODAG Root is also responsible (in both storing and non-storing mode) for being able to, when downward routes are supported, maintain sufficient topological information to be able to construct routes to all destinations in the network.

In a given deployment, selected RPL Routers can be provisioned with the required energy, memory and computational resources so as to serve as DODAG Roots, and be administratively configured as such - with the remainder of the RPL Routers in the network being of typically lesser capacity. In storing mode, the DODAG root needs to keep a routing entry for all RPL Routers in the RPL instance. In non-storing mode, the resource requirements on the DODAG Root are likely much higher than in storing mode, as the DODAG Root needs to store a network graph containing complete routes to all destinations in the RPL instance, in order to calculate the routing table (whereas in storing mode, only the next hop for each destination in the RPL instance needs to be stored, and aggregation may be used to further reduce the resource requirements).

Data Traffic Flows   RPL makes a-priori assumptions of data traffic types, and explicitly defines three such traffic types:

1. sensor-to-root data traffic (multipoint-to-point), which is predominant,

2. root-to-sensor data traffic (point-to-multipoint), which is rare, and

3. sensor-to-sensor (point-to-point) data traffic, which is extremely rare.

RPL is suited for networks where sensor-to-root traffic is dominant, by distribution of DIO messages and building of a collection tree. The one way traffic from the sensor to the root can be forwarded through the preferred parent.

However, the data traffic characteristics, assumed by RPL, do not represent a universal distribution of traffic types in LLNs. There are scenarios where sensor-to-sensor traffic is a more common occurrence, e.g., in Building Automation scenarios. In addition, there are scenarios, where all traffic is bi-directional. For example, the IETF protocol for use in constrained environments, CoAP [89, 40], makes use of acknowledgments to control packet loss and ensure that packets are received by the packet destination. In the four message types defined for CoAP: confirmable, acknowledgement, reset and non-confirmable, the first three are dedicated for sending/acknowledgement cycle.

The DAO Mechanisms: Downward and Point-to-Point Routes   In RPL, the "mode of operation" stipulates that either downward routes are not supported (MOP=0), or that they are supported by way of either storing or non-storing mode. In case downward routes are supported, RPL does not provide any mechanism for discriminating between which routes should or should not be maintained. In particular, in order to calculate routes to a given destination, all intermediaries between the DODAG Root and that destination must themselves be reachable effectively rendering downward routes in RPL an "all-or-none" situation.

The basic mechanisms in RPL force the choice between requiring all RPL Routers to have sufficient memory to store route entries for all destinations (storing mode) or suffer increased risk of fragmentation, and thus loss of data packets, while consuming network capacity by way of source routing through the DODAG Root (non-storing mode).

445

In addition, RPL does not explicitly specify how the DAO message are sent, which are used to build "downward" routes from root to sensors. This would make the different implementations unlikely to be interoperable.

6 Routing in Wired/Wireless Internetworks with OSPF

Protocols reviewed in section 5 have been specifically designed for spontaneous wireless networks. However, the increasing deployment of wireless technologies and the integration of different sorts of flexible networks with the Internet is leading to more complex inter-networks, neither purely wired networks nor purely spontaneous wireless networks, resulting from the interconnection of wireless mesh networks with fixed, wired networking infrastructure, inside Internet's Autonomous Systems. Figure 19 shows schematically a *compound Autonomous System*, in which fixed and wireless mesh networks are interconnected in the same routing domain.

In these scenarios, a classic IGP (in IP networks, typically OSPF) is used for routing in the fixed net-
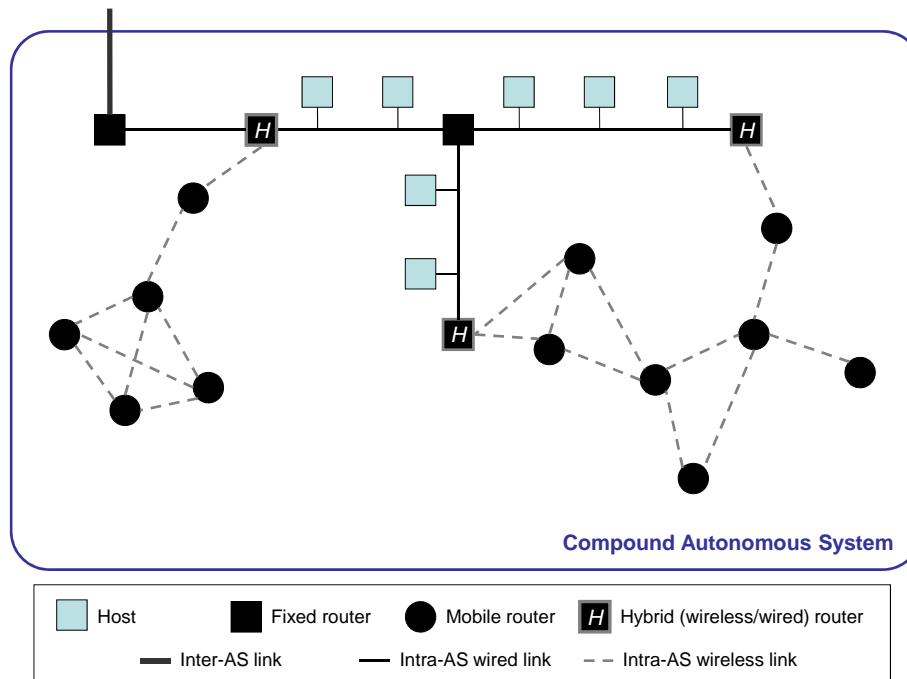

Figure 19: A compound Autonomous System.

work inside the AS. Rather than using an additional protocol for routing in the wireless mesh network inside the AS, it makes sense to explore approaches *extending* the protocol already used in the AS, so that it can take into consideration the issues described in section 3, run efficiently over wireless dynamic networks, handle the heterogeneity of the hybrid internetwork and thus perform routing over the whole compound AS.

Extension for hybrid internetworks of a protocol already in use can significantly reduce the transition costs (technical implementation, engineer training...), as only minor changes, or no changes at all, will be needed in the networks using the original protocol. It may be also benefial in terms of networking management complexity and routing performance, as a single (extended) routing protocol is more bearable than several protocols running in different parts of the internetwork. In the latter case, route distribution between different protocols operating at wireless and wired networks needs to be performed in specific *hybrid routers* (see Figure 19); this adds another layer of networking complexity and is likely to cause routing suboptimality. The advantages of extending a protocol in use come, however, at the expense of increasing the complexity and narrowing the space for optimization in the extended protocol, which needs to cope efficiently with a broader range of networking scenarios.

This section reviews the IETF extensions of the Open Shortest Path First (OSPF) protocol for MANETs. Section 6.1 shortly reviews the basics of OSPF, the main IGP for IP networks and a major representative of the link-state routing family, and indicates the reasons that prevent OSPF to be used "as-is" in wireless multi-hop ad hoc networks. Section 6.2 describes the main elements of the three extensions for MANETs standardized by the IETF: Multi-Point Relays (MPR-OSPF, specified in RFC 5449), MANET Designated Routers (OSPF-MDR, specified in RFC 5614) and Overlapping Relays (OR/SP, specified in RFC 5820).

6.1   Open Shortest Path First Protocol (OSPF)

OSPF [74, 41] is a link-state routing protocol for IP networks. Each router maintains a local **Link State Database** (LSDB), representing the full network topology. The protocol ensures that each router has the same LSDB and, thus, the exact same view of the network topology. Paths to every possible destination are derived from the **Shortest Path Tree** (SPT) that every router computes, by way of Dijkstra's algorithm [52].

Routers acquire information about their 2-hop (bi-directional) neighborhood and advertise their own presence and their 1-hop neighbors by periodically exchanging **Hello** messages with all their neighbors, in the way described in section 4.1.

Topology information is also disseminated through the network by way of **Link State Advertisements** (LSAs). Each such LSA lists mainly the current adjacencies of the router which generated the LSA. The local LSDB stored by a router contains the most recent LSAs received from every other router in the network.

Each router synchronizes its LSDB with a subset of its bidirectional neighbors. Synchronization between two neighboring routers is performed on a master-slave basis, by exchanging summaries of all LSAs in their LSDB, and then allowing each router to request retransmission of missing or locally outdated link-state advertisements. Links between a router and its synchronized neighbors are called **adjacencies**. The set of adjacencies is expected to form a network-wide connected backbone, connecting all routers in the network, in order to ensure paths can be computed correctly.

Finally, routers also acquire remote topology information by receiving LSAs. LSAs are flooded through the entire network in reliable fashion (explicit acknowledgements and retransmissions) via the backbone formed by adjacencies. Thus, any router which has formed adjacencies must advertise this periodically by way of constructing an LSA and performing LSA flooding.

$$\text{SPT links} \subset \text{Adjacent links} \subset \text{Bi-directional links} \tag{1}$$

Remote topology information is then used for the construction of the Shortest Path Tree: each router computes the shortest paths based on the information contained in the set of received LSAs.

This operation implies that OSPF exchanges control traffic and performs routing according to two principles:

1. Data traffic is routed to the corresponding destination through links contained in the Shortest Path Tree.

2. Data and control and traffic (LSAs and acknowledgements) is sent over adjacent (synchronized) links.

Interface Types Rules for flooding and adjacency handling vary for the different *interface types* supported by OSPF. Four main interface types are specified in RFC 2328 [74]:

- *Point-to-point interfaces* are those connected to point-to-point links. Such a link only permits communicating with a single (neighboring) interface.

- *Broadcast interfaces* participate in a broadcast link, in which any interface can directly communicate with any other interface. A classic example of broadcast link is Ethernet.

- *Non-Broadcast Multiple Access (NBMA) interfaces*, for non-broadcast networks (*i.e.*, networks supporting more than two routers, but without broadcast capability) in which each pair of interfaces can communicate directly. This interface type may be used with X.25 and ATM networks with Switched Virtual Circuits (SVC).

- *Point-to-multipoint interfaces*, for those non-broadcast networks in which direct communication between any pair of interface is not guaranteed. This may be the case, for instance, in Frame Relay networks using only Permanent Virtual Circuits (PVC), if not every pair of routers have a PVC between them.

OSPF only provides support for the two first interface types. In the NBMA and point-to-multipoint cases, OSPF *emulates* the behavior of a broadcast link and point-to-point links, repetively. For NBMA networks, LSA flooding and LSDB synchronization are handled by way of **Designated Routers** (DRs). A Designated Router (as well as a Backup Designated Router, BDR, expected to become DR in case of DR's failure) is elected from among routers whose interfaces are connected to the same link. DRs (and BDRs) form adjacencies with all the routers connected to the same link, and the Designated Router becomes responsible for flooding of LSAs, originated by routers on that link. A router point-to-multipoint link, in turn, is handled as a set of independent point-to-point links, one per neighboring router with which direct communication is available.

6.2 MANET Extensions: A Wireless Interface for OSPF

Standard interface types for non-broadcast networks (point-to-multipoint and NBMA) are not adapted for operation in a wireless multi-hop ad hoc network. As discussed in section 3.2, routers in a wireless multi-hop network may not agree on which routers are connected to a given link. This implies that the DR-based mechanisms of NBMA cannot be directly used in wireless multi-hop ad hoc networks. DR election may be inconsistent between different routers, causing flooding to disfunction and, possibly even preventing the protocol from converging. The use of the point-to-point interface, in turn, does not scale in these dynamic

networks: point-to-point emulation for every pair of interfaces directly reachable to each other causes an excessive control traffic overhead, even for relatively small networks, as shown experimentally in Henderson *et al.* [59]. This fact has led the research and industrial OSPF community to develop a new interface type to support the characteristics of wireless multi-hop ad hoc networks.

This new interface type needs to optimize the operation of (1) describing local topology in LSAs, (2) performing LSA flooding and (3) establishing and maintaining adjacencies in the context of wireless communication. Different approaches have been explored at the IETF, which have led to three different extensions of OSPF, consisting of three different interfaces for wireless multi-hop networks (or MANETs, in IETF's terminology).

Multi-Point Relays   MPR-OSPF [20] use Multi-Point Relays (MPR [83], see section 4.2) to optimize topology description, LSA flooding and LSDB synchronization. Nodes select MPRs from among their bidirectional neighbors in order to provide 2-hop coverage, and use them to disseminate their LSAs. A router becomes adjacent to both neighbors which it has selected as multi-point relays (MPRs) and neighbors which have selected the router as their multi-point relay (MPR selectors). Each router advertises in its LSAs its own MPRs and MPR selectors; consequently, the Shortest Path Tree is constructed over the set of adjacencies.

Overlapping Relays & Smart Peering   The Overlapping Relays / Smart Peering (OR/SP) extension of OSPF [86] floods LSAs via MPR as in MPR-OSPF, where the multi-point relays selected among the adjacent (synchronized) neighbors of the electing router. Adjacencies are selected following the Smart Peering (SP) rule, in which a neighbor becomes adjacent if it is not already reachable through the computing router's current Shortest Path Tree. The SP criterion reduces dramatically the number of synchronized links in the network. LSAs list adjacent neighbors, and may also list additional bidirectional neighbors (so-called *unsynchronized adjacencies*). The SPT is thus constructed over adjacencies and a subset of bidirectional neighbors.

MANET Designated Routers   OSPF-MDR [77] relies on two Connected Dominating Sets (CDS): the MANET Designated Routers (MDR) backbone and the Backup MDRs (BMDR) backbone. Both extend the standard OSPF (for NBMA networks) notions of "Designated Routers" and "Backup Designated Routers" to MANETs. This implies that routers behave differently depending on their role. MDRs are the only nodes allowed to flood LSAs. Every non-MDR router becomes adjacent at least to the closest MDR, and MDRs must become adjacent to other MDRs. LSAs list a configurable subset of links of the originator, which must at least include the adjacent neighbors. The SPT is thus constructed over adjacencies and a subset of bidirectional neighbors.

Compatibility with the OSPF routing philosophy detailed in section 6.1 varies significantly depending on the considered OSPF extension. MPR-OSPF is designed to preserve the two principles in OSPF routing: shortest, synchronized paths for data traffic and synchronized links for control traffic. Under the Overlapping Relays extension, data traffic paths are synchronized, but they are not necessarily optimal, as routers only synchronize a small fraction of their available links. Although providing several configuration parameters to tune the protocol's performance, the MANET Designated Routers (OSPF-MDR) also try to minimize the control traffic by reducing the number of synchronized links, even when this may lead to path suboptimality for data traffic.

Preserving OSPF routing principles Performed experiments suggest that extensions providing (theoretical) shortest paths for data traffic achieve a better performance than those neglecting shortest paths or allowing suboptimal routing in a wireless multi-hop network [19, 17]. Further analysis showed that preserving the second principle (all traffic is sent over synchronized links) in OSPF over mobile ad hoc networks, in the way that MPR-OSPF does, requires a significant amount of overhead due to the LSDB exchange between routers becoming *adjacent* (synchronized), and does not bring substantial benefit, due to short lifetime of several synchronized links in a wireless multi-hop dynamic network [45].

Why maintain LSDB synchronization in Extended OSPF ? LSDB synchronization between two routers proves useful in classic (wired) Internet internetworks, but is an expensive operation to perform in a dynamic (wireless multi-hop or mobile ad hoc) network. This is the reason why other link-state protocols such as OLSR, designed specifically for wireless mesh and mobile ad hoc networks, does not provide any mechanism for synchronizing the LSDBs of neighboring routers: topology information is only disseminated through the network by way of LSA flooding (see section 5.1). In the case of extended OSPF, there are two reasons for maintaining the notion of LSDB synchronization:

1. **OSPF backwards compatibility**. In standard OSPF [74, 41], the notion of *adjacency* is essential in the protocol's architecture and the router's operation, regardless of the specific types used for running OSPF in the router's interfaces.

2. **Routing in heterogeneous internetworks**. Unlike OLSR, extended OSPF is expected to run over hybrid internetworks (or compound Autonomous Systems, see Figure 19), that is, internetworks in which wired networks handled by standard OSPF interface types coexist and are interconnected with wireless multi-hop networks using the adapted wireless interface of (extended) OSPF. In these scenarios, in which some nodes (with wireless interfaces) are exposed to frequent disconnections from the network (meaning that their LSDBs may be no longer updated for a while) and others maintain stable links with their neighbors (those with wired interfaces), the fact that every router is expected to synchronize its LSDB with *at least* one of its neighbors provides an upper bound for the maximum time that a router A (in the wireless region of the internetwork) stays *disconnected* (that is, unaware of its local topology) from another router B (in the wired region of the internetwork) after missing an LSA flooded by router B. This becomes an issue as the time between consecutive LSA flooding processes from B is typically high – as wired links are stable and thus require less frequent updates about their state than wireless ones.

Further Extensions: adapted LSDB synchronization, MPR+SP and SLOT-OSPF In this context, some additional approaches can be explored beyond the three standardized extensions of OSPF. Clausen *et al.* [29], for example, propose a LSDB synchronization process based on the periodic broadcasting of *signatures* of the LSDB by every router to its neighborhood. These signatures allow neighbors of the originator to detect topology inconsistencies with its own LSDB, and request unicast retransmission of the corresponding LSAs. This turns the standard OSPF synchronization mechanism, based on a router-to-router LSDB exchange, to a router-to-neighborhood mechanism that takes advantage from the semibroadcast nature of communication in a wireless multi-hop network.

Without modifying the standard OSPF adjacency-forming process, LSDB synchronization can be kept, but the number of adjacencies per router should be reduced as much as possible, given the high cost of synchronization in terms of overhead and its small benefit in a dynamic network, with short-lived links. Data traffic should be sent over shortest paths (that is, optimal paths over the network, according to the available LSDB information and the metric in use), but these paths do not need to be synchronized. This

leads to combine in the same OSPF extension the mechanisms to provide shortest paths (MPR selection for topology description) and the mechanisms reducing the most the number of adjacencies to be established per router (*e.g.*, the Smart Peering rule used in the Overlapping Relays extension). The resulting extension, denominated MPR+SP, presents a better routing performance than extensions MPR-OSPF and OR/SP in which it is based, as shown in Cordero *et al.* [45]. Similarly, extension SLOT-OSPF [18] using the Relative Neighbor Graph (RNG [93]) for establishing adjacencies and MPR selection for computing shortest paths, also achieves better results in terms of delivery ratio and control traffic overhead than the standard extension to which it compares. In both cases (MPR+SP and SLOT-OSPF) shortest path computation, for which a comprehensive view of the network topology (with most of the links) is required, is splitted from the adjacency-forming criterion, which aims to reduce as much as possible the number of LSDB synchronizations to be performed. This split enables a further optimization of the protocol routing performance.

7 Conclusion: Integrating Spantaneous Wireless Networks in the IP Architecture

This chapter reviewed recent trends towards more collaborative network layer paradigms, accommodating spontaneous wireless networks. The thread followed throughout the chapter is the compatibility, in practice, with standard IP protocols currently at work in today's Internet. Indeed, absent such compatibility, slim are the chances that a given solution would actually be deployed and have a concrete impact. If one cannot just "reboot" the Internet to accommodate a convenient fresh start, one can nevertheless drive a continuous evolution of the Internet towards what is needed to allow seamless spontaneous wireless networking. In other words, research in this domain has to not only discover an alternative state in which things would work better, but also discover smooth transitions towards this alternative state, starting from the state we are currently in. The IETF is one of the important venues where such transitions are discussed, evaluated and designed. This chapter thus focused on standards developed by the IETF, which are relevant for spontaneous wireless networks.

In principle, spontaneous wireless networking is IP-disruptive: the way in which communication is performed in spontaneous wireless networks challenges some of the fundamental assumptions underlying traditional computer networking and the legacy IP networking architecture. The first part of this chapter has focused on identifying and discussing the impact of spontaneous wireless networking paradigms on layer 3, and has studied an alternative architectural model that could integrate spontaneous wireless networks in the IP networking architecture.

Due to their harsh characteristics, spontaneous wireless networks cannot be efficiently managed by standard protocols at layer 3 and above. In particular, legacy routing and flooding mechanisms are unsuitable to efficiently track low bandwidth, asymmetric, time-variant and lossy communication channels, between devices that may be mobile and thus create even more instability in the network topology. The second part of this chapter reviews various advanced techniques have been recently developed in order to accommodate these demanding characteristics: efficient flooding, non-trivial link metrics, neighborhood discovery, jittering techniques, duplicate detection mechanisms. These techniques are employed by several routing protocols developed by the IETF, mainly targeting Mobile Ad Hoc Networks (MANETs) and Low-Power Lossy Networks (LLNs), two categories of spontaneous wireless networks.

Taking a step back, it is perhaps worthy to observe that there are essentially four categories of solutions

to deal with IP-disruptive characteristics [15]:

Adaptation layer developments. This type of solution proposes to design intermediate layers, which interface between two of the legacy layers, *i.e.* from bottom up: (1) the physical layer, (2) the MAC layer, (3) the network layer, (4) the transport layer and (5) the application layer. Such approaches enable interoperability with legacy software by providing a black-box which emulates an appropriate behavior, compatible with upper layers, operating on top of disruptive lower layers. The system that results from such an approach is thus significantly more complex than the legacy system, in that it introduces a whole new "world" of protocols in addition to the legacy protocols. However, this approach can be effective in practice: a current example is 6LoWPAN [4], which designed a series of mechanisms at layer 2.5 (*i.e.* sitting between layer 2 and 3), enabling the operation of standard IP protocols at layers 3 and above on the IEEE 802.15.4 MAC layer.

Intra-layer optimizations. This type of solution proposes to modify or replace specific protocols currently in use within a legacy layer, to cope with IP-disruptive characteristics from lower layers. Most of the efforts that are mentioned in this chapter fall in this category. There are however limits to what one can achieve when taking this approach: it is unlikely that one can achieve game-changing innovation if one is allowed to replace only a single, small part of the whole system. Yet other types of solutions have thus been proposed, described in the following.

Cross-layer optimizations. This type of solution proposes to partially or totally abolish the distinction between two or more legacy layers, to produce a new system that performs significantly better, thanks to new protocols that can leverage cross-layer information to better cope with IP-disruptive lower layers. One example of such an approach is the XPRESS cross-layer stack [67], which collapses transport, network, and MAC layers and uses backpressure to provide better performance in wireless mesh networks. Cross-layer approaches are probably the most disruptive type of approaches, as their deployability and interoperability with standard legacy software is in general difficult to assess if at all possible – lack of interoperability is often the price to pay for radical performance improvements. There is however yet another type of solution proposing drastic changes while maintaining interoperability with legacy layers, as described below.

Top layer developments. This type of solution aims at building a radically new system sitting on top of the legacy protocol stack, at the application layer. Essentially, such an approach considers the Internet as a black box providing a service equivalent to a cable connecting source(s) and destination(s), and provides novel mechanisms efficiently using this cable to cope with IP-disruptive characteristics. One example of such construction is the experimental Delay Tolerant Networking (DTN) architecture developed by the Internet Research Task Force (IRTF) [7] [87] [84], including a specific routing protocol targeting DTNs [69].

It can be anticipated that innovative networking paradigms will continue to appear in the future, providing improvements at the price of IP-disruptive characteristics. However, in order to deploy or advance towards these new paradigms, one of the above approaches will have to be employed.

References

[1] 3GPP Telecommunications Standards Body. ONLINE: http://www.3gpp.org.

[2] Austrian Wireless Community Network. ONLINE: http://www.funkfeuer.at.

[3] Barcelona Wireless Community Network. ONLINE: http://www.guifi.net.

[4] IPv6 over Low power WPAN (6LoWPAN) IETF Working Group. ONLINE: https://datatracker.ietf.org/wg/6lowpan/.

[5] One Laptop Per Child. ONLINE: http://one.laptop.org.

[6] The Commotion Wireless Project. ONLINE: https://code.commotionwireless.net/projects/commotion.

[7] The Delay-Tolerant Networking IRTF Research Group (DTNRG),. ONLINE: http://irtf.org/dtnrg.

[8] The Freifunk Wireless Community Networks. ONLINE: http://www.freifunk.net.

[9] Customers Angered as iPhones Overload AT&T. The New York Times, September 2009.

[10] ITU-T G.9956: Narrow-Band OFDM power line communication transceivers - Data link layer specification, November 2011.

[11] The IEEE Standards Association, 802.11. ONLINE: http://standards.ieee.org/about/get/802/802.11.html.

[12] IEEE Standards Association, 802.15.4. ONLINE: http://www.ieee802.org/15/pub/TG4.html.

[13] The IEEE Standards Association, 802.16. ONLINE: http://ieee802.org/16/.

[14] ABRAMSON, N. The ALOHA System - Another alternative for computer communications. AFIPS 37, pp. 281-285.

[15] BACCELLI, E. IP-Disruptive Wireless Networking: Integration in the Internet. Habilitation Thesis, Université Pierre et Marie Curie Sorbonne.

[16] BACCELLI, E., CLAUSEN, T. H., HERBERG, U., AND PERKINS, C. E. Ip links in multihop ad hoc wireless networks ? In *Proceedings of the 17th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)* (Croatia, September 2009), p. 5.

[17] BACCELLI, E., CORDERO, J. A., AND JACQUET, P. Multi-hop relaying techniques with ospf on ad hoc networks. In *Proceedings of the 4th IEEE International Conference on Sensor Networks and Communications (ICSNC)* (Porto, Portugal, September 2009).

[18] BACCELLI, E., CORDERO, J. A., AND JACQUET, P. Optimization of Critical Data Synchronization via Link Overlay RNG in Mobile Ad Hoc Networks. Proceedings of the 7th IEEE International Conference Mobile Ad-hoc and Sensor Systems (MASS), San Francisco, CA, USA.

[19] BACCELLI, E., CORDERO, J. A., AND JACQUET, P. Ospf over multi-hop ad hoc wireless communications. *International Journal of Computer Networks and Communications (IJCNC), vol. 2, num. 5* (September 2010).

[20] BACCELLI, E., JACQUET, P., NGUYEN, D., AND CLAUSEN, T. OSPF Multipoint Relay (MPR) Extension for Ad Hoc Networks. RFC 5449 (Experimental), Feb. 2009.

[21] BACCELLI, E., AND PERKINS, C. Multi-hop Ad Hoc Wireless Communication. Internet Draft, draft-baccelli-manet-multihop-communication-02, work in progress, July 2013.

[22] BACCELLI, E., AND TOWNSLEY, M. IP Addressing Model in Ad Hoc Networks. RFC 5889 (Informational), Sept. 2010.

[23] BAKER, F. Requirements for IP Version 4 Routers. RFC 1812 (Proposed Standard), June 1995.

[24] BELLMAN, R. On a routing problem. *Quarterly of Applied Mathematics, no. 16, vol. 1, pp. 87-90* (1958).

[25] BORMANN, C., ERSUE, M., AND KERANEN, A. Terminology for Constrained Node Networks. Internet Draft, draft-ietf-lwig-terminology-04, work in progress, April 2013.

[26] CALLON, R. Use of OSI IS-IS for routing in TCP/IP and dual environments. RFC 1195 (Proposed Standard), Dec. 1990.

[27] CHAKERES, I., MACKER, J., AND CLAUSEN, T. Mobile Ad hoc Network Architecture, November 2007. Internet Draft, draft-ietf-autoconf-manetarch-07, work in progress.

[28] CLAUSEN, T. A MANET Architectural Model. Inria Research Report n. 6145, January 2007.

[29] CLAUSEN, T., BACCELLI, E., AND JACQUET, P. Ospf-style database exchange and reliable synchronization in the optimized link-state routing protocol. Proceedings of the 1st IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON'2004), Santa Clara (CA), USA.

[30] CLAUSEN, T., CAMACHO, A., YI, J., DE VERDIERE, A. C., IGARASHI, Y., SATOH, H., AND MORII, Y. Experience with the loadng routing protocol for llns, December 2012. Internet Draft, work in progress, draft-lavenu-lln-loadng-interoperability-report-04.

[31] CLAUSEN, T., COLIN DE VERDIERE, A., YI, J., HERBERG, U., AND IGARASHI, Y. Observations of RPL: IPv6 Routing Protocol for Low power and Lossy Networks, February 2013. Internet Draft, work in progress, draft-clausen-lln-rpl-experiences-06, work in progress.

[32] CLAUSEN, T., DE VERDIERE, A. C., YI, J., NIKTASH, A., IGARASHI, Y., SATOH, H., AND HERBERG, U. The lln on-demand ad hoc distance-vector routing protocol - next generation, July 2013. Internet Draft, work in progress, draft-clausen-lln-loadng-04.

[33] CLAUSEN, T., AND DEARLOVE, C. Representing Multi-Value Time in Mobile Ad Hoc Networks (MANETs). RFC 5497 (Proposed Standard), Mar. 2009.

[34] CLAUSEN, T., DEARLOVE, C., AND ADAMSON, B. Jitter Considerations in Mobile Ad Hoc Networks (MANETs). RFC 5148 (Informational), Feb. 2008.

[35] CLAUSEN, T., DEARLOVE, C., AND DEAN, J. Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP). RFC 6130 (Proposed Standard), Apr. 2011.

[36] CLAUSEN, T., DEARLOVE, C., DEAN, J., AND ADJIH, C. Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format. RFC 5444 (Proposed Standard), Feb. 2009.

[37] CLAUSEN, T., DEARLOVE, C., AND JACQUET, P. The Optimized Link State Routing Protocol version 2. Internet Draft, draft-ietf-manet-olsrv2-19, work in progress, March 2013.

[38] CLAUSEN, T., HERBERG, U., AND PHILIPP, M. A critical evaluation of the "ipv6 routing protocol for low power and lossy networks". Proceedings of the 5th IEEE International Conference on Wireless & Mobile Computing, Networking & Communication (WiMob).

[39] CLAUSEN, T., AND JACQUET, P. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), Oct. 2003.

[40] COLITTI, W., STEENHAUT, K., AND DE CARO, N. Integrating wireless sensor networks with the web. Proceedings of the Proceedings of the IEEE Workshop on Internet of Things Technology and Architectures.

[41] COLTUN, R., FERGUSON, D., MOY, J., AND LINDEM, A. OSPF for IPv6. RFC 5340 (Proposed Standard), July 2008.

[42] COMER, D. E. *Internetworking with TCP/IP (Vol. 1): Principles, protocols and architecture, 4th Edition*. Prentice Hall, 2000.

[43] CORDERO, J. A. Mpr-based pruning techniques for shortest path tree computation. In *Proceedings of the 18th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)* (Split, Croatia, September 2010), p. 5.

[44] CORDERO, J. A. A probabilistic study of the delay caused by jittering in wireless flooding. *Wireless Personal Communications* (2013), 1–25.

[45] CORDERO, J. A., CLAUSEN, T., AND BACCELLI, E. Mpr+sp: Towards a unified mpr-based manet extension for ospf. Proceedings of the 44th Annual Hawaii International Conference on System Sciences (HICSS'2002), Garden Island (Hawaii), USA.

[46] CORDERO, J. A., JACQUET, P., AND BACCELLI, E. Impact of jitter-based techniques on flooding over wireless ad hoc networks: Model and analysis. Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM 2012), Orlando, USA.

[47] CORDERO, J. A., YI, J., AND CLAUSEN, T. Optimization of Jitter Configuration for Reactive Route Discovery in Wireless Mesh Networks. Proceedings of the 11th International Symposium on Modeling and Optimization in Mobile, Ad hoc and Wireless Networks (WiOpt 2013), Tsukuba City of Science, Japan (to appear).

[48] CORSON, S., AND MACKER, J. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. RFC 2501 (Informational), Jan. 1999.

[49] COUTO, D. S. J. D., AGUAYO, D., CHAMBERS, B. A., AND MORRIS, R. Performance of multihop wireless networks: Shortest path is not enough. *ACM SIGCOMM Computer Communication Review, vol. 33, issue 1, pp. 83-88* (January 2003).

[50] DE COUTO, D. S. J., AGUAYO, D., BICKET, J., AND MORRIS, R. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom)* (San Diego, California, USA, Sep 2003), p. 8.

[51] DEARLOVE, C., CLAUSEN, T., AND JACQUET, P. Link Metrics for the Mobile Ad Hoc Network (MANET) Routing Protocol OLSRv2 - Rationale, April 2013. Internet Draft, draft-ietf-manet-olsrv2-metrics-rationale-03, work in progress.

[52] DIJKSTRA, E. A note in two problems in connection with graphs. *Numerische Mathematik, no. 1* (1959).

[53] FORD, L. R. J. Network flow theory. *RAND Corporation, paper P-923* (1956).

[54] FRIEDMAN, R., HAY, D., AND KLIOT, G. Jittering Broadcast Transmissions in MANETs: Quantification and Implementation Strategies. Tech. rep., Department of Computer Science, Technion, 2009.

[55] GERLA, M., TANG, K., AND BAGRODIA, R. Tcp performance in wireless multi-hop networks. *Proceedings of the IEEE WMCSA Conference, pp. 41-50* (1999).

[56] GOYAL, M., BACCELLI, E., PHILIPP, M., BRANT, A., AND MARTOCCI, J. Reactive discovery of point-to-point routes in low power and lossy networks, March 2013. IETF Internet Draft, draft-ietf-roll-p2p-rpl-17.

[57] HAWKINSON, J., AND BATES, T. Guidelines for creation, selection, and registration of an Autonomous System (AS). RFC 1930 (Best Current Practice), Mar. 1996.

[58] HEDRICK, C. Routing Information Protocol. RFC 1058 (Historic), June 1988.

[59] HENDERSON, T. R., SPAGNOLO, P., AND KIM, J. K. A wireless interface type for ospf. Proceedings of the Military Communications Conference (MILCOM'03), Seattle, WA, USA.

[60] HERBERG, U., AND CLAUSEN, T. A comparative performance study of the routing protocols load and rpl with bi-directional traffic in low-power and lossy networks (lln). Proceedings of the 8th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN).

[61] HIERTZ, G., MAX, S., ZHAO, R., DENTENEER, D., AND BERLEMANN, L. Principles of ieee 802.11s. In *Proceedings of WiMAN in conjunction with the 16th ICCCN* (Honolulu, Hawaii, USA, Aug 2007), p. 6.

[62] JOHNSON, D., HU, Y., AND MALTZ, D. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728 (Experimental), Feb. 2007.

[63] KARP, B., AND KUNG, H. T. Gpsr: Gredy perimeter stateless routing for wireless networks. In *Proceedings of the MobiCom'2000* (Boston, MA, USA, August 2000).

[64] KIM, K., PARK, S. D., MONTENEGRO, G., YOO, S., AND KUSHALNAGAR, N. 6LoWPAN Ad Hoc On-Demand Distance Vector Routing, June 2007. Internet Draft, work in progress, draft-daniel-6lowpan-load-adhoc-routing-03.

[65] KOHNO, M. Perspective on Future Internet Routing. Asia Future Internet Hong Kong Workshop, February 2011.

[66] KOTZ, D., NEWPORT, C., AND ELLIOTT, C. The Mistaken Axioms of Wireless-Network Research. Technical Report TR2003-467, July 2003.

[67] LAUFER, R., SALONIDIS, T., LUNDGREN, H., AND LEGUYADEC, P. XPRESS: A Cross-layer Backpressure Architecture for Wireless Multi-hop Networks. *ACM MobiCom* (September 2011).

[68] LEVIS, P., CLAUSEN, T., HUI, J., GNAWALI, O., AND KO, J. The Trickle Algorithm. RFC 6206 (Proposed Standard), Mar. 2011.

[69] LINDGREN, A., DORIA, A., DAVIES, E., AND GRASIC, S. Probabilistic Routing Protocol for Intermittently Connected Networks. RFC 6693 (Experimental), Aug. 2012.

[70] MACKER, J. Simplified Multicast Forwarding. RFC 6621 (Experimental), May 2012.

[71] MALKIN, G. RIP Version 2. RFC 2453 (INTERNET STANDARD), Nov. 1998.

[72] MALKIN, G., AND MINNEAR, R. RIPng for IPv6. RFC 2080 (Proposed Standard), Jan. 1997.

[73] MONTENEGRO, G., KUSHALNAGAR, N., HUI, J., AND CULLER, D. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard), Sept. 2007.

[74] MOY, J. OSPF Version 2. RFC 2328 (INTERNET STANDARD), Apr. 1998.

[75] NARTEN, T., NORDMARK, E., SIMPSON, W., AND SOLIMAN, H. Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (Draft Standard), Sept. 2007.

[76] NI, S.-Y., TSENG, Y.-C., CHEN, Y.-S., AND SHEU, J.-P. The broadcast storm problem in a mobile ad hoc network. Proceedings of ACM MobiCom'99, Seattle, USA.

[77] OGIER, R., AND SPAGNOLO, P. Mobile Ad Hoc Network (MANET) Extension of OSPF Using Connected Dominating Set (CDS) Flooding. RFC 5614 (Experimental), Aug. 2009.

[78] ORAN, D. OSI IS-IS Intra-domain Routing Protocol. RFC 1142 (Informational), Feb. 1990.

[79] PARK, J. C., AND KASERA, S. K. Expected data rate: An accurate high-throughput path metric for multi-hop wireless routing. In *Proceedings of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2005)* (Santa Clara, CA, USA, September 2005).

[80] PERKINS, C., BELDING-ROYER, E., AND DAS, S. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[81] PERKINS, C., RATLIFF, S., AND DOWDELL, J. Dynamic manet on-demand (aodvv2) routing, March 2013. IETF Internet Draft, draft-ietf-manet-aodvv2-00.

[82] PERLMAN, R. *Interconnections (2nd Edition): bridges, routers, switches, and internetworking protocols*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[83] QAYYUM, A., VIENNOT, L., AND LAOUITI, A. Multipoint relaying for flooding broadcast messages in mobile wireless networks. Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'2002), Big Island (Hawaii), USA.

[84] RAMADAS, M., BURLEIGH, S., AND FARRELL, S. Licklider Transmission Protocol - Specification. RFC 5326 (Experimental), Sept. 2008.

[85] REKHTER, Y., AND LI, T. A Border Gateway Protocol 4 (BGP-4). RFC 1771 (Draft Standard), Mar. 1995.

[86] ROY, A., AND CHANDRA, M. Extensions to OSPF to Support Mobile Ad Hoc Networking. RFC 5820 (Experimental), Mar. 2010.

[87] SCOTT, K., AND BURLEIGH, S. Bundle Protocol Specification. RFC 5050 (Experimental), Nov. 2007.

[88] SHELBY, Z., CHAKRABARTI, S., NORDMARK, E., AND BORMANN, C. Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). RFC 6775 (Proposed Standard), Nov. 2012.

[89] SHELBY, Z., HARTKE, K., AND BORMANN, C. Constrained Application Protocol (CoAP). Internet Draft, draft-ietf-core-coap-14, work in progress, March 2013.

[90] STANICA, R., CHAPUT, E., AND BEYLOT, A.-L. Broadcast communication in vehicular ad-hoc network safety applications. In *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC 2011)* (Las Vegas, NV, USA, January 2011).

[91] TANENBAUM, A. S., AND WETHERALL, D. J. *Computer Networks, 5th Edition*. Prentice Hall, 2011.

[92] The Internet Engineering Task Force (IETF). ONLINE: http://www.ietf.org.

[93] TOUSSAINT, G. T. The relative neighborhood graph of a finite planar set. *Pattern Recognition, vol. 12, no. 4* (1980), 261–268.

[94] TSE, D., AND VISWANATH, P. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[95] VASSEUR, J. Terminology in Low power And Lossy Networks. Internet Draft, draft-ietf-roll-terminology-12, work in progress, March 2013.

[96] VASSEUR, J., AGARWAL, N., HUI, J., SHELBY, Z., BERTRAND, P., AND CHAUVENET, C. RPL: The IP routing protocol designed for low power and lossy networks. IPSO Working Paper n. 7, April 2011.

[97] WANG, Z., AND CROWCROFT, J. Bandwidth-delay based routing algorithms. Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'95).

[98] WINTER, T., THUBERT, P., BRANDT, A., HUI, J., KELSEY, R., LEVIS, P., PISTER, K., STRUIK, R., VASSEUR, J., AND ALEXANDER, R. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard), Mar. 2012.

[99] XIE, W., GOYAL, M., HOSSEINI, H., MARTOCCI, J., BASHIR, Y., BACCELLI, E., AND DURRESI, A. A performance analysis of point-to-point routing along a directed acyclic graph in low power and lossy networks. In *Network-Based Information Systems (NBiS), 2010 13th International Conference on* (2010), pp. 111–116.

[100] YI, J., CORDERO, J. A., AND CLAUSEN, T. Jitter Considerations in On-Demand Route Discovery for Mobile Ad Hoc Networks. Proceedings of the 16th International Conference on Network-Based Information Systems (NBiS'2013), Gwanju, South Korea (to appear).