1.1 What is the Internet?

In this book we use the public Internet, a specific computer network (and one which probably most readers have used), as our principle vehicle for discussing computer networking protocols. But what is the Internet? We would like to give you a one-sentence definition of the Internet, a definition that you can take home and share with your family and friends. Alas, the Internet is very complex, both in terms of its hardware and software components, as well as the services it provides.

A Nuts and Bolts Description

Instead of giving a one-sentence definition, let's try a more descriptive approach. There are a couple of ways to do this. One way is to describe the nuts and bolts of the Internet, that is, the basic hardware and software components that make up the Internet. Another way is to describe the Internet in terms of a networking infrastructure that provides services to distributed applications. Let's begin with the nuts-and-bolts description, using Figure 1.1-1 to illustrate our discussion.

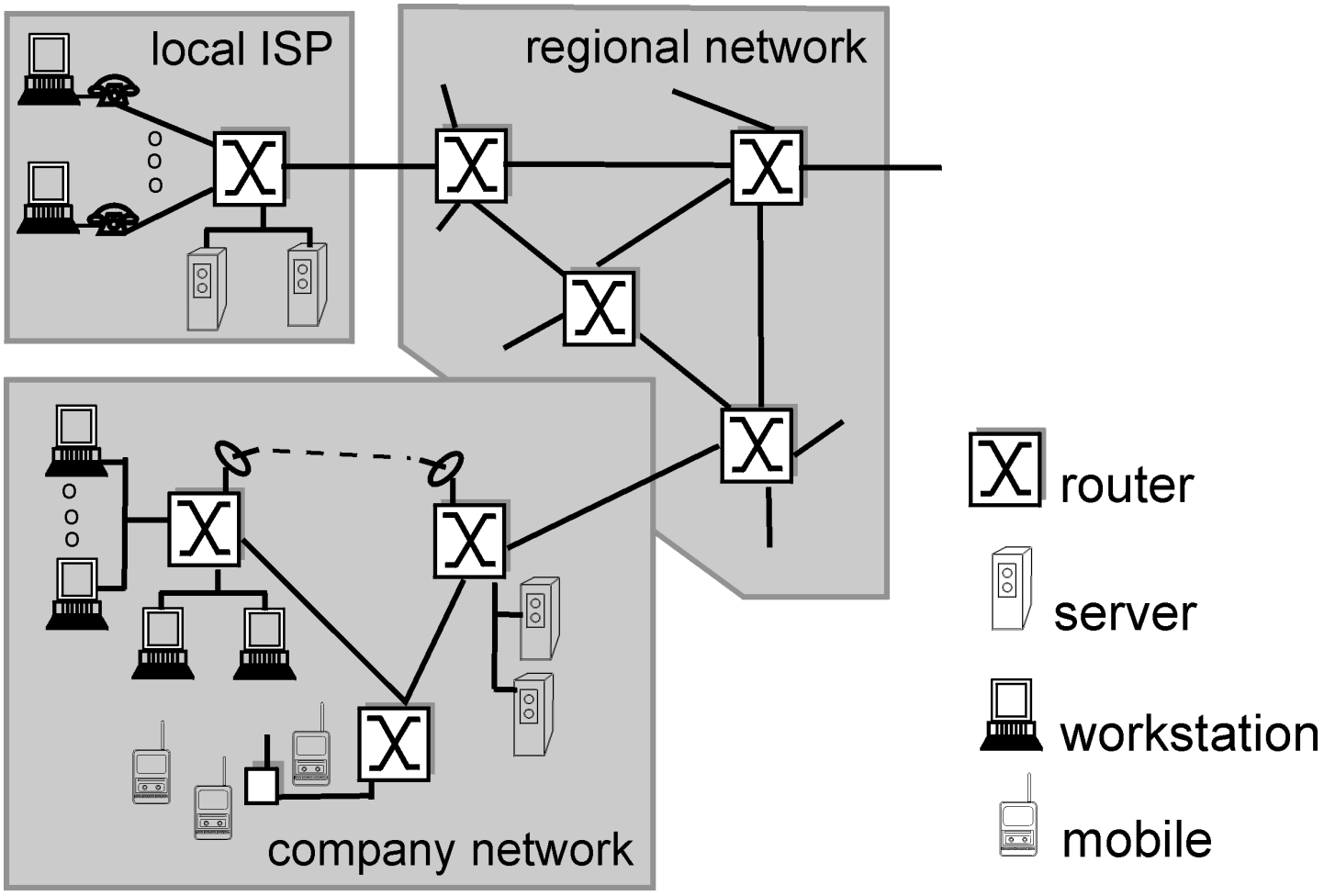

Figure 1.1-1: Some "pieces" of the Internet

- The public Internet is a world-wide **computer network**, i.e., a network that interconnects millions of computing devices throughout the world. Most of these computing devices are traditional desktop PCs, Unix-based workstations, and so called "servers" that store and transmit information such as WWW pages and e-mail messages. Increasingly, non-traditional

computing devices such as Web TVs, mobile computers, pagers and toasters are being connected to the Internet. (Toasters are not the only rather unusual devices to have been hooked up to the Internet; see the The Future of the Living Room.) In the Internet jargon, all of these devices are called **hosts** or **end systems**. The Internet applications with which many of us are familiar, such as the WWW and e-mail, are **network application programs** that run on such end systems. We will look into Internet end systems in more detail in section 1.3 and then delve deeply into the study of network applications in Chapter 2.

- End systems, as well as most other "pieces" of the Internet, run **protocols** that control the sending and receiving of information within the Internet. **TCP** (the Transmission Control Protocol) and **IP** (the Internet Protocol) are two of the most important protocols in the Internet. The Internet's principle protocols are collectively known as **TCP/IP protocols**. We begin looking into protocols in section 1.2. But that's just a start --much of this entire book is concerned with computer network protocols!

- End systems are connected together by **communication links**. We'll see in section 1.5 that there are many types of communication links. Links are made up of different types of **physical media:** coaxial cable, copper wire, fiber optics, and radio spectrum. Different links can transmit data at different rates. The link transmission rate is often called the **link bandwidth,** and is typically measured in bits/second.

- Usually, end systems are not directly attached to each other via a single communication link. Instead, they are indirectly connected to each other through intermediate switching devices known as **routers.** A router takes information arriving on one of its incoming communication links and then forwards that information on one of its outgoing communication links. The **IP protocol** specifies the format of the information that is sent and received among routers and end systems. The path that transmitted information takes from the sending end system, through a series of communications links and routers, to the receiving end system is known as a **route** or **path** through the network. We introduce routing in more detail in section 1.4, and study the algorithms used to determine routes, as well as the internal structure of a router itself, in Chapter 4.

- Rather than provide a *dedicated* path between communicating end systems, the Internet uses a technique known as **packet switching** that allows multiple communicating end systems to share a path, or parts of a path, at the same time. We will see that packet switching can often use a link more "efficiently" than circuit switching (where each pair of communicating end systems gets a dedicated path). The earliest ancestors of the Internet were the first packet-switched networks; today's public Internet is the *grande dame* of all existing packet-switched networks.

- The Internet is really a **network of networks**. That is, the Internet is an interconnected set of privately and publicly owned and managed networks. Any network connected to the Internet must run the IP protocol and conform to certain naming and addressing conventions. Other than these few constraints, however, a network operator can configure and run its network (i. e., its little "piece" of the Internet) however it chooses. Because of the universal use of the IP protocol in the Internet, the IP protocol is sometimes referred to as the **Internet dail tone**.

- The topology of the Internet, i.e., the structure of the interconnection among the various pieces of the Internet, is **loosely hierarchical**. Roughly speaking, from bottom-to-top, the hierarchy consists of end systems connected to local **Internet Service Providers (ISPs)** though **access networks.** An access network may be a so-called local area network within a company or university, a dial telephone line with a modem, or a high-speed cable-based or phone-based access network. Local ISP's are in turn connected to regional ISPs, which are in turn connected to national and international ISPs. The national and international ISPs are connected together at the highest tier in the hierarchy. New tiers and branches (i.e., new networks, and new networks of networks) can be added just as a new piece of Lego can be attached to an existing Lego construction. In the first half of 1996, approximately 40,000 *new* network addresses were added to the Internet [Network 1996] - an astounding growth rate.

- At the technical and developmental level, the Internet is made possible through creation, testing and implementation of **Internet Standards**. These standards are developed by the Internet Engineering Task Force (IETF). The IETF standards documents are called **RFCs** (request for comments). RFCs started out as general request for comments (hence the name) to resolve architecture problems which faced the precursor to the Internet. RFCs, though not formally standards, have evolved to the point where they are cited as such. RFCs tend to be quite technical and detailed. They define protocols such as TCP,

IP, HTTP (for the Web) and SMTP (for open-standards e-mail). There are more than 2000 different RFC's

The public Internet (i.e., the global network of networks discussed above) is the network that one typically refers to as *the* Internet. There are also many private networks, such as certain corporate and government networks, whose hosts are not accessible from (i.e., they can not exchange messages with) hosts outside of that private network. These private networks are often referred to as **intranets**, as they often use the same "internet technology" (e.g., the same types of host, routers, links, protocols, and standards) as the public Internet.

A Service Description

The discussion above has identified many of the pieces that make up the Internet. Let's now leave the nuts and bolts description and take a more abstract, service-oriented, view:

- The Internet allows **distributed applications** running on its end systems to exchange data with each other. These applications include remote login, file transfer, electronic mail, audio and video streaming, real-time audio and video conferencing, distributed games, the World Wide Web, and much much more [AT&T 1998]. It is worth emphasizing that the Web is not a separate network but rather just one of many distributed applications that use the communication services provided by the Internet. The Web *could* also run over a network besides the Internet. One reason that the Internet is the communication medium of choice for the Web, however, is that no other existing packet-switched network connects more than 43 million [Network 1999] computers together and has 100 million or so users [Almanac]. (By the way, determining the number of computers hooked up to the Internet is a very difficult task, as no one is responsible for maintaining a list of who's connected. When a new network is added to the Internet, its administrators do not need to report which end systems are connected to that network. Similarly, an exiting network does not report its changes in connected end systems to any central authority.)

- The Internet provides two services to its distributed applications: a **connection-oriented service** and a **connectionless service**. Loosely speaking, connection-oriented service guarantees that data transmitted from a sender to a receiver will eventually be delivered to the receiver in-order and in its entirety. Connectionless service does not make any guarantees about eventual delivery. Typically, a distributed application makes use of one or the other of these two services and not both. We examine these two different services in section 1..3 and in great detail in Chapter 3.

- Currently the Internet does not provide a service that makes promises about *how long* it will take to deliver the data from sender to receiver. And except for increasing your access bit rate to your Internet Service Provider (ISP), you currently cannot obtain better service (e.g., shorter delays) by paying more -- a state of affairs that some (particularly Americans!) find odd. We'll take a look at state-of-the art Internet research that is aimed at changing this situation in Chapter 6.

Our second description of the Internet - in terms of the services it provides to distributed applications -- is a non-traditional, but important, one. Increasingly, advances in the "nuts and bolts" components of the Internet are being driven by the needs of new applications. So it's important to keep in mind that the Internet is an *infrastructure* in which new applications are being constantly invented and deployed.

We have given two descriptions of the Internet, one in terms of the hardware and software components that make up the Internet, the other in terms of the services it provides to distributed applications. But perhaps you are even more confused as to what the Internet is. What is packet switching, TCP/IP and connection-oriented service? What are routers? What kinds of communication links are present in the Internet? What is a distributed application? What does the Internet have to do with children's toys? If you feel a bit overwhelmed by all of this now, don't worry - the purpose of this book is to introduce you to both the nuts and bolts of the Internet, as well as the principles that govern how and why it works. We will explain these important terms and questions in the subsequent sections and chapters.

Some Good Hyperlinks

As every Internet researcher knows, some of the best and most accurate information about the Internet and its protocols is not in hard copy books, journals, or magazines. The best stuff about the Internet is in the Internet itself! Of course, there's really too much material to sift through, and sometimes the gems are few and far between. Below, we list a few generally excellent WWW sites for network- and Internet-related material. Throughout the book, we will also present links to relevant, high quality URL's that provide background, original (i.e., a citation), or advanced material related to the particular topic under study. Here is a set of key links that you will want to consult while you proceed through this book:

Internet Engineering Task Force (IETF): The IETF is an open international community concerned with the development and operation of the Internet and its architecture. The IETF was formally established by the Internet Architecture Board (IAB) in 1986. The IETF meets three times a year; much of its ongoing work is conducted via mailing lists by working groups. Typically, based upon previous IETF proceedings, working groups will convene at meetings of the IETF to discuss the work of the IETF working groups. The IETF is administered by the Internet Society, whose WWW site contains lots of high-quality, Internet-related material.

The World Wide Web Consortium (W3C): The W3C was founded in 1994 to develop common protocols for the evolution of the World Wide Web. This an outstanding site with fascinating information on emerging Web technologies, protocols and standards.

The Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE): These are the two main international professional societies that have technical conferences, magazines, and journals in the networking area. The ACM Special Interest Group in Data Communications (SIGCOMM), the IEEE Communications Society, and the IEEE Computer Society are the groups within these bodies whose efforts are most closely related to networking.

Connected: An Internet Encyclopedia: An attempt to take the Internet tradition of open, free protocol specifications, merge it with a 1990s Web presentation, and produce a readable and useful reference to the technical operation of the Internet. The site contains material on over 100 Internet topics.

Data communications tutorials from the online magazine Data Communications: One of the better magazines for data communications technology. The site includes many excellent tutorials.

Media History Project: You may be wondering how the Internet got started. Or you may wonder how electrical communications got started in the first place. And you may even wonder about what preceded electrical communications! Fortunately, the Web contains an abundance of excellent resources available on these subjects. This site promotes the study of media history from petroglyths to pixels. It covers the history of digital media, mass media, electrical media, print media, and even oral and scribal culture.

References

[Almanac 1998] Computer Industry Almanac, December 1998, http://www.c-i-a.com/
[AT&T 1998] "Killer Apps," AT&T WWW page http://www.att.com/attlabs/brainspin/networks/killerapps.html
[Network 1996] Network Wizards, Internet Domain Survey, July 1996, http://www.nw.com/zone/WWW-9607/report.html
[Network 1999] Network Wizards, Internet Domain Survey, January 1999, http://www.nw.com/zone/WWW/top.html

Return to Table Of Contents

1.2.What is a Protocol?

Now that we've got a bit of a feel for what the "Internet" is, let's consider another important word is the title of this book: "protocol." What *is* a protocol? What does a protocol *do*? How would you recognize a protocol if you met one?

A Human Analogy

It is probably easiest to understand the notion of a computer network protocol by first considering some human analogies, since we humans execute protocols all of the time. Consider what you do when you want to ask someone for the time of day. A typical exchange is shown in Figure 1.2-1. Human protocol (or good manners, at least) dictates that one first offers a greeting (the first "Hi" in Figure 1.2-1) to initiate communication with someone else. The typical response to a "Hi" message (at least outside of New York City) is a returned "Hi" message. Implicitly, one then takes a cordial "Hi" response as an indication that one can proceed ahead and ask for the time of day. A different response to the initial "Hi" (such as "Don't bother me!", or "I don't speak English," or an unprintable reply that one might receive in New York City) might indicate an unwillingness or inability to communicate. In this case, the human protocol would be to not ask for the time of day. Sometimes one gets no reponse at all to a question, in which case one typically gives up asking that person for the time. Note that in our human protocol, *there are specific messages we send, and specific actions we take in response to the received reply messages or other events (such as no reply within some given amount of time)*. Clearly, transmitted and received messages, and actions taken when these message are sent or received or other events occur, play a central role in a human protocol. If people run different protocols (e.g., if one person has manners but the other does not, or if one understands the concept of time and the other does not) the protocols do not interoperate and no useful work can be accomplished. The same is true in networking -- it takes two (or more) communicating entities running the same protocol in order to accomplish a task.

Let's consider a second human analogy. Suppose you're in a college class (a computer networking class, for example!). The teacher is droning on about protocols and you're confused. The teacher stops to ask, "Are there any questions?" (a message that is transmitted to, and received by, all students who are not sleeping). You raise your hand (transmitting an implicit message to the teacher). Your teacher acknowledges you with a smile, saying "Yes ....." (a transmitted message encouraging you to ask your question - teachers *love* to be asked questions) and you then ask your question (i.e., transmit your message to your teacher). Your teacher hears your question (receives your question message) and answers (transmits a reply to you). Once again, we see that the transmission and receipt of messages, and a set of conventional actions taken when these mesages are sent and received, are at the heart of this question-and-answer protocol.

Network Protocols

A network protocol is similar to a human protocol, except that the entities exchanging messages and taking actions are hardware or software components of a computer network, components that we will study shortly in the following sections. All activity in the Internet that involves two or more communicating remote entities is governed by a protocol. Protocols in routers determine a packet's path from source to destination; hardware-implemented protocols in the network interface cards of two physically connected computers control the flow of bits on the "wire" between the two computers; a congestion control protocol controls the rate at which packets are transmitted between sender and receiver. Protocols are running everywhere in the Internet, and consequently much of this book is about computer network protocols.

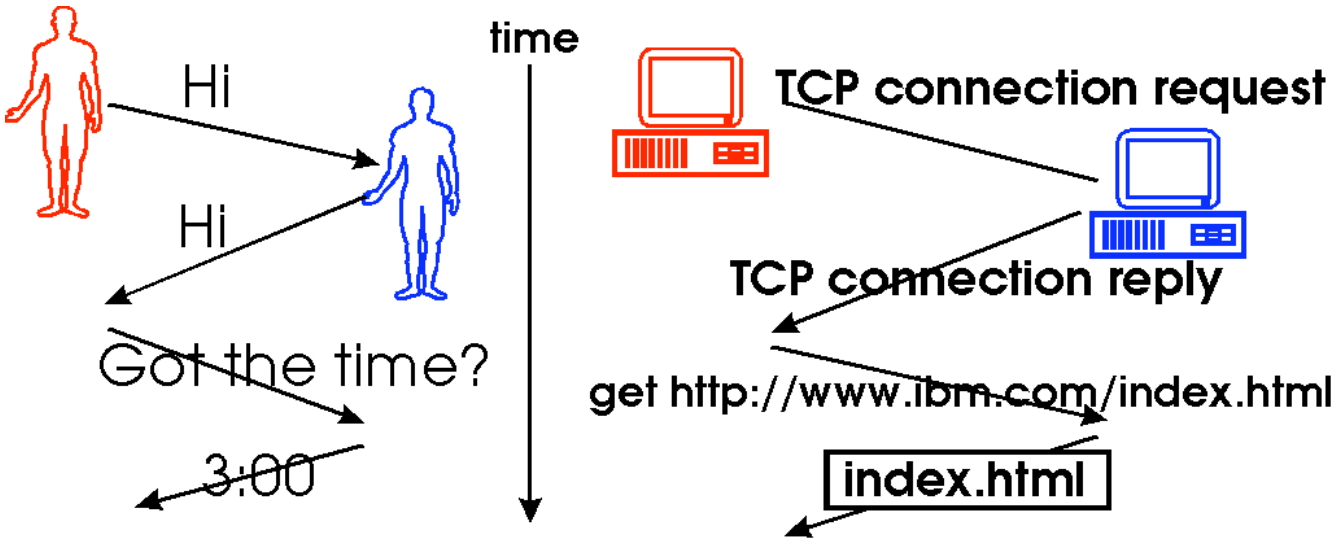

Figure 1.2-1: A human protocol and a computer network protocol

As an example of a computer network protocol with which you are probably familiar, consider what happens when you make a request to a WWW server, i.e., when you type in the URL of a WWW page into your web browser. The scenario is illustrated in the right half of Figure 1.2-1. First, your computer will send a so-called "connection request" message to the WWW server and wait for a reply. The WWW server will eventually receive your connection request message and return a "connection reply" message. Knowing that it is now OK to request the WWW document, your computer then sends the name of the WWW page it wants to fetch from that WWW server in a "get" message. Finally, the WWW server returns the contents of the WWW document to your computer.

Given the human and networking examples above, the exchange of messages and the actions taken when these messages are sent and received are the key defining elements of a protocol:

*A **protocol** defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message.*

The Internet, and computer networks in general, make extensive use of protocols. Different protocols are used to accomplish different communication tasks. As you read through this book, you will learn that some protocols are simple and straightforward, while others are complex and intellectually deep. Mastering the field of computer networking is equivalent to understanding the what, why and how of networking protocols.

Return to Table Of Contents

1.3 The Network Edge

In the previous sections we presented a high-level description of the Internet and networking protocols. We are now going to delve a bit more deeply into the components of the Internet. We begin in this section at the edge of network and look at the components with which we are most familiar -- the computers (e.g., PCs and workstations) that we use on a daily basis. In the next section we will move from the network edge to the network core and examine switching and routing in computer networks. Then in Section 1.5 we will discuss the actual physical links that carry the signals sent between the computers and the switches.

1.3.1 End Systems, Clients and Servers

In computer networking jargon, the computers that we use on a daily basis are often referred to as or **hosts** or **end systems**. They are referred to as "hosts" because they host (run) application-level programs such as a Web browser or server program, or an e-mail program. They are also referred to as "end systems" because they sit at the "edge" of the Internet, as shown in Figure 1.3-1. Throughout this book we will use the terms hosts and end systems interchangeably, that is, host = end system.

Hosts are sometimes further divided into two categories: **clients** and **servers**. Informally, clients often tend to be desktop PC's or workstations, while servers are more powerful machines. But there is a more precise meaning of a client and a server in computer networking. In the so-called **client-server model**, a client program running on one end system requests and receives information from a server running on another end system. This client-server model is undoubtedly the most prevalent structure for Internet applications. We will study the client-server model in detail in Chapter 2. The Web, e-mail, file transfer, remote login (e.g., Telnet), newgroups and many other popular applications adopt the client-server model. Since a client typically runs on one computer and the server runs on another computer, client-server Internet applications are, by definition, **distributed applications.** The client and the server interact with each other by communicating (i.e., sending each other messages) over the Internet. At this level of abstraction, the routers, links and other "pieces" of the Internet serve as a "black box" that transfers messages between the distributed, communicating components of an Internet application. This is the level of abstraction depicted in Figure 1.3-1.

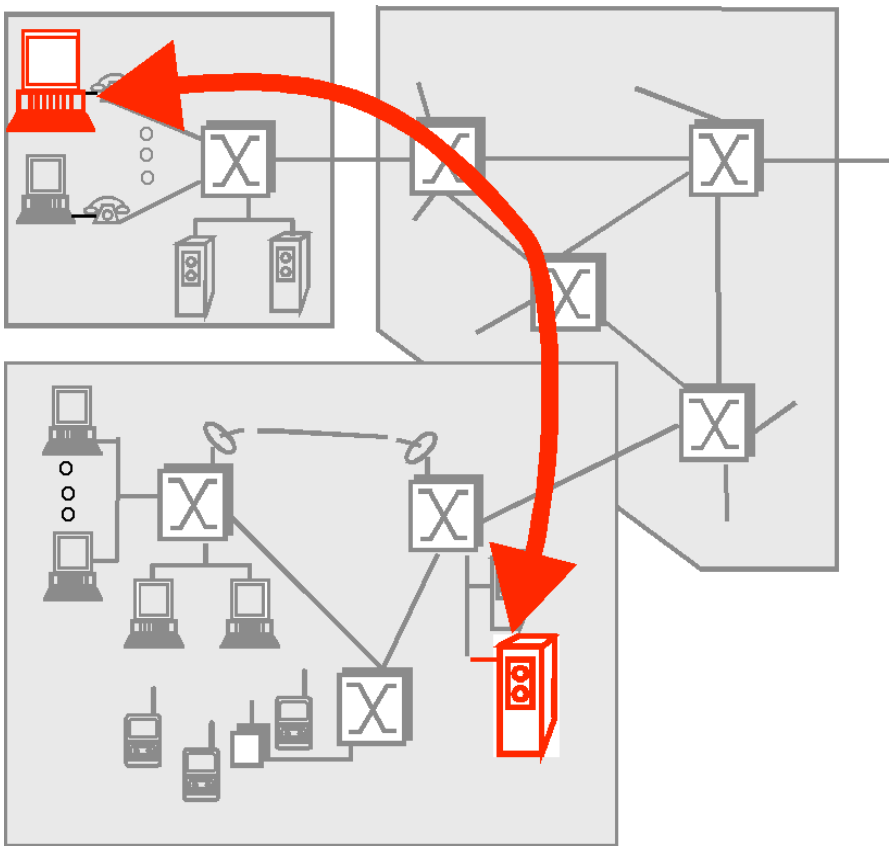

Figure 1.3-1: End system Interaction

Computers (e.g., a PC or a workstation), operating as clients and servers, are the most prevalent type of end system. However, an increasing number of alternative devices, such as so-called network computers and thin clients [Thinworld 1998], Web TV's and set top boxes [Mills 1998], digital cameras, and other devices are being attached to the Internet as end systems. An interesting discussion of the continuing evolution of Internet applications is [AT&T 1998].

1.3.2 Connectionless and Connection-Oriented Services

We have seen that end systems exchange messages with each other according to an application-level protocol in order to accomplish some task. The links, routers and other pieces of the Internet provide the means to transport these messages between the end system applications. But what are the characteristics of this communication service that is provided? The Internet, and more generally TCP/IP networks, provide two types of services to its applications: **connectionless service** and **connection-oriented service**. A developer creating an Internet application (e.g., an email application, a file transfer application, a Web application or an Internet phone application) must program the application to use one of these two services. Here, we only briefly describe these two services; we shall discuss them in much more detail in Chapter 3, which covers transport layer protocols.

Connection-Oriented Service

When an application uses the connection-oriented service, the client and the server (residing in different end systems) send control packets to each other before sending packets with real data (such as e-mail messages). This so-called handshaking procedure alerts the client and server, allowing them to prepare for an onslaught of packets. It is interesting to note that this initial hand-shaking procedure is similar to the protocol used in human interaction. The exchange of "hi's" we saw in Figure 1.2-1 is an example of a human "handshaking protocol" (even though handshaking is not literally taking place between the two people). The two TCP messages that are exchanged as part of the WWW interaction shown in Figure 1.2-1 are two of the three messages exchanged when TCP sets up a connection between a sender and receiver. The third TCP message (not shown) that forms the final part of the TCP three-way handshake (see Section 3.7) is contained in the `get` message shown in Figure 1.2-1.

Once the handshaking procedure is finished, a "connection" is said to be established between the two end systems. But the two end systems are connected in a very loose manner, hence the terminology "connection-oriented". In particular, only the end systems themselves are aware of this connection; the packet switches (i.e., routers) within the Internet are completely oblivious to the connection. This is because a TCP connection is nothing more than allocated resources (buffers) and state variables in the end systems. The packet switches do not maintain any connection state information.

The Internet's connection oriented service comes bundled with several other services, including reliable data transfer, flow control and congestion control. By **reliable data transfer**, we mean that an application can rely on the connection to deliver all of its data without error and in the proper order. Reliability in the Internet is achieved through the use of *acknowledgments* and *retransmissions*. To get a preliminary idea about how the Internet implements the reliable transport service, consider an application that has established a connection between end systems A and B. When end system B receives a packet from A, it sends an acknowledgment; when end system A receives the acknowledgment, it knows that the corresponding packet has definitely been received. When end system A doesn't receive an acknowledgment, it assumes that the packet it sent was not received by B; it therefore retransmits the packet.**Flow control** makes sure that neither side of a connection overwhelms the other side by sending too many packets too fast. Indeed, the application at one one side of the connection may not be able to process information as quickly as it receives the information. Therefore, there is a risk of overwhelming either side of an application. The flow-control service forces the sending end system to reduce its rate whenever there is such a risk. We shall see in Chapter 3 that the Internet implements the flow control service by using sender and receiver buffers in the communicating end systems. The Internet's **congestion control** service helps prevent the Internet from entering a state of grid lock. When a router becomes congested, its buffers can overflow and packet loss can occur. In such circumstances, if every pair of communicating end systems continues to pump packets into the network as fast as they can, gridlock sets in and few packets are delivered to their destinations. The Internet avoids this problem by forcing end systems to diminish the rate at which they send packets into the network during periods of congestion. End systems are alerted to the existence of severe congestion when they stop receiving acknowledgments for the packets they have sent.

We emphasize here that although the Internet's connection-oriented service comes bundled with reliable data transfer, flow control and congestion control, these three features are by no means essential components of a connection-oriented service. A different type of computer network may provide a connection-oriented service to its applications without bundling in one or more of these features. Indeed, any protocol that performs handshaking between the communicating entities before transferring data is a connection-orieinted service [Iren].

The Internet's connection-oriented service has a name -- **TCP** (Transmission Control Protocol); the initial version of the TCP protocol is defined in the Internet Request for Comments RFC 793 [RFC 793]. The *services* that TCP provides to an application include reliable transport, flow control and congestion control. It is important to note that an application need only care about the services that are provided; it need not to worry about *how* TCP actually implements reliability, flow control, or congestion control. *We*, of course, are *very* interested in how TCP implements these services and we shall cover these topics in detail in Chapter 3.

Connectionless Service

There is no handshaking with the Internet's connectionless service. When one side of an application wants to send packets to another side of an application, the sending application simply sends the packets. Since there is no handshaking procedure prior to the transmission of the packets, data can be delivered faster. But there are no acknowledgments either, so a source never knows for sure which packets arrive at the destination. Moreover, the service makes no provision for flow control or congestion control. The Internet's connectionless service is provided by **UDP** (User Datagram Protocol); UDP is defined in the Internet Request for Comments RFC 768 [RFC 768].

Most of the more familiar Internet applications use TCP, the Internet's connection-oriented service. These applications include Telnet (remote login), SMTP (for electronic mail), FTP (for file transfer), and HTTP (for the Web). Nevertheless, UDP, the Internet's connectionless service, is used by many applications, including many of the emerging multimedia applications, such as Internet phone, audio-on-demand, and video conferencing.

References

[AT&T 1998] "Killer Apps," AT&T WWW page http://www.att.com/attlabs/brainspin/networks/killerapps.html

[Iren] S.Iren, P.Amer, P.Conrad, "The Transport Layer: Tutorial and Survey," *ACM Computing Surveys,* June 1999

[Thinworld 1998] Thinworld homepage, http://www.thinworld.com/

[Mills 1998] S. Mills, "TV set-tops set to take off ", CNET News.com, Oct. 1998

[RFC 768] J. Postel, " Datagram Protocol," RFC 768, Aug. 1980.

[RFC 793] J. Postel, "Transmission Control Protocol," RFC 793, September 1981.

Return to Table of Contents

1.4 The Network Core

Having examined the endsystems and end-end transport service model of the Internet in section 1.3, let us now delve more deeply into the "inside" of the network. In this section we study the network core -- the mesh of routers that interconnect the Internet's endsystems. Figure 1.4-1 highlights the network core in red.

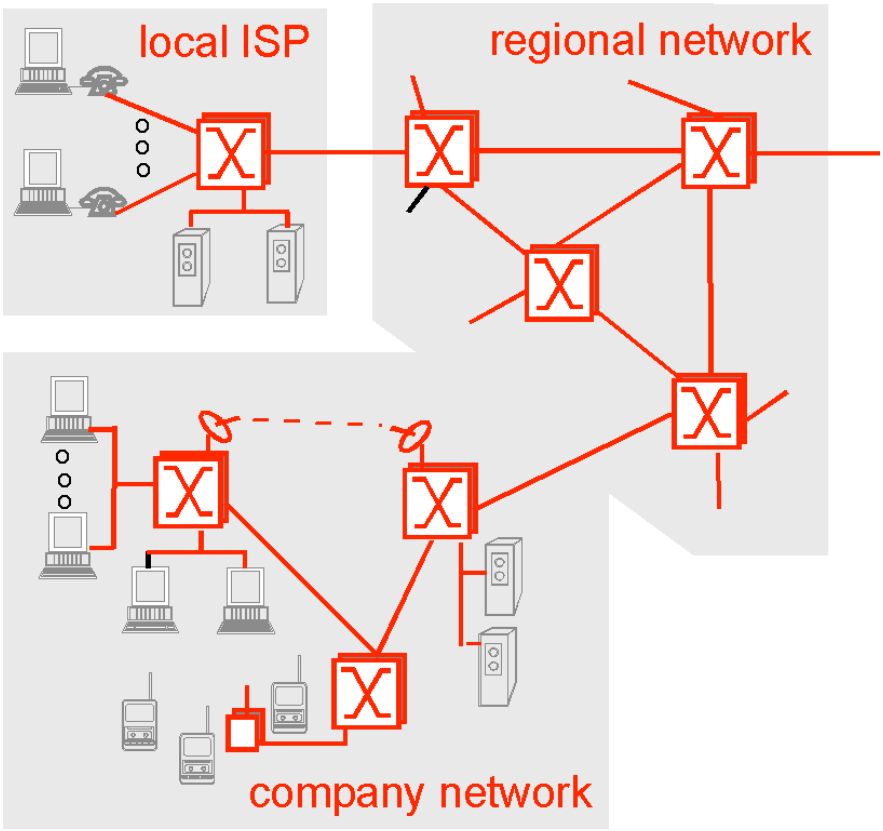

Figure 1.4-1: The network core

1.4.1 Circuit Switching, Packet Switching and Message Switching

There are two fundamental approaches towards building a network core: **circuit switching** and **packet switching**. In circuit-switched networks, the resources needed along a path (buffers, link bandwidth) to provide for communication between the endsystems are *reserved* for the duration of the session. In packet-switched networks, these resources are *not* reserved; a session's messages use the resource on demand, and as a consequence, may have to wait (i.e., queue) for access to a communication link. As a simple analogy, consider two restaurants -- one which requires reservations and another which neither requires reservations nor accepts them. For the restaurant that requires reservations, we have to go

through the hassle of first calling (or sending an e-mail!) before we leave home. But when we arrive at the restaurant we can, in principle, immediately communicate with the waiter and order our meal. For the restaurant that does not require reservations, we don't need to bother to reserve a table. But when we arrive at the restaurant, we may have to wait for a table before we can communicate with the waiter.

The ubiquitous telephone networks are examples of circuit-switched networks. Consider what happens when one person wants to send information (voice or facsimile) to another over a telephone network. Before the sender can send the information, the network must first establish a connection between the sender and the receiver. In contrast with the TCP connection that we discussed in the previous section, this is a bona fide connection for which the switches on the path between the sender and receiver maintain connection state for that connection. In the jargon of telephony, this connection is called a **circuit.** When the network establishes the circuit, it also reserves a constant transmission rate in the network's links for the duration of the connection. This reservation allows the sender to transfer the data to the receiver at the *guaranteed* constant rate.

Today's Internet is a quintessential packet-switched network. Consider what happens when one host wants to send a packet to another host over a packet-switched network. As with circuit-switching, the packet is transmitted over a series of communication links. But with packet-switching, the packet is sent into the network without reserving any bandwidth whatsoever. If one of the links is congested because other packets need to be transmitted over the link at the same time, then our packet will have to wait in a buffer at the sending side of the transmission line, and suffer a delay. The Internet makes its *best effort* to deliver the data in a timely manner. But it does not make any guarantees.

Not all telecommunication networks can be neatly classified as pure circuit-switched networks or pure packet-switched networks. For example, for networks based on the ATM technology, a connection can make a reservation and yet its messages may still wait for congested resources! Nevertheless, this fundamental classification into packet- and circuit-switched networks is an excellent starting point in understanding telecommunication network technology.

Circuit Switching

This book is about computer networks, the Internet and packet switching, not about telephone networks and circuit switching. Nevertheless, it is important to understand why the Internet and other computer networks use packet switching rather than the more traditional circuit-switching technology used in the telephone networks. For this reason, we now give a brief overview of circuit switching.

Figure 1.4-2 illustrates a circuit-switched network. In this network the three circuit switches are interconnected by two links; each of these links has *n* circuits, so that each link can support *n* simultaneous connections. The endsystems (e.g., PCs and workstations) are each directly connected to one of the switches. (Ordinary telephones are also connected to the switches, but they are not shown in the diagram.) Notice that some of the hosts have analog access to the switches, whereas others have direct digital access. For analog access, a modem is required. When two hosts desire to communicate,

the network establishes a dedicated *end-to-end circuit* between two hosts. (Conference calls between more than two devices are, of course, also possible. But to keep things simple, let's suppose for now that there are only two hosts for each connection.) Thus in order for host A to send messages to host B, the network must first reserve one circuit on each of two links.

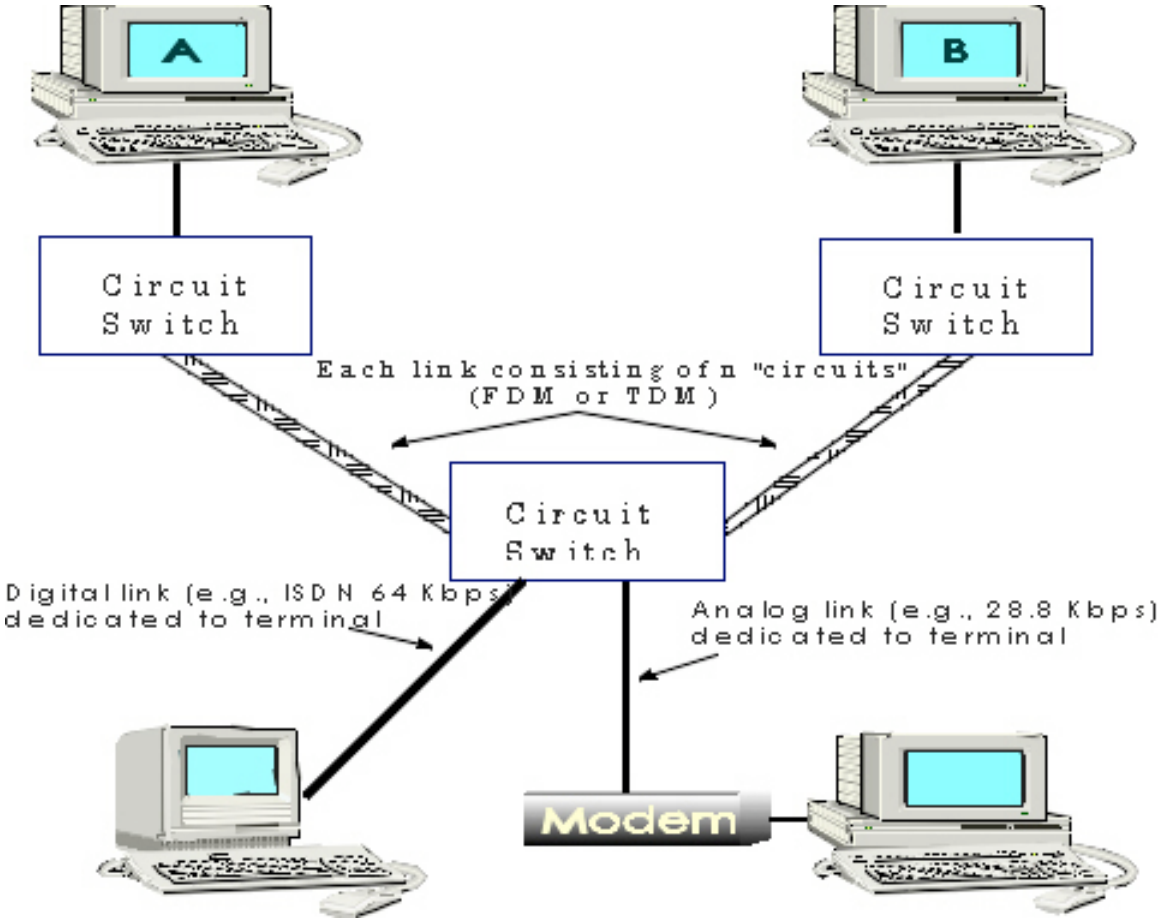

Figure 1.4-2: A simple circuit-switched network consisting of three circuit switches interconnected with two links. Each link has n circuits; each end-to-end circuit over a link gets the fraction 1/n of the link's bandwidth for the duration of the circuit. The *n* circuits in a link can be either TDM or FDM circuits.

A circuit in a link is implemented with either **frequency division multiplexing (FDM)** or **time-division multiplexing (TDM).** With FDM, the frequency spectrum of a link is shared among the connections established across the link. Specifically, the link dedicates a frequency band to each connection for the duration of the connection. In telephone networks, this frequency band typically has a width of 4 kHz. The width of the band is called, not surprisingly, the **bandwidth**. FM radio stations also use FDM to share microwave frequency spectrum.

The trend in modern telephony is to replace FDM with TDM. The majority of the links in most telephone systems in the United States and in other developed countries currently employ TDM. For a TDM link, time is divided into frames of fixed duration and each frame is divided into a fixed number of time slots. When the network establish a connection across a link, the network dedicates one time slot in every frame to the connection. These slots are dedicated for the sole use of that connection, with a time slot available for use (in every frame) to transmit the connection's data.

Figure 1.4.3 illustrates FDM and TDM for a specific network link. For FDM, the frequency domain is segmented into a number of circuits, each of bandwidth 4 KHz (i.e., 4,000 Hertz or 4,000 cycles per second). For TDM, the time domain is segmented into four circuits; each circuit is assigned the same dedicated slot in the revolving TDM frames. The transmission rate of the frame is equal to the frame rate multiplied by the number of bits in a slot. For example, if the link transmits 8,000 frames per second and each slot consists of 8 bits, then the transmission rate is 64 Kbps.

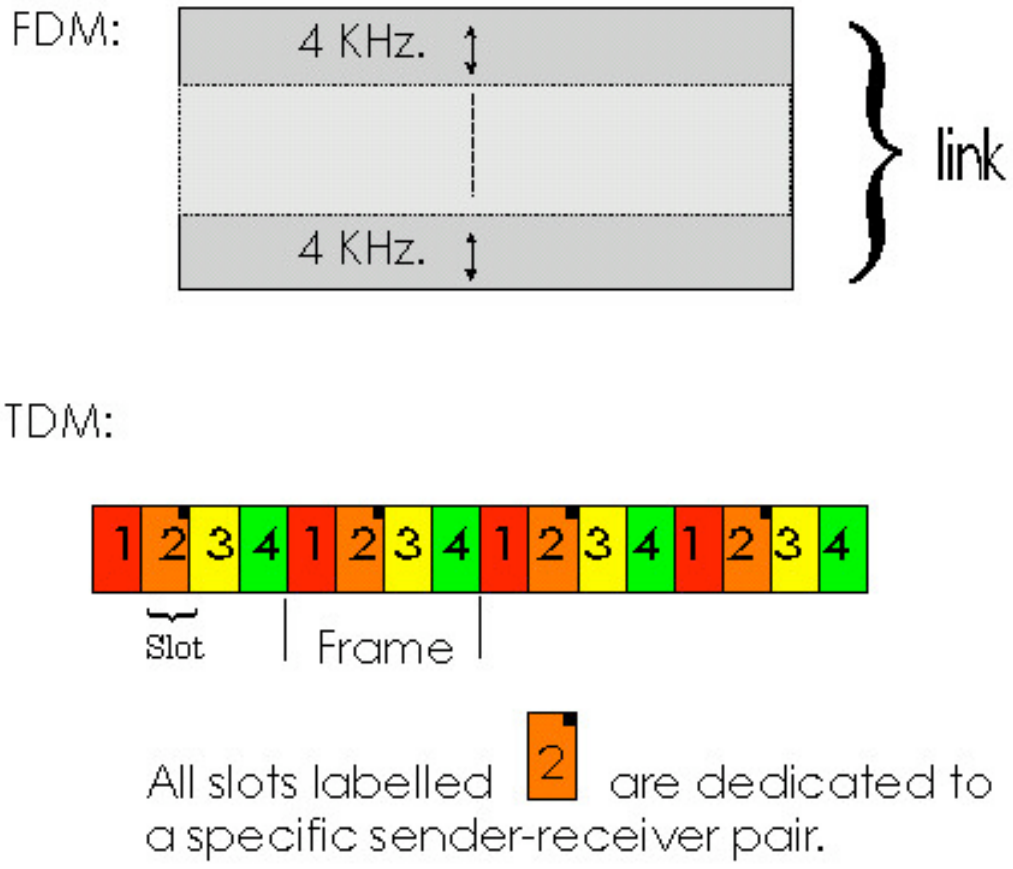

Figure 1.4-3: With FDM, each circuit continuously gets a fraction of the bandwidth. With TDM, each circuit gets all of the bandwidth periodically during brief intervals of time (i.e., during slots).

Proponents of packet switching have always argued that circuit switching is wasteful because the

dedicated circuits are idle during **silent periods**. For example, when one of the conversants in a telephone call stops talking, the idle network resources (frequency bands or slots in the links along the connection's route) cannot be used by other ongoing connections. As another example of how these resources can be underutilized, consider a radiologist who uses a circuit-switched network to remotely access a series of x-rays. The radiologist sets up a connection, requests an image, contemplates the image, and then requests a new image. Network resources are wasted during the radiologist's contemplation periods. Proponents of packet switching also enjoy pointing out that establishing end-to-end circuits and reserving end-to-end bandwidth is complicated and requires complex signaling software to coordinate the operation of the switches along the end-to-end path.

Before we finish our discussion of circuit switching, let's work through a numerical example that should shed further insight on the matter. Let us consider how long it takes to send a file of 640 Kbits from host A to host B over a circuit-switched network. Suppose that all links in the network use TDM with 24 slots and have bit rate 1.536 Mbps. Also suppose that it takes 500 msec to establish an end-to-end circuit before A can begin to transmit the file. How long does it take to send the file? Each circuit has a transmission rate of (1.536 Mbps)/24 = 64 Kbps, so it takes (640 Kbits)/(64 Kbps) = 10 seconds to transmit the file. To this 10 seconds we add the the circuit establishment time, giving 10.5 seconds to send the file. Note that the transmission time is independent of the number links: the transmission time would be 10 seconds if the end-to-end circuit passes through one link or one-hundred links. AT&T Labs provides an interactive site [AT&T 1998] to explore transmission delay for various file types and transmission technologies.

Packet Switching

We saw in sections 1.2 and 1.3. that application-level protocols exchange **messages** in accomplishing their task. Messages can contain anything the protocol designer desires. Messages may perform a control function (e.g., the "hi" messages in our handshaking example) or can contain data, such as an ASCII file, a Postscript file, a Web page, a digital audio file. In modern packet-switched networks, the source breaks long messages into smaller **packets**. Between source and destination, each of these packets traverse communication links and **packet switches** (also known as **routers)**. Packets are transmitted over each communication link at a rate equal to the *full* transmission rate of the link. Most packet switches use **store and forward transmission** at the inputs to the links. Store-and-forward transmission means that the switch must receive the entire packet before it can begin to transmit the first bit of the packet onto the outbound link. Thus store-and-forward packet-switches introduce a **store-and-forward delay** at the input to each link along the packet's route. This delay is proportional to the packet's length in bits. In particular, if a packet consists of L bits, and the packet is to be forwarded onto an outbound link of R bps, then the store-and-forward delay at the switch is L/R seconds.

Within each router there are multiple buffers (also called queues), with each link having an **input buffer** (to store packets that have just arrived to that link) and an **output buffer**. The output buffers play a key role in packet switching. If an arriving packet needs to be transmitted across a link but finds the link busy with the transmission of another packet, the arriving packet must wait in the output buffer. Thus, in

addition to the store-and-forward delays, packets suffer output buffer **queueing delays**. These delays are variable and depend on the level of congestion in the network. Since the amount of buffer space is finite, an arriving packet may find that the buffer is completely filled with other packets waiting for transmission. In this case, **packet loss** will occur - either the arriving packet or one of the already-queued packets will be dropped. Returning to our restaurant analogy from earlier in this section, the queueing delay is analogous to the amount of time one spends waiting for a table. Packet loss is analogous to being told by the waiter that you must leave the premises because there are already too many other people waiting at the bar for a table.

Figure 1.4-4 illustrates a simple packet-switched network. Suppose Hosts A and B are sending packets to Host E. Hosts A and B first send their packets along 28.8 Kbps links to the first packet switch. The packet switch directs these packets to the 1.544 Mbps link. If there is congestion at this link, the packets queue in the link's output buffer before they can be transmitted onto the link. Consider now how Host A and Host B packets are transmitted onto this link. As shown in Figure 1.4-4, the sequence of A and B packets does not follow any periodic ordering; the ordering is random or statistical -- packets are sent whenever they happen to be present at the link. For this reason, we often say that packet switching employs **statistical multiplexing**. Statistical multiplexing sharply contrasts with time-division multiplexing (TDM), for which each host gets the same slot in a revolving TDM frame.

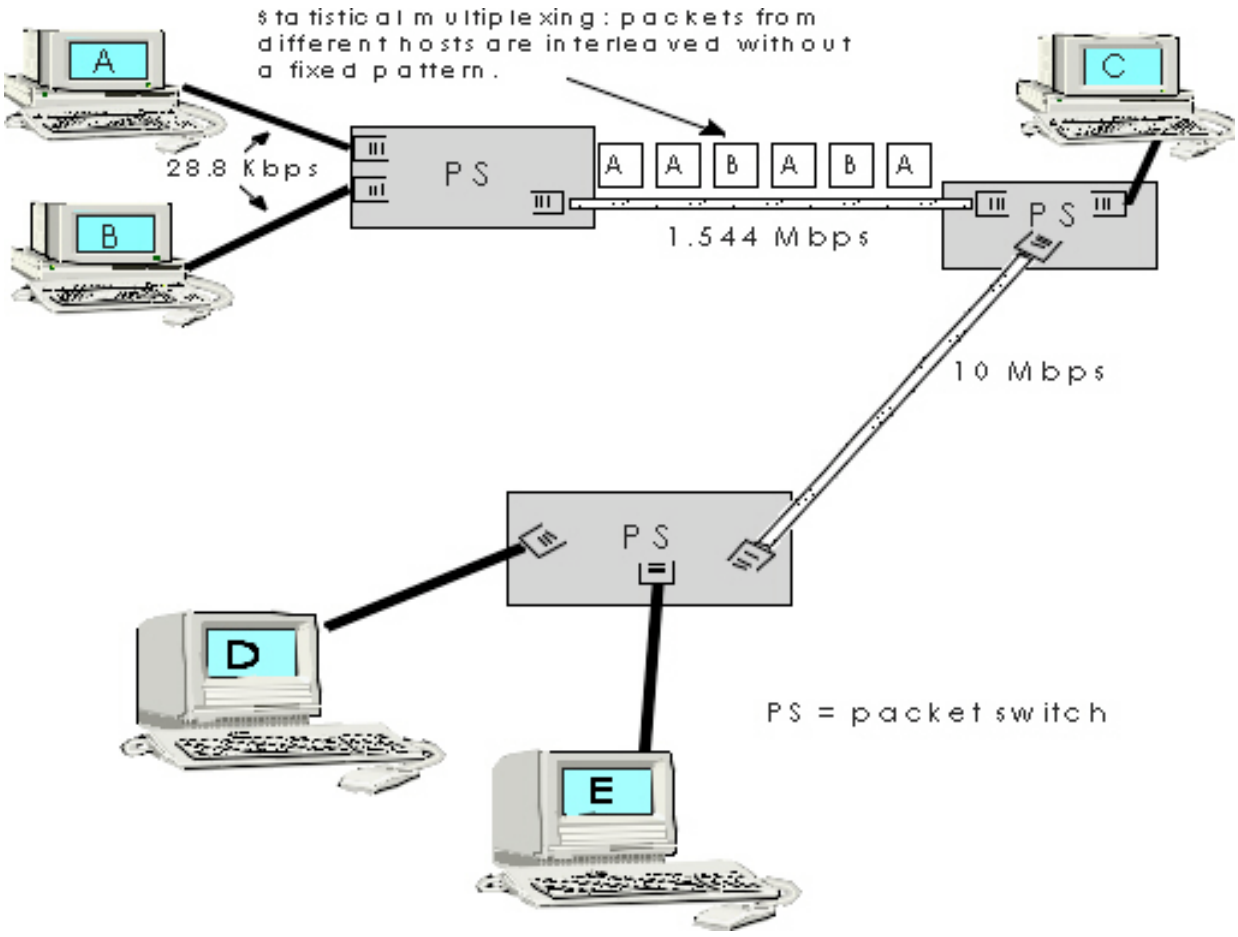

Figure 1.4-4: Packet switching

Let us now consider how long it takes to send a packet of *L* bits from host A to another host across a packet-switched network. Let us suppose that there are *Q* links between A and E, each of rate *R* bps. Assume that queueing delays and end-to-end propagation delays are negligible and that there is no connection establishment. The packet must first be transmitted onto the first link emanating from host A; this takes *L/R* seconds. It must then be transmitted on each of the *Q-1* remaining links, that is, it must be stored-and-forwarded *Q-1* times. Thus the total delay is *QL/R*.

Packet Switching versus Circuit Switching

Having described circuit switching and packet switching, let us compare the two. Opponents of packet switching have often argued that the packet switching is not suitable for real-time services (e.g., telephone calls and video conference calls) due to its variable and unpredictable delays. Proponents of packet switching argue that (1) it offers better sharing of bandwidth than circuit switching and (2) it is simpler, more efficient, and less costly to implement than circuit-switching. Generally speaking, people who do not like to hassle with restaurant reservations prefer packet switching to circuit switching.

Why is packet-switching more efficient? Let us look at a simple example. Suppose users share a 1 Mbps link. Also suppose that each user alternates between periods of activity (when it generates data at a constant rate of 100Kbits/sec) and periods of inactivity (when it generates no data). Suppose further that a user is active only 10% of the time (and is idle drinking coffee during the remaining 90% of the time). With circuit-switching, 100 Kbps must be *reserved* for *each* user at all times. Thus, the link can support only ten simultaneous users. With packet switching, if there are 35 users, the probability that there are 10 or more simultaneously active users is less than .0004. If there are 10 or less simultaneously active users (which happens with probability .9996), the aggregate arrival rate of data is less than 1Mbps (the output rate of the link). Thus, users' packets flow through the link essentially without delay, as is the case with circuit switching. When there are more than 10 simultaneously active users, then the aggregate arrival rate of packets will exceed the output capacity of the link, and the output queue will begin to grow (until the aggregate input rate falls back below 1Mbps, at which point the queue will begin to diminish in length). Because the probability of having ten or more simultaneously active users is very very small, packet-switching almost always has the same delay performance as circuit switching, *but does so while allowing for more than three times the number of users*.

Although packet switching and circuit switching are both very prevalent in today's telecommunication networks, the trend is certainly in the direction of packet switching. Even many of today's circuit-switched telephone networks are slowly migrating towards packet switching. In particular, telephone networks often convert to packet switching for the expensive overseas portion of a telephone call.

Message Switching

In a modern packet-switched network, the source host segments long messages into smaller packets and sends the smaller packets into the network; the receiver reassembles the packets back into the original message. But why bother to segment the messages into packets in the first place, only to have to reassemble packets into messages? Doesn't this place an additional and unnecessary burden on the source and destination? Although the segmentation and reassembly do complicate the design of the source and receiver, researchers and network designers concluded in the early days of packet switching that the advantages of segmentation greatly compensate for its complexity. Before discussing some of these advantages, we need to introduce some terminology. We say that a packet-switched network performs **message switching** if the sources do not segment messages, i.e., they send a message into the network as a whole. Thus message switching is a specific kind of packet switching, whereby the packets traversing the network are themselves entire messages.

Figure 1.4-5 illustrates message switching in a route consisting of two packet switches (PSs) and three links. With message switching, the message stays in tact as it traverses the network. Because the switches are store-and-forward packet switches, a packet switch must receive the entire message before it can begin to forward the message on an outbound link.

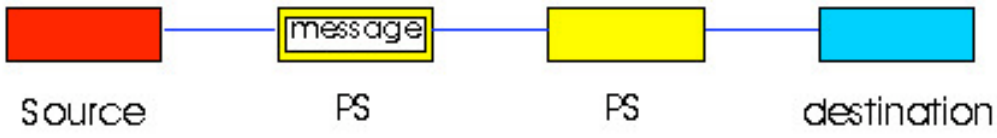

Figure 1.4-5: A simple message-switched network

Figure 1.4-6 illustrates packet switching for the same network. In this example the original message has been divided into five distinct packets. In Figure 1.4-6, the first packet has arrived at the destination, the second and third packets are in transit in the network, and the last two packets are still in the source. Again, because the switches are store-and-forward packet switches, a packet switch must receive an entire packet before it can begin to forward the packet on an outbound link.

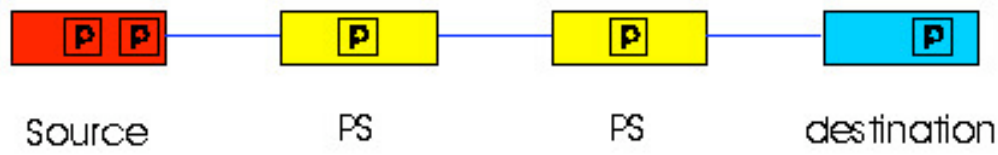

Figure 1.4-6: A simple packet-switched network

One major advantage of packet switching (with segmented messages) is that it achieves end-to-end delays that are typically much smaller than the delays associated with message-switching. We illustrate this point with the following simple example. Consider a message that is 7.5 Mbits long. Suppose that between source and destination there are two packet switches and three links, and that each link has a transmission rate of 1.5Mbps. Assuming there is no congestion in the network, how much time is required to move the message from source to destination with message switching? It takes the source 5 seconds to move the message from the source to the first switch. Because the switches use store-and-forward, the first switch cannot begin to transmit any bits in the message onto the link until this first switch has received the entire message. Once the first switch has received the entire message, it takes 5 seconds to move the message from the first switch to the second switch. Thus it takes ten seconds to move the message from the source to the second switch. Following this logic we see that a total of 15 seconds is needed to move the message from source to destination. These delays are illustrated in Figure 1.4-7.

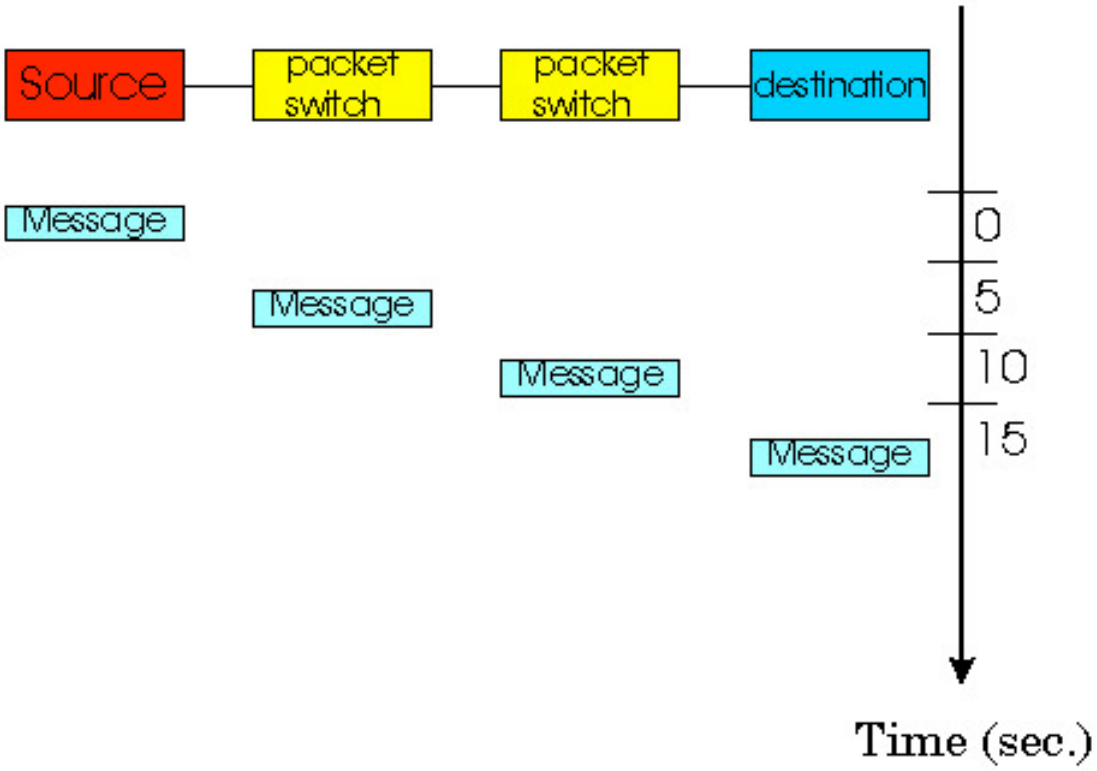

Figure 1.4-7: Timing of message transfer of a 7.5 Mbit message in a message-switched network

Continuing with the same example, now suppose that the source breaks the message into 5000 packets, with each packet being 1.5 Kbits long. Again assuming that there is no congestion in the network, how long does it take to move the 5000 packets from source to destination? It takes the source 1 msec to move the first packet from the source to the first switch. And it takes the first switch 1 msec to move this first packet from the first to the second switch. But while the first packet is being moved from the first switch to the second switch, the second packet is *simultaneously* moved from the source to the first switch. Thus the second packet reaches the first switch at time = 2 msec. Following this logic we see that the last packet is completely received at the first switch at time = 5000 msec = 5 seconds. Since this last packet has to be transmitted on two more links, the last packet is received by the destination at 5.002 seconds:.

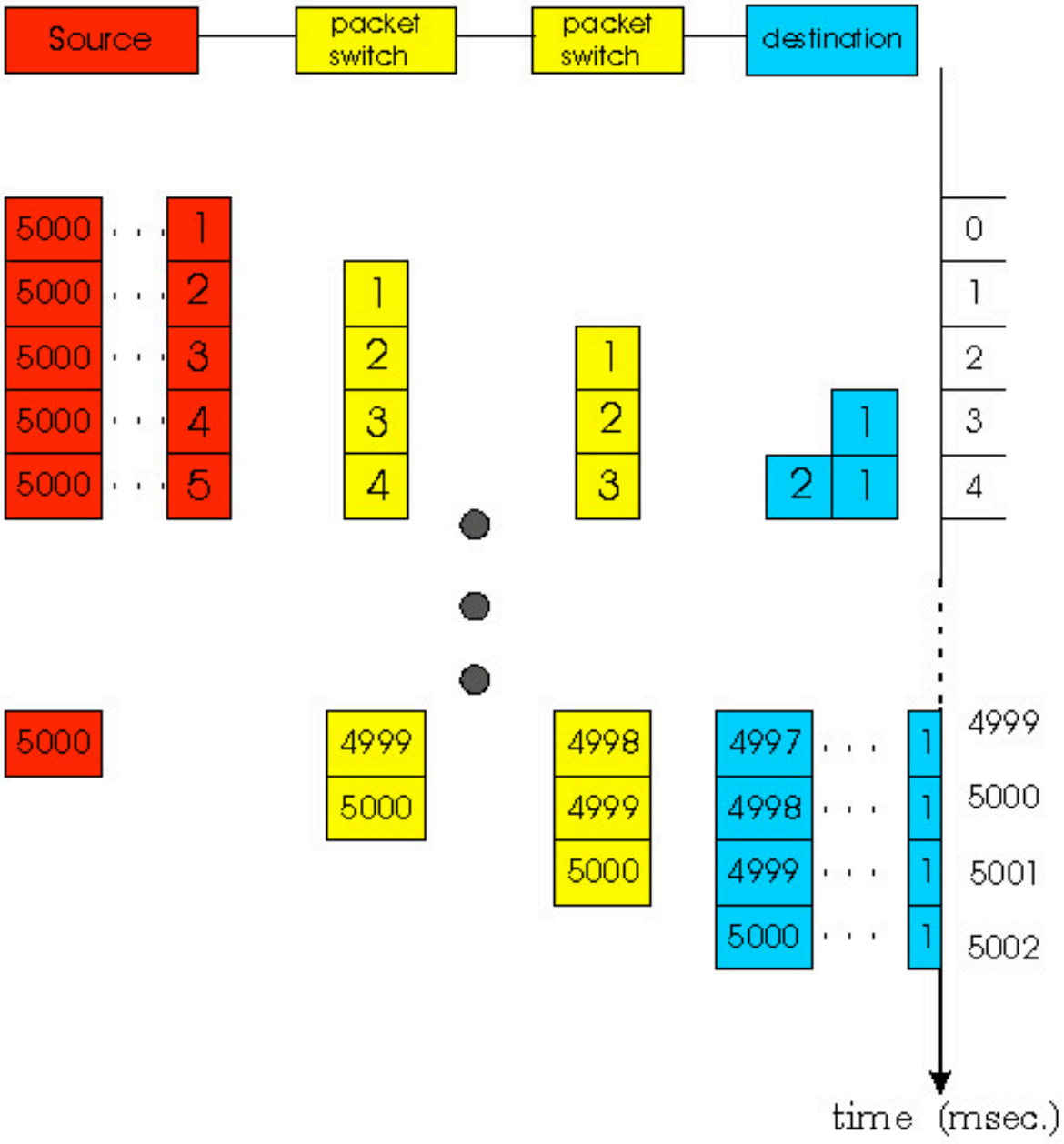

Figure 1.4-8: Timing of packet transfer of a 7.5 Mbit message, divided into 5000 packets, in a packet-switched network

Amazingly enough, packet-switching has reduced the message-switching delay by a factor of three! But why is this so? What is packet-switching doing that is different from message switching? The key difference is that message switching is performing sequential transmission whereas packet switching is performing parallel transmission. Observe that with message switching, while one node (the source or one of the switches) is transmitting, the remaining nodes are idle. With packet switching, once the first packet reaches the last switch, three nodes transmit at the same time.

Packet switching has yet another important advantage over message switching. As we will discuss later

in this book, bit errors can be introduced into packets as they transit the network. When a switch detects an error in a packet, it typically discards the entire packet. So, if the entire message is a packet and one bit in the message gets corrupted, the entire message is discarded. If, on the other hand, the message is segmented into many packets and one bit in one of the packets is corrupted, then only that one packet is discarded.

Packet switching is not without its disadvantages, however, with respect to message switching. We will see that each packet or message must carry, in addition to the data being sent from the sending application to the receiving application, an amount of control information. This information, which is carried in the packet or message **header**, might include the identity of the sender and receiver and a packet or message identifier (e.g., number). Since the amount of header information would be approximately the same for a message or a packet, the amount of header overhead per byte of data is higher for packet switching than for message switching.

Before moving on to the next subsection, you are highly encouraged to explore the Message Switching Java Applet. This applet will allow you to experiment with different message and packet sizes, and will allow you to examine the effect of additional propagation delays.

1.4.2 Routing in Data Networks

There are two broad classes of packet-switched networks: datagram networks and virtual-circuit networks. They differ according to whether they route packets according to host destination addresses or according to virtual circuit numbers. We shall call any network that routes packets according to host destination addresses a **datagram network.** The IP protocol of the Internet routes packets according to the destination addresses; hence the Internet is a datagram network. We shall call any network that routes packets according to virtual-circuit numbers a **virtual-circuit network**. Examples of packet-switching technologies that use virtual circuits include X.25, frame relay, and ATM.

Virtual Circuit Networks

A virtual circuit (VC) consists of (1) a path (i.e., a series of links and packet switches) between the source and destination hosts, (2) virtual circuit numbers, one number for each link along the path, and (3) entries in VC-number translation tables in each packet switch along the path. Once a VC is established between source and destination, packets can be sent with the appropriate VC numbers. Because a VC has a different VC number on each link, an intermediate packet switch must replace the VC number of each traversing packet with a new one. The new VC number is obtained from the VC-number translation table.

To illustrate the concept, consider the network shown in Figure 1.4-9. Suppose host A requests that the network establish a VC between itself and host B. Suppose that the network chooses the pathA - PS1 -

PS2 - B and assigns VC numbers 12, 22, 32 to the three links in this path. Then, when a packet as part of this VC leaves host A, the value in the VC number field is 12; when it leaves PS1, the value is 22; and when it leaves PS2, the value is 32. The numbers next to the links of PS1 are the **interface numbers**.

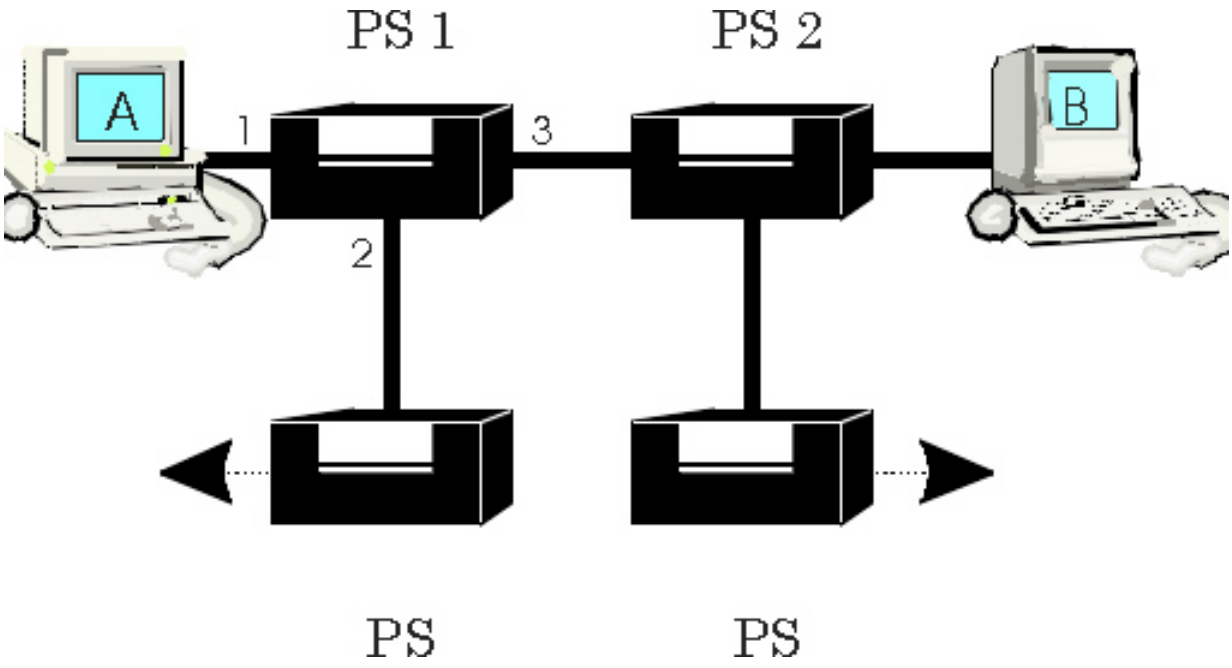

Figure 1.4-9: A simple virtual circuit network

How does the switch determine the replacement VC number for a packet traversing the switch? Each switch has a VC number translation table; for example, the VC number translation table in PS 1 might look something like this:

Incoming Interface	Incoming VC#	Outgoing Interface	Outgoing VC#
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

Whenever a new VC is established across a switch, an entry is added to the VC number table. Similarly, whenever a VC terminates, the entries in each table along its path are removed.

You might be wondering why a packet doesn't just keep the same VC number on each of the links along its route? The answer to this question is twofold. First, by replacing the number from link to link, the length of the VC field is reduced. Second, and more importantly, by permitting a different VC number for each link along the path of the VC, a network management function is simplified. Specifically, with the multiple VC numbers, each link in the path can choose a VC number independently of what the other links in the path chose. If a common number were required for all links along the path, the switches would have to exchange and process a substantial number of messages to agree on the VC number to be used for a connection.

If a network employs virtual circuits, then the network's switches must maintain **state information** for the ongoing connections. Specifically, each time a new connection is established across a switch, a new connection entry must be added to the switch's VC-number translation table; and each time a connection is released, an entry must be removed from the table. Note that even if there is no VC number translation, it is still necessary to maintain state information that associates VC numbers to interface numbers. The issue of whether or not a switch or router maintains state information for each ongoing connection is a crucial one - one which we return to shortly below.

Datagram Networks

Datagam networks are analogous in many respects to the postal services . When a sender sends a letter to a destination, the sender wraps the letter in an envelope and writes the destination address on the envelope. This destination address has a hierarchical structure. For example, letters sent to a location in the United States include the country (the USA), the state (e.g., Pennsylvania), the city (e.g., Philadelphia), the street (e.g., Walnut Street) and the number of the house on the street (e.g., 421). The postal services use the address on the envelope to route the letter to its destination. For example, if the letter is sent from France, then a postal office in France will first direct the letter to a postal center in the USA. This postal center in the USA will then send the letter to a postal center in Philadelphia. Finally a mail person working in Philadelphia will deliver the letter to its ultimate destination.

In a datagram network, each packet that traverses the network contains in its header the address of the destination. As with postal addresses, this address has a hierarchical structure. When a packet arrives at a packet switch in the network, the packet switch examines a portion of the packet's destination address and forwards the packet to an adjacent switch. More specifically, each packet switch has a routing table which maps destination addresses (or portions of the destination addresses) to an outbound link. When a packet arrives at switch, the switch examines the address and indexes its table with this address to find the appropriate outbound link. The switch then sends the packet into this outbound link.

The whole routing process is also analogous to the car driver who does not use maps but instead prefers to ask for directions. For example, suppose Joe is driving from Philadelphia to 156 Lakeside Drive in Orlando, Florida. Joe first drives to his neighborhood gas station and asks how to get to 156 Lakeside Drive in Orlando, Florida. The gas station attendant extracts the Florida portion of the address and tells Joe that he needs to get onto the interstate highway I-95 South, which has an entrance just next to the gas

station. He also tells Joe that once he enters Florida he should ask someone else there. Joe then takes I-95 South until he gets to Jacksonville, Florida, at which point he asks another gas station attendant for directions. The attendant extracts the Orlando portion of the address and tells Joe that he should continue on I-95 to Daytona Beach and then ask someone else. In Daytona Beach another gas station attendant also extracts the Orlando portion of the address and tells Joe that he should take I-4 directly to Orlando. Joe takes I-4 and gets off at the Orlando exit. Joe goes to another gas station attendant, and this time the attendant extracts the Lakeside Drive portion of the address, and tells Joe the road he must follow to get to Lakeside Drive. Once Joe reaches Lakeside Drive he asks a kid on a bicycle how to get to his destination. The kid extracts the 156 portion of the address and points to the house. Joe finally reaches his ultimate destination.

We will be discussing routing in datagram networks in great detail in this book. But for now we mention that, in contrast with VC networks, *datagram networks do not maintain connection state information in their switches*. In fact, a switch in a pure datagram network is completely oblivious to any flows of traffic that may be passing through it -- it makes routing decisions for each individual packet. Because VC networks must maintain connection state information in their switches, opponents of VC networks argue that VC networks are overly complex. These opponents include most researchers and engineers in the Internet community. Proponents of VC networks feel that VCs can offer applications a wider variety of networking services. Many researchers and engineers in the ATM community are outspoken advocates for VCs.

How would you like to actually see the route packets take in the Internet? We now invite you to get your hands dirty by interacting with the Traceroute program.

Network Taxonomy

We have now introduced several important networking concepts: circuit switching, packet switching, message switching, virtual circuits, connectionless service, and connection oriented service. How does it all fit together?

First, in our simple view of the World, a telecommunications network either employs circuit-switching or packet-switching:

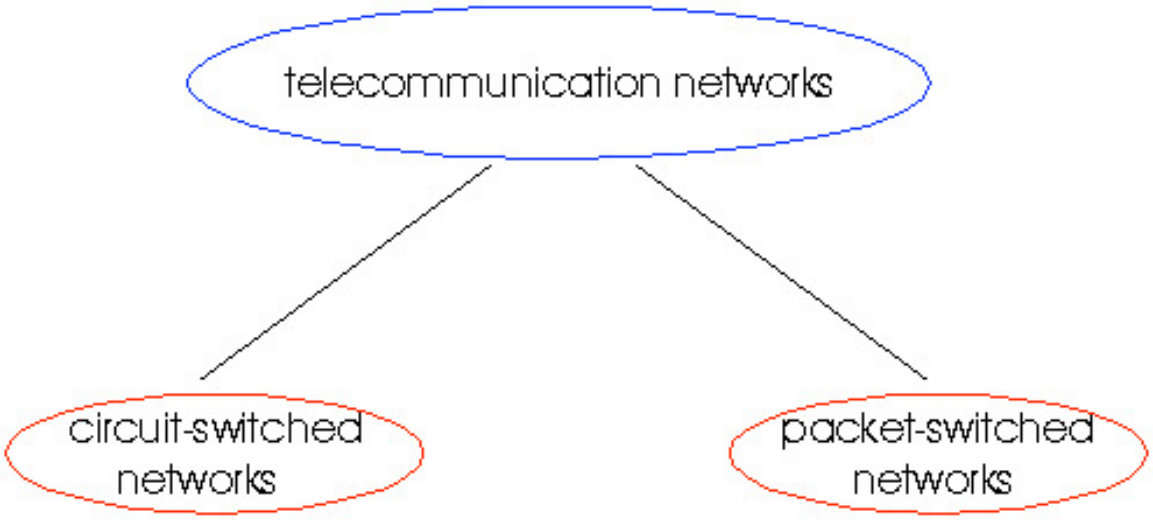

Figure 1.4-10: highest-level distinction among telecommunication networks: circuit-switched or packet-switched?

A link in a circuit-switched network can employ either FDM or TDM:

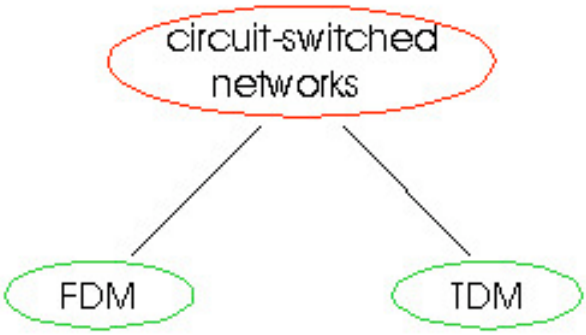

Figure 1.4-11: Circuit switching implementation: FDM or TDM?

Packet switch networks are either virtual-circuit networks or datagram networks. Switches in virtual-circuit networks route packets according to the packets' VC numbers and maintain connection state. Switches in datagram networks route packets according to the packets' destination addresses and do not maintain connection state:

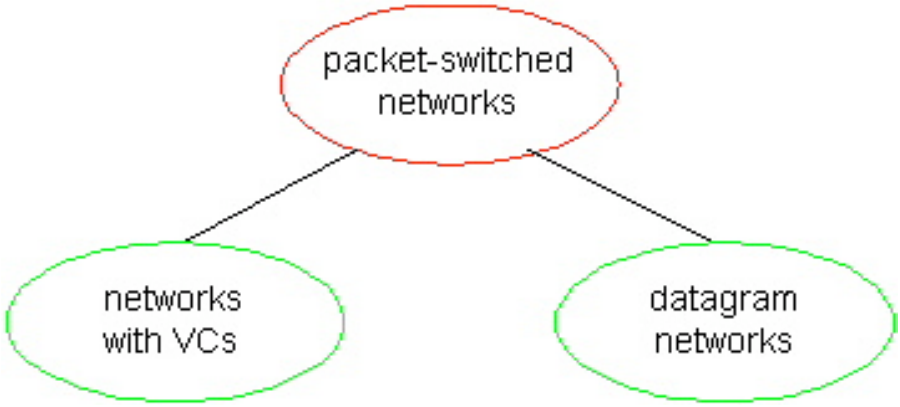

Figure 1.4-12: Packet switching implementation: virtual circuits or datagrams?

Examples of packet-switched networks which use VCs include X.25, frame relay, and ATM. A packet-switched network either (1) uses VCs for all of its message routing, or (2) uses destination addresses for all of its message routing. It doesn't employ both routing techniques. (This last statement is a bit of a white lie, as there are networks that use datagram routing "on top of" VC routing. This is the case for "IP over ATM," as we shall cover later in the book.)

A datagram network is *not*, however, either a connectionless or a connection-oriented network. Indeed, a datagram network can provide the connectionless service to some of its applications and the connection-oriented service to other applications. For example, the Internet, which is a datagram network, is a datagram network that provides both connectionless and connection-oriented service to its applications. We saw in section 1.3 that these services are provided in the Internet by the UDP and TCP protocols, respectively. Networks with VCs - such as X.25, Frame Relay, and ATM - are always, however, connection-oriented.

Return to Table Of Contents

Interactive Java Applet:

Message Switching & Packet Switching

This interactive applet enables you to actually see why packet switching can have much smaller delays than message switching when packets pass through store-and-forward switches. In this applet there are four nodes: a source (node A), a destination (node B), and two store-and-forward switches. Each packet sent from the source must be transmitted over three links before it reaches the destination. Each of these links has a transmission rate of 4 Kbps and an optional propagation delay of one second.

Each small rectangle represents 1 Kbit of data. When you press Start, the rectangles are grouped into one packet in the transmit buffer of the source. The packet is transmitted to the first switch, where it must be stored before it is forwarded. The packet then continues towards the destination.

To simulate **message switching**, set the packet size equal to the message size. To simulate **packet switching**, set the packet size to less than the message size. To examine the effect of link propagation delays, check the appropriate boxes for optional propagation delays. For a variety of scenarios, it is highly recommended that you calculate the end-to-end delay analytically and then verify your calculation with the applet.

Tracing Routes in the Internet

Traceoute is a popular program for tracing a packet's route from any source host to any destination host in the Internet. Before we explain what traceroute does and how it works, first try running the traceroute program. In the box below, enter the name of any host, such as *surf.eurecomf.fr* or *www.mit.edu*. The host name that you enter will be sent to a server located at IBM Israel in Tel-Aviv, Israel. The host in Tel-Aviv will respond with the route taken from Tel-Aviv to the host you have listed in the box below. After running the program, return to this page for a discussion of the traceroute program.

Host address or name Submit

Leave empty to find the route to your browser.

After having traced the route from Tel-Aviv to your favorite host, try it again with a new starting place -- Dana Point in sunny southern California.

Host address or name Submit

What Traceroute Does and How It Works

The main packet switches in the Internet are called **routers,** and routers use **datagram routing**. Specifically, when a source constructs a packet, it appends the destination address onto the packet. When the packet arrives at a router, the switch determines the appropriate outgoing link for the packet by examining the packet's destination address.

Traceroute is a little program that can run in any Internet host. When the user specifies a destination host name, the program sends multiple packets towards that destination. As these packets work their way towards the destinations, they pass through a series of routers. When a router receives one of these packets, it sends a little message back to the source. This message contains the name and address of the router.

More specifically, suppose there are *N-1* routers between the soruce and the destination. Then the source will send *N* packets into the network, with each packet addressed to the ultimate destination. These packets are also marked *1* through *N*, with the first of the *N* packets marked *1* and the last of the *N* packets marked *N*. When the *n*th router receives the *n*th packet marked *n*, the router destroys the packet and sends a message to the source. And when the destination host receives the *N*th packet, the

destination destoys it as well, but again returns a message back to the source. The source records the time that elapses from when it sends a packet until when it receives the corresponding return message; it also records the name and address of the router (or the destination host) that returns the message. In this manner, the source can reconstruct the route taken by packets flowing from source to destination, and the source can determine the round-trip delays to all the intervening routers. Traceroute actually repeats the experiment just described three times, so the source actually sends *3*N* packets to the destination.

The [RFC 1393] describes traceout in detail. The Internet Encyclopedia as also gives an overview of how traceroute works.

Here is an example of the output of the traceroute program, where the route is being traced from the source host *eniac.seas.upenn.edu* (at the University of Pennsylvania) to *diane.ibp.fr* (at the University of Paris VI). The output has six columns: the first column is the *n* value described above, i.e., the number of the router along the route; the second column is the name of the router; the third column is the address of the router (of the form xxx.xxx.xxx.xxx); the last three columns are the round-trip delays for three experiments. If the source receives less than three messages from any given router, because of packet loss in the network, traceroute places an asterisk just after the router number and reports less than three round-trip times for that router.

1 GW.CIS.UPENN.EDU (130.91.6.254) 3 ms 2 ms 1 ms

2 DEFAULT7-GW.UPENN.EDU (165.123.247.8) 3 ms 1 ms 2 ms

3 192.204.183.1 (192.204.183.1) 3 ms 4 ms 3 ms

4 border2-hssi1-0.WestOrange.mci.net (204.70.66.5) 6 ms 6 ms 6 ms

5 core1-fddi-1.WestOrange.mci.net (204.70.64.33) 7 ms 6 ms 6 ms

6 somerouter.sprintlink.net (206.157.77.106) 16 ms 305 ms 192 ms

7 somerouter.sprintlink.net (206.157.77.106) 20 ms 196 ms 18 ms

8 sl-dc-6-H2/0-T3.sprintlink.net (144.228.10.33) 19 ms 18 ms 24 ms

9 198.67.0.1 (198.67.0.1) 19 ms 24 ms 18 ms

10 gsl-dc-3-Fddi0/0.gsl.net (204.59.144.197) 19 ms 18 ms 20 ms

11 * raspail-ip.eurogate.net (194.206.207.6) 133 ms 94 ms

12 raspail-ip2.eurogate.net (194.206.207.57) 93 ms 95 ms 97 ms

13 194.206.207.17 (194.206.207.17) 200 ms 94 ms 209 ms

14 stamand1.renater.ft.net (192.93.43.185) 105 ms 101 ms 105 ms

15 stlambert.rerif.ft.net (192.93.43.117) 108 ms 102 ms 95 ms

16 danton1.rerif.ft.net (193.48.53.50) 110 ms 97 ms 91 ms

17 u-jussieu-paris.rerif.ft.net (193.48.58.122) 94 ms 96 ms 100 ms

18 r-jusren.reseau.jussieu.fr (192.44.54.126) 100 ms 94 ms 100 ms

19 r-ibp.reseau.jussieu.fr (134.157.254.250) 96 ms 100 ms 94 ms

20 masi.ibp.fr (132.227.60.23) 121 ms 100 ms 97 ms

21 * diane.ibp.fr (132.227.64.48) 105 ms 102 ms

In the above trace there are no routers between the source and the destination. Most of these routers have a name, and all of them have addresses. For example, the name of router 8 is sl-dc-6-H2/0-T3.sprintlink.net and its address is 144.228.10.33. Looking at the data provided for this same router, we see that in the first of the three trials the roundtrip delay between the source and the router 8 was 19 msec. The roundtrip delays for the subsequent two trials were 18 and 24 msec. These roundtrip delays include packet propagation delays, router processing delays, and queueing delays due to congestion in the Internet. Because the congestion is varying with time, the roundtrip delay to a router *n* can actually be longer than the roundtrip delay to router *n+1*. Note in the above example that there is a big jump in the round-trip delay when going from router 10 to router 11. This is because the link between routers 10 and 11 is a transatlantic link.

Want to try out traceroute from some other starting points besides Tel-Aviv and Dana Point? Then visit Yahoo's List of sites offering route tracing.

References

[RFC 1393] G. Malkin, "Traceroute Using an IP Option," RFC 1393, January 1993.

Return to Table Of Contents

1.5 Access Networks and Physical Media

In sections 1.3 and 1.4 we have examined the roles of end systems and routers in a network architecture. In this section we consider the **access network** - the physical link(s) that connect an end system to its **edge router**, i.e., the first router on a path from the end system to any other distant end system.. Since access network technology is closely tied to physical media technology (fiber, coaxial pair, twisted pair telephone wire, radio spectrum), we consider these two topics together in this section.

1.5.1 Access Networks

Figure 1.5-1 shows the access networks' links highlighted in red.

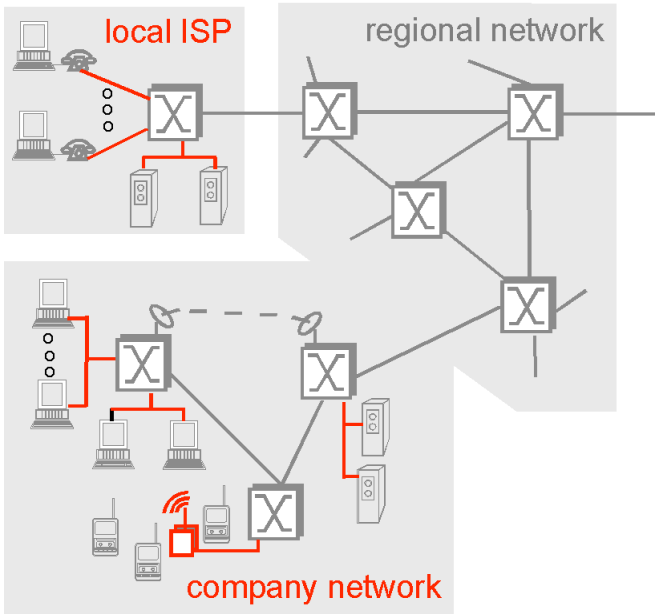

Figure 1.5-1: Access networks

Access networks can be loosely divided into three categories:

- **residential access networks**, connecting a home end system into the network;
- **institutional access networks**, connecting an end system in a business or educational institution into the network;
- **mobile access networks**, connecting a mobile end system into the network

These categories are not hard and fast; some corporate end systems may well use the access network technology that we ascribe to residential access networks, and vice versa. Our descriptions below are meant to hold for the common (if not every) case.

Residential Access Networks

A residential access network connects a home end system (typically a PC, but perhaps a Web TV or other residential system) to an edge router. Probably the most common form of home access is using a **modem** over a POTS (plain old telephone system) dialup line to an Internet service provider (ISP). The home modem converts the digital output of the PC into analog format for transmission over the analog phone line. A modem in the ISP converts the analog signal back into digital form for input to the ISP router. In this case, the "access network" is simply a point-to-point dialup link into an edge router. The point-to-point link

is your ordinary twisted-pair phone line. (We will discuss twisted pair later in this section.) Today's modem speeds allow dialup access at rates up to 56 Kbps. However, due to the poor quality of twisted-pair line between many homes and ISPs, many users get an effective rate significantly less than 56 Kbps. For an in depth discussion of the practical aspects of modems see the Institute for Global Communications (IGC) web page on Modems and Data Communications.

While dialup modems require conversion of the end system's digital data into analog form for transmission, so-called narrowband **ISDN** technology (Integrated Services Digital Network) [Pacific Bell 1998] allows for all-digital transmission of data from a home end system over ISDN "telephone" lines to a phone company central office. Although ISDN was originally conceived as a way to carry digital data from one end of the phone system to another, it is also an important network access technology that provides higher speed access (e.g., 128 Kbps) from the home into a data network such as the Internet. In this case, ISDN can be thought of simply as a "better modem" [NAS 1995]. A good source for additional WWW information on ISDN is Dan Kegel's ISDN page.

Dialup modems and narrowband ISDN are already widely deployed technologies. Two new technologies, **Asymmetric Digital Subscriber Line (ADSL)** [ADSL 1998] and **hybrid fiber coaxial cable (HFC)** [Cable 1998] are currently being deployed. ADSL is conceptually similar to dialup modems: it is a new modem technology again running over existing twisted pair telephone lines, but can transmit at rates of up to about 8 Mbps from the ISP router to a home end system. The data rate in the reverse direction, from the home end system to the central office router, is less than 1 Mbps. The asymmetry in the access speeds gives rise to the term "Asymmetric" in ADSL. The asymmetry in the data rates reflects the belief that home users are more likely to be a consumer of information (bringing data into their homes) than a producer of information.

ADSL uses frequency division multiplexing, as described in the previous section. In particular, ADSL divides the communication link between the home the ISP into three non-overlapping frequency bands:

- a **high-speed downstream channel**, in the 50 KHz to 1 MHz band;
- a **medium-speed upstream channel**, in the 4 KHz to 50 KHz band;
- and an **ordinary POTs two-way telephone channel**, in the 0 to 4 KHz band.

One of the features of ADSL is that the service allows the user to make an ordinary telephone call, using the POTs channel, while simultaneously surfing the Web. This feature is not available with standard dailup modems. The actually amount of downstream and upstream bandwidth available to the user is a function of the distance between the home modem and the ISP modem, the gauge of the twisted pair line, and the degree of electrical interference. For a high-quality line with negligible electrical interference, an 8 Mbps downstream transmission rate is possible if the distance between the home and the ISP is less than 3,000 meters; the downstream transmission rate drops to about 2 Mbps for a distance of 6,000 meters. The upstream rate ranges from 16 Kbps to 1 Mbps.

While ADSL, ISDN and dailup modems all use ordinary phone lines, HFC access networks are extensions of the current cable network used for broadcasting cable television. In a traditional cable system, a cable head end station broadcasts through a distribution of coaxial cable and amplifiers to residences. (We discuss coaxial cable later in this chapter.) As illustrated in Figure 1.5-2, fiber optics (also to be discussed soon) connect the cable head end to neighborhood-level junctions, from which traditional coaxial cable is then used to reach individual houses and apartments. Each neighborhood juncture typically supports 500 to 5000 homes.

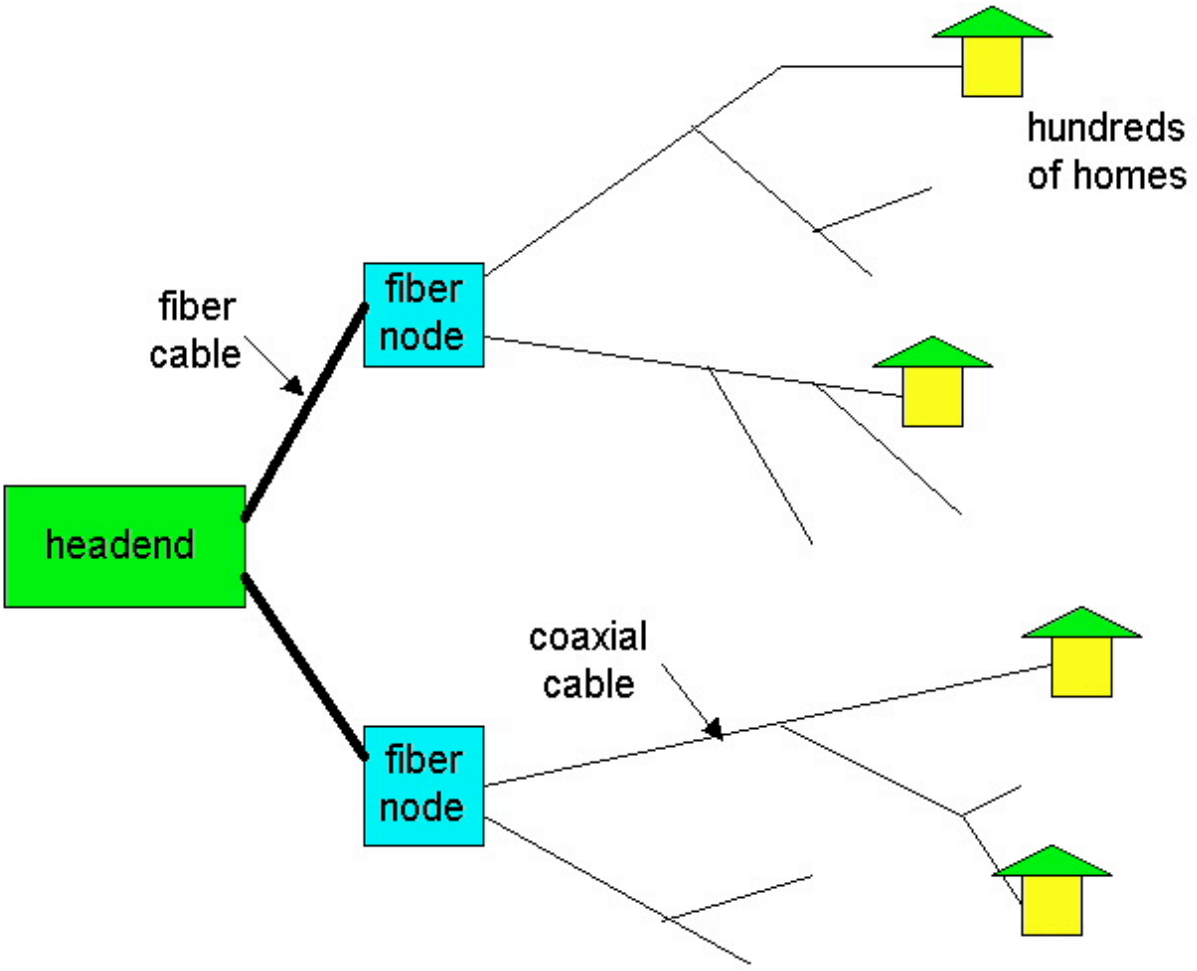

Figure 1.5-2: A hybrid fiber-coax access network

As with ADSL, HFC requires special modems, called cable modems. Companies that provide cable Internet access require their customers to either purchase or lease a modem. One such company is CyberCable, which uses Motorola's CyberSurfer Cable Modem and provides high-speed Internet access to most of the neighborhoods in Paris. Typically, the cable modem is an external device and connects to the home PC through a 10-BaseT Ethernet port. (We will discuss Ethernet in great detail in Chapter 5.) Cable modems divide the HFC network into two channels, a downstream and an upstream channel. As with ADSL, the downstream channel is typically allocated more bandwidth and hence a larger transmission rate. For example, the downstream rate of the CyberCable system is 10 Mbps and the upstream rate is 768 Kbps. However, with HFC (and not with ADSL), these rates are shared among the homes, as we discuss below.

One important characteristic of the HFC is that it is a shared broadcast medium. In particular, every packet sent by the headend travels downstream on every link to every home; and every packet sent by a home travels on the upstream channel to the headend. For this reason, if several users are receiving different Internet videos on the downstream channel, actual rate at which each user receives its video will be significantly less than downstream rate. On the other hand, if all the active users are Web surfing, then each of the users may actually receive Web pages at the full downstream rate, as a small collection of users will rarely receive a Web page at exactly the same time. Because the upstream channel is also shared, packets sent by two different homes at the same time will collide, which further decreases the effective upstream bandwidth. (We will discuss this collision issue in some detail when we discuss Ethernet in Chapter 5.) Advocates of ADSL are quick to point out that ADSL is a point-to-point connection between the home and ISP, and therefore all the ADSL bandwidth is dedicated rather than shared. Cable advocates, however, argue that a reasonably dimensioned HFC network provides higher bandwidths than ADSL [@Home

1998]. The battle between ADSL and HFC for high speed residential access has clearly begun, e.g., [@Home 1998].

Enterprise Access Networks

In enterprise access networks, a local area network (LAN) is used to connect an end system to an edge router. As we will see in Chapter 5, there are many different types of LAN technology. However, Ethernet technology is currently by far the most prevalent access technology in enterprise networks. Ethernet operates 10 Mbps or 100Mbps (and now even at 1 Gbps). It uses either twisted-pair copper wire are coaxial cable to connect a number of end systems with each other and with an edge router. The edge router is responsible for routing packets that have destinations outside of that LAN. Like HFC, Ethernet uses a shared medium, so that end users share the the transmission rate of the LAN. More recently, shared Ethernet technology has been migrating towards switched Ethernet technology. Switched Ethernet uses multiple coaxial cable or twisted pair Ethernet segments connected at a "switch" to allow the full bandwidth an Ethernet to be delivered to different users on the same LAN simultaneously [Cisco 1998]. We will explore shared and switched Ethernet in some detail in Chapter 5.

Mobile Access Networks

Mobile access networks use the radio spectrum to connect a mobile end system (e.g., a laptop PC or a PDA with a wireless modem) to a base station, as shown in Figure 1.5-1. This base station, in turn, is connected to an edge router of a data network.

An emerging standard for wireless data networking is Cellular Digital Packet Data (CDPD) [Wireless 1998]. As the name suggests, a CDPD network operates as an overlay network (i.e., as a separate, smaller "virtual" network, as a piece of the larger network) within the cellular telephone network. A CDPD network thus uses the same radio spectrum as the cellular phone system, and operates at speeds in the 10's of Kbits per second. As with cable-based access networks and shared Ethernet, CDPD end systems must share the transmission media with other CDPD end systems within the cell covered by a base station. A media access control (MAC) protocol is used to arbitrate channel sharing among the CDPD end systems; we will cover MAC protocols in detail in Chapter 5.

The CDPD system supports the IP protocol, and thus allows an IP end system to exchange IP packets over the wireless channel with an IP base station. A CDPD network can actually support multiple network layer protocols; in addition to IP, the ISO CNLP protocol is also supported. CDPD does not provide for any protocols above the network layer. From an Internet perspective, CDPD can be viewed as extending the Internet dialtone (i.e., the ability to transfer IP packets) across a wireless link between a mobile end system and an Internet router. An excellent introduction to CDPD is [Waung 98].

1.5.2 Physical Media

In the previous subsection we gave an overview of some of the most important access network technologies in the Internet. While describing these technologies, we also indicated the physical media used. For example, we said that HFC uses a combination of fiber cable and coaxial cable. We said that ordinary modems, ISDN, and ADSL use twisted-pair copper wire. And we said that mobile access network use the radio spectrum. In this subsection we provide a brief overview of these and other transmission media that are commonly employed in the Internet.

In order to define what is meant by a "physical medium,", let us reflect on the brief life of a bit. Consider a bit traveling from one end system, through a series of links and routers, to another end system. This poor bit gets transmitted many, many times! The source end-system first transmits the bit and shortly thereafter the first router in the series receives the bit; the first router then transmits the bit and shortly afterwards the second router receives the bit, etc. Thus our bit, when traveling from source to destination, passes through a series of transmitter-receiver pairs. For each transmitter-receiver pair, the bit is sent by propagating electromagnetic waves across a **physical medium**. The physical medium can take many shapes and forms, and does not have to be of the same type for each transmitter-receiver pair along the path. Examples of physical media include

twisted-pair copper wire, coaxial cable, multimode fiber optic cable, terrestrial radio spectrum and satellite radio spectrum. Physical media fall into two categories: **guided media** and **unguided media**. With guided media, the waves are guided along a solid medium, such as a fiber-optic cable, a twisted-pair cooper wire or a coaxial cable. With unguided media, the waves propagate in the atmosphere and in outer space, such as in a digital satellite channel or in a CDPD system.

Some Popular Physical Media

Suppose you want to wire a building to allow computers to access the Internet or an intranet -- should you use twisted-pair copper wire, coaxial cable, or fiber optics? Which of these media gives the highest bit rates over the longest distances? We shall address these questions below.

But before we get into the characteristics of the various guided medium types, let us say a few words about their costs. The actual cost of the physical link (copper wire, fiber optic cable, etc.) is often relatively minor compared with the other networking costs. In particular, the labor cost associated with the installation of the physical link can be orders of magnitude higher than the cost of the material. For this reason, many builders install twisted pair, optical fiber, and coaxial cable to every room in a building. Even if only one medium is initially used, there is a good chance that another medium could be used in the near future, and so money is saved but not having to lay additional wires.

Twisted-Pair Copper Wire

The least-expensive and most commonly-used transmission medium is twisted-pair copper wire. For over one-hundred years it has been used by telephone networks. In fact, more than 99% of the wired connections from the telephone handset to the local telephone switch use twisted-pair copper wire. Most of us have seen twisted pair in our homes and work environments. Twisted pair consists of two insulated copper wires, each about 1 mm thick, arranged in a regular spiral pattern; see Figure 1.5-3. The wires are twisted together to reduce the electrical interference from similar pairs close by. Typically, a number of pairs are bundled together in a cable by wrapping the pairs in a protective shield. A wire pair constitutes a single communication link.

Figure 1.5-3: Twisted Pair

Unshielded twisted pair (UTP) is commonly used for computer networks within a building, that is, for local area networks (LANs). Data rates for LANs using twisted pair today range from 10 Mbps to 100 Mbps. The data rates that can be achieved depend on the thickness of the wire and the distance between transmitter and receiver. Two types of UTP are common in LANs: category 3 and category 5. Category 3 corresponds to voice-grade twisted pair, commonly found in office buildings. Office buildings are often prewired with two or more parallel pairs of category 3 twisted pair; one pair is used for telephone communication, and the additional pairs can be used for additional telephone lines or for LAN networking. 10 Mbps Ethernet, one of the most prevalent LAN types, can use category 3 UTP. Category 5, with its more twists per centimeter and Teflon insulation, can handle higher bit rates. 100 Mbps Ethernet running on category 5 UTP has become very popular in recent years. In recent years, category 5 UTP has become common for preinstallation in new office buildings.

When fiber-optic technology emerged in the 1980s, many people disparaged twisted-pair because of its relatively low bit rates. Some people even felt that fiber optic technology would completely replace twisted pair. But twisted pair did not give up so easily. Modern twisted-pair technology, such as category 5 UTP, can achieve data rates of 100 Mbps for distances up to a few hundred meters. Even higher rates are possible over shorter distances. In the end, twisted-pair has emerged as the dominant solution for high-speed LAN networking.

As discussed in Section 1.5.1, twisted-pair is also commonly used for residential Internet access. We saw that dial-up modem technology enables access at rates of up to 56 Kbps over twisted pair. We also saw that ISDN is available in many

communities, providing access rates of about 128 Kbps over twisted pair. We also saw that ADSL (Asymmetric Digital Subscriber Loop) technology has enabled residential users to access the Web at rates in excess of 6 Mbps over twisted pair.

Coaxial-Cable

Like twisted pair, coaxial cable consists of two copper conductors, but the two conductors are concentric rather than parallel. With this construction and a special insulation and shielding, coaxial cable can have higher bit rates than twisted pair. Coaxial cable comes in two varieties: **baseband coaxial cable** and **broadband coaxial cable**.

Baseband coaxial cable, also called 50-ohm cable, is about a centimeter thick, lightweight, and easy to bend. It is commonly used in LANs; in fact, the computer you use at work or at school is probably connected to a LAN with either baseband coaxial cable or with UTP. Take a look at the the connection to your computer's interface card. If you see a telephone-like jack and some wire that resembles telephone wire, you are using UTP; if you see a T-connector and a cable running out of both sides of the T-connector, you are using baseband coaxial cable. The terminology "baseband" comes from the fact that the stream of bits is dumped directly into the cable, without shifting the signal to a different frequency band. 10 Mbps Ethernets can use either UTP or baseband coaxial cable. As we will discuss in the Chapter 5, it is a little more expensive to use UTP for 10 Mbps Ethernet, as UTP requires an additional networking device, called a **hub**.

Broadband coaxial cable, also called 75-ohm cable, is quite a bit thicker, heavier, and stiffer than the baseband variety. It was once commonly used in LANs and can still be found in some older installations. For LANs, baseband cable is now preferable, since it is less expensive, easier to physically handle, and does not require attachment cables. Broadband cable, however, is quite common in cable television systems. As we saw in Section 1.5.1, cable television systems have been recently been coupled with cable modems to provide residential users with Web access at rates of 10 Mbps or higher. With broadband coaxial cable, the transmitter shifts the digital signal to a specific frequency band, and the resulting analog signal is sent from the transmitter to one or more receivers. Both baseband and broadband coaxial cable can be used as a guided **shared medium**. Specifically, a number of end systems can be connected directly to the cable, and all the end systems receive whatever any one of the computers transmits. We will look at this issue in more detail in Chapter 5.

Fiber Optics

An optical fiber is a thin, flexible medium that conducts pulses of light, with each pulse representing a bit. A single optical fiber can support tremendous bit rates, up to tens or even hundreds of gigabits per second. They are immune to electromagnetic interference, have very low signal attenuation up to 100 kilometers, and are very hard to tap. These characteristics have made fiber optics the preferred long-haul guided transmission media, particularly for overseas links. Many of the long-distance telephone networks in the United States and elsewhere now use fiber optics exclusively. Fiber optics is also prevalent in the backbone of the Internet. However, the high cost of optical devices -- such as transmitters, receivers, and switches -- has hindered their deployment for short-haul transport, such as in a LAN or into the home in a residential access network. AT&T Labs provides an excellent site on fiber optics, including several nice animations.

Terrestrial and Satellite Radio Channels

Radio channels carry signals in the electromagnetic spectrum. They are an attractive media because require no physical "wire" to be installed, can penetrate walls, provide connectivity to a mobile user, and can potentially carry a signal for long distances. The characteristics a radio channel depend significantly on the propagation environment and the distance over which a signal is to be carried. Environmental considerations determine path loss and shadow fading (which decrease in signal strength as it travels over a distance and around/through obstructing objects), multipath fading (due to signal reflection off of interfering objects), and interference (due to other radio channels or electromagnetic signals).

Terrestrial radio channels can be broadly classified into two groups: those that operate as local area networks (typically spanning 10's to a few hundred meters) and wide-area radio channels that are used for mobile data services (typically operating within a metropolitan region). A number of wireless LAN products are on the market, operating in the 1 to 10's of Mbps range.

Mobile data services (such as the CDPD standard we touched on in section 1.3), typically provide channels that operate at 10's of Kbps. See [Goodman 97] for a survey and discussion of the technology and products.

A communication satellite links two or more earth-based microwave transmitter/receivers, known as ground stations. The satellite receives transmissions on one frequency band, regenerates the signal using a repeater (discussed below), and transmits the signal on another frequency. Satellites can provide bandwidths in the gigabit per second range. Two types of satellites are used in communications: geostationary satellites and low-altitude satellites.

Geostationary satellites permanently remain above the same spot on the Earth. This stationary presence is achieved by placing the satellite in orbit at 36,000 kilometers above the Earth's surface. This huge distance between from ground station though satellite back to ground station introduces a substantial signal propagation delay of 250 milliseconds. Nevertheless, satellites links are often used in telephone networks and in the backbone of the Internet.

Low-altitude satellites are placed much closer to the Earth and do not remain permanently above one spot on the Earth. They rotate around the Earth just as the Moon rotates around the Earth. To provide continuous coverage to an area, many satellites to be placed in orbit. There are currently many low-altitude communication systems in development. The Iridium system, for example, consists of 66 low-altitude satellites. Lloyd's satellite constellations provides and collects information on Iridium as well as other satellite constellation systems. The low-altitude satellite technology may be used for Internet access sometime in the future.

Return to Table Of Contents

References

[@Home 1998] @Home, "xDSL vs. @HOME™ 'S Hybrid-fiber-coaxial (HFC) Cable Modem Network: the Facts," 1998.
[ADSL 1998] ADSL Forum, ADSL Tutorial, 1998.
[Cable 1998] Cable Data News, "Overview of Cable Modem Technology and Services," 1998.
[Cisco 1998] Cisco, "Designing Switched LAN Internetworks," 1998.
[Goodman 1997] D. Goodman (Chair), "The Evolution of Untethered Communications," National Academy Press, December 1997.
[NAS 1995] National Academy of Sciences, "The Unpredictable Certainty: Information Infrastructure Through 2000," 1995.
[Pacific Bell 1998] Pacific Bell, "ISDN Users Guide," http://www.pacbell.com/products/business/fastrak/networking/isdn/info/isdn-guide/index.html
[Waung 1998] W. Waung, "Wireless Mobile Data Networking The CDPD Approach," Wireless Data Forum, 1998.
[Wireless 1998] Wireless Data Forum, "CDPD System Specification Release 1.1," 1998

1.6 Delay and Loss in Packet-Switched Networks

Having now briefly considered the major "pieces" of the Internet architecture - the applications, end systems, end-to-end transport protocols, routers, and links - let us now consider what can happen to a packet as it travels from its source to its destination. Recall that a packet starts in a host (the source), passes through a series of routers, and ends its journey in another host (the destination). As a packet travels from one node (host or router) to the subsequent node (host or router) along this path, the packet suffers from several different types of delays at *each* node along the path. The most important of these delays are the **nodal processing delay**, **queuing delay**, **transmission delay** and **propagation delay;** together, these delays accumulate to give a **total nodal delay**. In order to acquire a deep understanding of packet switching and computer networks, we must understand the nature and importance of these delays.

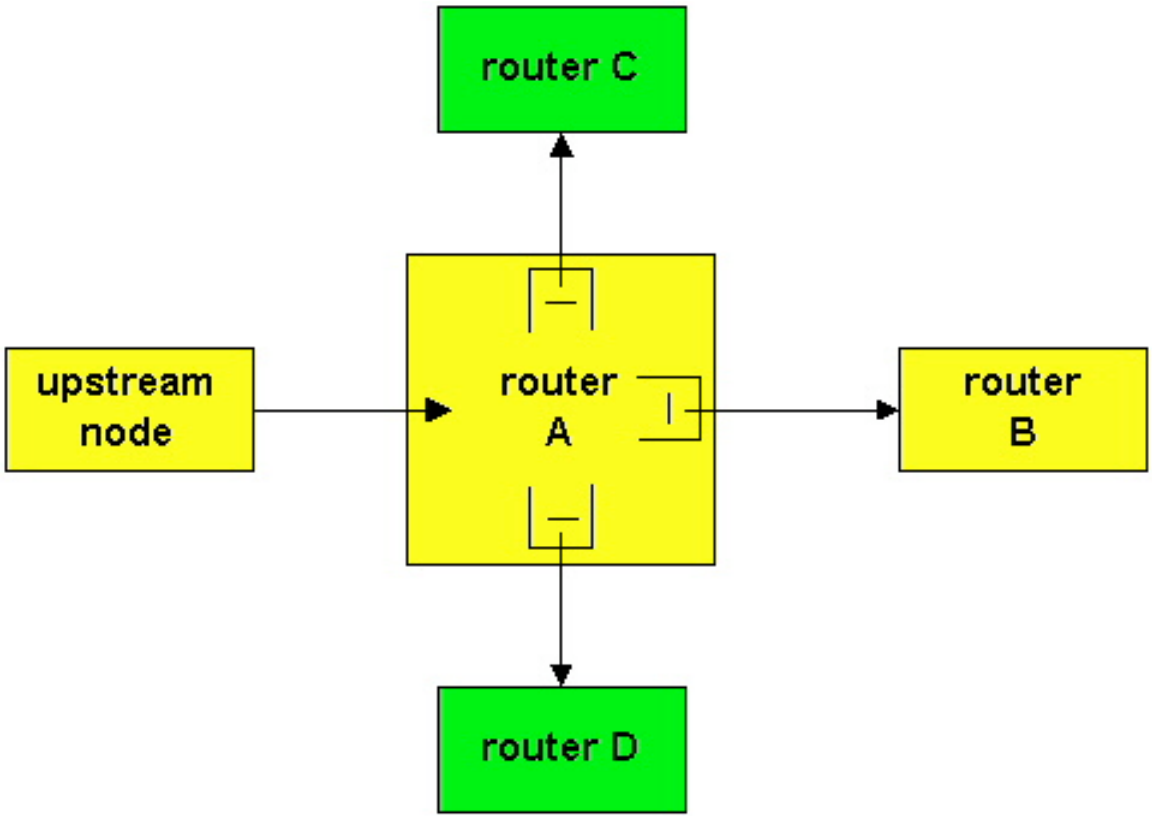

Figure 1.6-1: The delay through router A

Let us explore these delays in the context of Figure 1.6-1. As part of its end-to-end route between source and destination, a packet is sent from the upstream node through router, A, to router B. Our goal is to characterize the nodal delay at router A. Note that router A has three outbound links, one leading to

router B, another leading to router C, and yet another leading to router D. Each link is preceded a queue (also known as a buffer). When the packet arrives at router A (from the upstream node), router A examines the packet's header to determine the appropriate outbound link for the packet, and then directs the packet to the link. In this example, the outbound link for the packet is the one that leads to router B. A packet can only be transmitted on a link if there is no other packet currently being transmitted on the link and if there are no other packets preceding it in the queue; if the link is currently busy or if there are other packets already queued for the link, the newly arriving packet will then join the queue.

The time required to examine the packet's header and determine where to direct the packet is part of the **processing delay**. The processing delay can also include other factors, such as the time needed to check for bit-level errors in the packet that occurred in transmitting the packet's bits from the upstream router to router A. After this nodal processing, the router directs the packet to the queue that precedes the link to router B. (In section 4.7 we will study the details of how a router operates.) At the queue, the packet experiences a **queuing delay** as it waits to be transmitted onto the link. The queuing delay of a specific packet will depend on the number of other, earlier-arriving packets that are queued and waiting for transmission across the link; the delay of a given packet can vary significantly from packet to packet. If the queue is empty and no other packet is currently being transmitted, then our packet's queuing delay is zero. On the other hand, if the traffic is heavy and many other packets are also waiting to be transmitted, the queuing delay will be long. We will see shortly that the number of packets that an arriving packet might expect to find on arrival (informally, the average number of queued packets, which is proportional to the average delay experienced by packets) is a function of the intensity and nature of the traffic arriving to the queue.

Assuming that packets are transmitted in first-come-first-serve manner, as is common in the Internet, our packet can be transmitted once all the packets that have arrived before it have been transmitted. Denote the length of the packet by L bits and denote the transmission rate of the link (from router A to router B) by R bits/sec. The rate R is determined by transmission rate of the link to router B. For example, for a 10 Mbps Ethernet link, the rate is R=10 Mbps; for a 100 Mbps Ethernet link, the rate is R=100 Mbps. The **transmission delay** (also called the store-and-forward delay, as discussed in Section 1.4) is L/R. This is the amount of time required to transmit all of the packet's bits into the link.

Once a bit is pushed onto the link, it needs to propagate to router B. The time required to propagate from the beginning of the link to router B is the **propagation delay**. The bit propagates at the propagation speed of the link. The propagation speed depends on the physical medium of the link (i.e., multimode fiber, twisted-pair copper wire, etc.) and is in the range of

*2*108 meters/sec to 3*108 meters/sec,*

equal to, or a little less than, the speed of light. The propagation delay is the distance between two routers divided by the propagation speed. That is, the propagation delay is d/s, where d is the distance between router A and router B and s is the propagation speed of the link. Once the last bit of the packet propagates to node B, it and all the preceding bits of the packet are stored in router B. The whole

process then continues with router B now performing the forwarding.

Newcomers to the field of computer networking sometimes have difficulty understanding the difference between transmission delay and propagation delay. The difference is subtle but important. The transmission delay is the amount of time required for the router to push out the packet; it is a function of the packet's length and the transmission rate of the link, but has nothing to do with the distance between the two routers. The propagation delay, on the other hand, is the time it takes a bit to propagate from one router to the next; it is a function of the distance between the two routers, but has nothing to do with the packet's length or the transmission rate of the link.

An analogy might clarify the notions of transmission and propagation delay. Consider a highway which has a toll booth every 100 kilometers. You can think of the highway segments between toll booths as links and the toll booths as routers. Suppose that cars travel (i.e., propagate) on the highway at a rate of 100 km/hour (i.e., when a car leaves a toll booth it instantaneously accelerates to 100 km/hour and maintains that speed between toll booths). Suppose that there is a caravan of 10 cars that are traveling together, and that these ten cars follow each other in a fixed order. You can think of each car as a bit and the caravan as a packet. Also suppose that each toll booth services (i.e., transmits) a car at a rate of one car per 12 seconds, and that it is late at night so that the caravan's cars are only cars on the highway. Finally, suppose that whenever the first car of the caravan arrives at a toll booth, it waits at the entrance until the nine other cars have arrived and lined up behind it. (Thus the entire caravan must be "stored" at the toll booth before it can begin to be "forwarded".) The time required for the toll booth to push the entire caravan onto the highway is 10/(5 cars/minute) = 2 minutes. This time is analogous to the transmission delay in a router. The time required for a car to travel from the exit of one toll booth to the next toll booth is 100 Km/(100 km/hour) = 1 hour. This time is analogous to propagation delay. Therefore the time from when the caravan is "stored" in front of a toll booth until the caravan is "stored" in front of the next toll booth is the sum of "transmission delay" and "the propagation delay" - in this example, 62 minutes.

Let's explore this analogy a bit more. What would happen if the toll-booth service time for a caravan were greater than the time for a car to travel between toll booths? For example, suppose cars travel at rate 1000 km/hr and the toll booth services cars at rate one car per minute. Then the traveling delay between toll booths is 6 minutes and the time to serve a caravan is 10 minutes. In this case, the first few cars in the caravan will arrive at the second toll booth before the last cars in caravan leave the first toll booth. This situation also arises in packet-switched networks - the first bits in a packet can arrive at a router while many of the remaining bits in the packet are still waiting to be transmitted by the preceding router.

If we let d_{proc}, d_{queue}, d_{trans} and d_{prop} denote the processing, queuing, transmission and propagation delays, then the total nodal delay is given by

$$d_{\text{nodal}} = dproc + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}} .$$

The contribution of these delay components can vary significantly. For example, d_{prop} can be negligible (e.g., a couple of microseconds) for a link connecting two routers on the same university campus; however, d_{prop} is hundreds of milliseconds for two routers interconnected by a geostationary satellite link, and can be the dominant term in d_{nodal}. Similarly, d_{trans} can be range from negligible to significant. Its contribution is typically negligible for transmission rates of 10 Mbps and higher (e.g., for LANs); however, it can be hundreds of milliseconds for large Internet packets sent over 28.8 kbps modem links. The processing delay, *dproc* , is often negligible; however, it strongly influences a router's maximum throughput, which is the maximum rate at which a router can forward packets.

Queuing Delay

The most complicated and interesting component of nodal delay is the queuing delay d_{queue}. In fact, queuing delay is so important and interesting in computer networking that thousands of papers and numerous of books have been written about it [Bertsekas 1992] [Daigle 1991] [Kleinrock 1975] [Kleinrock 1976] [Ross 1995]! We only give a high-level, intuitive discussion of queuing delay here; the more curious reader may want to browse through some of the books (or even eventually write a Ph.D. thesis on the subject!). Unlike the other three delays (namely, *dproc* , d_{trans} and d_{prop}), the queuing delay can vary from packet to packet. For example, if ten packets arrive to an empty queue at the same time, the first packet transmitted will suffer no queuing delay, while the last packet transmitted will suffer a relatively large queuing delay (while it waits for the other nine packets to be transmitted). Therefore, when characterizing queuing delay, one typically uses statistical measures, such as average queuing delay, variance of queuing delay and the probability that the queuing delay exceeds some specified value.

When is the queuing delay big and when is it insignificant? The answer to this question depends largely on the rate at which traffic arrives to the queue, the transmission rate of the link, and the nature of the arriving traffic, i.e., whether the traffic arrives periodically or whether it arrives in bursts. To gain some insight here, let *a* denote the average rate at which packets arrive to the queue (*a* is units of packets/sec). Recall that *R* is the transmission rate, i.e., it is the rate (in bits/sec) at which bits are pushed out of the queue. Also suppose, for simplicity, that all packets consist of *L* bits. Then the average rate at which bits arrive to the queue is *La* bits/sec. Finally, assume that the queue is very big, so that it can hold essentially an infinite number of bits. The ratio *La/R,* called the **traffic intensity**, often plays an important role in estimating the extent of the queuing delay. If $La/R > 1$, then the average rate at which bits arrive to the queue exceeds the rate at which the bits can be transmitted from the queue. In this unfortunate situation, the queue will tend to increase without bound and the queuing delay will approach infinity! Therefore, one of the golden rules in traffic engineering is: *design your system so that the traffic intensity is no greater than one*.

Now consider the case $La/R =< 1$. Here, the nature of the arriving traffic impacts the queuing delay. For example, if packets arrive periodically, i.e., one packet arrives every *L/R* seconds, then every packet will arrive to an empty queue and there will be no queuing delay. On the other hand, if packets arrive in

bursts but periodically, there can be a significant average queuing delay. For example, suppose N packets arrive at the same time every $(L/R)N$ seconds. Then the first packet transmitted has no queuing delay; the second packet transmitted has a queuing delay of L/R seconds; and more generally, the nth packet transmitted has a queuing delay of $(n-1)L/R$ seconds. We leave it as an exercise for the reader to calculate the average queuing delay in this example.

The two examples described above of periodic arrivals are a bit academic. Typically the arrival process to a queue is *random*, i.e., the arrivals do not follow any pattern; packets are spaced apart by random amounts of time. In this more realistic case, the quantity La/R is not usually sufficient to fully characterize the delay statistics. Nonetheless, it is useful in gaining an intuitive understanding of the extent of the queuing delay. In particular, if traffic intensity is close to zero, then packets are pushed out at a rate much higher than the packet arrival rate; therefore, the average queuing delay will be close to zero. On the other hand, when the traffic intensity is close to 1, there will be intervals of time when the arrival rate exceeds the transmission capacity (due to the burstiness of arrivals), and a queue will form. As the traffic intensity approaches 1, the average queue length gets larger and larger. The qualitative dependence of average queuing delay on the traffic intensity is shown in Figure 1.6-2 below.

One important aspect of Figure 1.6-2 is the fact that as the traffic intensity approaches 1, the average queueing delay increases rapidly. A small percentage increase in the intensity will result in a much larger percentage-wise increase in delay. Perhaps you have experienced this phenomenon on the highway. If you regularly drive on a road that is typically congested, the fact that the road is typically congested means that its traffic intensity is close to 1. If some event causes an even slightly-larger-than-usual amount of traffic, the delays you experience can be huge.

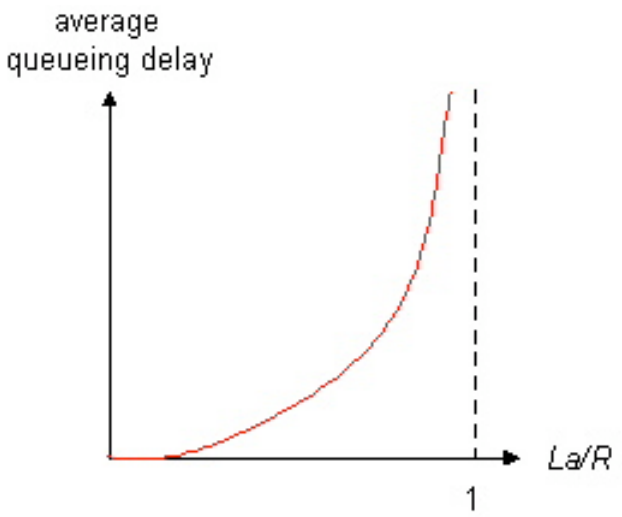

Figure 1.6-2: Dependence of average queuing delay on traffic intensity.

Packet Loss

In our discussions above, we have assumed that the queue is capable of holding an infinite number of packets. In reality a queue preceding a link has finite capacity, although the queuing capacity greatly depends on the switch design and cost. Because the queue capacity is a finite, packet delays do not really approach infinity as the traffic intensity approaches one. Instead, a packet can arrive to find a full queue. With no place to store such a packet, a router will **drop** that packet; that is, the packet will be **lost**. From an end-system viewpoint, this will look like a packet having been transmitted into the network core, but never emerging from the network at the destination. The fraction of lost packets increases as the traffic intensity increases. Therefore, performance at a node is often measured not only in terms of delay, but also in terms of the probability of packet loss. As we shall discuss in the subsequent chapters, a lost packet may be retransmitted on an end-to-end basis, by either the application or by the transport layer protocol.

End-to-End Delay

Our discussion up to this point has been focused on the nodal delay, i.e., the delay at a single router. Let us conclude our discussion by briefly considering the delay from source to destination. To get a handle on this concept, suppose there are *Q-1* routers between the source host and the destination host. Let us also suppose that the network is uncongested (so that queuing delays are negligible), the processing delay at each router and at the source host is dproc, the transmission rate out of each router and out of the source host is *R* bits/sec, and the propagation delay between each pair or routers and between the source host and the first router is d_{prop}. The nodal delays accumulate and give an end-to-end delay,

$$d_{endend} = Q\,(dproc + d_{trans} + d_{prop})\ ,$$

where once again $d_{trans} = L/R$, where *L* is the packet size. We leave it to the reader to generalize this formula to the case of heterogeneous delays at the nodes and to the presence of an average queuing delay at each node.

Return to Table Of Contents

References

[Bertsekas 1992] D. Bertsekas and R. Gallager, *Data Networks, 2nd Edition*, Prentice Hall, Englewood Cliffs, N.J., 1992

[Daigle 1991] J.N. Daigle, *Queuing Theory for Telecommunications*, Addision-Wesley, Reading Massachusetts, 1991.

[Kleinrock 1975] L. Kleinrock, *Queuing Systems, Volume 1*, John Wiley, New York, 1975.

[Kleinrock 1976] L. Kleinrock, *Queuing Systems, Volume 2*, John Wiley, New York, 1976.

[Ross 1995] K.W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks*, Springer, Berlin, 1995.

1.7 Protocol Layers and Their Service Models

From our discussion thus far, it is apparent that the Internet is an *extremely* complicated system. We have seen that there are many "pieces" to the Internet: numerous applications and protocols, various types of end systems and connections between end systems, routers, and various types of link-level media. Given this enormous complexity, is there any hope of *organizing* network architecture, or at least our discussion of network architecture? Fortunately, the answers to both questions is "yes."

Before attempting to organize our thoughts on Internet architecture, let's look for a human analogy. Actually, we deal with complex systems all the time in our every day life. Imagine if someone asked *you* to describe, for example, the airline system. How would you find the structure to describe this complex system that has ticketing agents, baggage checkers, gate personnel, pilots and airplanes, air traffic control, and a worldwide system for routing airplanes? One way to describe this system might be to describe the series of actions you take (or others take for you) when you fly on an airline. You purchase your ticket, check your bags, go to the gate and eventually get loaded onto the plane. The plane takes off and is routed to its destination. After your plane lands, you de-plane at the gate and claim your bags. If the trip was bad, you complain about the flight to the ticket agent (getting nothing for your effort). This scenario is shown in Figure 1.7-1.

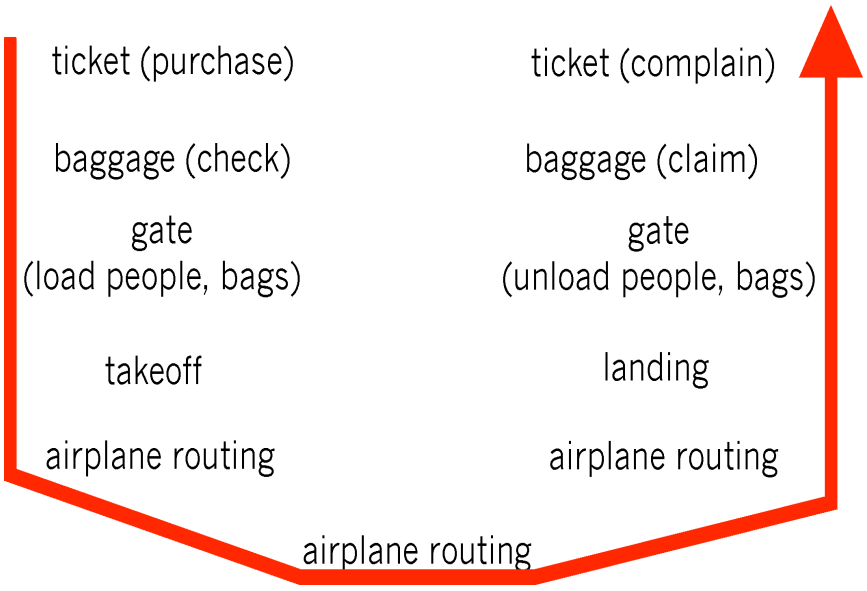

Figure 1.7-1: Taking an airplane trip: actions

Already, we can see some analogies here with computer networking: you are being shipped from source to destination by the airline; a packet is shipped from source host to destination host in the Internet. But this is not quite the analogy we are after. We are looking for some *structure* in Figure 1.7-1. Looking at Figure 1.7-1, we note that there is a ticketing function at each end; there is also a baggage function for already ticketed passengers, and a gate function for already-ticketed and already-baggage-checked passengers. For passengers who have made it through the gate (i.e., passengers who are already ticketed, baggage-checked, and through the gate), there is a takeoff and landing function, and while in flight, there is an airplane routing function. This suggests that we can look at the functionality in Figure 1.7-1 in a *horizontal* manner, as shown in Figure 1.7-2.

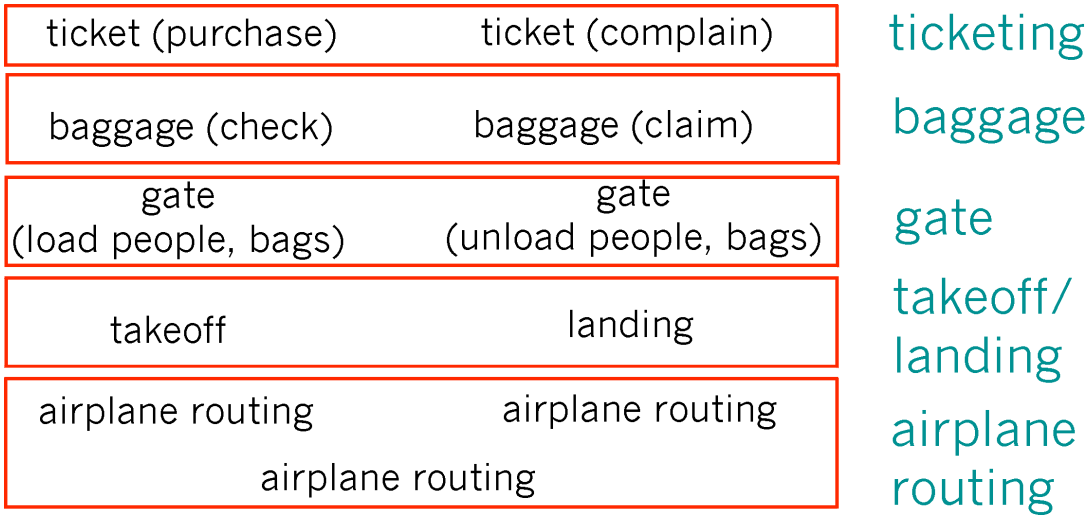

Figure 1.7-2: horizontal "layering" of airline functionality

Figure 1.7-2 has divided the airline functionality into *layers*, providing a framework in which we can discuss airline travel. Now, when we want to describe a part of airline travel we can talk about a specific, well-defined component of airline travel. For example, when we discuss gate functionality, we know we are discussing functionality that sits "below" baggage handling, and "above" takeoff and landing. We note that each layer, combined with the layers below it, implement some functionality, some *service*. At the ticketing layer and below, airline-counter-to-airline-counter transfer of a person is accomplished. At the baggage layer and below, baggage-check-to-baggage-claim transfer of a person and their bags in accomplished. Note that the baggage layer provides this service only to an already ticketed person. At the gate layer, departure-gate-to-arrival-gate transfer of a person and their bags is accomplished. At the takeoff/landing layer, runway-to-runway transfer of a person (actually, many people) and their bags, is accomplished. Each layer provides its functionality (service) by *(i)* performing certain actions within that layer (e.g., at the gate layer, loading and unloading people from an airplane) and by *(ii)* using the services of the layer directly below it (e.g., in the gate layer, using the runway-to-runway passenger transfer service of the takeoff/landing layer).

As noted above, a layered architecture allows us to discuss a well-defined, specific part of a large and complex system. This itself is of considerable value. When a system has a layered structure it is also much easier to change the *implementation* of the service provided by the layer. As long as the layer provides the same service to the layer above it, and uses the same services from the layer below it, the remainder of the system remains unchanged when a layer's implementation is changed. (Note that changing the implementation of a service is very different from changing the service itself!) For example, if the gate functions were changed (e.g., to have people board and disembark by height), the remainder of the airline system would remain unchanged since the gate layer still provides the same function (loading and unloading people); it simply implements that function in a different manner after the change. For large and complex systems that are constantly being updated, the ability to change the implementation of a service without affecting other components of the system is another important advantage of layering.

But enough with airlines. Let's now turn our attention to network protocols. To reduce design complexity, network designers organize protocols -- and the network hardware and software that implements the protocols -- in **layers**. With a layered protocol architecture, each protocol belongs to one of the layers. It's important to realize that a protocol in layer *n* is *distributed* among the network entities (including end systems and packet switches) that implement that protocol, just as the functions in our layered airline architecture were distributed between the departing and arriving airports. In other words, there's a "piece" of

layer *n* in each of the network entities. These "pieces" communicate with each other by exchanging layer-n messages. These messages are called layer-n protocol data units, or more commonly ***n-PDUs***. The contents and format of an *n-PDU*, as well as the manner in which the *n-PDUs* are exchanged among the network elements, are defined by a layer-n protocol. When taken together, the protocols of the various layers are called the **protocol stack**.

When layer *n* of Host A sends an *n-PDU* to layer *n* of Host B, layer *n* of Host A *passes* the *n-PDU* to layer *n*-1 and then lets layer *n*-1 deliver the *n-PDU* to layer *n* of B; thus layer *n* is said to *rely* on layer *n*-1 to deliver its n-PDU to the destination. A key concept is that of the **service model** of a layer. Layer *n*-1 is said to offer **services** to layer *n*. For example, layer *n*-1 might guarantee that the *n-PDU* will arrive without error at layer *n* in the destination within one second, or it might only guarantee that the *n-PDU* will eventually arrive at the destination without any assurances about error.

The concept of protocol layering is a fairly abstract and is sometimes difficult to grasp at first. This concept will become clear as we study the Internet layers and their constituent protocols in greater detail. But let use now try to shed some insight on protocol layering and protocol stacks with an example. Consider a network which organizes its communication protocols in four layers. Because there are four layers, there are four types of PDUs: 1-PDUs, 2-PDUs, 3-PDUs and 4-PDUs. As shown in Figure 1.7-3, the application, operating at the highest layer, layer 4, creates a message, M. Any message created at this highest layer is a 4-PDU. The message M itself may consist of many different fields (in much the same way as a structure or record in a programming language may contain different fields); it is up to the application to define and interpret the fields in the message. The fields might contain the name of the sender, a code indicating the type of the message, and some additional data.

Within the source host, the contents of the entire message M is then "passed" down the protocol stack to layer 3. In the example in Figure 1.7-3, layer 3 in the source host divides a 4-PDU, M, into two parts, M_1 and M_2. The layer 3 in the source host then adds to M_1 and M_2, so-called **headers**, to create two layer 3 PDUs. Headers contain the additional information needed by the sending and receiving sides of layer 3 to implement the service that layer 3 provides to layer 4. The procedure continues in the source, adding more header at each layer, until the 1-PDUs are created. The 1-PDUs are sent out of the source host onto a physical link. At the other end, the destination host receives 1-PDUs and directs them up the protocol stack. At each layer, the corresponding header is removed. Finally, M is reassembled from M_1 and M_2 and then passed on to the application.

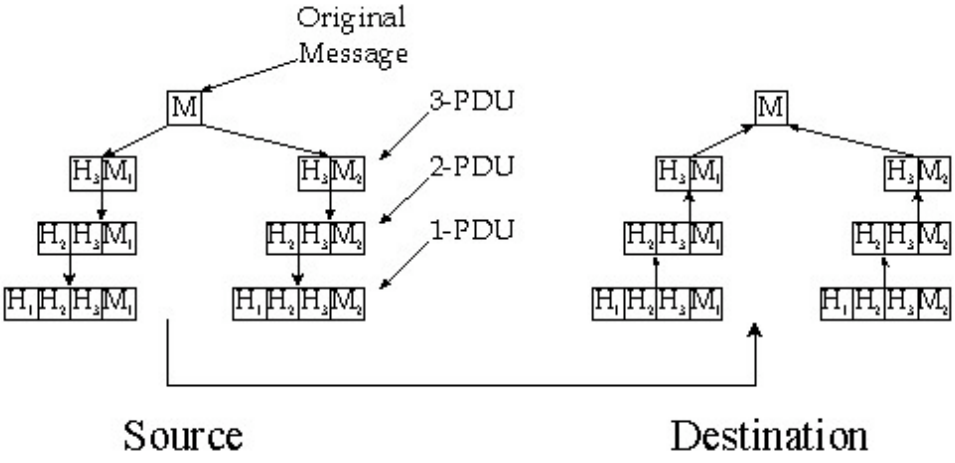

Figure 1.7-3: different PDU's at different layers in the protocol architecture

Note that in Figure 1.7-3, layer *n* uses the services of layer *n*-1. For example, once layer 4 creates the message M, it passes the message down to layer 3 and relies on layer 3 to deliver the message to layer 4 at the destination.

Interesting enough, this notion of relying on lower layer services is prevalent in many other forms of communication. For

example, consider ordinary postal mail. When you write a letter, you include envelope information such as the destination address and the return address with the letter. The letter along with the address information can be considered a PDU at the highest layer of the protocol stack. You then drop the PDU in a mailbox. At this point, the letter is out of your hands. The postal service may then add some of its own internal information onto your letter, essentially adding a header to your letter. For example, in the United States a barcode is often printed on your letter.

Once you drop your envelope into a mailbox, you *rely* on the services of the postal service to deliver the letter to the correct destination in a timely manner. For example, you don't worry about whether a postal truck will break down while carrying the letter. Instead the postal service takes care of this, presumably with well-defined plans to recover from such failures. Furthermore, within the postal service itself there are layers, and the protocols at one layer rely on and use the services of the layer below.

In order for one layer to interoperate with the layer below it, the interfaces between the two layers must be precisely defined. Standards bodies define precisely the interfaces between adjacent layers (e.g., the format of the PDUs passed between the layers) and permit the developers of networking software and hardware to implement the interior of the layers as they please. Therefore, if a new and improved implementation of a layer is released, the new implementation can replace the old implementation and, in theory, the layers will continue to interoperate.

In a computer network, each layer may perform one or more of the following generic set of tasks:

- **Error control**, which makes the logical channel between the layers in two peer network elements more reliable.
- **Flow control**, which avoids overwhelming a slower peer with PDUs.
- **Segmentation and Reassembly**, which at the transmitting side divides large data chunks into smaller pieces; and at the receiving side reassembles the smaller pieces into the original large chunk.
- **Multiplexing**, which allows several higher-level sessions to share a single lower-level connection.
- **Connection setup**, which provides the handshaking with a peer.

Protocol layering has conceptual and structural advantages. We mention, however, that some researchers and networking engineers are vehemently opposed to layering [Wakeman 1992]. One potential drawback of layering is that one layer may duplicate lower-layer functionality. For example, many protocol stacks provide error recovery on both a link basis and an end-to-end basis. A second potential drawback is that functionality at one layer may need information (e.g., a timestamp value) that is present only in another layer; this violates the goal of separation of layers.

1.7.1 The Internet Protocol Stack

The Internet stack consists of five layers: the physical, data link, network, transport and application layers. Rather than use the cumbersome terminology PDU-n for each of the five layers, we instead give special names to the PDUs in four of the five layers: frame, datagram, segment, and message. We avoid naming a data unit for the physical layer, as no name is commonly used at this layer. The Internet stack and the corresponding PDU names are illustrated in Figure 1.7-4.

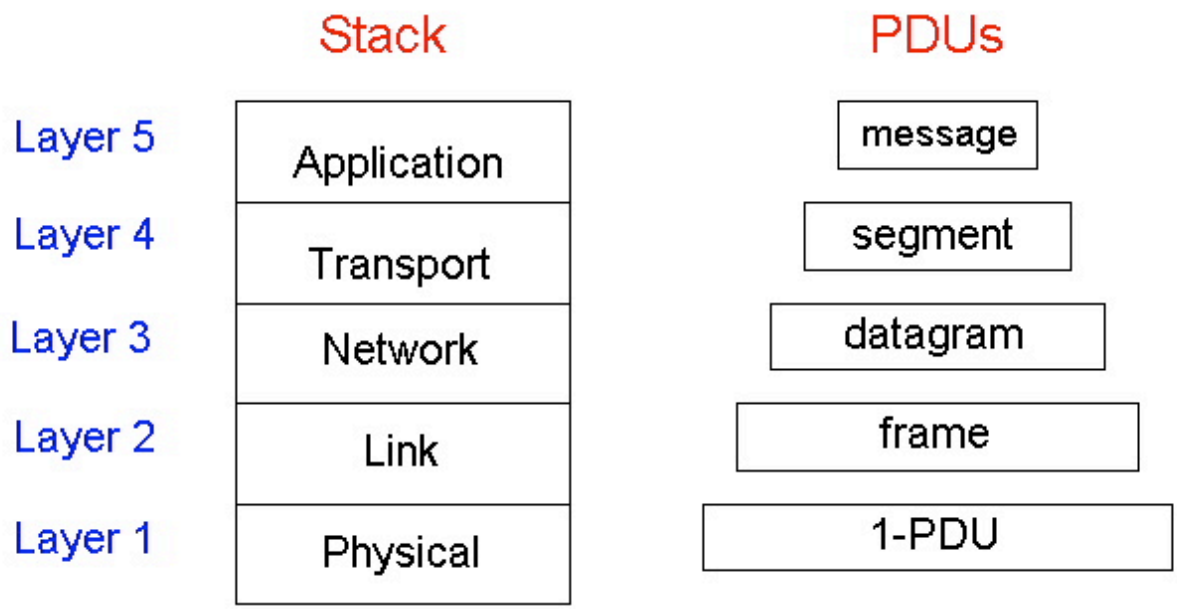

Figure 1.7-4: The protocol stack, and protocol data units

A protocol layer can be implemented in software, in hardware, or using a combination of the two. Application-layer protocols -- such as HTTP and SMTP -- are almost always implemented in software in the end systems; so are transport layer protocols. Because the physical layer and data link layers are responsible for handling communication over a specific link, they are typically implemented in a network interface card (e.g., Ethernet or ATM interface cards) associated with a given link. The network layer is often a mixed implementation of hardware and software.

We now summarize the Internet layers and the services they provide:

- **Application layer:** The application layer is responsible for supporting network applications. The application layer includes many protocols, including HTTP to support the Web, SMTP to support electronic mail, and FTP to support file transfer. We shall see in Chapter 2 that it is very easy to create our own new application-layer protocols.

- **Transport layer:** The transport layer is responsible for transporting application-layer messages between the client and server sides of an application. In the Internet there are two transport protocols, TCP and UDP, either of which can transport application-layer messages. TCP provides a connection-oriented service to its applications. This service includes guaranteed delivery of application-layer messages to the destination and flow control (i.e., sender/receiver speed matching). TCP also segments long messages into shorter segments and provides a congestion control mechanism, so that a source throttles its transmission rate when the network is congested. The UDP protocol provides its applications a connnectionless service, which (as we saw in section 1.3) is very much a no-frills service.

- **Network layer:** The network layer is responsible for routing datagrams from one host to another. The Internet's network layer has two principle components. First it has a protocol that defines the fields in the IP datagram as well as how the end systems and routers act on these fields. This protocol is the celebrated IP protocol.. There is only one IP protocol, and all Internet components that have a network layer must run the IP protocol. The Internet's network layer also contains routing protocols that determine the routes that datagrams take between sources and destinations. The Internet has many routing protocols. As we saw in section 1.4, the Internet is network of networks and within a network, the

network administrator can run any routing protocol desired. Although the network layer contains both the IP protocol and numerous routing protocols, it is often simply referred to as the IP layer, reflecting that fact that IP is the glue that binds the Internet together.

The Internet transport layer protocols (TCP and UDP) in a source host passes a transport layer segment and a destination address to the IP layer, just as you give the postal service a letter with a destination address. The IP layer then provides the service of routing the segment to its destination. When the packet arrives at the destination, IP passes the segment to the transport layer within the destination.

- **Link layer:** The network layer routes a packet through a series of packet switches (i.e., routers) between the source and destination. To move a packet from one node (host or packet switch) to the next node in the route, the network layer must rely on the services of the link layer. In particular, at each node IP passes the datagram to the link layer, which delivers the datagram to the next node along the route. At this next node, the link layer passes the IP datagram to the network layer. The process is analogous to the postal worker at a mailing center who puts a letter into a plane, which will deliver the letter to the next postal center along the route. The services provided at the link layer depend on the specific link-layer protocol that is employed over the link. For example, some protocols provide reliable delivery on a link basis, i.e., from transmitting node, over one link, to receiving node. Note that this reliable delivery service is different from the reliable delivery service of TCP, which provides reliable delivery from one end system to another. Examples of link layers include Ethernet and PPP; in some contexts, ATM and frame relay can be considered link layers. As datagrams typically need to traverse several links to travel from source to destination, a datagram may be handled by different link-layer protocols at different links along its route. For example, a datagram may be handled by Ethernet on one link and then PPP on the next link. IP will receive a different service from each of the different link-layer protocols.

- **Physical layer:** While the job of the link layer is to move entire frames from one network element to an adjacent network element, the job of the physical layer is to move the *individual bits* within the frame from one node to the next. The protocols in this layer are again link dependent, and further depend on the actual transmission medium of the link (e. g., twisted-pair copper wire, single mode fiber optics). For example, Ethernet has many physical layer protocols: one for twisted-pair copper wire, another for coaxial cable, another for fiber, etc. In each case, a bit is moved across the link in a different way.

If you examine the Table Of Contents, you will see that we have roughly organized this book using the layers of the Internet protocol stack. We take a **top-down approach**, first covering the application layer and then preceding downwards.

1.7.2 Network Entities and Layers

The most important network entities are end systems and packet switches. As we shall discuss later in this book, there are two two types of packet switches: routers and bridges. We presented an overview of routers in the earlier sections. Bridges will be discussed in detail in Chapter 5 whereas routers will be covered in more detail in Chapter 4. Similar to end systems, routers and bridges organize the networking hardware and software into layers. But routers and bridges do not implement *all* of the layers in the protocol stack; they typically only implement the bottom layers. As shown in Figure 1.7-5, bridges implement layers 1 and 2; routers implement layers 1 through 3. This means, for example, that Internet routers are capable of implementing the IP protocol (a layer 3 protocol), while bridges are not. We will see later that while bridges do not recognize IP addresses, they are capable of recognizing layer 2 addresses, such as Ethernet addresses. Note that hosts implement all five layers; this is consistent with the view that the Internet architecture puts much of its complexity at the "edges" of the network. Repeaters, yet another kind of network entity to be discussed in Chapter 5, implement only layer 1 functionality.

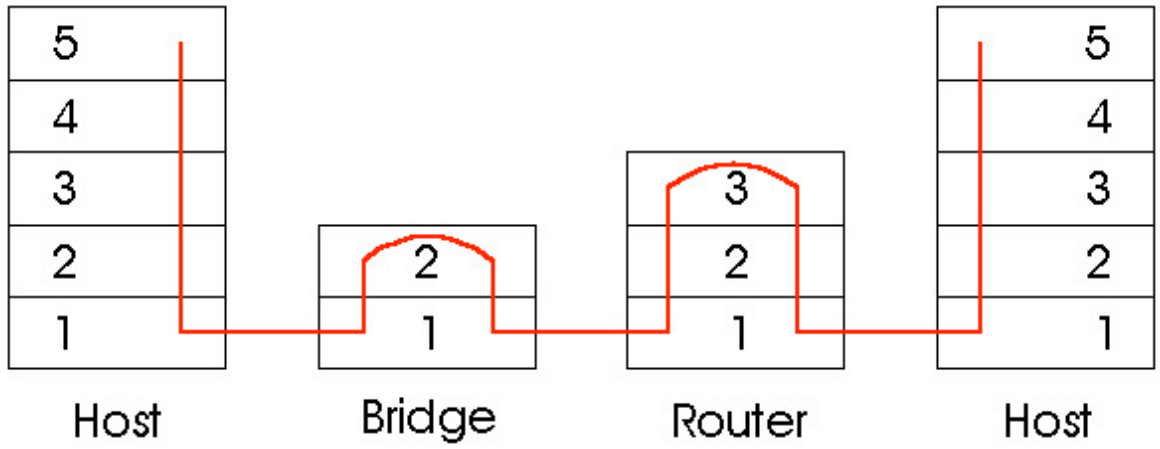

Figure 1.7-5: Hosts, routers and bridges - each contain a different set of layers, reflecting their differences in functionality

References

[Wakeman 1992] Ian Wakeman, Jon Crowcroft, Zheng Wang, and Dejan Sirovica, "Layering considered harmful," *IEEE Network,* January 1992, p. 7.

Return to Table Of Contents

1.8 Internet Backbones, NAPs and ISPs

Our discussion of layering in the previous section has perhaps given the impression that the Internet is a carefully organized and highly intertwined structure. This is certainly true in the sense that all of the network entities (end systems, routers and bridges) use a common set of protocols, enabling the entities to communicate with each other. If one wanted to change, remove, or add a protocol, one would have to follow a long and arduous procedure to get approval from the IETF, which will (among other things) make sure that the changes are consistent with the highly intertwined structure. However, from a topological perspective, to many people the Internet seems to be growing in a chaotic manner, with new sections, branches and wings popping up in random places on a daily basis. Indeed, unlike the protocols, the Internet's topology can grow and evolve without approval from a central authority. Let us now try to a grip on the seemingly nebulous Internet topology.

As we mentioned at the beginning of this chapter, the topology of the Internet is loosely hierarchical. Roughly speaking, from bottom-to-top the hierarchy consists of end systems (PCs, workstations, etc.) connected to local Internet Service Providers (ISPs). The local ISPs are in turn connected to regional ISPs, which are in turn connected to national and international ISPs. The national and international ISPs are connected together at the highest tier in the hierarchy. New tiers and branches can be added just as a new piece of Lego can be attached to an existing Lego construction.

In this section we describe the topology of the Internet in the United States as of 1999. Let's begin at the top of the hierarchy and work our way down. Residing at the very top of the hierarchy are the national ISPs, which are called **National Backbone Provider (NBPs)**. The NBPs form independent backbone networks that span North America (and typically abroad as well). Just as there are multiple long-distance telephone companies in the USA, there are multiple NBPs that compete with each other for traffic and customers. The existing NBPs include internetMCI, SprintLink, PSINet, UUNet Technologies, and AGIS. The NBPs typically have high-bandwidth transmission links, with bandwidths ranging from 1.5 Mbps to 622 Mbps and higher. Each **NBP** also has numerous hubs which interconnect its links and at which **regional ISPs** can tap into the NBP.

The NBPs themselves must be interconnected to each other. To see this, suppose one regional ISP, say MidWestnet, is connected to the MCI NBP and another regional ISP, say EastCoastnet, is connected to Sprint's NBP. How can traffic be sent from MidWestnet to EastCoastnet? The solution is to introduce switching centers, called **Network Access Points (NAPs)**, which interconnect the NBPs, thereby allowing each regional ISP to pass traffic to any other regional ISP. To keep us all confused, some of the NAPs are not referred to as NAPs but instead as MAEs (Metropolitan Area Exchanges). In the United States, many of the NAPs are run by RBOCs (Regional Bell Operating Companies); for example, PacBell has a NAP in San Francisco and Ameritech has a NAP in Chicago. For a list of major NBP's (those connected into at least three MAPs/MAE's), see [Haynal 99].

Because the NAPs relay and switch tremendous volumes of Internet traffic, they are typically in themselves complex high-speed switching networks concentrated in a small geographical area (for example, a single building). Often the NAPs use high-speed ATM switching technology in the heart of the NAP, with IP riding on top of ATM. (We provide a brief introduction to ATM at the end of this chapter, and discuss IP-over-ATM in Chapter 5) Figure 1.8-1 illustrates PacBell's San Francisco NAP, The details of Figure 1.8-1 are unimportant for us now; it is worthwhile to note, however, that the NBP hubs can themselves be complex data networks.

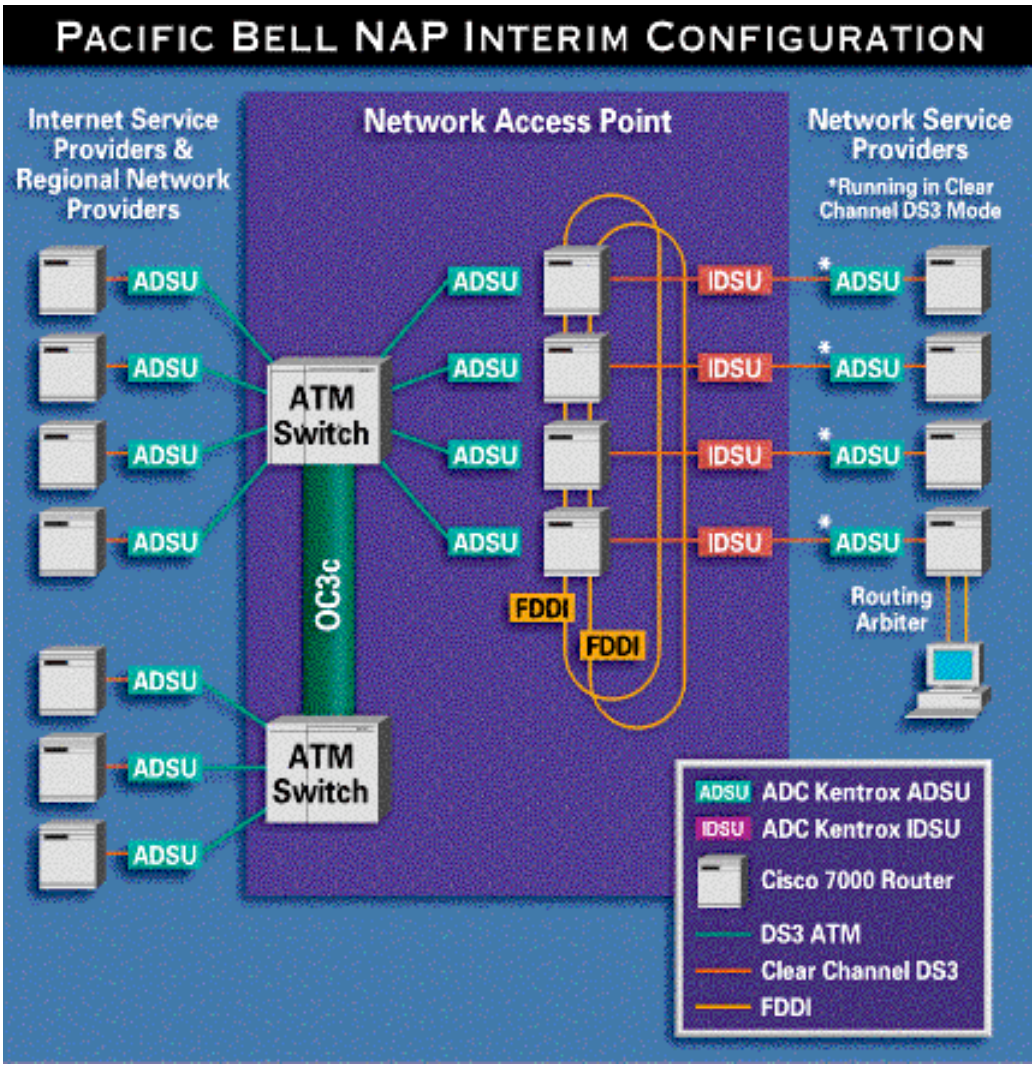

Figure 1.8-1: The PacBell NAP Architecture (courtesy of the Pacific Bell Web site).

The astute reader may have noticed that ATM technology, which uses virtual circuits, can be found at certain places within the Internet. But earlier we said that the "Internet is a datagram network and does not use virtual circuits". We admit now that this statement stretches the truth a little bit . We made this

statement because it helps the reader to see the forest through the trees by not having the main issues obscured. The truth is that there are virtual circuits in the Internet, but they are in localized pockets of the Internet and they are buried deep down in the protocol stack, typically at layer 2. If you find this confusing, just pretend for now that the Internet does not employ any technology that uses virtual circuits. This is not too far from the truth.

Running an NBP is not cheap. In June 1996, the cost of leasing 45 Mbps fiber optics from coast-to-coast, as well as the additional hardware required, was approximately $150,000 per month. And the fees that an NBP pays the NAPs to connect to the NAPs can exceed $300,000 annually. NBPs and NAPs also have significant capital costs in equipment for high-speed networking. An NBP earns money by charging a monthly fee to the regional ISPs that connect to it. The fee that an NBP charges to a regional ISP typically depends on the bandwidth of the connection between the regional ISP and the NBP; clearly a 1.5 Mbps connection would be charged less than a 45 Mbps connection. Once the fixed-bandwidth connection is in place, the regional ISP can pump and receive as much data as it pleases, up to the bandwidth of the connection, at no additional cost. If an NBP has significant revenues from the regional ISPs that connect to it, it may be able to cover the high capital and monthly costs of setting up and maintaining an NBP.

A regional ISP is also a complex network, consisting of routers and transmission links with rates ranging from 64 Kbps upward. A regional ISP typically taps into an NBP (at an NBP hub), but it can also tap directly into an NAP, in which case the regional NBP pays a monthly fee to a NAP instead of to a NBP. A regional ISP can also tap into the Internet backbone at two or more distinct points (for example, at an NBP hub or at a NAP). How does a regional ISP cover its costs? To answer this question, let's jump to the bottom of the hierarchy.

End systems gain access to the Internet by connecting to a local ISP. Universities and corporations can act as local ISPs, but backbone service providers can also serve as a local ISP. Many local ISPs are small "mom and pop" companies, however. A popular WWW site known simple as "The List" contains link to nearly 8000 local, regional, and backbone ISPs [List 1999]. The local ISPs tap into one of the regional ISPs in its region. Analogous to the fee structure between the regional ISP and the NBP, the local ISP pays a monthly fee to its regional ISP which depends on the bandwidth of the connection. Finally, the local ISP charges its customers (typically) a flat, monthly fee for Internet access: the higher the transmission rate of the connection, the higher the monthly fee.

We conclude this section by mentioning that anyone of us can become a local ISP as soon as we have an Internet connection. All we need to do is purchase the necessary equipment (for example, router and modem pool) that is needed to allow other users to connect to our so-called "point of presence." Thus, new tiers and branches can be added to the Internet topology just as a new piece of Lego can be attached to an existing Lego construction.

Return to Table Of Contents

References

[Haynal 99] R. Haynal, "Internet Backbones," http://navigators.com/isp.html
[List 1999] "The List: The Definitive ISP Buyer's Guide," http://thelist.internet.com/
