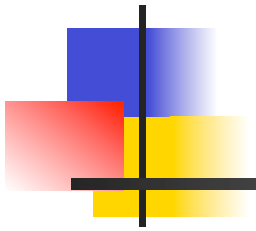


Cap. 2 – O nível aplicação

(2ª parte)

Nota prévia

A estrutura da apresentação é semelhante e utiliza algumas das figuras, textos e outros materiais do livro de base do curso

James F. Kurose and Keith W. Ross, "Computer Networking - A Top-Down Approach Featuring the Internet," Addison Wesley Longman, Inc., 3rd Edition, 2005

Organização do capítulo

- **Aplicações em rede ou distribuídas**
- **Conceitos de base, paradigmas e tipos de transportes**
- **Protocolo HTTP (Web)**
- **O DNS ("Domain Name System")**
- **Protocolo SMTP — Correio electrónico**
- **Transferência de ficheiros - Protocolo FTP e sistemas P2P**
- **Os protocolos RTP e SIP**

DNS - Domain Name System

O serviço de designação da Internet

Objectivos do DNS

- **No início dos anos 80 a Internet tinha algumas centenas de hosts pelo que para a designação dos mesmos bastava um ficheiro ("hosts.txt" mantido pelo NIC - *Network Information Center*) que era copiado periodicamente**

- **Com o aumento do número de *hosts* tal método tornou-se inviável**

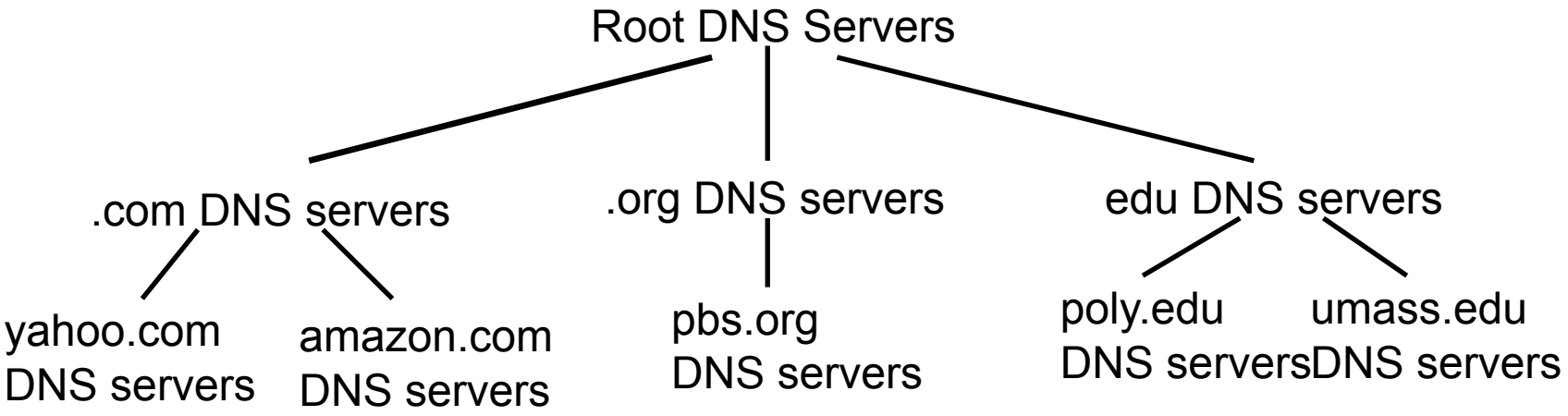
- **Introduziu-se então o DNS que é uma base de dados distribuída, replicada e hierárquica de registo de nomes e atributos de objectos (*hosts*, ...). Tal como hoje o conhecemos está definido pelos RFCs 1034 e 1035 de 1987**

- **O DNS tem uma larga escala, disponibilidade e eficiência**

O que é o DNS ?

- ***Base de dados distribuída*** **implementada por uma hierarquia de servidores de nomes**
- ***Protocolo do nível aplicacional*** **que permite a tradução de nomes de *hosts, routers*, etc. em endereços IP (para além do endereço é possível conhecer outros atributos associados a um nome)**
  - **Nota: trata-se de uma função essencial implementada ao nível aplicacional**
  - **Mais uma vez a complexidade é tratada pela periferia.**

Uma base de dados distribuída e hierárquica



Se o cliente quer conhecer o IP de www.amazon.com:

- **O cliente contacta um *root server* para encontrar o servidor de *.com***
- **O cliente contacta o servidor de *.com* para obter o servidor de *amazon.com***
- **O cliente contacta o servidor de *amazon.com* para obter o endereço IP de *www.amazon.com***

TLDs — Top Level Domains

O DNS define um Domain Name Space hierarquico organizado em Top Level Domains (TLDs) e Second Level Domains (SLDs). OS TLDs dividem-se em:

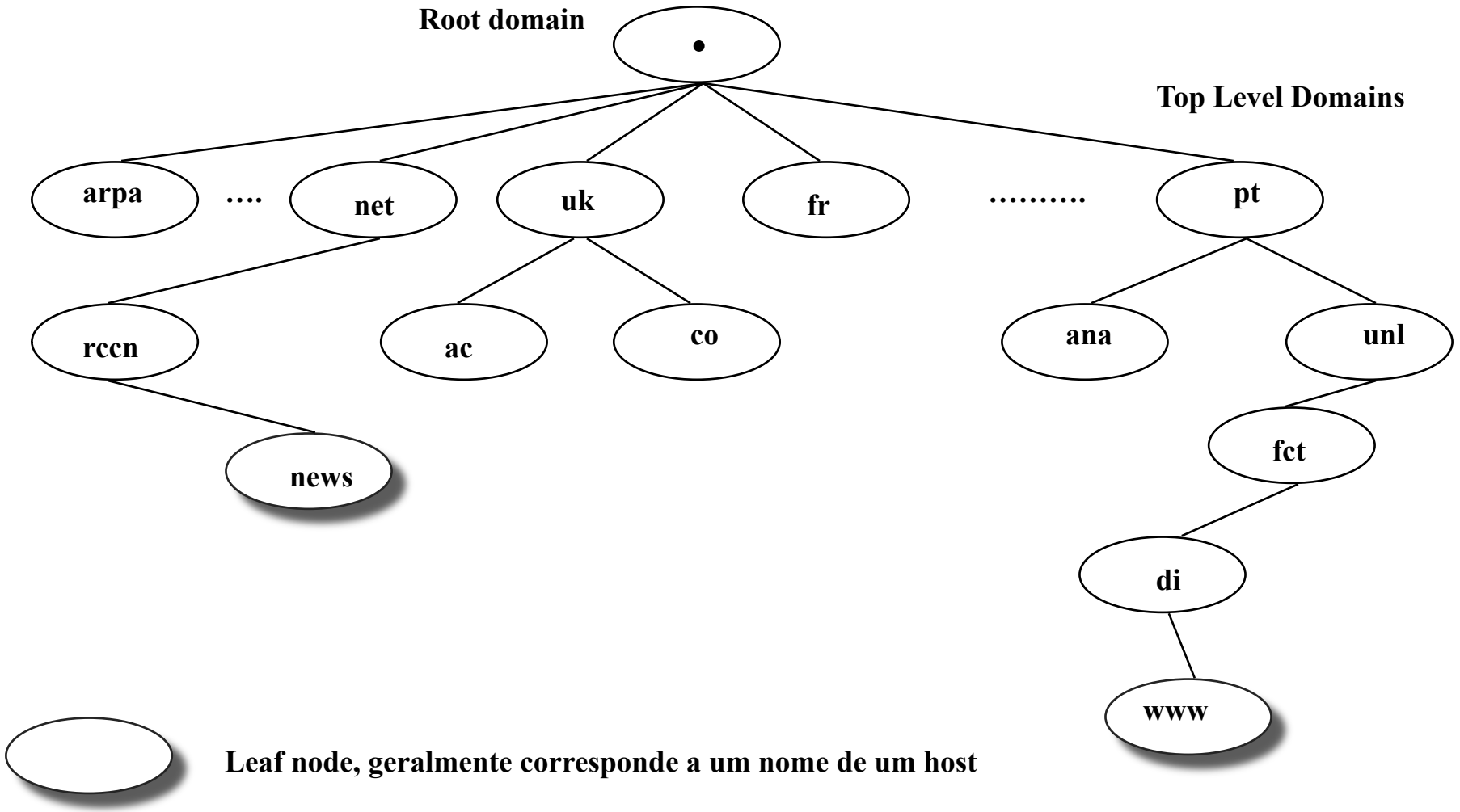
*** Generic Top Level Domains**

COM
EDU
GOV
NET
ORG
MIL
…..

*** National Top Level Domains (ISO 3166 Country Codes)**

AU	**Austrália**
BR	**Brasil**
…	
PT	**Portugal**
…	

A árvore dos domínios



Sintaxe dos nomes DNS

São construídos de baixo para cima

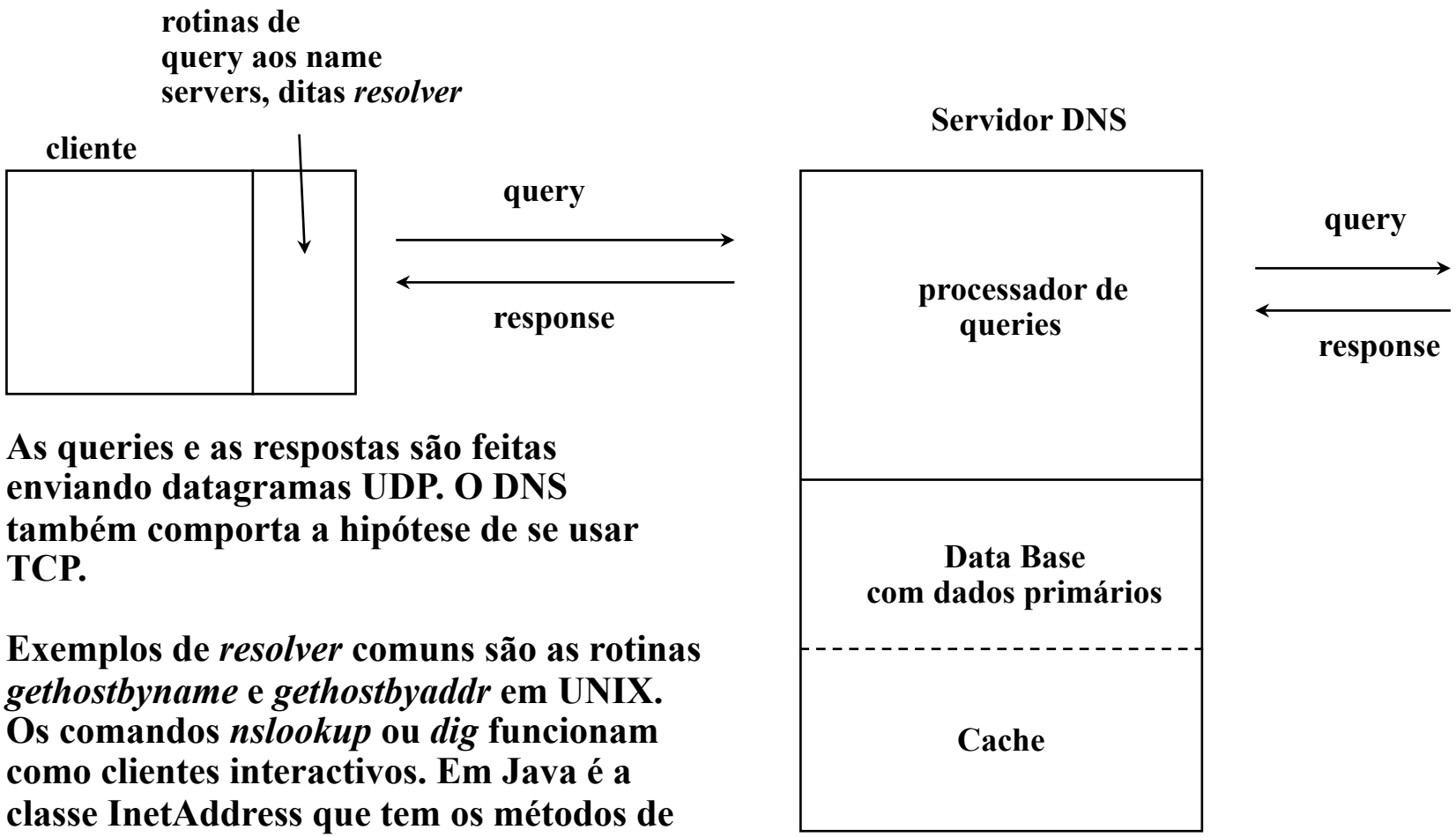
Um "*fully qualified domain name*" termina sempre num ponto (que se pode omitir quando não há dúvidas de interpretação)

Exemplo:

www . di . fct . unl . pt .

Nota: os nomes são "case insensitive"; cada componente pode ter até 63 caracteres e um nome pode ter até 255 caracteres

O DNS está estruturado em cliente / servidor



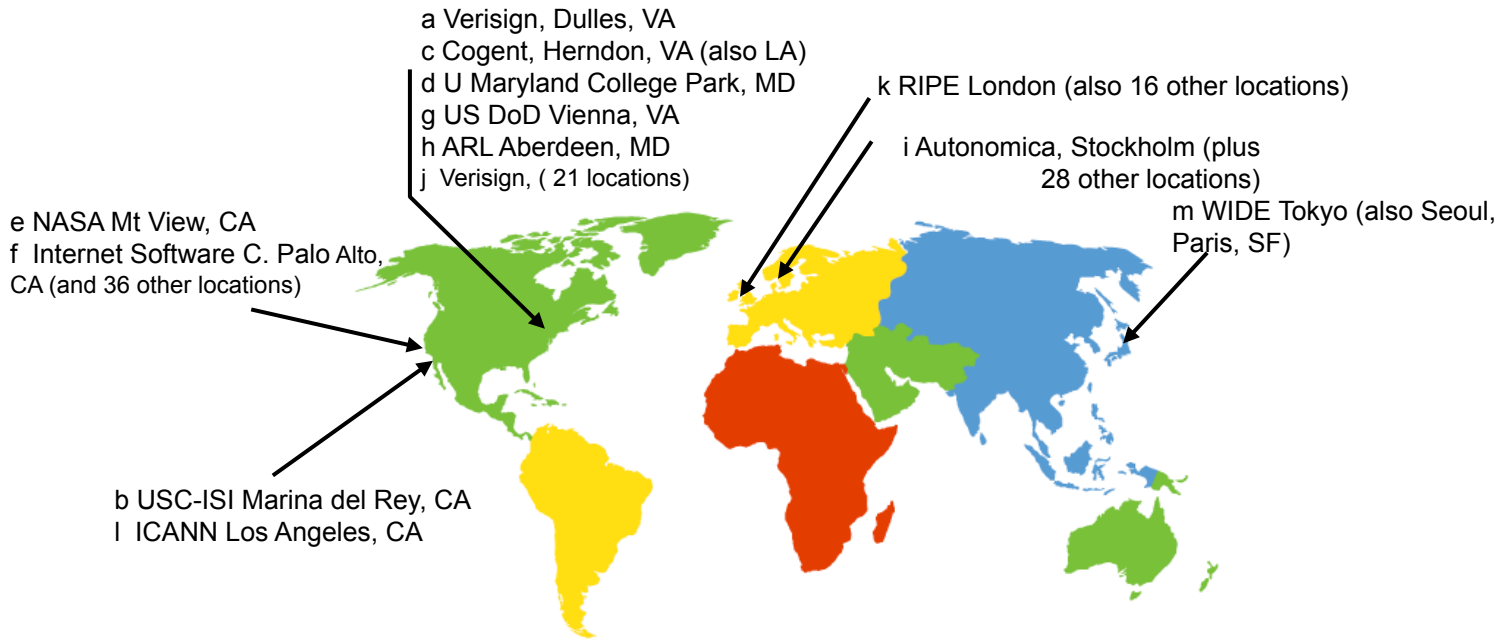
As queries e as respostas são feitas enviando datagramas UDP. O DNS também comporta a hipótese de se usar TCP.

Exemplos de *resolver* comuns são as rotinas *gethostbyname* e *gethostbyaddr* em UNIX. Os comandos *nslookup* ou *dig* funcionam como clientes interactivos. Em Java é a classe InetAddress que tem os métodos de interrogação do DNS.

Servidores DNS

- **Porque não centralizar o DNS?**
  - Concentração do tráfego
  - Ponto central de falha
  - Base de dados centralizada distante
  - Manutenção difícil ou impossível
  - Nenhum servidor único conhece todos os nomes pois tal não escalaria
- **Servidores locais:**
  - Cada ISP, instituição, etc. tem vários servidores locais que são usados directamente pelos utilizadores
  - as *queries* DNS dos utilizadores são dirigidas a estes servidores
- **Servidores com autoridade:**
  - Conhecem os dados verdadeiros sobre um nome
  - Os outros servidores podem fazer *caching* desses dados
- **Servidores ROOT e TLD:**
  - São servidores autoritários com papéis especiais

Root Name Servers (em 2004)



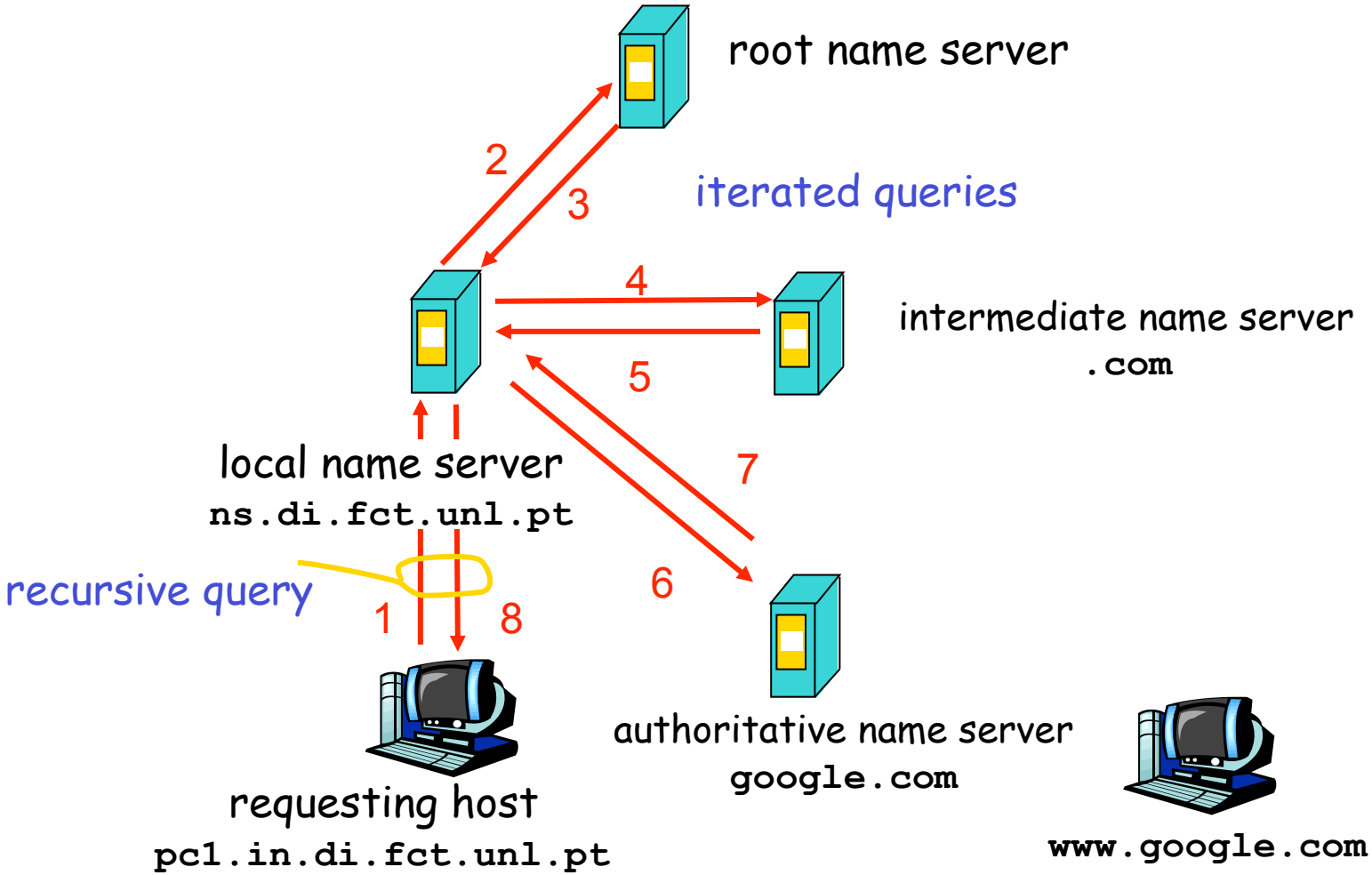
13 root name servers worldwide

Os *Root Name Servers* são servidores com autoridade sobre a zona "."

Local Name Server

- Não pertence necessariamente à hierarquia (pode ser um "caching only server")
- Cada ISP e cada instituição grande tem pelo menos um
  - Geralmente também designado "*default name servers*"
- Quando um host local realiza uma *query*, esta é enviada para este servidor
  - Actua como proxy do serviço

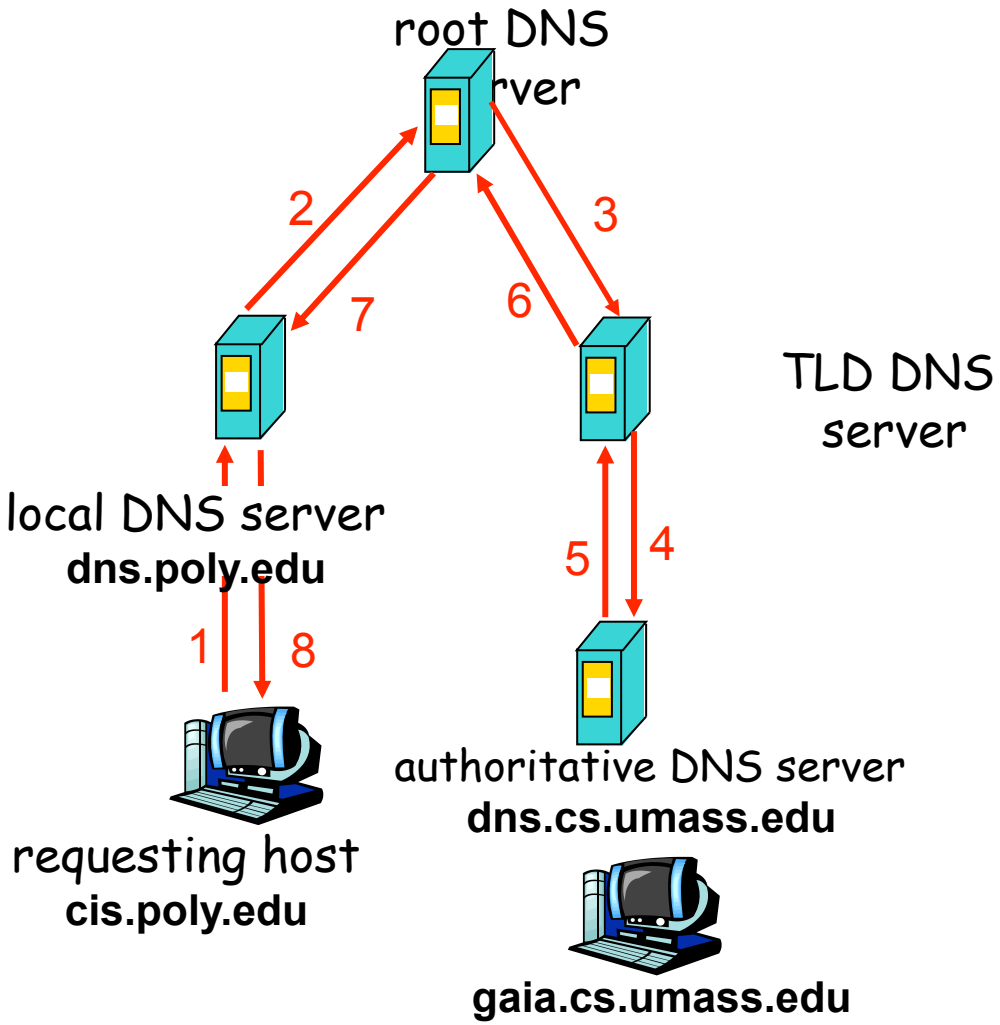
Pedidos recursivos e iterativos



Exemplo de resolução de um nome DNS

Pedido recursivo:

- **O protocolo é executado completamente por cada servidor sem respostas incompletas ao cliente**
- **Cria alguma complexidade suplementar nos servidores ? Sim !**
- **Por essa razão só é aceite pelos local *name servers***



O DNS baseia-se em caching generalizado

- **Todos os servidores localizam os servidores de root e fazem caching dos seus endereços**
- **Todos os servidores quando obtêm uma resposta, mantêm-na em cache e dessa forma respondem imediatamente se aparecer um pedido semelhante**
- **Desta forma, os endereços dos servidores TLD estão sempre em cache**
- **Uma entrada é mantida na cache até um limite de tempo controlado pelo administrador do servidor responsável pelo nome cached através do atributo TTL**

Registos DNS (DNS Resource Records)

DNS: db distribuída com registos (RR)

Formato de um RR: **(name, type, value, ttl)**

- Type=A
  - **O nome é um** hostname
  - **O valor é um endereço IP do host**
- Type=NS
  - **O nome é um domínio** (e.g. foo.com)
  - **O valor é o hostname de um servidor do domínio**
- Type=CNAME
  - **O nome** é um alias para o nome "canónico" (o nome real)
  - **O valor é o nome canónico**
- Type=MX
  - **O valor é o nome de um mail server do domínio e a respectiva prioridade**

Nomes e atributos

Os dados guardados pelo DNS estão estruturados na forma: nome, atributo, atributo, atributo, …..

Um atributo tem um tipo. Podem aparecer vários atributos do mesmo tipo. Ao triplo (nome, tipo do atributo, atributo) chama-se um Resource Record (RR). Os tipos de atributos ou tipo de RRs mais conhecidos são:

Tipo	*Descrição do atributo*
SOA	**Start Of Authority**
A	**ip Address associado ao nome**
CNAME	**Canonical NAME (introdução de aliases)**
MX	**Mail eXchange associado ao nome**
NS	**Name Server associado ao nome**
TXT	**TeXT (comentário) associado ao nome**
HINFO	**Host INFOrmation**
PTR	**Pointer**
…….	

Exemplo fictício

di.fct.unl.pt	**IN**	**SOA**	**ns.di.fct.unl.pt root.di.fct. unl.pt (200110113, 28800, 7200, 604800, 86400)**
;			
di.fct.unl.pt	**IN**	**NS**	**ns.di.fct.unl.pt**
di.fct.unl.pt	**IN**	**NS**	**ftp.di.fct.dns.pt**
di.fct.unl.pt	**IN**	**TXT**	**"DI – FCT/UNL Portugal"**
di.fct.unl.pt	**IN**	**MX**	**20 ns.di.fct.unl.pt**
di.fct.unl.pt	**IN**	**MX**	**30 ftp.di.fct.unl.pt**
ftp.di.fct.unl.pt	**IN**	**A**	**192.34.67.34**
router.di.fct.unl.pt	**IN**	**A**	**192.34.67.254**
ns.di.fct.unl.pt	**IN**	**A**	**192.34.67.1**
mail.di.fct.unl.pt	**IN**	**CNAME**	**ns.di.fct.unl.pt**
;			
;			
asc.di.fct.unl.pt	**IN**	**NS**	**corton.di.fct.unl.pt**
asc.di.fct.unl.pt	**IN**	**NS**	**ns.asc.di.fct.unl.pt**
…….			

Para interrogar o DNS

1) Ver o ficheiro /etc/resolv.conf para saber o(s) endereço(s) dos servidores que o seu host está a usar

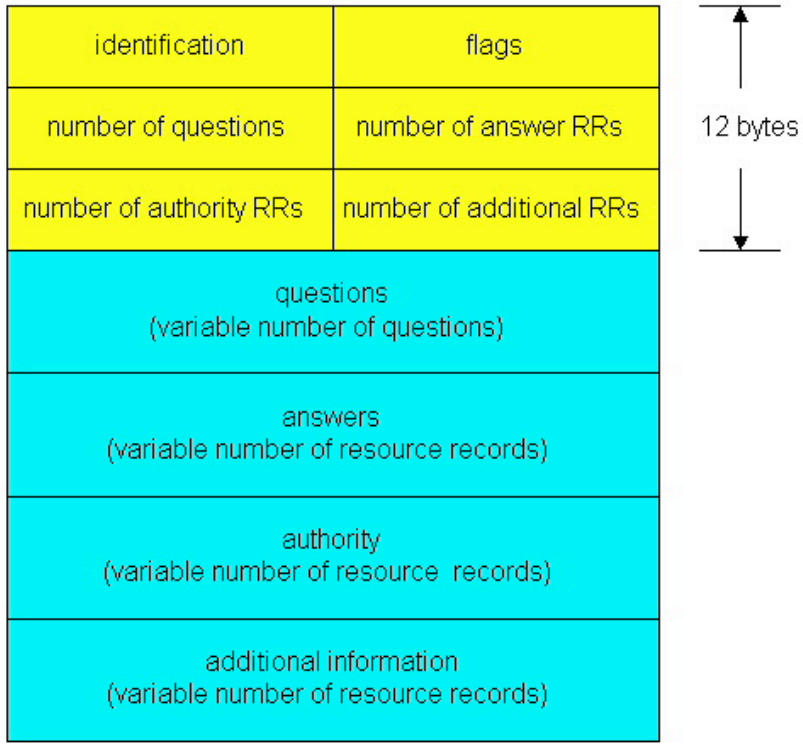
2) Usar o comando: dig [@server name] name [RR type]

3) ou o comando: nslookup [name]

O Protocolo do DNS

Protocolo DNS : mensagens de pergunta e resposta, ambas com o mesmo formato

- Cabeçalho:
- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



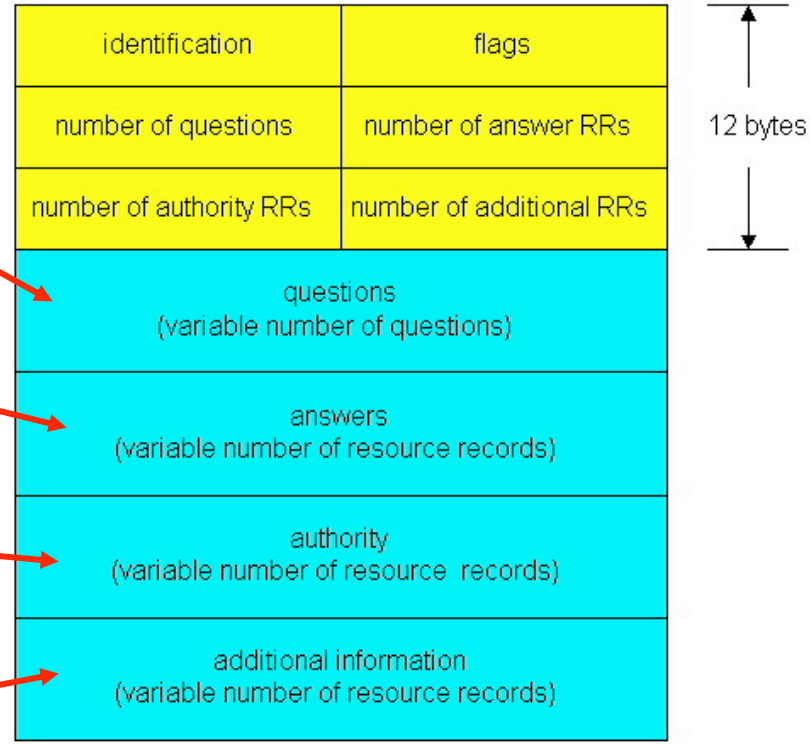
Continuação

Nome e tipos envolvidos na questão

RRs em resposta à pergunta

RRs descrevendo os servidores com autoridade envolvidos

Informação adicional eventualmente útil



Campos da mensagem

Cabeçalhos - as flags permitem saber se é um pedido ou uma resposta, se a resposta é autoritária, se se deseja recursividade e se a mesma está disponível, códigos de erro, etc; o campo identificação é posto pelo cliente e conservado pelo servidor para que o cliente possa ligar o pedido à resposta.

Query - pergunta a fazer ou feita

Answer - o que o servidor consegue saber em resposta a essa pergunta (pode ser informação cached)

Authority - dados sobre os name servers com autoridade sobre os dados listados na resposta

Additional - dados que podem vir a ser úteis (informações suplementares que podem evitar mais perguntas).

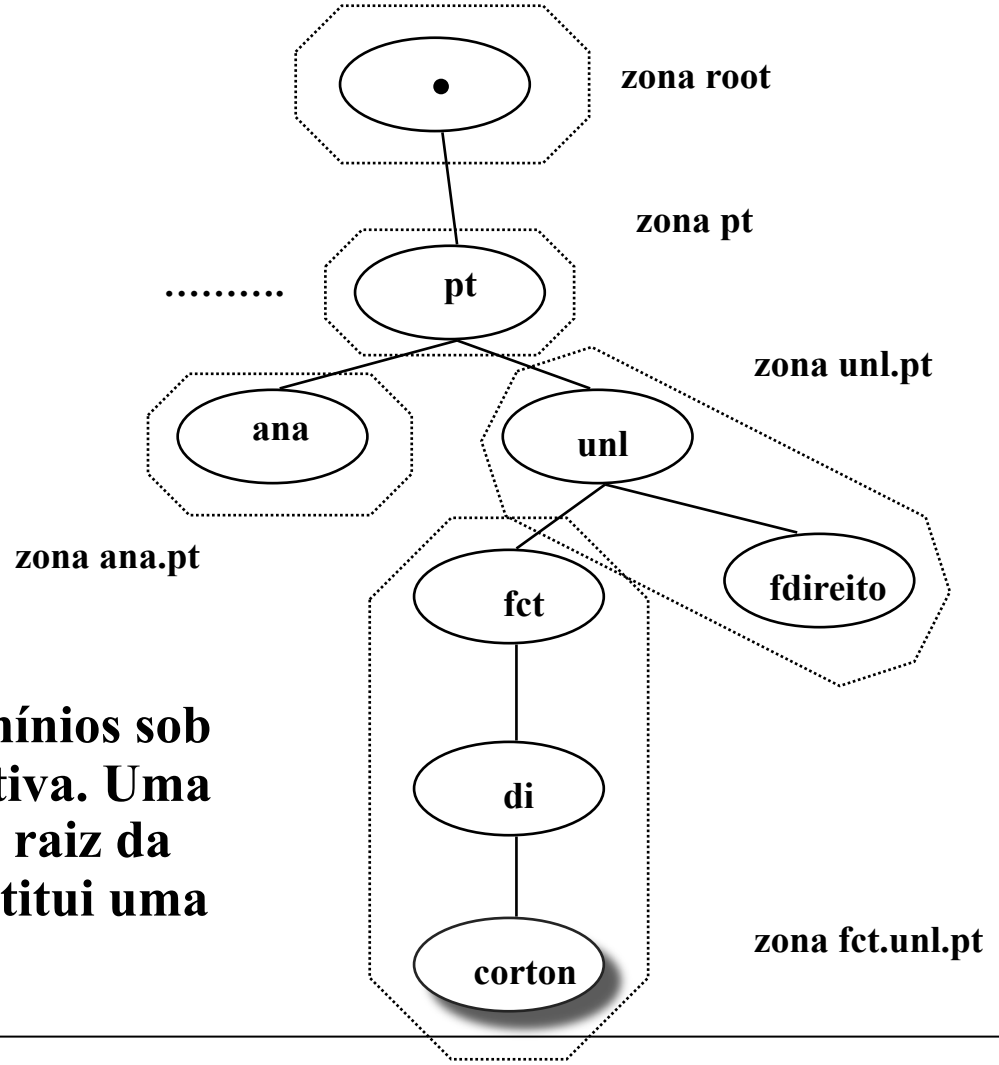
Aspectos suplementares

- Nos transparentes anteriores foi sugerido que a cada componente de um nome DNS (....domínio....) correspondia de facto um servidor da hierarquia
- Na verdade o DNS é mais complexo e envolve uma outra subdivisão da árvores de nomes que se designam zonas e vários tipos de servidores
- Por outro lado, vários servidores são autoritários sobre cada zona (conjunto de domínios) e existe um protocolo de manutenção da coerência dos diferentes servidores da zona
- Finalmente, o DNS não só mantém atributos associados aos nomes dos domínios, como também mantém uma função inversa que a um endereço IP faz corresponder um nome

Definições

- Uma zona é um conjunto de nomes que têm uma raiz comum e cuja administração está sob a mesma autoridade
- Qualquer nó não terminal de uma zona pode dar origem à raiz de uma outra zona, dita delegada
- Associado a cada zona existe um conjunto de servidores com autoridade sobre a zona. Este conjunto é formado por um servidor primário (o único que aceita modificações dos dados da zona) e vários secundários (que replicam a informação do primário)
- Um servidor pode ser primário ou secundário de várias zonas
- Se um servidor tem autoridade sobre uma zona, então ele conhece todos os nomes dessa zona (e respectivos atributos)

Noção de zona de autoridade administrativa



Uma zona é um conjunto de domínios sob a mesma autoridade administrativa. Uma zona é identificada pelo domínio raiz da zona. O conjunto das zonas constitui uma partição do espaço de nomes.

Tipos de servidores

Servidor primário de uma zona

Servidor da zona. Tem o controlo sobre as actualizações da zona. Tem autoridade sobre a zona (conhece a verdade sobre a zona).

Servidor secundário

Servidor da zona. Mantém uma cópia da informação da zona mas não aceita actualizações da mesma. Tem autoridade sobre a zona (conhece a verdade sobre a zona).

Servidores de *caching*

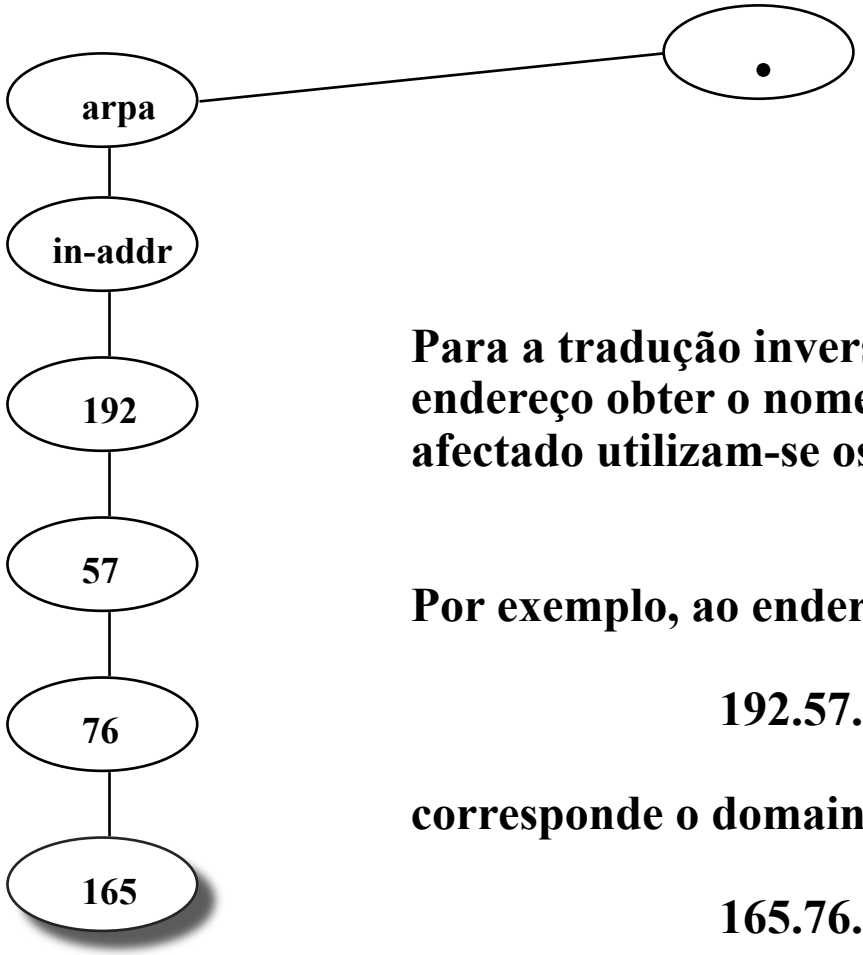
Obtêm e mantêm informações a pedido. No entanto, não têm autoridade sobre as mesmas pois têm uma visão incompleta e eventualmente desactualizada

Nota: todos os servidores fazem *caching* das informações que obtêm

Coerência da replicação

- Os servidores secundários contactam periodicamente o primário a saber se houve actualizações. Se é o caso, tiram uma nova cópia da zona
- Sempre que há modificações, o primário tenta "dizer aos secundários" que o contactem
- Quando um servidor qualquer adquire alguma informação, coloca-a na sua cache. À informação *cached* está associado um TTL (Time To Live). Ultrapassado este limite, a cache é invalidada
- Os mecanismos usados para assegurar a coerência foram reformulados, ver o RFC 2136.

Domínios de Reverse-mapping



Para a tradução inversa. Isto é, para a partir de um endereço obter o nome do host a que ele está afectado utilizam-se os domínios de reverse-mapping

Por exemplo, ao endereço:

192.57.76.165

corresponde o domain name:

165.76.57.192.in-addr.arpa.

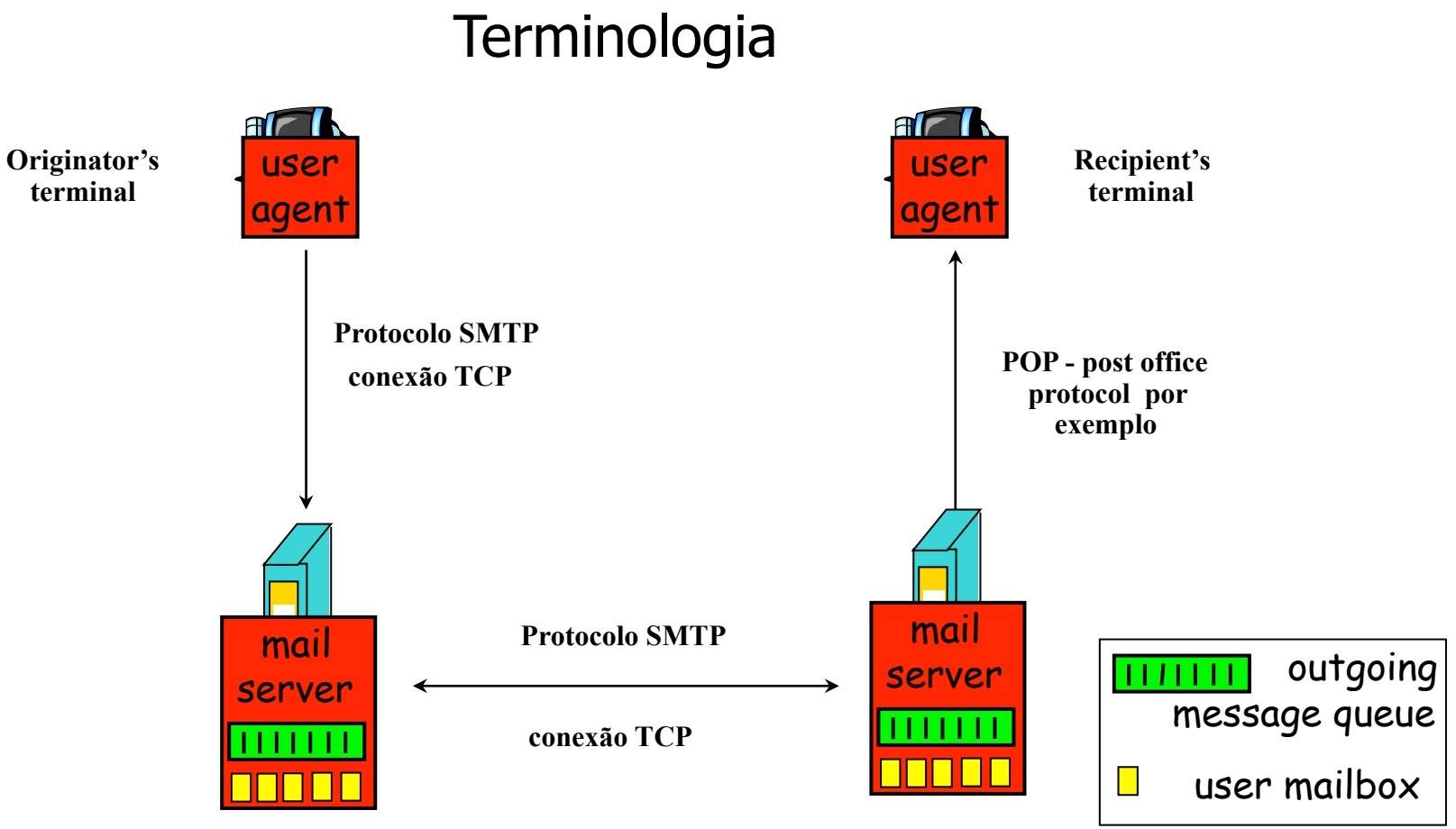
Organização do capítulo

- **Aplicações em rede ou distribuídas**
- **Conceitos de base, paradigmas e tipos de transportes**
- **Protocolo HTTP (Web)**
- **O DNS ("Domain Name System")**
- **Protocolo SMTP — Correio electrónico**
- **Transferência de ficheiros - Protocolo FTP e sistemas P2P**
- **Os protocolos RTP e SIP**

Correio electrónico

Um serviço de troca e gestão de mensagens diferidas para a Internet

SMTP - Simple Mail Transfer Protocol



SMTP - características

- RFC 821 define o protocolo
- RFC 822 define o formato das mensagens e endereços (user@domain name)
- O cliente SMTP estabelece uma conexão TCP para o servidor SMTP na porta 25 (smtp)
- Por essa conexão passam comandos e mensagens
- Três fases da interacção: greetings, troca de mensagens, fim
- Geralmente, a transferência faz-se em NVT ASCII ou "ASCII 7 bits" (mas cada vez menos)

Comandos SMTP

Comandos mais correntes:

HELO message	**- O cliente identifica-se**
MAIL from: address	**- Endereço origem**
RCPT to: address	**- Endereço destino**
DATA ……	**- Mensagem**
QUIT	**- Fim**
EXPN address	**- Expande um endereço local**
VRFY address	**- Verifica se existe localmente**

O servidor responde sempre com um diagnóstico (Exemplo: “250 OK”)

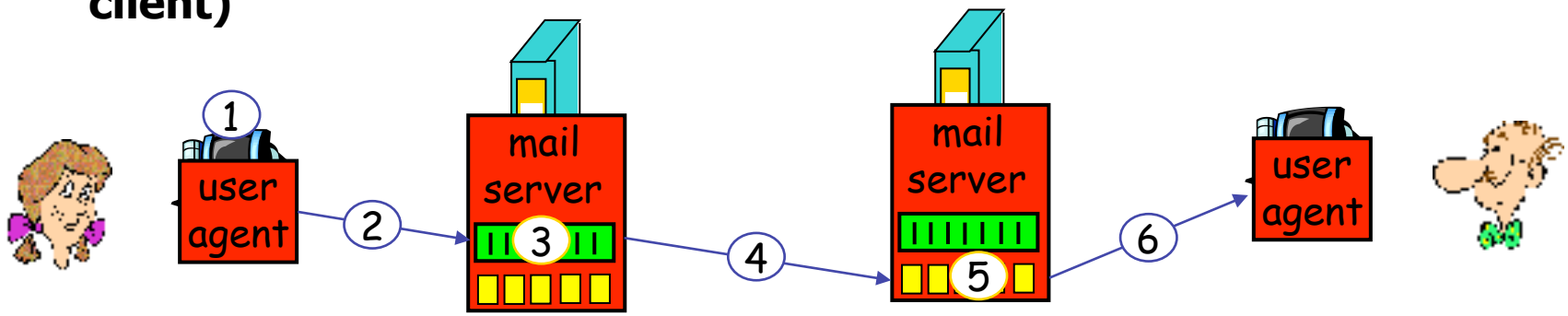
Exemplo de uma sessão SMTP

220 beta.di.fct.unl.pt SMTP Server Ready ***(servidor beta.di.fct.unl..pt)***
HELO alpha.tap.pt **(cliente alpha.tap.pt)**
250 beta.di.fct.unl.pt

MAIL FROM: Smith@tap.pt
250 OK
RCPT TO: Silva@di.fct.unl.pt
250 OK
RCPT TO: Machado@di.fct.unl.pt
250 OK
DATA
354 Start mail input; end with <CR><LF>.<CR><LF>
…….
…….
……..
<CR><LF>.<CR><LF>
250 OK
QUIT
221 beta.di.fct.unl.pt Service closing transmission channel

Cenário: Alice envia uma mensagem ao Bob

1) Alice através do seu UA compõe uma mensagem para bob@someschool.edu
2) A mensagem é enviada para o mail server da Alice e fica em fila de espera até poder ser transmitida para o mail server de someschool.edu
3) O servidor da Alice abre uma conexão TCP para o mail server someschool.edu (SMTP client)
4) Através da conexão SMTP a mensagem é entregue ao servidor someschool.edu
5) E a mensagem é colocada na mailbox de Bob
6) Mais tarde, Bob através do seu UA, vai ler a mensagem da Alice



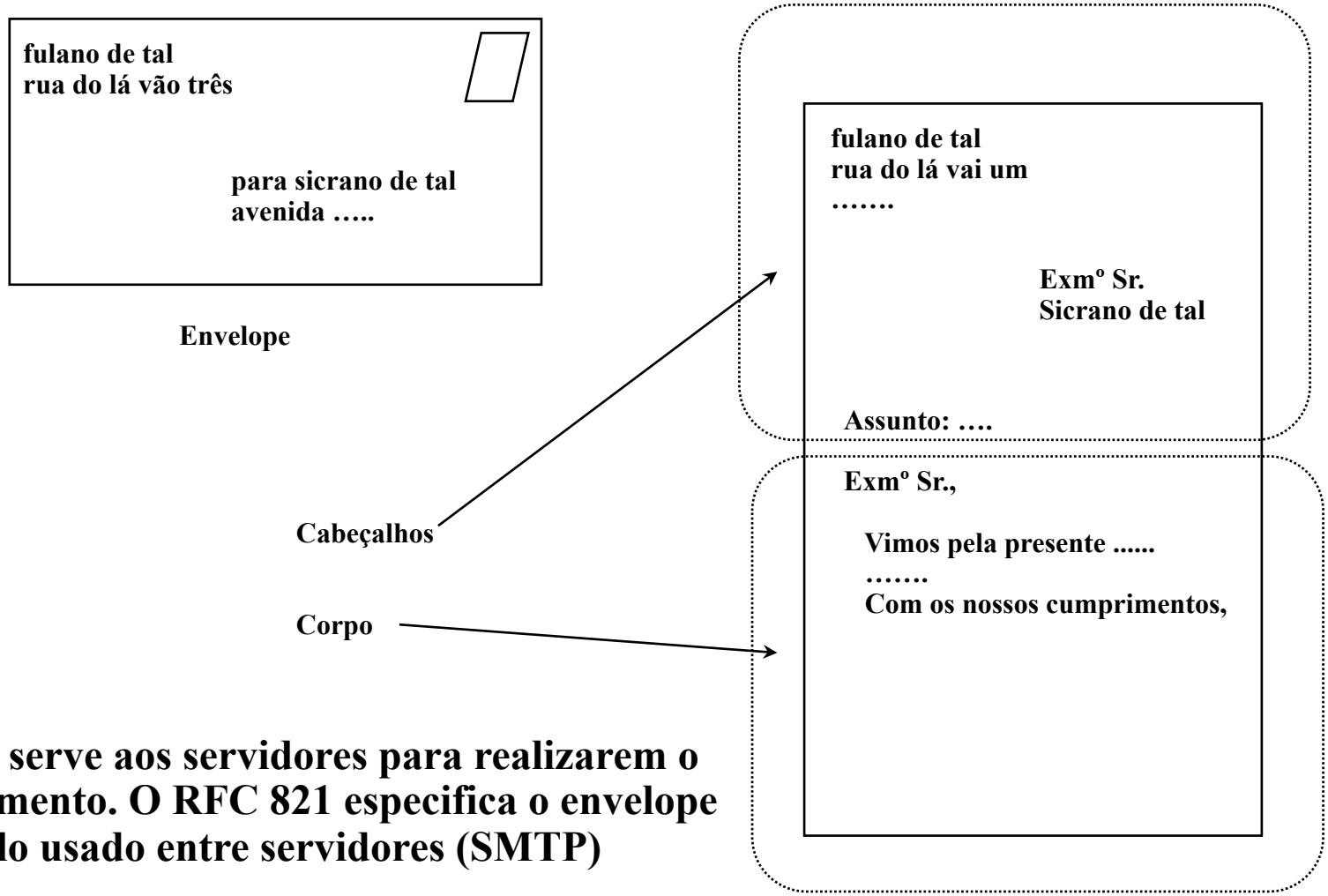
Routing de e-mail

Um endereço de e-mail segundo o RFC 822 contem um *domain name* na parte direita; este *domain name* especifica o servidor que gere a mailbox do destinatário da mensagem. Para se chegar ao mesmo, eventualmente, é necessário passar através de outros servidores intermédios. Para indicar estes caminhos alternativos, usam-se os chamados "*mail exchange resource records*" no DNS. Exemplo:

di.fct.unl.pt	**IN**	**MX**	**20**	**mail.di.fct.unl.pt.**
di.fct.unl.pt	**IN**	**MX**	**30**	**ftp.di.fct.unl.pt**
di.fct.unl.pt	**IN**	**MX**	**100**	**gatekeeper.rccn.net**

Este mecanismo permite introduzir redundância no processo de entrega de e-mail ao destinatário final e permite igualmente introduzir no espaço de endereçamento do e-mail nomes que não são directamente nomes de servidores ou que são gateways de e-mail, etc.

Envelopes, cabeçalhos e corpo da mensagem



O envelope serve aos servidores para realizarem o encaminhamento. O RFC 821 especifica o envelope e o protocolo usado entre servidores (SMTP)

Exemplo de mensagem

From J.Legatheaux.Martins@di.fct.unl.pt Sun May 28 12:05:16 1996
Return-Path: J.Legatheaux.Martins@di.fct.unl.pt
Received: from ns.di.fct.unl.pt (192.34.156.10) by gate.tap.pt
with SMTP id MAA05407 for <silva@tap.pt>; Sun May 28 12:02:10 1996
Received: from pc2.in.di.fct.unl.pt (192.34.156.56) by ns.di.fct.unl.pt
with SMTP id MAA05407 for <silva@tap.pt>; Sun May 28 12:02:1 1996
Received: by pc2.in.di.fct.unl.pt id MAA05407; Sun May 28 12:01:45 1996
Date: Sun May 28 12:01:45 1996
From: Jose Legatheaux Martins <J.Legatheaux.Martins@di.fct.unl.pt>
Message-Id: <199605281201.AA05407@ pc2.in.di.fct.unl.pt>
To: silva@tap.pt
Subject: Temos de ir ao Algarve no fim de semana
Status: …..

Caro Silva,

……..

Headers + Body = Content

O RFC 822 especifica o formato dos endereços e os diferentes campos do cabeçalho (obrigatórios e opcionais).

Exemplos:

Received
Message-Id
From
Date
Reply-To
To
Subject
Priority
Content-Type
Sender
……

MIME - Multipurpose Internet Mail Extensions

Para ultrapassar os limites impostos pelo RFC 822 (mensagem em NVT ASCII ou ASCIIsó com 7 bits), foram introduzidas pelos RFCs 1341 e 1521 convenções para transmitir:

+ mensagens com caracteres acentuados
+ mensagens com alfabetos não latinos (exemplo: russo)
+ mensagens em linguagens sem alfabetos (exemplo: chinês)
+ multimedia
+ mensagens estruturadas

Para tal foram introduzidos 5 novos campos de cabeçalho:

Mime-Version:	**- presença obrigatória**
Content-Description:	**- descrição do que está na mensagem**
Content-Id:	**- idem message Id**
Content-Transfer-Encoding:	**- como é a codificação do conteúdo**
Content-Type:	**- Natureza e estrutura da mensagem**

Content-transfer-encoding

Message body encoding

ASCII text (7 bit US ASCII, linhas limitadas a 1000 caracteres)
8 bit (any 8 bit char set, e. g. ISO-8859-1, linhas limitadas a 1000 caracteres)
base64 encoding ou ASCII armor
(cada grupo de 3 octetos, 24 bits, é separado em 4 grupos de 6 bits e cada 6 bits, com valores de 0 a 63, é codificado através de um código ASCII; "A" codifica 0, "Z" - 25, "a" - 26, "z" - 51, "0" - 52, "9" - 61, "+" - 62, "/" - 63).

Non-ASCII characters in headers

=?charset=?encoding=?encoded-text?=

charset = US ASCII ou ISO-8859-x
encoding = Q - quoted printable (exemplo: "é" é codificado por =E9 hexa)
B - base64 encoding

MIME types

`Content-Type: type/subtype; parameters`

- Text
- example subtypes: **`plain`**, **`html`**

- Image
- example subtypes: **`jpeg`**, **`gif`**

- Audio
- example subtypes: **`basic`** (8-bit mu-law encoded), **`32kadpcm`** **(32 kbps coding)**

- Video
- example subtypes: **`mpeg`**, **`quicktime`**

- Application
- other data that must be processed by reader before "viewable"
- example subtypes: **`msword`**, **`octet-stream`**

Mensagem multipart

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
--98766789--
```

Extensões ao SMTP - RFC 1425

Extended SMTP ou ESMTP

cliente envia "EHLO" em vez de "HELO"

server responde RSET (Reset, I donn't speak ESMTP)

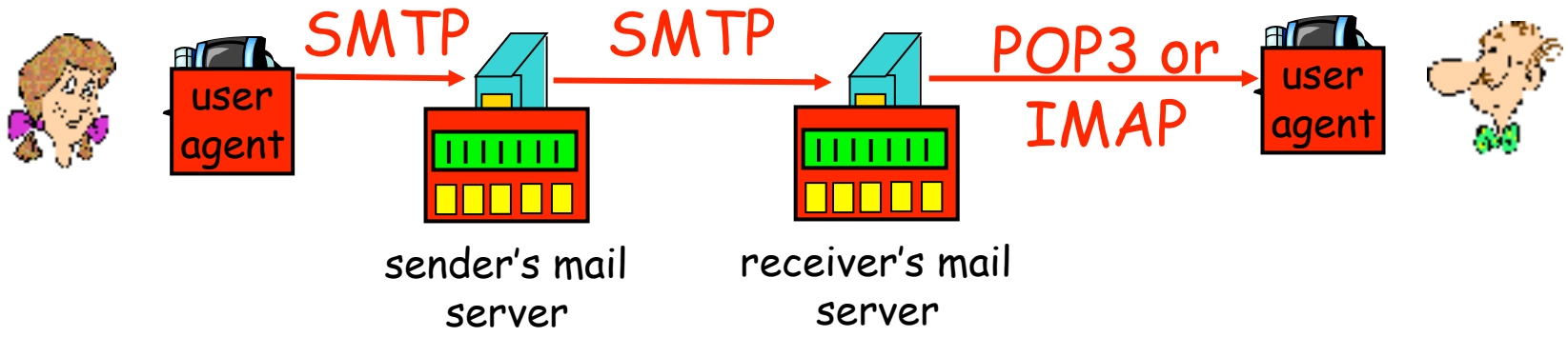
ou aceita; neste caso novos comandos são aceites como por exemplo:

8BITMIME - o cliente pode enviar mensagens MIME com 8 bits

SIZE - indica o tamanho máxima das mensagens aceites

….

Protocolos de acesso ao correio



- **SMTP: entrega de mensagens de servidor em servidor**
- **Protocolos de acesso ao correio: acesso à mailbox**
  - **POP: Post Office Protocol [RFC 1939]**
    - **autenticação (agent <-->server) and download das mensagens**
  - **IMAP: Internet Mail Access Protocol [RFC 1730]**
    - **Mais possibilidades (mais complexo)**
    - **Manipulação de folders e de mensagens no servidor**
  - **HTTP: Hotmail , Yahoo! Mail, Gmail, etc.**

Protocolo POP3

Fase de autenticação

- Comandos do cliente:
  - `user:` indica username
  - `pass:` indica password
- O servidor responde
  - `+OK`
  - `-ERR`

Fase da transacção, cliente:

- `list:` lista mensagens
- `retr:` obtem uma mensagem
- `dele:` suprime uma mensagem
- `quit`

```
S: +OK POP3 server ready
C: user alice
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 e IMAP

POP3

- Nos exemplos anteriores usa-se o modo "download and delete"
- Se o Bob muda de cliente não pode voltar a ler o e-mail
- "Download-and-keep": as mensagens podem ser downloaded para vários clientes
- POP3 tem a noção de cliente e actividade de um cliente

IMAP

- Mantém todas as mensagens no servidor
- Permite ao utilizador organizar as mensagens em pastas
- IMAP mantém estado para o utilizador e as suas sessões

SMTP e HTTP

- **SMTP**
- **O smtp usa conexões persistentes**
- **O smtp requer que a mensagem seja em 7-bit ascii (header & body)**
- **Certos caracteres não podem fazer parte da mensagem (e.g., `CRLF.CRLF`). Logo, a mensagem tem de ser codificada de forma especial**
- **O servidor smtp usa `CRLF.CRLF` para detectar o fim da mensagem**

- **Comparação com HTTP**
- **http: pull**
- **email: push**
- **Ambos usam interacção ASCII (command/response e status codes)**
- **http: cada objecto está encapsulado na sua própria mensagem**
- **smtp: múltiplos objectos podem ser enviados numa mensagem MIME estruturada**

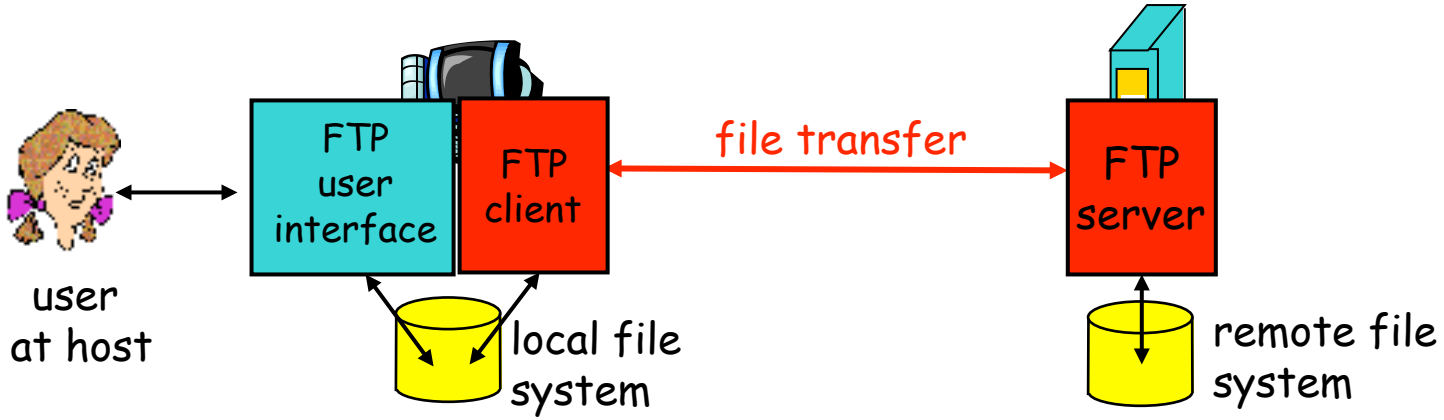
Organização do capítulo

- **Aplicações em rede ou distribuídas**
- **Conceitos de base, paradigmas e tipos de transportes**
- **Protocolo HTTP (Web)**
- **O DNS ("Domain Name System")**
- **Protocolo SMTP — Correio electrónico**
- **Transferência de ficheiros - Protocolo FTP e sistemas P2P**
- **Os protocolos RTP e SIP**

Transferência de ficheiros

Um serviço de gestão de ficheiros para a Internet

O serviço FTP - File Transfer Protocol



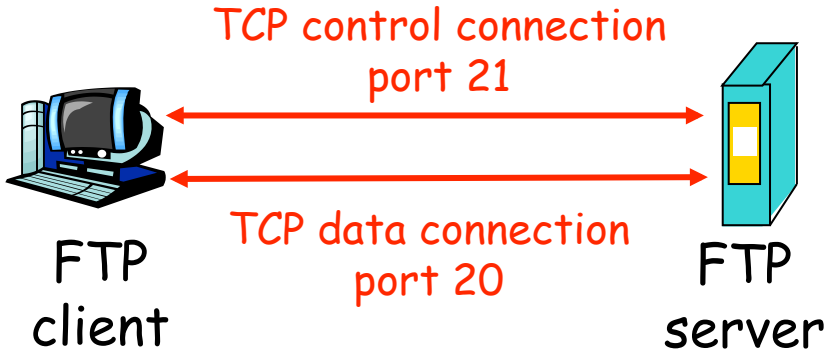
- transfere de e para o host remoto
- Modelo cliente / servidor
  - Cliente: inicia as transferências
  - Servidor: fica como o host remoto
- ftp: RFC 959
- servidor ftp: porta 21

O protocolo FTP - File Transfer Protocol

- **Baseado no modelo cliente/servidor usando TCP.**
- **O servidor espera as conexões dos clientes numa porta bem-conhecida (ver "/etc/services" => porta 21)**
- **O cliente acede ao serviço abrindo uma conexão para o servidor pretendido na porta pré-estabelecida.**
- **Nesta conexão de controlo é usada a representação ASCII dos comandos, parâmetros e diagnósticos.**
- **A conexão de controlo é usada para o cliente efectuar pedidos e receber as respostas do servidor.**
- **Após cada pedido deve ser recebida uma resposta (que indica o resultado e qual a acção a tomar em seguida)**

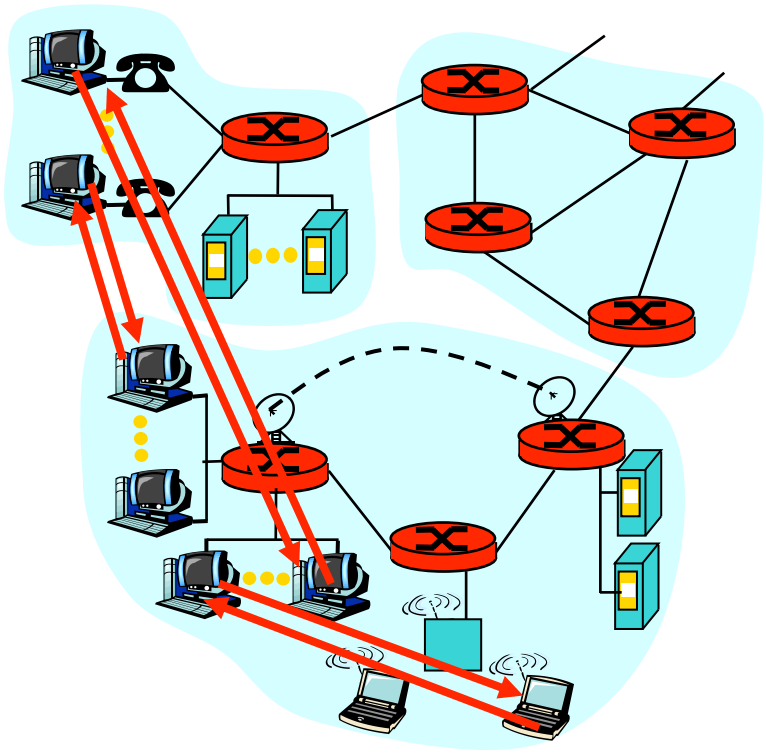
O protocolo FTP usa duas conexões TCP

- O cliente ftp abre uma conexão TCP para a porta 21 do servidor
- São usadas duas conexões TCP:
  - controlo: envio de comandos para o servidor e recepção das respostas.
    - "out of band control"
  - data: o ficheiro
- O servidor ftp mantém "estado": directório corrente, dados de autenticação



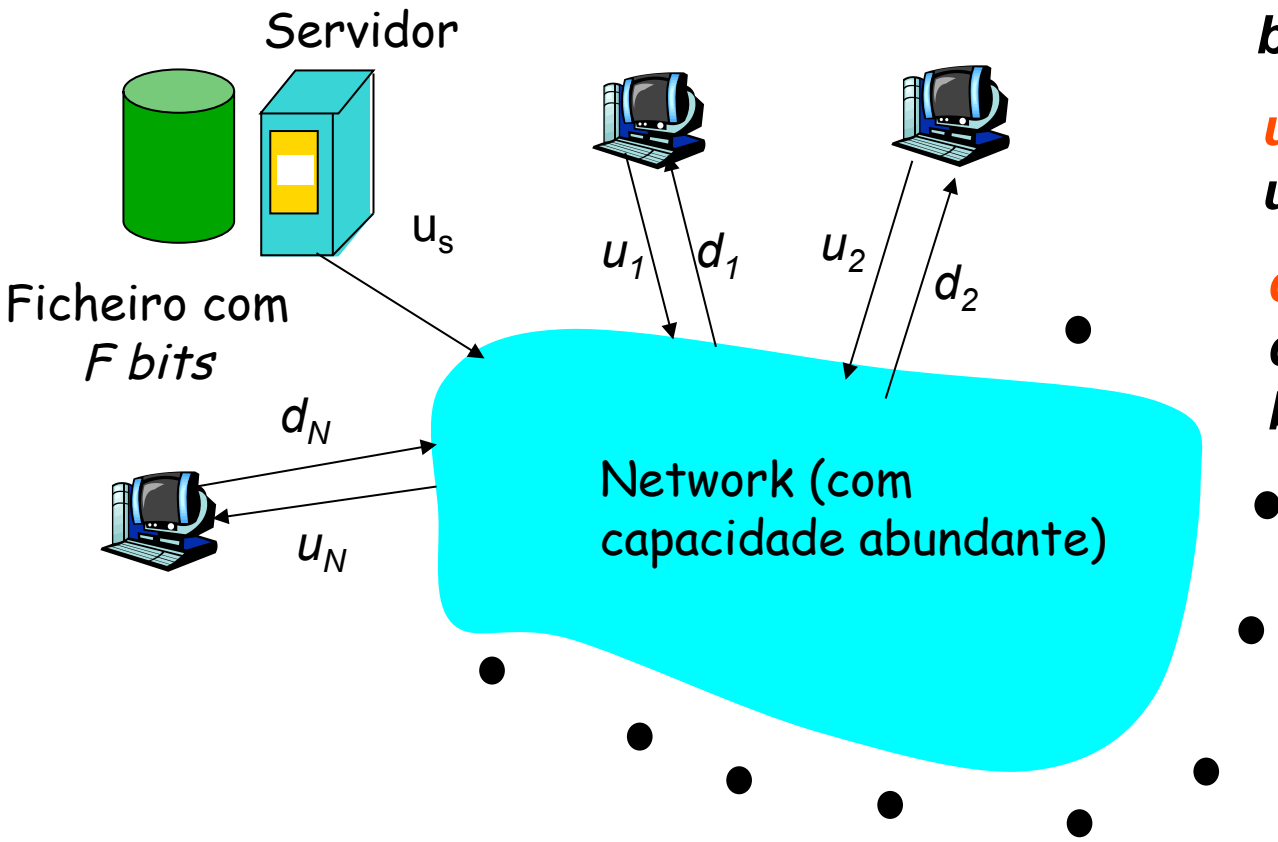
Arquitecturas P2P

- **Conjuntos de *hosts* (*peers*) arbitrários comunicam entre si**
- ***Peers* (nós, parceiros, ...) nem sempre estão ligados e podem trocar de endereço IP de cada vez que estão ligados**
- **Muito escalável**
- **Mas mais difícil de gerir**



Comparação de cliente/servidor e arquitecturas P2P

Questão : Quando demora transmitir um ficheiro F para N nós ?



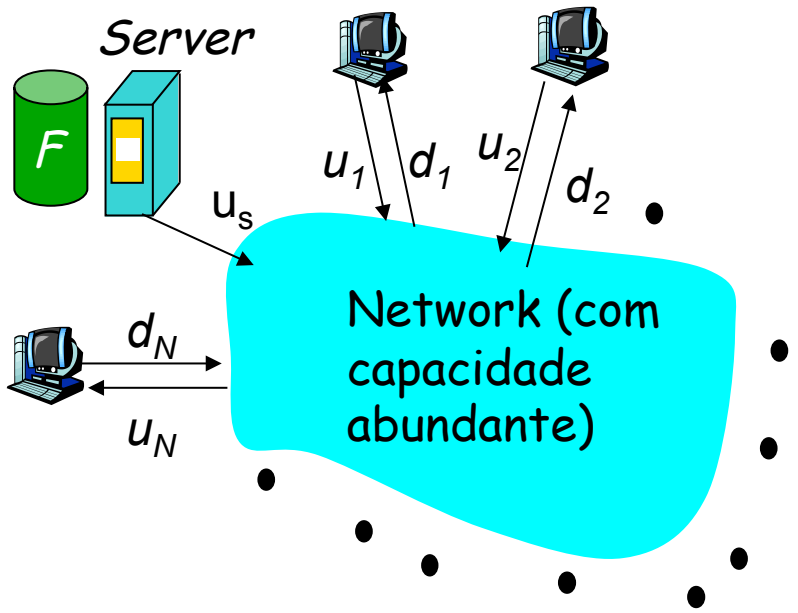
u_s: server upload bandwidth

u_i: client/peer i upload bandwidth

d_i: client/peer i download bandwidth

Tempo de distribuição de um ficheiro F com um servidor

- O servidor envia N cópias:
  - NF/u_s segundos
- O cliente i leva F/d_i segundos para fazer o *download*



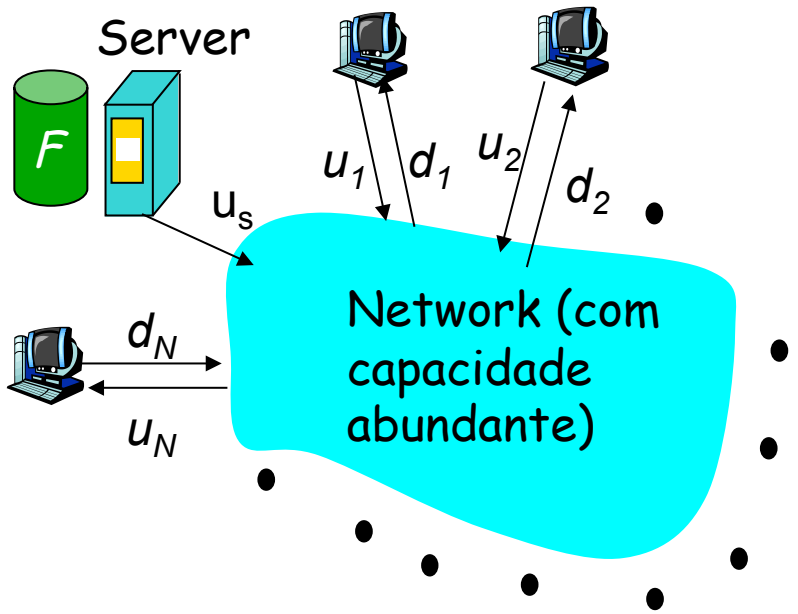
Tempo para fazer o *Download* de F para N = $d_{cs} = \max \{ NF/u_s, F/\min_i(d_i) \}$

Clientes (cliente/servidor)

Linear com N (para N significativo)

Tempo de distribuição de um ficheiro F com P2P

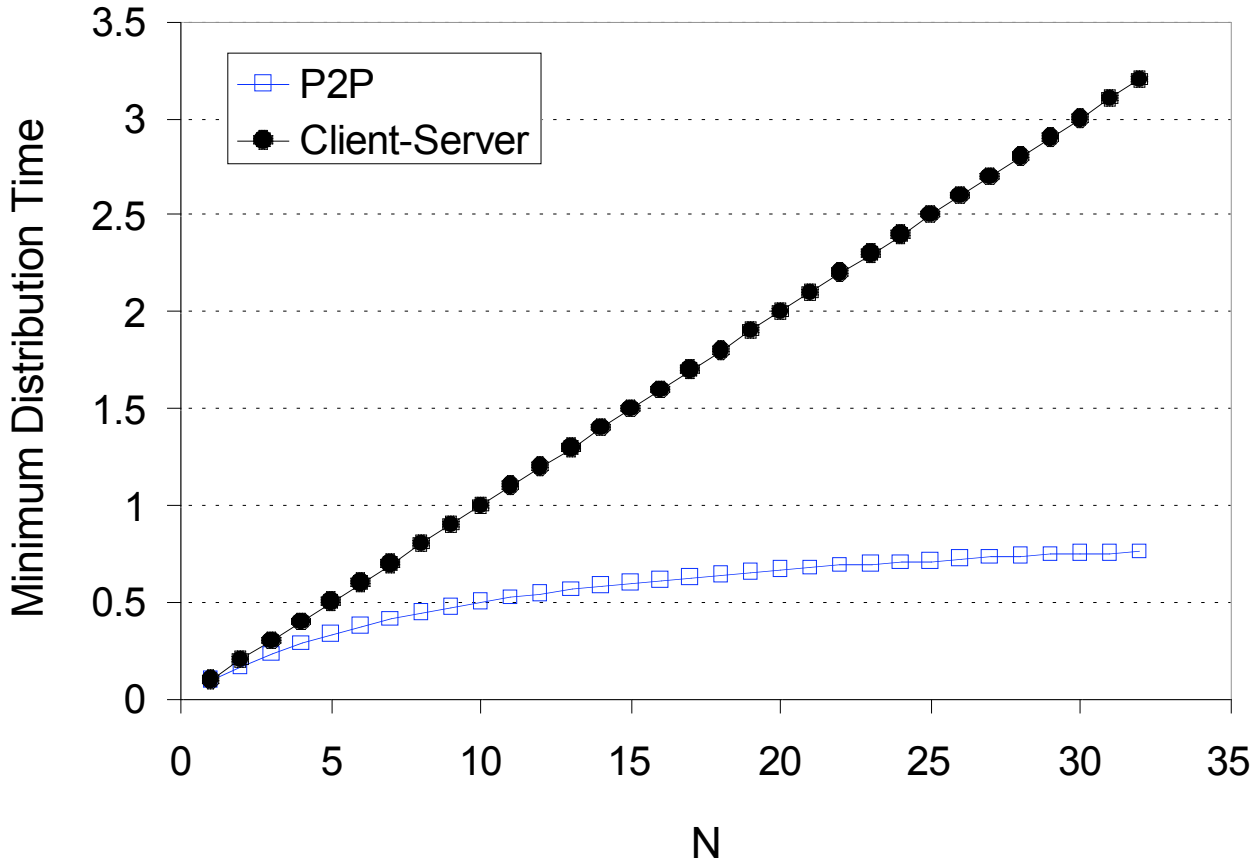
- O servidor envia uma cópia em F/u_s segundos
- O nó i leva F/d_i segundos a fazer o *download*
- NF bits têm de ser *downloaded* (agregados)
- O *upload rate* máximo é: $u_s + \sum_{i=1,N} u_i$



(assumindo que todos os nós enviam *chunks* (pedaços) de ficheiros para o mesmo nó)

$$d_{P2P} = \max \{ F/u_s, F/\min(d_i)_i, NF/(u_s + \sum_{i=1,N} u_i) \}$$

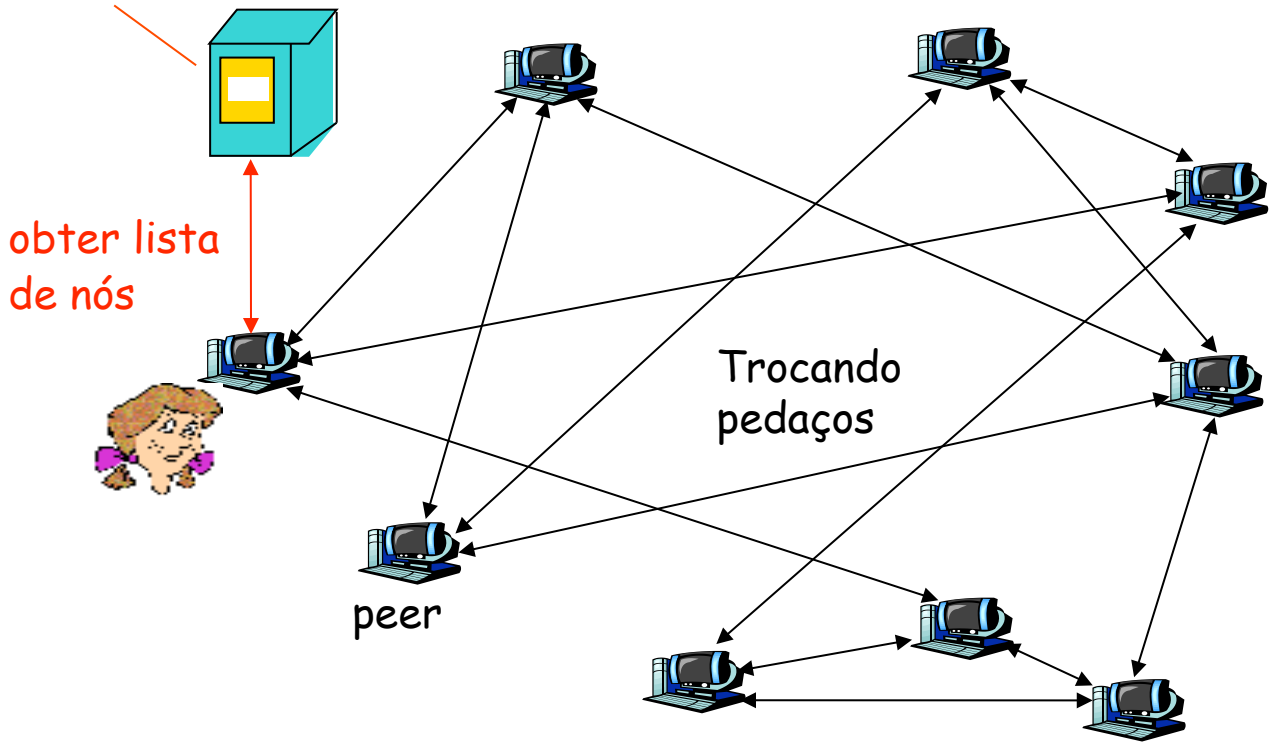
Comparação das duas arquitecturas



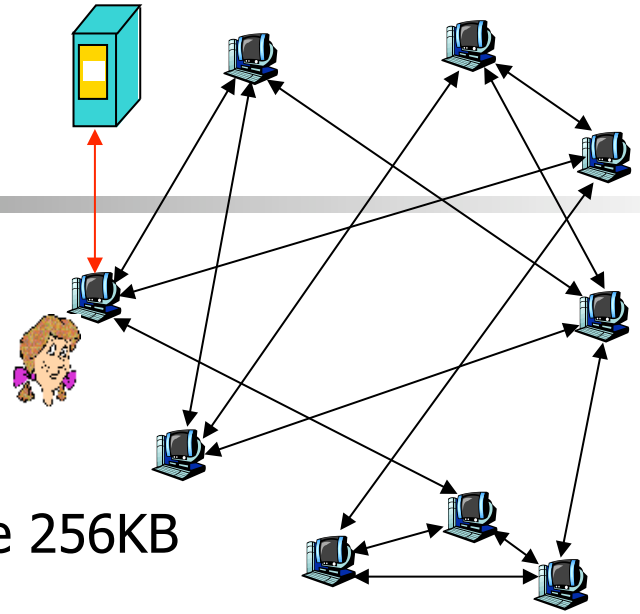
Caso de estudo P2P: BitTorrent

tracker: conhece os *nós* que participam na torrente

torrent: grupo de *nós* que trocam *pedaços* de um ficheiro



BitTorrent (1)



- O ficheiro é dividido em pedaços *chunks* de 256KB
- Cada nó que se junta:
  - Regista-se no *tracker* para receber uma lista de nós e liga-se a um subconjunto dos mesmos ("vizinhos")
  - Não tem *chunks*, mas vai receber alguns dentro de momentos
- Durante o *download*, o nó transmite pedaços para outros nós
- Cada nó pode juntar-se ou partir em qualquer momento
- Quando um nó consegue obter todo o ficheiro, pode (egoisticamente) sair ou (altruisticamente) ficar

BitTorrent (2)

Obtendo *Chunks*

- Em cada momento um grupo de nós tem um certo conjunto de *chunks*
- Perodicamente, cada nó pede aos outros a lista de *chunks* de cada um
- Cada nó selecciona os pedaços a obter segundo a estratégia:
  - *rarest first*

Enviando *Chunks: tit-for-tat*

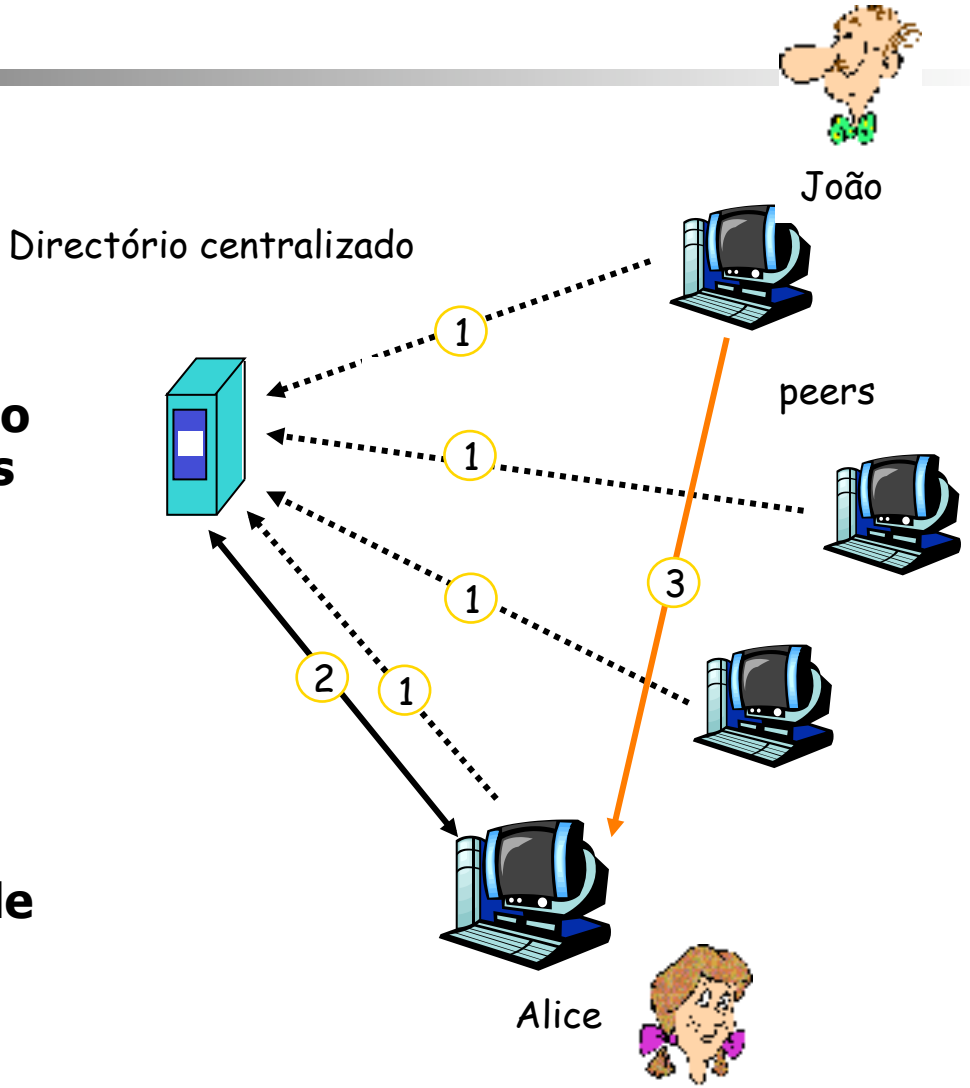
- O nó envia *chunks* para os 4 vizinhos que lhe estão a enviar *chunks* com maior velocidade
  - Esta lista é reavaliada de 10 em 10 segundos
- Em cada 30 segundos um nó selecciona aleatoriamente outro a quem começa a enviar *chunks*
  - Este novo nó pode vir a juntar-se aos top 4 ou ser abandonado

Pesquisa para transferência de ficheiros P2P

- **Exemplo:**
- Alice executa um cliente P2P no seu computador
- Cada vez que se liga à Internet muda de endereço IP
- Vai à procura de "Hey Jude"
- A aplicação mostra uma lista de parceiros com a canção "Hey Jude"
- A Alice escolhe um deles.
- O ficheiro é copiado por HTTP
- Outros utilizadores podem depois copiar a cópia da Alice
- O sistema da Alice é um cliente e um servidor (volátil) de HTTP
- Todos os nós são servidores, o que é muito escalável !

Sistema P2P que usa um índice centralizado

- **Solução "Napster" original**
- **Quando um parceiro se liga informa o sistema central do seu endereço IP e do conteúdo do seu repositório de ficheiros**
- **O índice cntralizado constituí um ponto central de falha**
- **Constituí também um ponto de estrabgulmento**



Query por inundação (flooding)

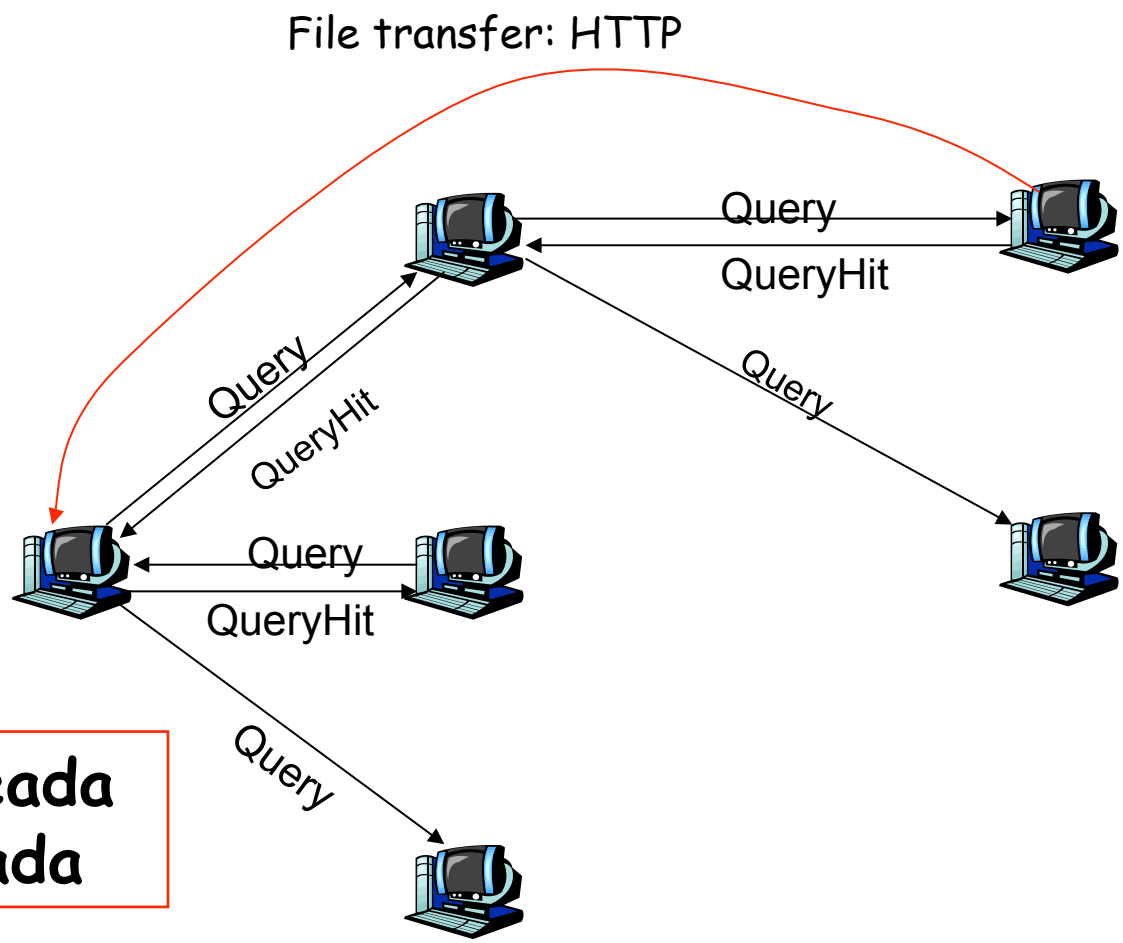
Gnutella:

- Completamente distribuído: sem servidor central
- Protocolo do domínio público
- Muitas implementações do protocolo disponíveis

- **Baseado na noção de rede sobreposta ou rede lógica ou sobreposta (*overlay network*)**
- O arco entre o parceiro X e o parceiro Y materializado por uma conexão TCP
- Todos os parceiros e as ligações formam a rede
- Os arcos são ligações não físicas
- Cada parceiro liga-se a entre meia e uma dúzia de parceiros

Protocolo Gnutella

- A pergunta (query) é enviada aos parceiros
- Os parceiros redirigem a mensagem se não conhecem a resposta
- A resposta (QueryHit) vem pelo caminho inverso



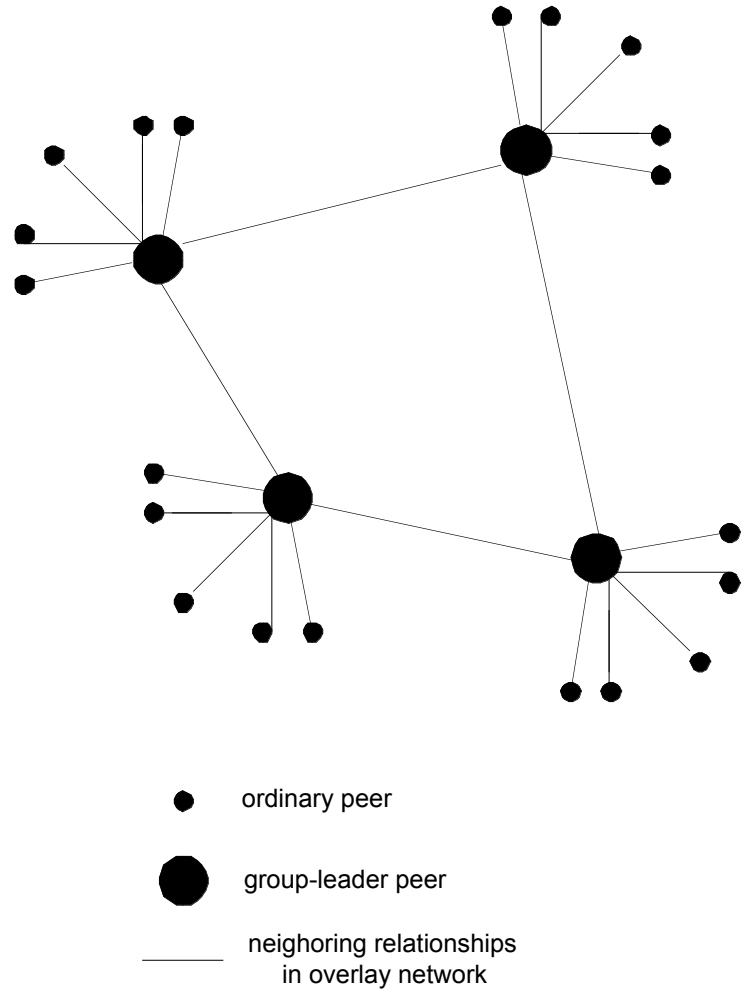
Escalabilidade baseada em inundação limitada

Gnutella: entrada na rede (*join*)

1) O parceiro X que entra usa uma lista de candidatos para encontrar parceiros iniciais
2) X tenta estabelecer conexões TCP sequencialmente com os parceiros da lista até obter uma conexão com um (Y)
3) X envia mensagens Ping para Y; Y redirige as mensagens Ping para os seus parceiros.
4) Todos os parceiros que recebem mensagens Ping respondem com mensagens Pong.
5) X recebe muitas mensagens Pong e pode estabelecer conexões TCP adicionais.

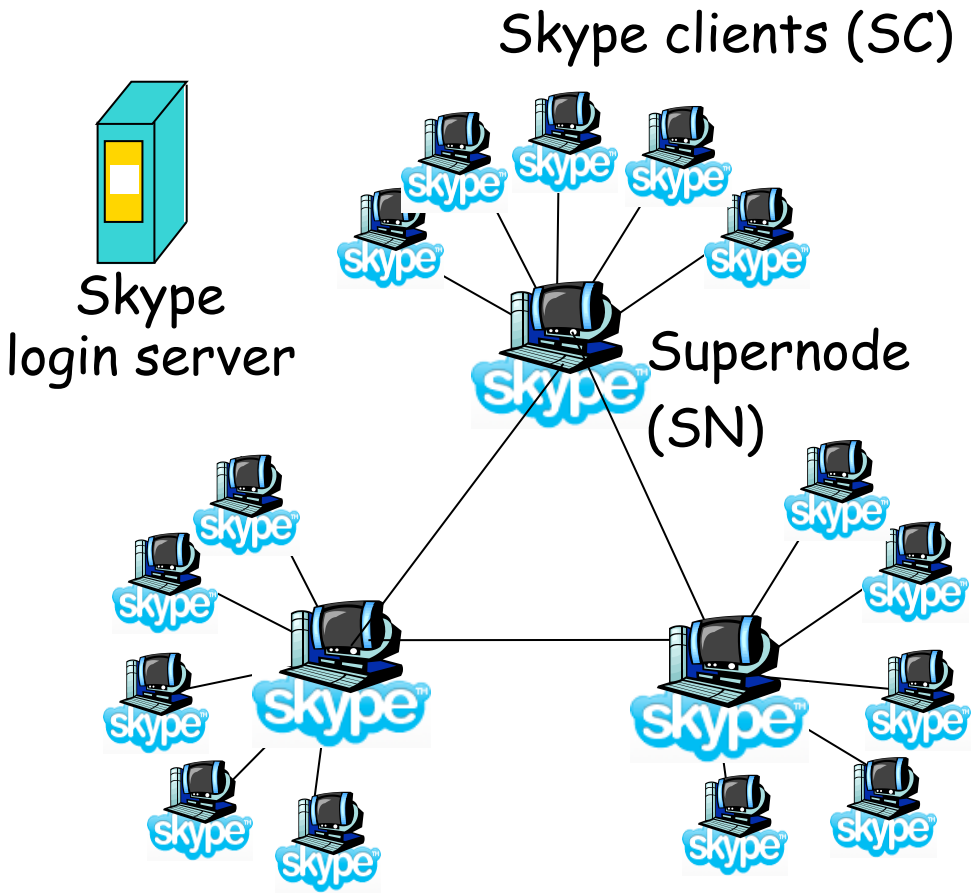
Rede "overlay" hiérarquica

- **Cada parceiro é líder de um grupo ou dependente de um líder.**
  - **Abre uma ligação TCP para o seu líder.**
  - **Os líderes abrem conexões entre si.**
- **Cada líder regista o conteúdo dos repositórios dos seus dependentes**
- **A pesquisa entre líders é realizada por *flooding***



Estudo de caso P2P: Skype

- **Aplicação P2P (pc-to-pc, pc-to-phone, phone-to-pc) para Voice-Over-IP (VoIP)**
- **Protocolo proprietário estudado por *"reverse-engineering"***
- **Baseado numa rede "Overlay" hierárquica**



Skype: realização da chamada

- O utilizador lança o Skype
- O SC regista-se com um SN
  - Existe uma lista de SNs de bootstrap
- O SC autentica-se com o SN
- Chamada: o SC1 contacta o SN indicando o ID do interlocutor
  - O SN contacta outros SNs com o fim de encontrar o endereço do SC2 com o ID e retorna-o para o SC1
- SC1 liga directamente para o SC2 através de TCP