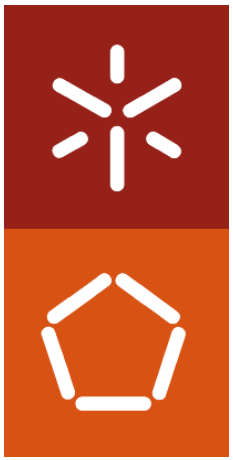
Nível de Aplicação

Mestrado Integrado em Engenharia de Telecomunicações e Informática

3º ano - 2º Semestre

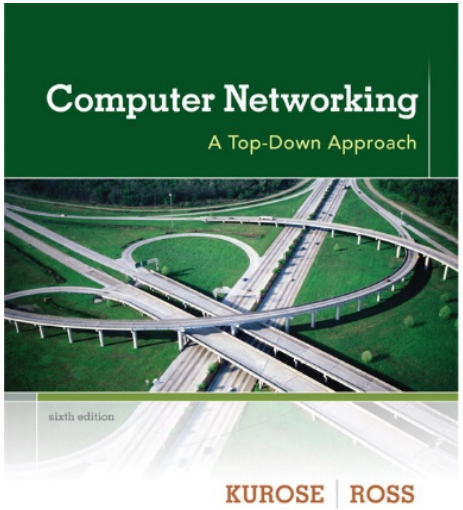
2015/2016



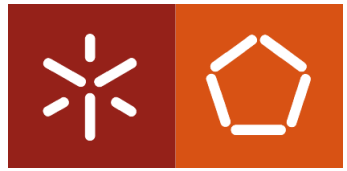
Sumário

- Conceitos gerais
- Paradigmas do nível de aplicação:
  - Paradigma cliente-servidor
  - Paradigma peer-to-peer
- Alguns protocolos do nível de aplicação
  - HTTP, FTP, SMTP/POP3/IMAP, DNS

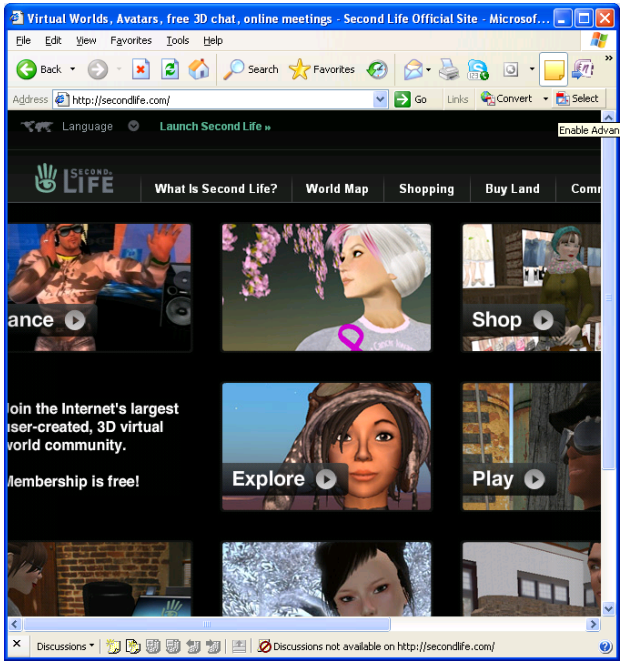
Estes slides são maioritariamente baseados no livro: *Computer Networking: A Top-Down Approach Featuring the Internet, Jim Kurose and Keith Ross,* Addison-Wesleyy

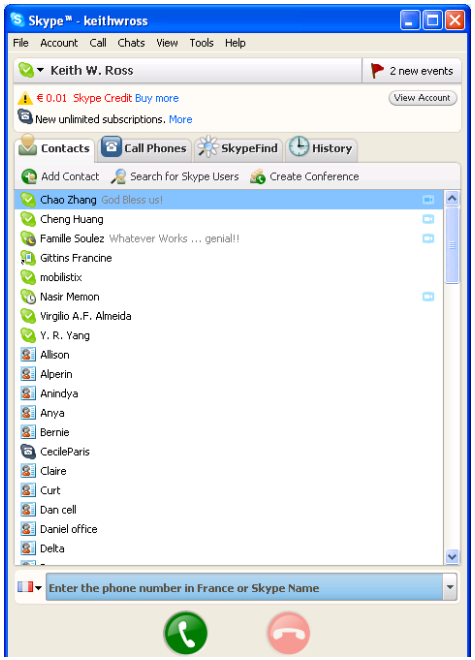


Algumas aplicações de rede



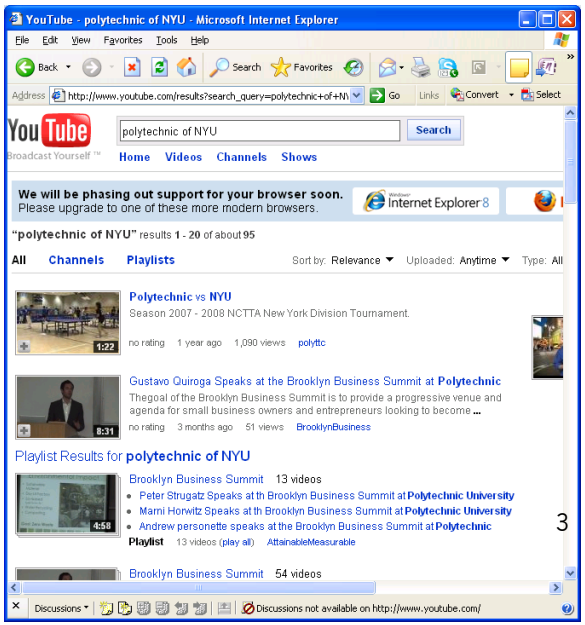
- e-mail
- web
- instant messaging
- acesso remoto
- P2P file sharing
- jogos em rede
- clips de video
- redes sociais
- voice over IP
- vídeo conferencia em tempo-real
- grid computing

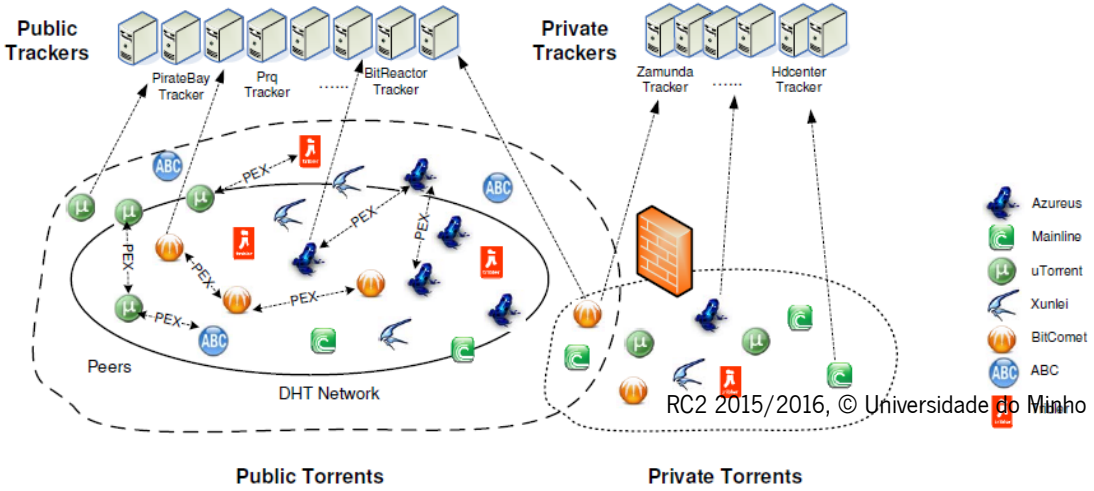




Public Discovery Sites

Private Discovery Sites





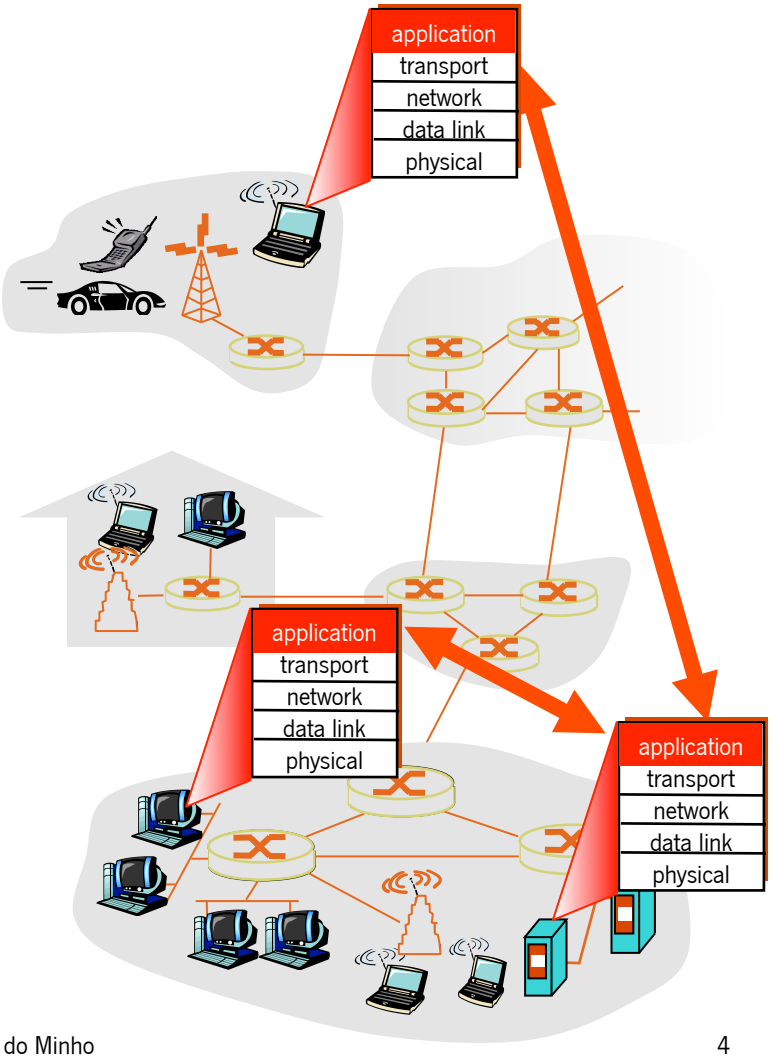
Criar uma aplicação em rede

Desenvolver um programa que:

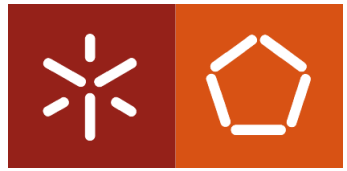
- corra em diferentes sistemas terminais
- comunique através da rede
- ex., o software do servidor web comunica com o software do browser

Não é preciso desenvolver software para os dispositivos do núcleo da rede:

- Os dispositivos do núcleo da rede não correm aplicações para o utilizador
- O facto das aplicações estarem hospedadas nos sistemas terminais permite um rápido desenvolvimento e propagação

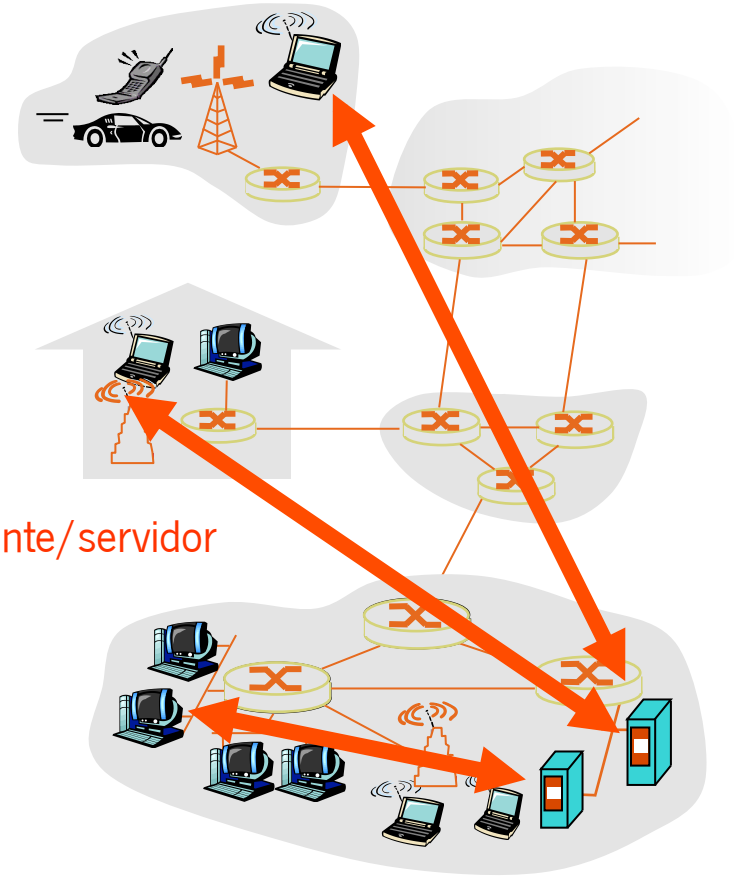


Arquitecturas do nível de aplicação



- **Cliente-servidor**
- **Peer-to-peer (P2P)**
- **Híbridas: cliente-servidor e peer-to-peer**

Arquitectura cliente-servidor



servidor:

- Dados no mesmo servidor
- Endereço IP permanente
- Escala aumentando o número de servidores

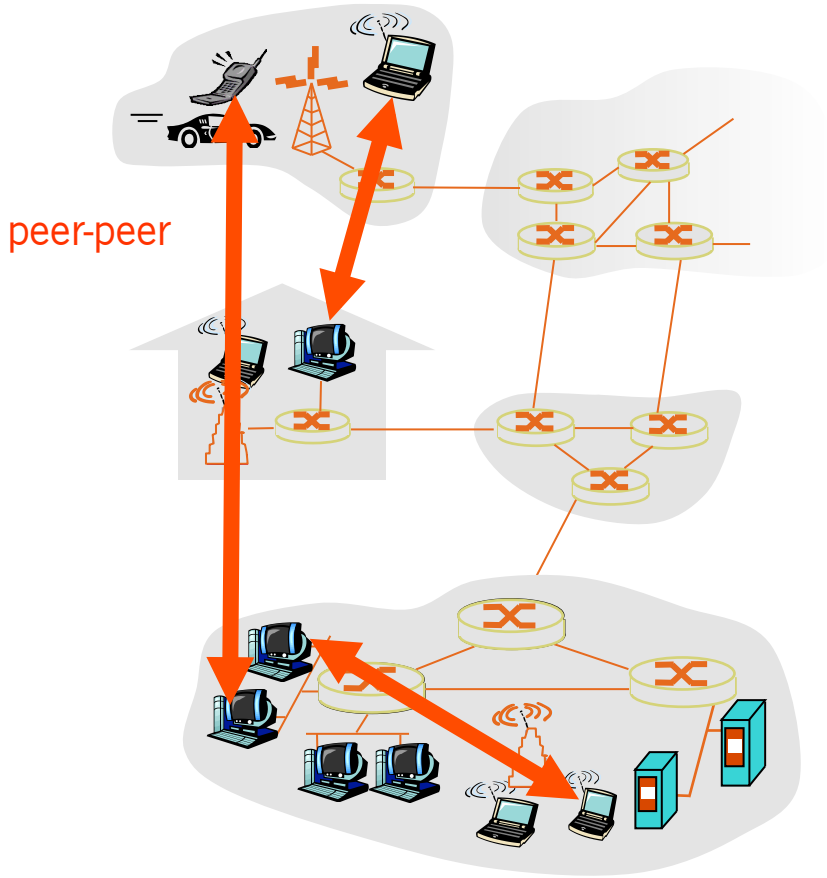
cliente:

- Comunica com o servidor
- Pode ter endereço IP dinâmico
- Não comunicam directamente um com o outro

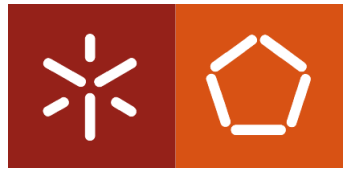
Arquitectura P2P pura

- Os dados estão distribuídos
- Sistemas terminais arbitrários comunicam directamente
- Os *peers* estão conectados temporariamente e mudam de endereço IP

São altamente escaláveis, mas difíceis de gerir

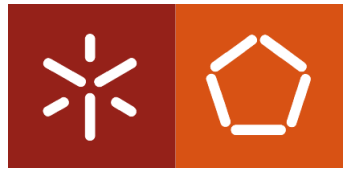


Híbridos: cliente-servidor e P2P



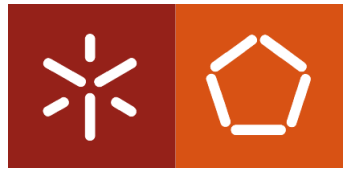
- **Skype**
  - Aplicação voice-over-IP P2P
  - Servidor centralizado: procura o endereço da parte remota
  - Conexão cliente-cliente: directa (não através do servidor)
- **Instant messaging**
  - A conversação entre dois utilizadores é P2P
  - Serviço centralizado: a presença e a localização do cliente é detectada
  - O utilizador regista o seu IP num servidor centralizado quando está online
  - O utilizador contacta o servidor central para encontrar o endereço IP da pessoa com quem quer manter uma conversação

Requisitos do serviço de transporte para as aplicações mais comuns



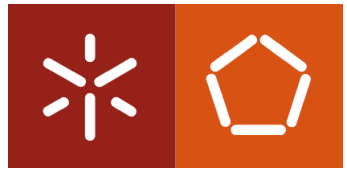
Aplicação	Perda de dados	Taxa de transmissão	Sensibilidade ao atraso
Trans. de ficheiros	não	elástica	não
e-mail	não	elástica	não
Páginas Web	não	elástica	não
áudio/vídeo em tempo-real	tolerante	áudio: 5kbps-1Mbps vídeo:10kbps-5Mbps	sim, 100's mseg
streaming áudio/vídeo	tolerante	idem	sim, poucos seg
Jogos interactivos	tolerante	poucos kbps	sim, 100 mseg
Mens. instantâneas	não	elástica	sim e não

Aplicações de Internet: protocolos de aplicação e de transporte

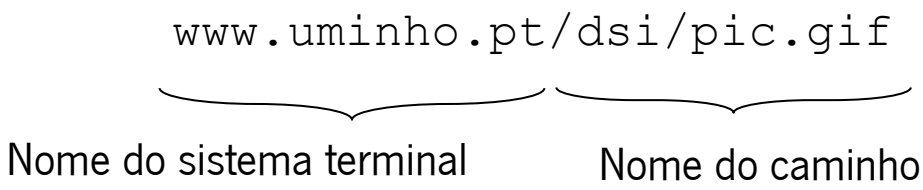


Aplicação	Protocolo do nível de transporte	Protocolo de transporte de suporte
e-mail	SMTP [RFC 2821]	TCP
Acesso remoto	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Transferência de fich.	FTP [RFC 959]	TCP
streaming multimedia	HTTP (ex Youtube), RTP [RFC 1889]	TCP or UDP
VoIP	SIP, RTP, proprietário (ex. Skype)	tipicamente UDP

Web e HTTP



- **Uma página Web consiste num conjunto de objectos**
- **Um objecto pode ser um ficheiro HTML, imagem JPEG, applet Java, ficheiro audio,...**
- **Uma página Web consiste num ficheiro HTML de base que inclui diversas referências a objectos**
- **Cada objecto é endereçável através duma URL**
- **Exemplo de URL:**



HTTP: Visão global

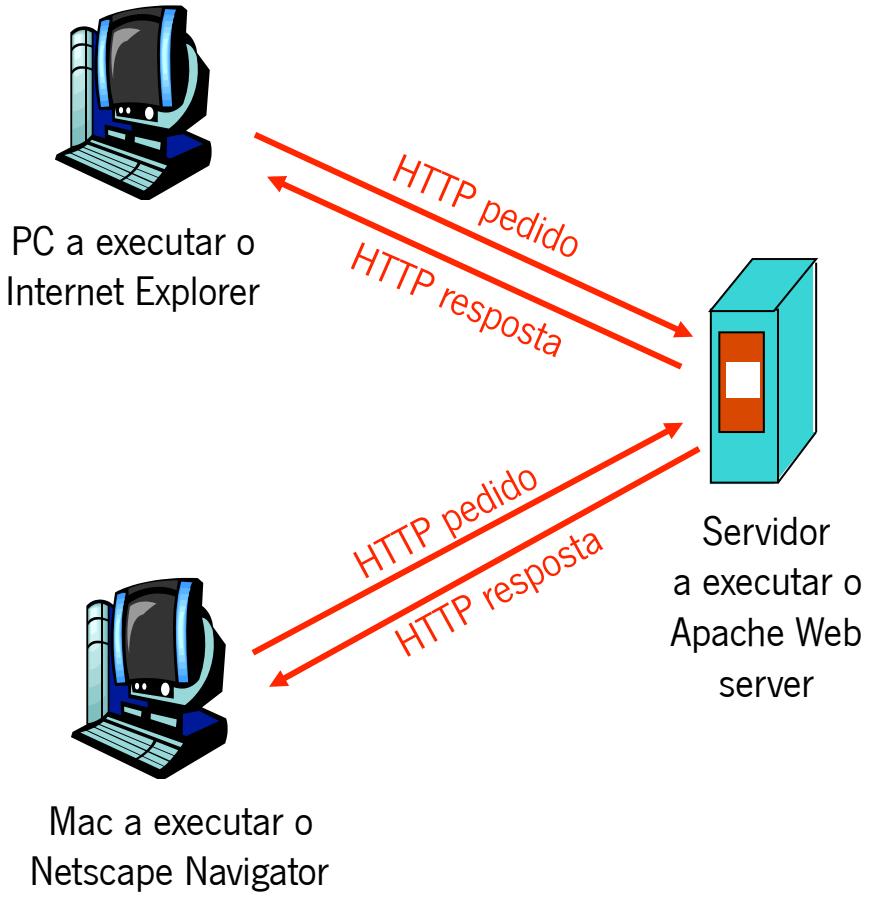
HTTP: hypertext transfer protocol

- **Protocolo do nível de aplicação da Web**
- **Modelo cliente/servidor**
  - *cliente:* browser que pede, recebe, e mostra os objectos Web
  - *servidor:* servidor Web server envia os objectos em resposta ao pedido do cliente

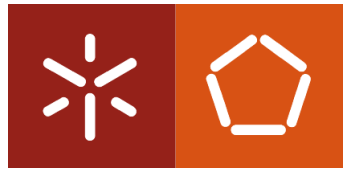
HTTP 1.0: RFC 1945

HTTP 1.1: RFC 2068

HTTP 2.0: RFC 7540



HTTP: Visão global (continuação)



Usa o TCP:

- O cliente inicia uma conexão TCP (cria um socket) ao servidor, porta 80
- O servidor aceita a conexão TCP do cliente
- Mensagens HTTP (mensagens do protocolo do nível da aplicação) são trocadas entre o browser (cliente HTTP) e o servidor Web (servidor HTTP)
- O TCP fecha a conexão

HTTP não guarda variáveis de estado

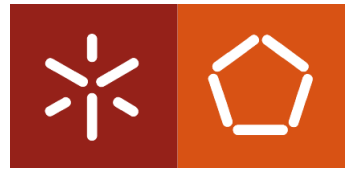
- O servidor não mantém informação sobre os pedidos dos clientes anteriores

Aparte

Protocolos que mantêm variáveis de estado são complexos!

- O historial dos pedidos(variáveis de estado) tem de ser mantidas/ actualizadas
- Se o servidor/cliente bloquearem, as variáveis de estado podem ficar inconsistentes

Conexões HTTP



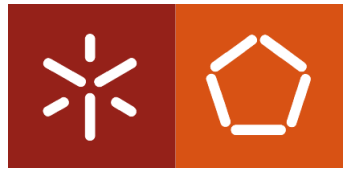
Não persistentes

- **Um objecto é enviado através de uma conexão TCP.**

Persistentes

- **Múltiplos objectos podem ser enviados numa única conexão TCP entre o cliente e o servidor.**

HTTP não persistente



Suponha que um utilizador introduz a URL `www.uminho.pt/dsi/home.index`

(contém texto e referências a 10 Imagens jpeg)

1a. O cliente HTTP inicia uma conexão TCP com o servidor HTTP (processo) www.uminho.pt na porta 80

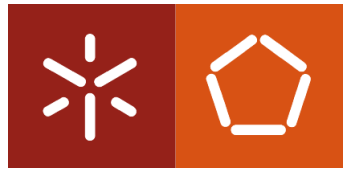
1b. O servidor HTTP do sistema terminal www.uminho.pt está à espera de conexões TCP na porta 80, o servidor aceita a conexão e envia uma notificação

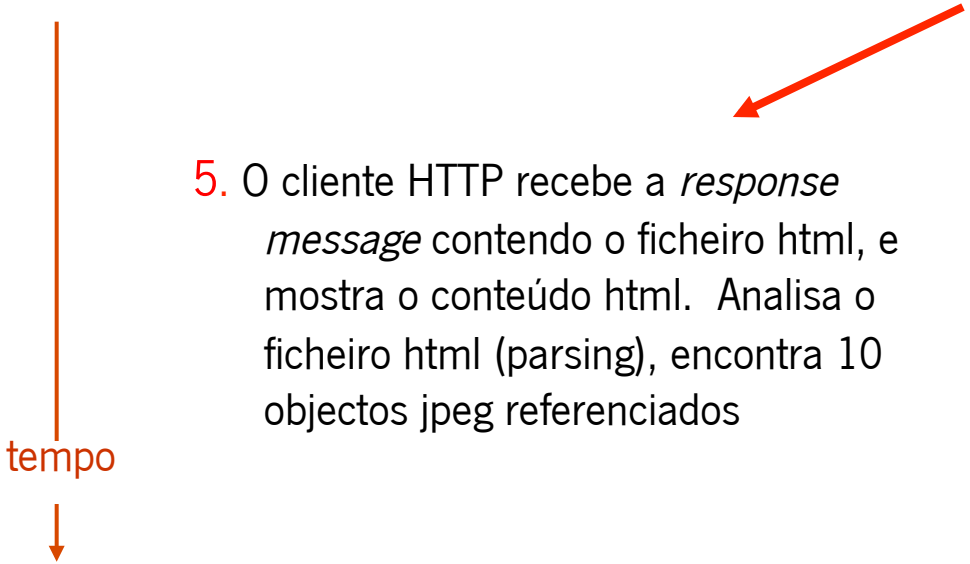
2. O cliente HTTP envia uma HTTP *request message* (que contém a URL) para o socket da conexão TCP.
A mensagem indica que o cliente pretende o objecto dsi/home.index

3. HTTP O servidor HTTP recebe a request message, forma uma *response message* que contém o objecto pedido, e envia a mensagem para o seu socket

tempo

HTTP não persistente (continuação)





4. O servidor HTTP fecha a conexão TCP.

5. O cliente HTTP recebe a *response message* contendo o ficheiro html, e mostra o conteúdo html. Analisa o ficheiro html (parsing), encontra 10 objectos jpeg referenciados

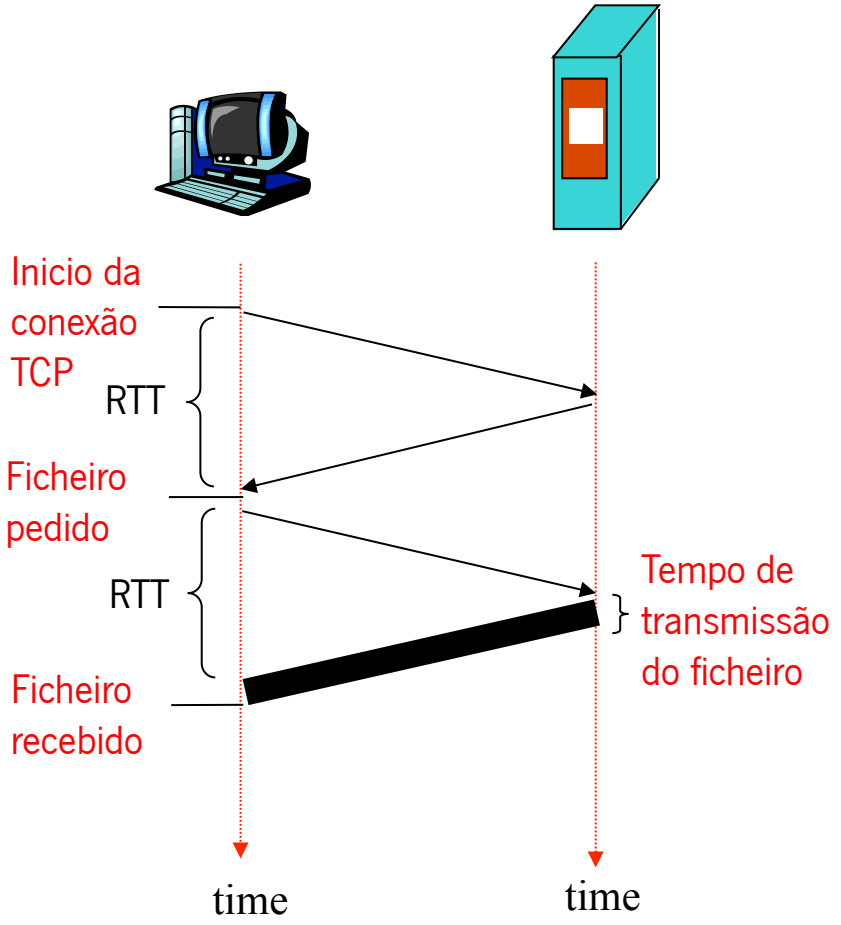
6. Os passos 1-5 são repetidos para cada um dos 10 objectos jpeg

HTTP não persistente: Tempo de Resposta

Tempo de resposta:

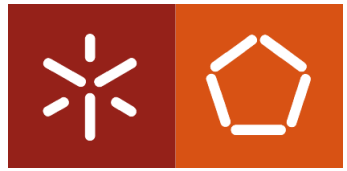
- **1 RTT para iniciar a conexão TCP**
- **1 RTT para o pedido HTTP e o tempo de transmissão do ficheiro**
- **Tempo de transmissão do ficheiro**

total = 2RTT+tempo de transmissão do ficheiro



HTTP

Persistente



HTTP não persistente:

- exige 2 RTTs por objecto
- O Sistema Operativo tem que reservar recursos para cada ligação TCP estabelecida
- Muitos browsers abrem ligações TCP paralelas para irem buscar os objectos referenciados

HTTP persistente:

- O servidor deixa a ligação aberta depois de enviar a mensagem de resposta
- Os pedidos HTTP posteiores são enviados através da mesma ligação

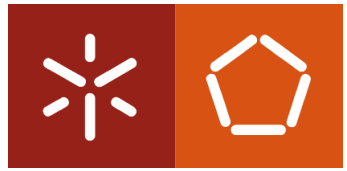
Persistente sem pipelining:

- O cliente envia um novo pedido apenas quando recebe a resposta ao anterior
- Um RTT por cada objecto referido

Persistente com pipelining:

- Modo por defeito no HTTP-1.1
- O cliente envia os pedido assim que os encontra no objecto referenciado
- No mínimo é consumido um RTT por todos os objectos referenciados

Mensagens HTTP



- **2 tipos de mensagens HTTP: *request*, *response***
- **Mensagem HTTP - request:**
  - ASCII

Request Line (Métodos: GET, POST, HEAD)

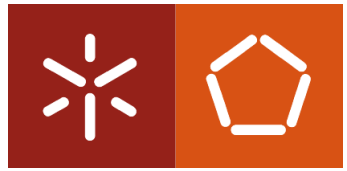
Linhas do cabeçalho (HEAD)

Carriage return, line feed: indica o final do cabeçalho

```
GET /dsi/page.html HTTP/1.1
Host: www.uminho.pt
User-agent: Mozilla/4.0
Connection: close
Accept-language:pt

extra carriage return,line feed
```

Entradas para Formulários



Método Post:

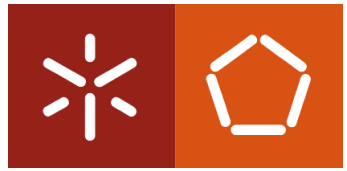
- **A página web inclui um formulário com entradas**
- **As entradas vão no body do ficheiro HTML e são lidas no servidor**

Método URL:

- **Usa o método GET**
- **As entradas vão na URL:**

```
www.somesite.com/animalsearch?monkeys&banana
```

Mensagem HTTP – response



Linha de estado (protocolo código de estado frase do código)

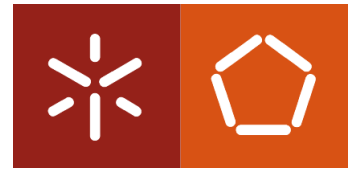
Linhas do cabeçalho

dados, ex., ficheiro HTML pedido

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

Dados dados dados dados dados ...
```

Códigos de Estado



Na primeira linha da mensagem *HTTP-response*
Alguns códigos:

200 OK

- Pedido bem sucedido

301 Moved Permanently

- O objecto pedido mudou de localização

400 Bad Request

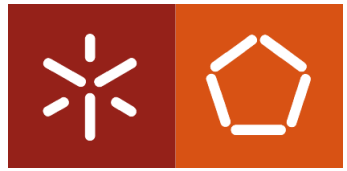
- A mensagem de request não foi reconhecida pelo servidor

404 Not Found

- O documento pedido não foi encontrado neste servidor

505 HTTP Version Not Supported

Cookies: informação de estado



A maioria dos sites Web usa cookies

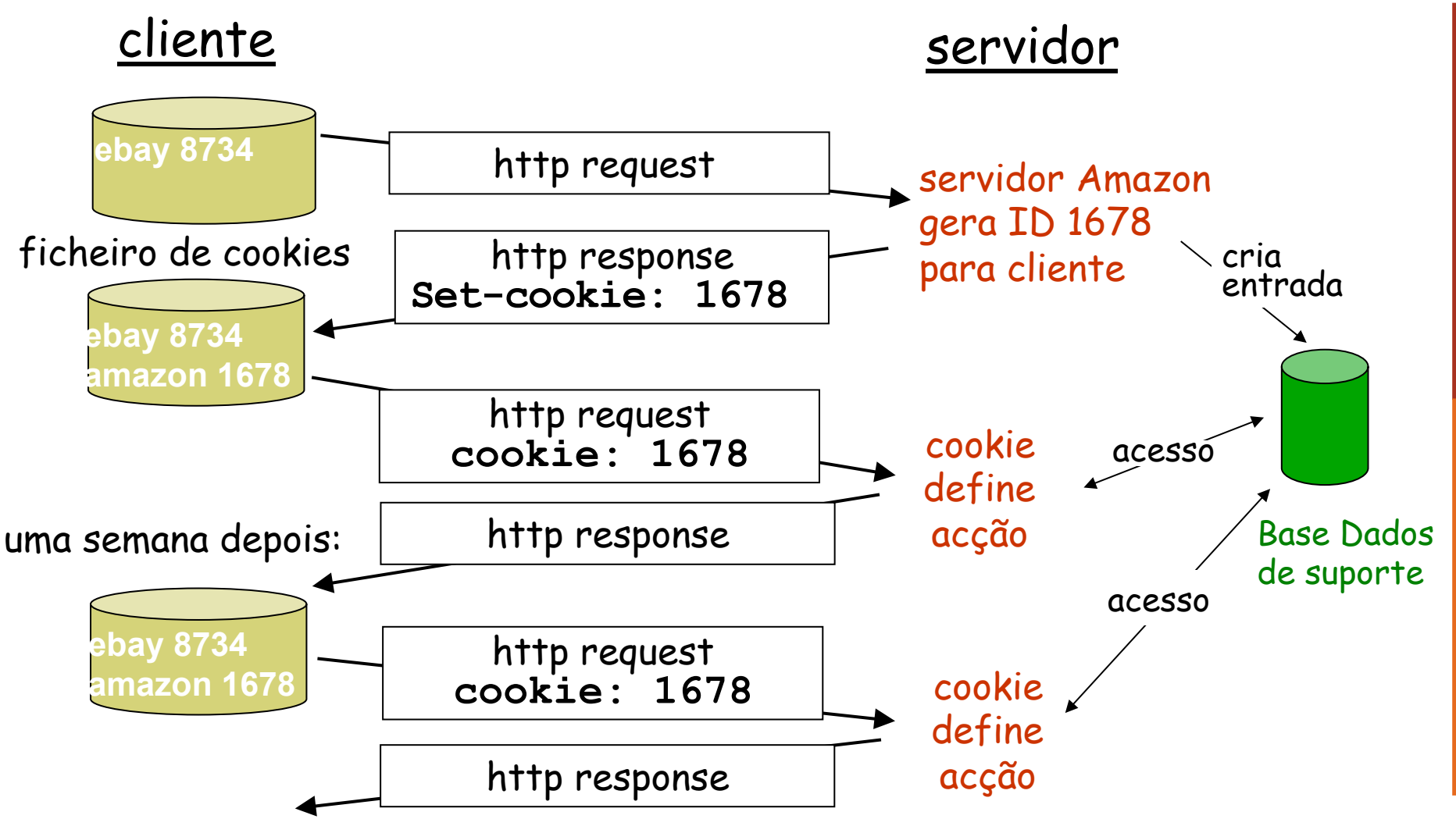
Quatro componentes:

1) Linha com *cookie* no cabeçalho da mensagem *HTTP response*
2) Linha com *cookie* no cabeçalho da mensagem HTTP *request*
3) Ficheiro com *cookies* mantido na máquina do utilizador, gerido pelo seu browser
4) Uma base de dados de suporte do lado servidor Web

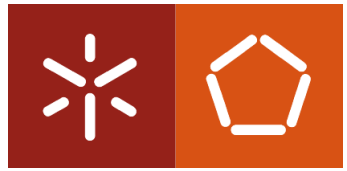
Exemplo:

- **Susana acede sempre à Internet a partir do seu PC**
- **visita um site de comércio electrónico pela primeira vez**
- **quando o primeiro pedido chega ao servidor Web, o servidor gera:**
  - Um Identificador (ID) único
  - Uma entrada na base de dados de suporte para esse ID

Cookies: informação de estado



Cookies: informação de estado



O que os cookies permitem:

- autorização
- cabaz de compras
- sugestões ao utilizador
- informação de sessão por utilizador (ex: Web e-mail)

efeitos colaterias

Os Cookies e a privacidade:

- os cookies ensinam muito aos servidores a respeito dos utilizadores e seus hábitos
- o utilizador pode fornecer nome e e-mail ao servidor

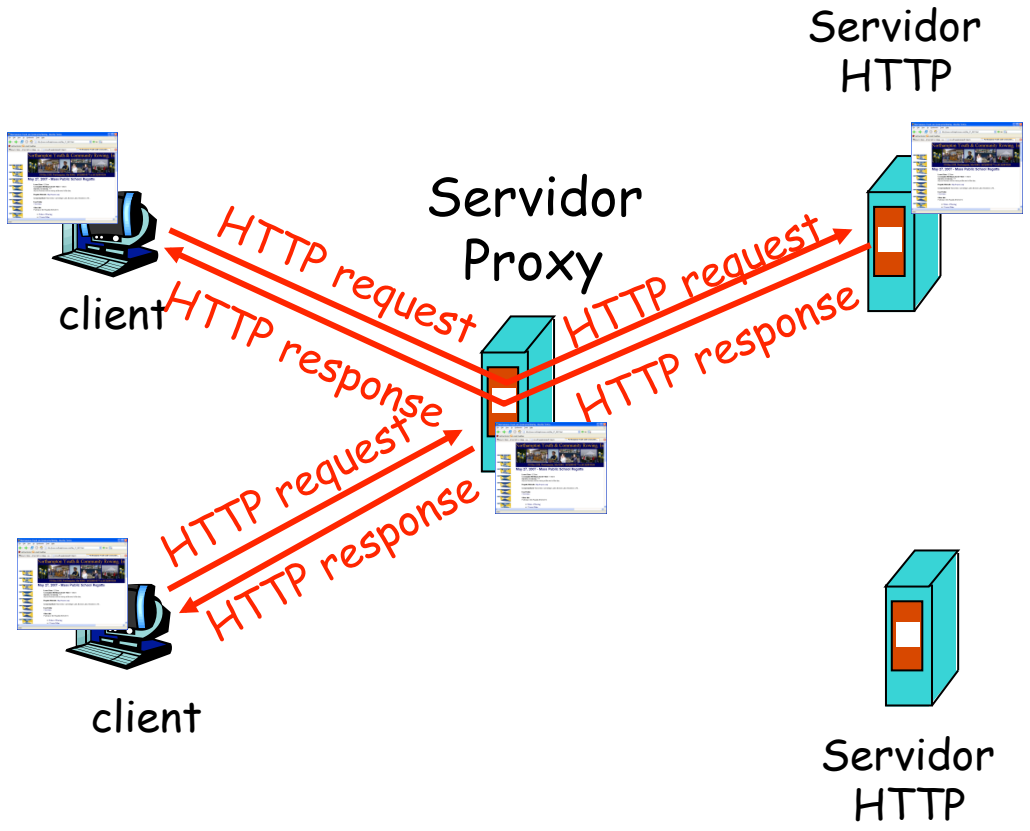
Como manter "estado":

- entidades protocolares: guardam estado por emissor/receptor entre transacções distintas
- cookies: forma como as mensagens http transportam a informação de estado

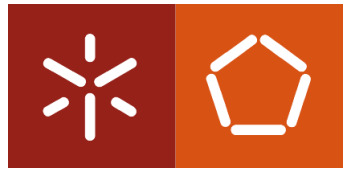
Web caches (servidor proxy)

Objectivo: satisfazer o pedido do cliente sem envolver o servidor HTTP alvo

- **O utilizador configura o cliente HTTP (browser) para aceder à Web através de um servidor proxy**
- **O browser envia todas as *HTTP request messages* para o servidor proxy**
  - Se o objecto requerido está na cache do proxy o servidor proxy retorna o objecto
  - Senão o servidor proxy contacta o servidor HTTP alvo, reenvia-lhe a *HTTP request message*, aguarda a resposta que retorna ao browser



Web caching



- o servidor proxy/cache tem que actuar simultaneamente como cliente e como servidor
- são tipicamente instalados pelos ISP ou pelas próprias instituições (universidades, empresas, ISP residenciais, etc)

Porquê *Web caching*?

- reduz o tempo de resposta para os pedidos dos clientes
- reduz o tráfego nos links de acesso ao exterior (os mais problemáticos para a instituição).
- Internet está povoada de *caches*: permitem que fornecedores de conteúdos mais "pobres" disponibilizem efectivamente os seus conteúdos (mas isso também as redes de partilha de ficheiros P2P)

Exemplo de Caching

Pressupostos

Tamanho médio dos objectos = 100,000 bits

- **Taxa média de pedidos efectuados pelos blowsers da instituição para servidores HTTP = 15/seg**
- **Tempo médio de atraso desde o pedido HTTP até à chegada da resposta = 2 sec**

Consequências

- **Utilização da LAN = 15%**

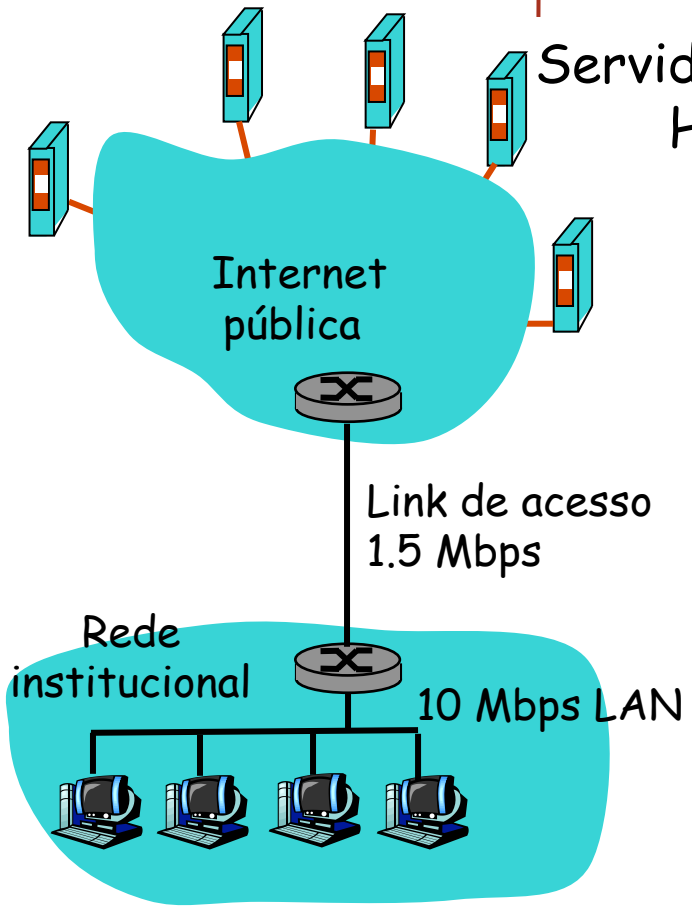
(15 pedidos/seg).(100Kbits/pedido)/(10Mbps)

- **Utilização do Link de acesso = 100%**

(15 pedidos/seg).(100Kbits/pedido)/(1.5Mbps)

- **Total delay = Internet delay + access delay + LAN delay**

= 2 seg + minutos + milisegundos



Exemplo de Caching (cont)

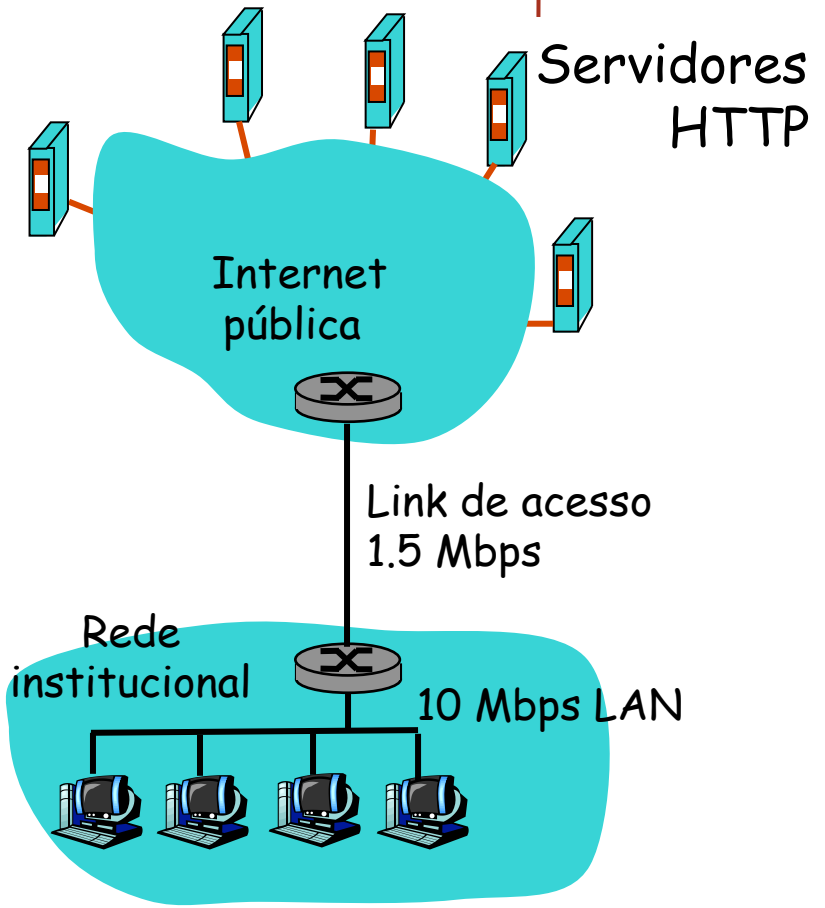
Solução possivel

- **Aumentar a largura de banda do link de acesso para 100 Mbps**

Consequência

- **Utilização da LAN = 15%**
- **Utilização do Link de Acesso = 15%**
- **Total delay = Internet delay + access delay + LAN delay**

  = 2 segs + msegs + msegs
- **É habitualmente muito dispendioso fazer o upgrade do link de acesso de uma instituição**



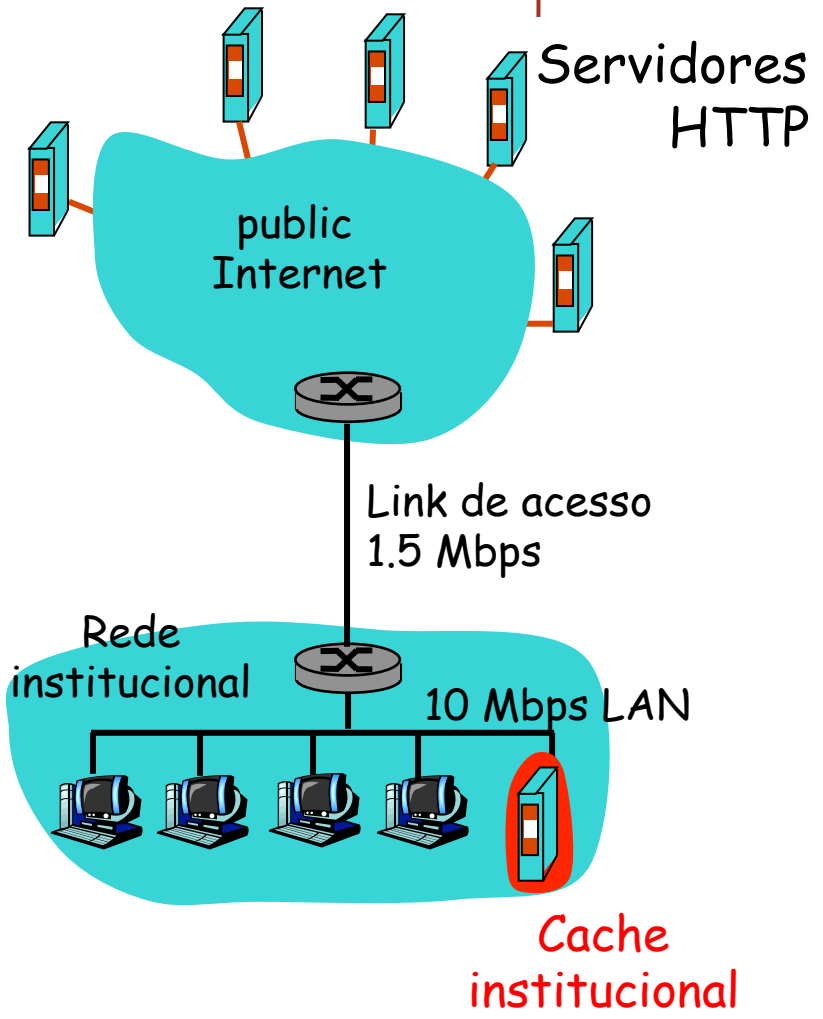
Exemplo de Caching (cont)

Solução possivel: instalar o Web Proxy

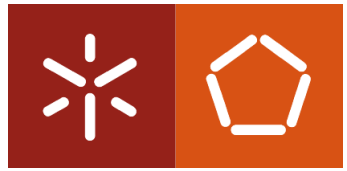
- **Se a taxa de acerto for de 0.4**

Consequências

- **40% dos pedidos serão satisfeitos imediatamente**
- **60% dos pedidos terão que ser redireccionados para o servidor HTTP respectivo**
- **A utilização do link de acesso será reduzida para 60% resultando em atrasos negligenciáveis (10 msec)**
- **total avg delay = Internet delay + access delay + LAN delay = .6*(2.01) secs + .4*milliseconds < 1.4 secs**



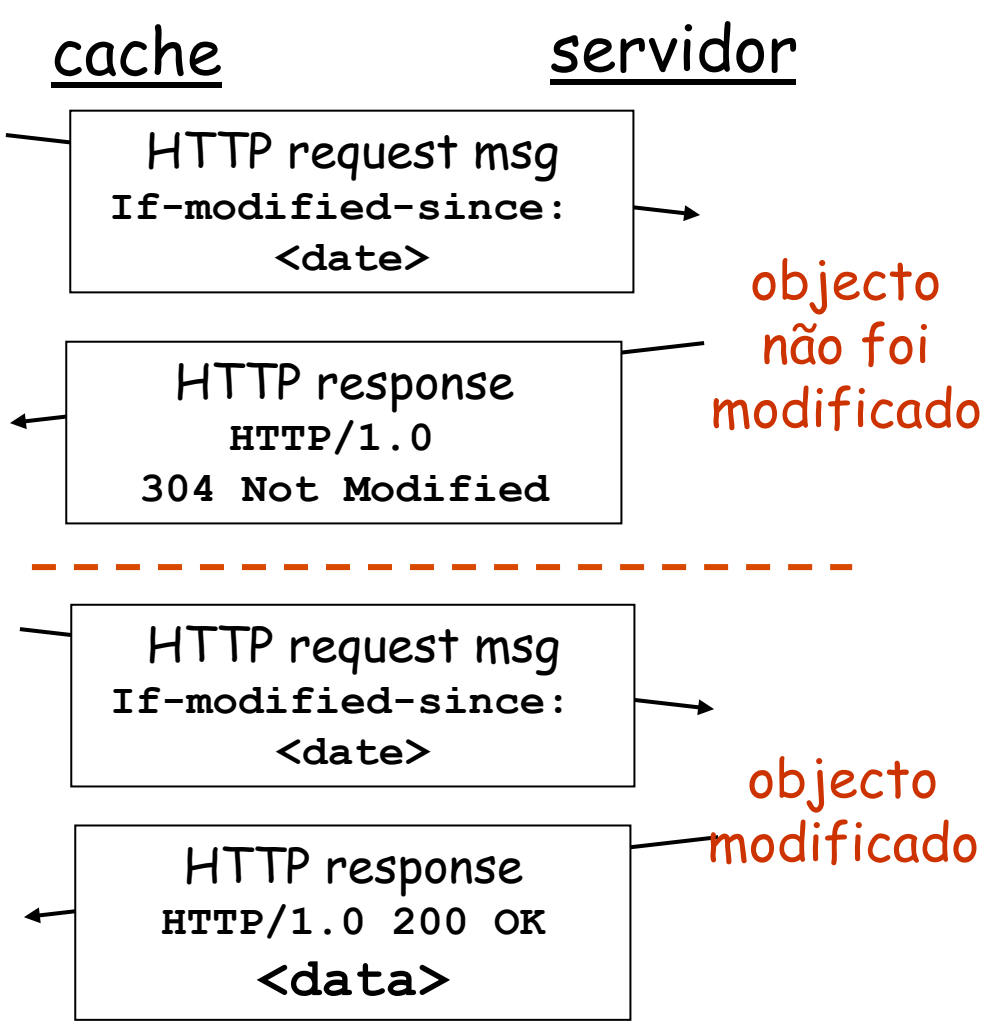
GET Condicional



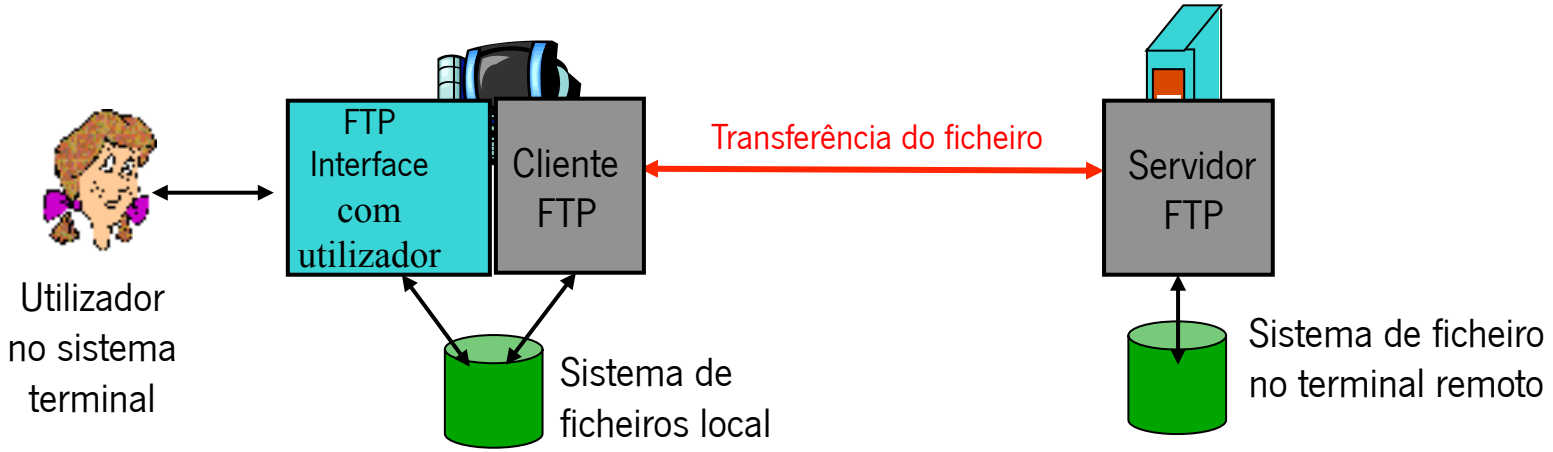
- **Objectivo: não enviar o objecto se a cópia mantida em cache está actualizada**
- ***cache*: inclui no cabeçalho do pedido HTTP, a data da cópia guardada na cache**

  `If-modified-since: <date>`
- ***servidor*: resposta não contém nenhum objecto se a cópia mantida em cache estiver actualizada:**

  `HTTP/1.0 304 Not Modified`



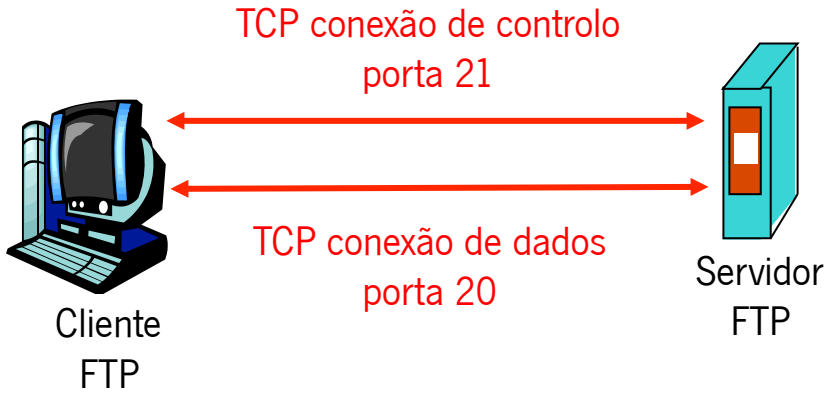
FTP: protocolo para transferência de ficheiros



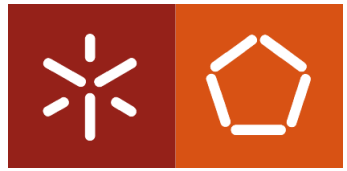
- **Transferir ficheiros para/do sistema terminal remoto**
- **Modelo cliente/servidor**
  - *cliente:* parte que inicia a transferência (para/do ST remoto)
  - *servidor:* sistema terminal remoto
- **ftp: RFC 959**
- **Servidor FTP: porta 21**

FTP: separa a conexão de dados da conexão de controlo

- **O cliente FTP contacta o servidor FTP na porta 21, o TCP é o protocolo de transporte**
- **O cliente é autorizado através da conexão de controlo**
- **O cliente navega na directoria do ST remoto através do envio de comandos pela conexão de controlo**
- **Quando o servidor recebe o comando para a transferência do ficheiro abre uma 2ª conexão TCP (para o ficheiro) para o cliente**
- **Após a transferência do ficheiro o servidor fecha a conexão**



FTP: comandos e respostas



Comandos simples:

- **Enviar como texto ASCII através da conexão de controlo**
- `USER nome-utilizador`
- `PASS palavra-passe`
- `LIST` **lista os ficheiros da directoria**
- `RETR nome-ficheiro` **para obter ficheiro**
- `STOR nome-ficheiro` **guarda o ficheiro no sistema terminal remoto**

Códigos de resposta simples:

- **Código de estado e frase de estado (como no HTTP)**
- `331 Username OK, password required`
- `125 data connection already open; transfer starting`
- `425 Can't open data connection`
- `452 Error writing file`

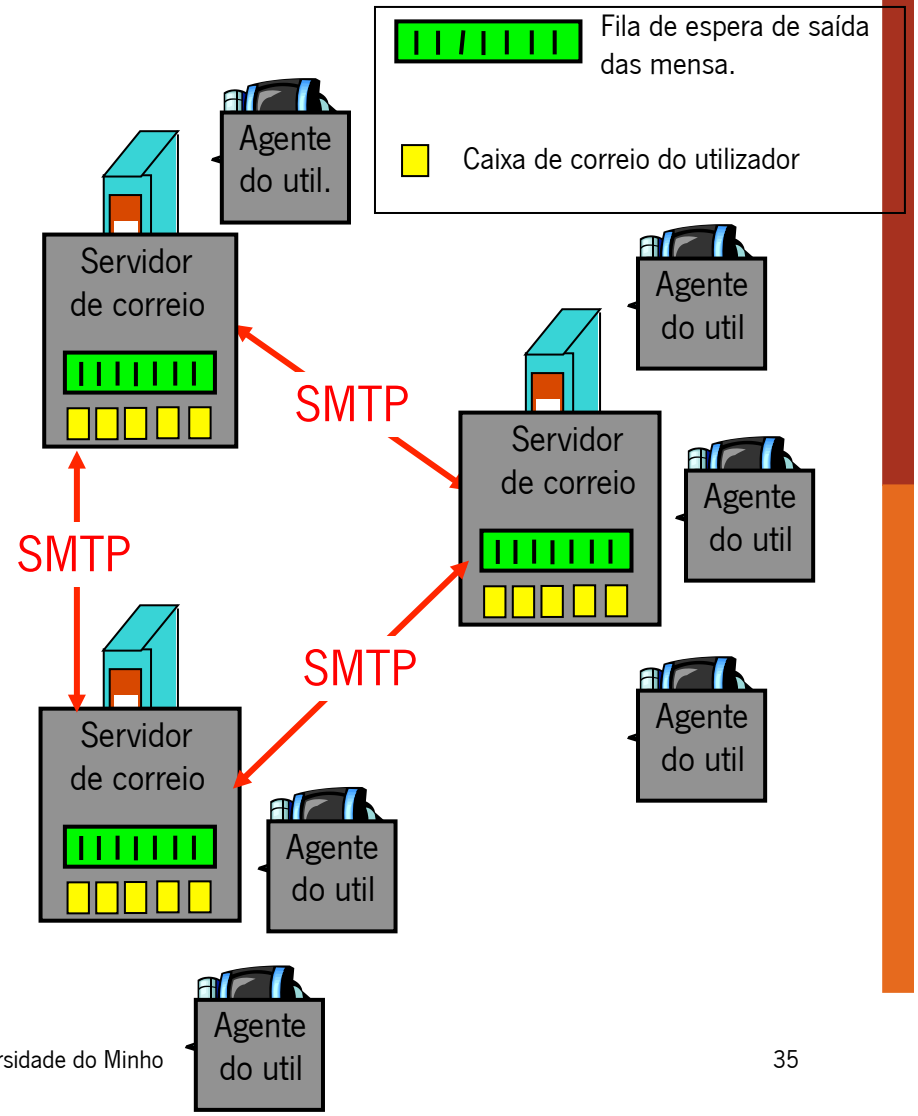
Correio electrónico

Os 3 principais componentes:

- **Agentes do utilizador**
- **Servidores de correio**
- **simple mail transfer protocol: SMTP**

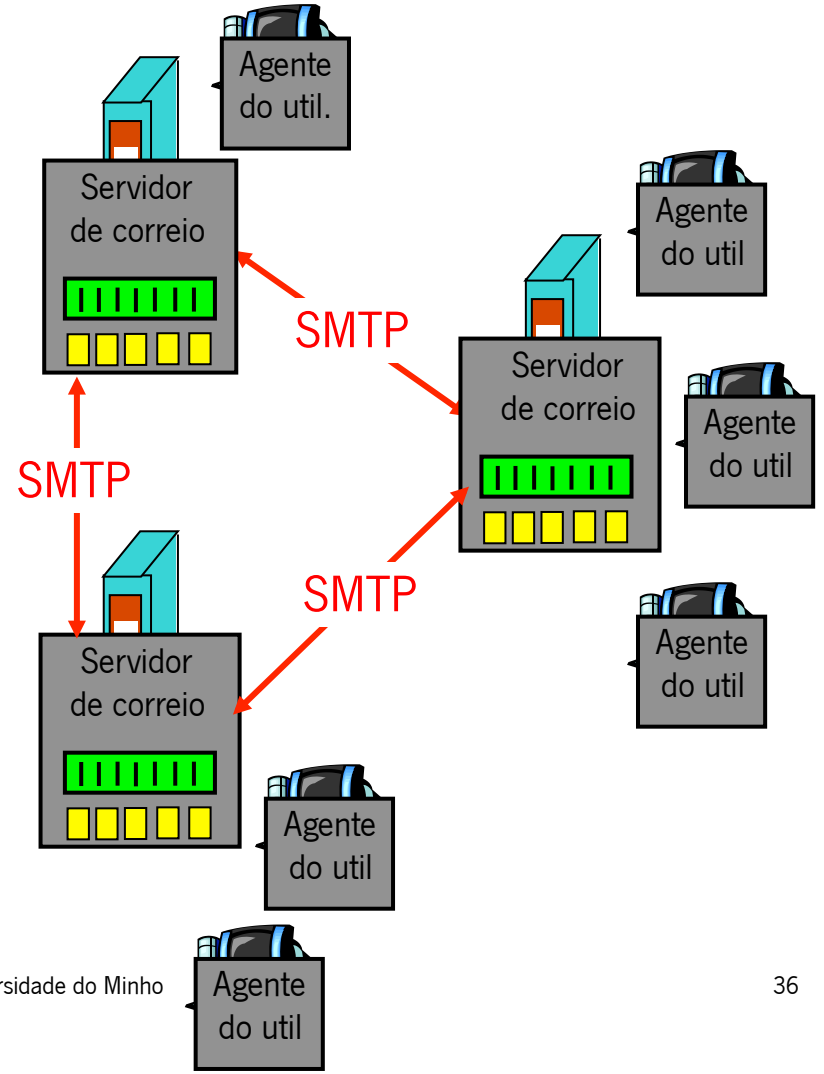
Agente do utilizador:

- **"Faz a leitura do correio"**
- **composição, edição e leitura da mensagens de correio**
- **Ex. Eudora, Outlook, Mozilla Thunderbird**
- **Mostra as mensagens que estão guardadas no servidor de correio**

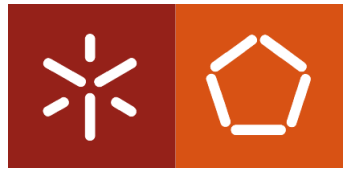


Servidor de correio electrónico

- **Caixas de correio** contêm as mensagens que se destinam aos utilizadores
- **Fila de espera das mensagens** para as mensagens que vão ser enviadas
- **Protocolo SMTP** utilizado pelo servidores de correio para trocarem mensagens de correio entre si, e para as receberem dos agentes

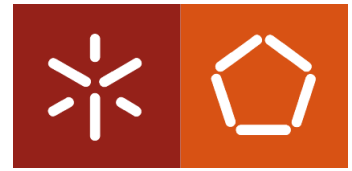


SMTP [RFC 2818]

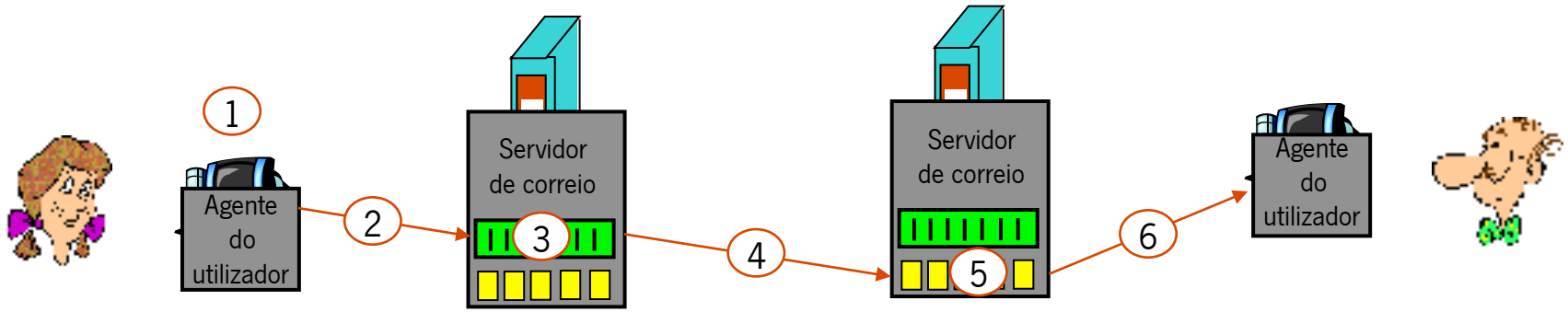


- **Usa o TCP para fazer a transmissão fiável das mensagens desde do cliente até servidor (porta 25)**
- **Transferência directa: do servidor emissor para o servidor receptor**
- **A transferência passa por 3 fases**
  - *handshaking* (cumprimento)
  - transferência das mensagens
  - término
- **Iterações com mensagens de command/response**
  - commands: texto ASCII
  - response: código e frase de estado
- **As mensagens têm de estar representadas em 7-bit ASCII**

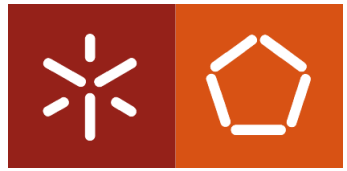
Exemplo: mensagem da Maria para o António



1) A Maria usa o agente de utilizador para escrever e enviar a mensagem para `antonio@uminho.pt`
2) O agente de utilizador da Maria envia a mensagem para o seu servidor de correio; colocando a mensagem na fila de espera
3) O lado de cliente do SMTP cria uma conexão TCP com o servidor de correio do António
4) O SMTP cliente envia a mensagem da Maria através da conexão TCP
5) O servidor de correio do António coloca a mensagem na caixa de correio do António
6) O António usa o seu agente de utilizador para ler a mensagem

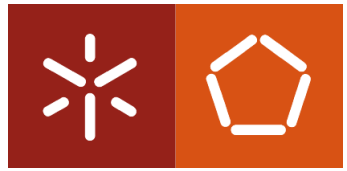


SMTP: resumo

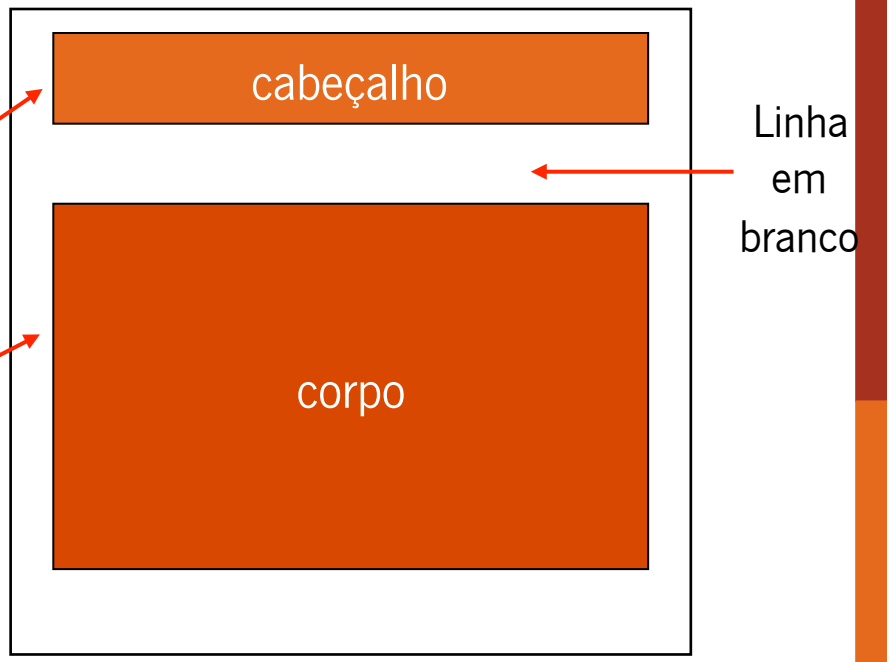


- O SMTP usa conexões persistentes
- O SMTP requer que as mensagens (cabeçalho & corpo) sejam em 7-bit ASCII
- O servidor SMTP usa CRLF.CRLF (CRLF-linha em branco) para determinar o fim da mensagem

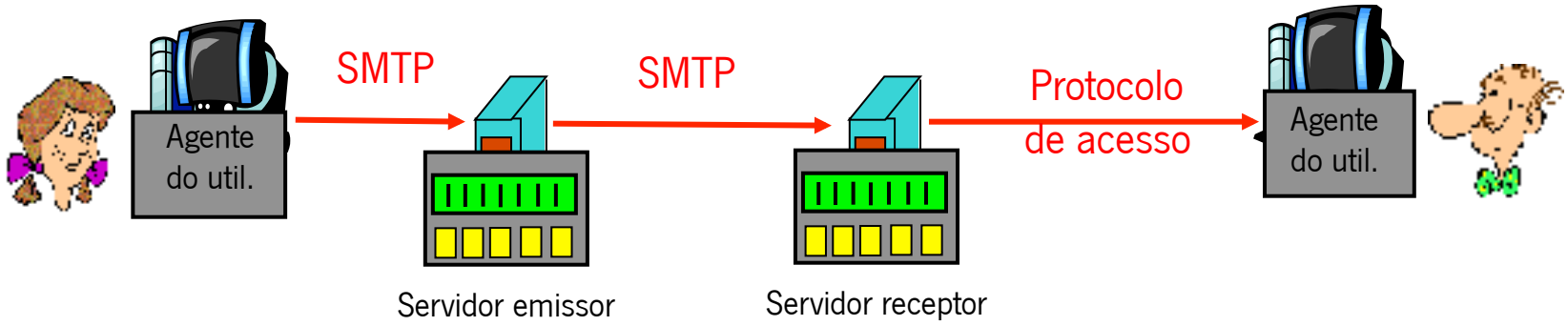
Formato das mensagens



- **SMTP: protocolo utilizado para a troca de mensagens de email**
- **RFC 822: Formato da mensagem:**
- **Linhas de cabeçalho ex:**
  - Para:
  - De:
  - Assunto:
- **Corpo**
  - O texto da mensagem", em caracteres ASCII

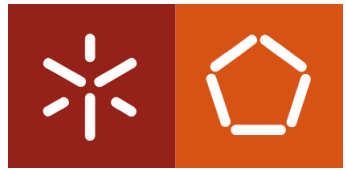


Protocolos de acesso ao correio electrónico



- **SMTP: Entrega/armazenamento no servidor receptor**
- **Protocolo de acesso ao correio: entrega das mensagens do servidor receptor ao agente do utilizador**
  - POP: Post Office Protocol [RFC 1939]
    - Autorização (agente <–> servidor) e download
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - mais funcionalidades (mais complexo)
    - manipulação das mensagens armazenadas no servidor
  - HTTP: gmail, Hotmail, Yahoo! Sapo, etc.

Protocolo POP3



Fase autorização

- **Comandos do cliente:**
  - `Utilizador:nome`
  - `Palavra-passe:` password
- **Respostas do servidor**
  - `+OK`
  - `-ERR`

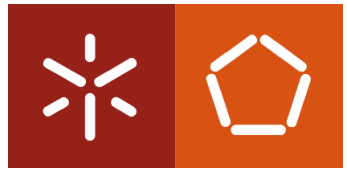
```
S: +OK POP3 server ready
C: user António
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

Fase de transferência, cliente:

- `list:` **lista o número de mensagens**
- `retr:` **entrega a mensagem**
- `dele:` **elimina**
- `Quit: sair`

```
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
```

DNS: *Domain Name System*



Pessoas: usam muitos identificadores:

- BI, NIF, passaporte

Internet: sistemas terminais e routers usam:

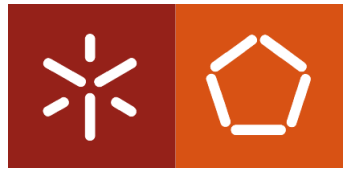
- Endereços IP (32 bit) – são usados para endereçar datagramas
- "nomes", ex, www.uminho.pt - são usados pelos humanos

Q: Como se mapeiam os endereços IP nos nomes?

Domain Name System:

- **Base de dados distribuída implementada numa hierarquia de muitos servidores de nomes**

DNS



Serviços DNS:

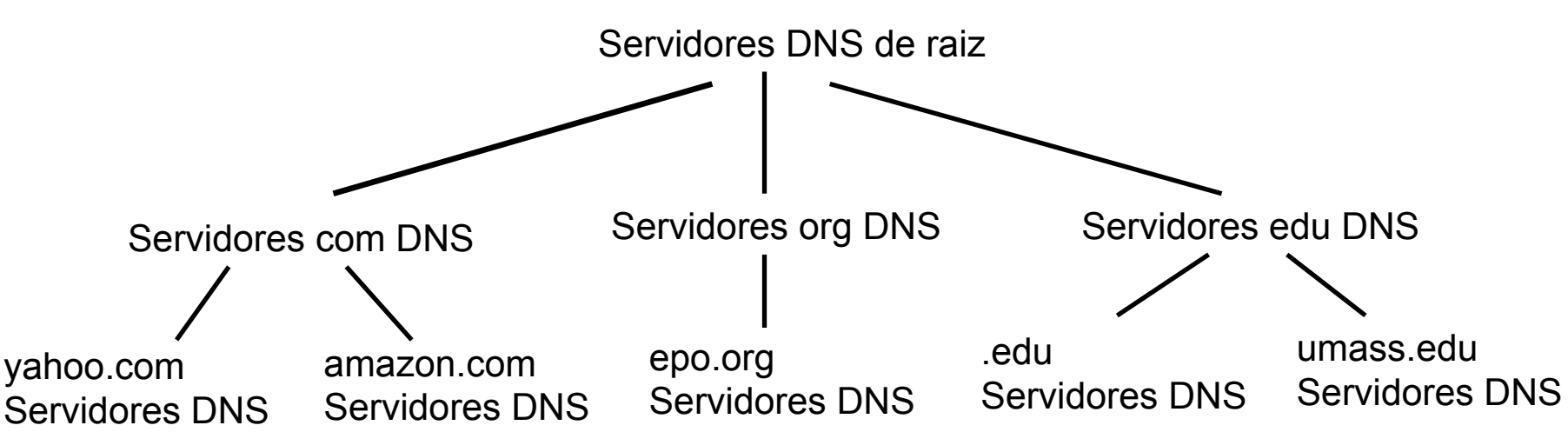
- **hostname:**
  - para representar o endereço IP
- **host aliasing:**
  - canónico, pseudónimos
- **mail server aliasing**
- **distribuição da carga:**
  - servidores web replicado: conjunto de endereços IP para um único nome canónico

Porque não se utiliza um DNS centralizado?

- **um único ponto de falha**
- **volume de tráfego**
- **distância da base de dados centralizada**
- **manutenção**

Portanto não escala!

DNS: base de dados hierárquica e distribuída

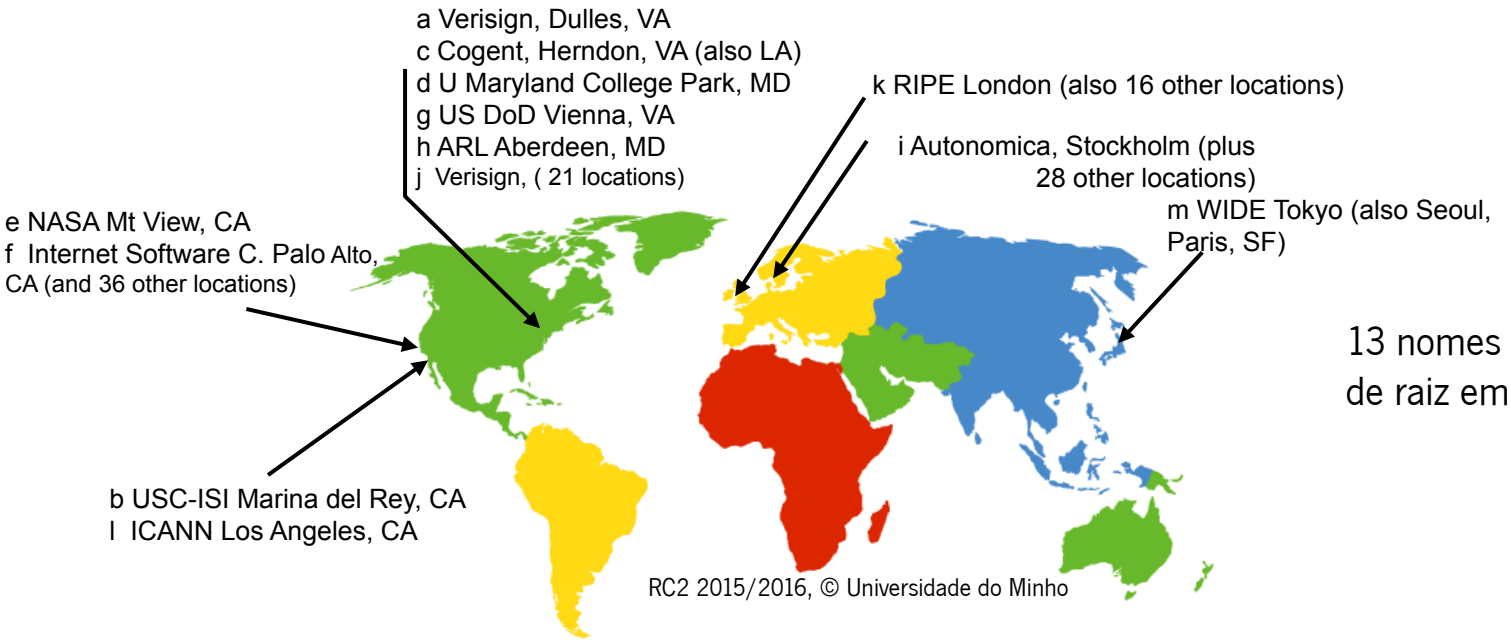


O Cliente pretende o IP para www.amazon.com; 1ª aproximação:

- **O cliente pede ao servidor de raiz para encontrar o servidor DNS .com**
- **O cliente pede ao servidor DNS .com para obter o servidor DNS amazon.com**
- **O cliente pede ao servidor DNS amazon.com para obter o endereço IP do nome www.amazon.com**

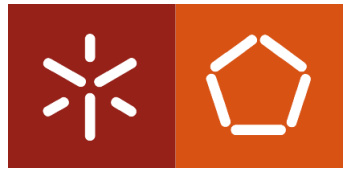
DNS: servidores de raiz

- **São contactados pelos servidores locais quando estes não conseguem resolver os nomes**
- **Nome do servidor de raiz:**
  - São contactados quando o nome é desconhecido pelo servidor local
  - Retorna o mapeamento do nome para o servidor local



13 nomes de servidores de raiz em todo mundo

DNS: TLD e servidores autorizados



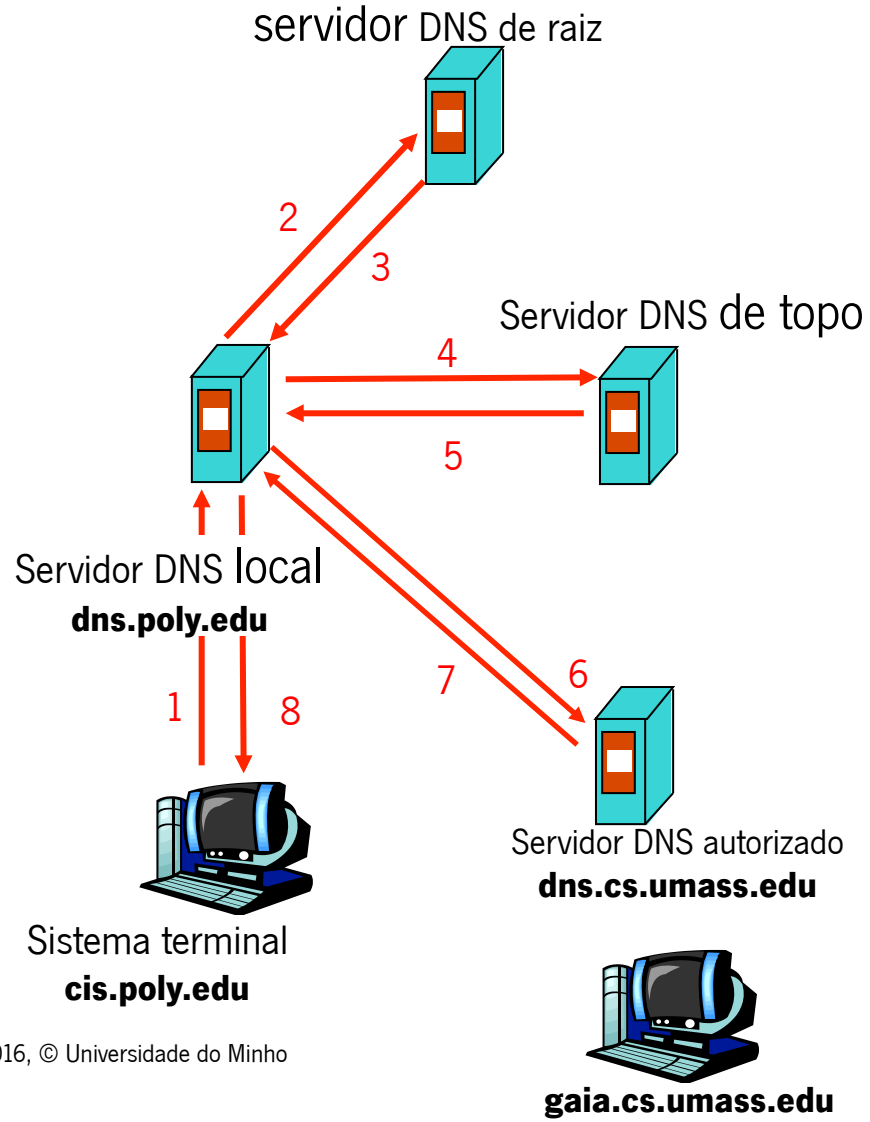
- **Top-level domain (TLD) servers:**
  - são responsáveis por com, org, net, edu, etc, e pelos domínios dos países pt, uk, fr, ca, jp, ...

- **Servidores de DNS autorizados:**
  - servidores DNS de organizações, estão autorizados a fornecer o mapeamento do hostname para IP para os servidores da organização (ex. Web, mail).
  - podem ser mantidos pela organização ou pelo ISP

DNS: resolução de nomes

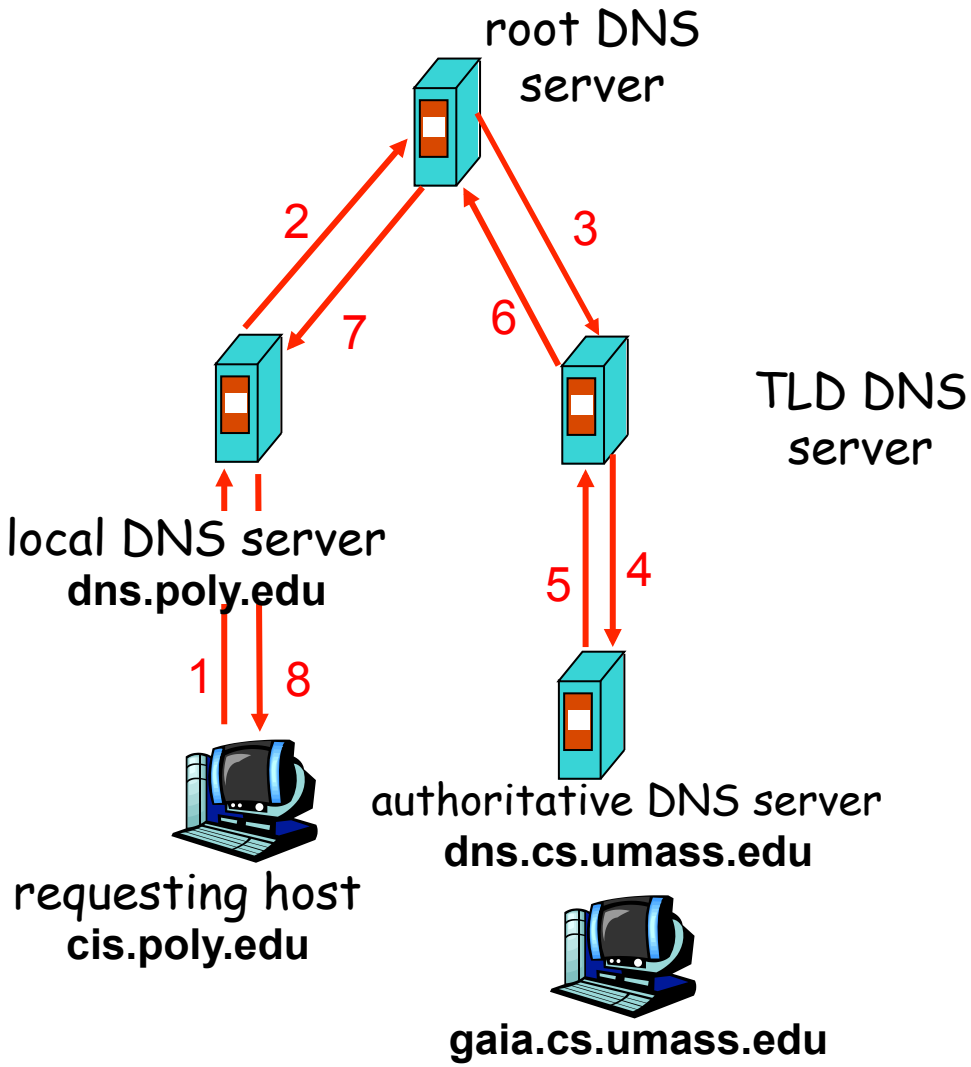
- **O sistema terminal com o nome cis.poly.edu pretende o endereço IP do nome gaia.cs.umass.edu**

- **Pedido interactivo:**
  - o servidor contactado retorna o nome do servidor que deverá ser contactado: "Eu não sei este nome, mas pergunta a este servidor"

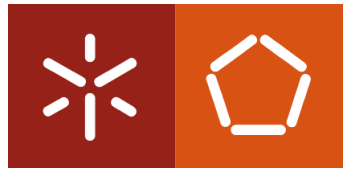


DNS: resolução de nomes

- **Pedido recursivo**
  - o servidor contactado contacta com outros servidores para retornar a resposta pretendida

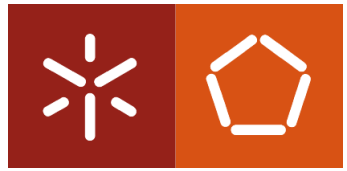


DNS: *caching* e actualização dos registos



- **Depois de aprender o mapeamento, o servidor de nomes guarda em *cache* o mapeamento**
  - os registos em *cache* expiram passado algum tempo e são eliminados
  - os servidores TLD normalmente estão na *cache* dos servidores de nomes locais
    - evitando por isso pedidos aos servidores de raiz

DNS: registos



DNS: Base de dados distribuída que armazena registos de recursos - resource records (RR)

Formato RR: **(nome, valor, tipo, ttl)**

- Tipo=A
  - **Nome** é o nome do sistema terminal
  - **Valor** é o endereço IP
- Tipo=NS
  - **Nome** é o domínio (ex. uminho.pt)
  - **Valor** é o nome do sistema terminal que está autorizado a ser o servidor de nomes deste domínio
- Tipo=CNAME
  - **Nome** é o pseudónimo.
  - **Valor** é o nome canónico

  Por ex: www.ibm.com é realmente servereast.backup2.ibm.com
- Tipo=MX
  - **valor** é o nome do servidor de correio associado com o **Nome**