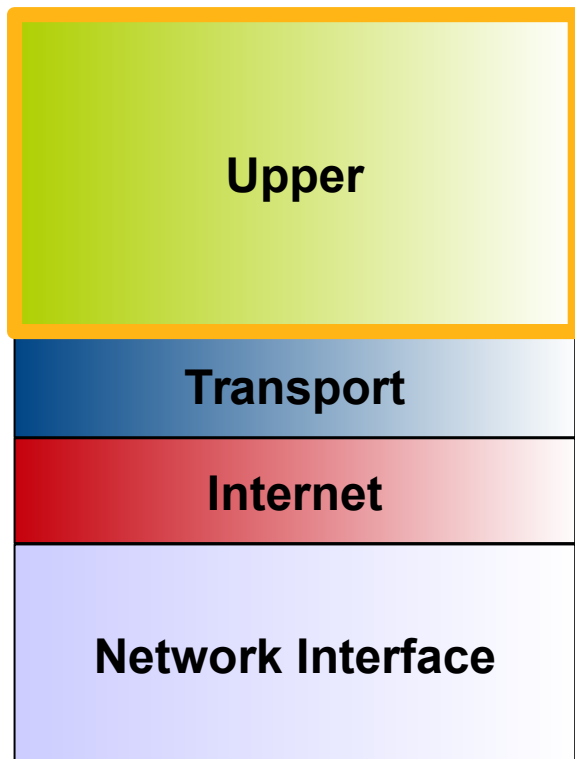


# Aplicações Modelo Cliente-Servidor

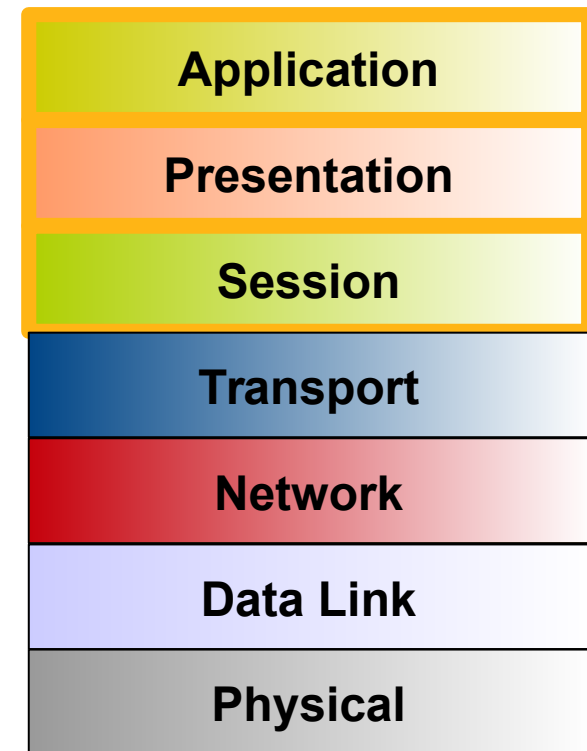
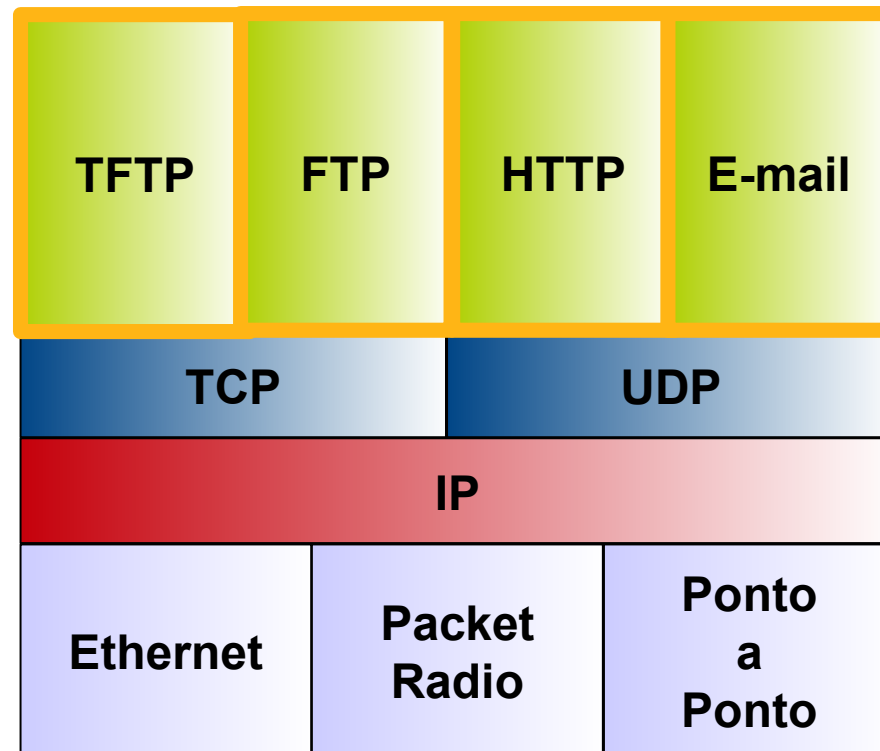
**Redes e Serviços**

**Licenciatura em Engenharia Informática  
DETI-UA**

# Modelo de referência TCP/IP



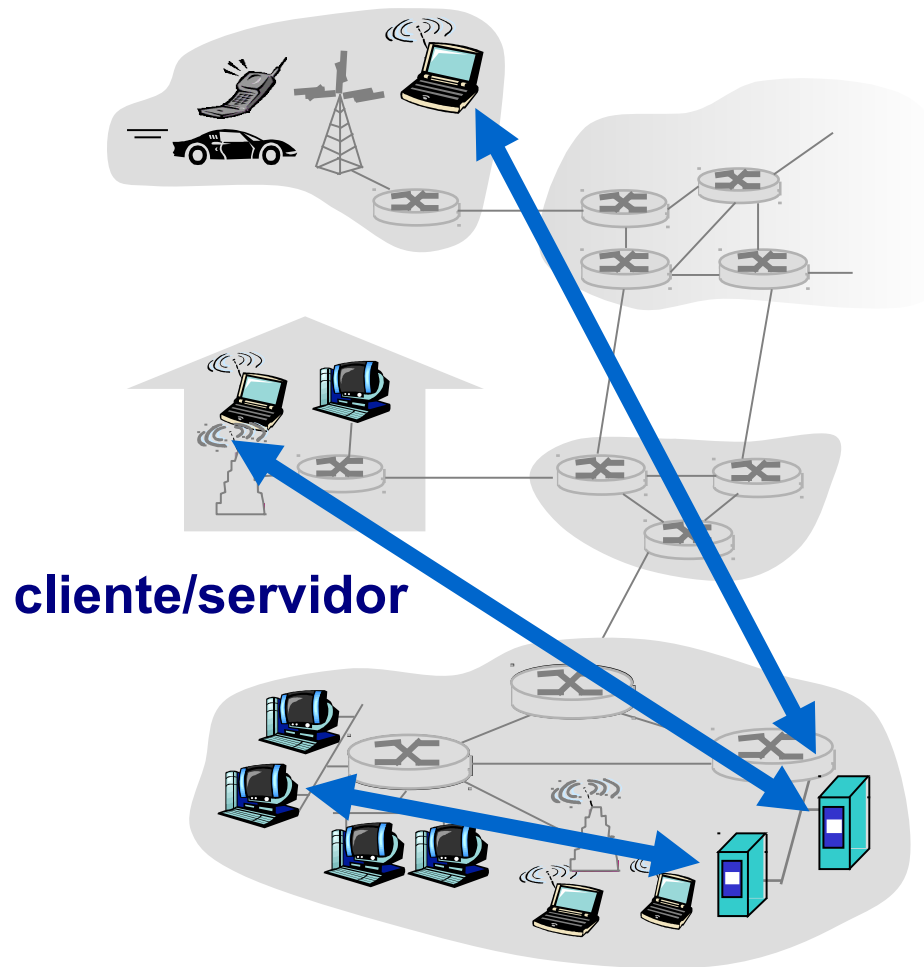
**TCP/IP**



**OSI**



# Arquitectura Cliente-Servidor



## Servidor:

- ◆ Sempre ligado (always on)
- ◆ Endereço IP permanente ou associação permanente entre nome e IP dinâmico

## Clientes:

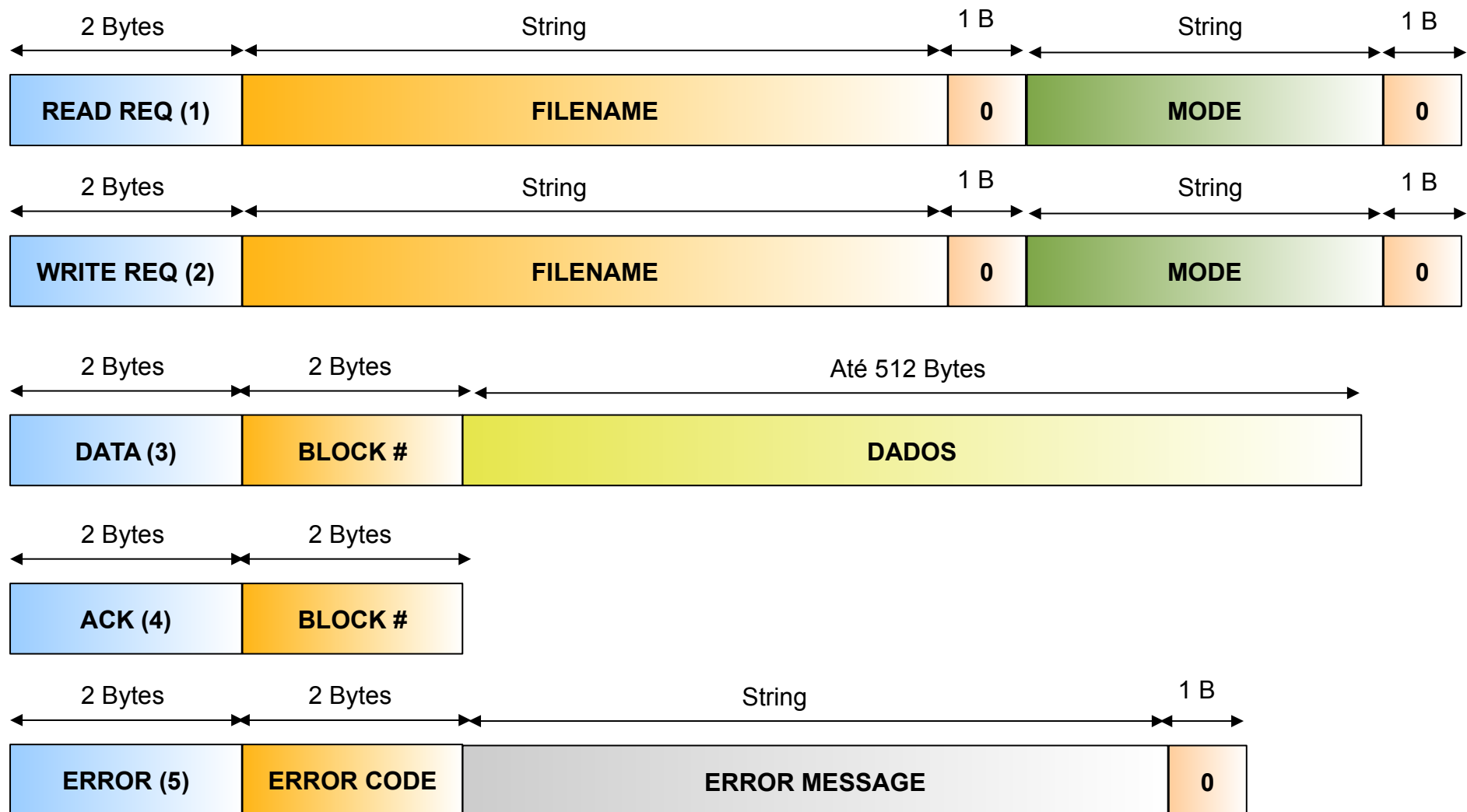
- ◆ Comunicam com o Servidor
- ◆ Podem estar ligados apenas durante a operação
- ◆ Podem ter endereços dinâmicos
- ◆ Não comunicam entre si

# Trivial File Transfer Protocol (TFTP)

- Serviço simples de transferência de ficheiros (IETF RFC 1350)
  - Não permite listar um directório
  - Não tem qualquer mecanismo de autenticação
- Corre sobre UDP
  - O pacote inicial do cliente é enviado para o servidor para o número de porto 69.
  - O servidor responde escolhendo um novo número de porto.
  - Os pacotes seguintes são enviados para o número de porto escolhido.
- Implementa o mecanismo de controlo de fluxo: Stop and Wait
- Baseado em 5 primitivas:
  - Read Request (RRQ)
  - Write Request (WRQ)
  - Data
  - Acknowledgement (ACK)
  - Error (ERR)



# Formato das mensagens

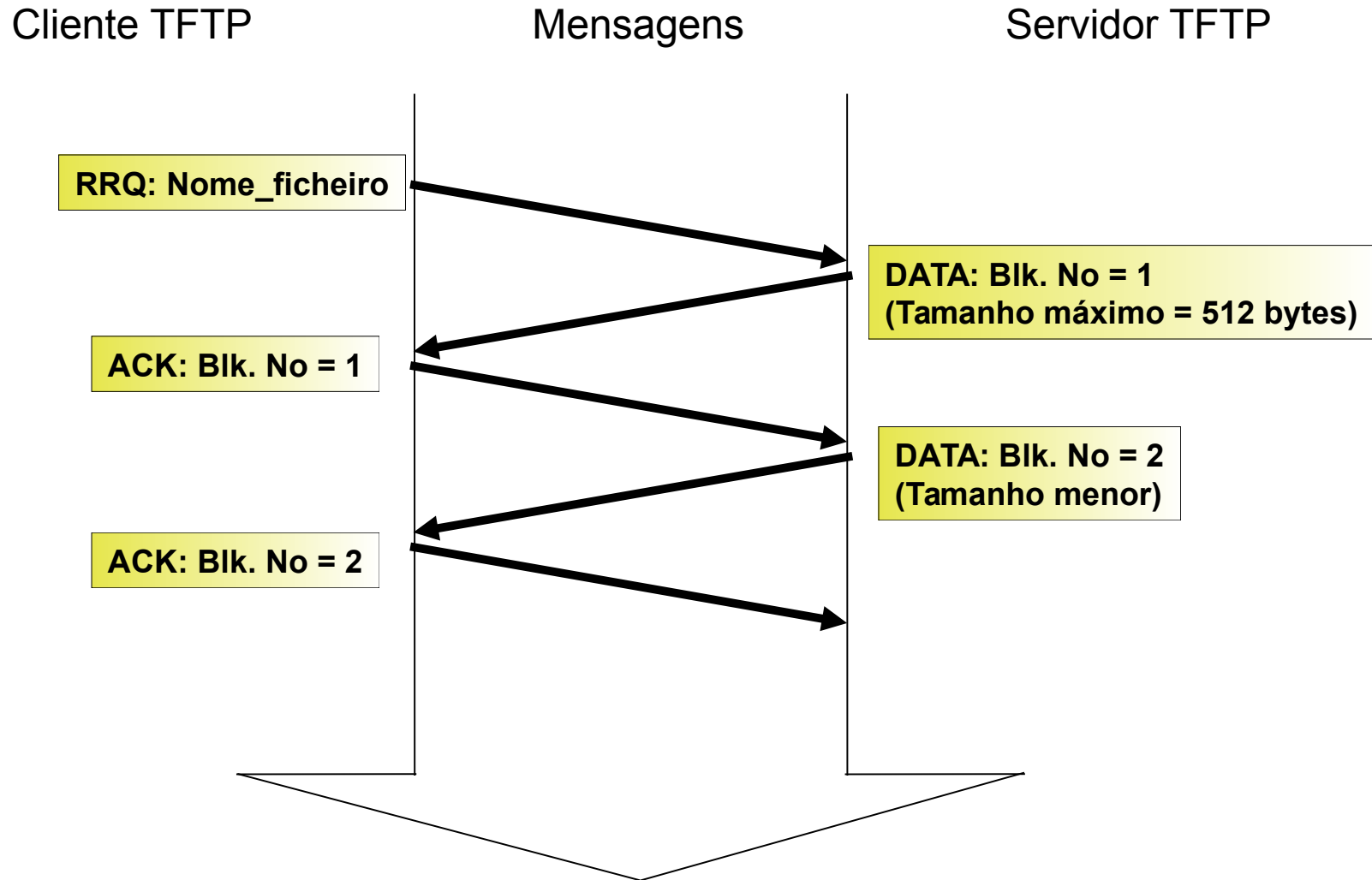


# Formato das mensagens

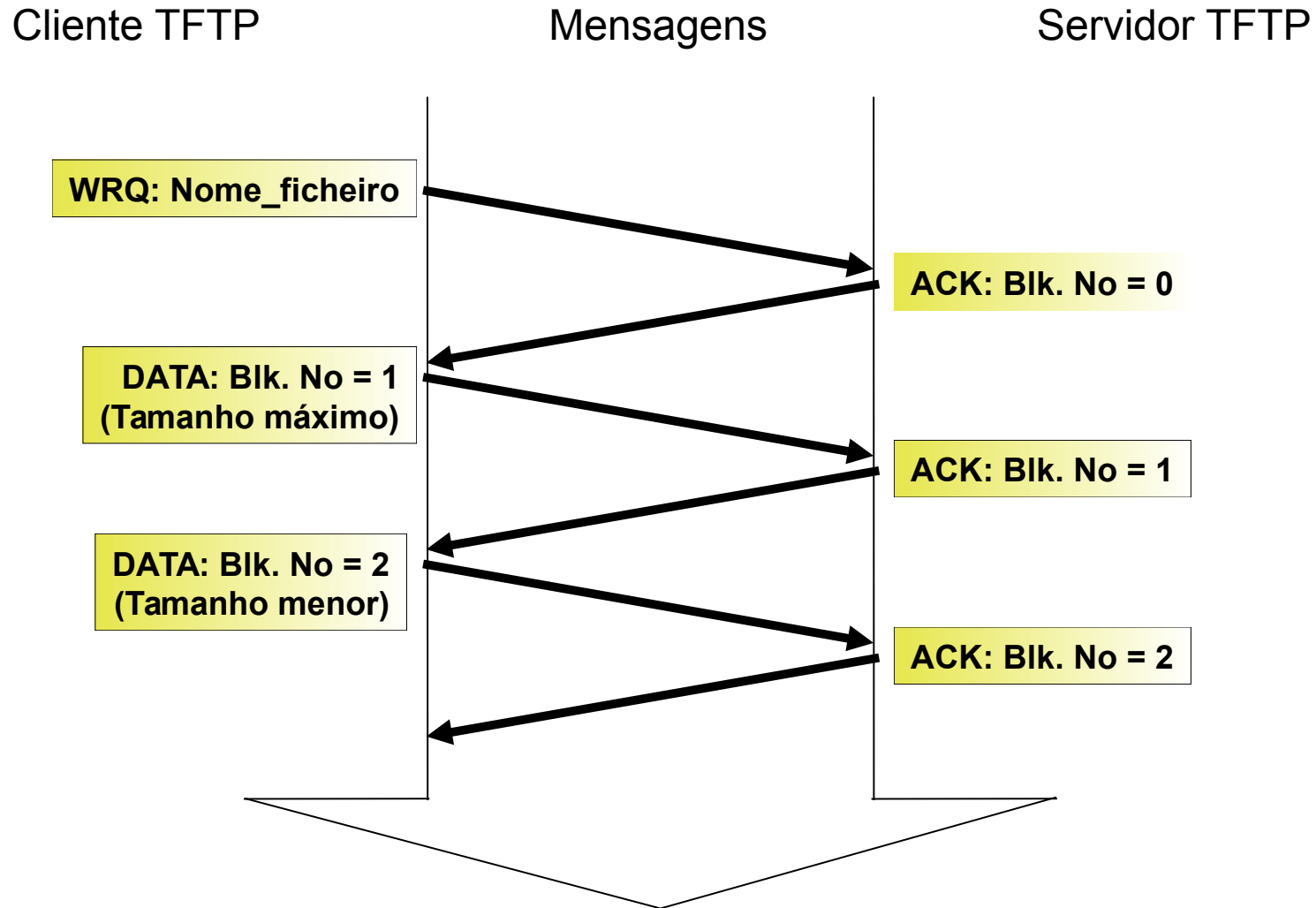
- FILENAME – string de caracteres ASCII que especifica o nome do ficheiro a ler ou a escrever.
- MODE – string de caracteres ASCII que especifica o modo da mensagem.
- BLOCK # – no ACK é igual ao número de bloco da mensagem recebida. O servidor utiliza o ACK para confirmar a recepção dos blocos de dados e o cliente usa os blocos de dados para confirmar a recepção dos ACK, excepto no caso de ACKs duplicados e de um ACK que termina uma ligação.
- ERROR – actua como um NACK; pode causar retransmissão da mensagem ou quebra da ligação.
- ERROR MESSAGE – string ASCII que explica o tipo de erro.
- ERROR CODE: 00 – Not defined; 01 – File not found; 02 – Access violation; 03 – Disk full; 04 – Invalid operation code; 05 – Unknown port number; 06 – File already exists; 07 – No such user



# Sessão *Read Request* em TFTP



# Sessão *Write Request* em TFTP





# *File Transfer Protocol (FTP)*

- Serviço de transferência de ficheiros
- Corre sobre TCP e o servidor usa
- dois números de porto
  - Ligação de controle: porto 21
  - Ligação de dados: porto 20
- Possui mecanismos de autenticação
  - Username + Password ou Username Anonymous
  - Credenciais são transmitidas em texto aberto
- Numa sessão FTP:
  - O cliente estabelece uma ligação de controlo com o servidor por onde são trocados os comandos FTP
  - A ligação de controle mantém-se activa até terminar a sessão FTP
  - Sempre que é necessário uma transferência de dados, o servidor estabelece uma ligação de dados do seu número de porto 20 para um número de porto previamente anunciado pelo cliente pelo comando PORT
  - No fim da transferência de dados, o servidor termina a ligação de dados



# Comandos e respostas FTP

## Sample commands:

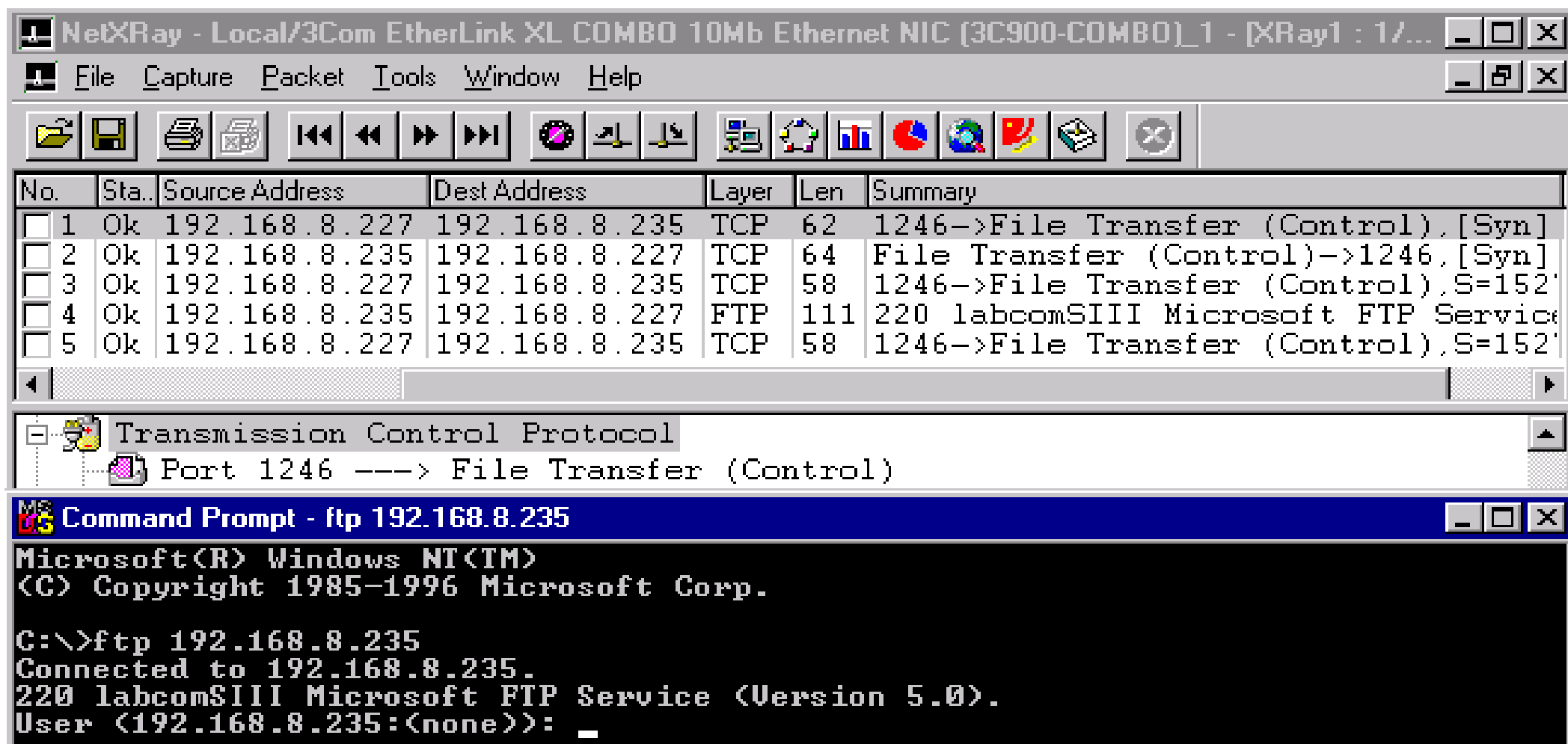
- sent as ASCII text over control channel
- **USER *username***
- **PASS *password***
- **LIST** return list of file in current directory
- **RETR *filename*** retrieves (gets) file
- **STOR *filename*** stores (puts) file onto remote host

## Sample return codes

- status code and phrase (as in HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**



# Ligação inicial ao servidor FTP



The screenshot displays two windows from a Windows NT environment. The top window is NetXRay, showing a packet capture of an FTP session. The bottom window is a Command Prompt where the user has successfully connected to the FTP server at 192.168.8.235.

**NetXRay - Local/3Com EtherLink XL COMBO 10Mb Ethernet NIC (3C900-COMBO)\_1 - [XRay1 : 1/...**

File Capture Packet Tools Window Help

No.	Sta.	Source Address	Dest Address	Layer	Len	Summary
1	Ok	192.168.8.227	192.168.8.235	TCP	62	1246->File Transfer (Control).[Syn]
2	Ok	192.168.8.235	192.168.8.227	TCP	64	File Transfer (Control)->1246.[Syn]
3	Ok	192.168.8.227	192.168.8.235	TCP	58	1246->File Transfer (Control).S=152'
4	Ok	192.168.8.235	192.168.8.227	FTP	111	220 labcomSIII Microsoft FTP Service
5	Ok	192.168.8.227	192.168.8.235	TCP	58	1246->File Transfer (Control).S=152'

Transmission Control Protocol  
Port 1246 ---> File Transfer (Control)

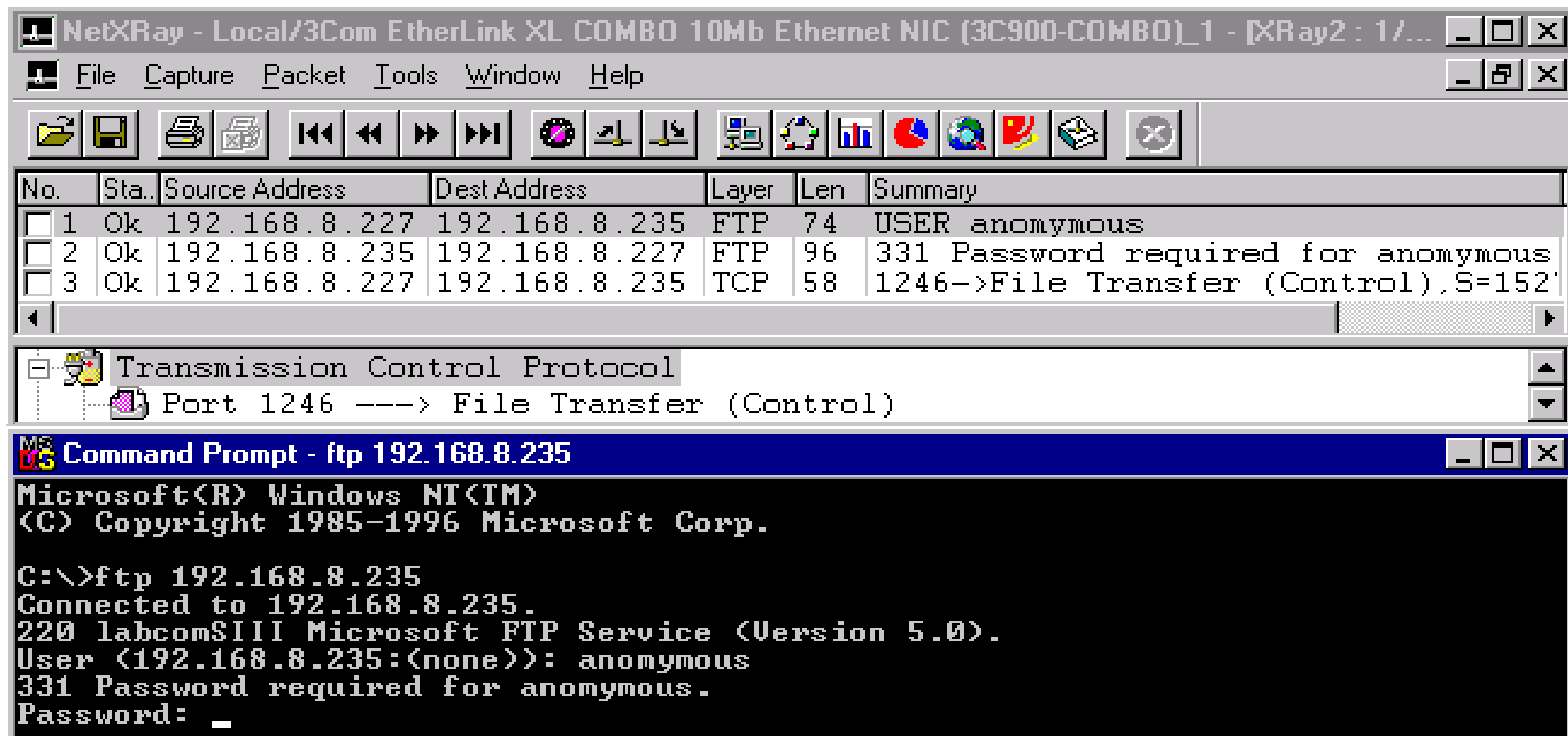
**MS Command Prompt - ftp 192.168.8.235**

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>ftp 192.168.8.235
Connected to 192.168.8.235.
220 labcomSIII Microsoft FTP Service (Version 5.0).
User (192.168.8.235:(none)): _
```



# Introdução inicial do *username*



The screenshot displays two windows from a Windows NT environment. The top window, titled "NetXRay - Local/3Com EtherLink XL COMBO 10Mb Ethernet NIC (3C900-COMBO)\_1 - [XRay2 : 1/...", shows a packet capture interface. The packet list table is as follows:

No.	Sta.	Source Address	Dest Address	Layer	Len	Summary
1	Ok	192.168.8.227	192.168.8.235	FTP	74	USER anonymous
2	Ok	192.168.8.235	192.168.8.227	FTP	96	331 Password required for anonymous
3	Ok	192.168.8.227	192.168.8.235	TCP	58	1246->File Transfer (Control),S=152'

Below the packet list, the "Transmission Control Protocol" pane shows a connection from "Port 1246 ---> File Transfer (Control)".

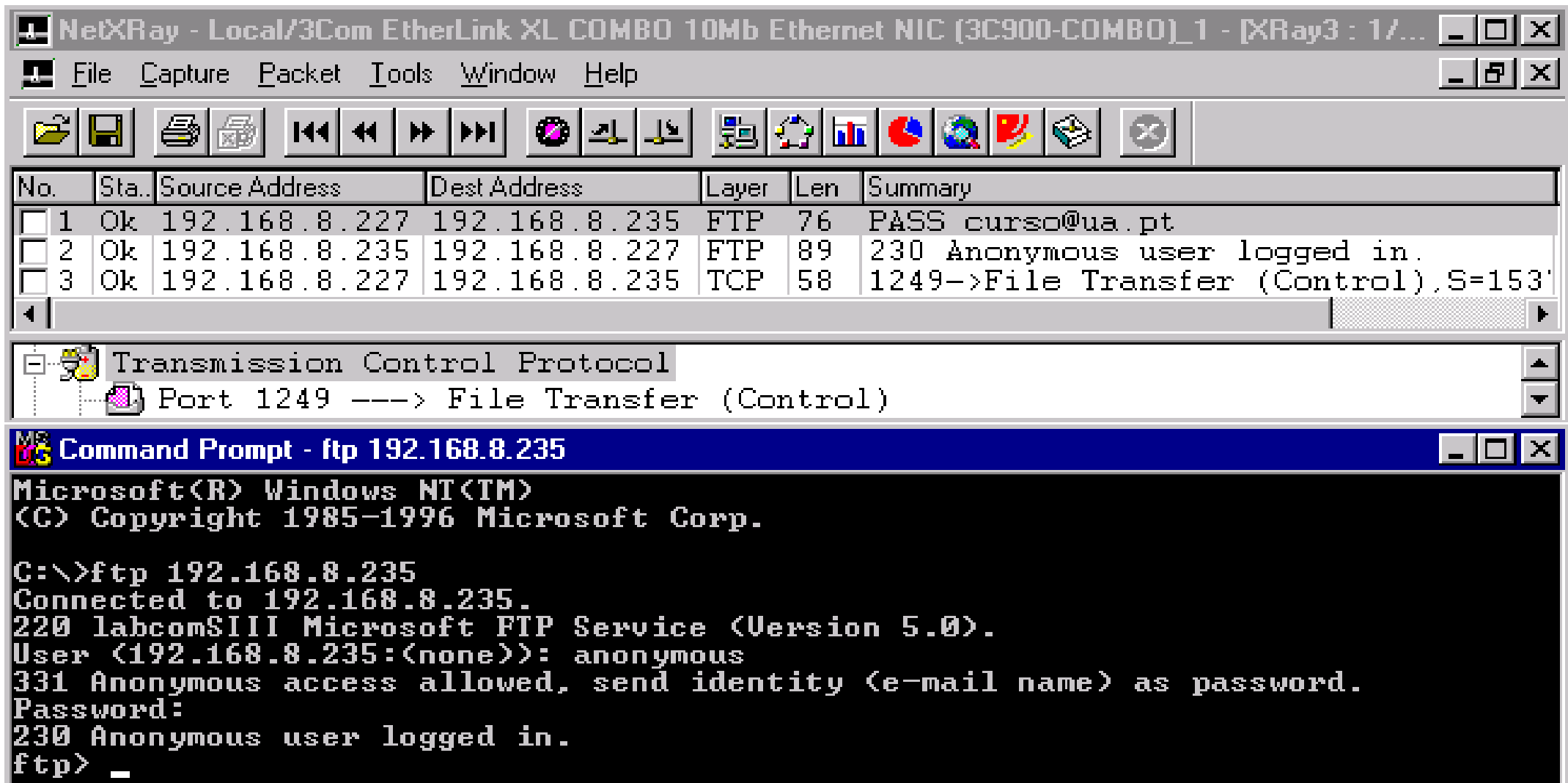
The bottom window, titled "Command Prompt - ftp 192.168.8.235", shows the following text:

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>ftp 192.168.8.235
Connected to 192.168.8.235.
220 labcomSIII Microsoft FTP Service (Version 5.0).
User (192.168.8.235:(none)): anonymous
331 Password required for anonymous.
Password: _
```



# Introdução inicial da *password*



The screenshot displays two windows from a Windows NT environment. The top window, titled "NetXRay - Local/3Com EtherLink XL COMBO 10Mb Ethernet NIC [3C900-COMBO]\_1 - [XRay3 : 1/...", shows a packet capture. The bottom window, titled "Command Prompt - ftp 192.168.8.235", shows the output of an FTP session.

**NetXRay Packet Capture:**

No.	Sta.	Source Address	Dest Address	Layer	Len	Summary
1	Ok	192.168.8.227	192.168.8.235	FTP	76	PASS curso@ua.pt
2	Ok	192.168.8.235	192.168.8.227	FTP	89	230 Anonymous user logged in.
3	Ok	192.168.8.227	192.168.8.235	TCP	58	1249->File Transfer (Control),S=153'

**Command Prompt Output:**

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>ftp 192.168.8.235
Connected to 192.168.8.235.
220 labcomSIII Microsoft FTP Service (Version 5.0).
User (192.168.8.235:(none)): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 Anonymous user logged in.
ftp> _
```



# Introdução do comando DIR

No.	Sta.	Source Address	Dest Address	Layer	Len	Summary
1	Ok	192.168.8.227	192.168.8.235	FTP	84	PORT 192,168,8,227,4,229
2	Ok	192.168.8.235	192.168.8.227	FTP	88	200 PORT command successful.
3	Ok	192.168.8.227	192.168.8.235	FTP	64	LIST
4	Ok	192.168.8.235	192.168.8.227	FTP	111	150 Opening ASCII mode data connect:
5	Ok	192.168.8.235	192.168.8.227	TCP	66	File Transfer (Default Data)->1253,
6	Ok	192.168.8.227	192.168.8.235	TCP	62	1253->File Transfer (Default Data),
7	Ok	192.168.8.235	192.168.8.227	TCP	64	File Transfer (Default Data)->1253,
8	Ok	192.168.8.235	192.168.8.227	FTP	210	Data (total 152 bytes), (More data)
9	Ok	192.168.8.235	192.168.8.227	TCP	64	File Transfer (Default Data)->1253,
10	Ok	192.168.8.227	192.168.8.235	TCP	58	1253->File Transfer (Default Data),
11	Ok	192.168.8.227	192.168.8.235	TCP	58	1253->File Transfer (Default Data),
12	Ok	192.168.8.235	192.168.8.227	TCP	64	File Transfer (Default Data)->1253,
13	Ok	192.168.8.227	192.168.8.235	TCP	58	1252->File Transfer (Control), S=154,
14	Ok	192.168.8.235	192.168.8.227	FTP	82	226 Transfer complete.
15	Ok	192.168.8.227	192.168.8.235	TCP	58	1252->File Transfer (Control), S=154,

File Transfer Protocol  
PORT 192,168,8,227,4,229

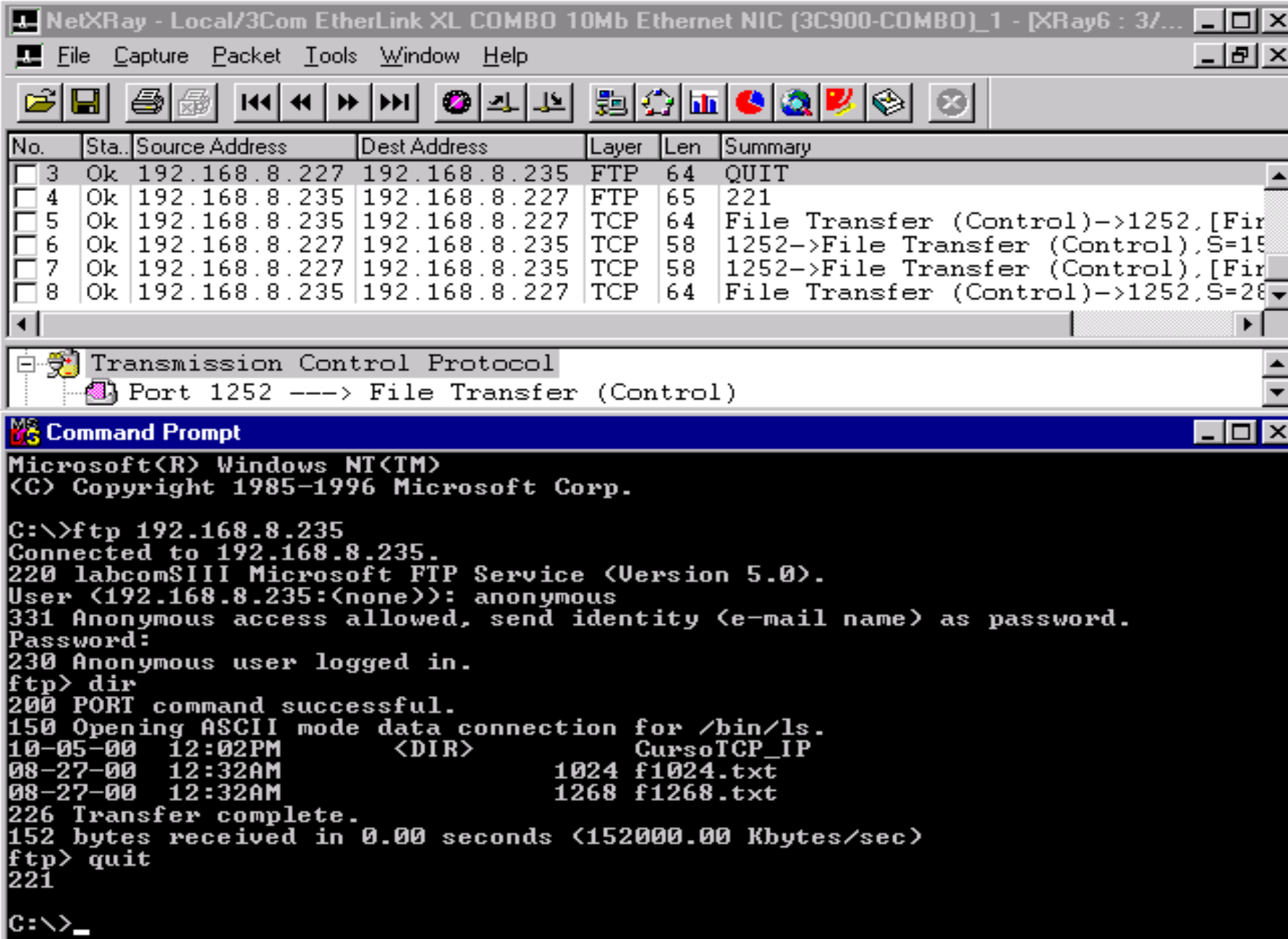
Command Prompt - ftp 192.168.8.235

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>ftp 192.168.8.235
Connected to 192.168.8.235.
220 labcomSIII Microsoft FTP Service (Version 5.0).
User (192.168.8.235:(none)): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 Anonymous user logged in.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
10-05-00 12:02PM <DIR> CursorTCP_IP
08-27-00 12:32AM 1024 f1024.txt
08-27-00 12:32AM 1268 f1268.txt
226 Transfer complete.
152 bytes received in 0.00 seconds (152000.00 Kbytes/sec)
ftp>
```



# Terminação da ligação (comando *quit*)



The screenshot displays two windows. The top window, titled "NetXRay - Local/3Com EtherLink XL COMBO 10Mb Ethernet NIC (3C900-COMBO)\_1 - [XRay6 : 3/...", shows a packet capture table with 8 entries. The bottom window, titled "Command Prompt", shows the execution of an FTP session from a Windows NT command line.

No.	Sta.	Source Address	Dest Address	Layer	Len	Summary
3	Ok	192.168.8.227	192.168.8.235	FTP	64	QUIT
4	Ok	192.168.8.235	192.168.8.227	FTP	65	221
5	Ok	192.168.8.235	192.168.8.227	TCP	64	File Transfer (Control)->1252,[Fir
6	Ok	192.168.8.227	192.168.8.235	TCP	58	1252->File Transfer (Control),S=19
7	Ok	192.168.8.227	192.168.8.235	TCP	58	1252->File Transfer (Control),[Fir
8	Ok	192.168.8.235	192.168.8.227	TCP	64	File Transfer (Control)->1252,S=28

Transmission Control Protocol  
Port 1252 ---> File Transfer (Control)

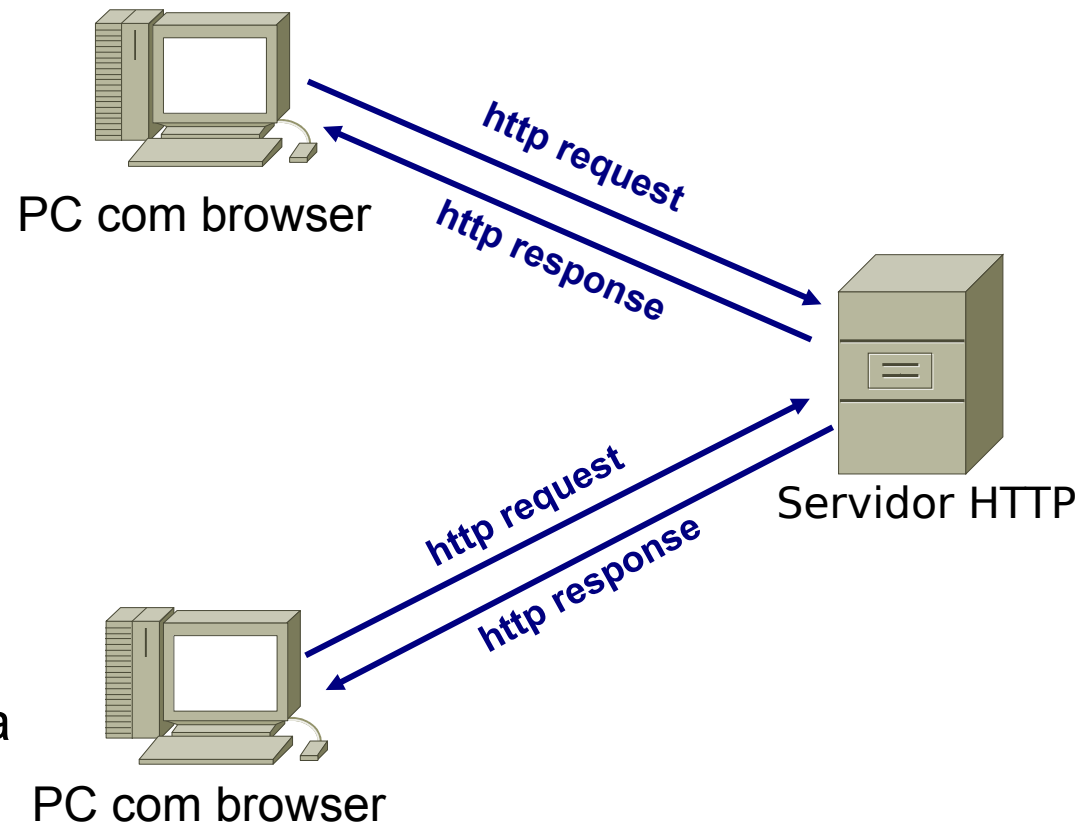
```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>ftp 192.168.8.235
Connected to 192.168.8.235.
220 labcomSIII Microsoft FTP Service (Version 5.0).
User (192.168.8.235:(none)): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 Anonymous user logged in.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
10-05-00 12:02PM <DIR> CursorTCP_IP
08-27-00 12:32AM 1024 f1024.txt
08-27-00 12:32AM 1268 f1268.txt
226 Transfer complete.
152 bytes received in 0.00 seconds (152000.00 Kbytes/sec)
ftp> quit
221

C:\>_
```

# Hyper-Text Transport Protocol (HTTP)

- O protocolo HTTP define as interações entre “Web browsers” e “Web servers” e os formatos das mensagens necessárias
- Cada interação é constituída por 2 ações:
  - ◆ O cliente envia uma mensagem *request* identificando um ficheiro que pretende receber
  - ◆ O servidor envia uma mensagem *response* com uma resposta negativa ou positiva (neste caso, com o conteúdo do ficheiro pedido)
  - ◆ É efetuado um pedido por cada ficheiro mesmo em páginas Web com múltiplos ficheiros





# Modos de utilização das ligações TCP

- Ligações TCP não persistentes:
  - ♦ O cliente estabelece um ligação TCP para enviar o request e o servidor termina a ligação TCP depois de enviar o response
  - ♦ O desempenho penalizado pelo tempo de estabelecimento de cada ligação TCP e pelo seu comportamento slow start
  - ♦ Possibilidade de usar ligações TCP paralelas (número configurável nos browsers) para diminuir o tempo de resposta do servidor
- Ligações TCP persistentes:
  - ♦ O cliente estabelece um ligação TCP para enviar o request com o pedido expreso para o servidor não terminar a ligação TCP depois de enviar o response
  - ♦ O servidor espera um tempo de timeout (tipicamente configurável) para terminar ligações que não são usadas
  - ♦ O cliente pode usar ligações previamente estabelecidas com pipelining (envia múltiplos requests sem esperar que os responses de requests anteriores tenham ainda chegado) ou sem pipelining



# Versões HTTP

- A versão HTTP 1.0 foi definida no RFC 1945
  - ♦ Suporta apenas ligações TCP não persistentes (a capacidade de estabelecimento de ligações TCP paralelas é independente da norma)
  - ♦ Motivação:
    - páginas Web pouco complexas
    - servidores com capacidade de processamento limitada
- A versão HTTP 1.1 foi definida no RFC 2616
  - ♦ Por defeito, funciona com ligações TCP persistentes e com pipelining
  - ♦ Motivação:
    - páginas Web muito complexas
    - servidores com capacidade de processamento muito superior
- As versões são compatíveis:
  - ♦ Um browser HTTP 1.1 dialoga com um servidor HTTP 1.0 e vice-versa

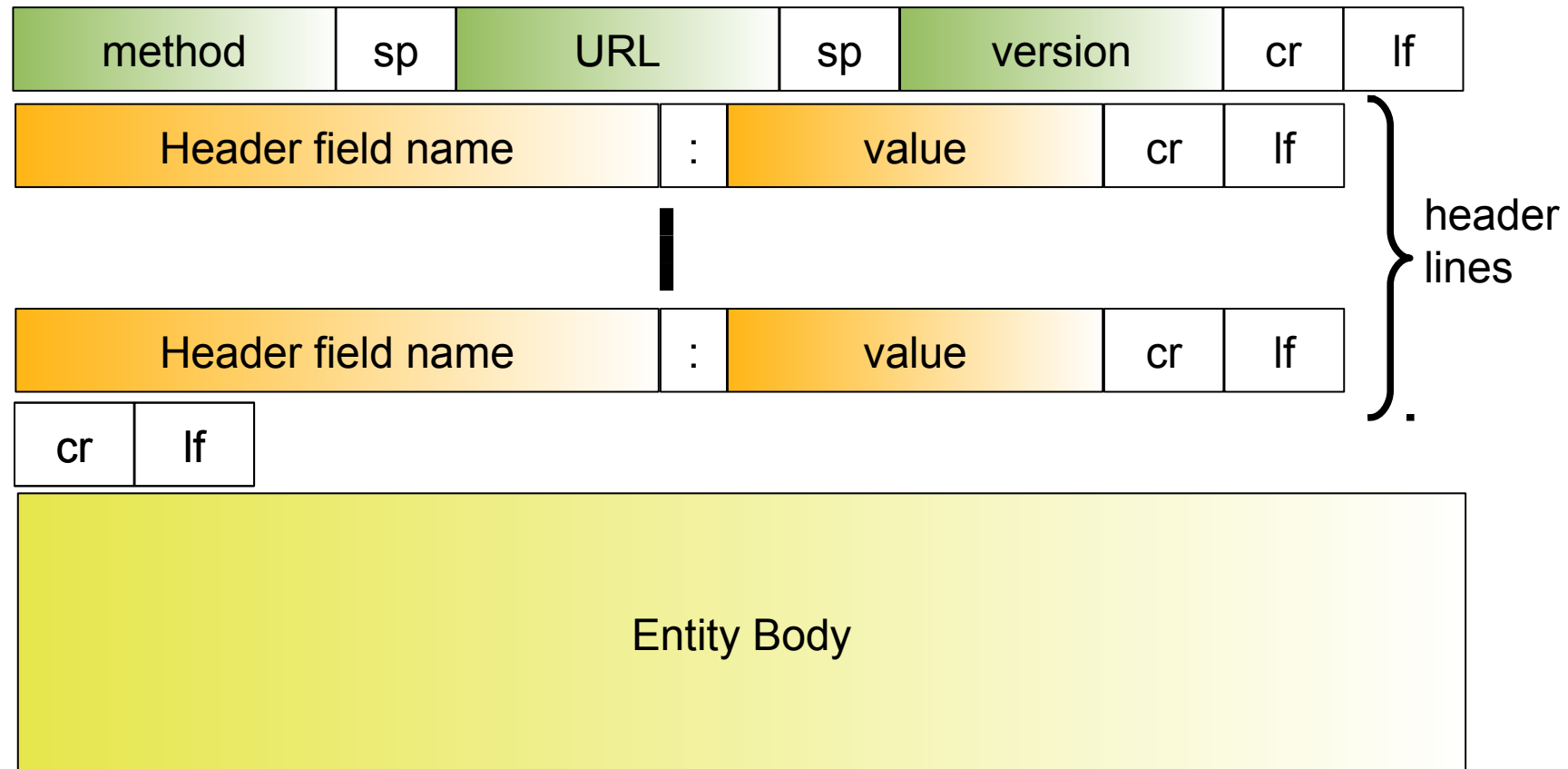


# *Uniform Resource Locator (URL)*

- `http://www.someschool.edu:1024/somedir/page.html`
- `http://`
  - ♦ Protocolo usado para comunicar com o servidor
  - ♦ Outros métodos tipicamente suportados: file, ftp, mailto
- `www.someschool.edu`
  - ♦ Nome DNS do endereço IP da estação do servidor Web
  - ♦ Pode ser o próprio endereço IP (193.136.173.5 ou www.ua.pt)
- `:1024/`
  - ♦ Indica o número de porto do servidor Web (opcional)
  - ♦ Se não existir, o browser usa o número de porto 80 por omissão
- `somedir/`
  - ♦ Caminho para o documento pretendido (opcional)
  - ♦ Se não existir, o documento está no directório principal
- `page.html`
  - ♦ Nome do ficheiro pretendido (a extensão indica um ficheiro HTML)



# Formato da mensagem HTTP *request*



# Mensagens HTTP *request* (exemplo)

```
GET /PageText.aspx?id=259 HTTP/1.1\r\n
Host: www.ua.pt\r\n
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-GB; rv:1.9.0.10)
Gecko/2009042523 Ubuntu/9.04 (jaunty) Firefox/3.0.10\r\n
Accept: text/html,application/xhtml+xml,application/xml;\r\n
Accept-Language: en-gb,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
Keep-Alive: 300\r\n
Connection: keep-alive\r\n
Referer: http://www.ua.pt/\r\n
```

- As mensagens são compostas em formato ASCII
- Começam por uma linha de pedido – method (GET, POST, HEAD...)
- Inclui um número variável de linhas de cabeçalho
  - ♦ Host: identifica o endereço do servidor
  - ♦ Connection: indica se o servidor deve terminar a ligação (close) ou não (keep-alive)
  - ♦ User-agent: especifica o tipo de browser (neste caso é o Mozilla/5.0 típico do browser Firefox)



# Mensagens HTTP *response* (exemplo)

```
HTTP/1.1 200 OK
Connection:close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0
Last-Modified: Mon, 22 Jun 1998 09:23:24 GMT
Content-Length: 6821
Content-Type: text/html
(carriage return, line feed)
(data, data, data, ...)
```

- Começam por uma linha de resposta
- Inclui um número variável de linhas de cabeçalho
- Termina com o conteúdo do ficheiro pedido



# Linhas de resposta HTTP

- 200 OK
  - ♦ Pedido aceite e o conteúdo do ficheiro é incluído na resposta
- 301 Moved Permanently
  - ♦ O ficheiro pedido foi transferido permanentemente
  - ♦ A resposta inclui uma linha de cabeçalho do tipo Location com a nova localização do ficheiro
- 400 Bad Request
  - ♦ O pedido não foi compreendido pelo servidor
- 404 Not Found
  - ♦ O ficheiro não existe no servidor
- 505 HTTP Version Not Supported
  - ♦ A versão HTTP do pedido não é suportada pelo servidor





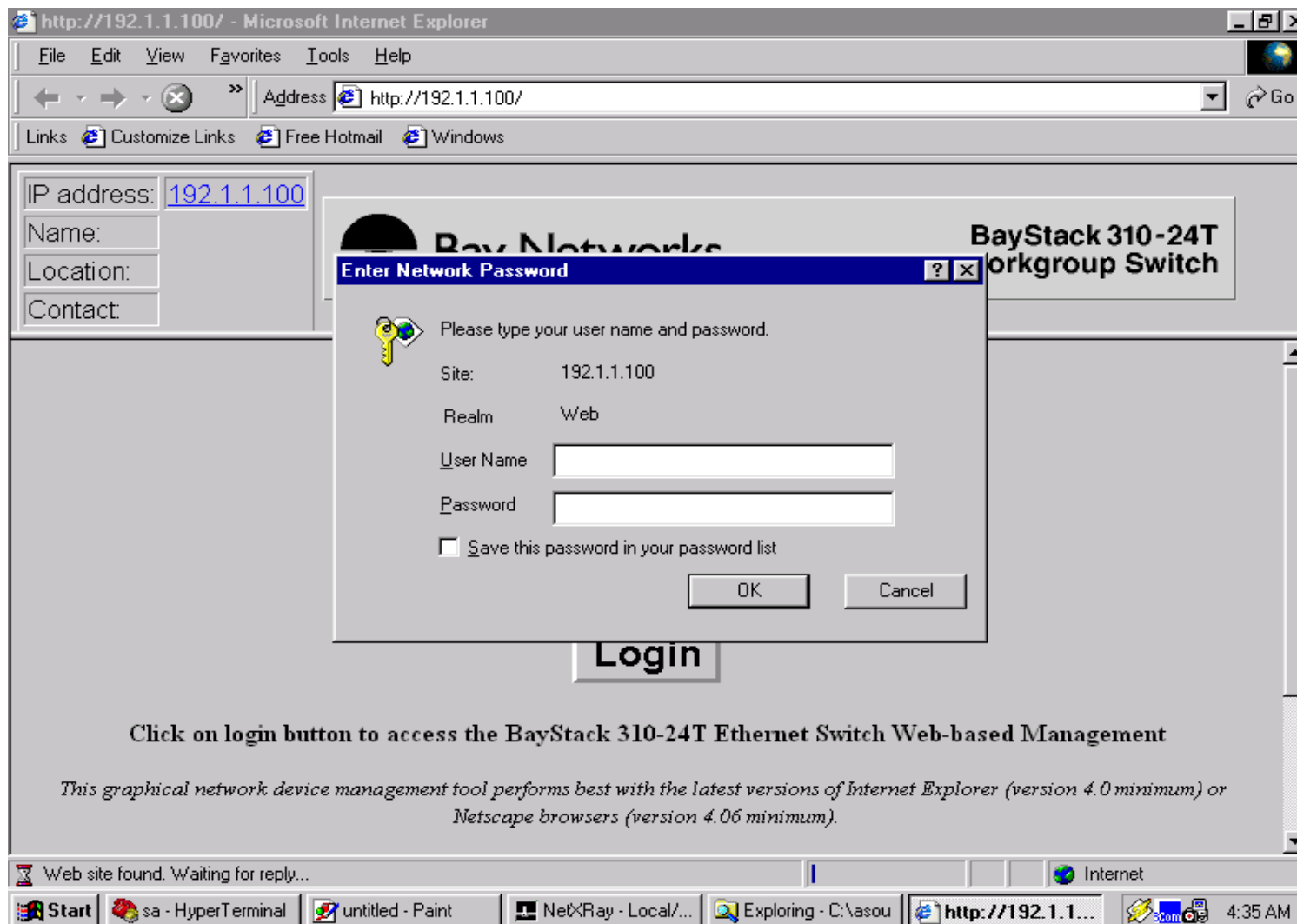
# HTTP - Processo de autenticação

- O HTTP inclui um processo de autenticação que permite limitar o acesso a ficheiros com base num username e password
- Uma mensagem request enviada por um browser para um ficheiro protegido é respondida pelo servidor com uma mensagem response em que a linha de resposta é:
  - 401 Authorization Required
- Esta resposta inclui uma linha de cabeçalho do tipo WWW-Authenticate indicando o método de autenticação a usar
- O novo pedido inclui uma linha de cabeçalho do tipo Authorization com a informação do username e password gerada pelo método pedido pelo servidor
- Tipicamente, o browser guarda a informação de username e password em memória para ser usada em futuras mensagens de request





# Processo de autenticação



# HTTP - Cookies

- A utilização de cookies é definida no RFC 2109
- Os cookies são uma forma do servidor identificar um terminal em diferentes pedidos feitos no tempo
- Permite ao servidor diferenciar a informação a disponibilizar por terminal
- A primeira vez que um terminal enviar um request a um servidor, o servidor inclui na resposta uma linha de cabeçalho, por exemplo:
  - Set-Cookie: uu=ad14cc7cf29d446b0b8e73c5606135efcbe4e58c; expires=Wed, 17-Jun-2009 15:47:29 GMT\r\n
- Se o browser for configurado para aceitar cookies, ele guarda este número juntamente com o identificador do servidor.
- Em futuros pedidos, o browser inclui a linha de cabeçalho:
  - Cookie: uu=ad14cc7cf29d446b0b8e73c5606135efcbe4e58c \r\n
- Deste modo, o servidor identifica o terminal.



# A mensagem GET condicional

- Se o browser suportar Web caching, permite:
  - ♦ minimizar os tempos de resposta
  - ♦ minimizar o tráfego na rede
- Se um ficheiro está em cache no terminal, o browser faz um pedido com uma linha de cabeçalho do tipo If-modified-since

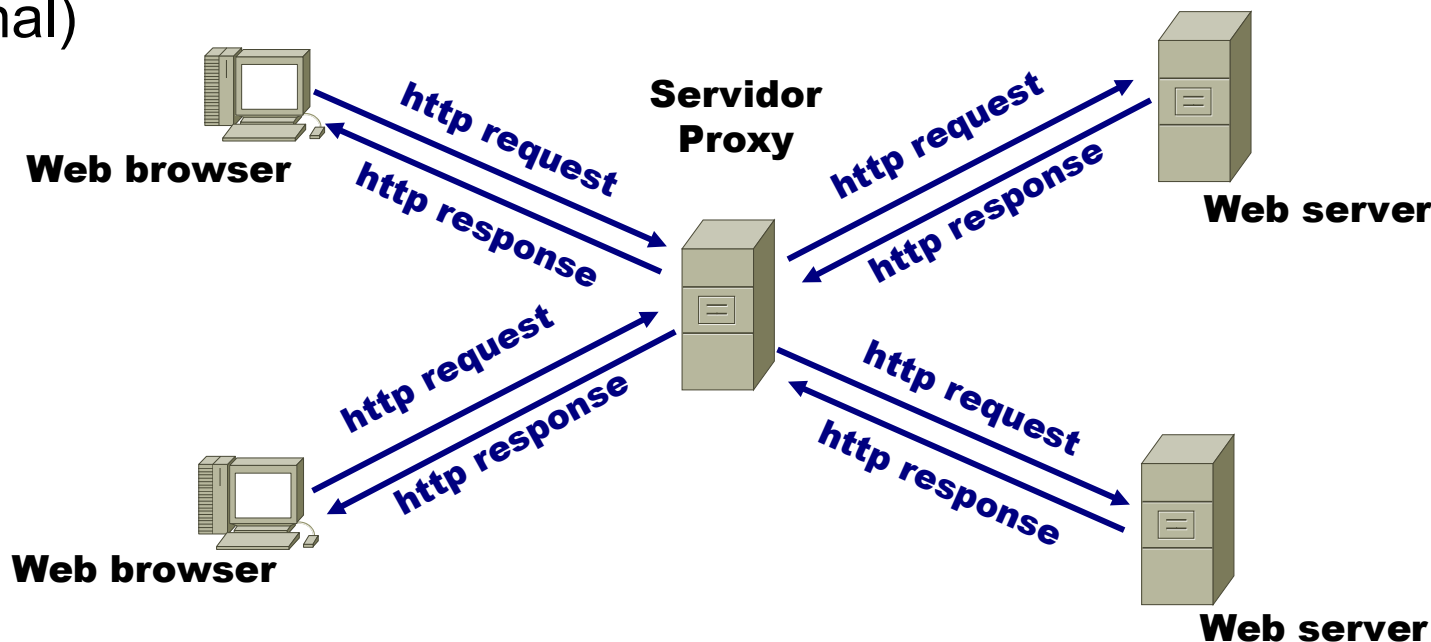
```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
If-modified-since: Mon, 22 Jun 1998 09:23:24 GMT
(carriage return, line feed)
```

```
HTTP/1.1 304 Not Modified
Date: Thu, 19 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0
(carriage return, line feed)
```



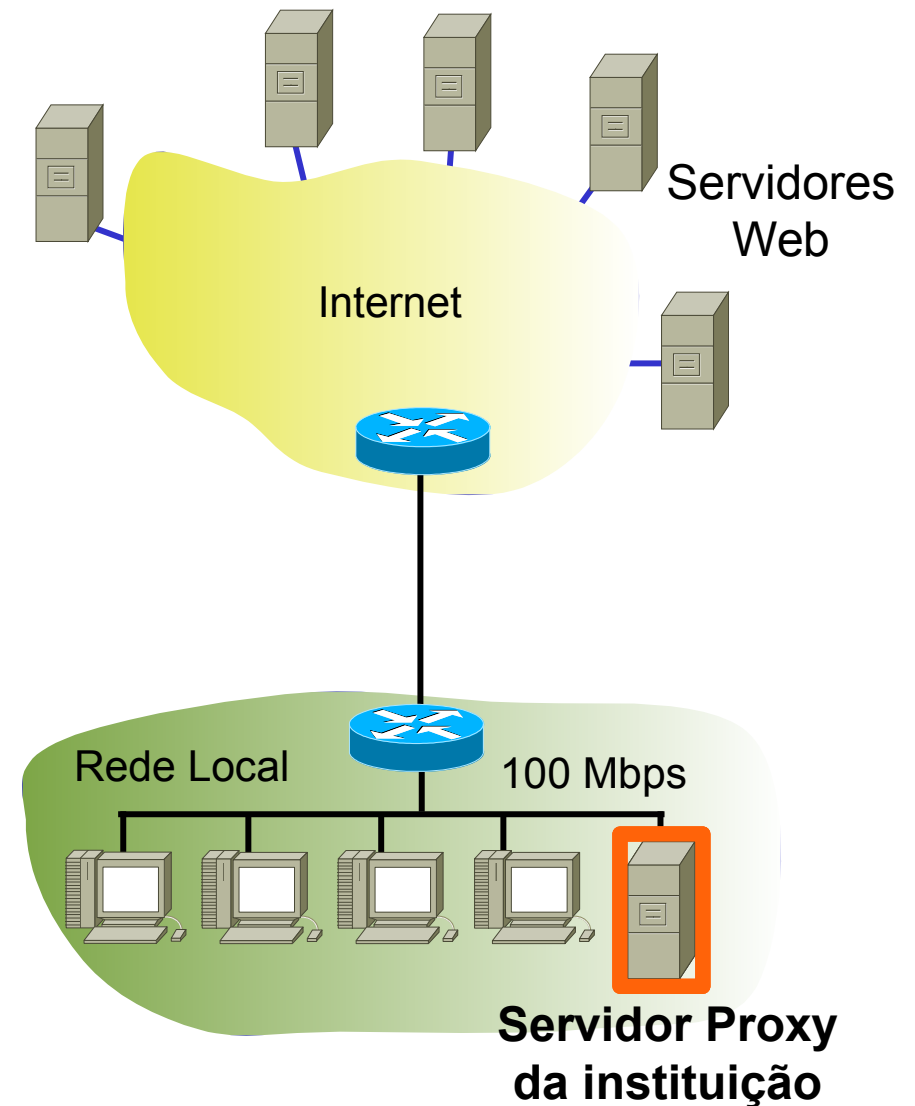
# Servidores Proxy HTTP

- Actua como um elemento intermédio entre o cliente e o servidor:
  - ◆ O cliente interage com o Servidor Proxy como se ele fosse o servidor Web
  - ◆ O Servidor Proxy interage com os servidores Web em nome dos clientes (para o servidor Web, o Servidor Proxy é o cliente)
- O servidor Proxy armazena todos os ficheiros pedidos pelos clientes (até ao limite da sua capacidade de armazenamento)
- Quando o cliente pede um ficheiro já existente no servidor Proxy, não é preciso voltar a pedi-lo ao servidor Web (envia apenas uma mensagem get condicional)



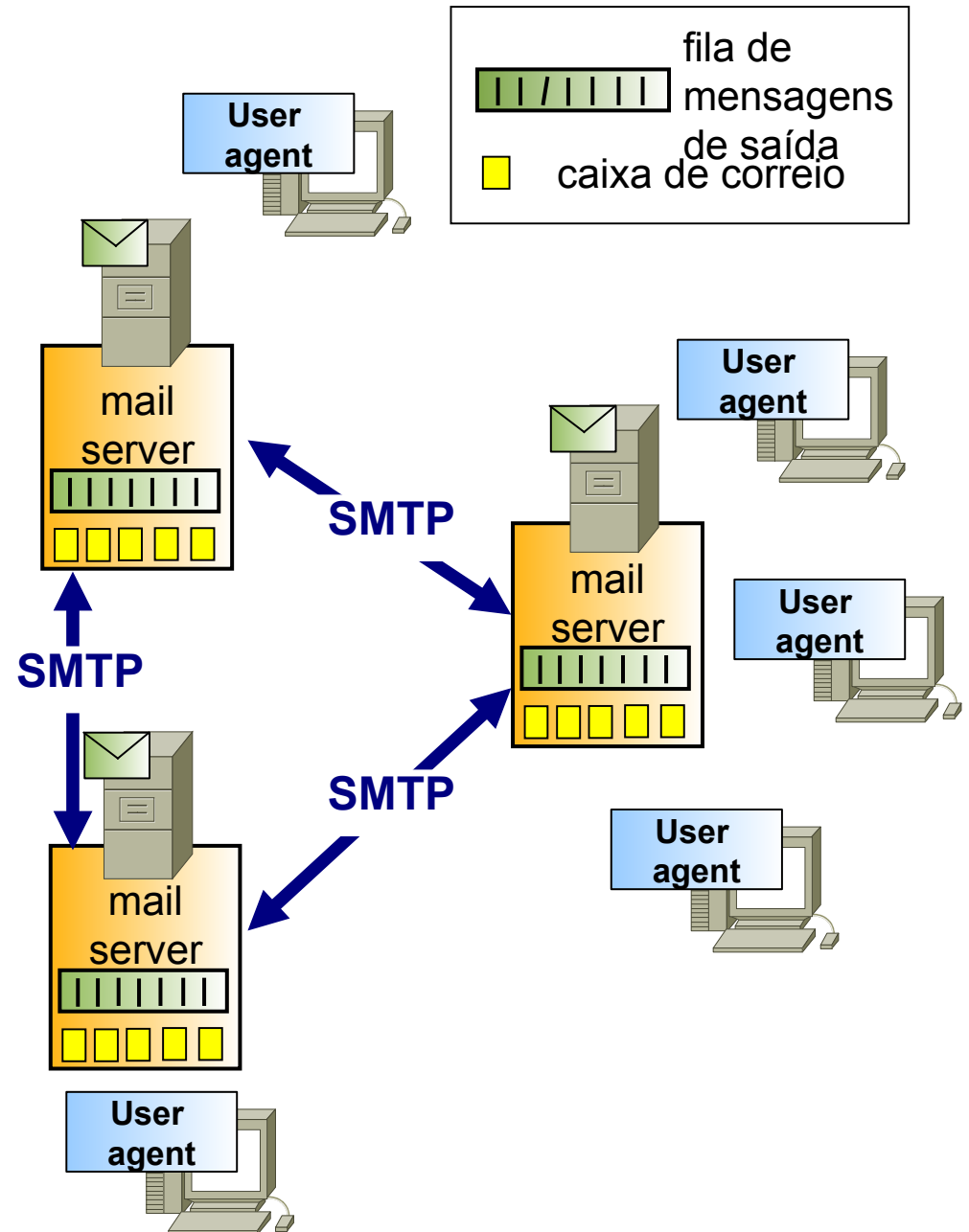
# Porquê servidores de proxy?

- Os servidores proxy nas empresas ou instituições:
- Diminuem os tempos de interação
- Reduzem o tráfego para a rede pública
- Os servidores proxy nas redes dos Internet Service Providers (ISPs):
- Permitem uma infraestrutura de distribuição automática dos conteúdos Web mais solicitados por elementos de rede que estão topologicamente perto dos clientes



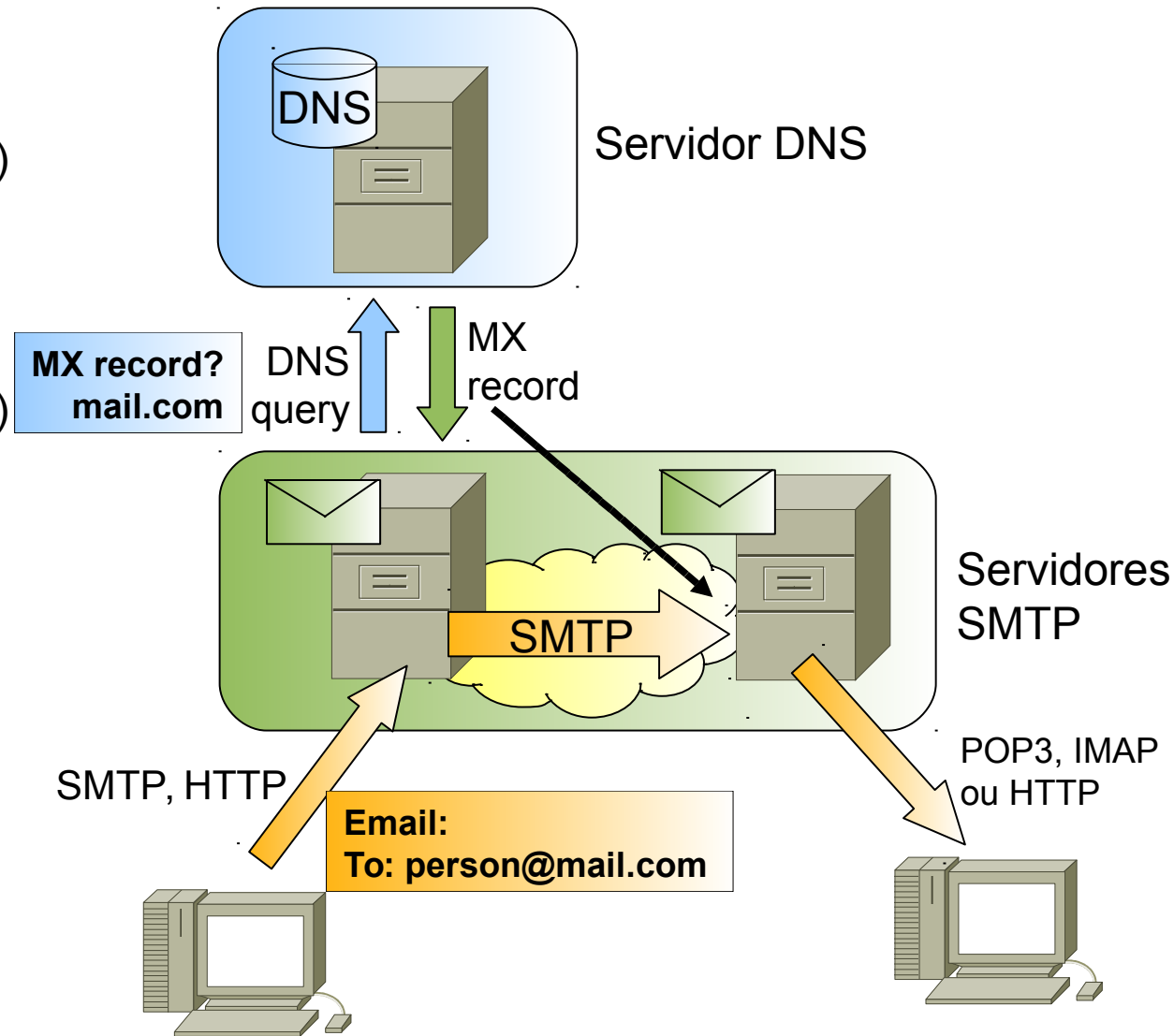
# Correio Electrónico

- User agents: aplicação em que o utilizador escreve, envia, recebe e lê mensagens de correio electrónico
  - ♦ O user agent troca mensagens com o seu mail server
- Mail server: servidor de correio electrónico que envia e recebe as mensagens de/para os seus clientes
- O mail server inclui:
  - ♦ uma caixa de correio (mailbox) por cada cliente onde deposita as mensagens que lhe são destinadas
  - ♦ uma fila de mensagens de saída com as mensagens dos seus clientes que ainda não foram enviadas



# Protocolos de Correio Electrónico

- Envio de mensagens de correio electrónico (entre mail servers)
  - ◆ SMTP (Simple Mail Transfer Protocol)
- Envio de mensagens de correio electrónico (do user agent para o mail server do emissor)
  - ◆ SMTP (Simple Mail Transfer Protocol)
  - ◆ HTTP (Hyper-Text Transport Protocol)
- Acesso à caixa de correio electrónico (envio do mail server para o user agent)
  - ◆ POP3 (Post Office Protocol – versão 3)
  - ◆ IMAP (Internet Mail Access Protocol)
  - ◆ HTTP (Hyper-Text Transport Protocol)





# SMTP (*Simple Mail Transfer Protocol*)

- Definido no RFC 821
- Corre sobre TCP e o número de porto do servidor é o 25
- A comunicação é estabelecida pela entidade que pretende enviar informação (push protocol)
- Comunicações directas: por omissão, o mail server do emissor envia as mensagens directamente para o mail server dos receptores
- O protocolo segue uma filosofia cliente/servidor
  - ◆ O cliente emite comandos
  - ◆ O servidor responde a comandos
  - ◆ À semelhança do HTTP, cada resposta é formada por um código de 3 algarismos seguido de uma frase opcional
- Mensagens de correio electrónico em formato ASCII de 7-bits terminando com "CRLF.CRLF"
  - ◆ CR- carriage return, LF- line feed





# Exemplo de uma interação SMTP

```
Server: 220 mail.ua.pt
Client: HELO mail.av.it.pt
Server: 250 Hello mail.av.it.pt
Client: MAIL FROM: <bob@av.it.pt>
Server: 250 bob@av.it.pt... Sender ok
Client: RCPT TO: <junior@det.ua.pt>
Server: 250 junior@det.ua.pt... Recipient ok
Client: DATA
Server: 354 Enter mail, end with "." on a line by itself
Client: Viva Junior!
Client: A que horas vamos comer?
Client: Bob!
Client: .
Server: 250 Message accepted for delivery
Client: QUIT
Server: 221 mail.ua.pt closing connection
```



# Formato das mensagens SMTP (RFC 822)

```
From: bob@av.it.pt  
To: junior@det.ua.pt  
Subject: Tens fome?  
  
Viva Junior!  
A que horas vamos comer?  
Bob!  
.
```

- Mensagens em formato ASCII
- Começam por um conjunto de linhas de cabeçalho seguido de uma linha vazia seguido do corpo da mensagem:
  - ♦ Algumas linhas (From e To) são obrigatórias na mensagem original
  - ♦ Outras linhas de cabeçalho (Subject, etc...) são opcionais
  - ♦ Algumas linhas de cabeçalho (Received, etc...) são inseridas pelos mail servers



# MIME (Multipurpose Internet Mail Extensions) para dados que não sejam ASCII

- Definido nos RFCs 2045 e 2046
- Permite enviar mensagens de diferentes tipos de informação
- Introduz as linhas de cabeçalho:
  - ♦ Content-Transfer-Encoding – indica o algoritmo de codificação do conteúdo da informação no formato ASCII
    - Exemplos: base64
  - ♦ Content-type: type/subtype; parameters – indica o tipo de informação
    - Exemplos: text/plain; charset="ISO-8859-1"
    - text/html
    - image/gif
    - image/jpeg
    - video/mpeg
    - video/quicktime
    - application/msword



# Mensagens SMTP com MIME

```
From: bob@av.it.pt
To: junior@det.ua.pt
Subject: Imagem fixe!
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

(base64 encoded data .....
.....
..... base 64 encoded data)
```

- No user agent do receptor, o conteúdo da mensagem
  - ♦ é decodificado pelo algoritmo base64 para obter o conteúdo original
  - ♦ de seguida, é entregue a um decodificador JPEG para visualização da imagem enviada



# A extensão MIME do tipo multipart

```
From: bob@av.it.pt
To: junior@det.ua.pt
Subject: Imagem fixe!
MIME-Version: 1.0
Content-Type: multipart/mixed; Boundary=98766789
```

--98766789

Content-Type: text/plain

Viva Junior,  
Junto envio a fotografia combinada.

Texto

--98766789

Content-Transfer-Encoding: base64

Content-Type: image/jpeg

(base64 encoded data .....  
.....  
..... base 64 encoded data)

Imagem

--98766789

Content-Type: text/plain

Bob

Texto

- A linha de cabeçalho multipart/mixed permite compor uma mensagem com múltiplos tipos de informação
- O seu parâmetro Boundary identifica a separação entre diferentes tipos de informação no corpo da mensagem.



# Mensagem SMTP recebida pelo destinatário

```
Received: from mail.iol.pt by mail.ua.pt; 12 Oct 98 15:30:01  
GMT  
Received: from mail.av.it.pt by mail.iol.pt; 12 Oct 98 15:27:39  
GMT  
From: bob@av.it.pt  
To: junior@iol.pt  
Subject: Imagem fixe!  
MIME-Version: 1.0  
Content-Transfer-Encoding: base64  
Content-Type: image/jpeg  
  
(base64 encoded data .....  
..... base 64 encoded data)
```

- Cada mail server insere uma linha de cabeçalho do tipo Received que identifica o servidor que enviou, o servidor que recebeu e o instante de tempo em que a mensagem foi recebida
  - ◆ No exemplo, o utilizador junior configurou o seu servidor mail.iol.pt para reencaminhar as mensagens para o servidor mail.ua.pt



# POP3 (*Post Office Protocol – versão 3*)

- Definido no RFC 1939
- Corre sobre TCP e o número de porto do servidor é o 110
- A comunicação é estabelecida pela entidade (user agent) que pretende receber informação (pull protocol)
- A transferência das mensagens é feita num de dois modos:
  - ♦ envio-e-remoção: as mensagens são removidas da caixa de correio do servidor após serem enviadas
  - ♦ envio-e-armazenamento: as mensagens são mantidas na caixa de correio após serem enviadas
- O protocolo é executado em 3 fases:
  - ♦ autenticação: o user agent envia o nome do utilizador e uma password
  - ♦ transacção: o mail server envia as mensagens que estão na caixa de correio (mailbox) do utilizador; o user agent indica para cada mensagem, se deve ser removida ou não da caixa de correio
  - ♦ actualização: o mail server remove da caixa de correio as mensagens indicadas pelo user agent para remoção





# Comandos POP3

- USER userid
- PASS password
- STAT
  - ♦ A resposta contém o número de mensagens e o tamanho total em bytes das mesmas.
  - ♦ Exemplo de uma resposta: +OK 3 345910
- LIST
  - ♦ A resposta é uma lista, onde cada linha identifica o número e tamanho em bytes de cada mensagem. Termina com uma linha apenas com um ponto.
  - ♦ Exemplo de uma resposta:  
+OK 3 messages  
1 1205  
2 305  
3 344400  
.
- RETR msg#
  - ♦ Recupera a mensagem com o número msg#
  - ♦ Exemplo: RETR 2
- DELE msg#
  - Apaga a mensagem com o número msg#
  - Exemplo: DELE 3
- ♦ RSET
  - ♦ Limpa qualquer marcação de mensagens a apagar.
- QUIT





# Exemplo de uma interação POP3

## Fase de autenticação:

- Comandos do *user agent*:
  - `user: username`
  - `pass: password`
- Respostas do *mail server*:
  - `+OK`
  - `-ERR`

## Fase de transacção, *user agent*:

- `list`: lista o tamanho de cada mensagem
- `retr`: pede cada mensagem pelo seu número
- `dele`: indica que a mensagem deve ser removida da caixa de correio
- `quit`: abandona a sessão POP3

```
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 2 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```



# IMAP (*Internet Mail Access Protocol*)

- Definido no RFC 2060
- Corre sobre TCP e o número de porto do servidor é o 143
- Relativamente ao POP3, o IMAP permite ao utilizador funcionalidades adicionais importantes:
  - ♦ criar e gerir um sistema de directórios de mensagens no servidor; fazer operações de procura no sistema de directórios – útil para utilizadores que usem o serviço de múltiplos terminais;
  - ♦ solicitar o envio de partes das mensagens de correio – útil quando o terminal está ligado à rede através de ligações de baixo débito
- Numa sessão, o servidor está num de 4 estados:
  - ♦ Estado não-autenticado: o estado inicial antes do user agent enviar o nome do utilizador e respectiva password
  - ♦ Estado autenticado: o user agent deve identificar um directório antes de enviar qualquer comando que afecte as mensagens de correio
  - ♦ Estado seleccionado: o user agent pode enviar comandos de gestão das mensagens (vizualizar, remover, transferir, etc...)
  - ♦ Estado de saída (logout): quando a sessão termina



# Comandos (Cliente) IMAP

- CAPABILITY
- LOGIN
- SELECT
  - ◆ Seleciona uma determinada pasta da caixa de email.
  - ◆ Igual ao comando EXAMINE.
- CREATE/DELETE/RENAME
  - ◆ Cria/apaga/renomeia pastas na caixa de email.
- LIST
  - ◆ Lista as pastas na caixa de email.
- SUBSCRIBE/UNSUBSCRIBE
  - ◆ Coloca/remove uma pasta do estado ativo.
- STATUS
  - ◆ Verifica o estado de uma determinada pasta da caixa de email.
- FETCH
  - ◆ Permite recuperar a totalidade da caixa de email ou partes de mensagens ou pastas.
- LOGOUT
- Outros: APPEND, EXPUNGE, SEARCH, STORE, COPY, ...

