# *Protocolos de Aplicação*

***(2º trabalho laboratorial)***

*FEUP/DEEC*
*Redes de Computadores*
*MIEIC – 2009/10*
*José Ruela*

# *Sumário*

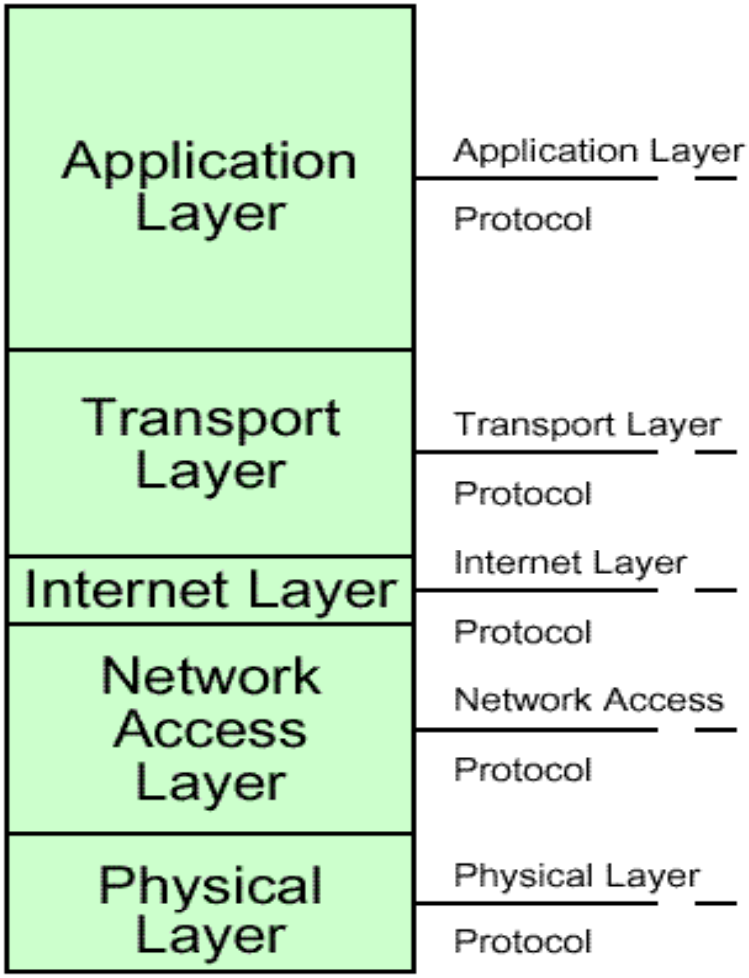
- Pilha protocolar TCP/IP
- Interface de *sockets*
- Protocolos de aplicação
  - » POP3, SMTP, FTP e HTTP
- Exemplos de trabalhos

# *Arquitectura protocolar TCP/IP*

- É a arquitectura dominante actualmente
  - » Os protocolos da família TCP/IP foram especificados e implementados antes da maior parte dos protocolos baseados no modelo OSI
  - » Um grande número de serviços e aplicações disponíveis actualmente usa TCP/IP

- Princípios
  - » As funções de comunicação são estruturadas em módulos
  - » Entidades comunicam com entidades homólogas (*peer entities*) noutros sistemas
  - » Num sistema
    - – Uma entidade usa serviços fornecidos por outras entidades
    - – Uma entidade fornece serviços a outras entidades
    - – Serviços podem ser fornecidos a camadas não adjacentes (ao contrário do modelo OSI)

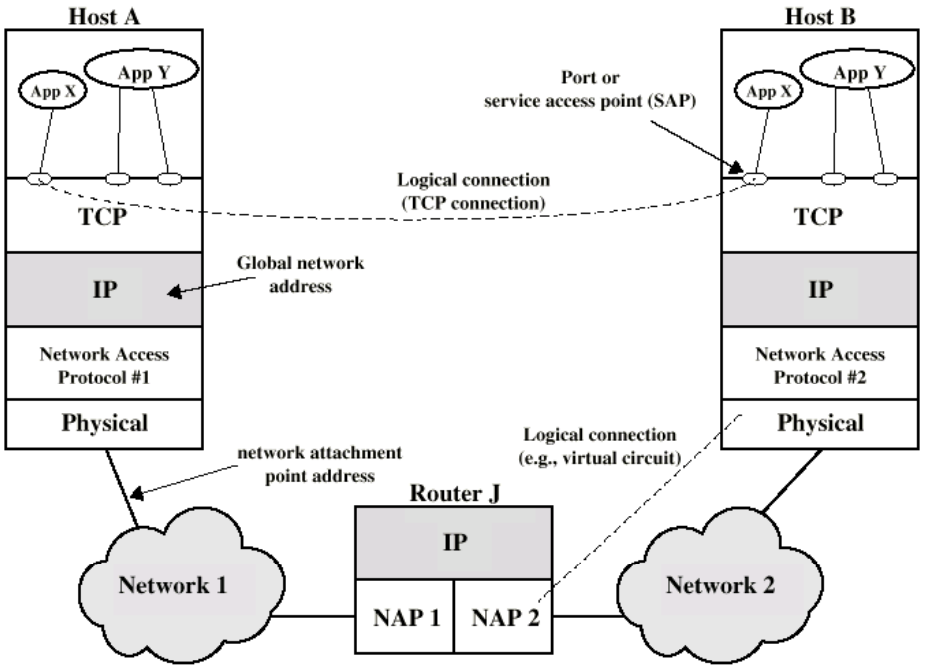
# *Pilha protocolar TCP/IP*

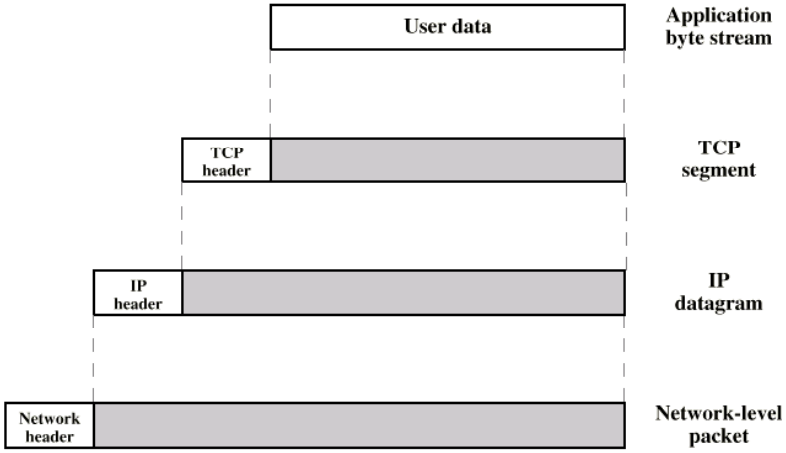



- Aplicação – serviços de utilizador
  - Comunicação entre processos ou aplicações
  - Modelo típico: cliente-servidor
  - Exemplos: HTTP, FTP, Telnet
- Transporte (TCP/UDP)
  - Transmissão de mensagens extremo a extremo
  - Independente do serviço de sub(redes) físicas
  - Transferência fiável (TCP) ou não fiável (UDP)
- Internet (IP)
  - Encaminhamento através de múltiplas (sub)redes interligadas (*internetworking*)
  - Implementado em computadores (*hosts*) e nós intermédios (*routers*)
- Acesso a uma rede (subrede)
  - Acesso a uma (sub)rede e comunicação entre estações (*hosts* / *routers*) ligadas à mesma (sub)rede física
- Interface física
  - Características eléctricas e mecânicas do acesso à (sub)rede (níveis de sinal, débitos de transmissão, conectores, etc.)

# *Algumas características do modelo TCP/IP*

- O IP (*Internet Protocol*) é implementado em todos os computadores (*hosts*) e *routers*
- Cada computador tem um endereço IP único em cada subrede a que pertence
- Cada processo num computador tem um endereço único (porta)



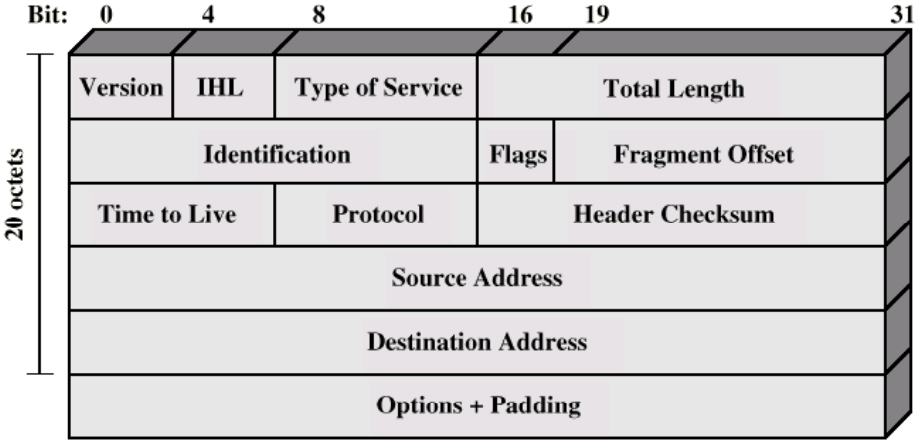



# *IP – Internet Protocol*

- RFC 791
- Entidade da pilha TCP/IP
- Protocolo de interligação de redes mais usado
- IP especificado em duas partes
  - » Serviços oferecidos aos níveis superiores
  - » Protocolo e formato do datagrama IP

# *Protocolo IP*

- **Version** – versão do protocolo (v4)
- **IHL** – comprimento do cabeçalho (em palavras de 32 bits): de 20 octetos (*default*) a 60 (máximo)
- **Type of Service** – tipo de serviço a fornecer pela rede
- **Total Length** – comprimento total do datagrama (máx. 65535 octetos)
- **Identification** – identificador comum a todos os fragmentos de um datagrama original
- **DF** – *Don't Fragment*
- **MF** – *More Fragments*
- **Fragment Offset**
- **Time To Live (TTL)** – limita o número de nós visitados por cada pacote; TTL é decrementado de cada vez que um pacote passa por um *router*; quando atinge o valor *0* o pacote é eliminado

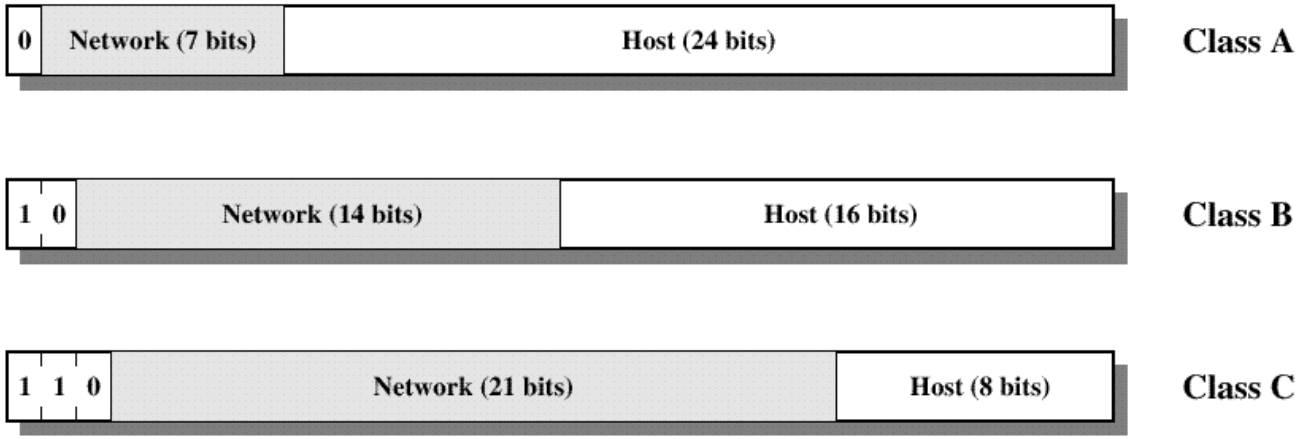



- **Protocol** – protocolo da camada de transporte encapsulado (exemplo: TCP, UDP)
- **Header Checksum** – campo de protecção do cabeçalho
- **Source Address** – endereço do emissor
- **Destination address** – endereço do destinatário
- **Options** – 1 octeto identifica a opção; 1 octeto contém o comprimento (opcional); exemplo: **Record Route**

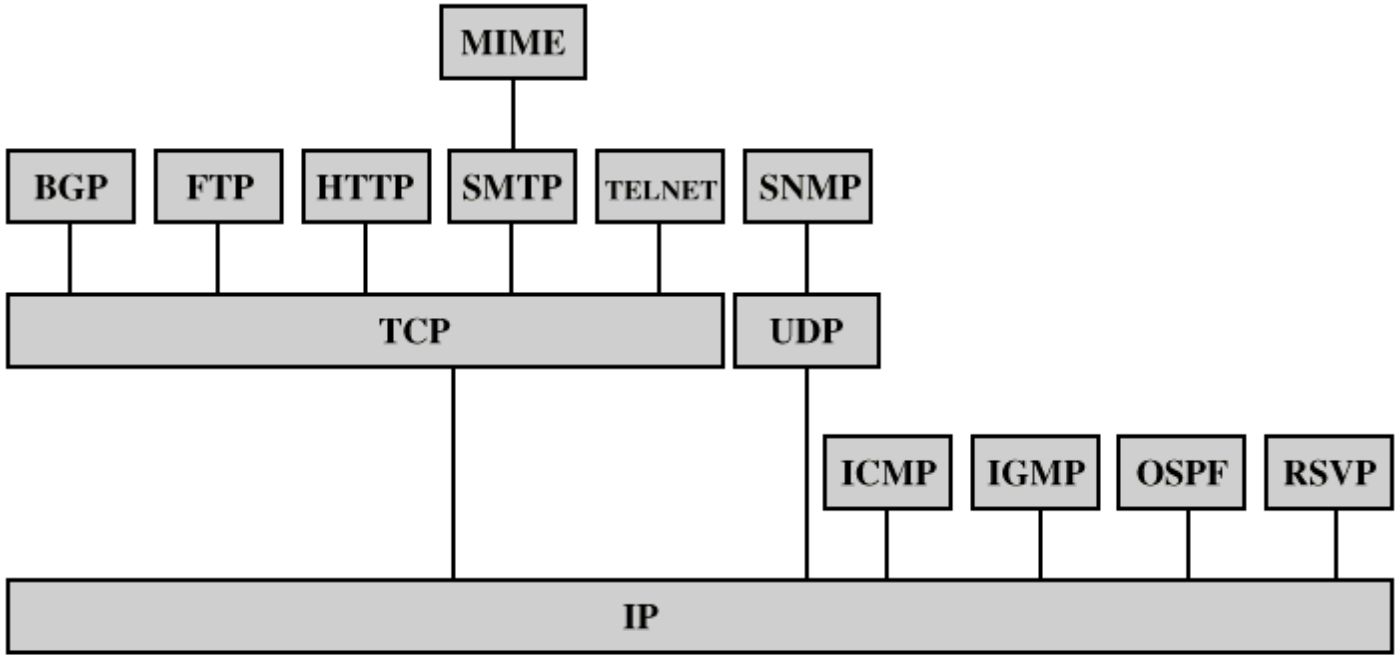
# *IPv4 – endereços*

- Endereços globais de 32 bits, estruturados em duas partes: rede (*netid*) e *host* (*hostid*)
  - » Originalmente os endereços eram baseados em classes (A, B, C, D, E)
    - – Prefixo de rede de comprimento fixo
  - » Endereços sem classes (CIDR)
    - – Prefixo de rede de comprimento variável

| *Classe* | *Valores* |
|---|---|
| A | 0.0.0.0 → 127.255.255.255 |
| B | 128.0.0.0 → 191.255.255.255 |
| C | 192.0.0.0 → 223.255.255.255 |
| D | 224.0.0.0 → 239.255.255.255 |
| E | 240.0.0.0 → 247.255.255.255 |



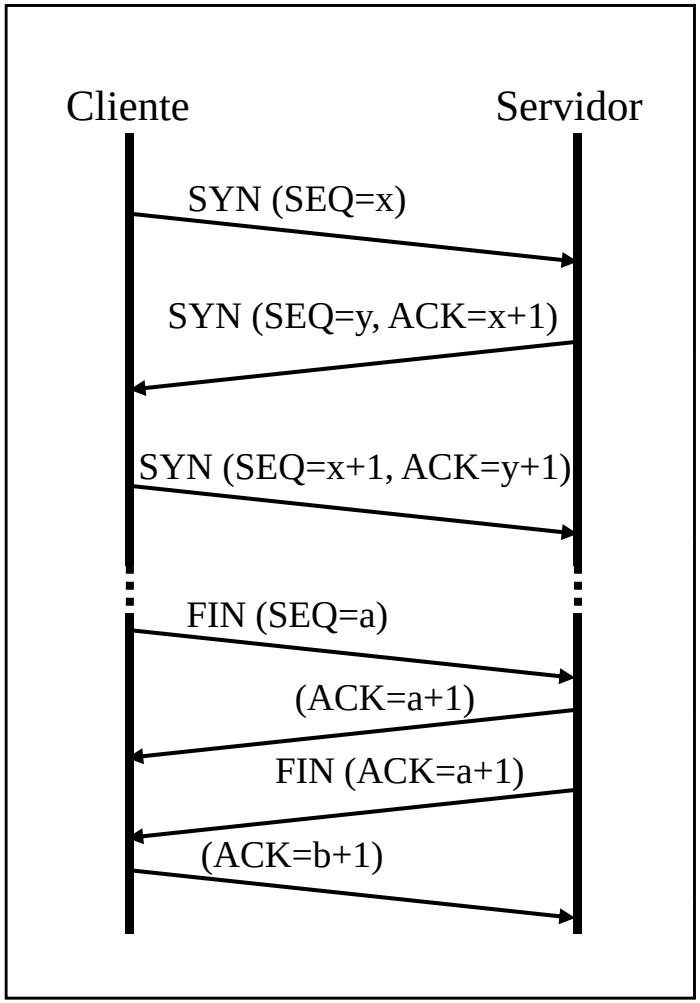
# *Protocolos da família TCP/IP*




BGP = Border Gateway Protocol
FTP = File Transfer Protocol
HTTP = Hypertext Transfer Protocol
ICMP = Internet Control Message Protocol
IGMP = Internet Group Management Protocol
IP = Internet Protocol
MIME = Multi-Purpose Internet Mail Extension
OSPF = Open Shortest Path First
RSVP = Resource ReSerVation Protocol
SMTP = Simple Mail Transfer Protocol
SNMP = Simple Network Management Protocol
TCP = Transmission Control Protocol
UDP = User Datagram Protocol

# *TCP – Transmission Control Protocol*

- RFC 793
- Características
  - » Assegura um fluxo de octetos extremo a extremo, fiável, sobre um suporte não fiável
  - » Protocolo orientado às conexões
  - » Conexões *full-duplex*
  - » Confirmação positiva (ACK)
  - » Recupera de perdas e erros (retransmissões) após *time-out*
  - » Entrega ordenada dos dados à aplicação
  - » Controlo de fluxo e de congestionamento
  - » Multiplexagem de várias conexões TCP sobre o mesmo endereço IP
- Estabelecimento de conexão TCP
  - » *3 way handshake*
  - » Modelo cliente-servidor



# *TCP – Transmission Control Protocol*

**Source Port** – porta de origem

**Destination Port** – porta de destinato

**Sequence Number** – identifica, no fluxo do emissor, a sequência de octetos enviada

**Acknowledgement Number** – corresponde ao número do octeto que se espera receber

**HLEN** – comprimento do cabeçalho TCP (em palavras de 32 bits)

**URG** – informa se o campo *Urgent Pointer* deve ser interpretado

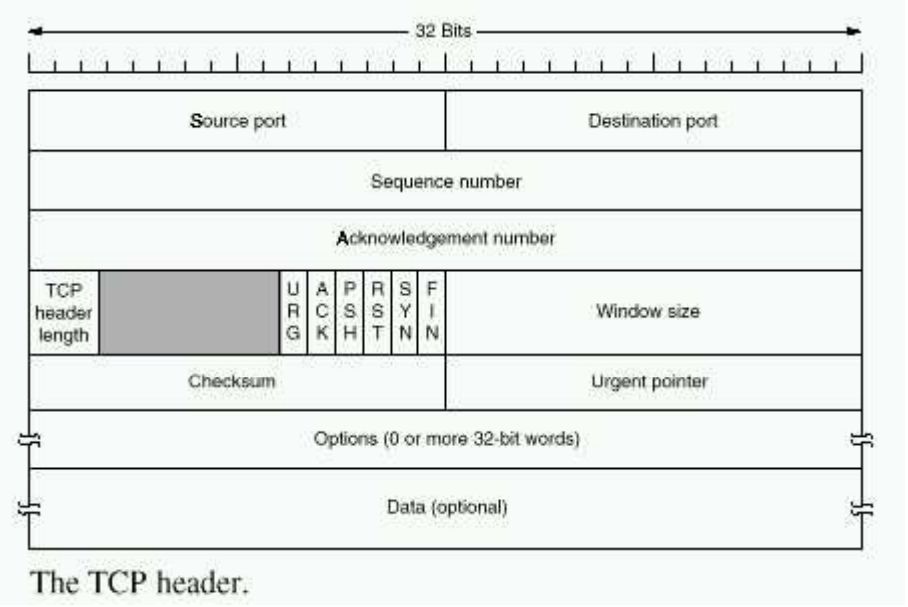
**ACK** – informa se o campo *Ack Nbr* é válido

**PSH** – permite forçar o envio imediato de dados (sem esperar dados adicionais)

**RST** – usado para reinicializar uma conexão

**SYN** – permite estabelecer uma conexão

**FIN** – permite terminar uma conexão



The TCP header.

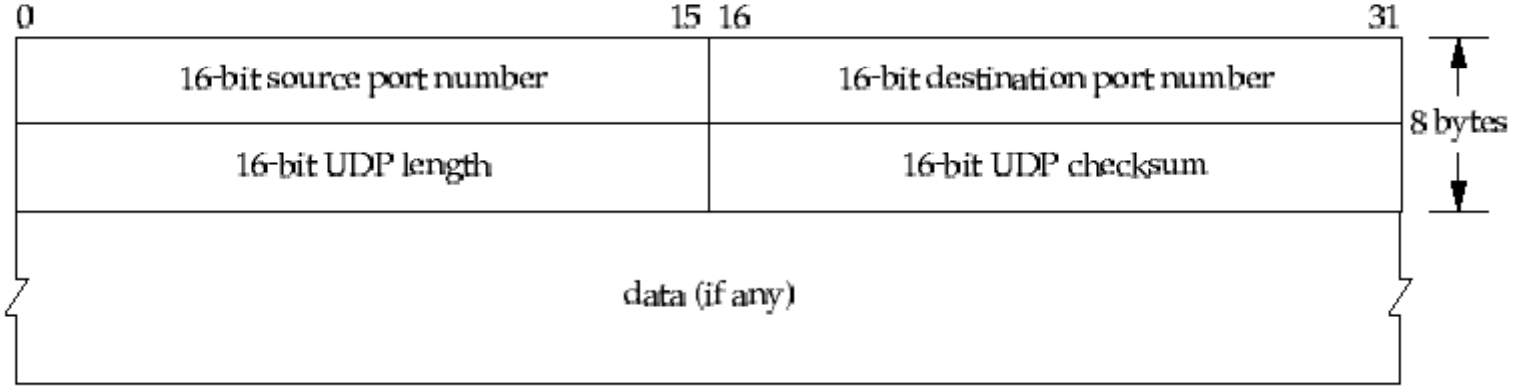
**Window Size –** número de octetos que o par (*peer*) da comunicação pode enviar sem confirmação (controlo de fluxo)

**Checksum –** abrange o cabeçalho, os dados e o pseudo-cabeçalho

# *UDP – User Datagram Protocol*

- RFC 768
- Características
  - » Protocolo de transporte, não orientado às conexões (*connectionless*)
  - » Serviço de entrega de pacotes não fiável
  - » Usa serviços IP
  - » Multiplexagem de vários fluxos UDP sobre o mesmo endereço IP

**UDP Header**




**UDP Length** – comprimento total do pacote

**UDP Checksum** – opcional

# *Berkeley sockets*

- API – *Application Programming Interface*
  - » Sistema operativo: UNIX
  - » Protocolos de comunicação
    - – TCP/IP
    - – UNIX
    - – XNS
  - » Estruturas de dados de endereços
  - » Primitivas
    - – socket()
    - – bind()
    - – connect()
    - – listen()
    - – accept()
    - – recvfrom()
    - – sendto()
    - – close()
  - » Associação entre par de *sockets*

# *Berkeley sockets*

- Estruturas de dados de endereços
  - » BSD

```
<sys/socket.h>
struct sockaddr {
   u_short       sa_family;       /*Address family - ex: AF_INET*/
   char          sa_data[14];     /*Protocol address*/
};
```

  - » Internet

```
<netinet/in.h>
struct in_addr {
   u_long        s_addr;
};
struct sockaddr_in {
   short    sin_family;        /*AF_INET*/
   u_short  sin_port;          /*Port number*/
   struct   in_addr sin_addr;  /*32 bit netid/hosdtid*/
   char     sin_zero[8];       /*unused*/
};
```

# *Berkeley sockets*

➔ `int socket(int family, int type, int protocol)`

*family*: AF_INET, AF_UNIX
*type*: SOCK_STREAM, SOCK_DGRAM, SOCK_RAW
*protocol*: protocolo a usar (com o valor 0, é determinado pelo sistema)

» Retorno
  – descritor de *socket*
  – -1, em caso de erro

➔ `int bind(int sockfd, struct sockaddr* myaddr, int addrlen)`

*sockfd*: descritor do *socket*
*myaddr*: endereço local (IP + porta)
*addrlen*: comprimento da estrutura *myaddr*

» Retorno
  – 0 em caso de sucesso
  – -1 em caso de erro

» Esta primitiva associa o *socket* ao endereço local *myaddr*

# *Berkeley sockets*

➔ `int connect(int sockfd, struct sockaddr* serveraddr, int addrlen)`

*serveraddr*: endereço do servidor remoto (IP + porta)

» Retorno
  – 0 em caso de sucesso
  – -1 em caso de erro

» TCP: estabelecimento de ligação com servidor remoto

» UDP: armazenamento do endereço *serveraddr*

➔ `int listen(int sockfd, int backlog)`

*backlog*: número de pedidos de ligação em fila de espera

» Retorno
  – 0 em caso de sucesso
  – -1 em caso de erro

» Primitiva especifica o número máximo de ligações em fila de espera

# *Berkeley sockets*

```
➔ int accept(int sockfd, struct sockaddr* peeraddr, int* addrlen)
```

*peeraddr*: estrutura usada para armazenar o endereço do cliente (IP + porta)

*addrlen*: apontador para o comprimento da estrutura *peeraddr*

» Retorno
  – descritor do *socket* aceite, endereço do cliente e respectivo comprimento
  – -1 em caso de erro

» Primitiva atende pedido de ligação e cria outro *socket* com as mesmas propriedades que o *sockfd*

```
➔ int send(int sockfd, const void* buf, int len, unsigned int flags)
➔ int recv(int sockfd, void* buf, int len, unsigned int flags)
```

*buf*: apontador para a posição de memória que contém/vai conter os dados

*flags*: MSG_OOB, MSG_PEEK, MSG_DONTROUTE

» Retorno
  – número de octetos escritos/lidos
  – 0 em caso de a ligação ter sido fechada
  – -1 em caso de erro

» Estas primitivas permitem o envio e a recepção de dados da rede

# *Berkeley sockets*

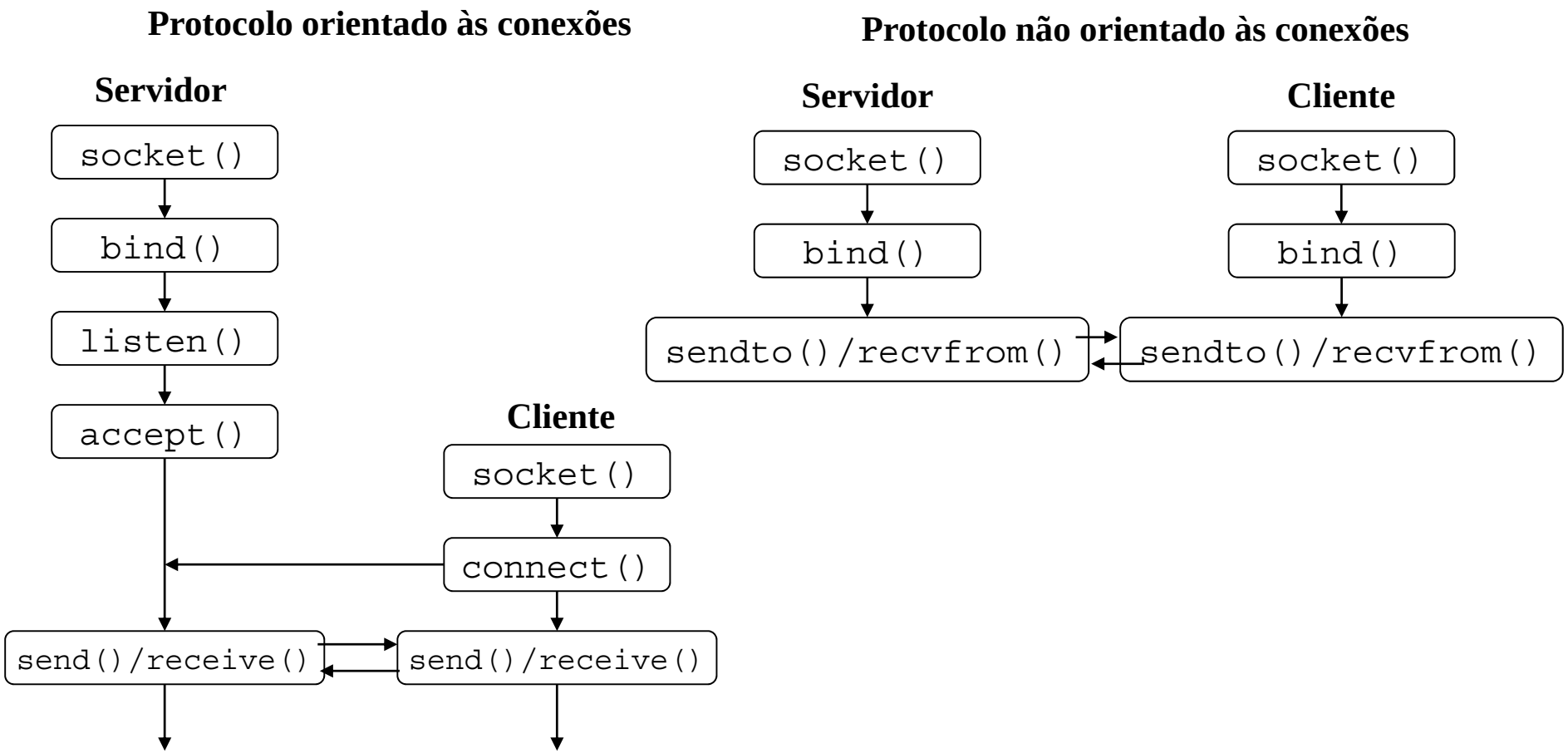
```
➔ int sendto(int sockfd, const void* buf, int len,
         unsigned int flags,
         struct sockaddr* to, int tolen)

➔ int recvfrom(int sockfd, void* buf, int len,
           unsigned int flags,
           struct sockaddr* from, int* fromlen)
```

- » *to*: endereço do destinatário do pacote
- » *from*: endereço do emissor presente no pacote recebido
- » Estas primitivas são semelhantes ao send() / recv() mas permitem adicionalmente o envio de mensagens em cenários *connectionless* (UDP), sem haver portanto estabelecimento de ligação

```
➔ int close(int sockfd)
```

- » Esta primitiva é usada para fechar o *socket*

# *Berkeley sockets*




**Nota**: o cliente de uma ligação TCP pode invocar a primitiva *bind()* antes de estabelecer a conexão

# *Berkeley sockets*

- Ordenação dos octetos
  - » Varia com a arquitectura (e.g., Intel é *little endian*, Motorola é *big endian*)
    - – *little endian* → *little end first* *big endian* → *big end first*
  - » *network byte order* → *big endian*
  - » Primitivas de conversão (long – 32 bits, short – 16 bits):

➔ **`u_long htonl(u_long hostlong)`**
➔ **`u_short htons(u_short hostshort)`**
➔ **`u_long ntohl(u_long netlong)`**
➔ **`u_short ntohs(u_short netshort)`**

- Conversão entre formatos de endereços
  - » *dotted decimal notation* para endereço Internet de 32 bits com ordenação de rede

➔ **`unsigned long inet_addr(char * cp)`**

  - » Endereço Internet de 32 bits com ordenação de rede para *dotted decimal notation*

➔ **`char* inet_ntoa(struct in_addr in)`**

# *Berkeley sockets*

- Opções dos *sockets*
  **`setsockopt()`**
  **`getsockopt()`**
  **`fcntl()`**
  **`ioctl()`**
- Entradas / Saídas assíncronas
  - » utilização de sinais
- Multiplexagem de Entradas/Saídas
  - » Rotina **`select()`**

- *Domain Name Service*
  - » Permite a obtenção do endereço de uma máquina a partir do nome

```
struct hostent*
gethostbyname (const char* name);

struct hostent{
      char*  hname;      /*nome oficial*/
      char** haliases;
      int    h_addrtype; /*AF_INET*/
      int    h_length;
      char** h_addr_list;
};

#define h_addr h_addr_list[0]
```

# *POP3*

POP3 – Post Office Protocol – version 3 (RFC 1939)

» Acesso a caixas de correio remotas para aceder ao correio armazenado num servidor

» Ligações TCP na porta 110

» Sessão

- Estados
  - AUTHORIZATION
  - TRANSACTION
  - UPDATE
- Comandos

Estado AUTHORIZATION

USER name
PASS password
QUIT

Estado TRANSACTION

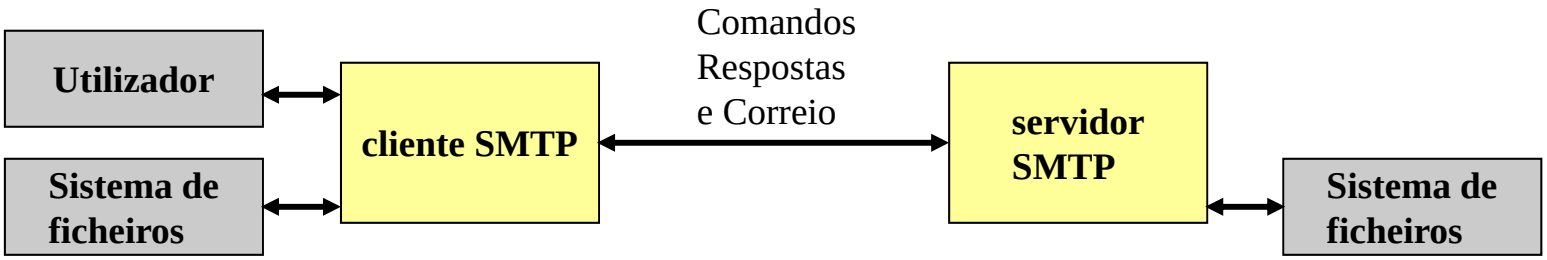
| | |
|---|---|
| STAT | LIST [msg] |
| NOOP | RETR msg |
| RSET | DELE msg |
| QUIT | |
| TOP msg n (extensão) | |
| UIDL [msg] (extensão) | |

```
S: <wait for connection on TCP port 110>
C: <open connection>
S:    +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C:    APOP mrose c4c9334bac560ecc979e58001b3e22fb
S:    +OK mrose's maildrop has 2 messages (320 octets)
C:    STAT
S:    +OK 2 320
C:    LIST
S:    +OK 2 messages (320 octets)
S:    1 120
S:    2 200
S:    .
C:    RETR 1
S:    +OK 120 octets
S:    <the POP3 server sends message 1>
S:    .
C:    DELE 1
S:    +OK message 1 deleted
C:    RETR 2
S:    +OK 200 octets
S:    <the POP3 server sends message 2>
S:    .
C:    DELE 2
S:    +OK message 2 deleted
C:    QUIT
S:    +OK dewey POP3 server signing off (maildrop empty)
C:  <close connection>
S:  <wait for next connection>
```

# *SMTP*

## SMTP – Simple Mail Transfer Protocol (RFC 2821)

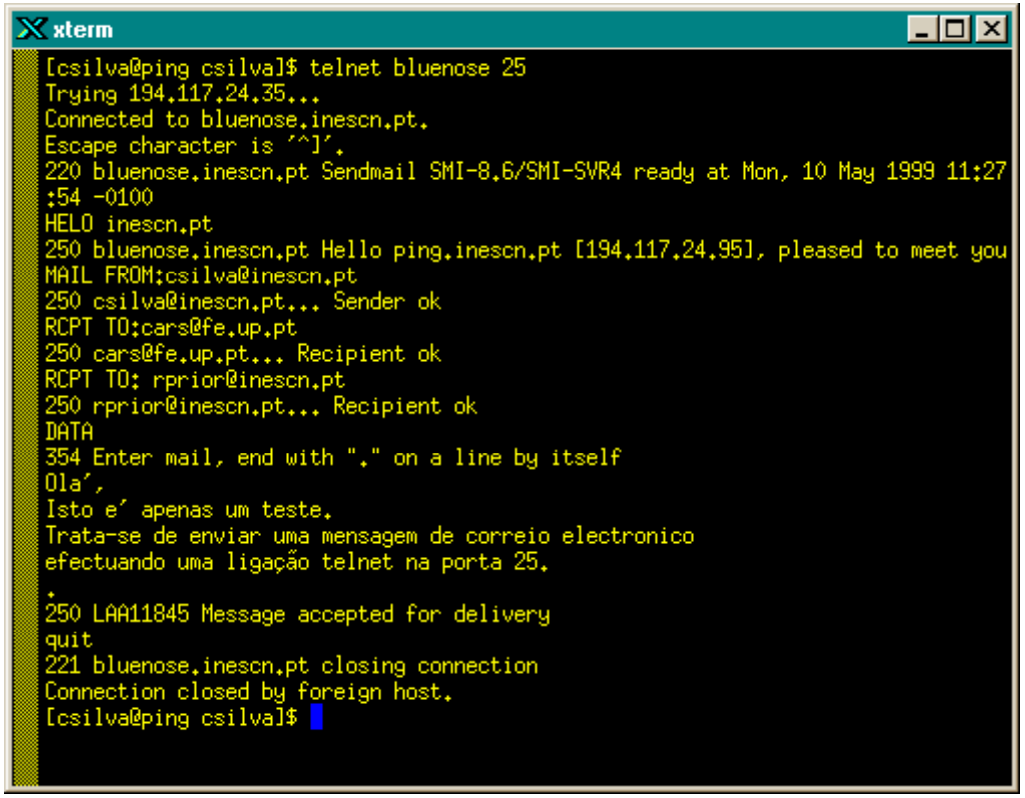
- Envia mensagens de correio de forma fiável
- Ligações TCP na porta 25




- Comandos
  - HELO<SP>domain<CRLF>
  - MAIL<SP>FROM:<reverse-path><CRLF>
  - RCPT<SP>TO:<forward-path><CRLF>
  - DATA<CRLF>
  - QUIT<CRLF>

# *SMTP*

## Exemplo




Outros RFCs relacionados: 2920, 3030, 2487

## Formato das mensagens

- » *Message Formats* (RFC 2822)
  - – Mensagens ASCII
- » MIME – *Multipurpose Internet Mail Extensions* (RFC 2045 a RFC 2049)
  - – Acentuação
  - – Outros alfabetos
  - – Audio e vídeo
  - – Binários

| Header | Meaning |
|---|---|
| To: | Email address(es) of primary recipient(s) |
| Cc: | Email address(es) of secondary recipient(s) |
| Bcc: | Email address(es) for blind carbon copies |
| From: | Person or people who created the message |
| Sender: | Email address of the actual sender |
| Received: | Line added by each transfer agent along the route |
| Return-Path: | Can be used to identify a path back to the sender |

RFC 822 header fields related to message transport.

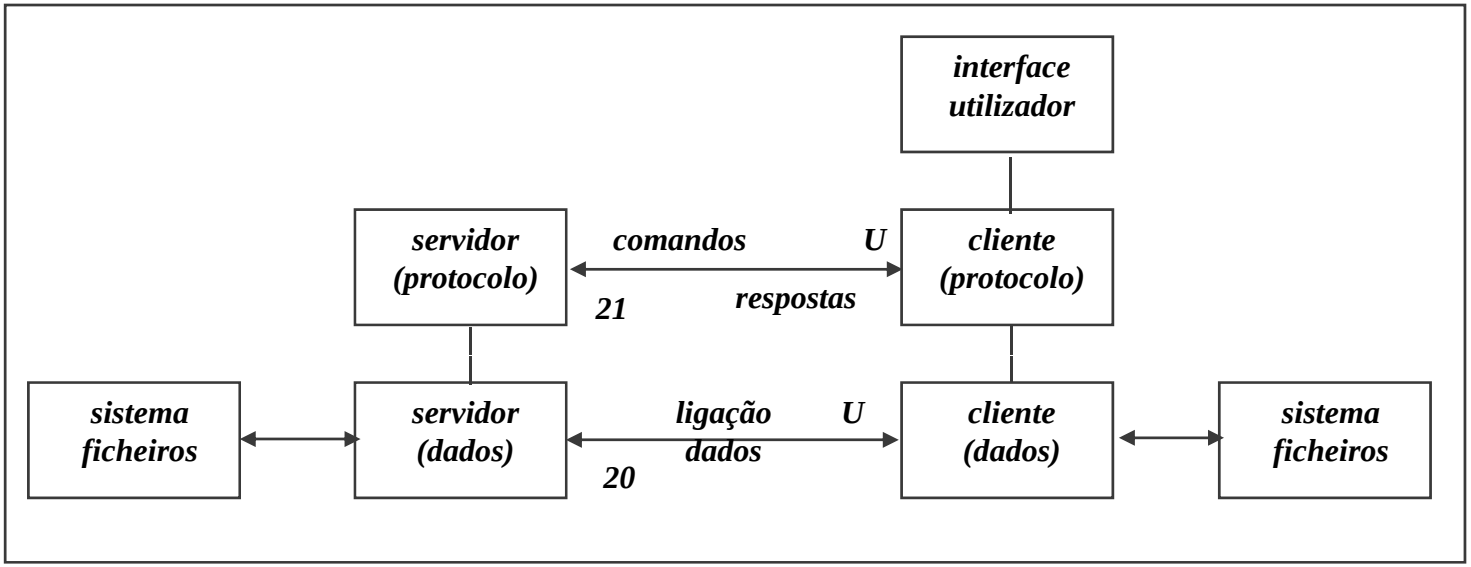
# *MIME – Multipurpose Internet Mail Extensions*

- RFC 2045 – Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies
- RFC 2046 – Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types
- RFC 2047 – MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text
- RFC 2048 – Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures
- RFC 2049 – Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples

# *FTP – File Transfer Protocol*

FTP – File Transfer Protocol (RFC 959)

- Transferência de ficheiros entre computadores (ASCII e binário)
- Modelo de Comunicação Cliente-Servidor
- Conexões TCP independentes para controlo da ligação e transferência de dados



# *FTP – exemplo*

```
LOCAL COMMANDS BY USER              ACTION INVOLVED

ftp (host) multics<CR>              Connect to host S, port L,
                                    establishing control connections.
                                    <---- 220 Service ready <CRLF>.
username Doe <CR>                   USER Doe<CRLF>---->
                                    <---- 331 User name ok,
                                              need password<CRLF>.
password mumble <CR>                PASS mumble<CRLF>---->
                                    <---- 230 User logged in<CRLF>.
retrieve (local type) ASCII<CR>
(local pathname) test 1 <CR>        User-FTP opens local file in ASCII.
(for. pathname) test.pl1<CR>        RETR test.pl1<CRLF> ---->
                                    <---- 150 File status okay;
                                          about to open data
                                          connection<CRLF>.
                                    Server makes data connection
                                    to port U.

                                    <---- 226 Closing data connection,
                                        file transfer successful<CRLF>.
type Image<CR>                      TYPE I<CRLF> ---->
                                    <---- 200 Command OK<CRLF>
store (local type) image<CR>
(local pathname) file dump<CR>      User-FTP opens local file in Image.
(for.pathname) >udd>cn>fd<CR>       STOR >udd>cn>fd<CRLF> ---->
                                    <---- 550 Access denied<CRLF>
terminate                           QUIT <CRLF> ---->
                                    Server closes all
                                    connections.
```

# *WWW – World Wide Web*

- WWW – World Wide Web
  - Acesso a documentos interligados e distribuídos por múltiplos computadores
- Modelo de Comunicação Cliente-Servidor
  - Conexão TCP
  - *Browser* = cliente  httpd = servidor, na porta 80
  - Protocolo – HTTP, *Hyper Text Transport Protocol*
- RFCs: RFC1945 (HTTP 1.0), RFC2616 (HTTP 1.1)
- Exemplo de obtenção de página
  - » URL= *http://www.w3.org/hypertext/WWW/TheProject.html*
    - *Browser* pergunta ao DNS (Domain Name Server) o endereço IP de www.w3.org
    - DNS responde com 18.23.0.23
    - *Browser* estabelece ligação TCP com httpd (em 18.23.0.23, na porta 80)
    - *Browser* envia GET */hypertext/WWW/TheProject.html*
    - Servidor em *www.w3.org* envia ficheiro *TheProject.html*
    - Ligação TCP é terminada
    - *Browser* mostra texto e obtém imagens associadas a *TheProject.html*

# *HTTP 0.9 – mensagens*

- HTTP-message:= Simple-Request | Simple-Response
  - » Simple-Request:= **GET SP** Request-URI **CRLF**
    - Request-URL:= absoluteURI | abs_path
    - absoluteURI:= scheme **:** *( uchar | reserved ) /* *usado em proxies*/*
    - abs_path:= / rel_path
  - » Simple-Response:= [Entity-Body]
    - Entity-Body = *OCTET
- Exemplo
  - » telnet alf.fe.up.pt 80 /* *Estabelecimento da ligação ao servidor* */*
  - » Cliente: GET /lixo.tmp
  - » Servidor: <HTML><HEAD>
    - <TITLE>404 File Not Found</TITLE>
    - </HEAD><BODY>
    - <H1>File Not Found</H1>
    - The requested URL /lixo.tmp was not found on this server.<P>
    - </BODY></HTML>

# *HTTP 1.0 – mensagens*

- HTTP-message:= Simple-Request | Simple-Response | Full-Request | Full-Response
  - » Full-Request:= Request-Line
    *(General-Header | Request-Header | Entity-Header) **CRLF** [ Entity-Body ]
    - ◆ Request-Line:= Method **SP** Request-URI **SP HTTP-**Version **CRLF**
    - ◆ Method:= **GET** | **HEAD** | **PUT** | extension-method
      - – GET – lê página, HEAD – lê cabeçalho da página, PUT – escreve página
      - – *Exemplo: GET /index.html HTTP/1.0*
      - – *Exemplo: GET http://www.inescn.pt/index.html HTTP/1.0*
  - » Full-Response = Status-Line
    *( General-Header | Response-Header | Entity-Header ) **CRLF** [ Entity-Body ]
    - ◆ Status-Line:= HTTP-Version **SP** Status-Code **SP** Reason-Phrase **CRLF**
    - ◆ Status-Code:= **200** | **400** | **404** ** 200= 0k, 400= bad request, 404= not found **
      - – *Exemplo: HTTP/1.0 200 Document follows*

# *HTTP 1.0 – acesso directo (exemplo)*

telnet www.inescn.pt 80 /* *Estabelecimento da ligação ao servidor* */

Cliente:

GET /index.html HTTP/1.0

Servidor:

HTTP/1.0 200 Document follows
Date: Fri, 03 May 1999 15:13:48 GMT
Server: NCSA/1.5
Content-type: text/html

<HTML>
…
</BODY>
</HTML>

# *HTTP 1.0 – acesso via proxy (exemplo)*

telnet **alf.fe.up.pt** 80 /* *Estabelecimento da ligação ao servidor* */

Cliente:

GET http://**www.inescn.pt**/index.html HTTP/1.0

Servidor:

HTTP/1.0 200 Document follows
Date: Fri, 03 May 1999 15:13:48 GMT
Server: NCSA/1.5
Content-type: text/html

<HTML>
…
</BODY>
</HTML>

# *HTTP 1.1 – mensagens*

- HTTP-message:= Request | Response

  - » Request:= Request-Line
    *(General-Header | Request-Header | Entity-Header) **CRLF** [ Entity-Body ]
    - ◆ Request-Line:= Method **SP** Request-URI **SP HTTP-**Version **CRLF**
    - ◆ Method:= **GET | OPTIONS | TRACE | HEAD | DELETE |**
      **PUT | POST | extension-method**
    - ◆ Request-URI = “*” | absoluteURI | abs_path
      - – OPTIONS – informação sobre opções de comunicação do servidor
      - – TRACE – *loopback* da mensagem
      - – DELETE – remoção da página
      - – POST – adição de nova informação no servidor
  - » Response = Status-Line
    *( General-Header | Response-Header | Entity-Header ) **CRLF** [ Entity-Body ]

# *HTTP 1.1 – alguns headers*

- Host, no Request-Header
  - Descreve Host e Porta
  - Exemplo: *Host: www.fe.up.pt*
- Content-Length, no Entity-Header
  - Comprimento em bytes do Entity-body
  - Exemplo: *Content-Length: 1024*
- Content-Type, no Entity-Header
  - Define tipos de mensagens
  - Exemplo: *Content-Type: image/gif*
- If-Modified-Since, no Request-Header
  - Usado com o método GET para obter documentos recentes
  - Exemplo: *If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT*
- Location, no Response-Header
  - Usado para redirecionar clientes para a nova localização do documento
  - Exemplo: *Location: http://www.fe.up.pt/index.html*

# *Exemplo – header HOST*

telnet www.fe.up.pt 80

Cliente:

```
GET /  HTTP/1.1
HOST: www.fe.up.pt
```

Servidor:

```
HTTP/1.1 200 OK
Date: Wed, 14 Nov 2001 13:02:47 GMT
Server: Apache/1.3.20 (Unix) mod_ssl/2.8.4 OpenSSL/0.9.6b
Last-Modified: Thu, 05 Jul 2001 13:55:20 GMT
ETag: "45e5-2d8-3b4471c8"
Accept-Ranges: bytes
Content-Length: 728
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
….
</html>
```

# *Exemplo – método OPTIONS*

telnet sifeup.fe.up.pt 80

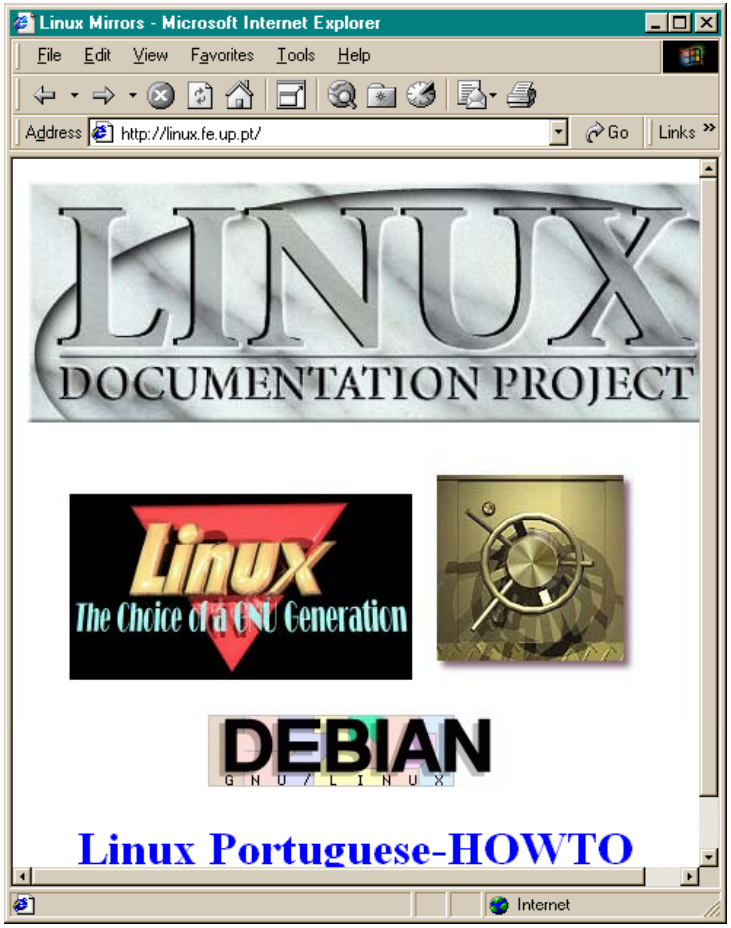
Cliente:

```
OPTIONS * HTTP/1.1
HOST: sifeup.fe.up.pt
```

Servidor:

```
HTTP/1.1 200 OK
Date: Wed, 14 Nov 2001 13:02:50 GMT
Server: Oracle HTTP Server Powered by Apache/1.3.12 (Unix)
    ApacheJServ/1.1 mod_4
Content-Length: 0
Allow: GET, HEAD, OPTIONS, TRACE
```

# *HTML*




```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from url=(0022)http://linux.fe.up.pt/ -->

<HTML>

<HEAD>
<TITLE>Linux Mirrors</TITLE>
<META content="text/html; charset=windows-1252" http-equiv=Content-Type>
<META content="MSHTML 5.00.2014.210" name=GENERATOR>
</HEAD>

<BODY bgColor=#ffffff>
<CENTER>
<A href="http://ae.fe.up.pt/LDP/">
<IMG alt="LDP Project" border=0 src="Linux Mirrors_files/LDP.jpg"
</A>
<P>
<A href="http://linux.fe.up.pt/linapps">
<IMG alt=" LINUX APPS " border=0 src="Linux Mirrors_files/linuxpov_sml.gif">
</A>
<A href="ftp://linux.fe.up.pt/pub/linux">
<IMG alt=" Linux FTP " border=0 src="Linux Mirrors_files/schicon15.jpg">
</A>
<P>
<A href="ftp://ftp.fe.up.pt/packages/unix/linux/debian">
<IMG alt=" Debian" border=0 src="Linux Mirrors_files/debian.jpg">
</A>
<A href="http://linux.fe.up.pt/howto">
<H1>Linux Portuguese-HOWTO </H1>
</A>
</CENTER>
</BODY>

</HTML>
```

# *HTML*

| Tag | Description |
|---|---|
| <HTML> ... </HTML> | Declares the Web page to be written in HTML |
| <HEAD> ... </HEAD> | Delimits the page's head |
| <TITLE> ... </TITLE> | Defines the title (not displayed on the page) |
| <BODY> ... </BODY> | Delimits the page's body |
| <H*n*> ... </H*n*> | Delimits a level *n* heading |
| <B> ... </B> | Set ... in boldface |
| <I> ... </I> | Set ... in italics |
| <UL> ... </UL> | Brackets an unordered (bulleted) list |
| <OL> ... </OL> | Brackets a numbered list |
| <MENU> ... </MENU> | Brackets a menu of <LI> items |
| <LI> | Start of a list item (there is no </LI>) |
| <BR> | Force a break here |
| <P> | Start of paragraph |
| <HR> | Horizontal rule |
| <PRE> ... </PRE> | Preformatted text; do not reformat |
| <IMG SRC="..."> | Load an image here |
| <A HREF="..."> ... </A> | Defines a hyperlink |

A selection of common HTML tags. Some have additional parameters

# *Trabalhos propostos – objectivos*

- Os trabalhos devem
  - » Usar a interface de *sockets*
  - » Utilizar TCP ou UDP
  - » Implementar pelo menos um protocolo de aplicação em conformidade com os RFCs respectivos

- Linguagem de programação – C

- Descrição de objectivos, requisitos mínimos, elementos de valorização, avaliação, demonstração e entrega do relatório
  - » Ver “objectivos” na página da disciplina

# *Exemplos de trabalhos (1)*

- Aplicação FTP
  - » Transferência de ficheiros entre dois servidores controlada por um terceiro sistema (cliente FTP)
- Cliente de *mail* – SMTP + POP3
- *Proxy* HTTP
- Cliente e Servidor HTTP
- Cliente IRC

# *Exemplos de trabalhos (2)*

- Servidor de mail para aviso de ausências
  - Objectivo: ler o correio periodicamente e efectuar o *reply* com um texto indicativo da ausência
  - Argumentos: servidor de POP3, servidor de SMTP, mensagem
- Agenda electrónica
  - Objectivo: permitir a marcação de reuniões / eventos para um conjunto de intervenientes
  - Argumentos: lista de endereços dos intervenientes, assunto, texto da convocatória, data para o envio
- Robot de procura
  - Objectivo: obter endereços em que conste uma ou mais palavras chave a partir de um endereço URL
  - Argumentos: URL de início, profundidade de pesquisa
  - Retorno: endereços URL
- Robot de *download*
  - Objectivo: obter uma cópia local e navegável de uma página, limitado a um dado grau de profundidade, a partir de um endereço URL
  - Argumentos: URL, profundidade de pesquisa
  - Retorno: cópia das páginas pedidas
- Outros
  - A definir ou por proposta de um grupo (requer aprovação)