Internet e protocolos web

- A Internet é uma rede descentralizada de recursos computacionais

- Tolerante a falhas
  (no single point of failure)

- Topologia tem de fornecer caminhos alternativos entre 2 computadores ligados à rede.

TCP/IP

- TCP/IP gere o envio e recepção de mensagens na Internet.

- Comutação de pacotes:

  - Mensagem é dividida em pacotes.
    Cada pacote é enviado de modo independente

  - Mensagem é reconstruída no destino

- TCP/IP fornece um serviço às aplicações que utilizam a Internet.

TCP/IP (cont.)

- A WWW é uma aplicação em rede que utiliza o TCP/IP para comunicar através da Internet.

- Quando um browser pede uma página a um servidor web, o TCP/IP cria uma ligação virtual entre os dois intervenientes.

- A conexão é virtual devido à comutação de pacotes.

- Contrasta com a rede telefónica em que é estabelecida uma conexão dedicada.

Endereços IP

- Cada computador necessita de ser identificado através de um endereço global único (endereço IP).

- Um computador ligado à rede necessita de pelo menos um endereço IP. Um nó que liga duas redes necessita de dois endereços IP.

- Endereços IP são números de 32 bits. (geralmente escritos na forma de 4 bytes separados por um ponto, ex: 134.148.250.28)

Portos

- Quando é criada uma ligação virtual entre 2 computadores, é associado um porto a cada um.

- Permite que possa haver múltiplas conexões a um computador feitas por aplicações diferentes.

- Por defeito, FTP utiliza o porto 21 e HTTP utiliza o porto 80.

HTTP
(Hypertext Transfer Protocol)

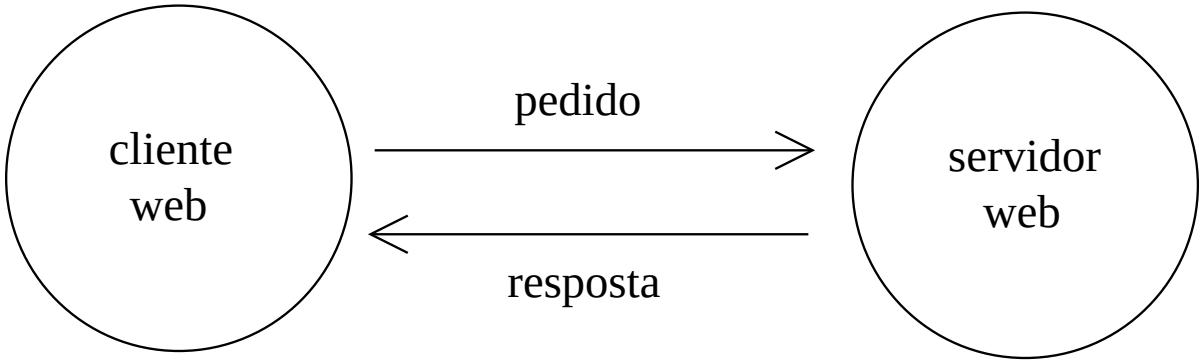
- HTTP é um protocolo que é utilizado para trocar informação na Web.

- HTTP é um protocolo da camada de aplicação (está construído sobre o TCP/IP).

- Primeira versão foi HTTP/0.9.
  Em 1996 apareceu a versão 1.0.
  Em 1999 apareceu a versão 1.1.

Pedidos e respostas HTTP

- Cliente (geralmente um browser web) faz o pedido de um recurso a um servidor HTTP.
- Servidor envia uma resposta de volta.
- A resposta HTTP engloba o recurso pedido (documento HTML, imagem, ...)
- Servidor HTTP = Servidor Web


cliente
web
pedido
resposta
servidor
web

Exemplo de pedido e resposta

- Podemos simular um pedido HTTP e observar a resposta através de uma ligação telnet ao porto 80.

- O pedido é feito e termina com uma linha em branco.

- Exemplo:

Exemplo

```
telnet diana.uceh.ualg.pt 80
Trying 10.10.23.13...
Connected to diana.uceh.ualg.pt.
Escape character is '^]'.
HEAD /index.html HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 15 Mar 2004 08:53:46 GMT
Server: Apache/1.3.26 (Unix) Debian GNU/Linux PHP/4.3.4 mod_
Last-Modified: Sun, 22 Feb 2004 19:11:21 GMT
ETag: "eb37d-1400-4038fed9"
Accept-Ranges: bytes
Content-Length: 5120
Connection: close
Content-Type: text/html; charset=iso-8859-1

Connection closed by foreign host.
```

Outro exemplo

```
telnet diana.uceh.ualg.pt 80
Trying 10.10.23.13...
Connected to diana.uceh.ualg.pt.
Escape character is '^]'.
GET /~figo/ola.html HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 15 Mar 2004 08:58:34 GMT
Server: Apache/1.3.26 (Unix) Debian GNU/Linux PHP/4.3.4 mod_
Last-Modified: Tue, 02 Dec 2003 08:50:14 GMT
ETag: "2112278a-6b-3fcc5246"
Accept-Ranges: bytes
Content-Length: 107
Connection: close
Content-Type: text/html; charset=iso-8859-1

<html>
<head>
<title>teste</title>
</head>

<body>

<p>
Olá, o meu nome é Luís Figo.
</p>

</body>
</html>
Connection closed by foreign host.
```

URLs
(Uniform Resource Locators)

- URL serve para endereçar um recurso na web.

- URL pode ser decomposta em 3 partes:

  1. Protocolo

  2. Host e identificação do serviço

  3. Identificação de um recurso

- Exemplo:
  `http://diana.uceh.ualg.pt/~figo/ola.html`

URLs (cont.)

- Protocolo pode ser:
  - `ftp://`
  - `http://`
  - `https://`
  - ...
- Host e identificação do serviço:

  `http://www.w3.org/Protocols/`

  `é traduzido em:`

  `http://18.29.1.35/Protocols/`

URLs (cont.)

- Por defeito, o HTTP utiliza o porto 80.

  `http://www.w3.org/Protocols/`

  `é equivalente a:`

  `http://www.w3.org:80/Protocols/`

- Podemos ligar-nos a um porto que não seja standard.

  `http://www.exemplo.com:8080`

- Neste caso, o servidor web corre na porta 8080.

URLs (cont.)

Identificação de um recurso

- é constituído por um caminho (path), parâmetros opcionais, e queries opcionais a serem processadas pelo servidor web.

- Geralmente, o path é relativo a um determinado directório no servidor web.

- Exemplo: O servidor web `www.exemplo.com` poderá guardar todos os documentos web debaixo do directório.

  `/usr/local/apache/htdocs`

- A resposta HTTP ao pedido:

`http://www.exemplo.com/marketing/index.html`

- Contem o ficheiro:

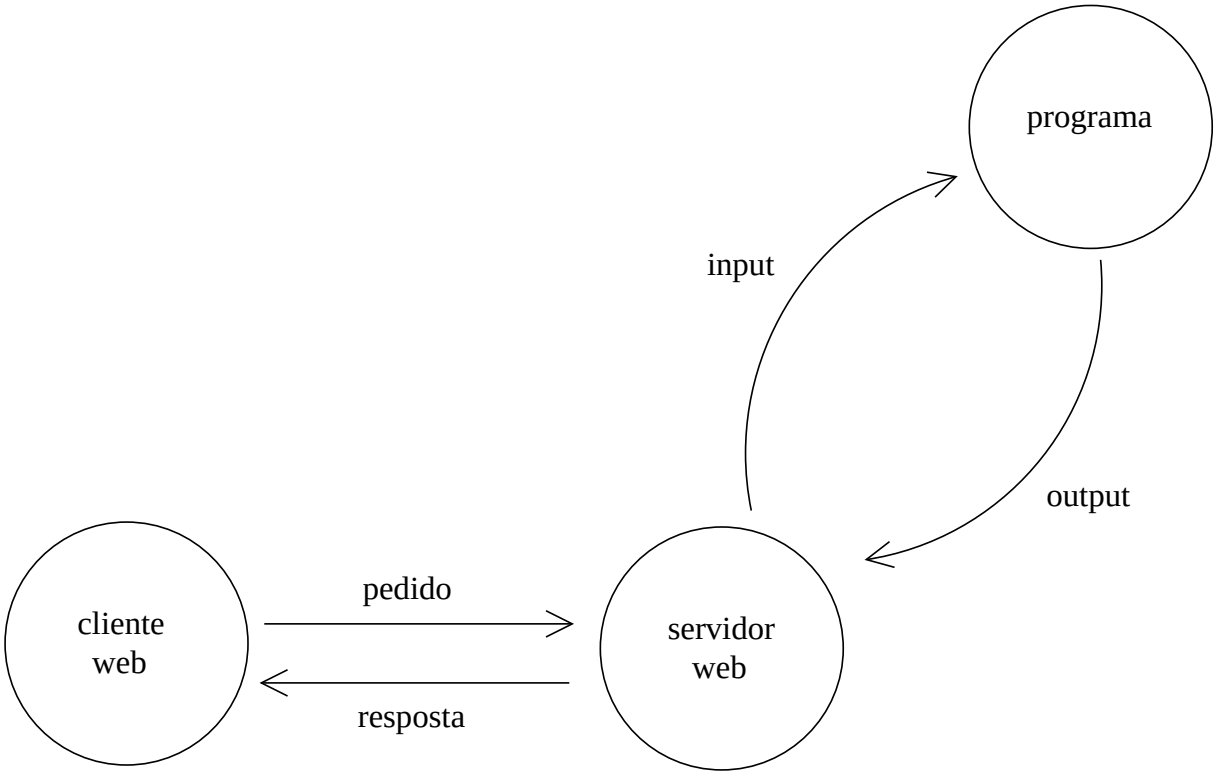
  `/usr/local/apache/htdocs/marketing/index.html`

- O caminho pode incluir queries a serem processadas pelo servidor web.

  `http://www.exemplo.com/search?q=red&r=victoria`

- A informação `q=red` e `r=victoria` pode ser usada por um script

Servidor web pode invocar um programa externo



URL encoding

- Caracteres especiais (ex: `;` `/` `?` `:` `&` `%`) têm de ser codificados na URL.

- Utiliza-se % seguido de 2 dígitos hexadecimais que representam o código ASCII do caracter.

- Exemplo:

```
"100% + more" --> "100%25%20%2B%20more"
```

Pedidos HTTP

- Um pedido HTTP contem um nome de um método, o recurso pretendido, e um conjunto de cabeçalhos.

- Alguns pedidos poderão ter um corpo de mensagem (ex: dados de um formulário).

- HTTP define vários métodos.

- Os mais usados são GET, POST, e HEAD.

Pedidos HTTP (cont.)

- GET - devolve o recurso.
  Podemos usar uma query para adicionar informação extra ao pedido (ex: parâmetros para uma pesquisa a uma BD)

- POST - envia dados para o servidor.
  Os dados são enviados no corpo da mensagem. Não são colocados no URL.

- HEAD - retorna os cabeçalhos associados ao recurso.

GET versus POST

Utilizar POST se:

- O resultado do pedido tem efeitos persistentes no servidor (ex: modifica algo numa BD)

- houver bastantes dados a serem enviados (formulários compridos).

Utilizar GET se:

- se pretende apenas fazer uma pesquisa.

- não há efeitos persistentes no servidor.

- os dados dos "input fields" têm menos de 1K.

Respostas HTTP

Resposta HTTP contem:

- Linha de status
- Cabeçalhos
- Corpo da mensagem
  (geralmente é o recurso pedido)

Respostas HTTP (cont.)

Status codes:

codes	meaning
100-199	Informational
200-299	Client request successful
300-399	Client request redirected
400-499	Cliente request error
500-599	Server errors