WEB E HTTP

DESCRIÇÃO GERAL DO HTTP:

Como era usada a Internet até a década de 90?

- Quem tinha acesso eram apenas: Pesquisadores, acadêmicos e estudantes universitários.
- As únicas funcionalidades possíveis para a internet eram transferência de arquivos, notícias e correio eletrônico.

Nova aplicação a WWW (World Wide Web):

- **Funcionando por Demanda** - O usuário só recebe algo que ele pedir, nada extra.
- **Qualquer pessoa** pode colocar informações na Web.
- **Áudio e Vídeo** disponível na Web.

Protocolo HTTP [RFC 1945 e 2616] (Hyper Text Transfer Protocol):

- Pertencente a **Camada de Aplicação da Web.**
- Implementado com o sistema de **Cliente** e **Servidor.**
- Os dois trocam *mensagens HTTP.*
- O protocolo que vai definir como as mensagens serão estruturadas e o modo como vão ser compartilhadas.

TERMINOLOGIA DA WEB:

- *A página Web é estruturada a partir de objetos*

  - Um objeto é um arquivo, naturalmente podendo ser de diferentes tipos, como:
    - .html
    - .jpeg
    - .js
    - .gif
    - .css
    - .java
    - .wav
    - etc.

  - A forma de acessar a Web é simples, utilizando uma **URL (Uniforme Resource Locator),** a mesma sendo sempre constituída por um arquivo-base **HTML (Hyper Text Model Language)** que referencia outros arquivos.

  - Cada URL tem **dois componentes:**
    - **Host:** Servidor que possui o objeto inicial.
    - **Caminho**: Caminho até chegar ao objeto desejado no servidor.

  - Além de tudo ainda temos os **Browsers (Ferramentas para acessar a Web como usuário) ou Navegadores:**
    - **Chrome.**
    - **Edge**
    - **Firefox**
    - **Opera**

IDEIA GERAL DA INTERAÇÃO ENTRE CLIENTE E SERVIDOR:

- Até 1997: **HTTP/1.0 [RFC 1945]**
- A partir de 1998: **HTTP/1.1 [RFC 2616]**
- o HTTP **usa TCP**, dessa forma ele consegue fazer o seguinte:
  - Não precisa ter preocupação com dados perdidos
  - Não precisa saber como que o TCP vai recuperar os dados perdidos nem reordenar os mesmos quando alcançados.
- O HTPP não possui estado, dessa forma ele manda os arquivos sem guardar nenhuma informação sobre os clientes, além de não precisar avisar que já enviou um objeto quando o envia de novo.
-

CONEXÕES PERSISTENTES E NÃO PERSISTENTES:

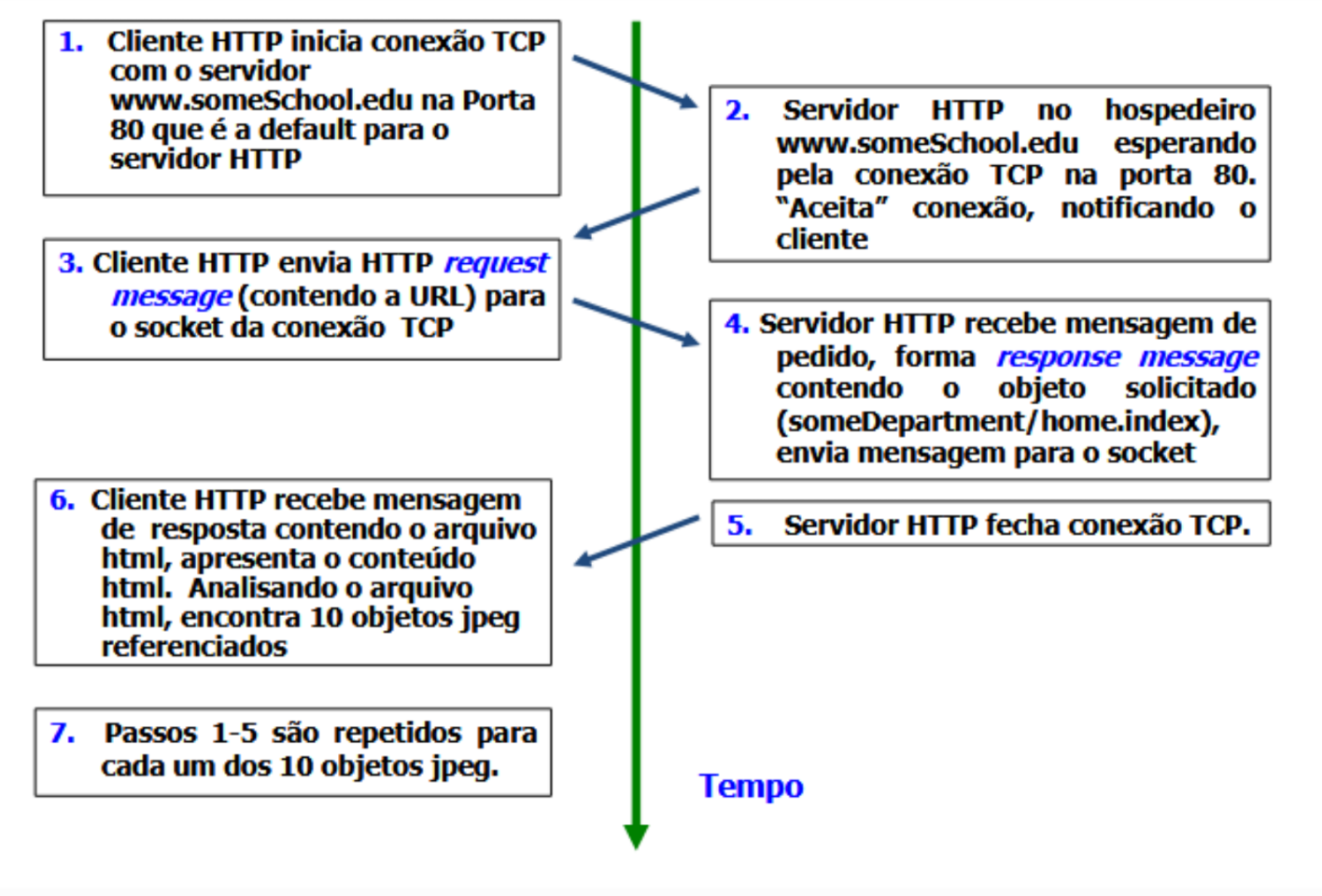
Conexões Persistentes:

- Vários objetos podem ser enviados com apenas uma conexão TCP.
- O HTTP/1.1 sempre está com conexão persistente no modo padrão.

Conexões Não Persistentes:

- No máximo um objeto por vez pode ser enviado pela conexão TCP.
- O HTTP/1.0 sempre usa conexão não persistent*e.*

Exemplo de uma Transferência de uma Página Web Server-Client em uma Conexão Não Persistente:



RTT (Round Trip Time)

- O RTT é o tempo de viagem de um pacote que saí do cliente, vai até o servidor e volta até o cliente. (incluindo os atrasos)
- Temos então nesse caso acima:
  - Um RTT para o início da conexão entre o cliente e o servidor.
  - Um RTT para a Requisição HTTP e a Resposta HTTP.
  - O tempo de transmissão do arquivo.

TEMPO TOTAL = 2RTT + Tempo de Transmissão

Desvantagens da Conexão Não Persistente:

- Sempre é criada uma nova conexão para cada objeto que for solicitado.
- Toda conexão tem diversos buffers e variáveis TCP.
- Tempo Total de 2RTT + Tempo de Transmissão.

Enquanto isso nas Conexões Persistentes:

- O servidor não fecha a conexão TCP depois de enviar a resposta.
- Todas as mensagens HTTP que vierem depois entre o mesmo cliente-servidor, vão usar a mesma conexão.

Conexão Persistente com Paralelismo e sem Paralelismo:

Com Paralelismo:

- É o padrão do HTTP/1.1
- O cliente fica enviando requisições sempre que encontra um objeto referenciado dentro do objeto original.
- Possuindo várias requisições ao mesmo tempo.

Sem Paralelismo:

- O cliente emite uma nova requisição apenas quando receber uma resposta de uma requisição anterior.
- Possuindo apenas uma requisição por vez.

FORMATO DA MENSAGEM HTTP:

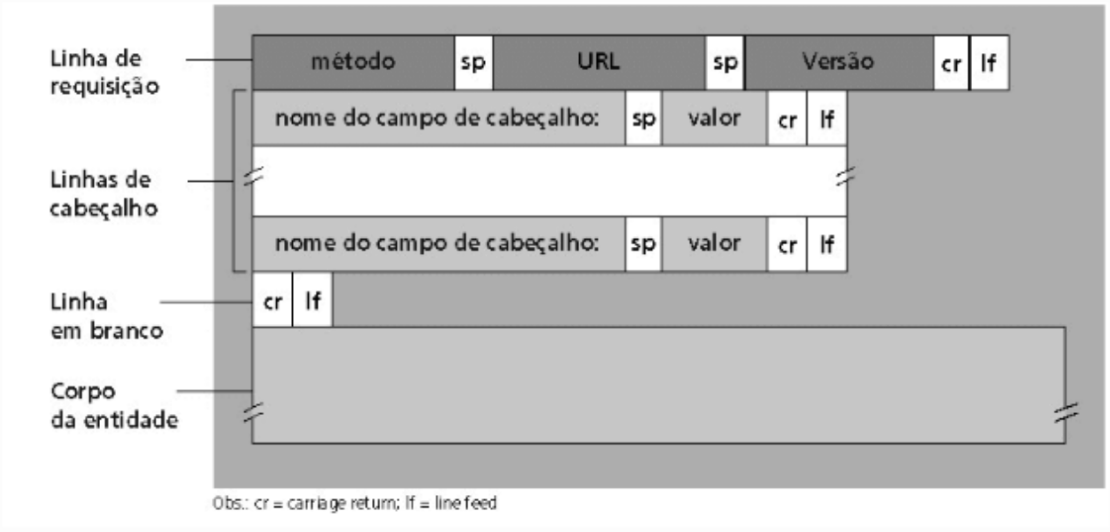
TIPOS DE MENSAGENS HTTP:

Requisição: É escrita em ASCII, então as pessoas podem ler e analisar. possuindo 5 linhas, cada uma dessa linhas possuem um 'carriage return' e 'line feed', que indicam o começo de uma nova linha.

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/4.0
Accept-language: fr
```

Analisando...

- Na primeira linha tempos 3 campos:
  - O **método** (GET, POST, HEAD, PUT)
  - O campo da **URL**
  - O campo da **versão HTTP**
- Depois temos linhas de cabeçalho, sendo elas:
  - **Host:** Onde o objeto reside.
  - **Connection**: diz para o servidor usar ou não usar conexões persistentes. (fechar a conexão tcp ou abrir)
  - **User-agent:** Especifica o Browser e a versão do mesmo.
  - **Accept-language:** Campo preenchido quando o usuário prefere uma versão em alguma língua específica.



- Vendo a requisição esquematizada, percebemos que tem um campo vazio, o **Corpo da Entidade,** o mesmo está vazio pois é uma requisição GET, se fosse uma POST, nessa parte do campo teríamos as informações que o usuário enviou.

Métodos:

- ***GET:*** A entrada é enviada na URL, querendo receber apenas uma busca.
- ***POST:*** O cliente preenche um formulário com dados que serão enviados para o servidor.
- ***HEAD:*** Responde a mensagem HTTP, mas deixa o objeto requisitado fora.
- ***PUT:*** Permite que o cliente envie um objeto para um caminho por meio da URL. (Disponível apenas no HTTP/1.1)
- ***DELETE:*** Permite que o cliente apague algum arquivo de dentro do servidor através da URL. (Disponível apenas no HTTP/1.1)

Resposta: A resposta, diferente da Requisição, sempre terão 6 linhas de cabeçalho.

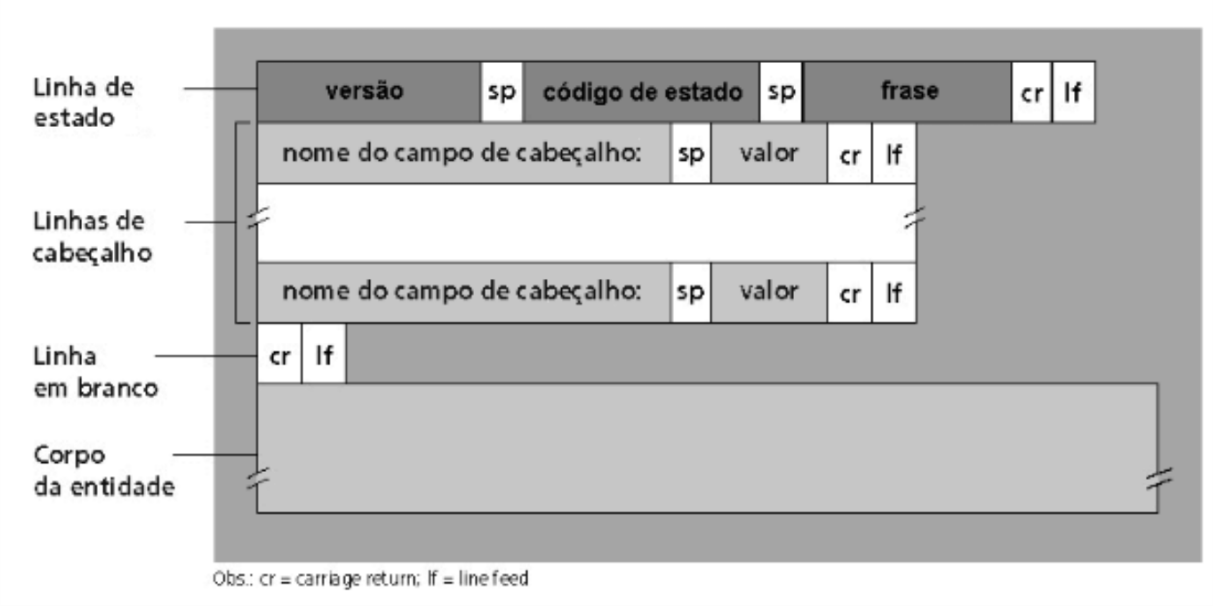
```
HTTP/1.1 200 Ok
Connection: close
Date: Thu, 03 Jul 2003 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Sun, 5 May 2003 09:23:24 GMT
Content-Length: 6821
Content-type: text/html

(data data data data ...)
```

Linha de estado:

- Versão do Protocolo HTTP
- Um código de estado (200)
- Uma mensagem que descreve o estado (Ok)

Linhas de cabeçalho:

- **Connection:** Indica o que o servidor irá fazer depois de enviar a mensagem.
- **Date:** Indica a data e hora em que a resposta foi criada e enviada.
- **Server:** Indica o tipo de servidor web que está sendo usado.
- **Last-Modified:** Hora e data em que o objeto foi criado ou teve uma modificação.
- **Content-Length:** Número de bytes do objeto.
- **Content-Type:** Mostra o tipo do objeto.



TIPOS DE CÓDIGOS E SEUS SIGNIFICADOS:

- **200 OK:** Requisição bem sucedida, objeto entregue.
- **301 Moved Permanently:** Objeto requisitado foi removido, nova localização no campo Location.
- **400 Bad Request:** Requisição não conseguiu ser compreendida.
- **404 Not Found:** O documento não existe no servidor.
- **505 HTTP Version Not Supported:** A versão HTTP requisitada não é suportada pelo servidor.

GET CONDICIONAL:

- Serve para o cache verificar se seus objetos estão atualizados.

- Ela é um GET condicional se:
  - Usar o método **GET**.
  - Possuir no cabeçalho a linha **If-modified-since:**

- Quando ocorre uma mensagem de requisição a um servidor Web, o cache guarda uma cópia do objeto localmente e envia uma cópia para o cliente.

- Se em uma outra mensagem, o mesmo objeto é requisitado e ele ainda está no cache, é enviado o **GET Condicional.**

- A linha IF-modified-since é exatamente igual a Last-

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 2 Jul 2003 09:23:24
```

  modified, que o servidor envia como resposta.

- Se o objeto não sofreu alteração, é enviado a resposta com o código 304 Not Modified, que indica que o objeto não sofreu modificação.