REDES DE COMPUTADORES E A INTERNET 5ª edição
Uma Abordagem Top-Down

2.2. A WEB E O HTTP

Prof. Othon Marcelo Nunes Batista
Mestre em Informática

Capítulo 2: Camada de aplicação

REDES DE COMPUTADORES E A INTERNET 5ª edição

Uma Abordagem Top-Down

- 2.1 Princípios de aplicações de rede
- 2.2 A Web e o HTTP
- 2.3 FTP
- 2.4 Correio eletrônico
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 Aplicações P2P
- 2.7 Programação de sockets com UDP
- 2.8 Programação de sockets com TCP

Web e HTTP

primeiro, algum jargão

- página Web consiste em objetos
- objeto pode ser arquivo HTML, imagem JPEG, applet Java, arquivo de áudio,...
- página Web consiste em arquivo HTML básico que inclui vários objetos referenciados
- cada objeto é endereçável por um URL
- exemplo de URL:

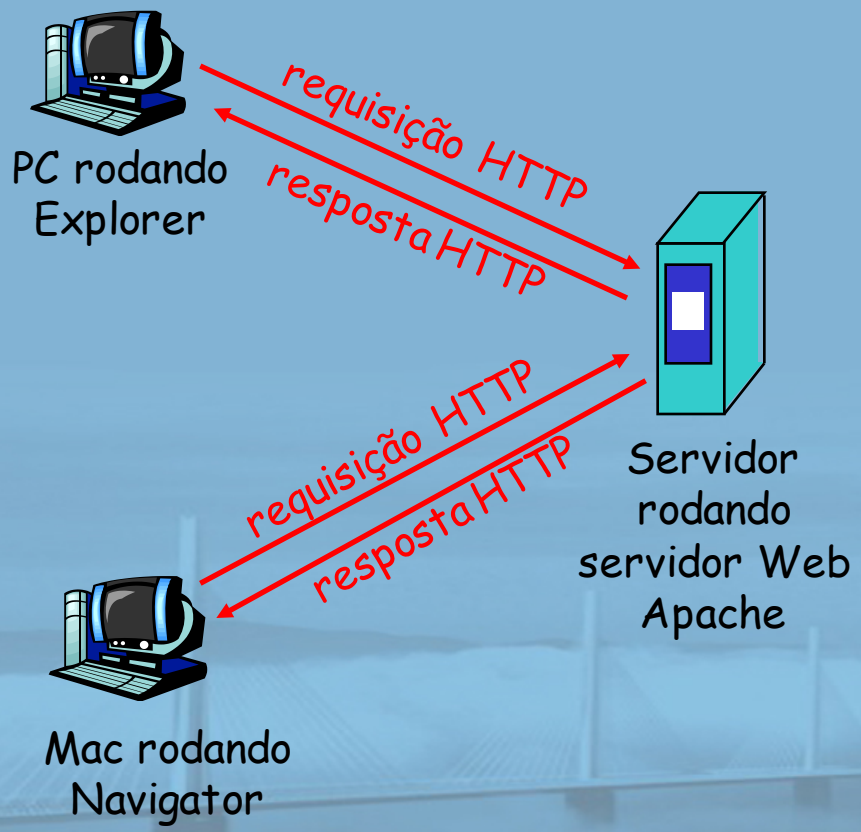
`www.someschool.edu/someDept/pic.gif`

nome do hospedeiro | nome do caminho

Visão geral do HTTP

HTTP: HyperText Transfer Protocol

- protocolo da camada de aplicação da Web
- modelo cliente/servidor
  - *cliente:* navegador que requisita, recebe, "exibe" objetos Web
  - *servidor:* servidor Web envia objetos em resposta a requisições



usa TCP:

- cliente inicia conexão TCP (cria socket) com servidor, porta 80
- servidor aceita conexão TCP do cliente
- mensagens HTTP (do protocolo da camada de aplicação) trocadas entre navegador (cliente HTTP) e servidor Web (servidor HTTP)
- conexão TCP fechada

HTTP é "sem estado"

- servidor não guarda informações sobre requisições passadas do cliente

aparte

Protocolos que mantêm "estado" são complexos!

- história passada (estado) deve ser mantida
- se servidor/cliente falhar, suas visões do "estado" podem ser incoerentes, devem ser reconciliadas

Conexões HTTP

HTTP não persistente

- no máximo um objeto é enviado por uma conexão TCP.

HTTP persistente

- múltiplos objetos podem ser enviados por uma única conexão TCP entre cliente e servidor.

HTTP não persistente

Suponha que o usuário digite o URL
`www.someSchool.edu/someDepartment/home.index`

(contém texto, referências a 10 imagens JPEG)

1a. Cliente HTTP inicia conexão TCP com servidor HTTP (processo) em www.someSchool.edu na porta 80.

1b. Servidor HTTP no hospedeiro www.someSchool.edu esperando conexão TCP na porta 80. "aceita" conexão, notificando cliente

2. Cliente HTTP envia *mensagem de requisição* HTTP (contendo URL) pelo socket de conexão TCP. Mensagem indica que cliente deseja o objeto someDepartment/home.index.

3. Servidor HTTP recebe mensagem de requisição, forma *mensagem de resposta* contendo objeto requisitado e envia mensagem para seu socket

tempo

HTTP não persistente

4. Servidor HTTP fecha conexão TCP.

5. Cliente HTTP recebe mensagem de resposta contendo arquivo html, exibe html. Analisando arquivo html, acha 10 objetos JPEG referenciados.

tempo

6. Etapas 1-5 repetidas para cada um dos 10 objetos JPEG.

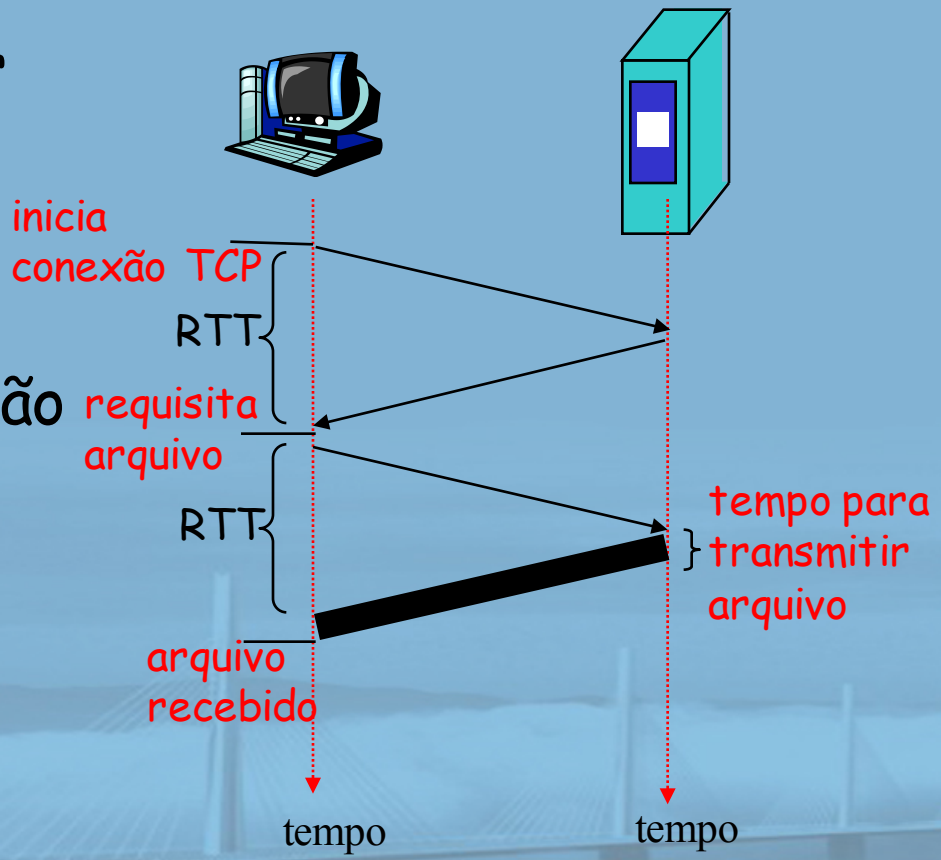
HTTP não persistente: tempo de resposta

definição de RTT: tempo para um pequeno pacote trafegar do cliente ao servidor e retornar.

tempo de resposta:

- um RTT para iniciar a conexão TCP
- um RTT para a requisição HTTP e primeiros bytes da resposta HTTP retornarem
- tempo de transmissão de arquivo

total = 2RTT + tempo de transmissão



HTTP persistente

problemas do HTTP não persistente:

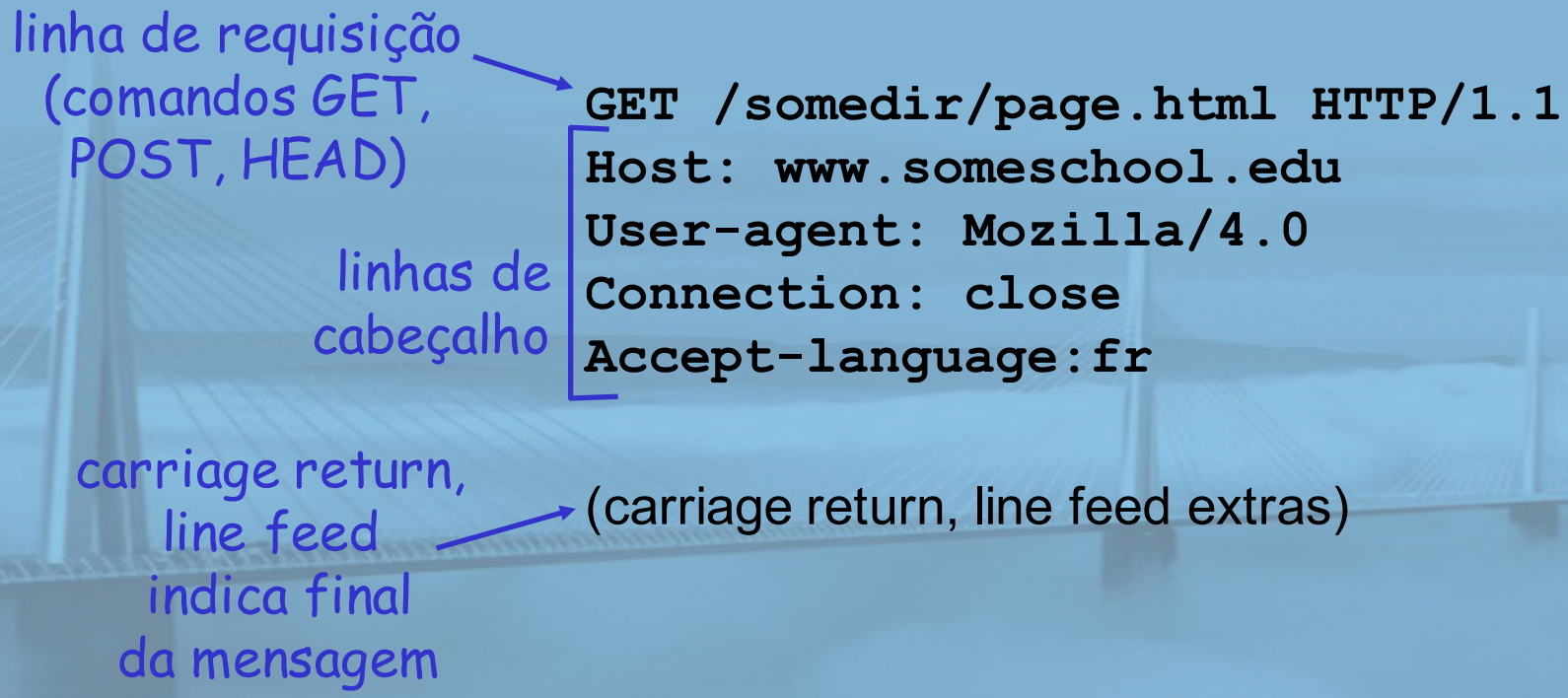
- requer 2 RTTs por objeto
- overhead do SO para *cada* conexão TCP
- navegadores geralmente abrem conexões TCP paralelas para buscar objetos referenciados

HTTP persistente:

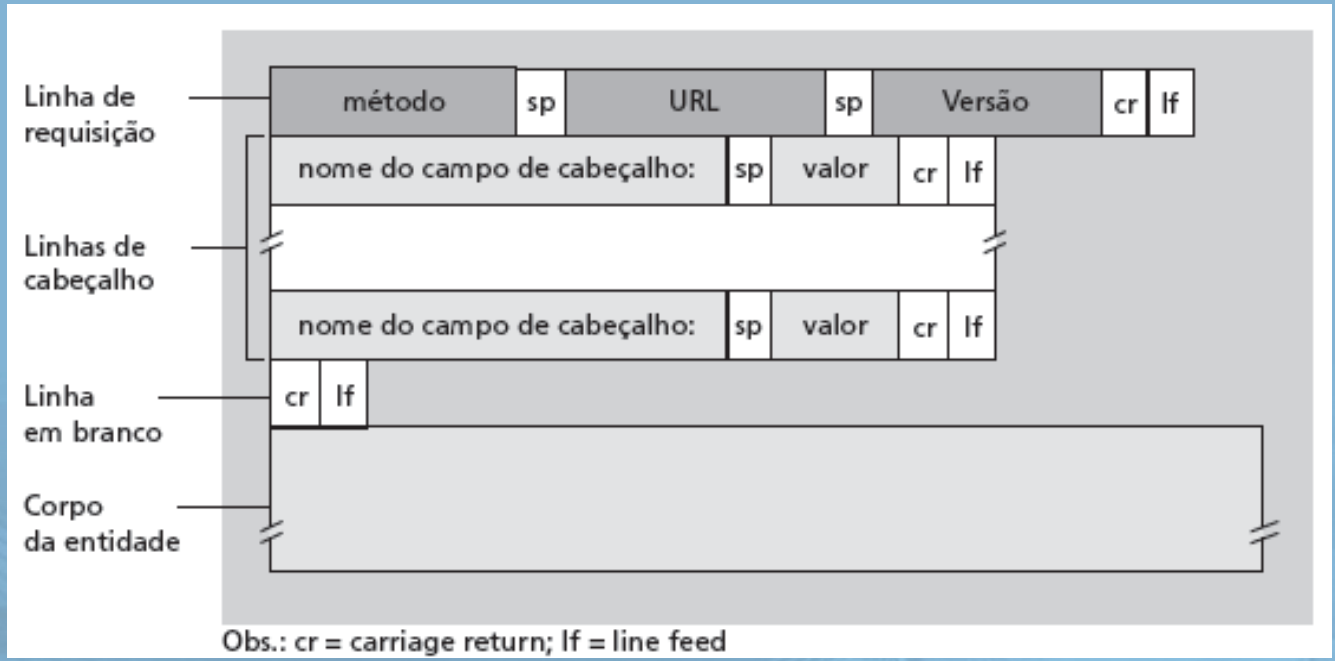
- servidor deixa a conexão aberta depois de enviar a resposta
- mensagens HTTP seguintes entre cliente/servidor enviadas pela conexão aberta
- cliente envia requisições assim que encontra um objeto referenciado
- no mínimo um RTT para todos os objetos referenciados

Mensagem de requisição HTTP

- dois tipos de mensagens HTTP: *requisição, resposta*
- mensagem de requisição HTTP:
  - ASCII (formato de texto legível)



Mensagem de requisição HTTP: formato geral



Obs.: cr = carriage return; lf = line feed

Upload da entrada do formulário

método POST:

- página Web geralmente inclui entrada do formulário
- entrada é enviada ao servidor no corpo da entidade

método do URL:

- usa o método GET
- entrada é enviada no campo de URL da linha de requisição:

```
www.umsite.com/buscaanimal?macacos&banana
```

Tipos de método

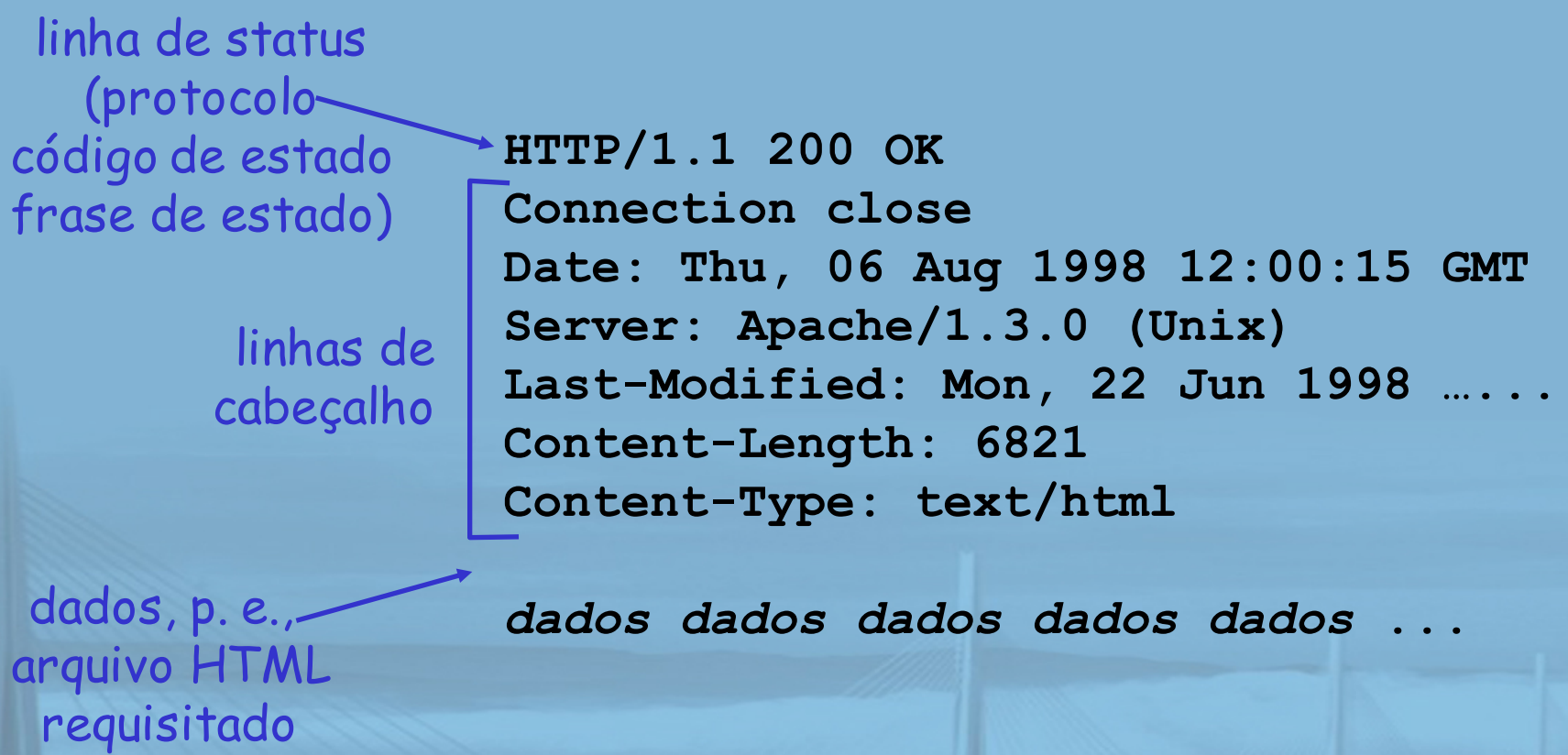
HTTP/1.0

- GET
- POST
- HEAD
  - pede ao servidor para deixar objeto requisitado fora da resposta

HTTP/1.1

- GET, POST, HEAD
- PUT
  - envia arquivo no corpo da entidade ao caminho especificado no campo de URL
- DELETE
  - exclui arquivo especificado no campo de URL

Mensagem de resposta HTTP



Códigos de estado da resposta HTTP

primeira linha da mensagem de resposta servidor->cliente

alguns exemplos de código:

200 OK

- requisição bem-sucedida, objeto requisitado mais adiante

301 Moved Permanently

- objeto requisitado movido, novo local especificado mais adiante na mensagem (Location:)

400 Bad Request

- mensagem de requisição não entendida pelo servidor

404 Not Found

- documento requisitado não localizado neste servidor

505 HTTP Version Not Supported

Testando o HTTP (lado cliente) você mesmo

1. Use Telnet para seu servidor Web favorito:

```
telnet cis.poly.edu 80
```

Abre conexão TCP com porta 80 (porta HTTP default do servidor) em cis.poly.edu. Qualquer coisa digitada é enviada à porta 80 em cis.poly.edu

2. Digite uma requisição HTTP GET:

```
GET /~ross/ HTTP/1.1
Host: cis.poly.edu
```

Digitando isto (pressione carriage return duas vezes), você envia esta requisição GET mínima (mas completa) ao servidor HTTP

3. Veja a mensagem de resposta enviada pelo servidor HTTP!

Estado usuário-servidor: cookies

Muitos sites importantes usam cookies

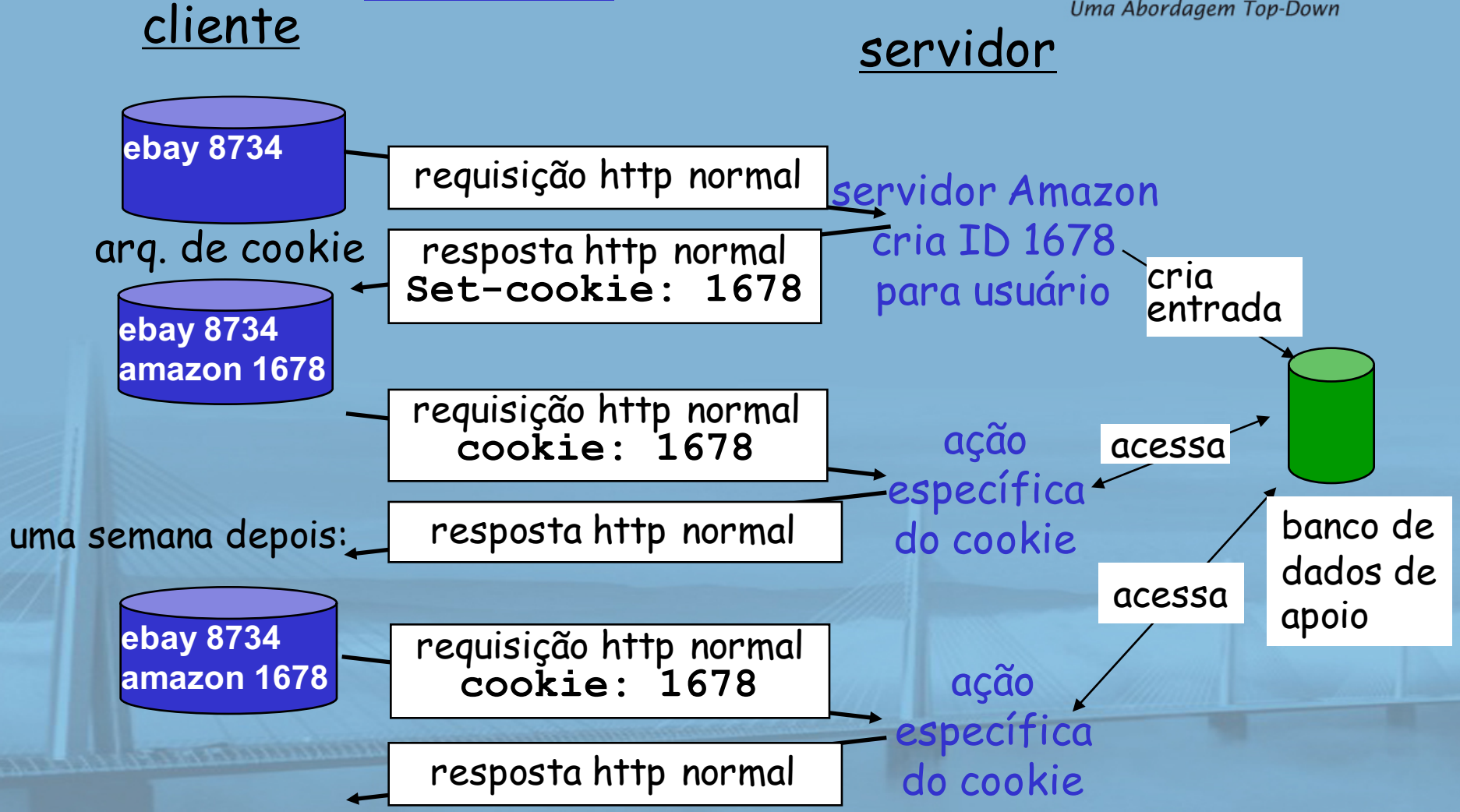
Quatro componentes:

1) linha de cabeçalho de cookie da mensagem de *resposta* HTTP
2) linha de cabeçalho de cookie na mensagem de *requisição* HTTP
3) arquivo de cookie na máquina do usuário, controlado pelo navegador do usuário
4) banco de dados de apoio no site Web

Exemplo:

- Susana sempre acessa a Internet pelo PC
- visita um site de comércio eletrônico pela primeira vez
- quando as primeiras requisições HTTP chegam ao site, este cria:
  - ID exclusivo
  - entrada no banco de dados de apoio para o ID

Estado usuário-servidor: cookies



Estado usuário-servidor: cookies

O que os cookies podem ter:

- autorização
- carrinhos de compras
- recomendações
- estado da sessão do usuário (e-mail da Web)

aparte

Cookies e privacidade:

- cookies permitem que os sites descubram muito sobre você
- você pode fornecer nome e e-mail aos sites

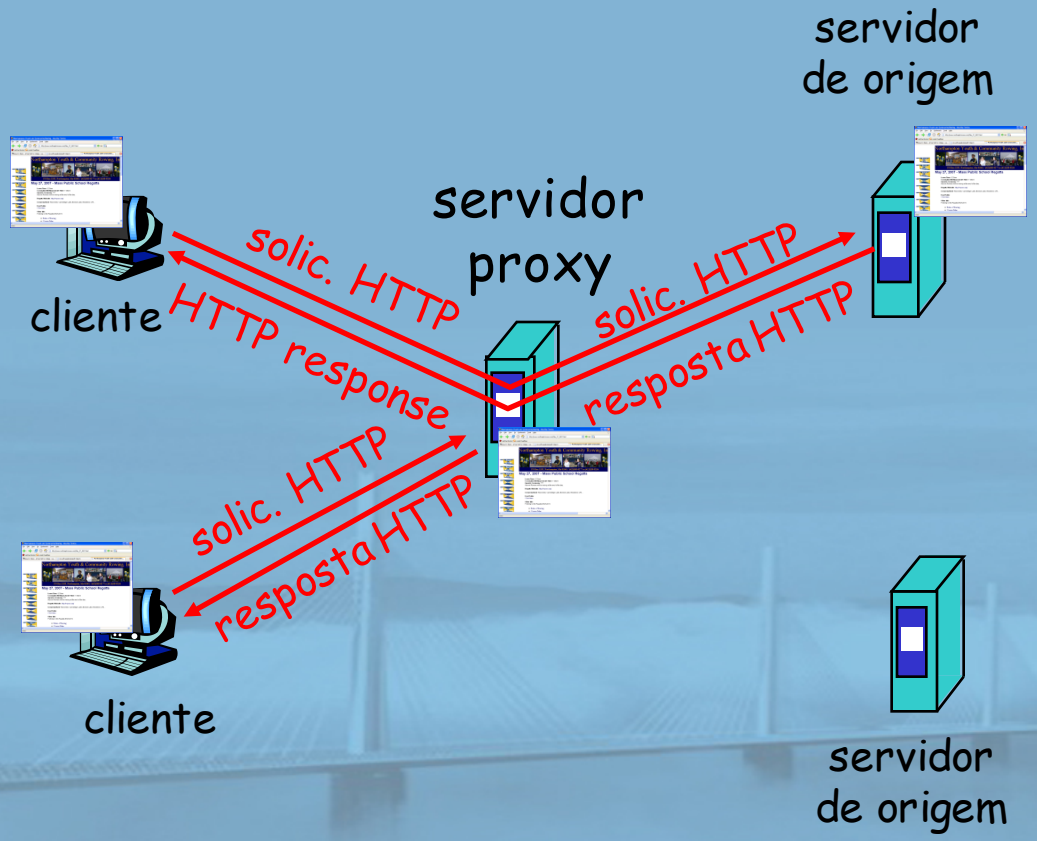
Como manter o "estado":

- extremidades do protocolo: mantêm estado no emissor/receptor por múltiplas transações
- cookies: mensagens HTTP transportam estado

Caches Web (servidor proxy)

objetivo: satisfazer a requisição do cliente sem envolver servidor de origem

- usuário prepara navegador: acessos à Web via cache
- navegador envia todas as requisições HTTP ao cache
  - objeto no cache: cache retorna objeto
  - ou cache requisita objeto do servidor de origem, depois retorna objeto ao cliente



Mais sobre caching Web

- cache atua como cliente e servidor
- normalmente, cache é instalado por ISP (da universidade, empresa, residencial)

Por que caching Web?

- reduz tempo de resposta à requisição do cliente
- reduz tráfego no enlace de acesso de uma instituição
- Internet densa com caches: permite que provedores de conteúdo "fracos" remetam conteúdo efetivamente (mas o mesmo ocorre com compartilhamento de arquivos P2P)

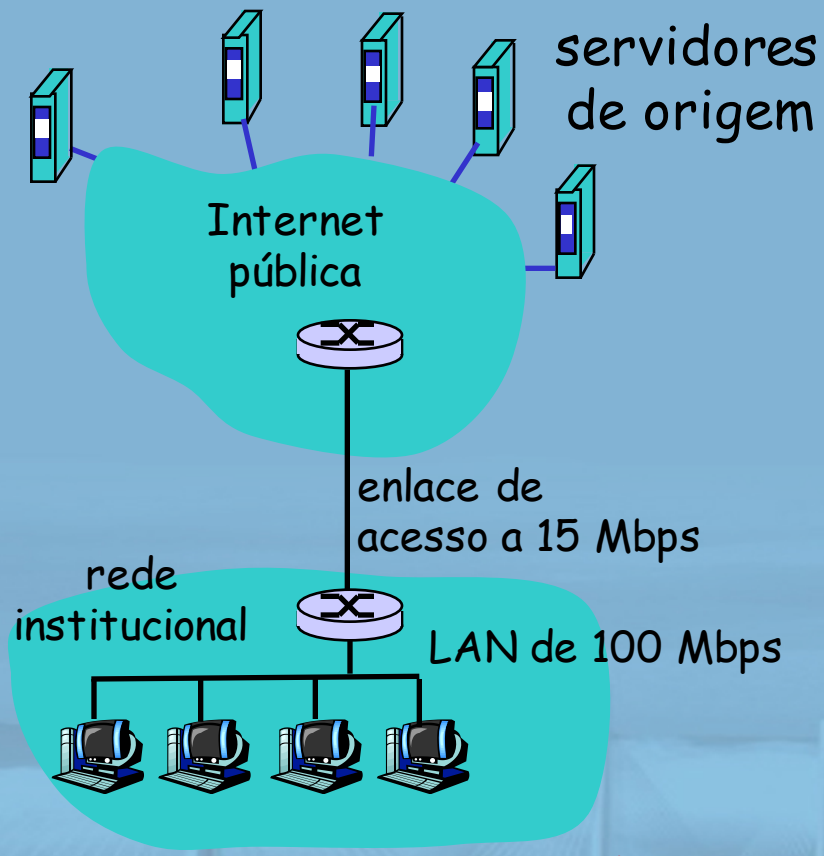
Exemplo de caching

suposições

- tamanho médio do objeto = 1.000.000 bits
- taxa de requisição média dos navegadores da instituição aos servidores de origem = 15/s
- atraso do roteador institucional a qualquer servidor de origem e de volta ao roteador = 2 s

consequências

- utilização na LAN = 15%
- utilização no enlace de acesso = 100%
- atraso total = atraso da Internet + atraso do acesso + atraso da LAN
  = 2 s + x minutos + y milissegundos



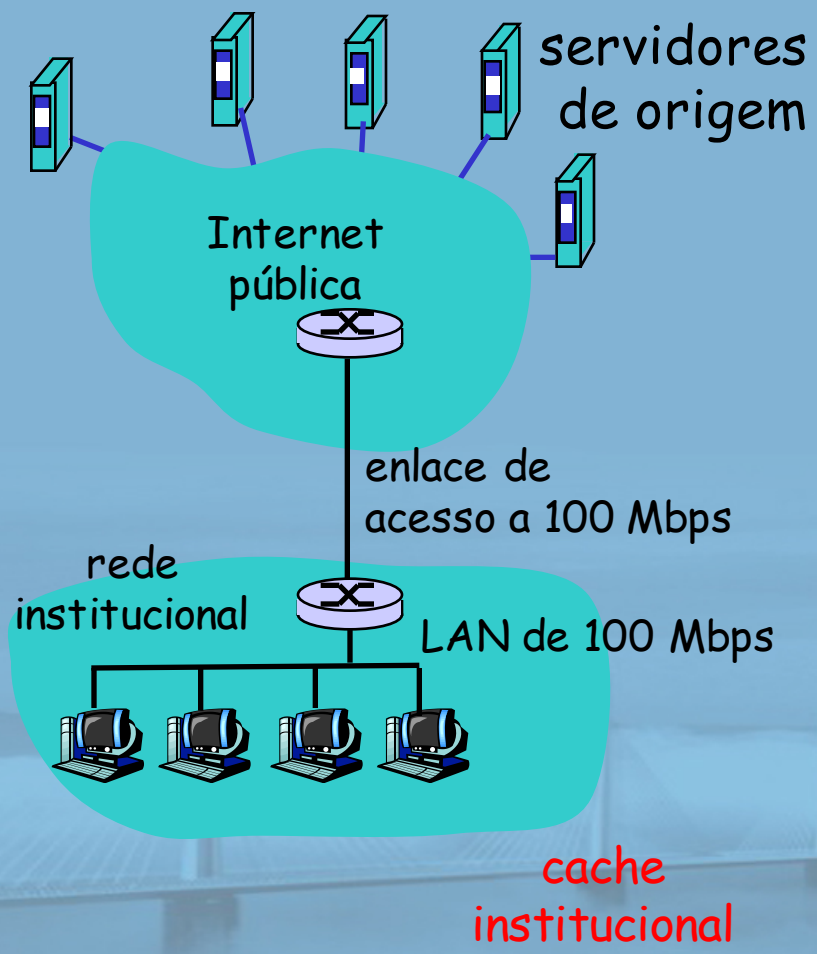
Exemplo de caching

solução possível

- aumentar largura de banda do enlace de acesso para, digamos, 100 Mbps

consequência

- utilização na LAN = 15%
- utilização no enlace de acesso = 15%
- atraso total = atraso da Internet + atraso do acesso + atraso da LAN = 2 s + x ms + y ms
- normalmente, uma atualização dispendiosa



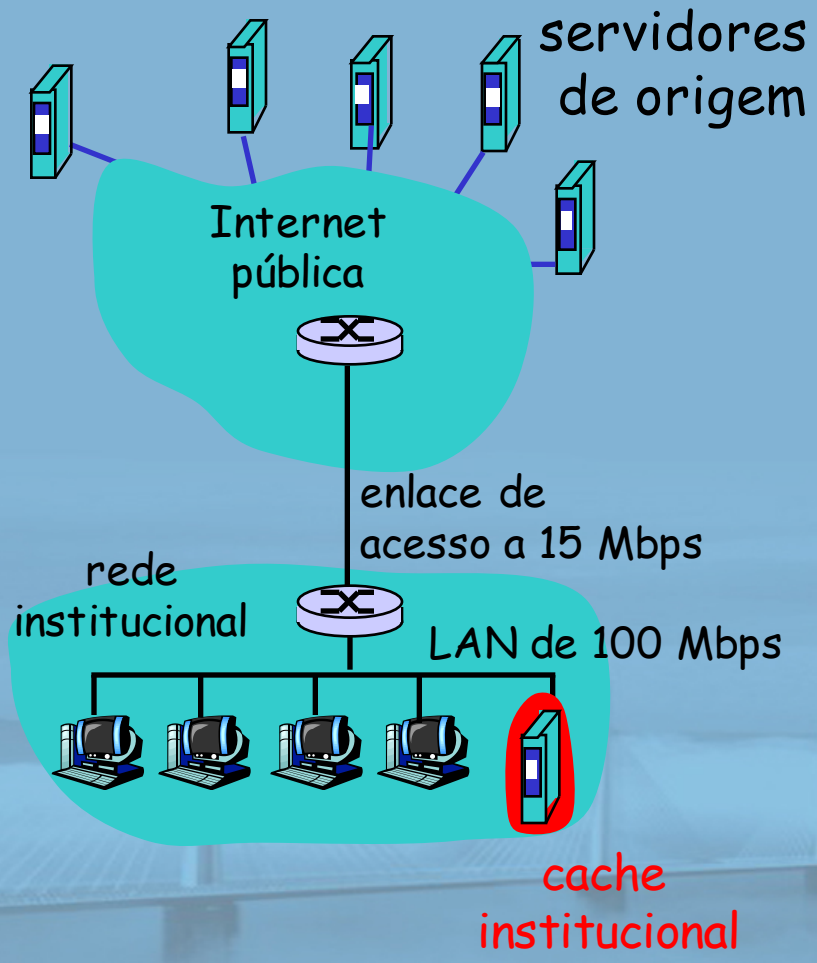
Exemplo de caching

possível solução: instalar cache

- suponha que índice de acerto é 0,4

consequência

- 40% de requisições serão satisfeitas imediatamente
- 60% de requisições satisfeitas pelo servidor de origem
- utilização do enlace de acesso reduzida para 60%, resultando em atrasos insignificantes (digamos, 10 ms)
- atraso médio total = atraso da Internet + atraso de acesso + atraso da LAN = 0,6*(2,01) s + 0,4*milissegundos < 1,4 s



GET condicional

- objetivo: não enviar objeto se o cache tiver versão atualizada
- cache: especifica data da cópia em cache na requisição HTTP

```
If-modified-since:
   <data>
```

- servidor: resposta não contém objeto se a cópia em cache estiver atualizada:

```
HTTP/1.0 304 Not
   Modified
```