Redes de Computadores

Parte 05 – camada de aplicação – *Web*

Prof. Kleber Vieira Cardoso

INSTITUTO DE INFORMÁTICA
UFG

Tópicos

- Visão geral do HTTP
- Conexões persistentes e não-persistentes
- Formato da mensagem HTTP
- Interação usuário-servidor: *cookies*
- *Web caching*
  - GET condicional

A *Web* e o HTTP

Algumas definições:

- Páginas *Web* consistem de objetos
- Objeto pode ser um arquivo HTML, uma imagem JPEG, um *applet* Java, um arquivo de áudio,...
- Páginas *Web* consistem de um arquivo base HTML que inclui vários objetos referenciados
- Cada objeto é endereçável por uma URL (*Uniform Resource Locator*)
- Exemplo de URL:

`www.universidade.br/diretorio/arq.html`

nome do hospedeiro | nome do caminho

Protocolo HTTP

HTTP: *HyperText Transfer Protocol*

- Protocolo da camada de aplicação da *Web*
- modelo cliente/servidor
  - *cliente:* navegador que pede, recebe, exibe objetos *Web*
  - *servidor:* servidor *Web* envia objetos em resposta a pedidos

Mais sobre o protocolo HTTP

Usa serviço de transporte TCP:

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

HTTP é "sem estado"

- servidor não mantém informação sobre pedidos anteriores do cliente

Nota

Protocolos que mantêm "estado" são complexos!

- História passada (estado) tem que ser guardada
- Caso servidor/cliente falhe, suas visões do "estado" podem ser inconsistentes, devem ser reconciliadas

Conexões HTTP

HTTP não persistente

- No máximo um objeto é enviado numa conexão TCP
  - Conexão é fechada após transferência
- Para receber múltiplos objetos é necessário criar múltiplas conexões

HTTP persistente

- Múltiplos objetos podem ser enviados sobre uma única conexão TCP entre cliente e servidor

Exemplo de HTTP não persistente

Supomos que usuário digita a URL www.exemplo.br/diretorio/inicial.html

(contém texto, referências a 10 imagens jpeg)

1a. Cliente HTTP inicia conexão TCP a servidor HTTP (processo) a www.exemplo.br. Porta 80 é padrão para servidor HTTP

1b. servidor HTTP no hospedeiro www.exemplo.br espera por conexão TCP na porta 80. "Aceita" conexão, avisando ao cliente

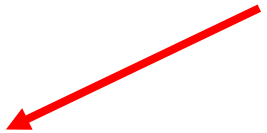
2. cliente HTTP envia *mensagem de pedido* de HTTP (contendo URL) através do *socket* da conexão TCP. A mensagem indica que o cliente deseja receber o objeto diretorio/inicial.html

3. servidor HTTP recebe mensagem de pedido, formula *mensagem de resposta* contendo objeto solicitado e envia a mensagem via *socket*

tempo

Exemplo de HTTP não persistente (cont.)

4. servidor HTTP encerra conexão TCP



5. cliente HTTP recebe mensagem de resposta contendo arquivo HTML, "visualiza" HTML. Analisando arquivo HTML, encontra 10 objetos jpeg referenciados

6. Passos 1 a 5 repetidos para cada um dos 10 objetos jpeg

tempo

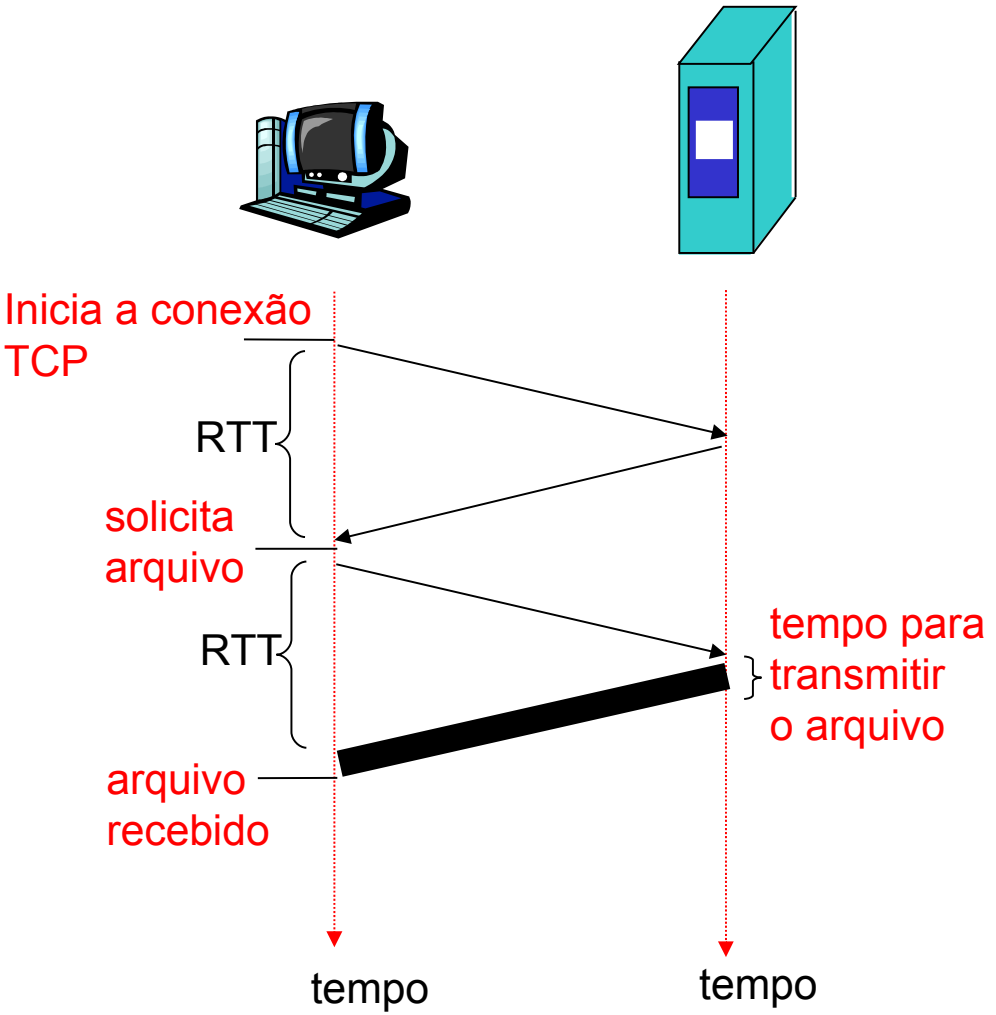
Modelagem do tempo de resposta

Definição de RTT *(Round Trip Time):* intervalo de tempo entre a ida e a volta de um pequeno pacote entre um cliente e um servidor

Tempo de resposta:

- um RTT para iniciar a conexão TCP
- um RTT para o pedido HTTP e o retorno dos primeiros bytes da resposta HTTP
- tempo de transmissão do arquivo

total = 2RTT+tempo de transmissão



HTTP persistente

Problemas com o HTTP não persistente:

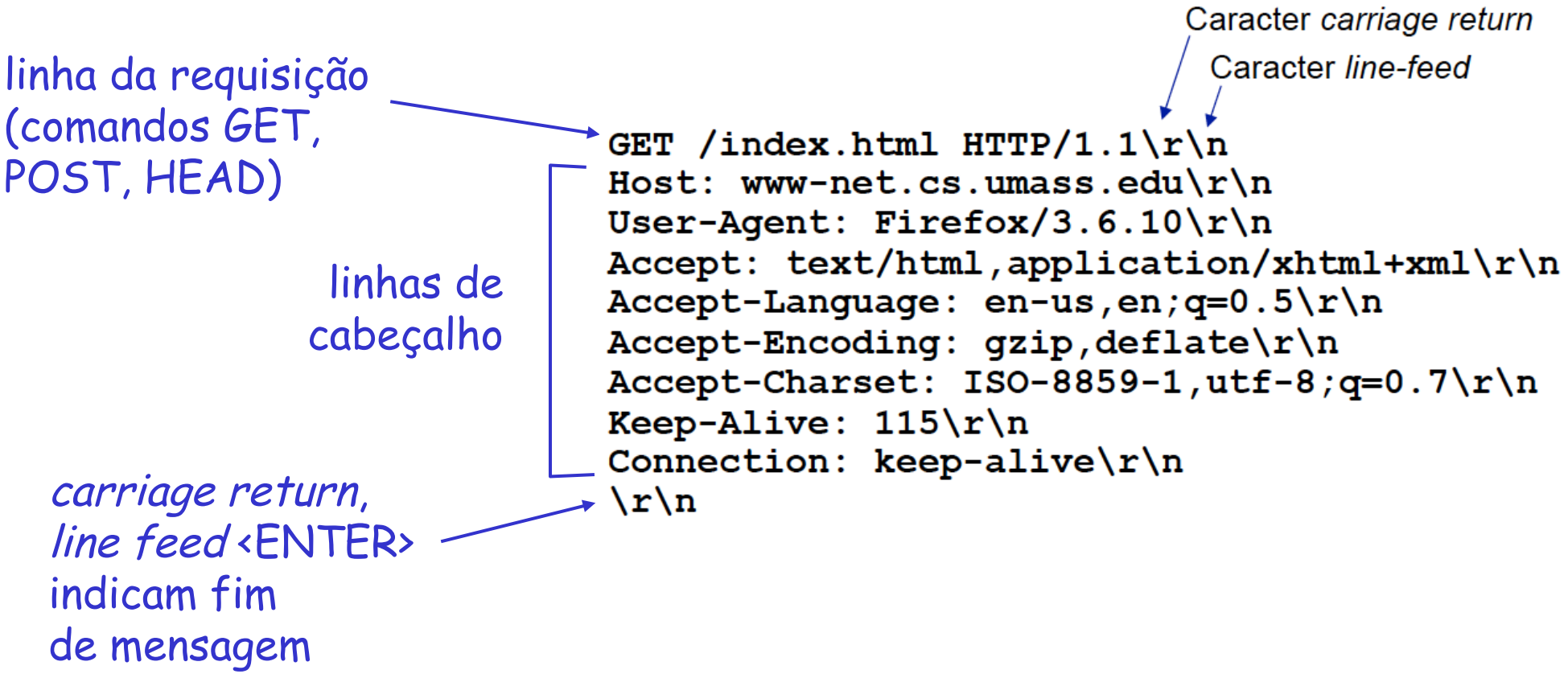
- requer 2 RTTs para cada objeto
- SO aloca recursos do hospedeiro para cada conexão TCP
- os navegadores frequentemente abrem conexões TCP paralelas para recuperar os objetos referenciados

HTTP persistente

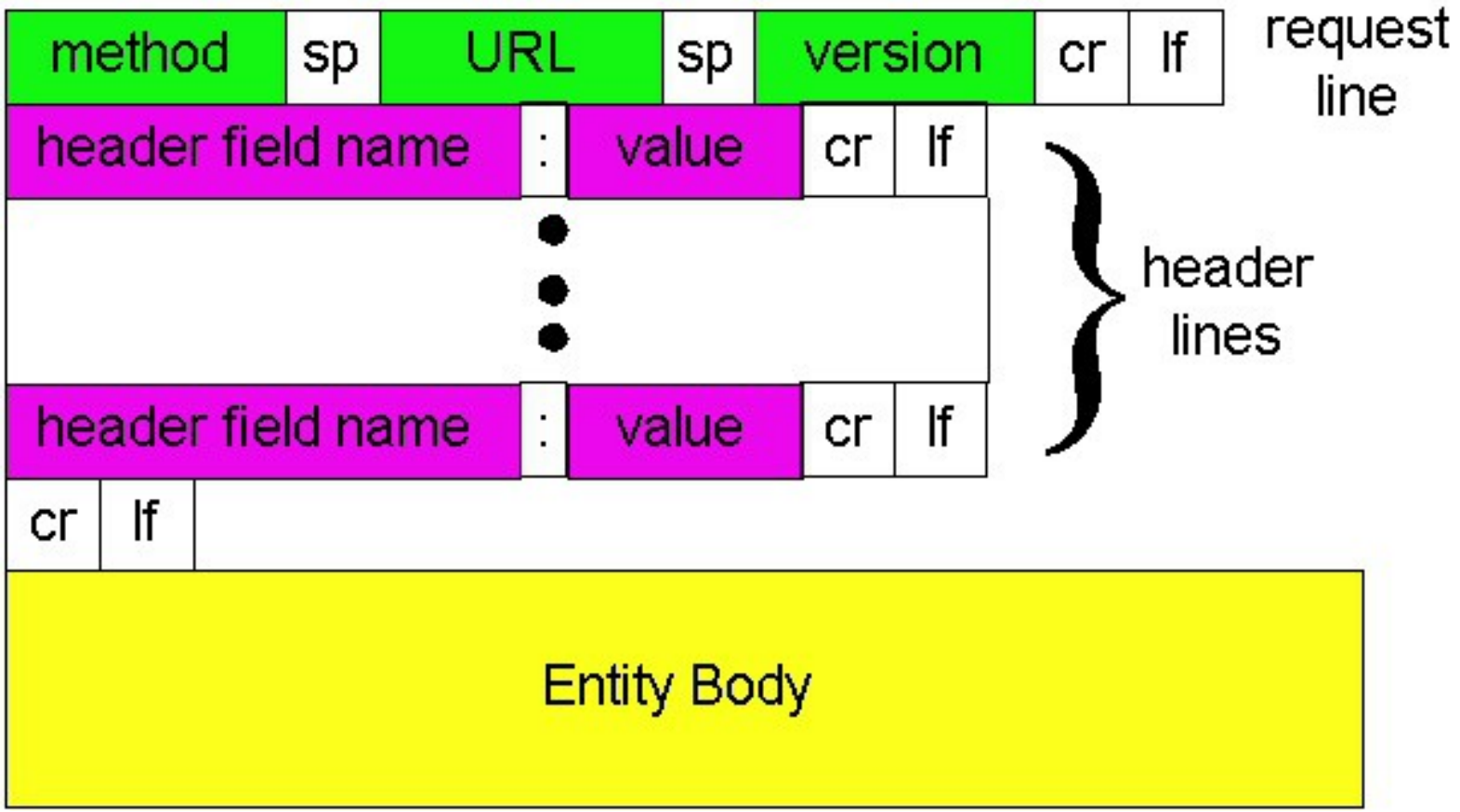
- o servidor deixa a conexão aberta após enviar a resposta
- mensagens HTTP seguintes entre o mesmo cliente/servidor são enviadas nesta conexão
- o cliente envia os pedidos logo que encontra um objeto referenciado
- pode ser necessário apenas um RTT para todos os objetos referenciados

Formato da mensagem HTTP: requisição

- Dois tipos de mensagem HTTP: *requisição, resposta*
- mensagem de requisição HTTP:
  - ASCII (formato legível por pessoas)



mensagem de requisição HTTP: formato geral



Enviando conteúdo de formulário

Método POST :

- Páginas *Web* frequentemente contêm formulário de entrada
- Conteúdo é enviado para o servidor no corpo da mensagem

GET + URL com campos:

- Usa o método GET
- Conteúdo é enviado para o servidor no campo URL. Exemplo:

```
www.exemplo.com.br/busca?chave=teste&max=10
```

Tipos de métodos

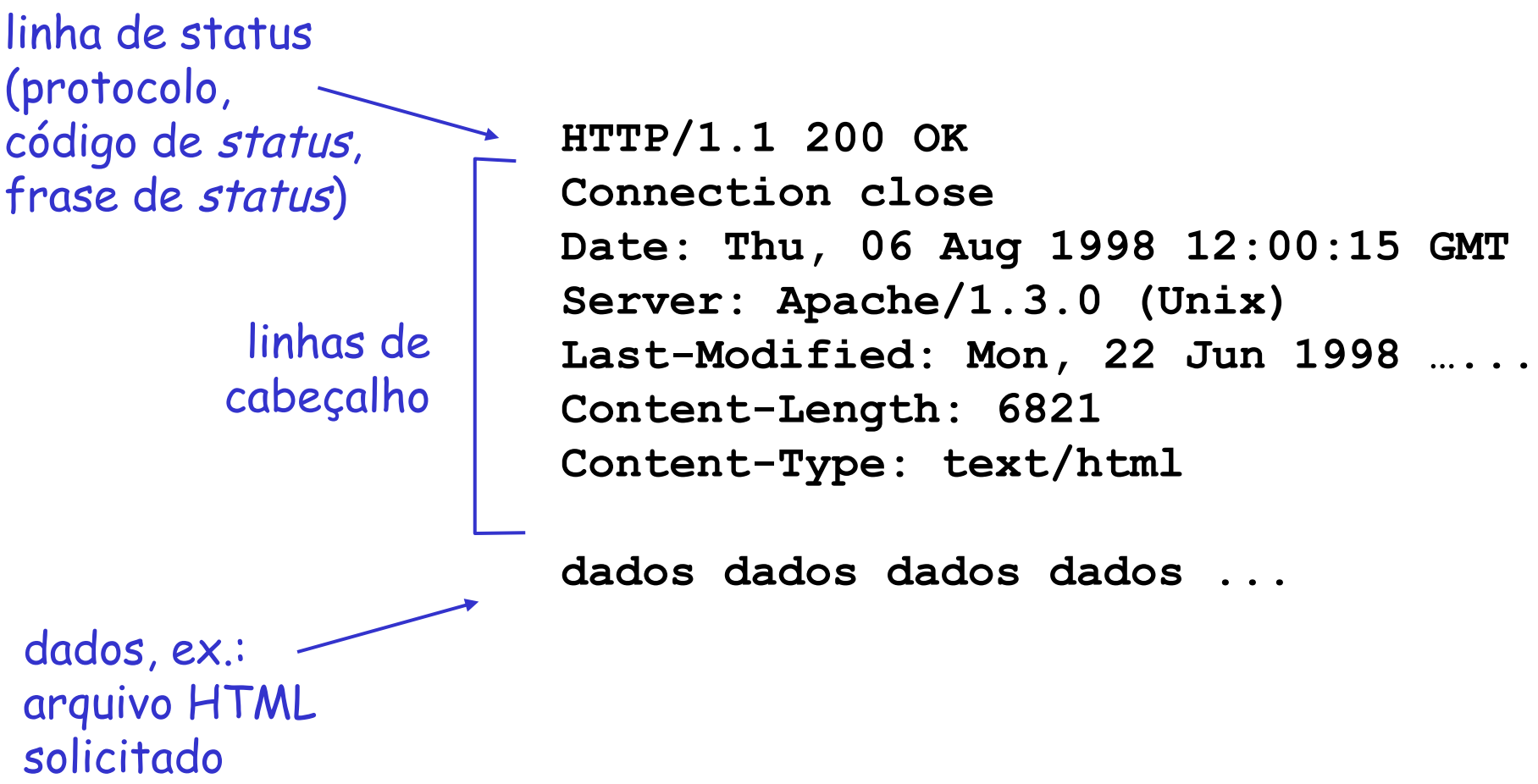
HTTP/1.0

- GET
- POST
- HEAD
  - Pede para o servidor não enviar o objeto requerido junto com a resposta

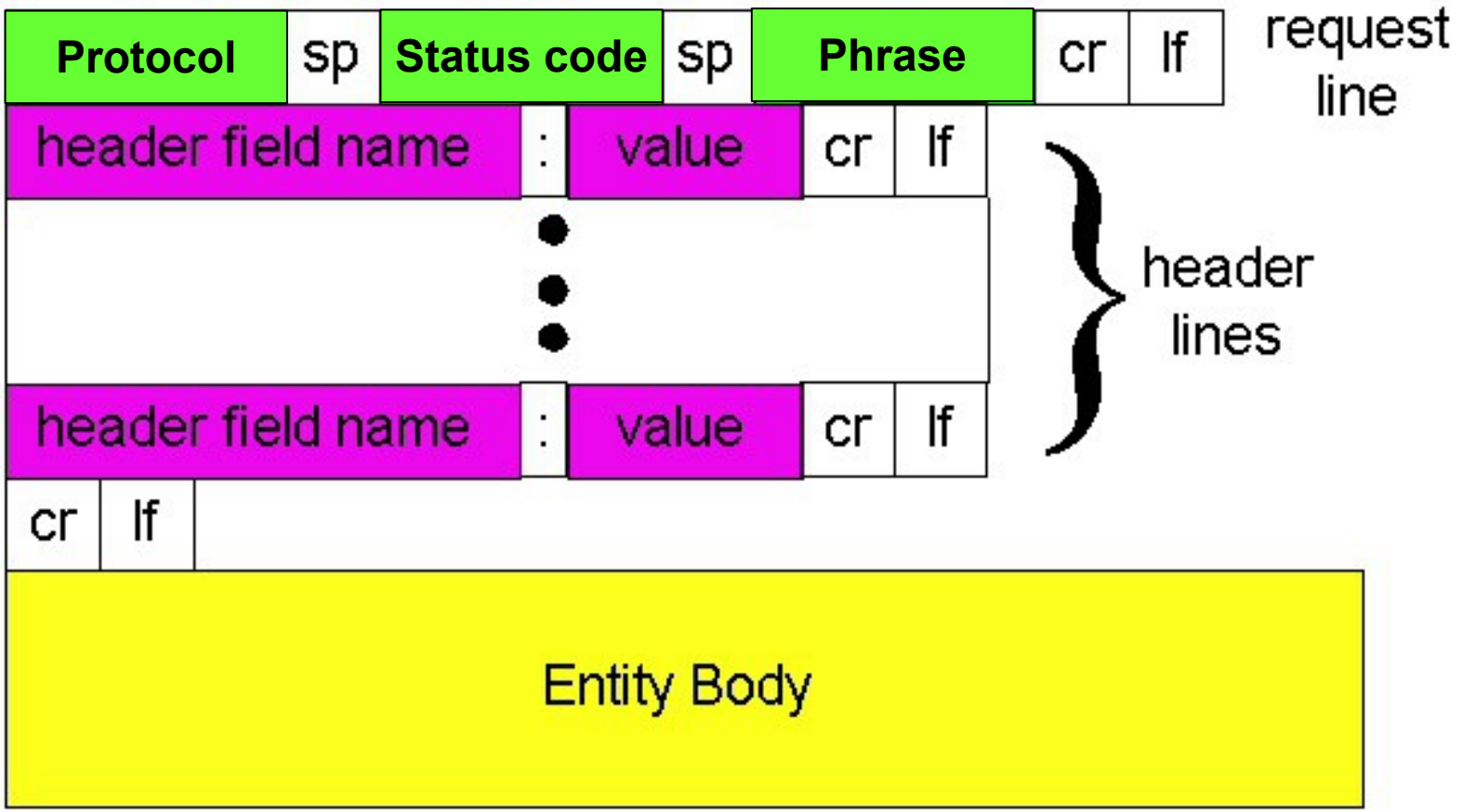
HTTP/1.1

- GET, POST, HEAD
- PUT
  - *Upload* de arquivo contido no corpo da mensagem para o caminho especificado no campo URL
- DELETE
  - Exclui arquivo especificado no campo URL

Formato de mensagem HTTP: resposta



mensagem de resposta HTTP: formato geral



Códigos de *status* da resposta HTTP

Na primeira linha da mensagem de resposta servidor→cliente. Alguns códigos típicos:

200 OK

- sucesso, objeto pedido segue mais adiante nesta mensagem

301 Moved Permanently

- objeto pedido mudou de lugar, nova localização especificada mais adiante nesta mensagem (*Location*:)

400 Bad Request

- mensagem de pedido não entendida pelo servidor

404 Not Found

- documento pedido não se encontra neste servidor

505 HTTP Version Not Supported

- versão de HTTP do pedido não usada por este servidor

Experimente com HTTP (do lado cliente)

1. Use cliente (nc ou telnet) para seu servidor WWW:

```
nc www.inf.ufg.br 80
```

Abre conexão TCP para a porta 80 (porta padrão do servidor HTTP) a www.inf.ufg.br. Qualquer coisa digitada é enviada para a porta 80 do www.inf.ufg.br

2. Digite um pedido GET HTTP:

```
GET /~kleber/exemplo.html HTTP/1.0
```

Digitando isto (deve teclar <ENTER> duas vezes), está enviando este pedido GET mínimo (porém completo) ao servidor HTTP

3. Examine a mensagem de resposta enviada pelo servidor HTTP

Cookies: manutenção do "estado" da conexão

Muitos *sites Web* usam *cookies*

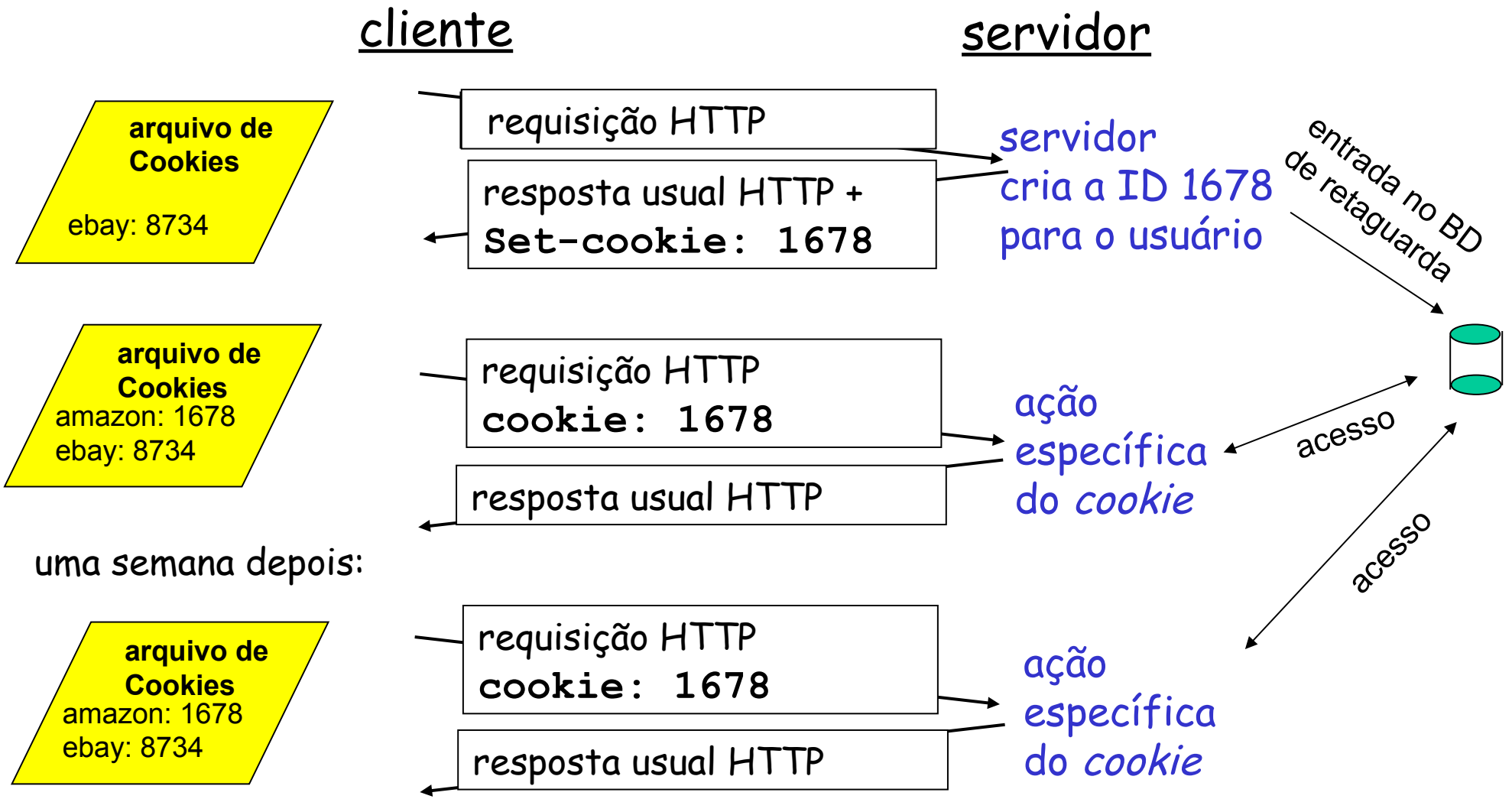
Quatro componentes:

1) linha de cabeçalho do *cookie* na mensagem de resposta HTTP
2) linha de cabeçalho do *cookie* na mensagem de pedido HTTP
3) arquivo do *cookie* mantido no *host* do usuário e gerenciado pelo navegador do usuário
4) BD de retaguarda no *site Web*

Exemplo:

- Usuário acessa a Internet sempre do mesmo PC
- Ela visita um *site* específico de comércio eletrônico pela primeira vez
- Quando os pedidos iniciais HTTP chegam no *site*, são gerados
  - um ID único
  - uma entrada para o ID no BD de retaguarda

Cookies: manutenção do "estado" (cont.)



Cookies (continuação)

O que os *cookies* podem obter:

- autorização
- carrinhos de compra
- recomendações
- estado da sessão do usuário (*Webmail*)

nota

Cookies e privacidade:

- *cookies* permitem que os *sites* aprendam muito sobre os usuários
- usuário pode fornecer nome e *e-mail* para os *sites*

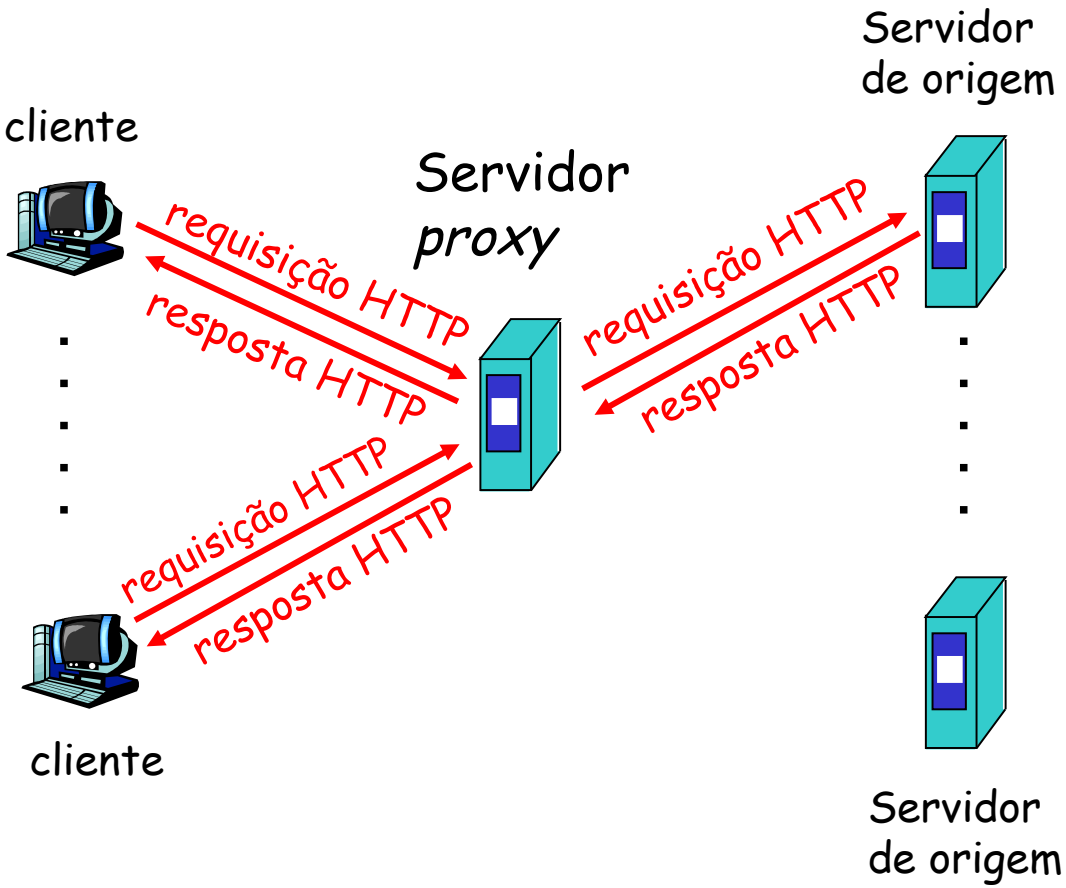
Como manter o "estado":

- Pontos finais do protocolo: mantêm o estado no transmissor/receptor para múltiplas transações
- *Cookies:* mensagens HTTP transportam o estado

Cache Web (servidor *proxy*)

Meta: atender pedido do cliente sem envolver servidor de origem

- Usuário configura navegador: acessos *Web* via *proxy*
  - Também é possível usar *proxy* transparente
- Cliente envia todos pedidos HTTP ao *proxy*
  - Se objeto estiver no *cache* do *proxy*, este o devolve imediatamente na resposta HTTP
  - Senão, solicita objeto do servidor de origem, depois devolve resposta HTTP ao cliente



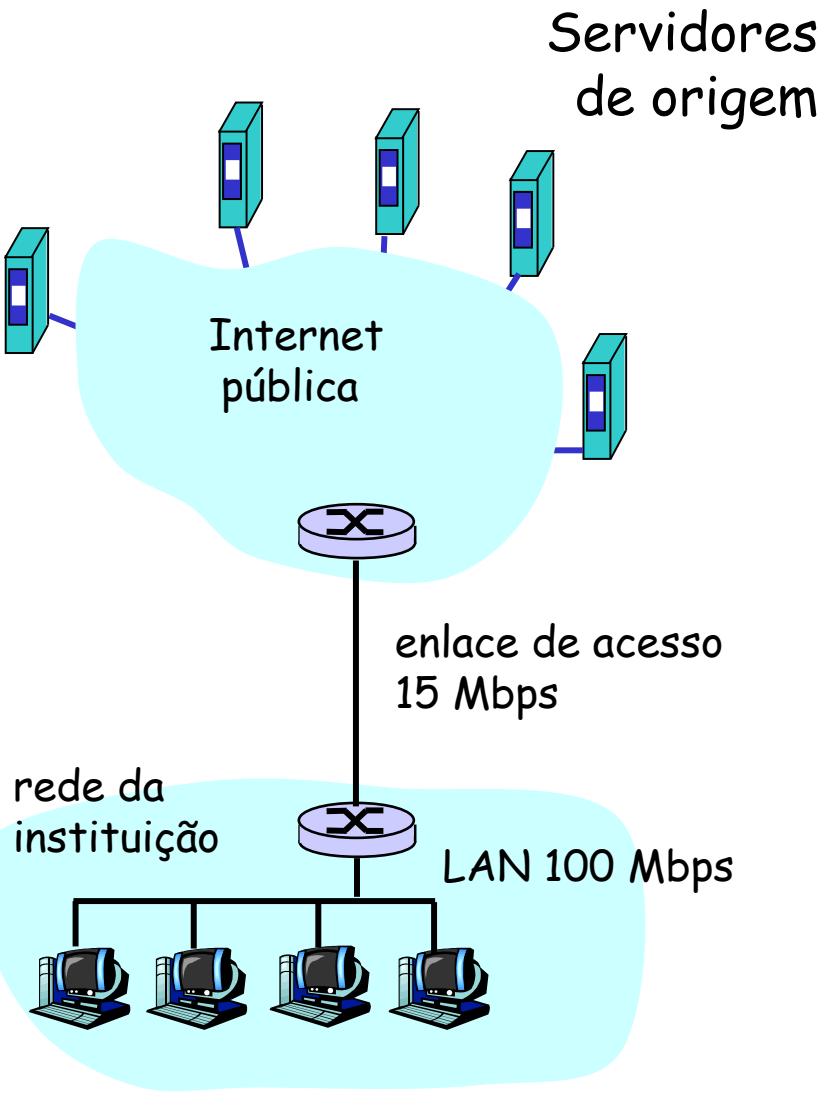
Mais sobre *Caches Web*

- *Cache* atua tanto como cliente quanto como servidor
- Tipicamente o *cache* é instalado por um ISP (universidade, empresa, ISP residencial)

Por que fazer cache Web?

- Redução do tempo de resposta para os pedidos do cliente
- Redução do tráfego no canal de acesso de uma instituição
- A Internet cheia de *caches* permitem que provedores de conteúdo com poucos recursos efetivamente forneçam conteúdo

Exemplo de *cache* (1)



Hipóteses

- Tamanho médio de um objeto = 1Mb
- Taxa média de solicitações dos navegadores de uma instituição para os servidores originais = 15 obj./seg
- Taxa média solicitada = 15 Mbps
- Atraso do roteador institucional para qualquer servidor origem e de volta ao roteador = 500mseg

Consequências

- Utilização da LAN = 15%
- Utilização do canal de acesso = 100% □ fila cresce até transbordar
- Atraso total médio = "atraso da Internet" + "atraso de acesso" + atraso na LAN = 500mseg + minutos + 10's milissegundos

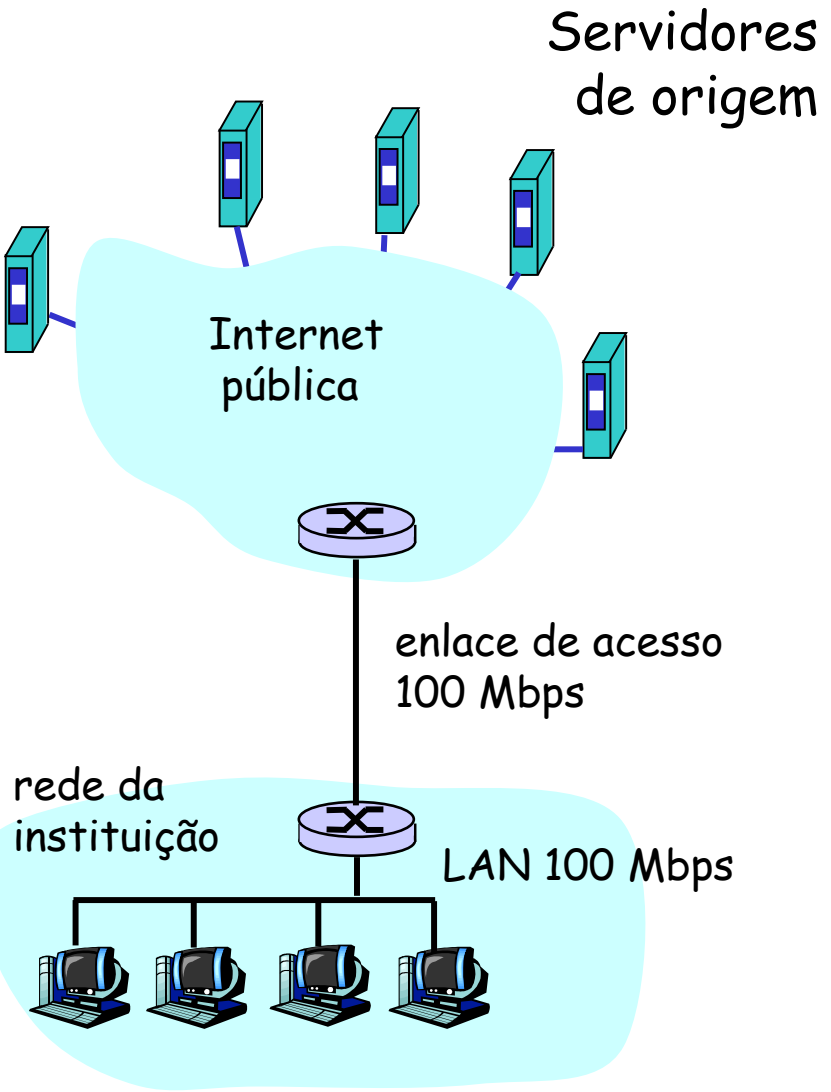
Exemplo de *cache* (2)

Solução em potencial

- Aumentar a largura de banda do canal de acesso para, por exemplo, 100 Mbps

Consequências

- Utilização da LAN = 15%
- Utilização do canal de acesso = 15%
- Atraso total médio = "atraso da Internet" + "atraso de acesso" + atraso na LAN = 500mseg + 10's msegs + 10's msegs
- Frequentemente, essa é uma ampliação cara



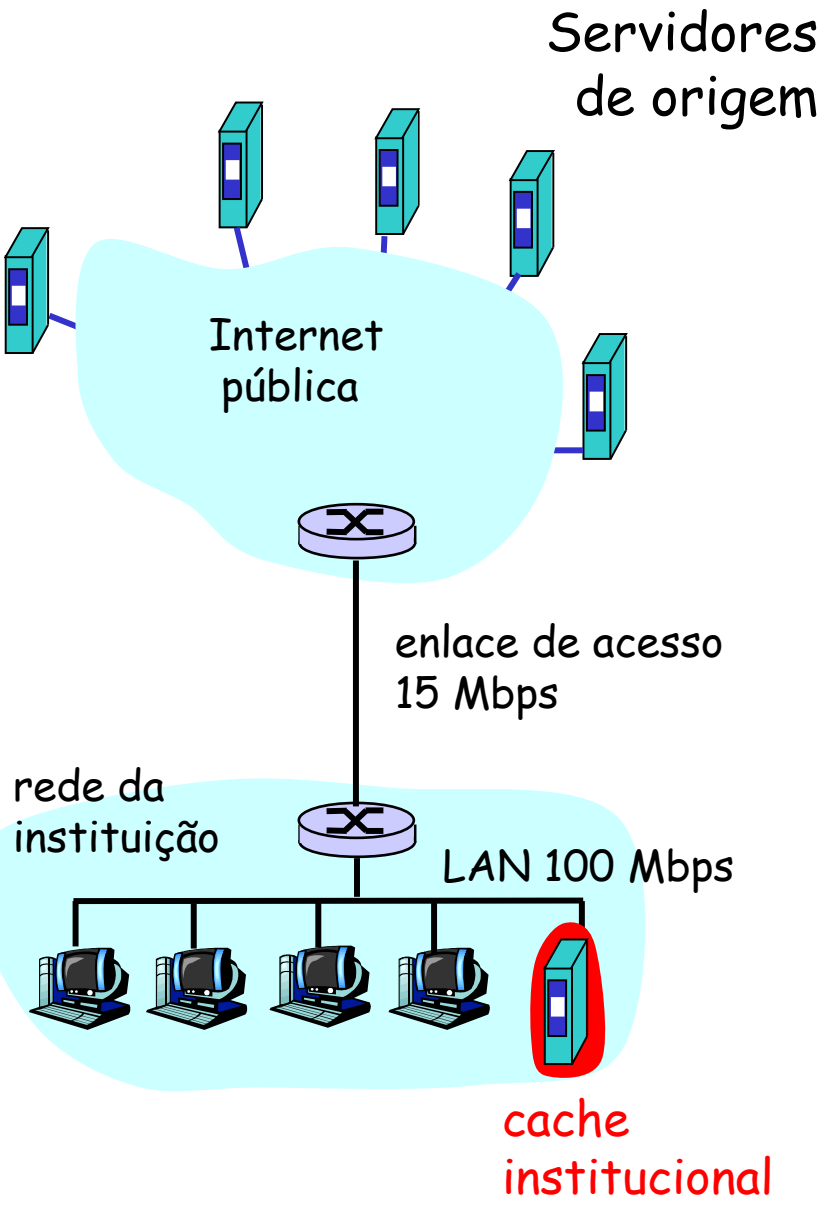
Exemplo de *cache* (3)

Instale uma cache

- Assuma que a taxa de acerto seja de 0,4

Consequências

- 40% dos pedidos serão atendidos quase que imediatamente
- 60% dos pedidos serão servidos pelos servidores de origem
- Utilização do canal de acesso é reduzido para 60%, resultando em atrasos desprezíveis (e.g., 10's mseg)
- Atraso total médio = "atraso da Internet" + "atraso de acesso" + atraso na LAN = 0,6*(500mseg + 10's mseg) + 10's msegs < 500mseg

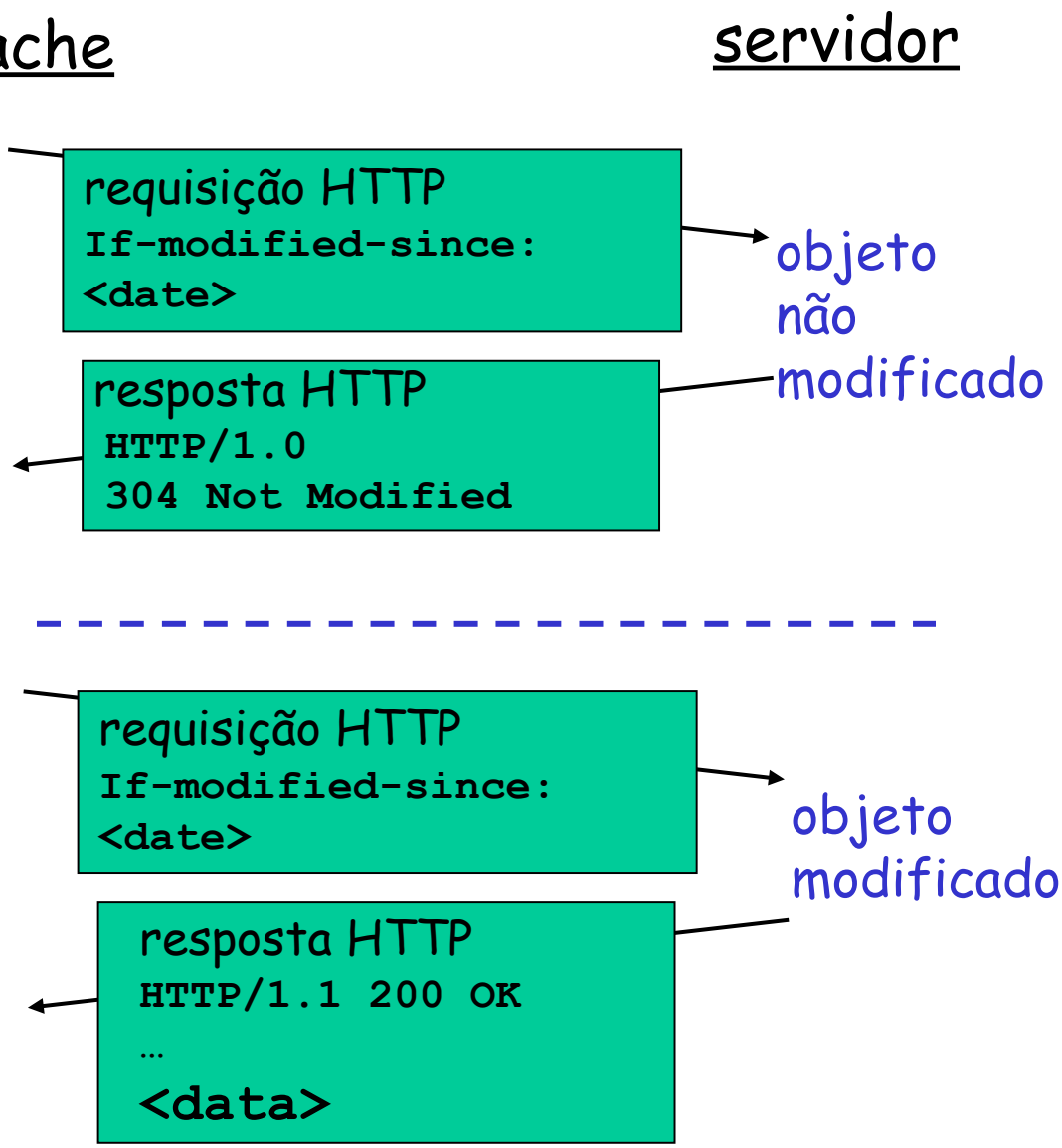


GET condicional

- Meta: não enviar objeto se cliente já tem (na *cache*) versão atual
- Cliente: especifica data da cópia na *cache* no pedido HTTP

  `If-modified-since: <date>`
- Servidor: envia resposta sem o objeto se cópia na *cache* é atual

  `HTTP/1.0 304 Not Modified`



Exercícios

1) Falso ou verdadeiro?

a) Um usuário requisita uma página *Web* que consiste em algum texto e três imagens. Para essa página, o cliente enviará uma mensagem de requisição e receberá quatro mensagens de resposta.

b) Duas páginas *Web* distintas (por exemplo, www.mit.edu/research.html e www.mit.edu/students.html) podem ser enviadas pela mesma conexão persistente.

c) Com conexões não persistentes entre navegador e servidor de origem, é possível que um único segmento TCP transporte duas mensagens distintas de requisição HTTP.

d) As mensagens de resposta HTTP nunca possuem um corpo de mensagem vazio.

Exercícios

2) Considere a seguinte cadeia de caracteres ASCII capturada por um *sniffer* de rede quando o navegador enviou uma mensagem HTTP GET (ou seja, o conteúdo real de uma mensagem HTTP GET). Responda às seguintes questões, indicando onde está a resposta na mensagem HTTP GET a seguir.

a) Qual é a URL do documento requisitado pelo navegador?

b) Qual versão do HTTP o navegador está rodando?

c) O navegador requisita uma conexão não persistente ou persistente?

d) Qual é o endereço IP do hospedeiro no qual o navegador está rodando?

e) Que tipo de navegador inicia essa mensagem? Por que é necessário o tipo de navegador em uma mensagem de requisição HTTP?

```
GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai
a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0 (
Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gec
ko/20040804 Netscape/7.2 (ax) <cr><lf>Accept:ex
t/xml, application/xml, application/xhtml+xml, text
/html;q=0.9, text/plain;q=0.8,image/png,*/*;q=0.5
<cr><lf>Accept-Language: en-us,en;q=0.5<cr><lf>Accept-
Encoding: zip,deflate<cr><lf>Accept-Charset: ISO
-8859-1,utf-8;q=0.7,*;q=0.7<cr><lf>Keep-Alive: 300<cr>
<lf>Connection:keep-alive<cr><lf><cr><lf>
```