



EACH

Escola de Artes, Ciências e Humanidades
da Universidade de São Paulo

Redes de Computadores

Capítulo 2 - Camada de Aplicação

Profa. Cíntia B. Margi
Agosto/2009

Cap. 2: Camada de Aplicação

- 2.1. Princípios de aplicações de rede
- 2.2. Web e HTTP
- 2.3. FTP
- 2.4. Correio eletrônico
 - SMTP, POP3, IMAP
- 2.5. DNS
- 2.6. Compartilhamento de arquivos P2P
- 2.7. Programação de socket com TCP
- 2.8. Programação de socket com UDP
- 2.9. Construindo um servidor Web



EACH

Escola de Artes, Ciências e Humanidades
da Universidade de São Paulo

Exemplos de aplicações de rede



Exemplos de aplicações de rede

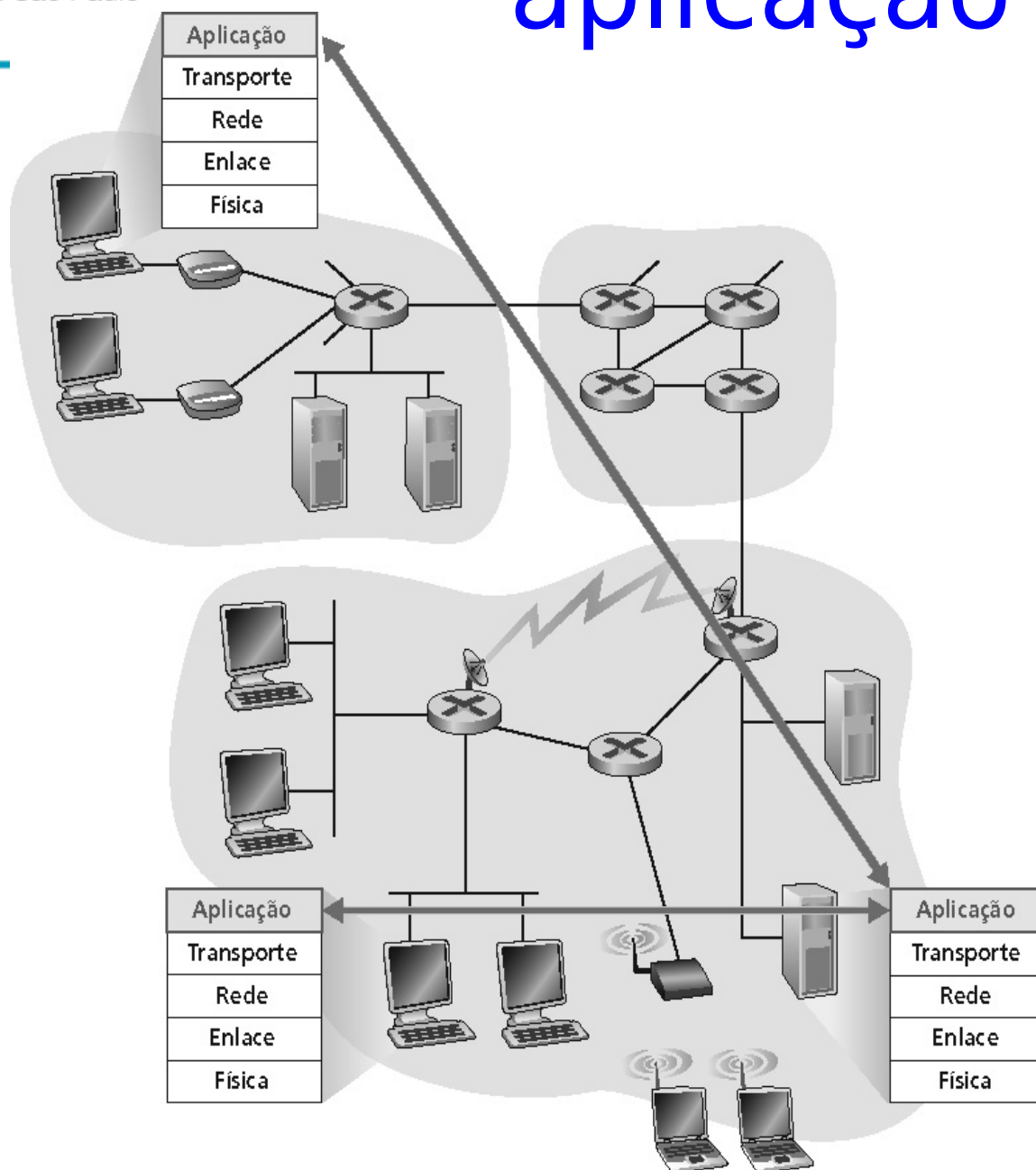
- E-mail
- Web
- Mensagem instantânea
- Login remoto (telnet e ssh)
- Compartilhamento de arquivos P2P
- Jogos de rede multiusuário
- Streaming de vídeos armazenados
- Telefonia via Internet
- Videoconferência em tempo real
- Computação paralela massiva



EACH

Escola de Artes, Ciências e Humanidades
da Universidade de São Paulo

Criando uma nova aplicação





Criando uma nova aplicação

Escrever programas que:

- executem sobre diferentes sistemas finais;
- se comuniquem através de uma rede.
 - Ex.: Web – software de servidor Web se comunicando com software do browser.
- Nenhum software é escrito para dispositivos no núcleo da rede!
 - não há camada de aplicação nos elementos de rede!



Arquiteturas de Aplicação

- Cliente-servidor
- **Peer-to-peer** (P2P)
- Híbrida de cliente-servidor e P2P
- Arquitetura da aplicação é diferente de arquitetura de rede!



Arquiteturas de Aplicação

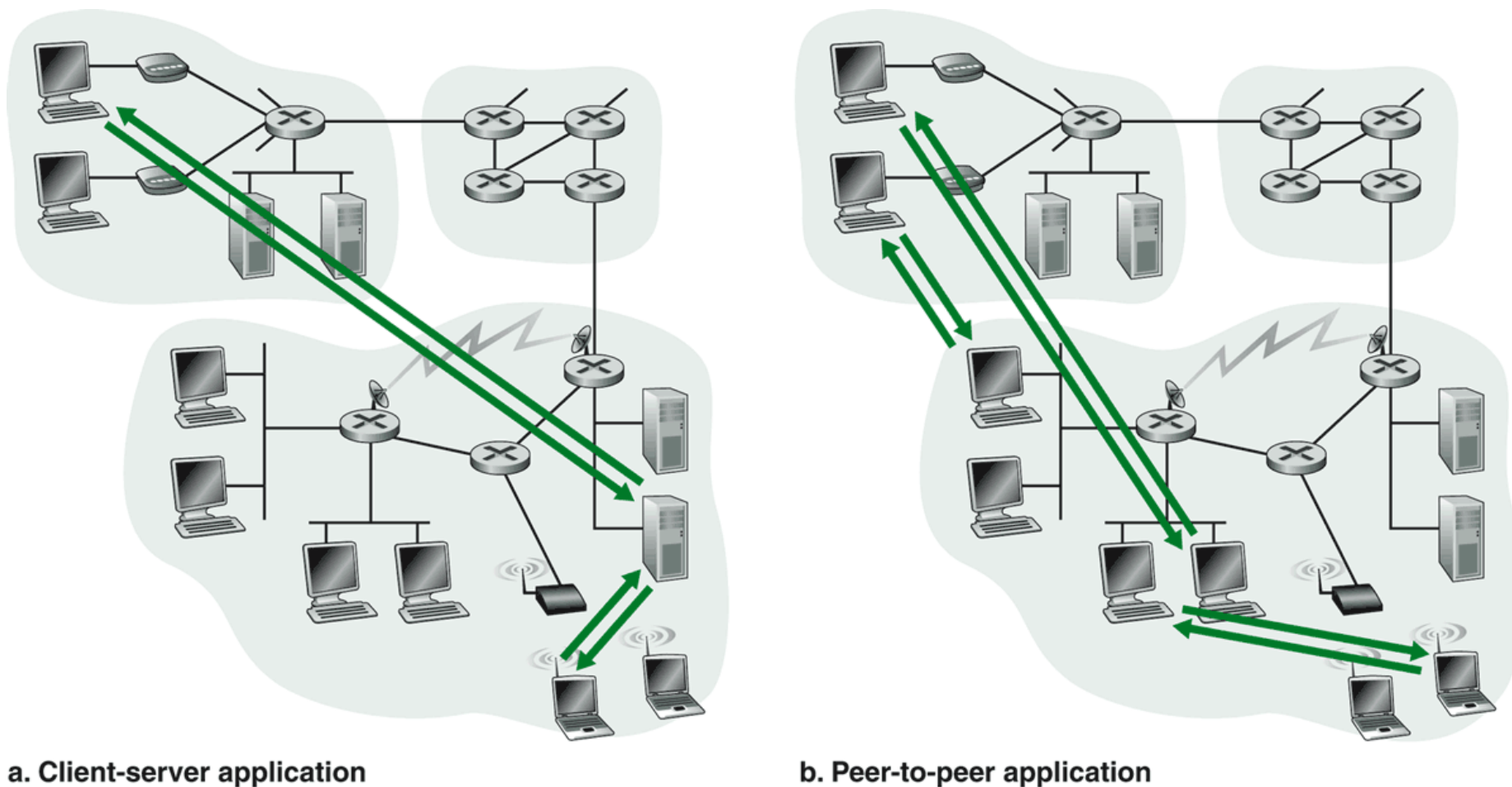


Figure 2.2 ♦ (a) Client-server architecture; (b) P2P architecture.



Arquitetura Cliente-Servidor

Servidor:

- computador hospedeiro sempre ativo;
- endereço IP permanente;
- fornece serviços solicitados pelo cliente.

Clientes:

- comunicam-se com o servidor;
- podem ser conectados intermitentemente;
- podem ter endereço IP dinâmico;
- não se comunicam diretamente uns com os outros.



Arquitetura Cliente-Servidor (cont.)

- O que acontece se o servidor recebe muitas requisições?
 - escalabilidade?
 - servidores virtuais -> *server farms*.
 - Akamai.



Arquitetura P2P (pura)

- P2P: peer-to-peer, ou par-a-par.
- Sistemas finais arbitrários comunicam-se diretamente.
- Não há servidor.
- Pares são intermitentemente conectados e trocam endereços IP.
- Ex.: Gnutella.
- Altamente escalável, mas difícil de gerenciar.



Arquiteturas Híbridas: Napster

- Transferência de arquivo P2P.
- Busca centralizada de arquivos:
 - conteúdo de registro dos pares no servidor central;
 - consulta de pares no mesmo servidor central para localizar o conteúdo.



Instant messaging

- Bate-papo entre dois usuários é P2P.
- Detecção/localização centralizada de presença:
 - usuário registra seu endereço IP com o servidor central quando fica on-line;
 - usuário contata o servidor central para encontrar endereços IP dos vizinhos.



Comunicação entre processos

- Dentro do mesmo hospedeiro:
 - se comunicam usando comunicação interprocesso (definido pelo SO).
- Em diferentes hospedeiros:
 - se comunicam por meio de troca de mensagens.



Processos clientes e servidores

- Processo cliente: processo que inicia a comunicação.
- Processo servidor: processo que espera para ser contatado.
- E no caso de P2P?
 - aquele que inicia a sessão é o cliente.



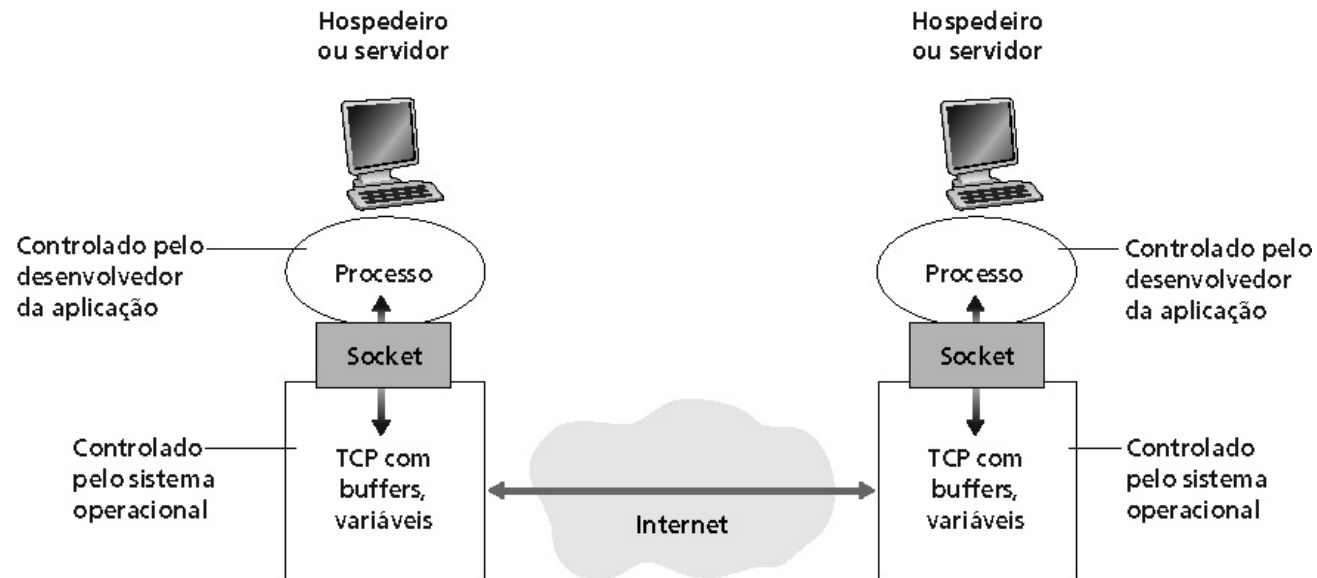
Sockets

- Analogia:
 - processo = casa;
 - socket = porta.
- Para enviar mensagem:
 - o processo empurra a mensagem para fora da porta.
 - confia na infra-estrutura de transporte no outro lado da porta.



Sockets (cont.)

- Infra-estrutura de transporte: TCP e UDP.
- API:
 - escolha do protocolo (TCP e UDP);
 - parâmetros de configuração.





Endereçamento de Processos

- Para um processo receber mensagens, ele deve ter um identificador!
- endereço IP de 32 bits (único). Mas é suficiente?
 - Não, muitos processos podem estar em execução no mesmo hospedeiro!



Endereçamento de Processos (cont.)

- O identificador inclui o endereço IP e o número da porta associada ao processo no hospedeiro!
- Conceito de multiplexação – camada de transporte do Modelo OSI.
- Exemplos de números de porta:
 - Servidor HTTP: 80
 - Servidor de Correio (SMTP): 25
 - Alocação de portas: <http://www.iana.org>



Protocolos de Camada de Aplicação

- Definem:
 - tipo das mensagens trocadas.
 - Ex.: mensagens de requisição e resposta.
 - sintaxe dos tipos de mensagem:
 - os campos nas mensagens e como são delineados.
 - semântica dos campos:
 - significado da informação nos campos.
 - regras para quando e como os processos enviam e respondem às mensagens.



Protocolos de Camada de Aplicação (cont.)

- Dois tipos de protocolos:
 - Domínio público: definidos nas RFCs.
 - recomendados para interoperabilidade.
 - Ex.: HTTP, SMTP
 - Proprietários:
 - Ex.: KaZaA
- O protocolo é parte da aplicação!!



Quais serviços uma aplicação necessita?

- Transferência confiável de dados:
 - aplicação tolerara perda?
 - sim: áudio.
 - não: transferência de arquivos, telnet.
- Temporização:
 - aplicações exigem baixos atrasos?
 - ex.: telefonia via Internet, jogos interativos.



Quais serviços uma aplicação necessita?

- Largura de banda:
 - aplicações exigem banda mínima?
 - ex.: multimídia.
 - “aplicações elásticas” melhoram quando a banda disponível aumenta (transferência de arquivos).



Requisitos de Aplicações de Rede

Aplicação	Perda de Dados	Largura de banda	Sensibilidade ao atraso
transferência de arquivos	sem perda	elástica	não
e-mail	sem perda	elástica	não
documentos web	sem perda	elástica (alguns kbps)	não
áudio/vídeo em tempo real	tolerante à perda	áudio: kbps – 1Mbps vídeo: 10Kbps – 5Mbps	sim: décimos de segundos
áudio/vídeo armazenado	tolerante à perda	áudio: kbps – 1Mbps vídeo: 10Kbps – 5Mbps	sim: alguns segundos
jogos interativos	tolerante à perda	alguns Kbps – 10Mbps	sim: décimos de segundos
mensagem instantânea	sem perda	elástica	sim e não



Aplicações e seus protocolos

Aplicação	Protocolo de Camada de Aplicação	Protocolo de transporte
transferência de arquivos	FTP (RFC959)	TCP
e-mail	SMTP (RFC2821)	TCP
web	HTTP (RFC2616)	TCP
Multimídia em tempo real	RTP ou proprietário	UDP



EACH

Escola de Artes, Ciências e Humanidades
da Universidade de São Paulo

Web e HTTP



Terminologia:

- Página Web consiste de objetos
- Objeto pode ser arquivo HTML, imagem JPEG, Java applet, arquivo de áudio,...
- A página Web consiste de arquivo-HTML base, que inclui vários objetos referenciados
- Cada objeto é endereçado por uma URL
- Exemplo de URL:

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

Nome do hospedeiro

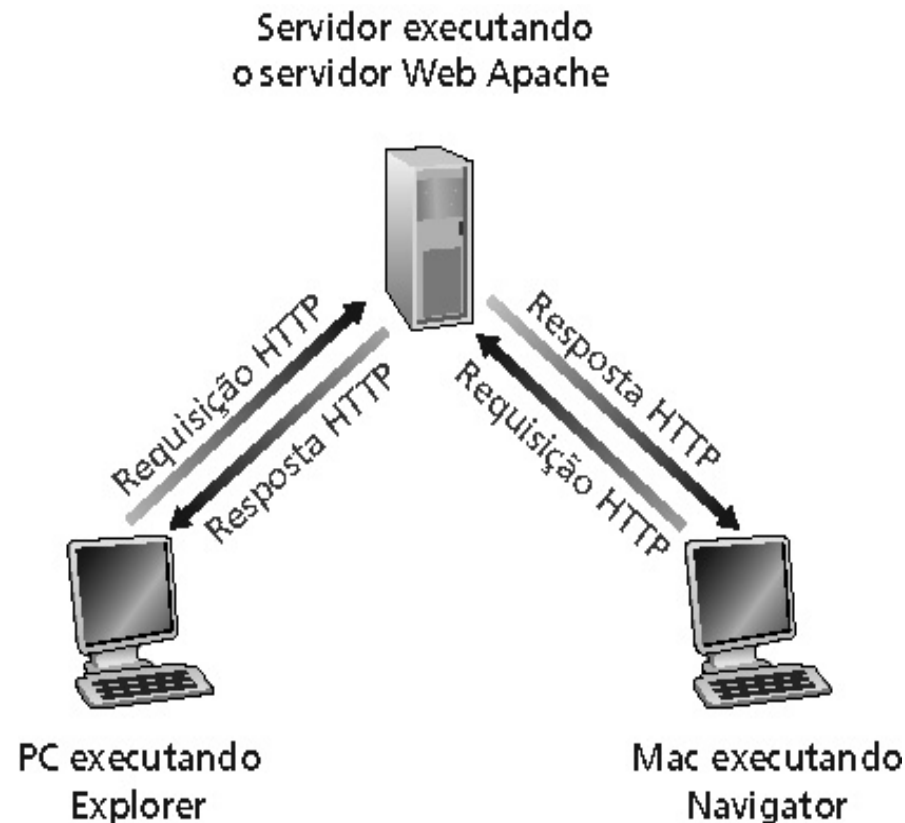
Nome do caminho



HTTP - Visão Geral

HTTP: hypertext transfer protocol

- Protocolo da camada de aplicação da Web.





HTTP - Visão Geral

Modelo cliente/servidor:

- Cliente: browser que solicita, recebe e apresenta objetos da Web;
 - Servidor: envia objetos em resposta a pedidos.
- Padrões:
 - HTTP 1.0: RFC 1945
 - HTTP 1.1: RFC 2616



HTTP - Funcionamento

- Utiliza TCP.
- Sequência de ações:
 - cliente inicia conexão TCP (cria socket) para o servidor na porta 80.
 - servidor aceita uma conexão TCP do cliente.
 - mensagens HTTP são trocadas entre o browser (cliente HTTP) e o servidor Web (servidor HTTP).
 - conexão TCP é fechada.



HTTP - Visão Geral

- HTTP é “stateless”
 - O servidor não mantém informação sobre os pedidos passados pelos clientes.
- **Protocolos que mantêm informações de “estado” são complexos!**
 - Histórico do passado (estado) deve ser mantido.
 - Se o servidor/cliente quebra, suas visões de “estado” podem ser inconsistentes, devendo ser reconciliadas.



Tipos de conexões HTTP

HTTP não persistente:

- no máximo, um objeto é enviado sobre uma conexão TCP.
- HTTP/1.0 utiliza HTTP não persistente.

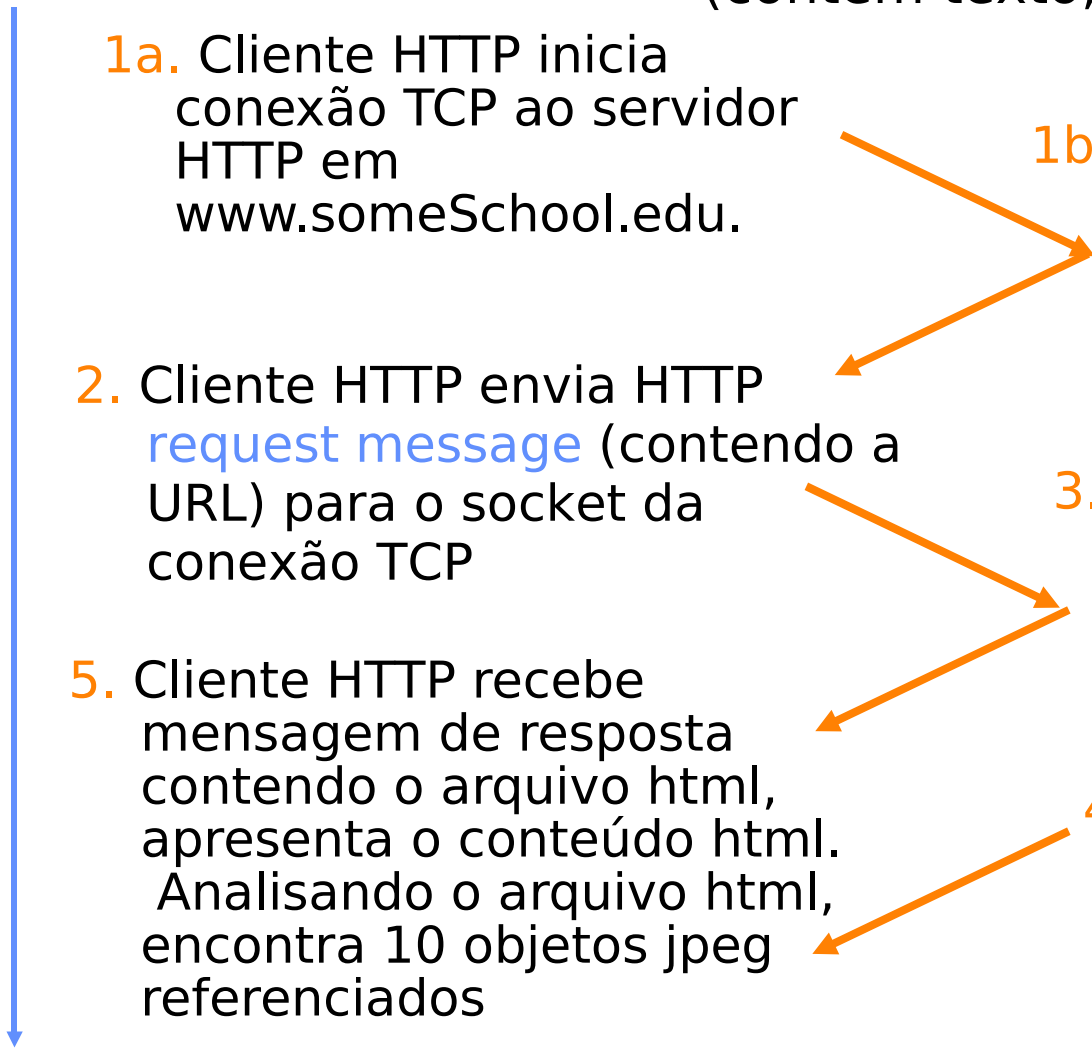
HTTP persistente:

- múltiplos objetos podem ser enviados sobre uma conexão;
- TCP entre o cliente e o servidor;
- HTTP/1.1 utiliza conexões persistentes em seu modo padrão.

HTTP não-persistente

URL: www.someSchool.edu/someDepartment/home.index

(contém texto, referências a 10 imagens jpeg)

- 
- 1a. Cliente HTTP inicia conexão TCP ao servidor HTTP em `www.someSchool.edu`.
 - 1b. Servidor HTTP no computador `www.someSchool.edu` esperando pela conexão TCP na porta 80. "Aceita" conexão, notificando o cliente.
 2. Cliente HTTP envia HTTP **request message** (contendo a URL) para o socket da conexão TCP
 3. Servidor HTTP recebe mensagem de pedido, forma **response message** contendo o objeto solicitado, envia mensagem para o socket
 4. Servidor HTTP fecha conexão TCP
 5. Cliente HTTP recebe mensagem de resposta contendo o arquivo html, apresenta o conteúdo html. Analisando o arquivo html, encontra 10 objetos jpeg referenciados
 6. Passos 1-5 são repetidos para cada um dos 10 objetos jpeg

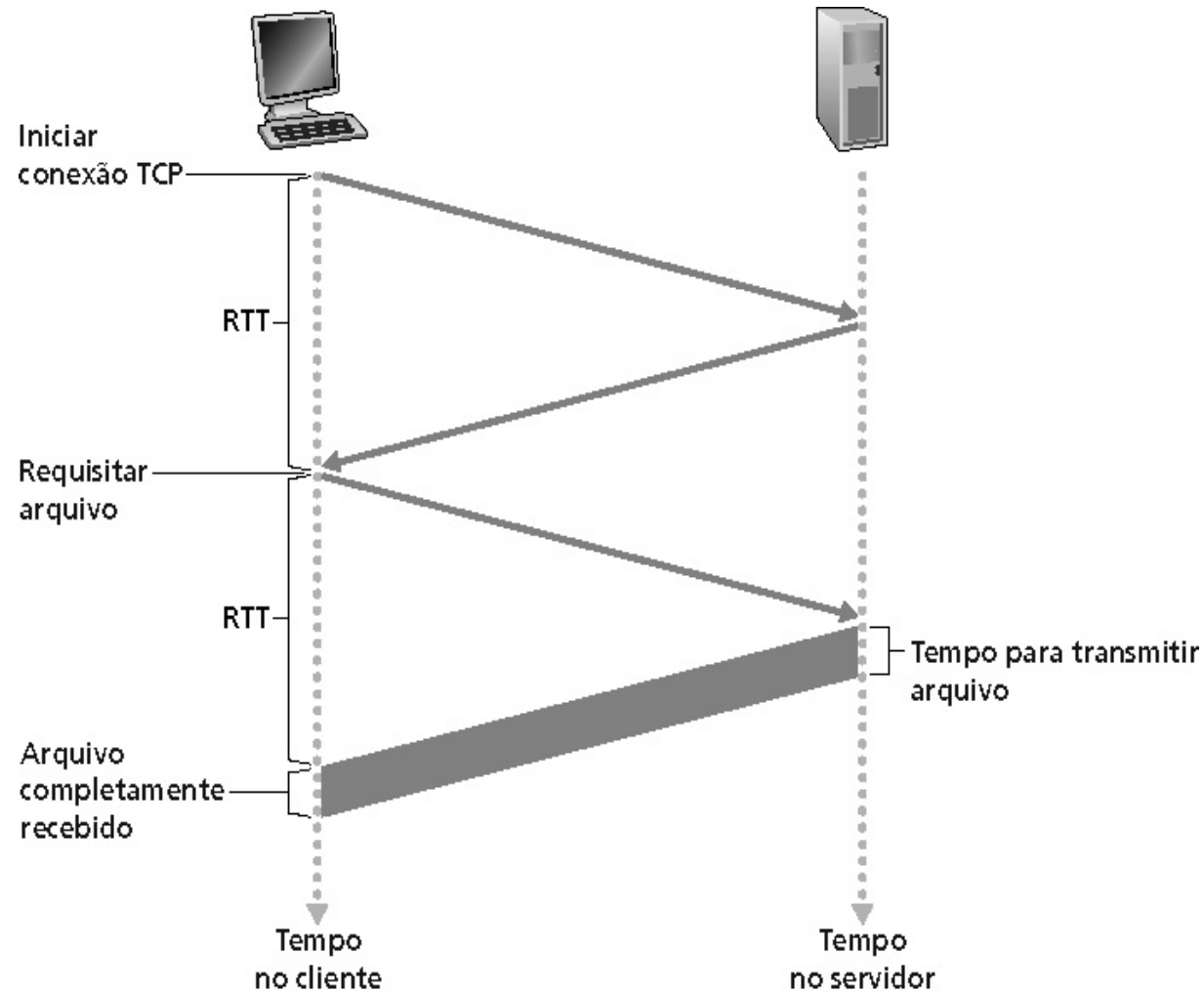
Tempo de resposta

Definição de RTT:

tempo para enviar um pequeno pacote que vai do cliente para o servidor e retorna.

Tempo de resposta:

- Um RTT para iniciar a conexão TCP
- Um RTT para requisição HTTP e primeiros bytes da resposta HTTP para retorno.
- Tempo de transmissão de arquivo.



Total = $2RTT$ + tempo de transmissão



HTTP persistente

- Servidor deixa a conexão aberta após enviar uma resposta.
- Mensagens HTTP subseqüentes entre o mesmo cliente/servidor são enviadas pela conexão.
- Dois modos de operação:
 - sem paralelismo;
 - com paralelismo.



HTTP persistente

- **Persistente sem paralelismo:**
 - cliente emite novas requisições apenas quando a resposta anterior for recebida;
 - tempo de resposta: um RTT para cada objeto referenciado.
- **Persistente com paralelismo:**
 - padrão no HTTP/1.1;
 - cliente envia requisições assim que encontra um objeto referenciado;
 - tempo de resposta: tão pequeno como um RTT para todos os objetos referenciados.



Mensagens HTTP

- Dois tipos de mensagens:
 - request (requisição);
 - response (resposta).



Mensagem de requisição HTTP

- formato ASCII;
- linhas separadas por CR/LF;
- linha obrigatória: linha de requisição;
- linhas de cabeçalho são opcionais.

Linha de requisição
(comandos GET, POST, HEAD)

Linhas de cabeçalho

```
GET /somedir/page.html HTTP/1.0
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/4.0
Accept-language: fr
```

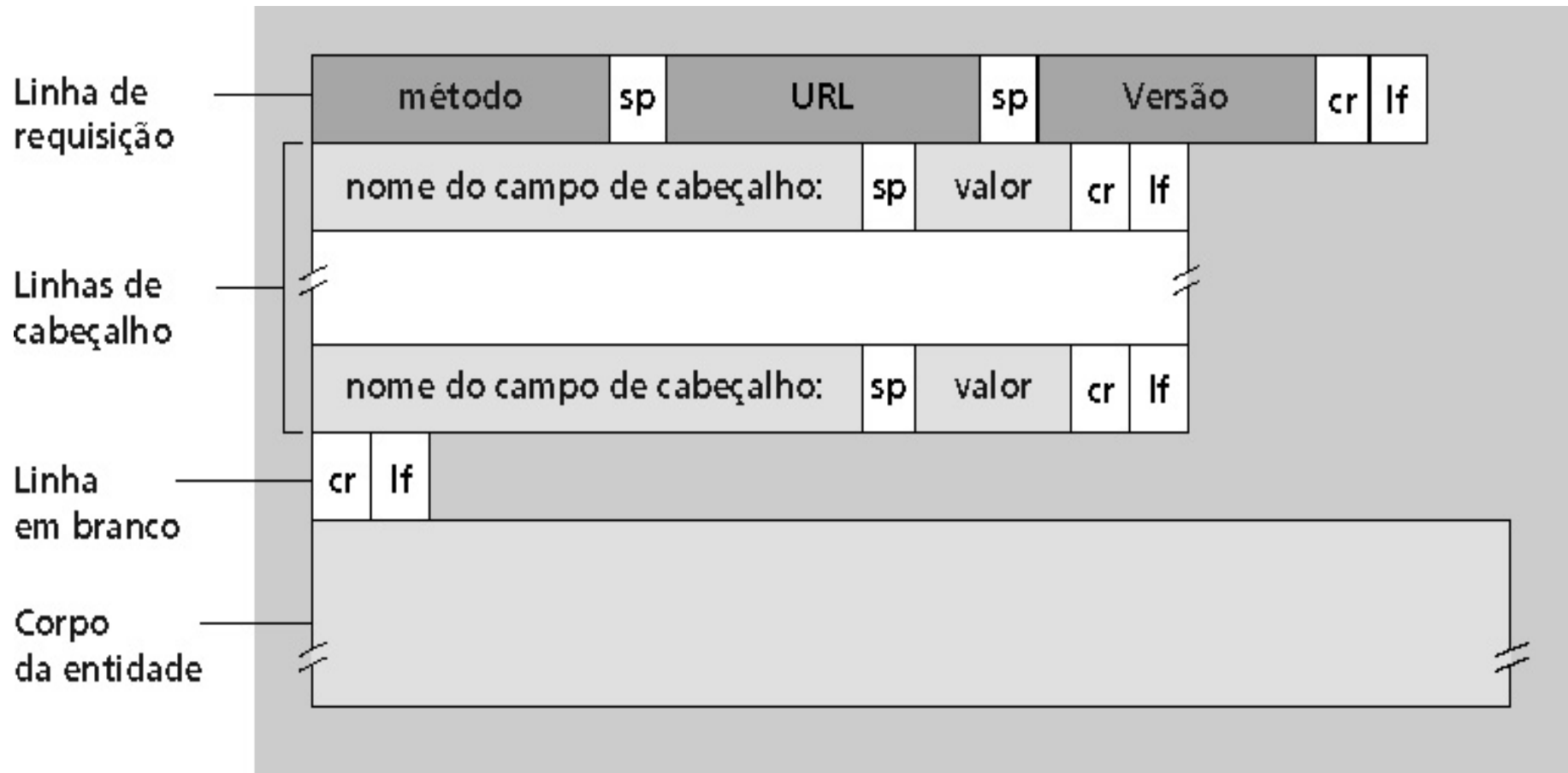
indica fim da mensagem

(carriage return, line feed adicional)



Mensagem de requisição

HTTP: Formato Geral





Tipos de Métodos – HTTP/1.0

- GET:
 - requisita objetos ao servidor web.
- POST:
 - usado para enviar dados ao servidor;
 - “corpo da entidade” contém dados fornecidos pelo usuário.
- HEAD:
 - similar ao GET;
 - porém servidor deixa o objeto requisitado fora da resposta;
 - usado por desenvolvedores para *debugging*.



Tipos de Métodos - HTTP/1.1

- GET, POST, HEAD.
- Outros métodos, entre eles:
 - PUT:
 - envia o arquivo no corpo da entidade para o caminho especificado no campo de URL.
 - DELETE:
 - apaga o arquivo especificado no campo de URL.
 - usados por ferramentas de edição para enviar/remover arquivos no servidor.



Mensagem de resposta HTTP

Linha de estado
(protocolo, código de
estado, msg de estado)

Linhas de cabeçalho

Linha em branco

Dados, ex.:
arquivo html

```
HTTP/1.0 200 OK
Connection: Close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 .....
Content-Length: 6821
Content-Type: text/html
```

```
data data data data data ...
```



Mensagem de resposta HTTP

Content-Type: text/html

- Como saber o tipo de arquivo?
 - <http://www.iana.org/assignments/media-types/>
- Alguns exemplos:
 - text/html
 - image/gif
 - image/jpeg

Códigos de Estado

- 200 OK: requisição bem-sucedida, objeto requisitado a seguir nesta mensagem.
- 301 Moved permanently: objeto requisitado foi movido, nova localização especificada a seguir nesta mensagem (Location:).
- 400 Bad request: mensagem de requisição não compreendida pelo servidor.
- 404 Not Found: documento requisitado não encontrado neste servidor.
- 505 HTTP version not supported



Simulando um cliente HTTP...

1. Telnet para um servidor Web:

```
telnet www.uol.com.br 80
```

Abre conexão TCP para a porta 80
(porta default do servidor HTTP)
em www.uol.com.br
Qualquer coisa digitada é enviada
para a porta 80 em www.uol.com.br

2. Digite um pedido GET HTTP:

```
GET / HTTP/1.1  
host: www.uol.com.br
```

Digitando isso (tecle carriage
return duas vezes), você envia este
pedido HTTP GET mínimo (mas
completo) ao servidor HTTP

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



Interação usuário- servidor: cookies

- Servidores http não possuem controle de estado.
- Uso de cookies permite que o usuário seja monitorado.
- A maioria dos grandes sites Web utiliza cookies:
 - Yahoo,
 - Amazon,
 - etc...

Interação usuário-servidor: cookies

Quatro componentes:

- 1) Linha de cabeçalho de cookie na mensagem de resposta HTTP.
- 2) Linha de cabeçalho de cookie na mensagem de requisição HTTP.
- 3) Arquivo de cookie mantido no computador do usuário e gerenciado pelo browser do usuário.
- 4) Banco de dados de apoio no website.

Cookies: mantendo o estado

Cliente

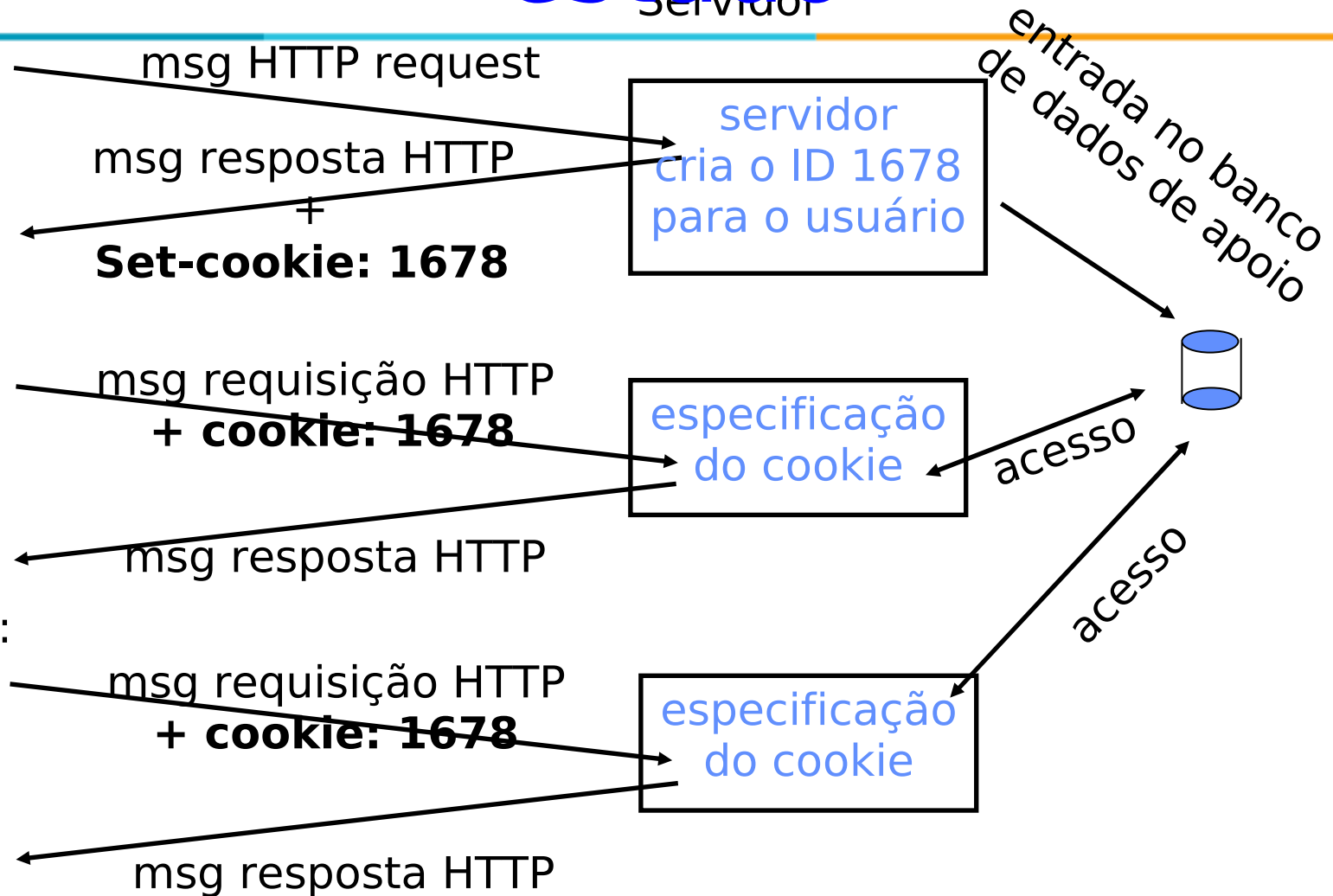
Servidor

arquivo Cookie

ebay: 8734

Cookie fileamazon: 1678
ebay: 8734

Uma semana depois:

Cookie fileamazon: 1678
ebay: 8734



Cookies

O que os cookies podem trazer:

- Autorização
- Cartões de compra
- Recomendações
- Estado de sessão do usuário (Web e-mail)



Cookies e privacidade

- Cookies permitem que sites saibam muito sobre você!
- Você pode fornecer nome e e-mail para os sites.
- Mecanismos de busca usam redirecionamento e cookies para saberem mais sobre você.
- Companhias de propaganda obtêm informações por meio dos sites.

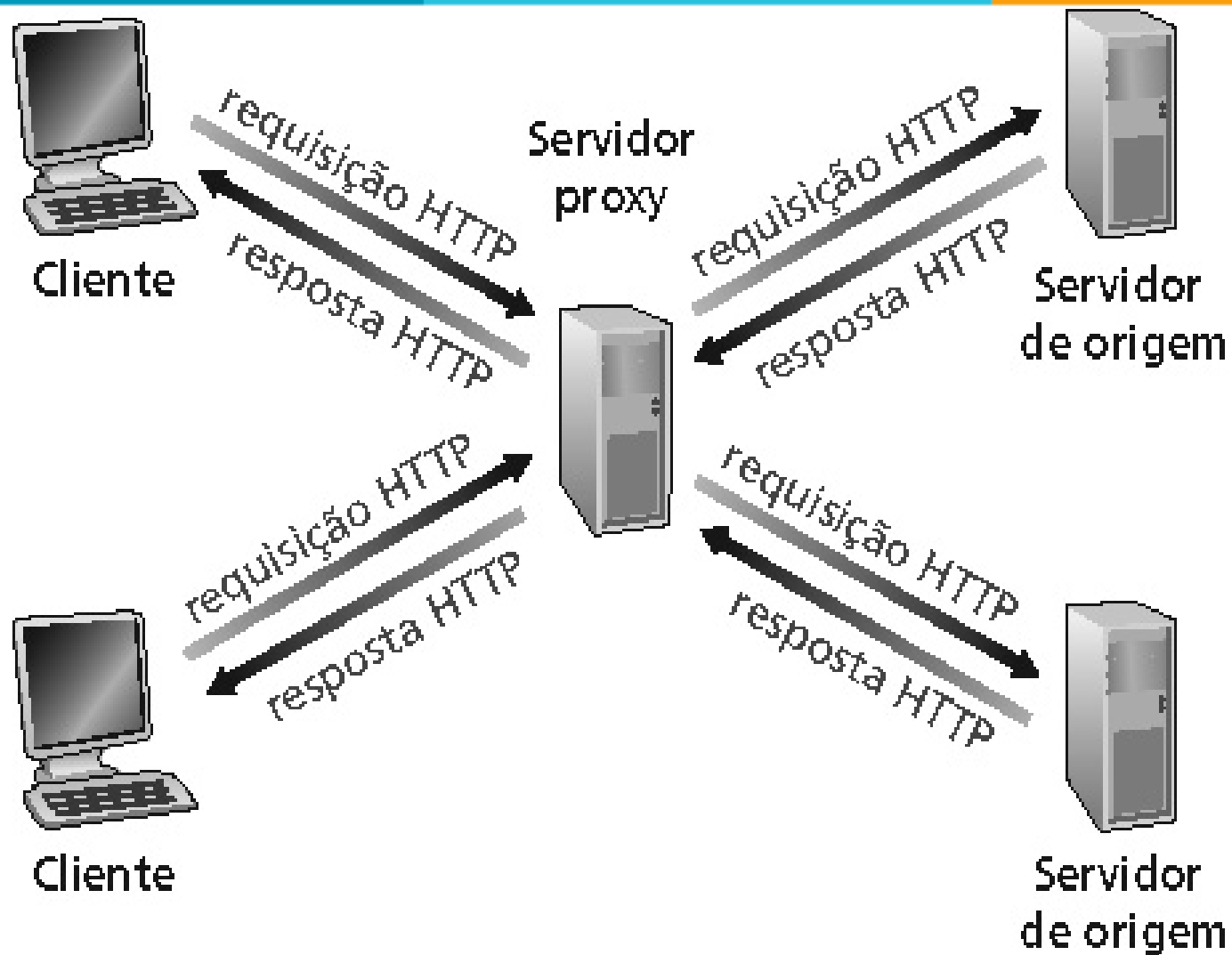


Caches Web

- Objetivo: atender o cliente sem envolver o servidor Web originador da informação.
- Tipicamente, é instalado pelo ISP (universidade, empresa, ISP residencial).
- Por que?
 - reduz tempo de resposta;
 - reduz tráfego no enlace de acesso.



Caches Web





Caches Web

- Caches reduzem tempo de resposta a requisição.
- Porém, as páginas armazenadas podem estar desatualizadas!!
- Como solucionar este problema?
 - HTTP possui o GET condicional.

GET condicional

- Cliente: especifica data da versão armazenada no pedido HTTP:

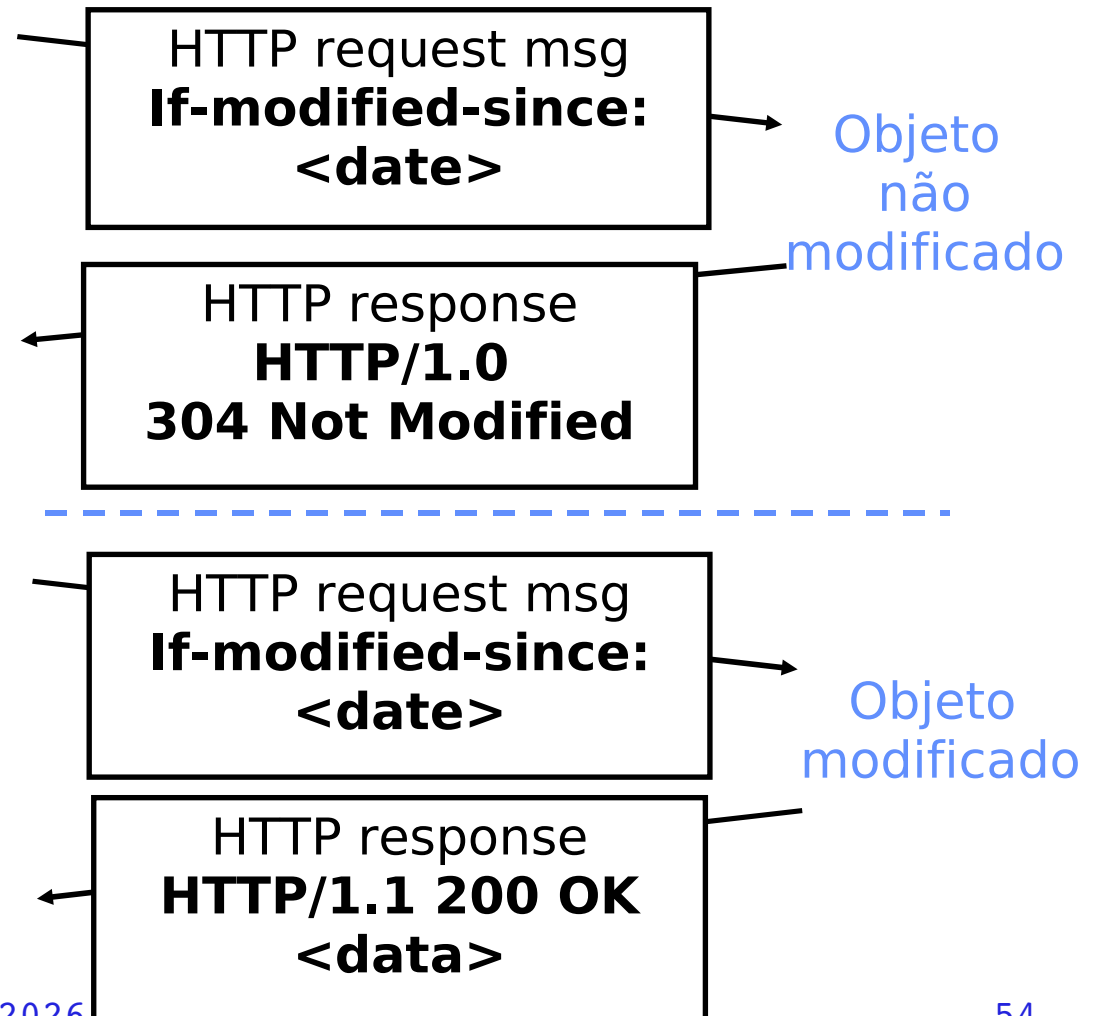
If-modified-since:
<date>

- Servidor: resposta não contém objeto se a cópia é atualizada:

HTTP/1.0 304 Not Modified

Cliente

Servidor





Perguntas???

- Próxima aula: DNS, Email, socket.
- Lab sobre HTTP e DNS no dia 30/ago.
- Tarefa:
 - lab ethereal/wireshark de familiarização;
 - enunciado disponível no CoL;
 - necessário para as próximas atividades.