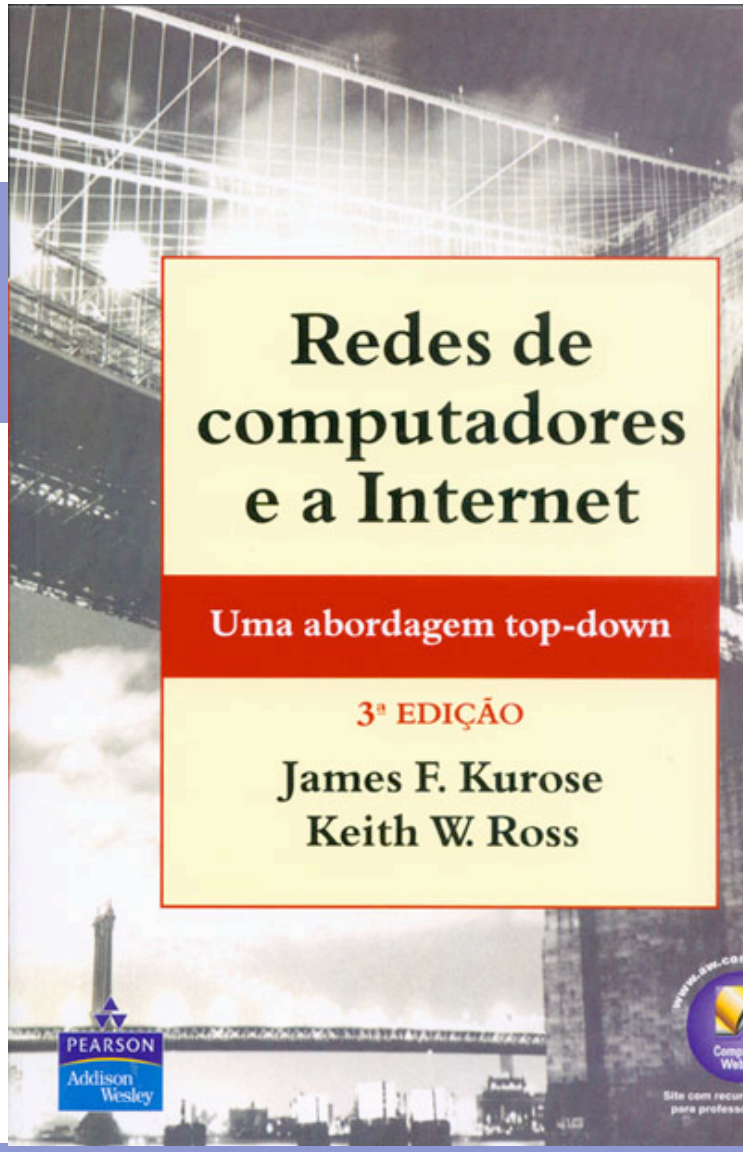
Redes de computadores e a Internet

Capítulo 2

Camada de aplicação



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

2 Parte 2: Camada de aplicação

Nossos objetivos:

- Conceitual, aspectos de implementação de protocolos de aplicação de redes
  - Modelos de serviço da camada de transporte
  - Paradigma cliente-servidor
  - Paradigma **peer-to-peer**
  - Aprender sobre protocolos examinando protocolos da camada de aplicação populares:
  - HTTP
  - FTP
  - SMTP/POP3/IMAP
    - DNS
- Programação de aplicações de rede
    - Socket API

2 Algumas aplicações de rede

- E-mail
- Web
- Mensagem instantânea
- Login remoto
- P2P file sharing
- Jogos de rede multiusuário
- Streaming stored videoclipes
- Telefonia via Internet
- Videoconferência em tempo real
- Computação paralela massiva

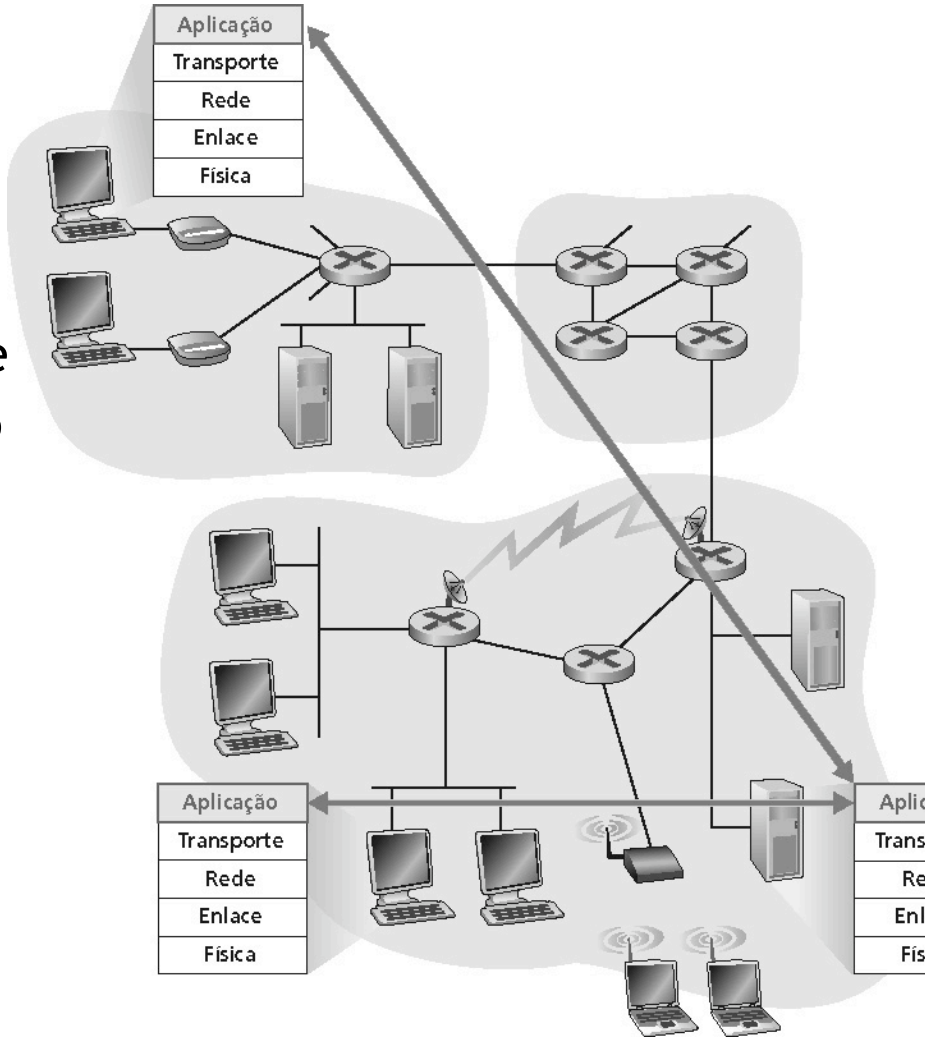
2 Criando uma nova aplicação de rede

Escrever programas que

- Executem sobre diferentes sistemas finais e
- 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

- Dispositivos do núcleo da rede não trabalham na camada de aplicação
- Esta estrutura permite um rápido desenvolvimento de aplicação



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

2 Arquiteturas de aplicação

- Cliente-servidor
- **Peer-to-peer** (P2P)
- Híbrida de cliente-servidor e P2P

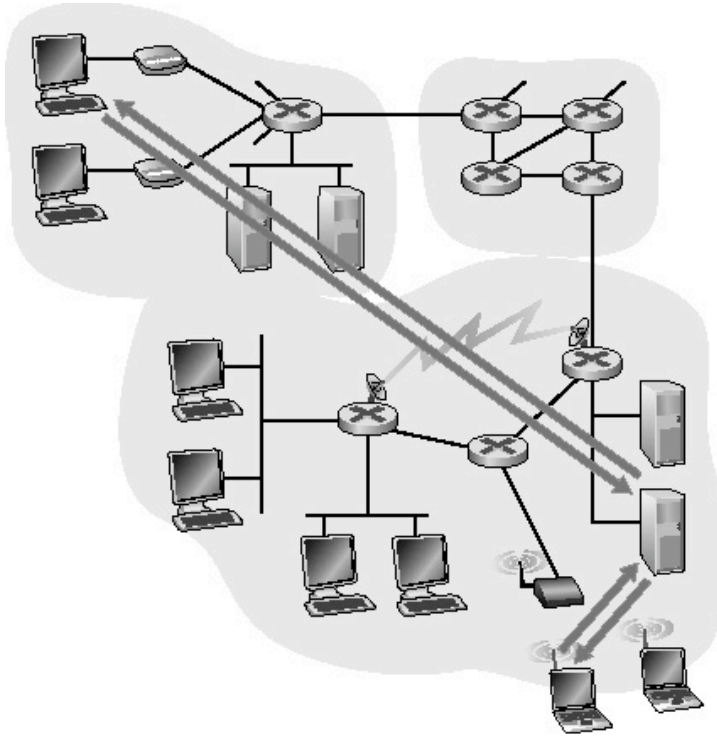
2 Arquitetura cliente-servidor

Servidor:

- 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

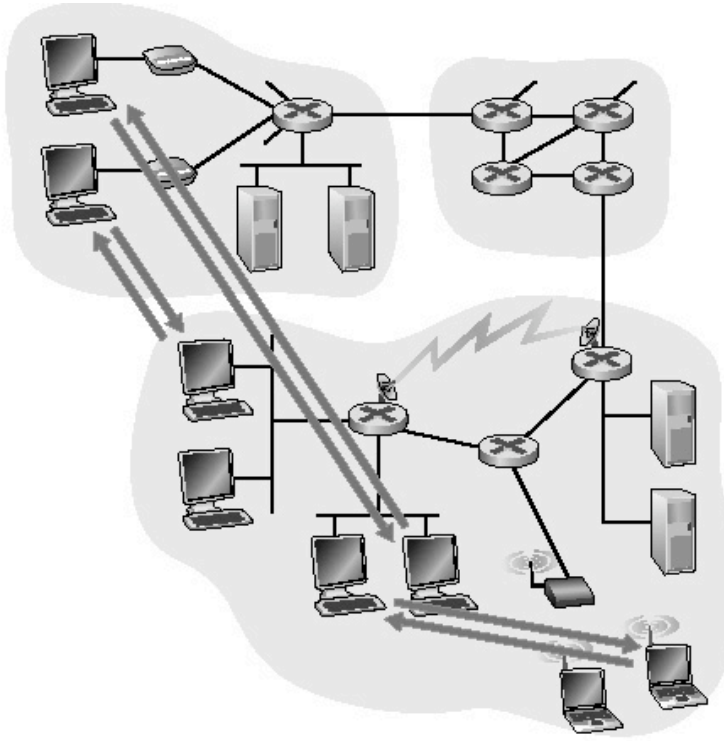


a. Aplicação cliente-servidor

2 Arquitetura P2P pura

- Nem sempre no servidor
- Sistemas finais arbitrários comunicam-se diretamente
- Pares são intermitentemente conectados e trocam endereços IP
- Ex.: Gnutella

Altamente escaláveis mas difíceis de gerenciar



b. Aplicação P2P

2 Híbrida de cliente-servidor e P2P

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

2 Comunicação de processos

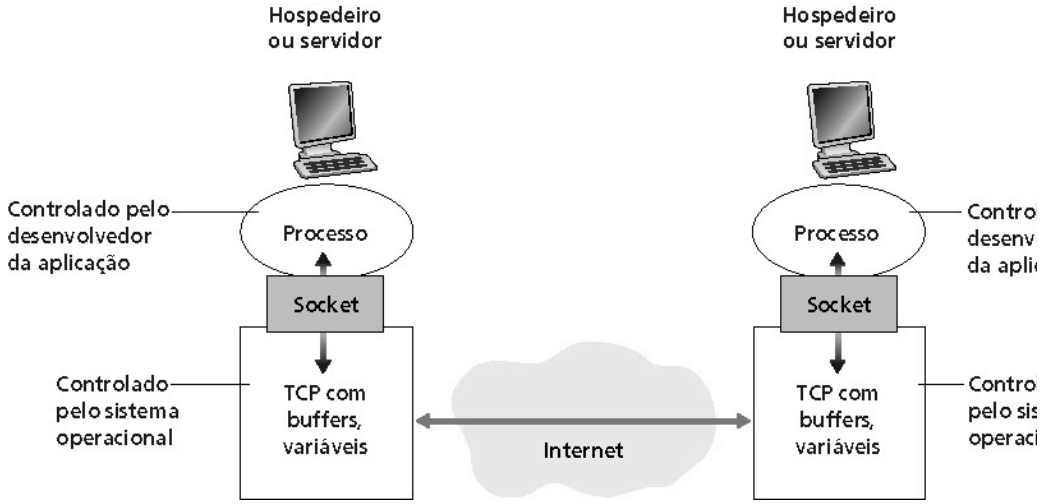
Processo: programa executando num hospedeiro

- Dentro do mesmo hospedeiro: dois processos se comunicam usando comunicação interprocesso (definido pelo OS)
- Processos em diferentes hospedeiros se comunicam por meio de troca de mensagens
- Processo cliente: processo que inicia a comunicação
- Processo servidor: processo que espera para ser contatado

Nota: aplicações com arquiteturas P2P possuem processos cliente e processos servidor

2 Sockets

- Um processo envia/recebe mensagens para/de seu socket
- O socket é análogo a uma porta
  - O processo de envio empurra a mensagem para fora da porta
  - O processo de envio confia na infra-estrutura de transporte no outro lado da porta que leva a mensagem para o socket no processo de recepção



- API: (1) escolha do protocolo de transporte; (2) habilidade para fixar poucos parâmetros (será explicado mais tarde)

2 Processos de endereçamento

- Para um processo receber mensagens, ele deve ter um identificador
- Um hospedeiro possui um único endereço IP de 32 bits
- P.: O endereço IP do hospedeiro onde o processo está executando é suficiente para identificar o processo?
- R.: Não, muitos processos podem estar em execução no mesmo hospedeiro
- O identificador inclui o endereço IP e o número da porta associada ao processo no hospedeiro
- Exemplos de números de porta:
  - Servidor HTTP: 80
  - Servidor de Correio: 25
- (mais detalhes serão mostrados mais tarde)

2 O protocolo da camada de aplicação define

- Tipo das mensagens trocadas, mensagens de requisição e resposta
- Sintaxe dos tipos de mensagem: os campos nas mensagens e como são delineados
- Semântica dos campos, ou seja, significado da informação nos campos
- Regras para quando e como os processos enviam e respondem às mensagens

Protocolos de domínio público:

- Definidos nas RFCs
- Recomendados para interoperabilidade
- Ex.: HTTP, SMTP

Protocolos proprietários:

- Ex.: KaZaA

2 De qual serviço de transporte uma aplicação necessita?

Perda de dados

- Algumas aplicações (ex.: áudio) podem tolerar alguma perda
- Outras aplicações (ex.: transferência de arquivos, telnet) exigem transferência de dados 100% confiável

Temporização

- Algumas aplicações (ex.: telefonia Internet, jogos interativos) exigem baixos atrasos para serem “efetivos”

Banda passante

- Algumas aplicações (ex.: multimídia) exigem uma banda mínima para serem “efetivas”
- Outras aplicações (“aplicações elásticas”) melhoram quando a banda disponível aumenta

2 Requisitos de transporte de aplicação comuns

Aplicação	Perdas	Banda	Sensível ao atraso
file transfer	sem perdas	elástica	não
e-mail	sem perdas	elástica	não
Web documents	tolerante	elástica	não
real-time áudio/vídeo	tolerante	áudio: 5 Kb-1 Mb vídeo:10 Kb-5 Mb	sim, 100's mseg
stored áudio/vídeo	tolerante	igual à anterior	sim, segundos
jogos interativos	tolerante	kbps	sim, 100's mseg
e-business	sem perda	elástica	sim

2 Serviços dos protocolos de transporte da Internet

Serviço TCP:

- **Orientado à conexão:** conexão requerida entre processos cliente e servidor
- **Transporte confiável** entre os processos de envio e recepção
- **Controle de fluxo:** o transmissor não sobrecarrega o receptor
- **Controle de congestionamento:** protege a rede do excesso de tráfego

  Não oferece: garantias de temporização e de banda mínima

Serviço UDP:

- Transferência de dados não confiável entre os processos transmissor e recept
- Não oferece: estabelecimento de conexão, confiabilidade, controle de fluxo de congestionamento, garantia de temporização e de banda mínima

P.: Por que ambos? Por que existe o UDP?

Aplicação e protocolos de transporte da Internet

Aplicação	Protocolo de aplicação	Protocolo de transporte
e-mail	smtp [RFC 821]	TCP
acesso de terminais remotos	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
transferência de arquivos	ftp [RFC 959]	TCP
streaming multimídia	RTP ou proprietário (ex.: RealNetworks)	TCP ou UDP
servidor de arquivos remoto	NSF	TCP ou UDP
telefonia Internet	RTP ou proprietário (ex.: Vocaltec)	tipicamente UDP

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

2 Web e HTTP

Primeiro alguns jargões

- 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

2 Visão geral do HTTP

HTTP: hypertext transfer protocol

- Protocolo da camada de aplicação da Web
- Modelo cliente/servidor
  - Cliente: browser que solicita, recebe e apresenta objetos da Web
  - Servidor: envia objetos em resposta a pedidos
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068

2 Visão geral do HTTP

Utiliza TCP:

- Cliente inicia conexão TCP (cria socket) para o servidor na porta 80
- Servidor aceita uma conexão TCP do cliente
- mensagens HTTP (mensagens do protocolo de camada de aplicação) são trocadas entre o browser (cliente HTTP) e o servidor Web (servidor HTTP)
- A conexão TCP é fechada

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

2 Conexões HTTP

HTTP não persistente

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

HTTP persistente

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

2 HTTP não persistente

Usuário entra com a URL: (contém texto, referências a 10 imagens j
`www.someSchool.edu/someDepartment/home.index`

1a. Cliente HTTP inicia conexão TCP ao servidor HTTP (processo) em www.someSchool.edu. Porta 80 é a default para o servidor HTTP

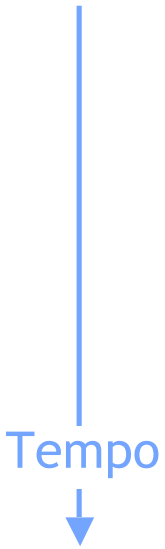
1b. Servidor HTTP no hospedeiro www.someSchool.edu esperando pela conexão TCP na porta 80. "Aceita" conexão, notificando o cliente

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 (someDepartment/home.index), envia mensagem para o socket

Tempo

2 HTTP não persistente



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

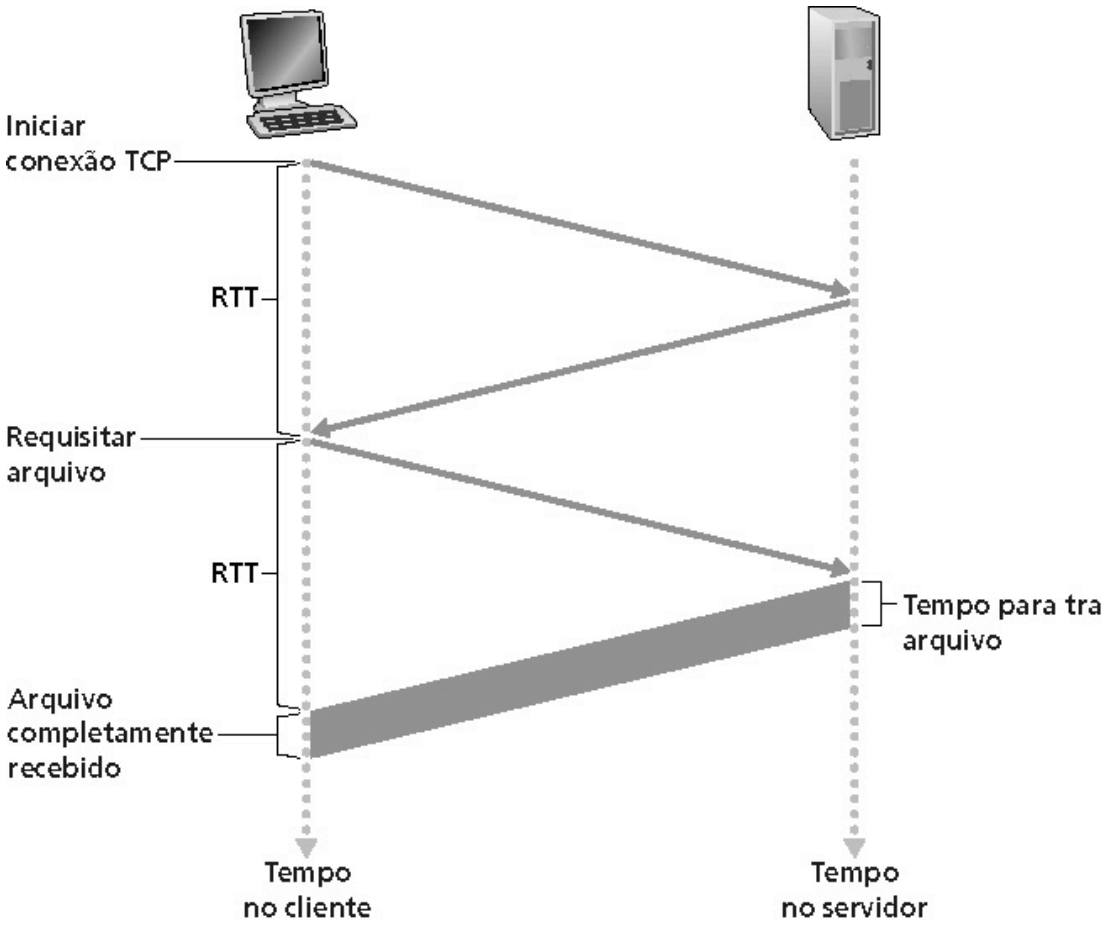
6. Passos 1-5 são repetidos para cada um dos 10 objetos jpeg

2 Modelagem do tempo de resposta

Definição de RRT: 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

2 HTTP persistente

Características do HTTP persistente:
- Requer 2 RTTs por objeto
- OS deve manipular e alocar recursos do hospedeiro para cada conexão TCP. Mas os browsers freqüentemente abrem conexões TCP paralelas para buscar objetos referenciados

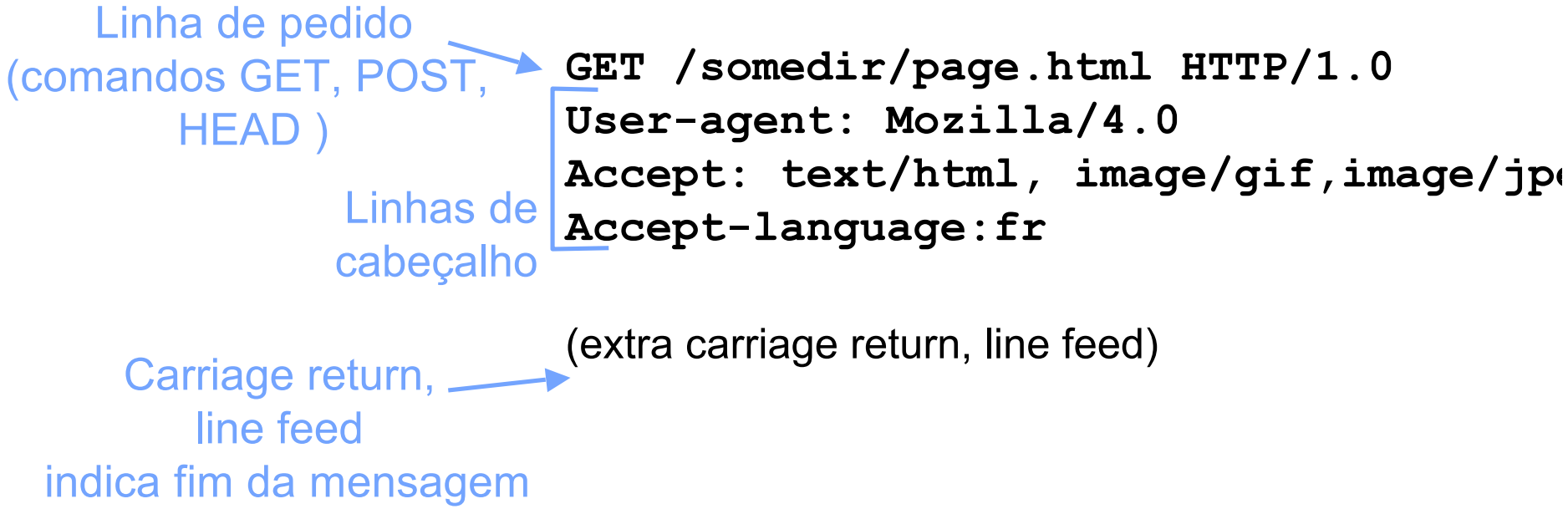
HTTP persistente
- Servidor deixa a conexão aberta após enviar uma resposta
- Mensagens HTTP subseqüentes entre o mesmo cliente/servidor são enviadas pela conexão

Persistente sem **pipelining**:
- O cliente emite novas requisições apenas quando a resposta anterior for recebida
- Um RTT para cada objeto referenciado

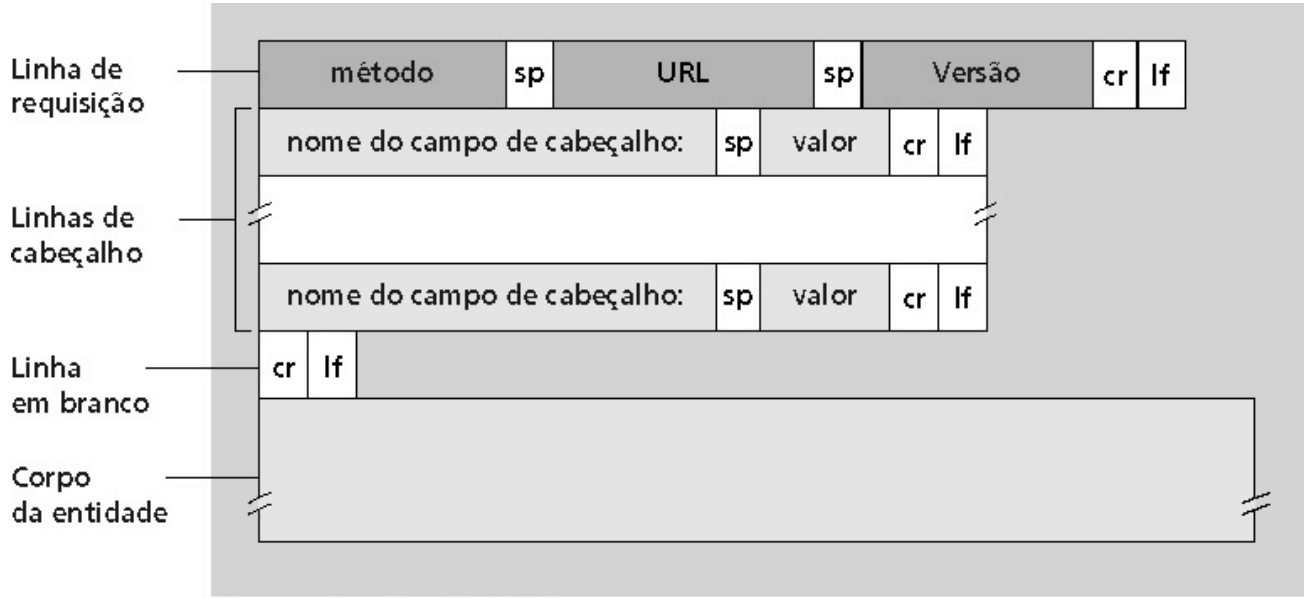
Persistente com **pipelining**:
- Padrão no HTTP/1.1
- O cliente envia requisições assim que encontra um objeto referenciado
- Tão pequeno como um RTT para todos os objetos referenciados

2 Mensagem HTTP request

- Dois tipos de mensagens HTTP: request, response
- HTTP request message:
  - ASCII (formato legível para humanos)



2 Mensagem HTTP request: formato geral



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

2 Entrada de formulário

Método Post:

- Página Web freqüentemente inclui entrada de formulário
- A entrada é enviada para o servidor no corpo da entidade

Método URL:

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

www.somesite.com/animalsearch?monkeys&banana

2 Tipos de métodos

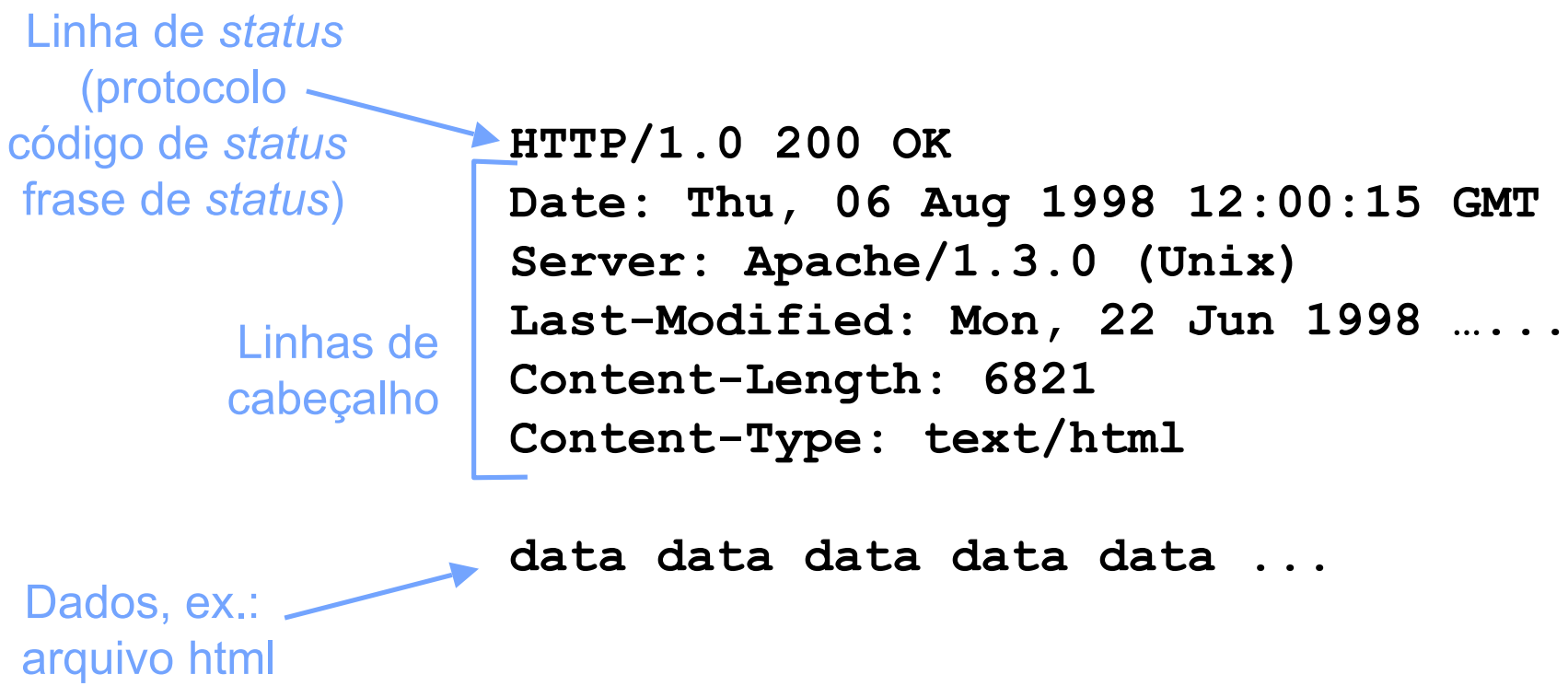
HTTP/1.0

- GET
- POST
- HEAD
  - Pede para o servidor deixar o objeto requisitado fora da resposta

HTTP/1.1

- GET, POST, HEAD
- 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

2 Mensagem HTTP response



2 Códigos de *status* das respostas

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

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

2 HTTP cliente: faça você mesmo!

1.Telnet para um servidor Web:

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

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

2.Digite um pedido GET HTTP:

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

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!

2 Estado usuário-servidor: cookies

A maioria dos grandes sites Web utiliza cookies

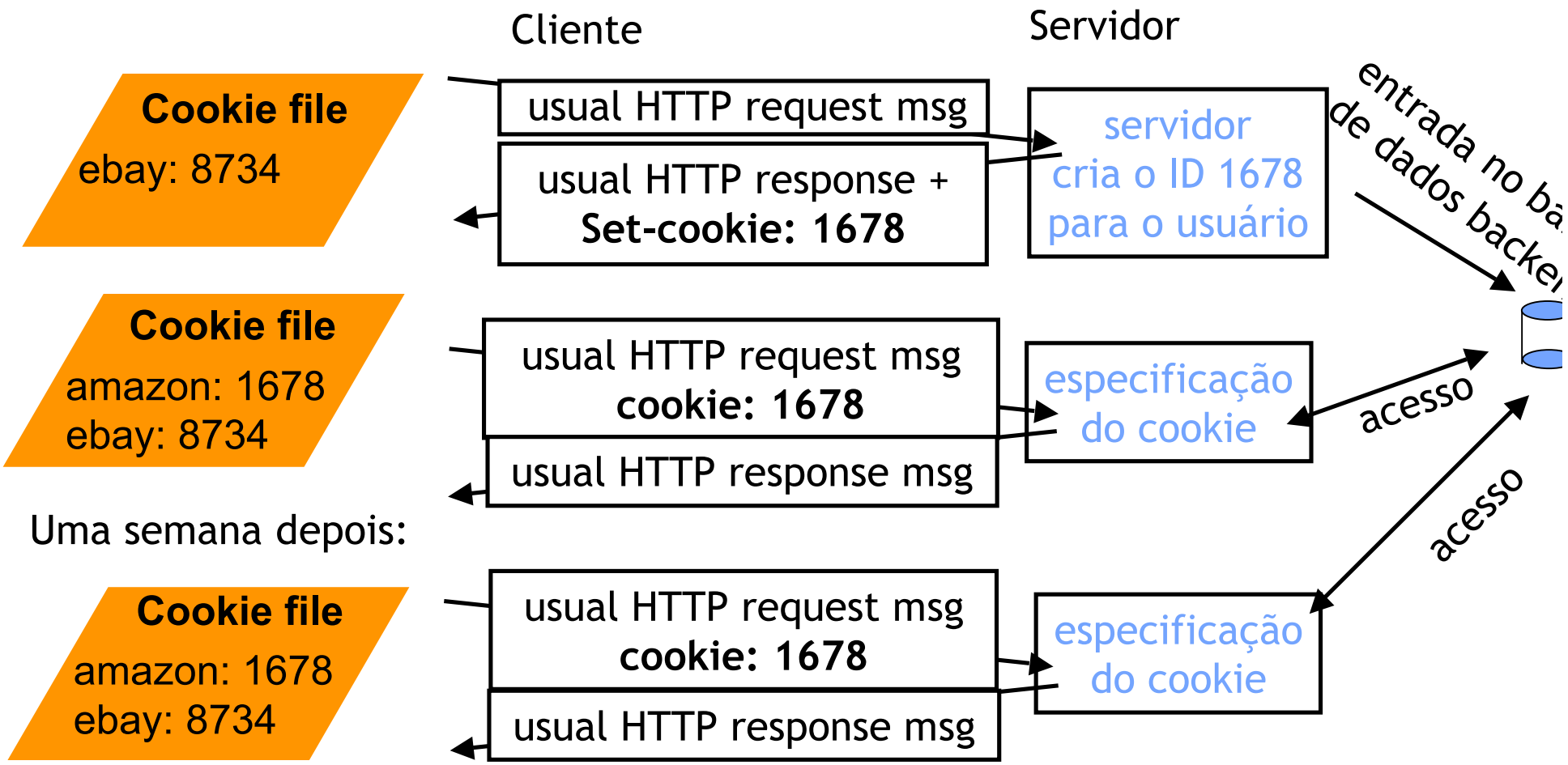
Quatro componentes:

1) Linha de cabeçalho do cookie na mensagem HTTP response
2) Linha de cabeçalho de cookie na mensagem HTTP request
3) Arquivo de cookie mantido no hospedeiro do usuário e manipulado pelo browser do usuário
4) Banco de dados **backend** no site Web

Exemplo:

- Susan acessa a Internet sempre do mesmo PC
- Ela visita um site específico de e-commerce pela primeira vez
- Quando a requisição HTTP inicial chega ao site, este cria um ID único e uma entrada no banco de dados **backend** para este ID

2 Cookies: mantendo “estado”



2 Cookies

O que os cookies podem trazer:

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

ASIDE

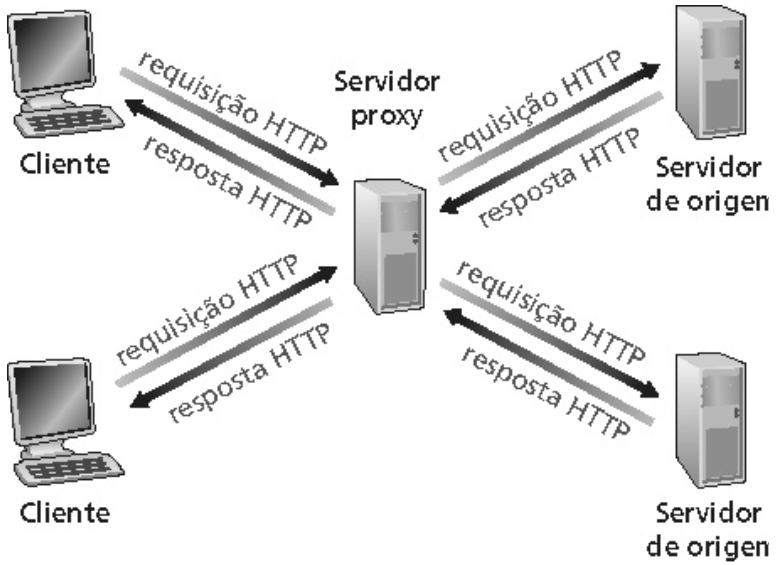
Cookies e privacidade:

- 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

2 Web caches (proxy server)

Objetivo: **atender o cliente sem envolver o servidor Web originador da informação**

- Usuário configura o browser: acesso Web é feito por meio de um proxy
- Cliente envia todos os pedidos HTTP para o Web cache
  - Se o objeto existe no Web cache: Web cache retorna o objeto
  - Ou o Web cache solicita objeto do servidor original e então envia o objeto ao cliente



2 Mais sobre Web caching

- O cache atua tanto no servidor como no cliente
- Tipicamente, o cache é instalado pelo ISP (universidade, empresa, ISP residencial)

Por que Web caching?

- Reduz o tempo de resposta para a requisição do cliente
- Reduz o tráfego num enlace de acesso de uma instituição
- A densidade de caches na Internet habilita os “fracos” provedores de conteúdo a efetivamente entregarem o conteúdo (mas fazendo P2P file sharing)

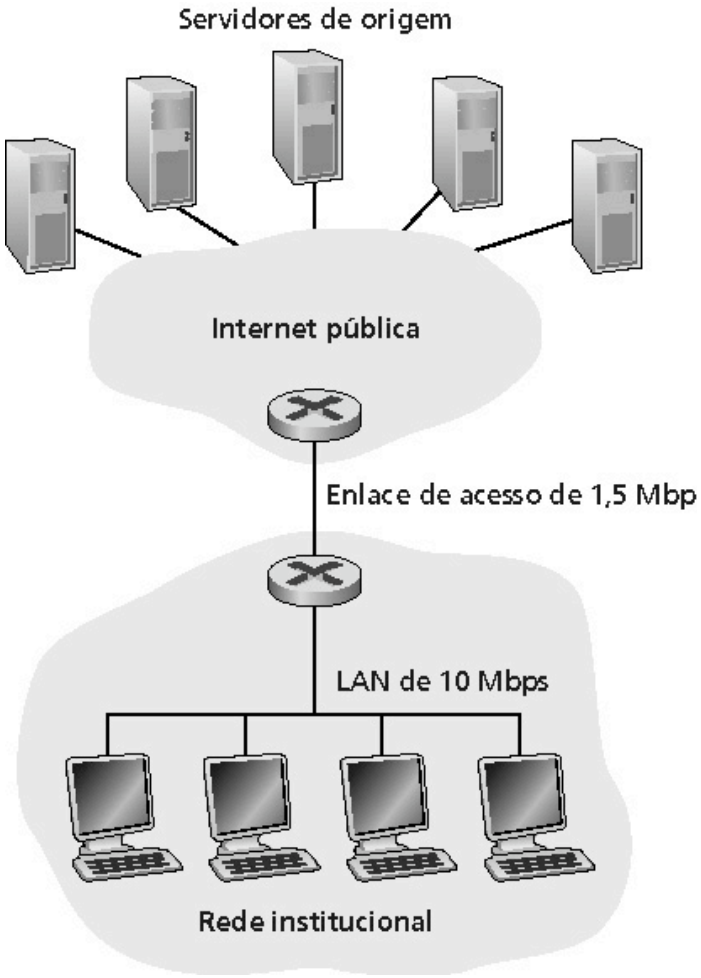
2 Exemplo de caching

Suponha:

- Tamanho médio objeto = 100.000 bits
- Taxa média de requisições dos browsers da instituição para os servidores de origem = 15/s
- Atraso do roteador institucional para ir a qualquer servidor de origem e retornar ao roteador = 2 s

Conseqüências:

- Utilização da LAN = 15%
- Utilização do link de acesso = 100%
- Atraso total = atraso da Internet + atraso de acesso + atraso da LAN = 2 segundos + minutos + milissegundos



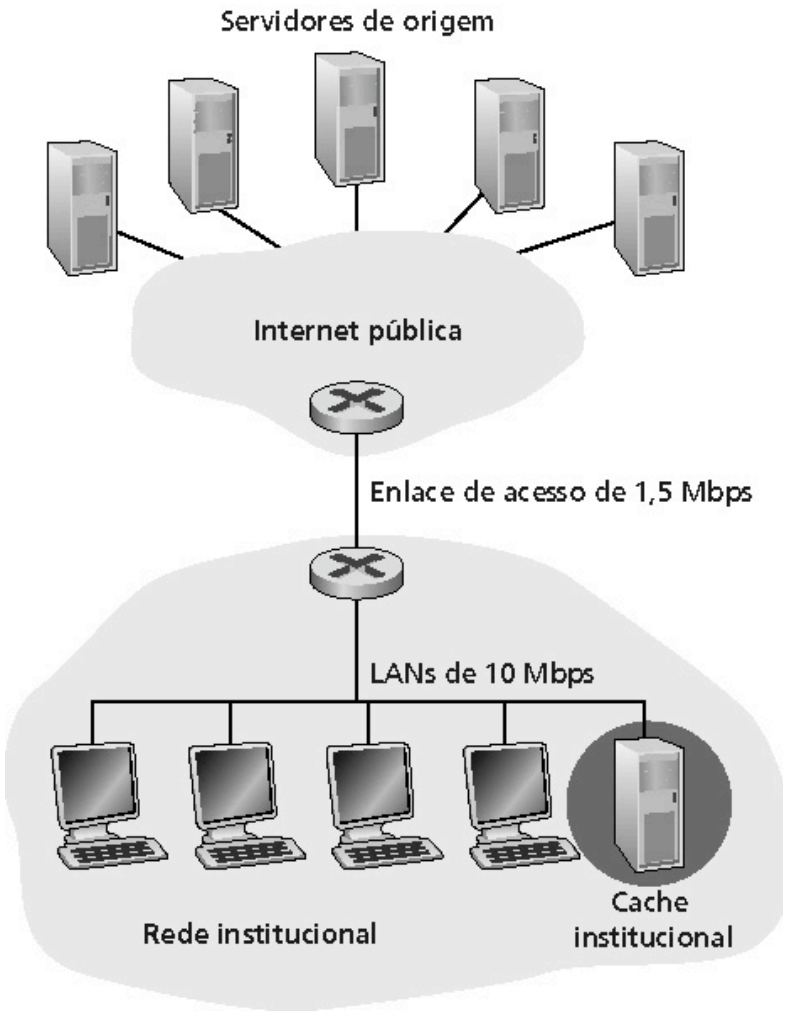
2 Exemplo de caching

Solução possível

- Aumentar a largura de banda do enlace de acesso, como, 10 Mbps

Conseqüências

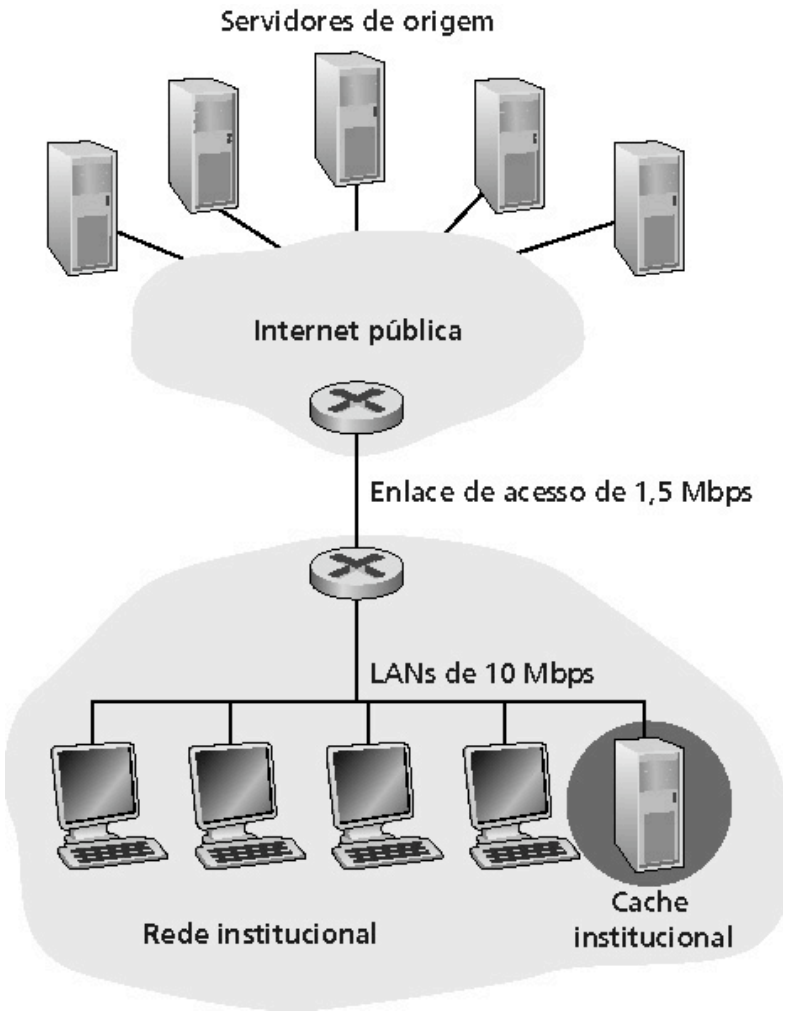
- Utilização da LAN = 15%
- Utilização do enlace de acesso = 15%
- Atraso total = atraso da Internet + atraso de acesso + atraso da LAN = 2 segundos + msegs + msegs
- Freqüentemente é um **upgrade** caro



2 Exemplo de caching

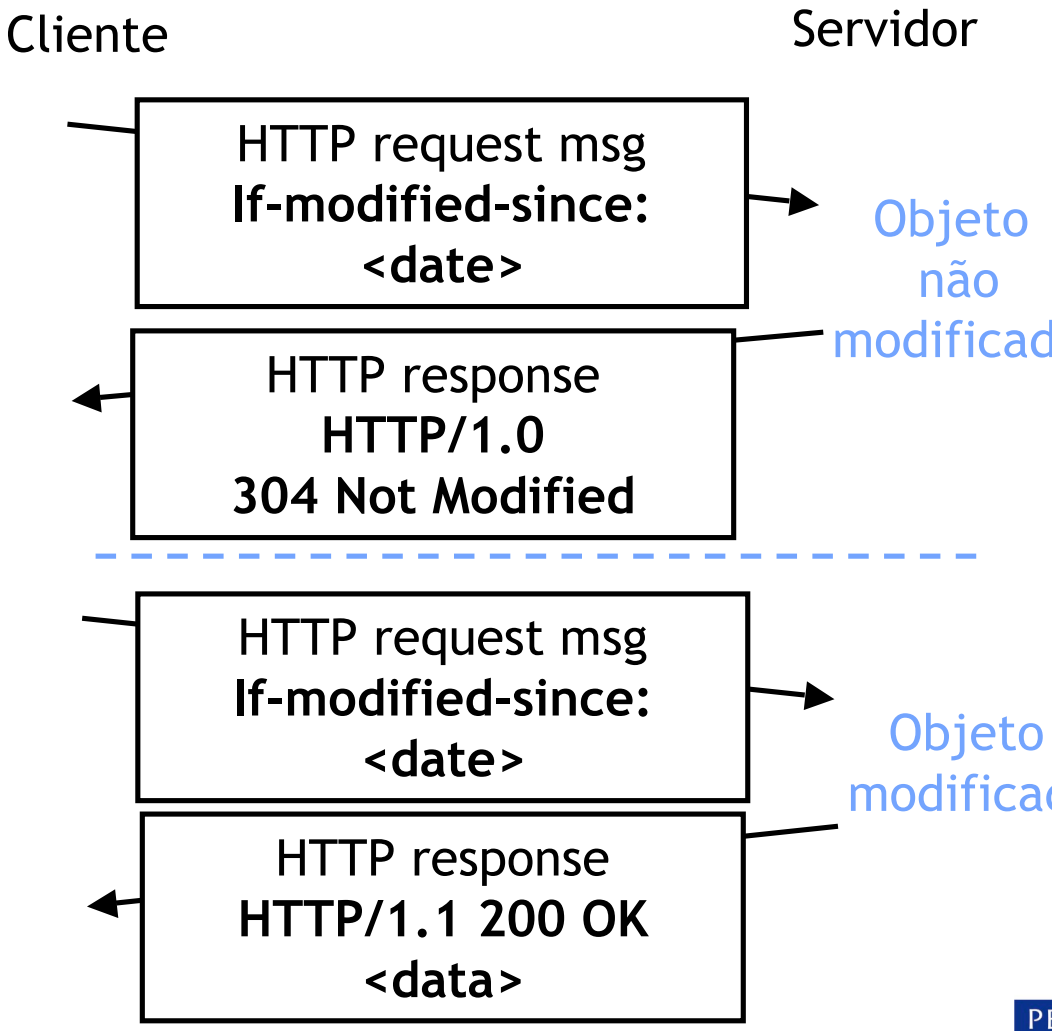
Instalação do cache
- Suponha que a taxa de acertos seja .4

Conseqüência
- 40% das requisições serão satisfeitas quase que imediatamente
- 60% das requisições serão satisfeitas pelo servidor de origem
- Utilização do enlace de acesso reduzida para 60%, resultando em atrasos insignificantes (como 10 mseg)
- Média de atraso total = atraso da Internet + atraso de acesso + atraso da LAN = .6*(2.01) segundos + milissegundos < 1,4 segundos



2 GET condicional

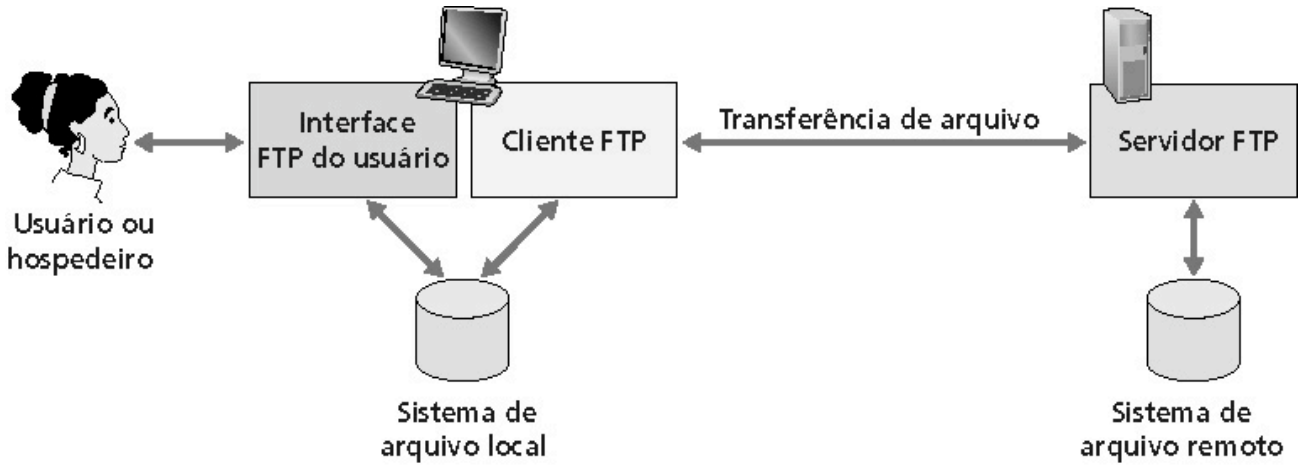
- Razão: não enviar objeto se a versão que o cliente já possui está atualizada
- Cliente: especifica data da versão armazenada no pedido HTTP
  - **If-modified-since: <date>**
- Servidor: resposta não contém objeto se a cópia é atualizada:
  HTTP/1.0 304 Not Modified



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

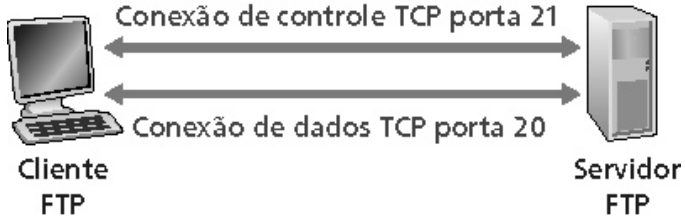
2 FTP: o protocolo de transferência de arquivos



- Transferência de arquivos de e para o computador remoto
- Modelo cliente servidor
  - Cliente: lado que inicia a transferência (seja de ou para o lado remoto)
  - Servidor: hospedeiro remoto
- FTP: RFC 959
- FTP servidor: porta 21

2 FTP: controle separado, conexões de dados

- Cliente FTP contata o servidor FTP na porta 21 especificando o TCP como protocolo de transporte
- Cliente obtém autorização pela conexão de controle
- Cliente procura o diretório remoto enviando comandos pela conexão de contro
- Quando o servidor recebe um comando para uma transferência de arquivo, ele abre uma conexão de dados TCP para o cliente
- Após a transferência de um arquivo, o servidor fecha a conexão
- Servidor abre uma segunda conexão de dados TCP para transferir outro arquivo
- Conexão de controle: "fora da banda"
- Servidor FTP mantém "estado": diretório atual, autenticação anterior



2 FTP comandos, respostas

Exemplos de comandos:

- Envie um texto ASCII sobre canal de controle
- **USER *username***
- **PASS *password***
- **LIST** retorna listagem do arquivo no diretório atual
- **RETR filename** recupera (obtém) o arquivo
- **STOR filename** armazena o arquivo no hospedeiro remoto

Exemplos de códigos de retorno

- Código de status e frase (como no HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

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

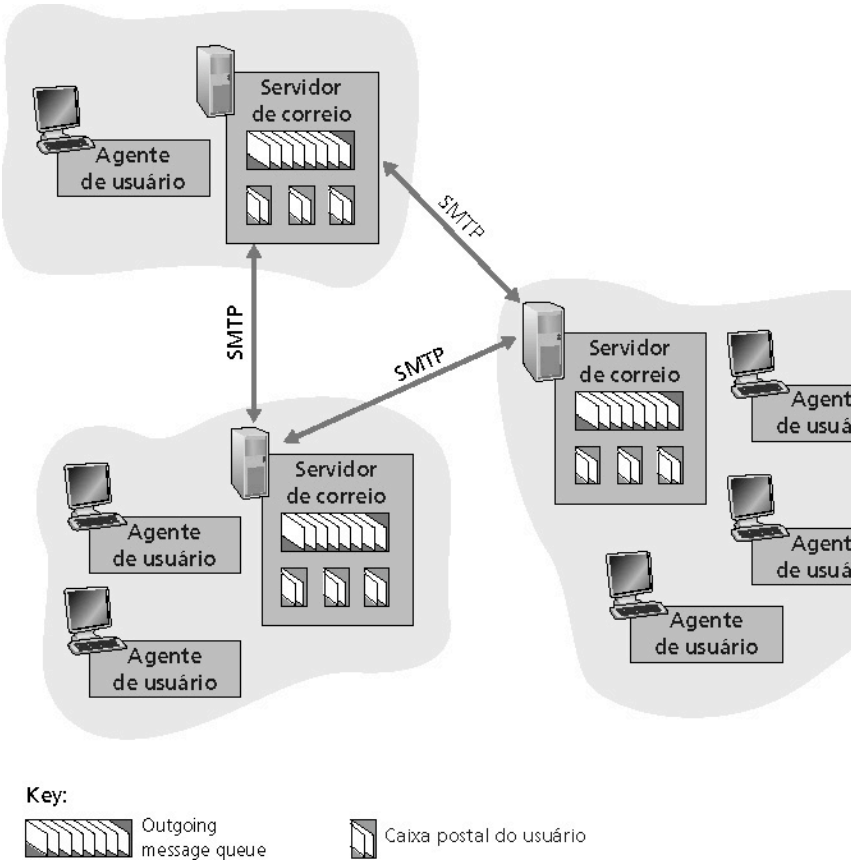
2 Correio eletrônico

Três componentes principais:

- Agentes de usuário
- Servidores de correio
- Simple mail transfer protocol: SMTP

Agente de usuário

- "leitor de correio"
- Composição, edição, leitura de mensagens de correio
- Ex.: Eudora, Outlook, elm, Netscape Messenger
- Mensagens de entrada e de saída são armazenadas no servidor



2 Correio eletrônico: servidores de correio

Servidores de correio

- Caixa postal contém mensagens que chegaram (ainda não lidas) para o usuário
- Fila de mensagens contém as mensagens de correio a serem enviadas

Protocolo SMTP permite aos servidores de correio trocarem mensagens entre si
- Cliente: servidor de correio que envia
- "Servidor": servidor de correio que recebe