

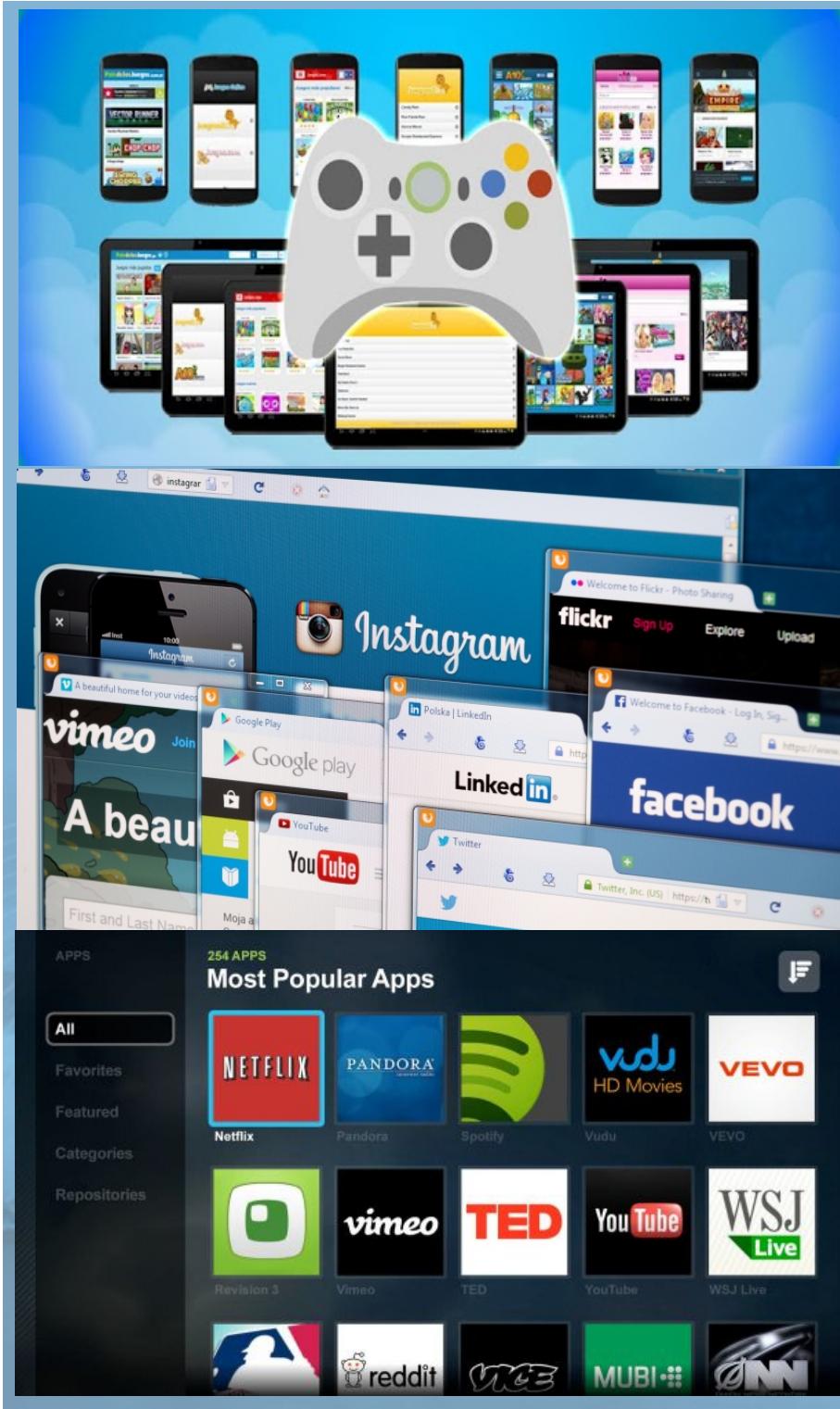
# Capítulo 2: Camada de aplicação

REDES DE  
COMPUTADORES  
E A INTERNET

5<sup>a</sup> edição

*Uma Abordagem Top-Down*

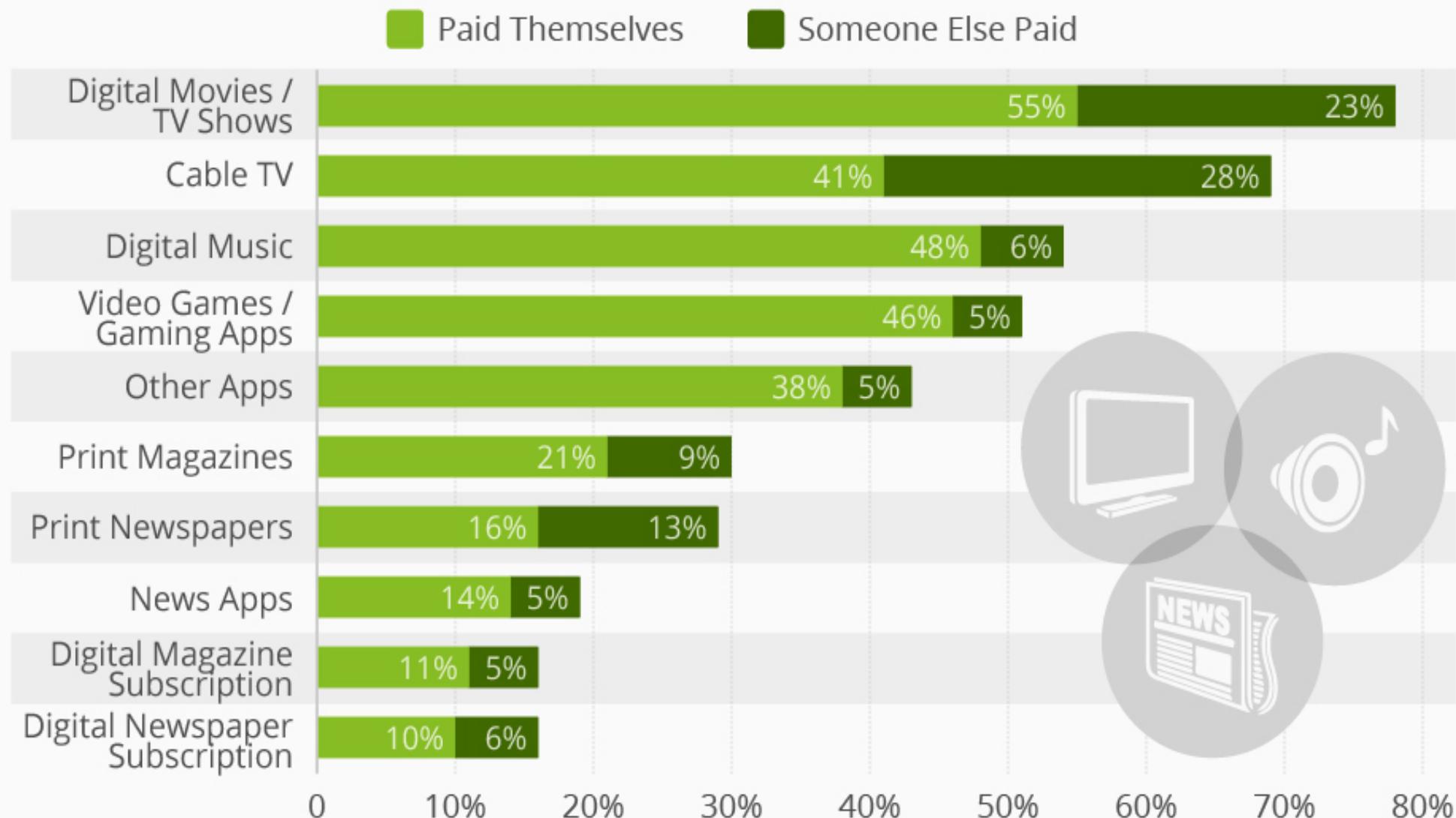
- 2.1 Princípios de aplicações de rede
- 2.2 A Web e o HTTP
- 2.3 FTP
- 2.4 Correio eletrônico
  - ❖ SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 Aplicações P2P
- 2.7 Programação de sockets com UDP
- 2.8 Programação de sockets com TCP



© 2010 Pearson Prentice Hall. Todos os direitos reservados.

# Millennials More Inclined to Pay for Entertainment Than for News

% of Millennials who regularly used the following paid products / services in the past year



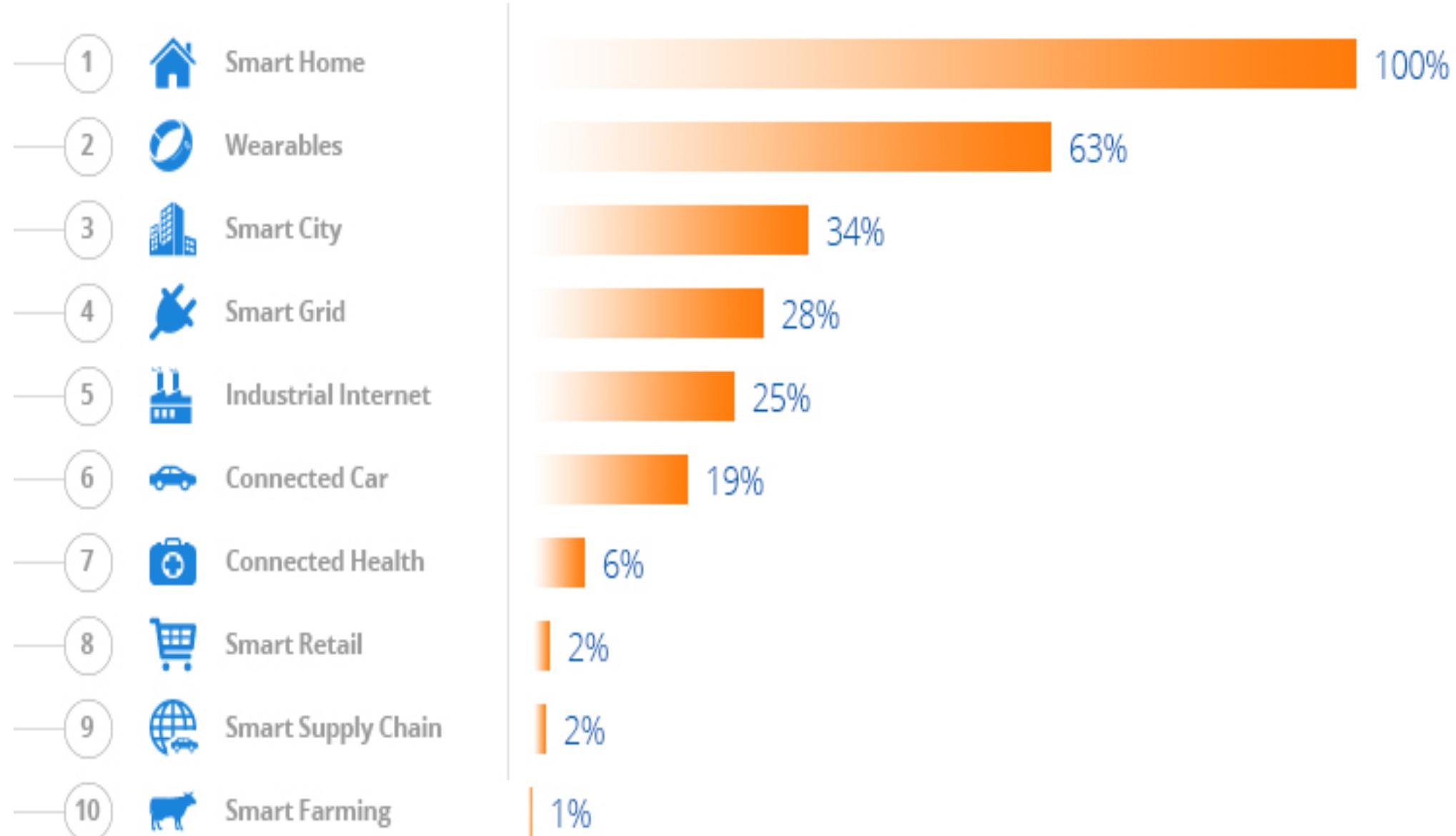
# The Top 10 Apps in the States

Mobile apps by unique visitors\*



# The 10 most popular Internet of Things applications

A ranking based on web analytics



# Ex aplicações em rede que usam a pilha de protocolos TCP/IP

- e-mail
- web
- mensagem instantânea
- login remoto
- compartilhamento de arquivos P2P
- jogos em rede multiusuários
- clipes de vídeo armazenados em fluxo contínuo
- redes sociais
- voice over IP
- vídeoconferência em tempo real
- computação em grade

REDES DE COMPUTADORES E A INTERNET 5<sup>a</sup> edição

*Uma Abordagem Top-Down*

# Criando uma aplicação de rede

Escreva programas que

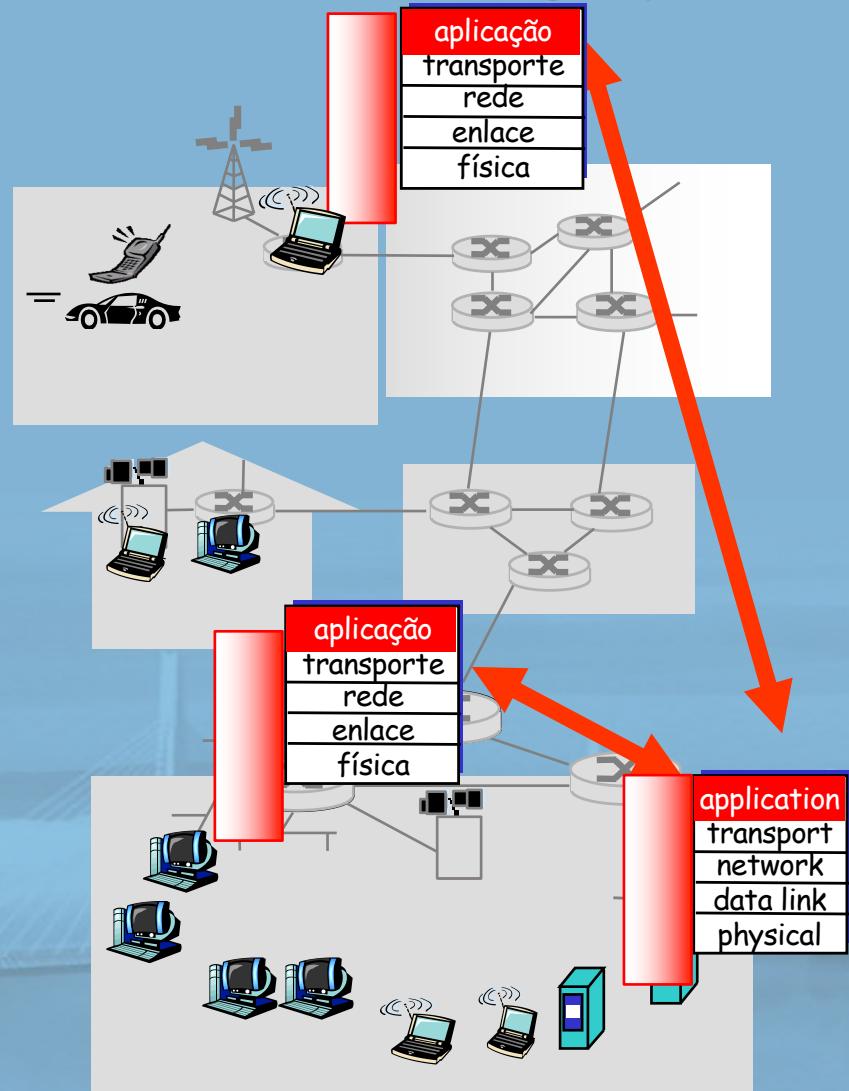
- ❖ executem em (diferentes) *sistemas finais*
- ❖ se comuniquem pela rede
- ❖ p. e., software de servidor Web se comunica com software de navegador Web

Não é preciso escrever software para dispositivos do núcleo da rede

- ❖ dispositivos do núcleo da rede não executam aplicações do usuário
- ❖ as aplicações nos sistemas finais permitem rápido desenvolvimento e propagação

REDES DE COMPUTADORES E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*



# Capítulo 2: Camada de aplicação

REDES DE  
COMPUTADORES  
E A INTERNET

5<sup>a</sup> 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

# Arquiteturas de aplicação

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

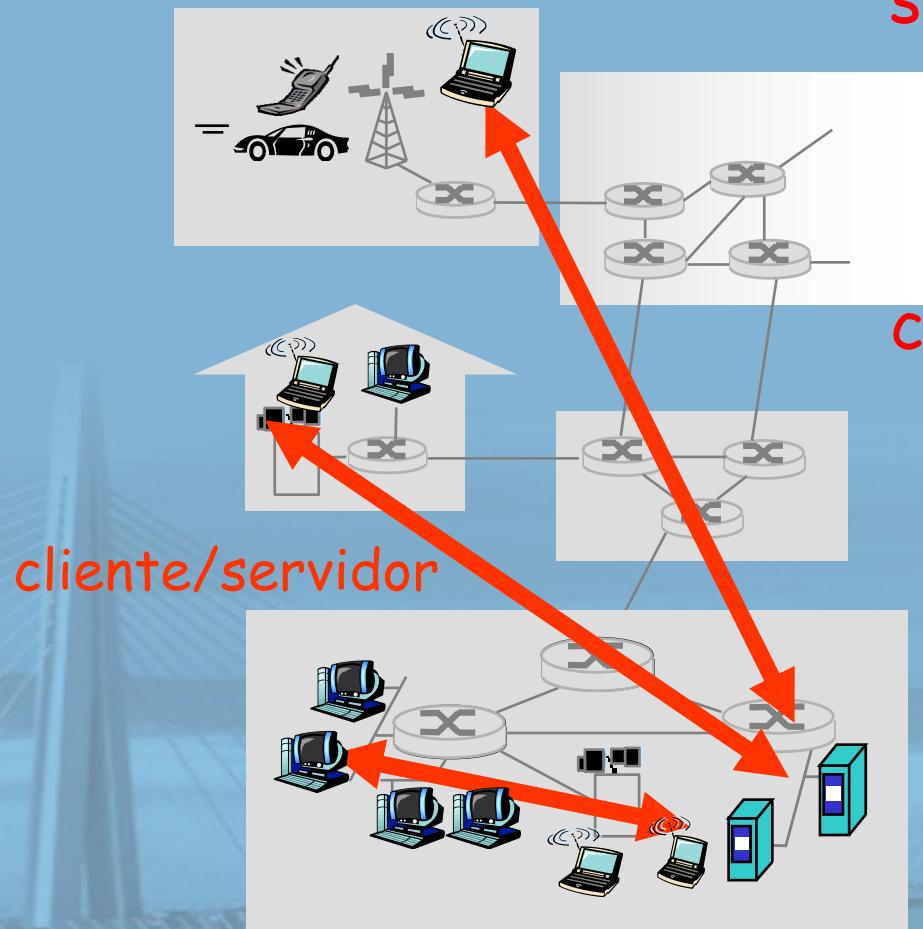
*Uma Abordagem Top-Down*

- Cliente-servidor
  - ❖ Incluindo centros de dados/cloud computing
- Peer-to-peer (P2P)
- Híbrida de cliente-servidor e P2P

# Arquitetura cliente-servidor

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*



## servidor:

- ❖ hospedeiro sempre ligado
- ❖ endereço IP permanente
- ❖ server farms por expansão

## clientes:

- ❖ comunicam-se com o servidor
- ❖ podem estar conectados intermitentemente
- ❖ podem ter endereços IP dinâmicos
- ❖ não se comunicam diretamente entre si

# Centros de dados da Google

REDES DE  
COMPUTADORES  
E A INTERNET

5<sup>a</sup> edição

*Uma Abordagem Top-Down*

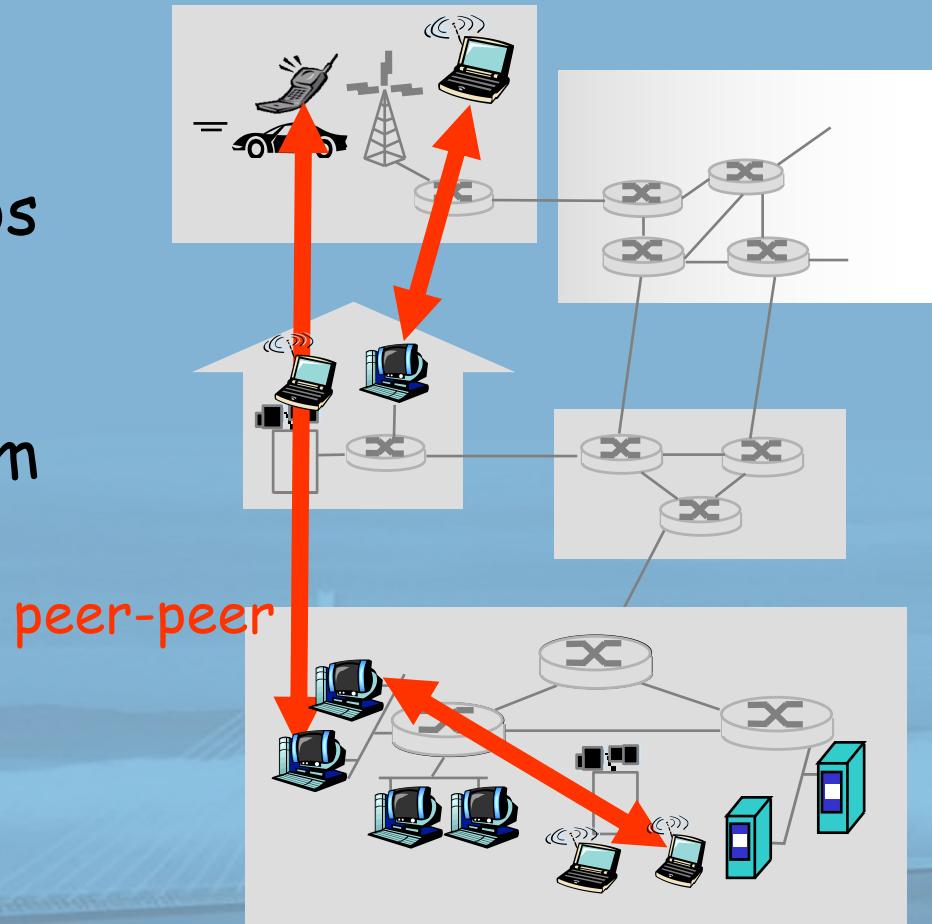
- custo estimado do centro de dados: \$600M
- Google gastou \$2,4B em 2007 em novos centros de dados
- cada centro de dados usa de 50 a 100 megawatts de potência



# Arquitetura P2P pura

- *nenhum servidor sempre ligado*
- *sistemas finais arbitrários se comunicam diretamente*
- *pares são conectados intermitentemente e mudam endereços IP*

**altamente escalável, mas difícil de administrar**



# Híbrido de cliente-servidor e P2P

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

## Skype

- ❖ aplicação P2P voice-over-IP P2P
- ❖ servidor centralizado: achando endereço da parte remota:
- ❖ conexão cliente-cliente: direta (não através de servidor)

## Mensagem instantânea

- ❖ bate-papo entre dois usuários é P2P
- ❖ serviço centralizado: detecção/localização da presença do cliente
  - ? usuário registra seu endereço IP com servidor central quando entra on-line
  - ? usuário contacta servidor central para descobrir endereços IP dos parceiros

# Processos se comunicando

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

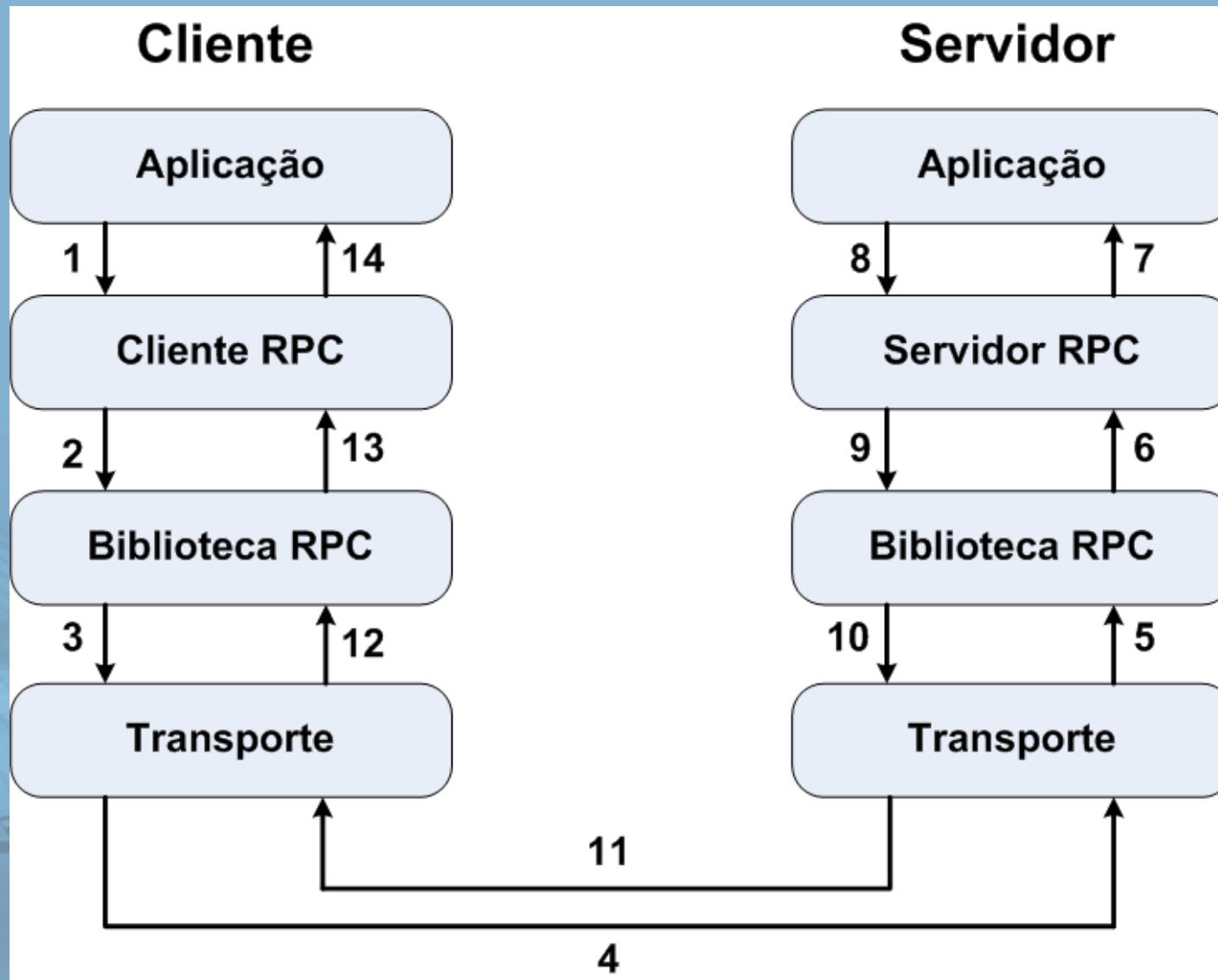
- processo: programa rodando dentro de um hospedeiro
  - no mesmo hospedeiro, dois processos se comunicam usando a **comunicação entre processos** (definida pelo SO).
  - processos em hospedeiros diferentes se comunicam trocando **mensagens**

- processo cliente:  
processo que inicia a comunicação
- processo servidor:  
processo que espera para ser contactado
- Nota: aplicações com arquiteturas P2P têm processos clientes & processos servidores

# Chamada Remota de Procedimento (Remote Procedure Call - RPC)

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*



# Sockets

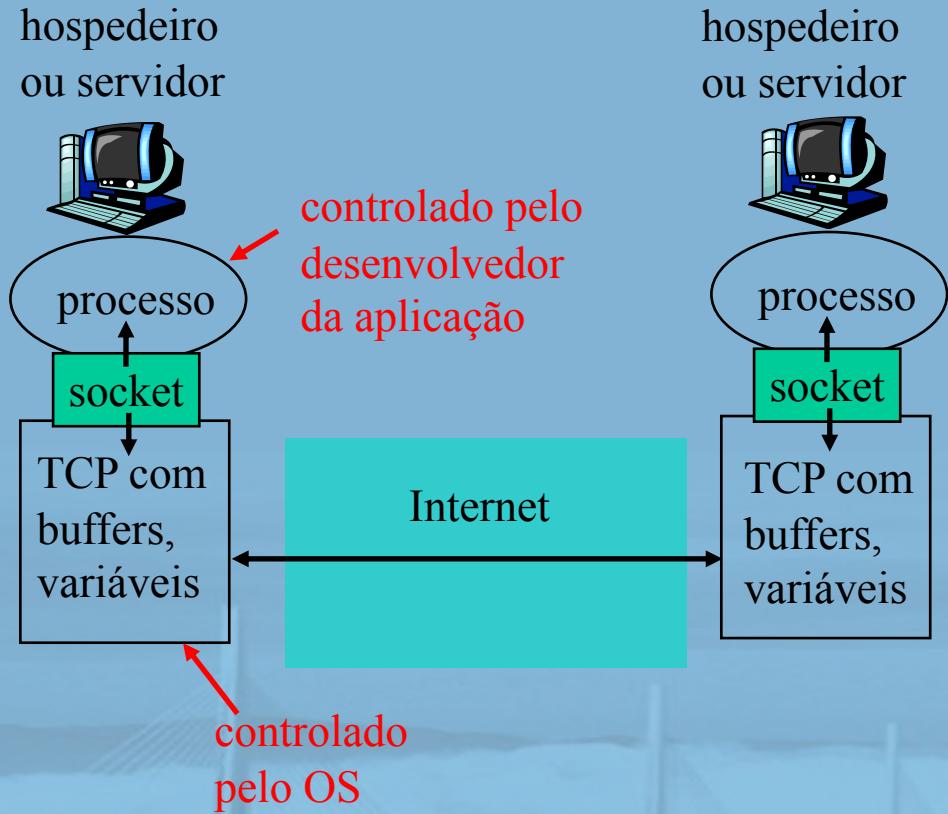
REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

- Permitem que pares de processos troquem dados estabelecendo canais diretos de comunicação bidirecional
- Usados principalmente para comunicação bidirecional entre vários processos em sistemas diferentes
  - ❖ mas podem ser usados para processos no mesmo sistema.
- Armazenados internamente como arquivos.
- O nome do arquivo é usado como endereço do socket, que é acessado por meio do VFS.

# Sockets

- processo envia/recebe mensagens de/para seu **socket**
- socket semelhante à porta
  - ❖ processo enviando empurra mensagem pela porta
  - ❖ processo enviando conta com infraestrutura de transporte no outro lado da porta, que leva a mensagem ao socket no processo receptor
- API: (1) escolha do protocolo de transporte; (2) capacidade de consertar alguns parâmetros (**muito mais sobre isso adiante**)



# Endereçando processos

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

Uma Abordagem Top-Down

para receber mensagens, processo deve ter *identificador*

dispositivo hospedeiro tem endereço IP exclusivo de 32 bits

exercício: use ipconfig do comando prompt para obter seu endereço IP (Windows)

*P:* Basta o endereço IP do hospedeiro em que o processo é executado para identificar o processo?

❖ *R:* Não, muitos processos podem estar rodando no mesmo hospedeiro

□ *Identificador* inclui *endereço IP* e *números de porta* associados ao processo no hospedeiro.

□ Exemplos de número de porta:

❖ servidor HTTP: 80

❖ servidor de correio: 25

# Definições de protocolo da camada de aplicação

- tipos de mensagens trocadas,
  - ❖ p. e., requisição, resposta
- sintaxe da mensagem:
  - ❖ que campos nas mensagens & como os campos são delineados
- semântica da mensagem
  - ❖ significado da informação nos campos
- regras de quando e como processos enviam & respondem a mensagens

REDES DE COMPUTADORES E A INTERNET 5<sup>a</sup> edição

*Uma Abordagem Top-Down*

**protocolos de domínio público:**

definidos em RFCs  
provê interoperabilidade  
p. e., HTTP, SMTP, BitTorrent

**protocolos proprietários:**  
p. e., Skype, ppstream

# Que serviço de transporte uma aplicação precisa?

## perda de dados

- algumas apls. (p. e., áudio) podem tolerar alguma perda
- outras apls. (p. e., transferência de arquivos, telnet) exigem transferência de dados 100% confiável

## temporização

- algumas apls. (p. e., telefonia na Internet jogos interativos) exigem pouco atraso para serem "eficazes"

## vazão

- algumas apls. (p. e., multimídia) exigem um mínimo de vazão para serem "eficazes"
- outras apls. ("apls. elásticas") utilizam qualquer vazão que receberem

## segurança

- criptografia, integridade de dados,...

REDES DE COMPUTADORES E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

# Requisitos de serviço de transporte das aplicações comuns

REDES DE COMPUTADORES E A INTERNET 5<sup>a</sup> edição

*Uma Abordagem Top-Down*

Aplicação	Perda de dados	Vazão	Sensível ao tempo
transf. arquivos	sem perda	elástica	não
e-mail	sem perda	elástica	não
documentos Web	sem perda	elástica	não
áudio/vídeo tempo real	tolerante a perda	áudio: 5 kbps-1 Mbps vídeo: 10 kbps-5 Mbps	sim, centenas de ms
áudio/vídeo armazenado	tolerante a perda	o mesmo que antes	sim, alguns seg
jogos interativos	tolerante a perda	poucos kbps ou mais	sim, centenas de ms
Mensagem instantânea	sem perda	elástica	sim e não

# Serviços de protocolos de transporte da Internet

REDES DE COMPUTADORES E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

## serviço TCP:

- ❑ *orientado a conexão*: preparação exigida entre processos cliente e servidor
- ❑ *transporte confiável* entre processo emissor e receptor
- ❑ *controle de fluxo*: emissor não sobrecarrega receptor
- ❑ *controle de congestionamento*: regula emissor quando a rede está sobrecarregada
- ❑ *não oferece*: temporização, garantias mínimas de vazão, segurança

## serviço UDP:

- transferência de dados não confiável entre processo emissor e receptor
- não oferece: preparação da conexão, confiabilidade, controle de fluxo, controle de congest., temporização, garantia de vazão ou segurança

P: por que se incomodar? Por que existe um UDP?

# Aplicações da Internet: aplicação, protocolos de transporte

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

Aplicação	Protocolo da camada de aplicação	Protocolo de transporte básico
e-mail	SMTP [RFC 2821]	TCP
acesso remoto	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
transf. arquivos	FTP [RFC 959]	TCP
multimídia com fluxo contínuo	HTTP (p. e., Youtube), RTP [RFC 1889]	TCP ou UDP
telefonia da Internet	SIP, RTP, proprietário (p. e., Skype)	normalmente UDP

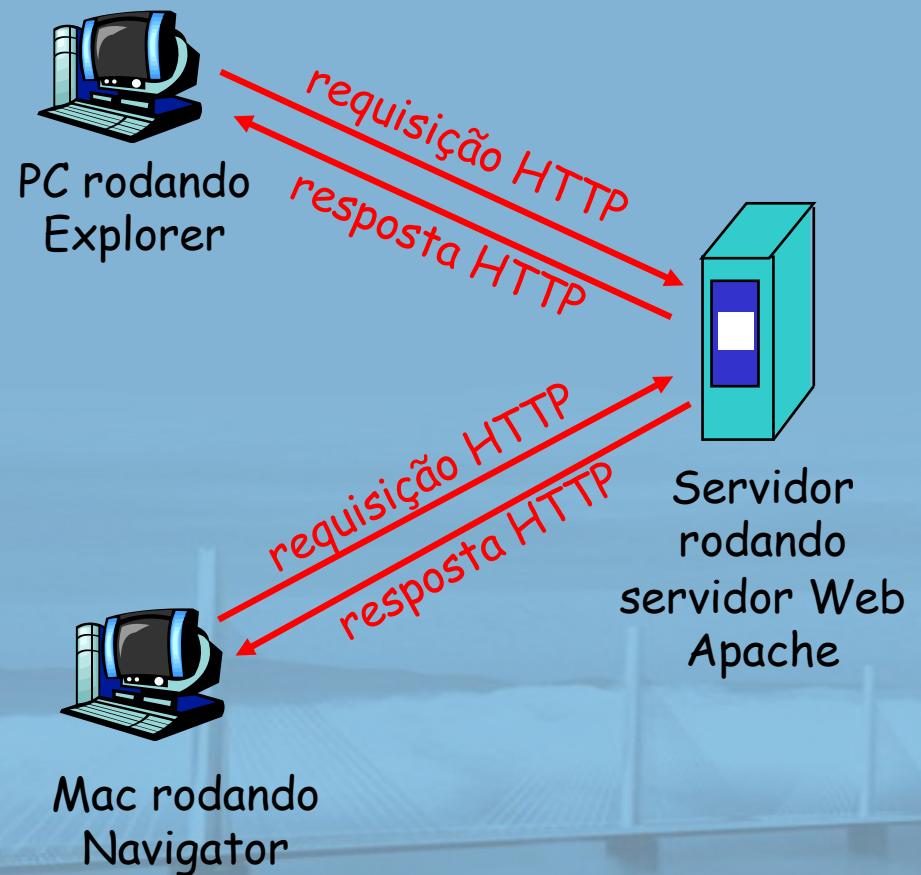
# Visão geral do HTTP

## 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

REDES DE COMPUTADORES E A INTERNET 5<sup>a</sup> edição

*Uma Abordagem Top-Down*



**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

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

## 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

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

(contém texto,  
referências a 10  
imagens JPEG)

Suponha que o usuário digite o URL

www.someSchool.edu/someDepartment/home.index

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

tempo ↓

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

4. Servidor HTTP fecha conexão TCP.

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

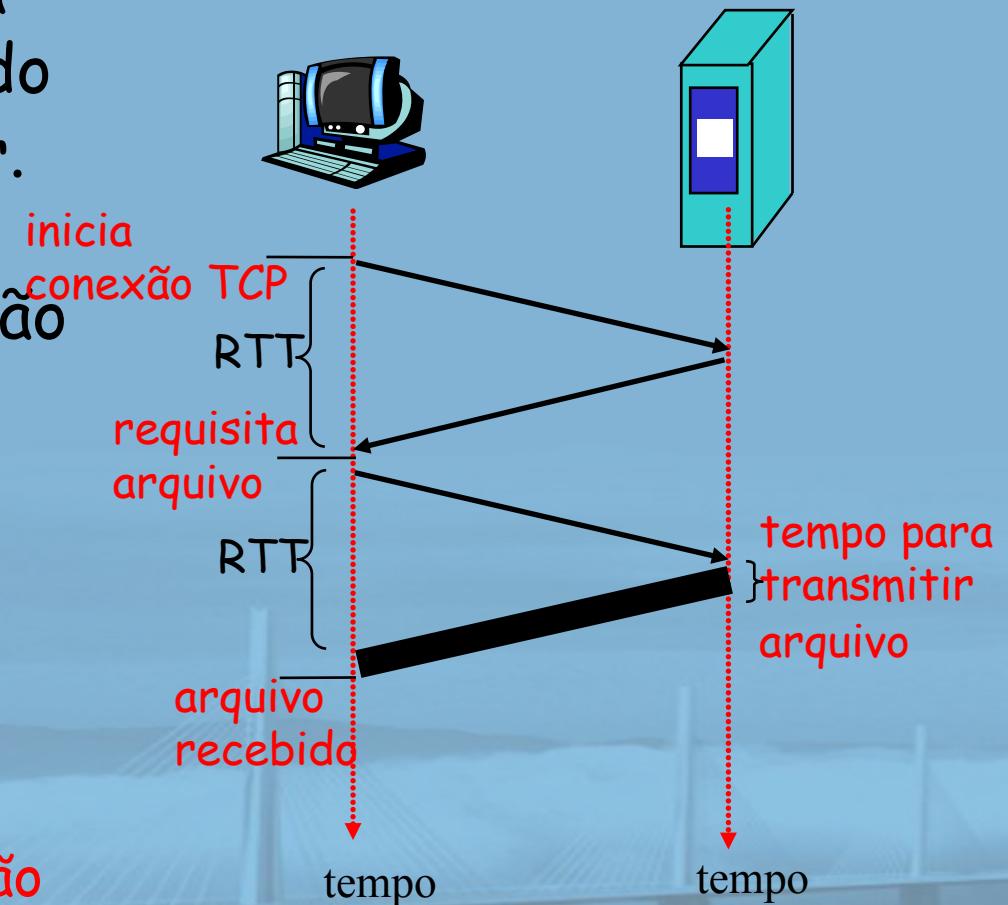
# HTTP não persistente: tempo de resposta

**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

$$\text{total} = 2\text{RTT} + \text{tempo de transmissão}$$



# HTTP persistente

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

## 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:

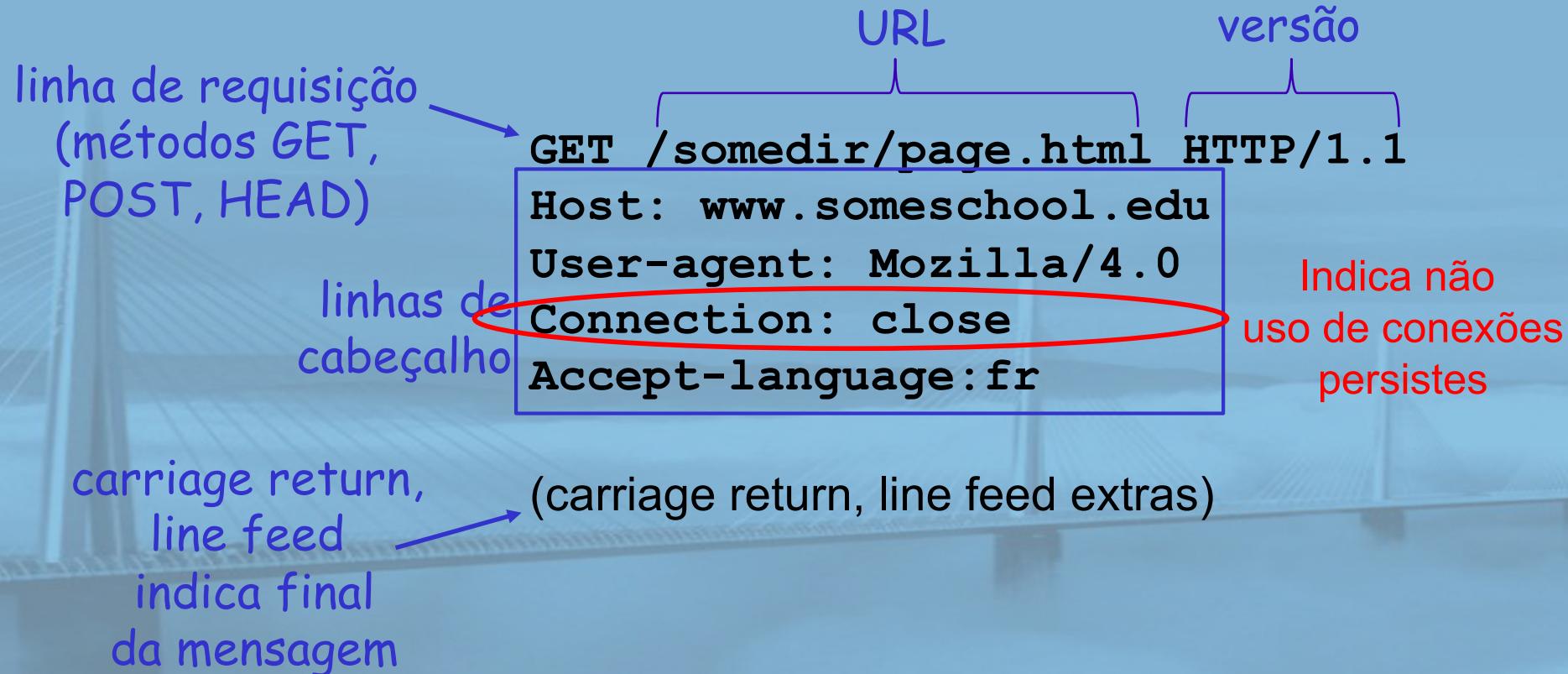
servidor deixa a conexão aberta depois de enviar a resposta mensagens HTTP seguintes entre cliente/servidor enviadas pela conexão aberta cliente envia requisições assim que encontra um objeto referenciado no mínimo um RTT para todos os objetos referenciados

# Mensagem de requisição HTTP

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

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



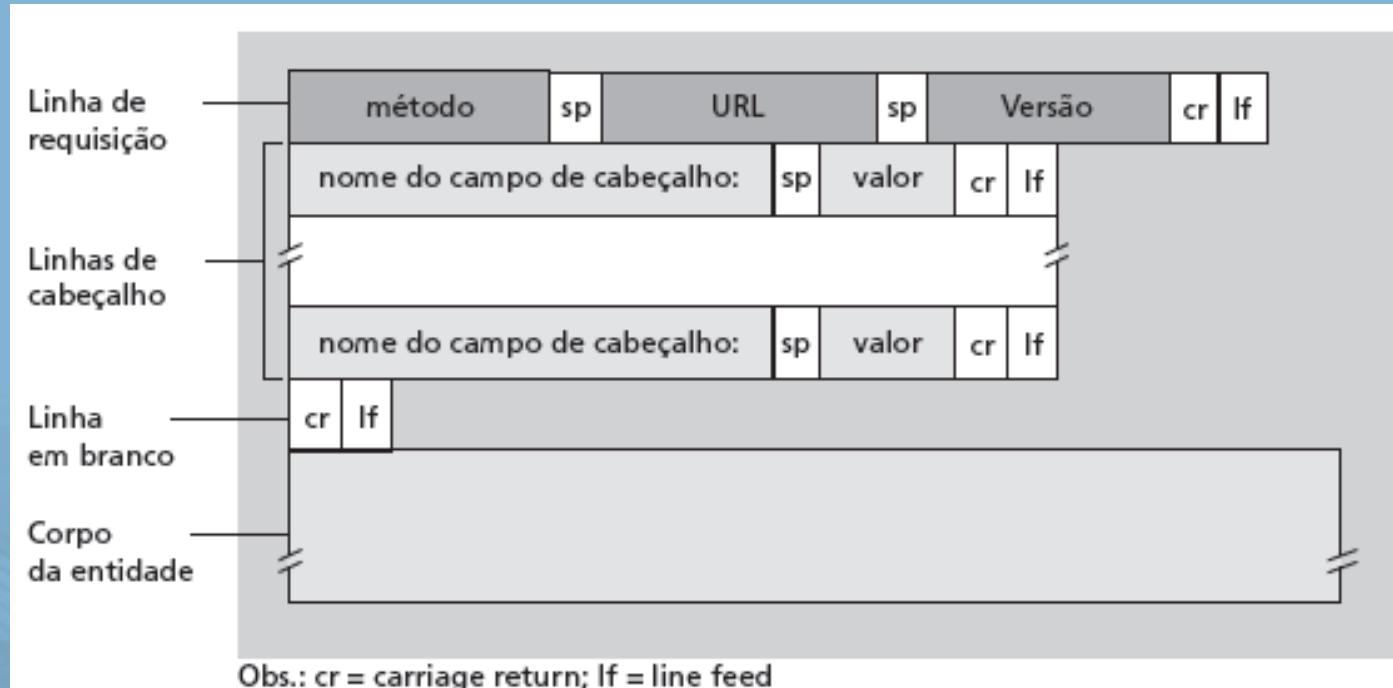
# Mensagem de requisição

## HTTP: formato geral

REDES DE  
COMPUTADORES  
E A INTERNET

5<sup>a</sup> edição

*Uma Abordagem Top-Down*



# Upload da entrada do formulário

REDES DE COMPUTADORES E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

## 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

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

## 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

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

linha de status  
(protocolo  
código de estado  
frase de estado)

linhas de  
cabeçalho

dados, p. e.,  
arquivo HTML  
requisitado

HTTP/1.1 200 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

*dados dados dados dados dados ...*

# Códigos de estado da resposta HTTP

REDES DE COMPUTADORES E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

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

REDES DE  
COMPUTADORES  
E A INTERNET

5<sup>a</sup> edição

*Uma Abordagem Top-Down*

## 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 [RFC 6265]

REDES DE  
COMPUTADORES  
E A INTERNET

5<sup>a</sup> edição

*Uma Abordagem Top-Down*

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

# Tarefa Prática

## Programação de rede com Socket

REDES DE  
COMPUTADORES  
E A INTERNET

5<sup>a</sup> edição

*Uma Abordagem Top-Down*

- Desenvolver uma aplicação de software no modelo cliente/servidor que se comuniquem com base no protocolo HTTP:
  - ❖ Cabe ao grupo a responsabilidade de definir sua própria aplicação
  - ❖ Obrigatório uso da API Socket (em qualquer linguagem, de preferência Java)

# Tarefa Prática

## Programação de rede com Socket

REDES DE  
COMPUTADORES  
E A INTERNET

5<sup>a</sup> edição

*Uma Abordagem Top-Down*

### **Informação aos Membros dos Grupos:**

Grupo: Mínimo 4, máximo 6. **Grupos fora dessa composição não terão trabalho aceito.**

Cada grupo deverá eleger um líder e este fará um relatório sobre o desempenho individual de cada membro da equipe, a ser apresentado no momento da avaliação dos resultados do trabalho

### **Critérios de avaliação:**

- Funcionamento da aplicação de acordo com o proposto (50%)
- Postura, linguagem, domínio do conteúdo, clareza e objetividade durante a exposição (20%)
- Criatividade e complexidade da aplicação (20%)
- Respeito ao tempo pré-determinado
- Critérios complementares (opcionais): assiduidade e participação em sala de aula.

# Aplicações IP: Partes

REDES DE  
COMPUTADORES  
E A INTERNET  
5<sup>a</sup> edição

*Uma Abordagem Top-Down*

