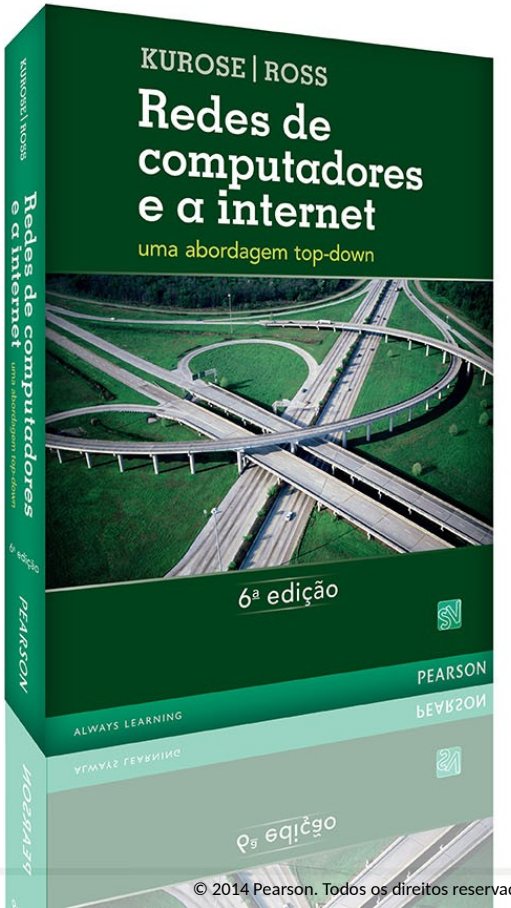
Capítulo 2

Camada de aplicação



Algumas aplicações de rede

- HTTP - web
- e-mail
- DNS
- 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 nuvem

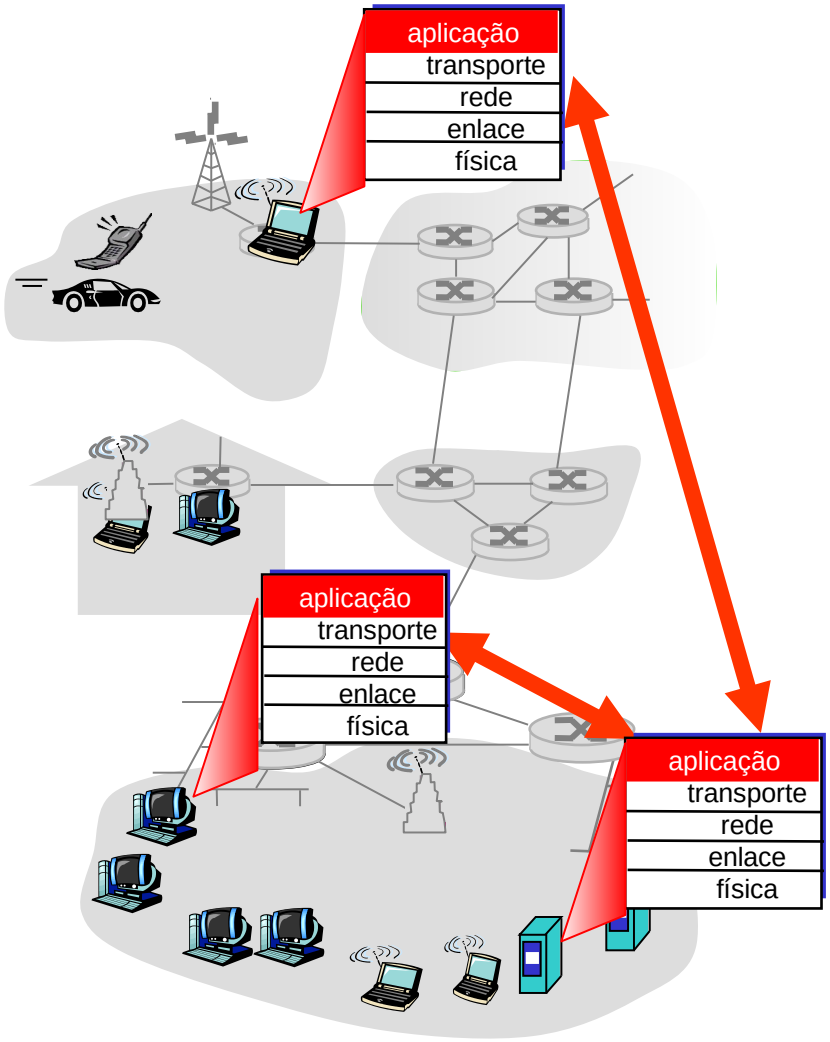
Criando uma aplicação de rede

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

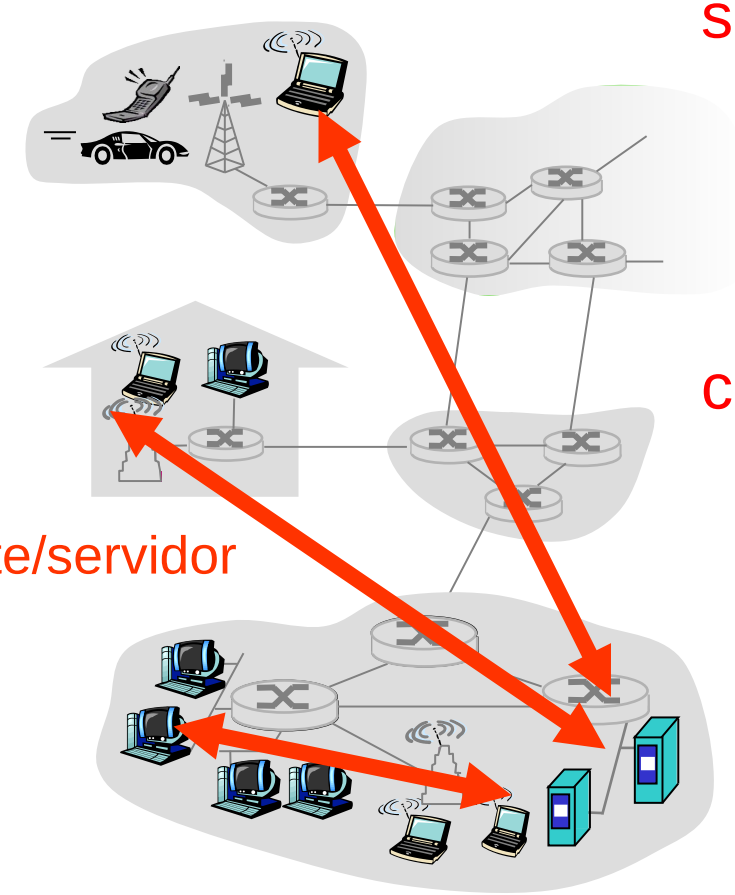
- dispositivos do núcleo da rede não executam aplicações do usuário
- as aplicações nos sistemas finais permitem rápido desenvolvimento e propagação



Arquiteturas de aplicação de rede

- A **arquitetura de rede** é fixa e provê um conjunto específico de serviços.
- A **arquitetura da aplicação** é projetada pelo programador e determina como a aplicação é organizada nos vários sistemas finais.
- Em uma **arquitetura cliente-servidor** há um hospedeiro sempre em funcionamento, denominado *servidor* (**datacenter**), que atende a requisições de muitos outros hospedeiros, denominados *clientes*. *Clientes* não se comunicam diretamente.

Arquitetura cliente-servidor



cliente/servidor

servidor:

- hospedeiro sempre ligado
- endereço IP permanente
- *server farms* para expansão

clientes:

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

Centros de dados da Google

- custo estimado do centro de dados: $600M
- Google gastou $2,4B em 2007 em novos centros de dados
- 2014: 1 centro de dados € 600 milhões - Holanda
- cada centro de dados consome de 50 a 100 megawatts de potência

Arquiteturas de aplicação de rede

- A **arquitetura P2P** utiliza a comunicação direta entre duplas de hospedeiros conectados alternadamente, denominados *pares*. Skype, BitTorrent, IPTV etc.

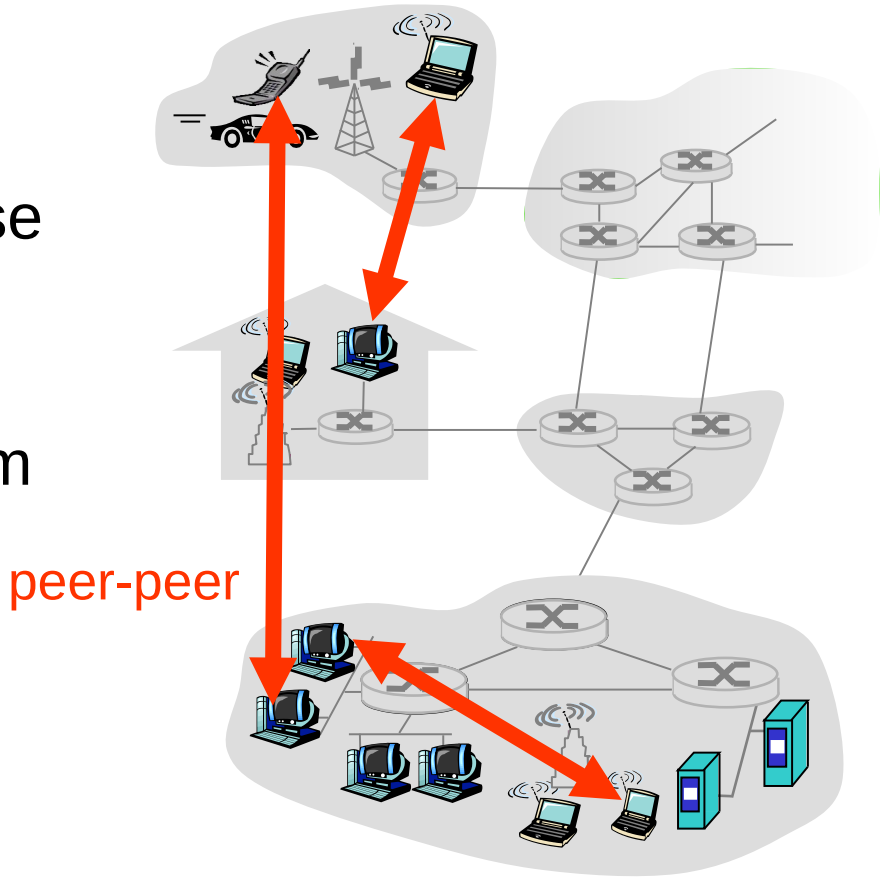
- Uma das características mais fortes da arquitetura P2P é sua **auto escalabilidade**.

- As futuras aplicações P2P estão diante de três principais desafios:

1. ISP Amigável.
2. Segurança.
3. Incentivos.

Arquitetura P2P pura

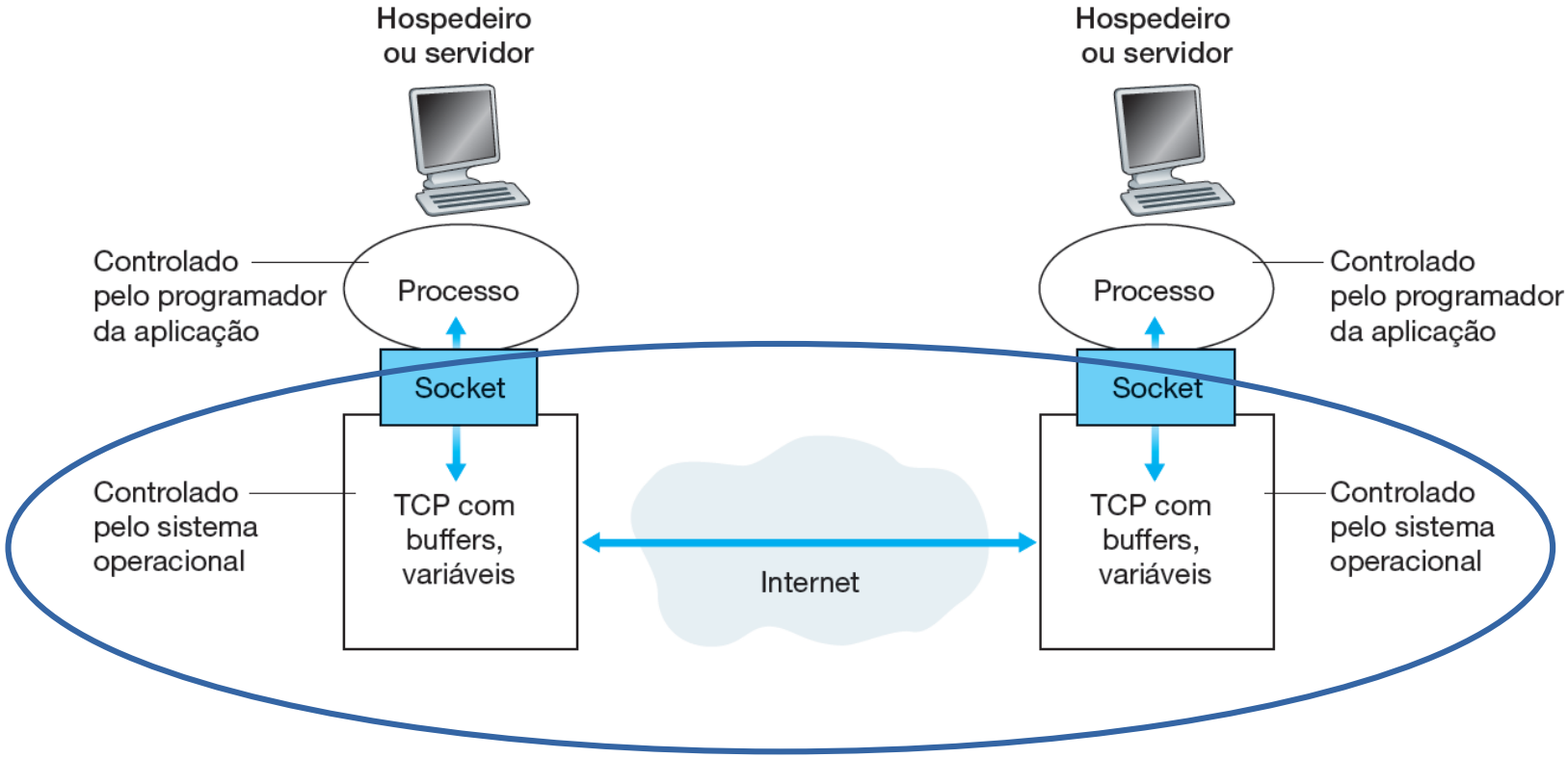
- *nenhum* servidor sempre ligado
- sistemas finais arbitrários se comunicam diretamente
- pares são conectados intermitentemente e mudam endereços IP



- altamente escalável, mas difícil de administrar

Comunicação entre processos

- Processos de aplicação (possivelmente em SOs diferentes), *sockets* e protocolo de transporte subjacente.
- Interface de programação da aplicação – API.



Comunicação entre processos

- Uma aplicação de rede consiste em pares de processos que enviam mensagens uns para os outros por meio de uma rede.
- Um processo envia mensagens para a rede e recebe mensagens dela através de uma interface de software denominada ***socket***.
- Para identificar o processo receptor, duas informações devem ser especificadas:

1. o endereço do hospedeiro (IP) e
2. um identificador que especifica o processo receptor no hospedeiro de destino (porta).

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,…

Serviços de transporte providos pela Internet

- A Internet disponibiliza dois protocolos de transporte para aplicações, o UDP e o TCP.
- 1920x1080p em 30fps: 28 Mbps. Netflix: 4 Mbps. Televisão digital: 6 Mbps.
- Requisitos de algumas aplicações de rede:

Aplicação	Perda de dados	Vazão	Sensibilidade ao tempo
Transferência / download de arquivo	Sem perda	Elástica	Não
E-mail	Sem perda	Elástica	Não
Documentos Web	Sem perda	Elástica (alguns kbits/s)	Não
Telefonia via Internet/ videoconferência	Tolerante à perda	Áudio: alguns kbits/s – 1Mbit/s Vídeo: 10 kbits/s – 5 Mbits/s	Sim: décimos de segundo
Áudio/vídeo armazenado	Tolerante à perda	Igual acima	Sim: alguns segundos
Jogos interativos	Tolerante à perda	Poucos kbits/s – 10 kbits/s	Sim: décimos de segundo
Mensagem instantânea	Sem perda	Elástico	Sim e não

Serviços de protocolos de transporte da Internet

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 congestionamento, temporização, garantia de vazão ou segurança
- Muito mais leve, ágil e rápido

Serviços de transporte providos pela Internet

- Aplicações populares da Internet, seus protocolos de camada de aplicação e seus protocolos de transporte subjacentes:

Aplicação	Protocolo de camada de aplicação	Protocolo de transporte subjacente
Correio eletrônico	SMTP [RFC 5321]	TCP
Acesso a terminal remoto	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Transferência de arquivos	FTP [RFC 959]	TCP
Multimídia em fluxo contínuo	HTTP (por exemplo, YouTube)	TCP
Telefonia por Internet	SIP [RFC 3261], RTP [RFC 3550] ou proprietária (por exemplo, Skype)	UDP ou TCP

Protocolos de camada de aplicação

Um protocolo de camada de aplicação define:

- Os tipos de mensagens trocadas.
- A sintaxe dos vários tipos de mensagens, tais como os campos da mensagem e como os campos são delineados.
- A semântica dos campos, isto é, o significado da informação nos campos.
- Regras para determinar quando e como um processo envia mensagens e responde a mensagens.

A Web e o HTTP

- Talvez o que mais atraia a maioria dos usuários da Web é que ela funciona por demanda.

- O HTTP — Protocolo de Transferência de Hipertexto (*HyperText Transfer Protocol*) —, o protocolo da camada de aplicação da Web, está no coração da Web e é definido no [RFC 1945] e no [RFC 2616].

- O HTTP é executado em dois programas:

1. um cliente e
2. outro servidor.

A Web e o HTTP

- Uma página Web é constituída de objetos.
- Um objeto é apenas um arquivo que se pode acessar com um único URL (*Uniform Resource Locator*). Exemplo….
- A maioria das páginas Web é constituída de um arquivo-base HTML e diversos objetos referenciados. Ver index.html odilson… (http://redes.sj.ifsc.edu.br/Redes_arq3.html)
- O HTTP usa o TCP como seu protocolo de transporte subjacente.
- O HTTP é denominado um protocolo sem estado.

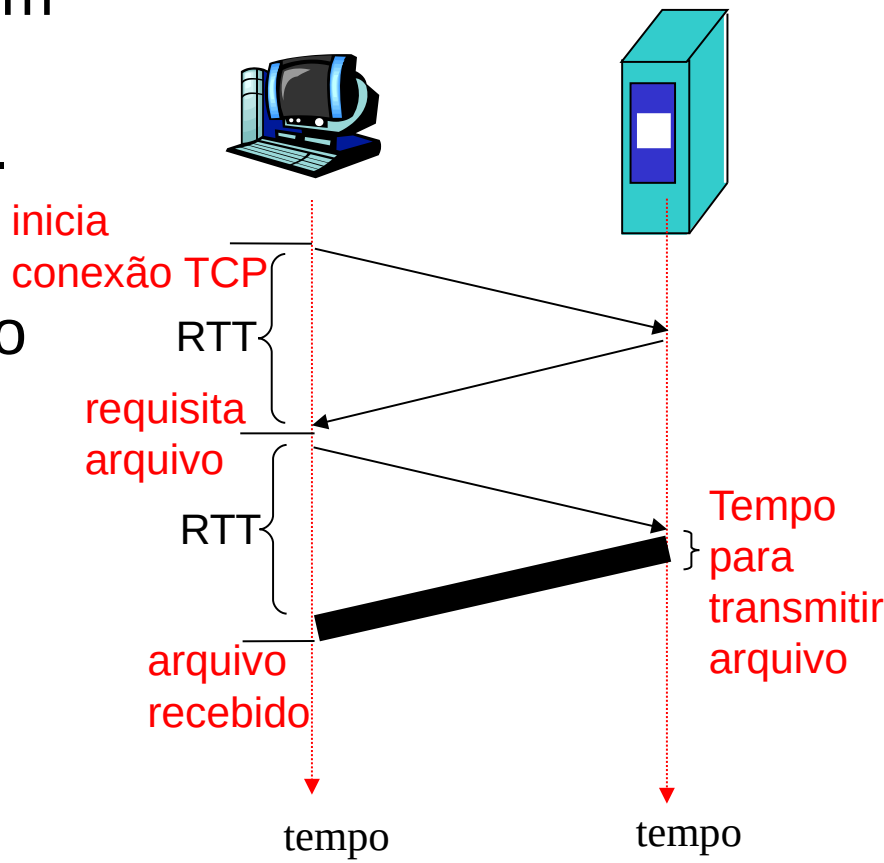
HTTP não persistente: tempo de resposta

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



Conexões persistentes e não persistentes

- Quando a interação cliente-servidor acontece por meio de conexão TCP, o programador da aplicação precisa tomar uma importante decisão:

- **Conexões não persistentes** — cada par de requisição/resposta deve ser enviado por uma conexão TCP distinta.

- **Conexões persistentes** — todas as requisições e suas respostas devem ser enviadas por uma mesma conexão TCP.

HTTP não persistente

Suponha que o usuário digite o URL

`www.someSchool.edu/someDepartment/home.index`

(contém texto, referências a 10 imagens JPEG)

Redes de Computadores

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

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.

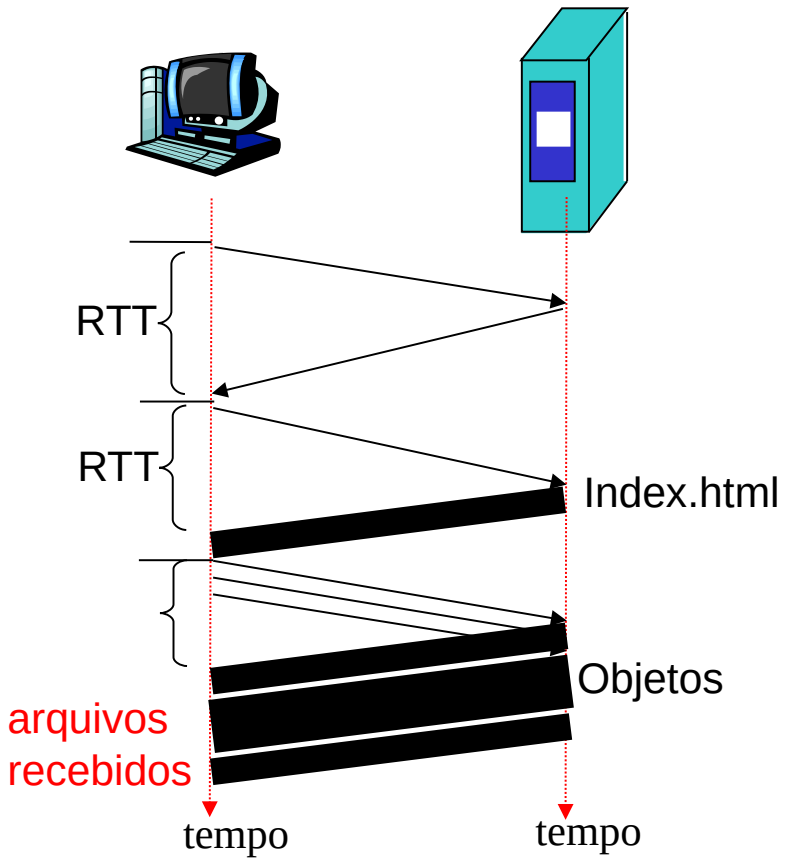
6. Etapas 1-5 repetidas para cada um dos 10 objetos JPEG.

Portanto, o tempo total para acesso a página é 20 RTT mais o tempo de transmissão dos 10 objetos

problemas do HTTP não persistente:

- requer 2 RTTs por objeto
- overhead do SO para *cada* conexão TCP

HTTP persistente

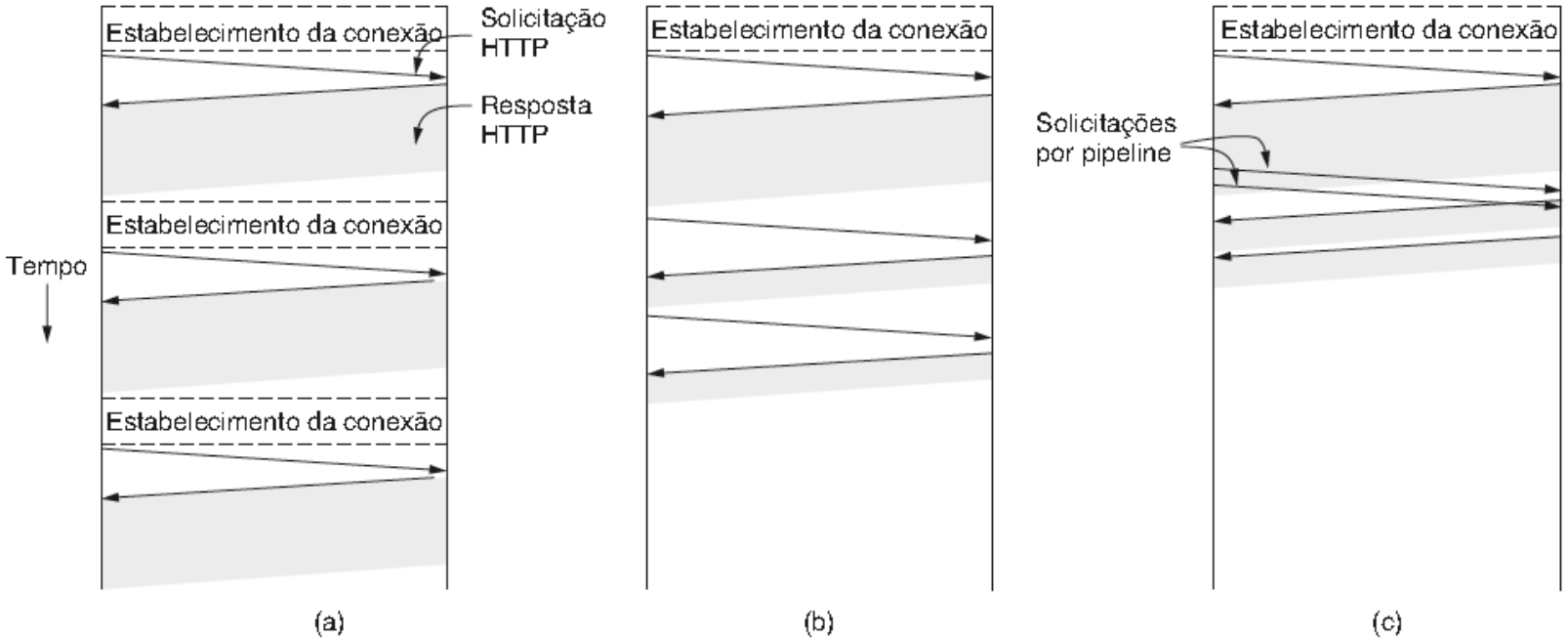


HTTP persistente:

- 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 dois RTT para todos os objetos referenciados
- Modo default: conexões persistentes com paralelismo (os navegadores modernos por padrão abrem de 5 a 10 conexões paralelas…)

Protocolo de Transferência de Hipertexto

- Navegadores geralmente abrem conexões TCP paralelas para buscar objetos referenciados



HTTP com:

(a) Múltiplas conexões e solicitações sequenciais – Não persistente.

(b) Conexão persistente e solicitações sequenciais.

(c) Conexão persistente com solicitações com paralelismo.

Formato da mensagem HTTP

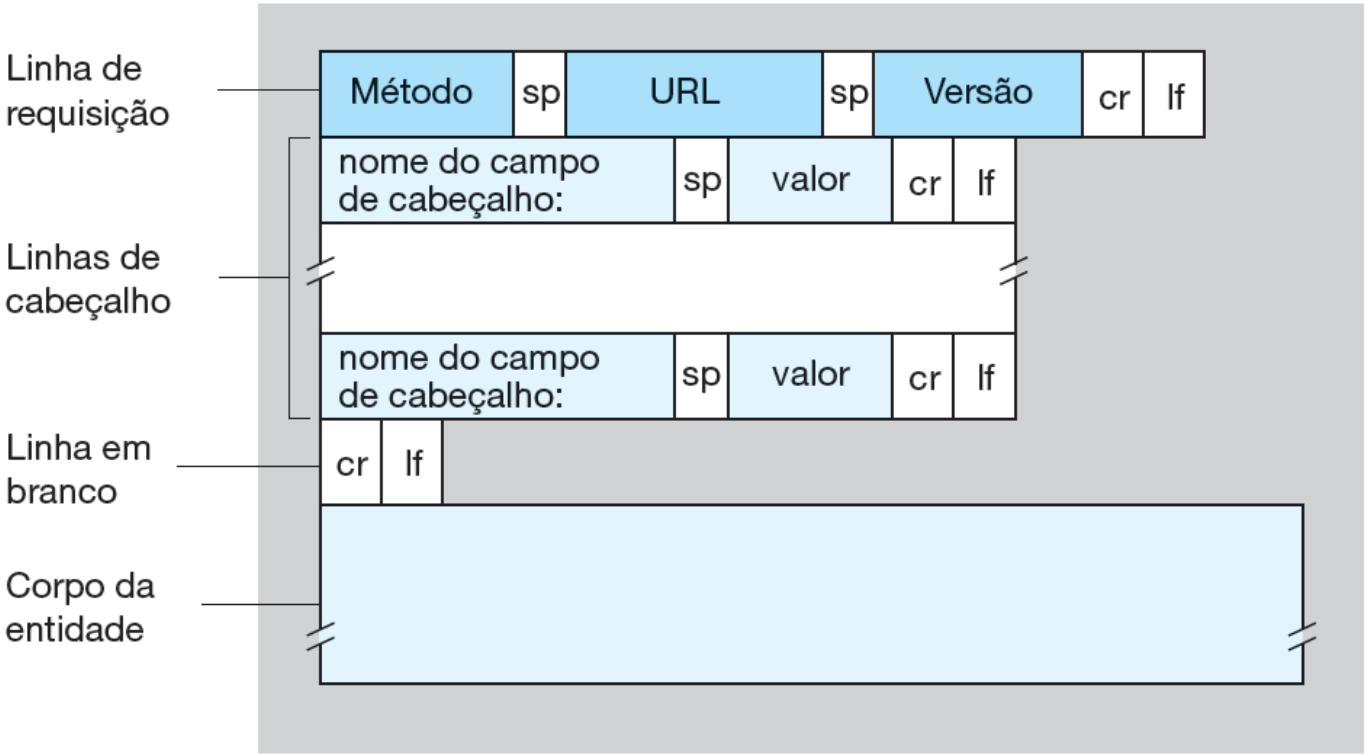
Mensagem de requisição HTTP

- Apresentamos a seguir uma mensagem de requisição HTTP típica:

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

Formato da mensagem HTTP

- Formato geral de uma mensagem de requisição HTTP



Formato da mensagem HTTP

Mensagem de resposta HTTP

- Apresentamos a seguir uma mensagem de resposta HTTP típica:

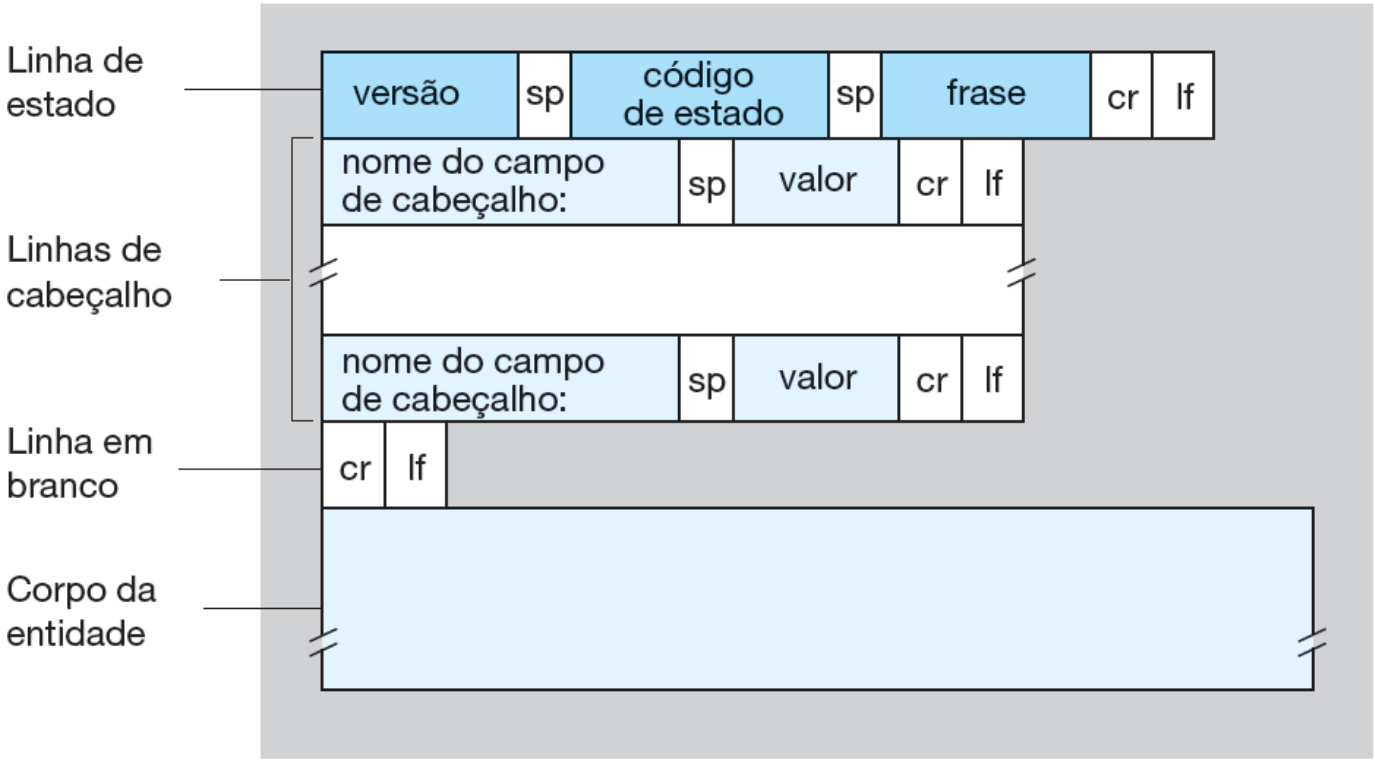
HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

(dados dados dados dados dados ...)

200 OK
201 Criado
…..
301 Movido
304 Não modificado
400 Requisição inválida
401 Não autorizado
402 Pagamento necessário
404 Não encontrado
…….

Formato da mensagem HTTP

- Formato geral de uma mensagem de resposta HTTP



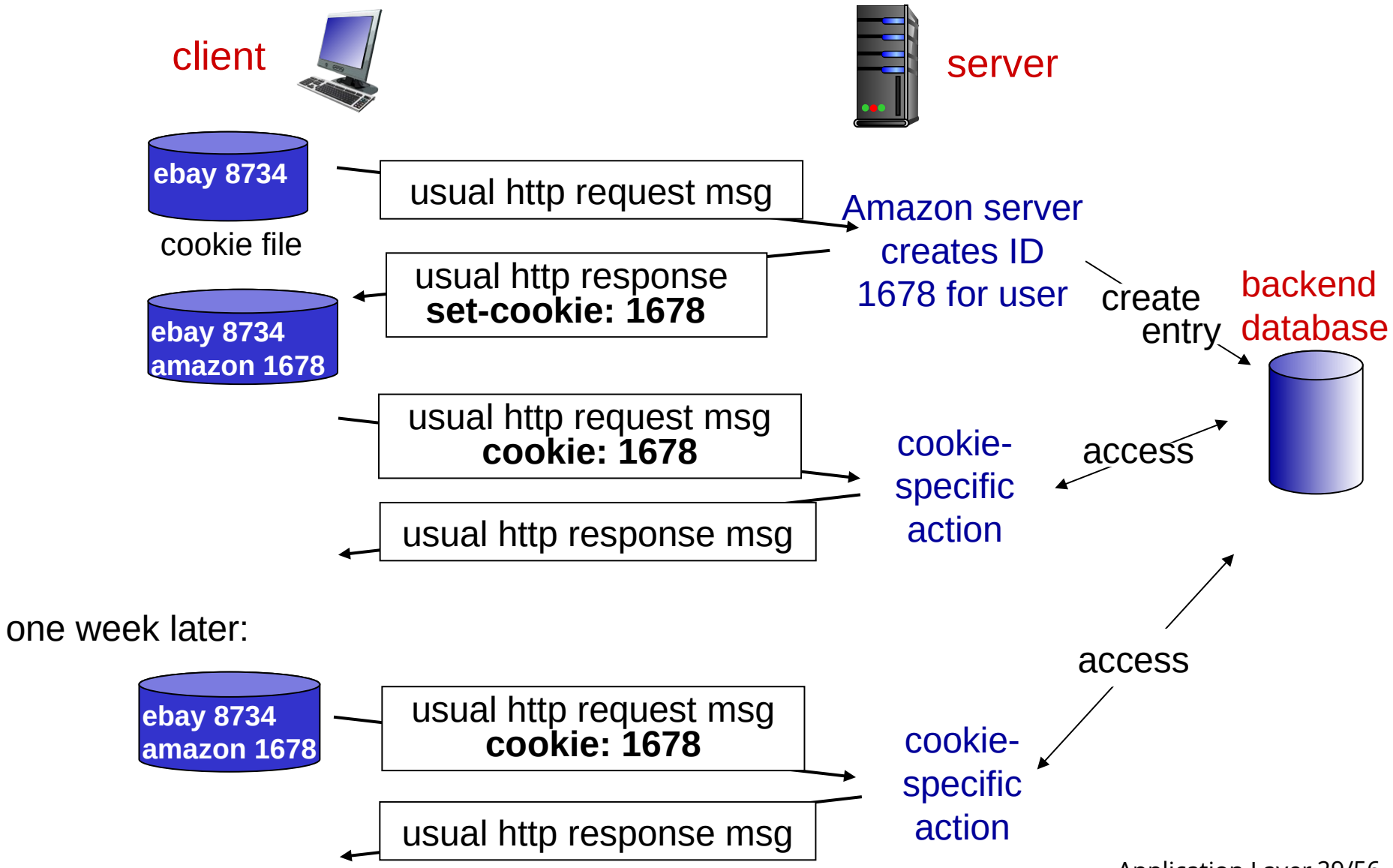
Interação usuário-servidor: *cookies*

Cookies, definidos no [RFC 6265], permitem que sites “monitorem” seus usuários através de um sistema de **estado** que permite, entre outras coisas, salvar navegações realizadas pelo “usuário”. No slide 19 é dito que o HTTP é um protocolo sem **estado**.

A tecnologia dos *cookies* tem quatro componentes:

1. uma linha de cabeçalho de *cookie* na mensagem de resposta HTTP;
2. uma linha de cabeçalho de *cookie* na mensagem de requisição HTTP;
3. um arquivo de *cookie* mantido no sistema final do usuário e gerenciado pelo navegador do usuário;
4. um banco de dados de apoio no site.

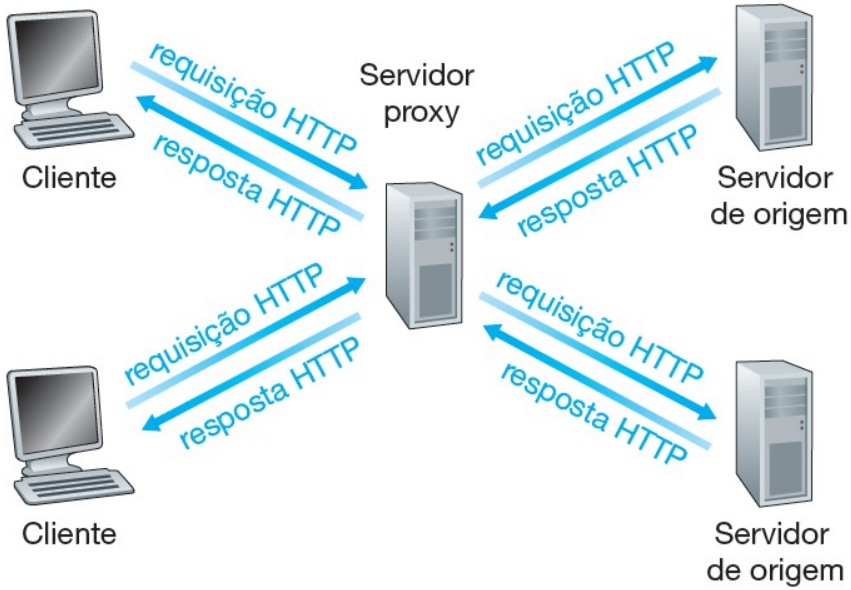
Cookies: keeping "state" (cont.)



Caches Web

- Um ***cache* Web** — também denominado **servidor *proxy*** — é uma entidade da rede que atende requisições HTTP em nome de um servidor Web de origem.

Clientes requisitando objetos por meio de um *cache* Web:



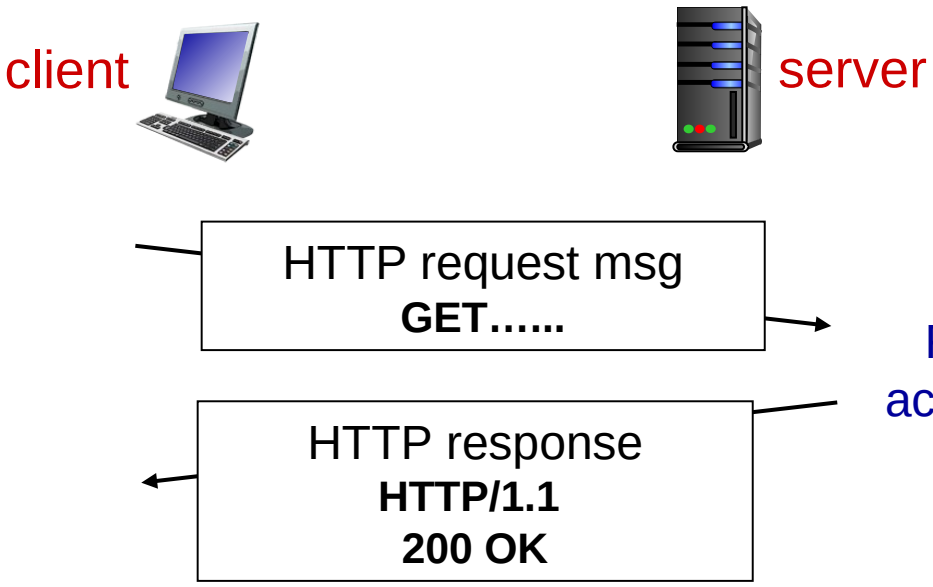
- **GET condicional** – mecanismo que permite que um *cache* verifique se seus objetos estão atualizados.

Conditional GET

- *Goal:* don't send object if cache has up-to-date cached version
  - no object transmission delay
  - lower link utilization
- *cache:* specify date of cached copy in HTTP request

  `If-modified-since: <date>`
- *server:* response contains no object if cached copy is up-to-date:

  `HTTP/1.1 304 Not Modified`

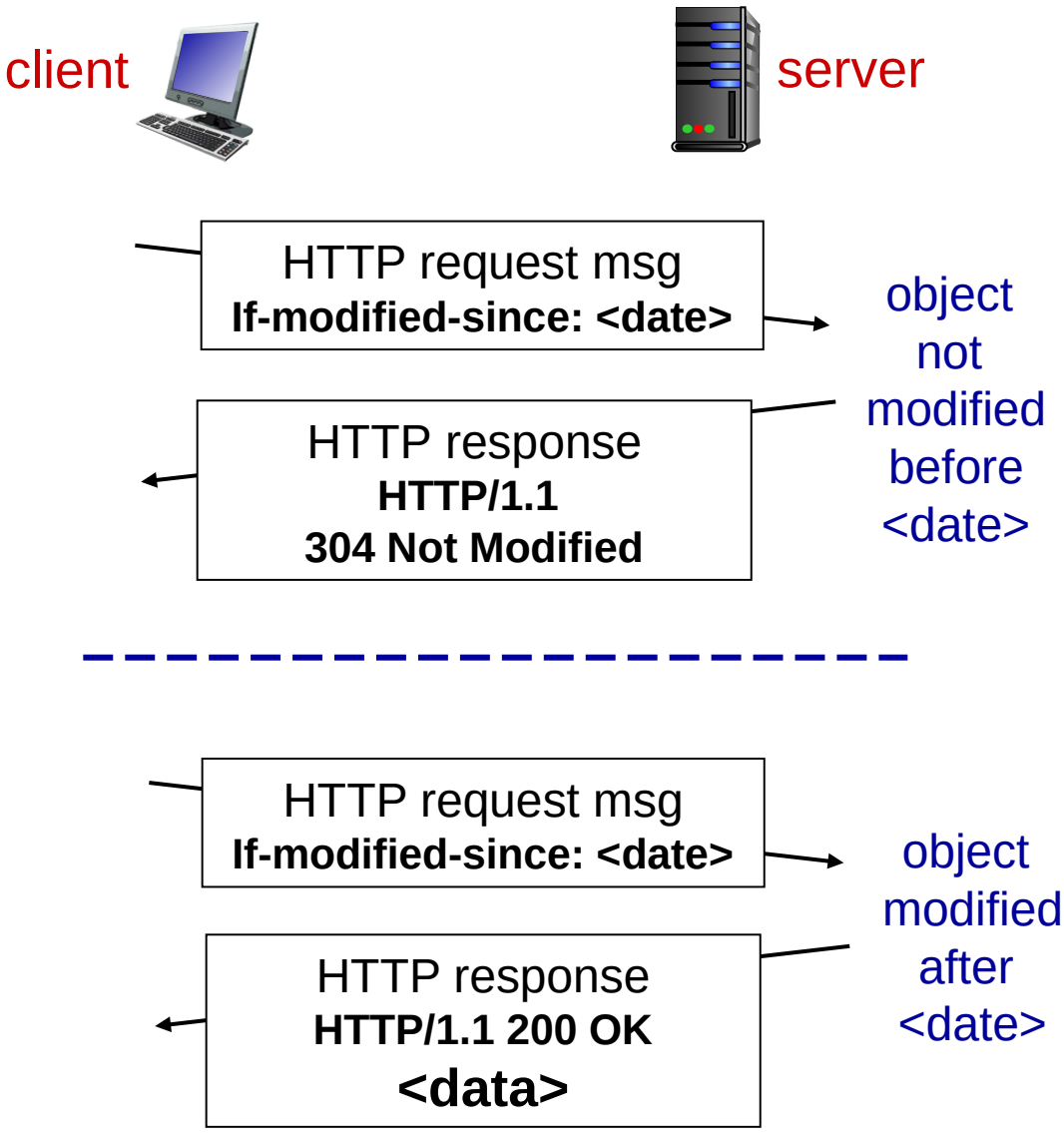


Conditional GET

- *Goal:* don't send object if cache has up-to-date cached version
  - no object transmission delay
  - lower link utilization
- *cache:* specify date of cached copy in HTTP request

  `If-modified-since: <date>`
- *server:* response contains no object if cached copy is up-to-date:

  `HTTP/1.1 304 Not Modified`

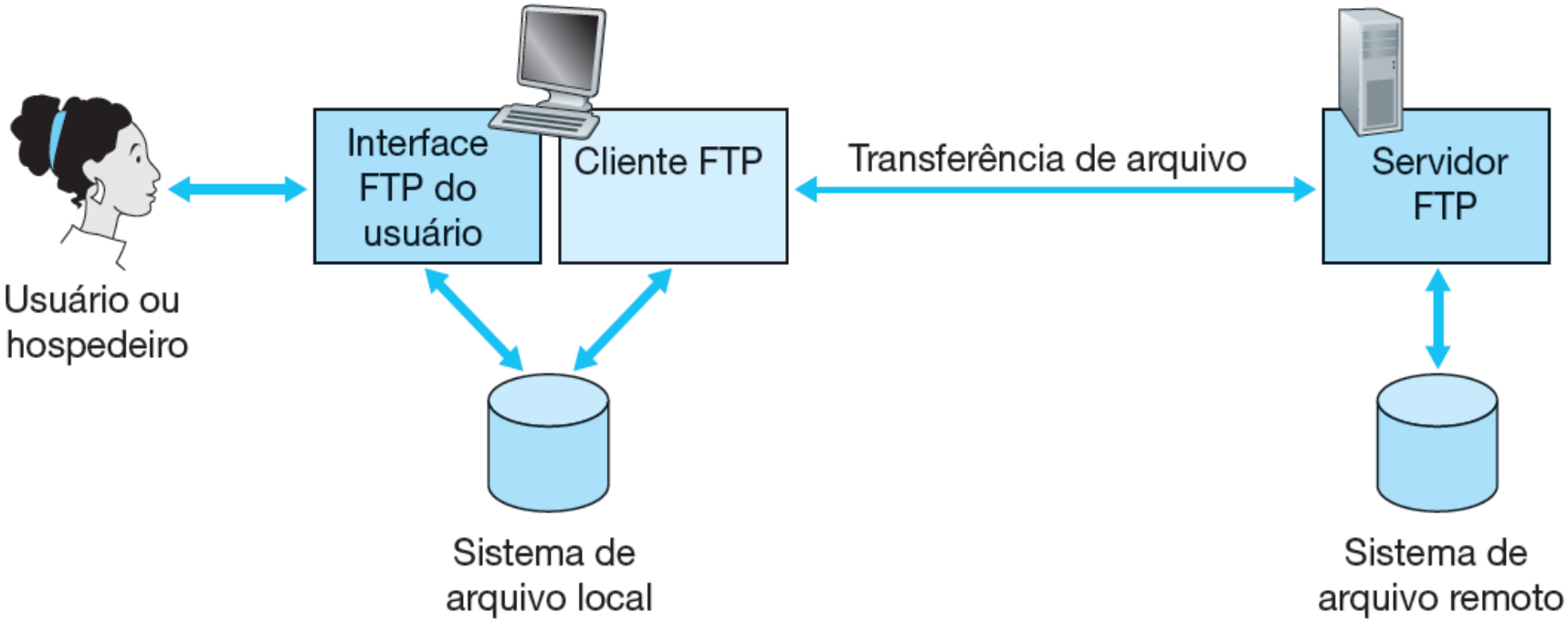


Transferência de arquivo: FTP

- Em uma sessão FTP típica, o usuário quer transferir arquivos de ou para um hospedeiro remoto.
- HTTP e FTP são protocolos de transferência de arquivos e têm muitas características em comum.

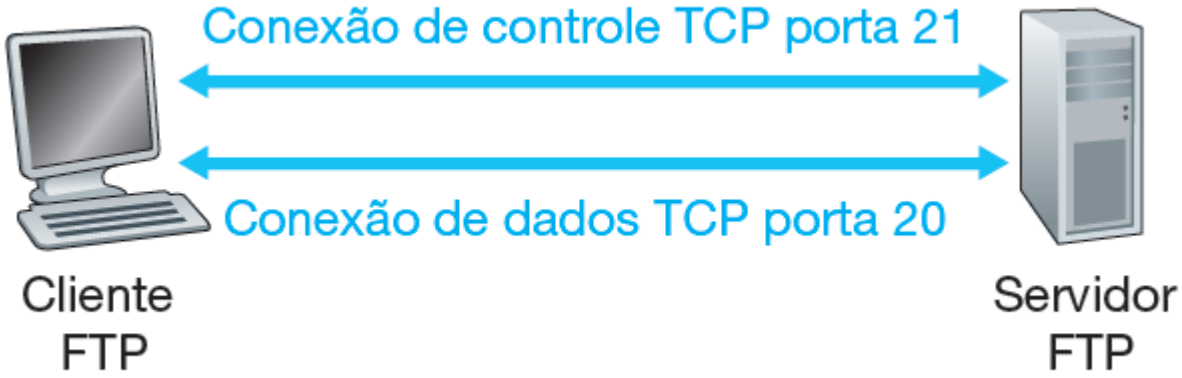
Transferência de arquivo: FTP

- FTP transporta arquivos entre sistemas de arquivo local e remoto:

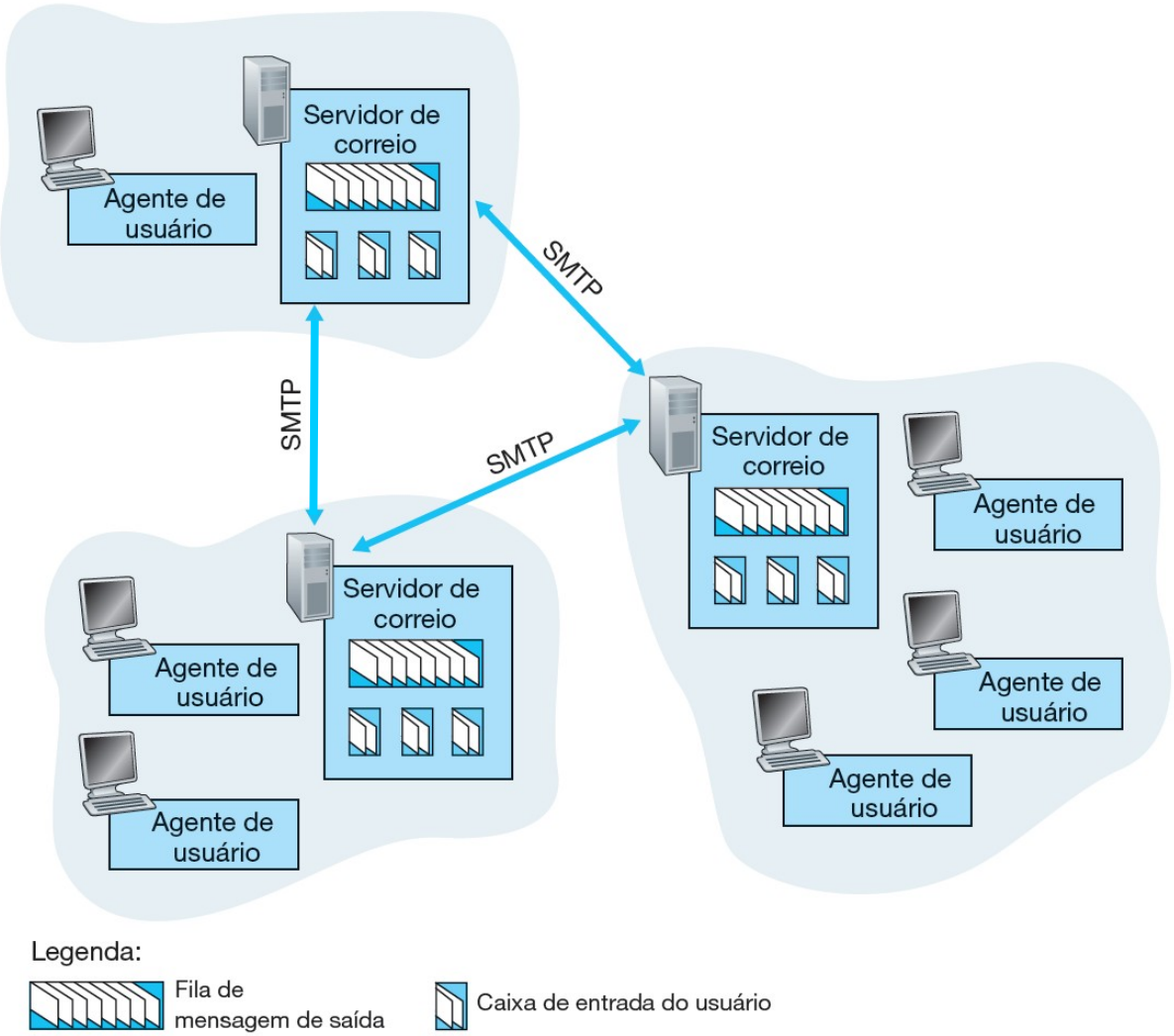


Transferência de arquivo: FTP

- Conexões de controle e de dados:



Correio eletrônico na Internet

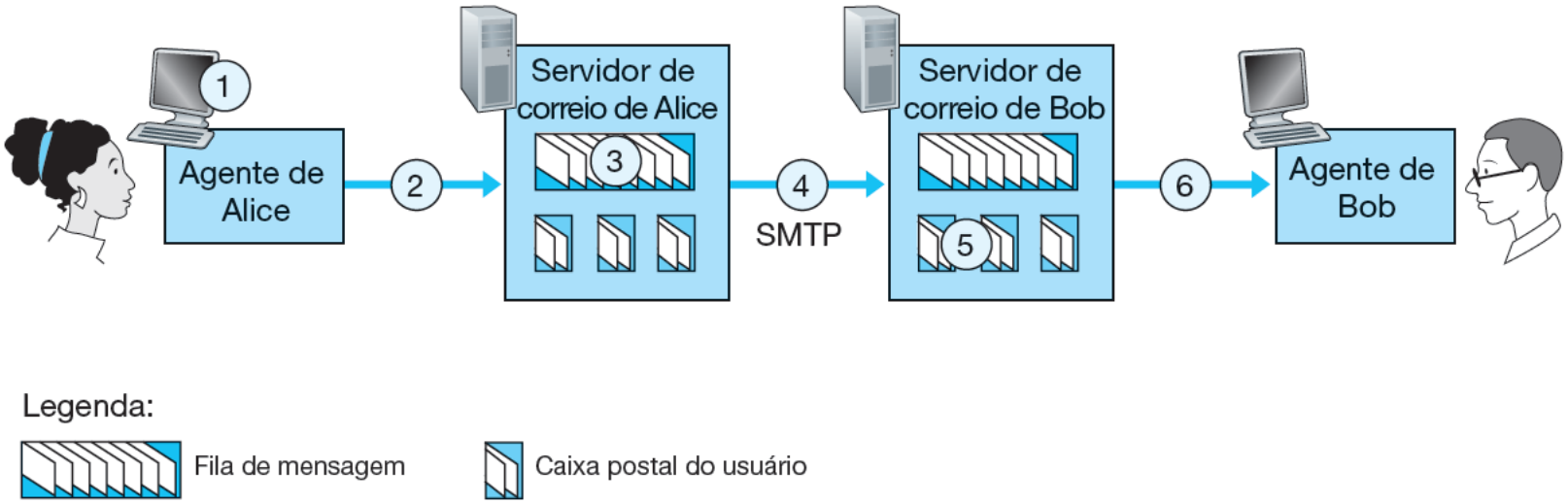


Uma visão do sistema de e-mail da Internet.

SMTP

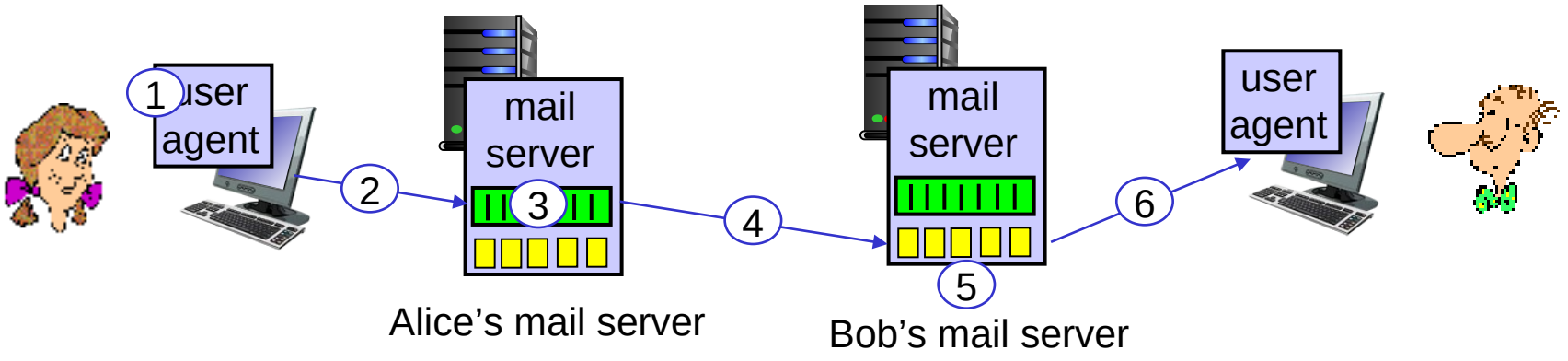
- O SMTP (*Simple Mail Transfer Protocol*) transfere mensagens de servidores de correio remetentes para servidores de correio destinatários.

Alice envia uma mensagem a Bob: Explicar 1, 2...6



Scenario: Alice sends message to Bob

1) Alice uses UA to compose message "to" `bob@someschool.edu`
2) Alice's UA sends message to her mail server; message placed in message queue
3) client side of SMTP opens TCP connection with Bob's mail server
4) SMTP client sends Alice's message over the TCP connection
5) Bob's mail server places the message in Bob's mailbox
6) Bob invokes his user agent to read message



Formatos de mensagem de correio

- Um cabeçalho de mensagem típico é semelhante a:

From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Searching for the meaning of life.

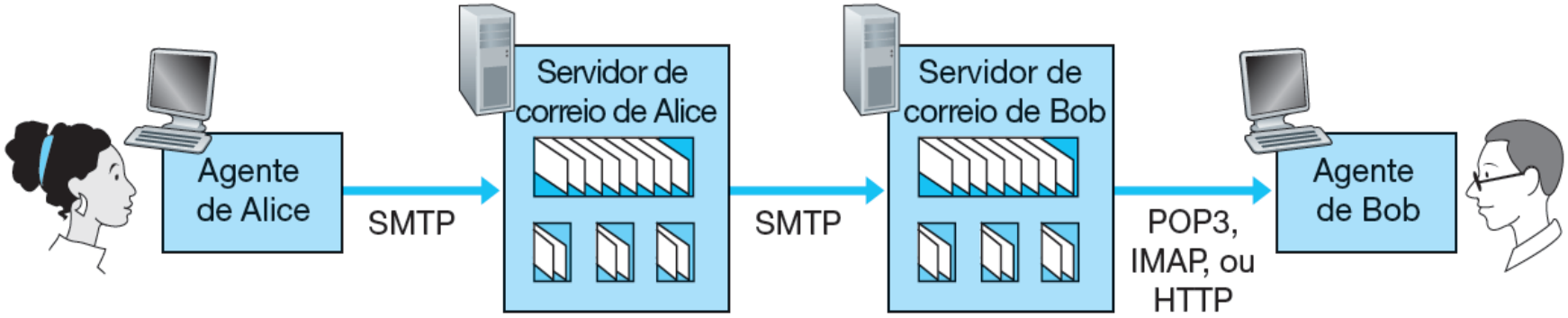
- Após o cabeçalho da mensagem, vem uma linha em branco e, em seguida, o corpo da mensagem (em ASCII).

Formatos de mensagem de correio


```
MIME-Version: 1.0
Received: by 10.60.30.197 with HTTP; Tue, 12 Apr 2016 10:26:30 -0700 (PDT)
Date: Tue, 12 Apr 2016 14:26:30 -0300
Delivered-To: odilsontv@gmail.com
Message-ID: <CAGywRsER-FHhMupoqab-YjAgqKsux+mFT6i1yeWVVtL02DWPPg@mail.gmail.com>
Subject: sdfg
From: Odilson <odilsontv@gmail.com>
To: odilson@tele.sj.ifsc.edu.br
Content-Type: multipart/alternative; boundary=bcaec54a33e43d2f9f05304cf614

--bcaec54a33e43d2f9f05304cf614
Content-Type: text/plain; charset=UTF-8

sdfg

--bcaec54a33e43d2f9f05304cf614
Content-Type: text/html; charset=UTF-8

<div dir="ltr">sdfg</div>

--bcaec54a33e43d2f9f05304cf614--
```

Protocolos de acesso ao correio

- Protocolos de e-mail e suas entidades comunicantes



DNS: o serviço de diretório da Internet

- Há duas maneiras de identificar um hospedeiro — por um nome de hospedeiro e por um endereço IP.
- Para conciliar isso, é necessário um serviço de diretório que traduza nomes de hospedeiro para endereços IP.
- Esta é a tarefa principal do DNS da Internet.
- O DNS é:
  - um banco de dados distribuído executado em uma hierarquia de servidores de DNS, e
  - um protocolo de camada de aplicação que permite que hospedeiros consultem o banco de dados distribuído.

DNS: o serviço de diretório da Internet

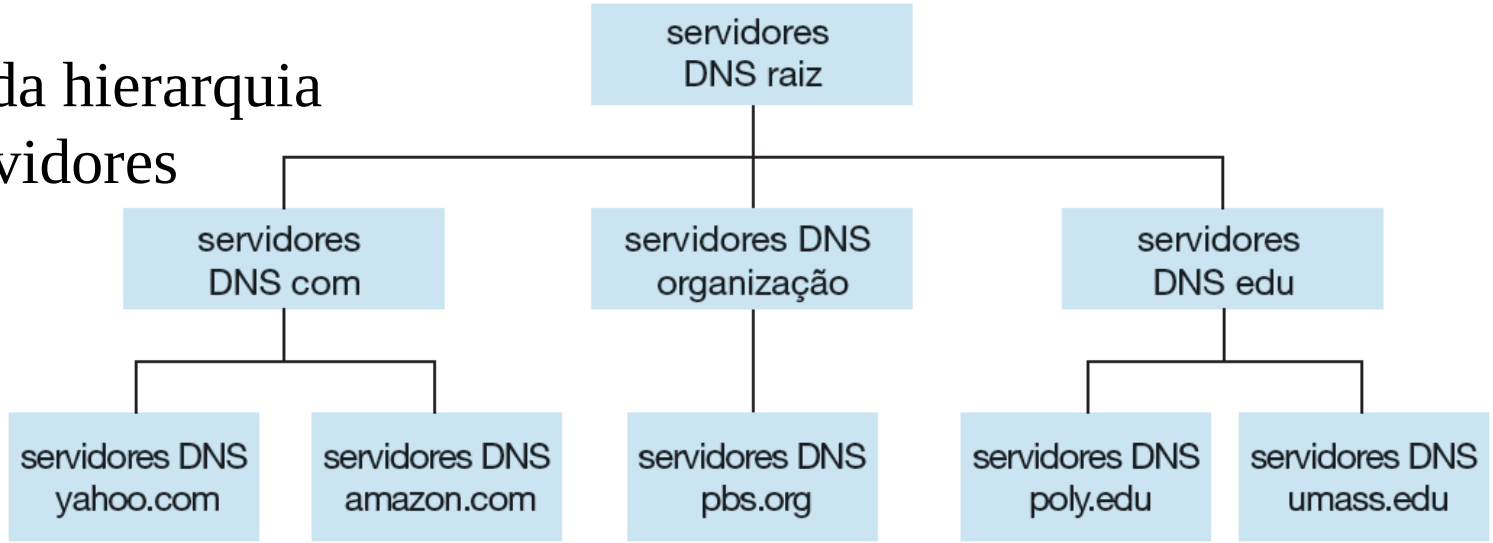
O DNS provê alguns outros serviços importantes além da tradução de nomes de hospedeiro para endereços IP:

- Apelidos (*aliasing*) de hospedeiro.
- Apelidos de servidor de correio.
- Distribuição de carga.

DNS: o serviço de diretório da Internet

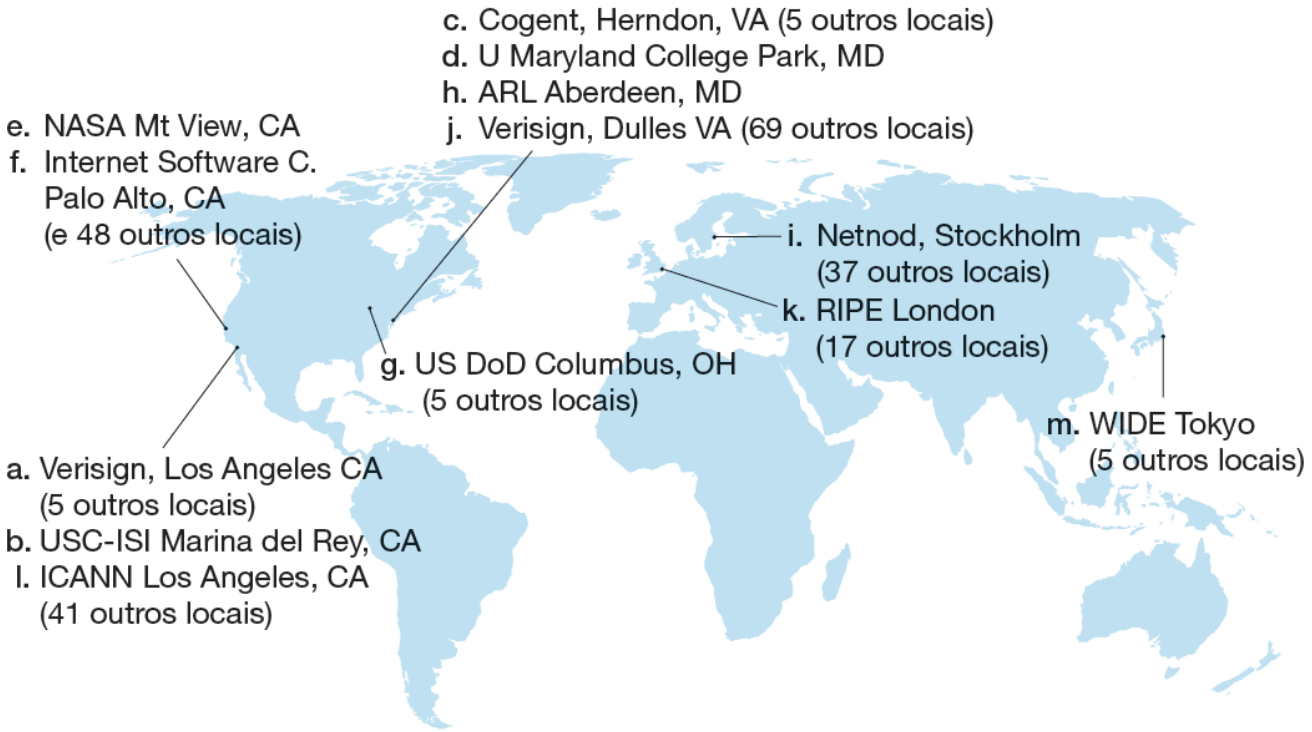
- Nenhum servidor DNS isolado tem todos os mapeamentos para todos os hospedeiros da Internet.
- Em vez disso, os mapeamentos são distribuídos pelos servidores DNS.

Parte da hierarquia de servidores DNS

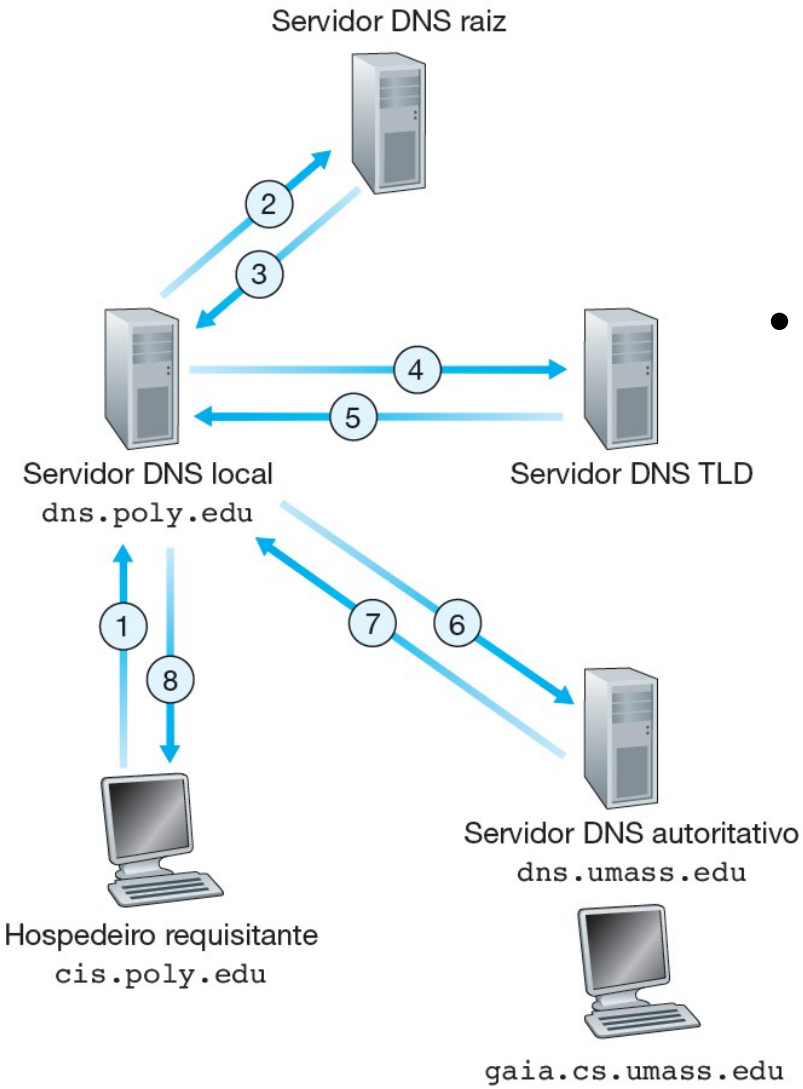


DNS: o serviço de diretório da Internet

- Servidores DNS raiz em 2012 (nome, organização, localização)



DNS: o serviço de diretório da Internet

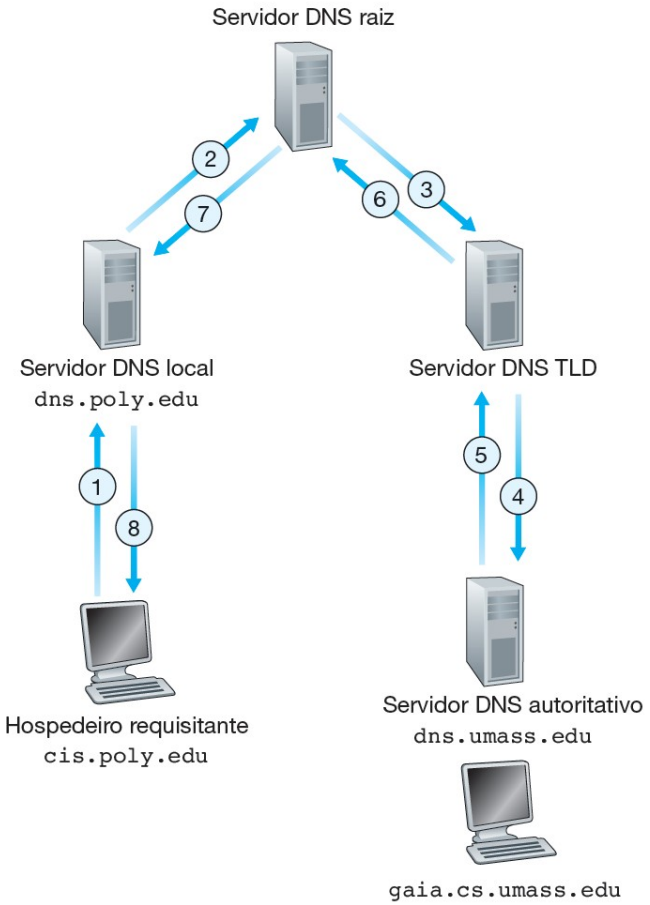


- Interação dos diversos servidores DNS:

DNS: o serviço de diretório da Internet

- O DNS explora extensivamente o *cache* para melhorar o desempenho quanto ao atraso e reduzir o número de mensagens DNS que dispara pela Internet.

- Consultas recursivas em DNS:



DNS records

DNS: distributed db storing resource records (RR)

RR format: **`(name, value, type, ttl)`**

type=A

- **`name`** is hostname
- **`value`** is IP address

type=AAAA

- **`name`** is hostname
- **`value`** is IPv6 address

type=NS

- **`name`** is domain (e.g., foo.com)
- **`value`** is hostname of authoritative name server for this domain

type=CNAME

- **`name`** is alias name for some "canonical" (the real) name
- **`www.ibm.com`** is really **`servereast.backup2.ibm.com`**
- **`value`** is canonical name

type=MX

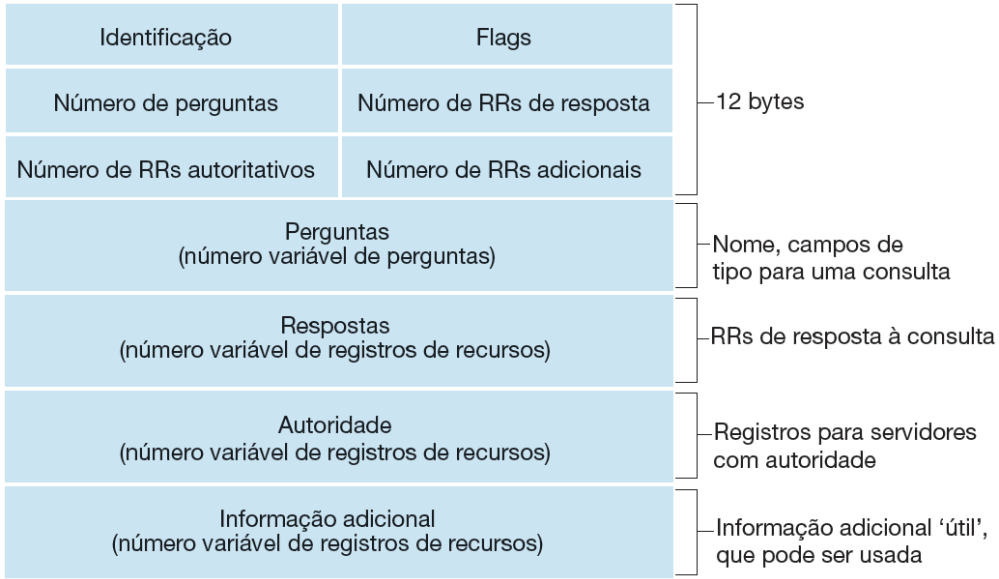
- **`value`** is name of mailserver associated with **`name`**

Registros e mensagens DNS

- Um registro de recurso é uma tupla de quatro elementos que contém os seguintes campos:

  (Name, Value, Type, TTL)

- Formato da mensagem DNS



- Dig +trace www.polito.it + WireShark

Exemplo de arquivo de configuração de SLD.

```
$TTL    86400
@   IN   SOA     ns.redes.edu.br. root (
              2016090900     ; Serial
                   604800     ; Refresh
                    86400     ; Retry
                  2419200     ; Expire
                    86400 )     ; Negative Cache TTL
;
@   IN   NS ns.redes.edu.br.
@   IN   MX 10  mail.redes.edu.br.
$ORIGIN redes.edu.br.
ns          IN  A    200.135.3.101
www     IN  A    200.135.3.102
www     IN  A    200.135.3.103
www     IN  A    200.135.3.104
www     IN  A    200.135.3.105
www     IN  A    200.135.3.106
www     IN  A    200.135.3.107
mail      IN  A    200.135.3.109
ftp         IN  CNAME mail.redes.edu.br.
```

Arquitetura P2P

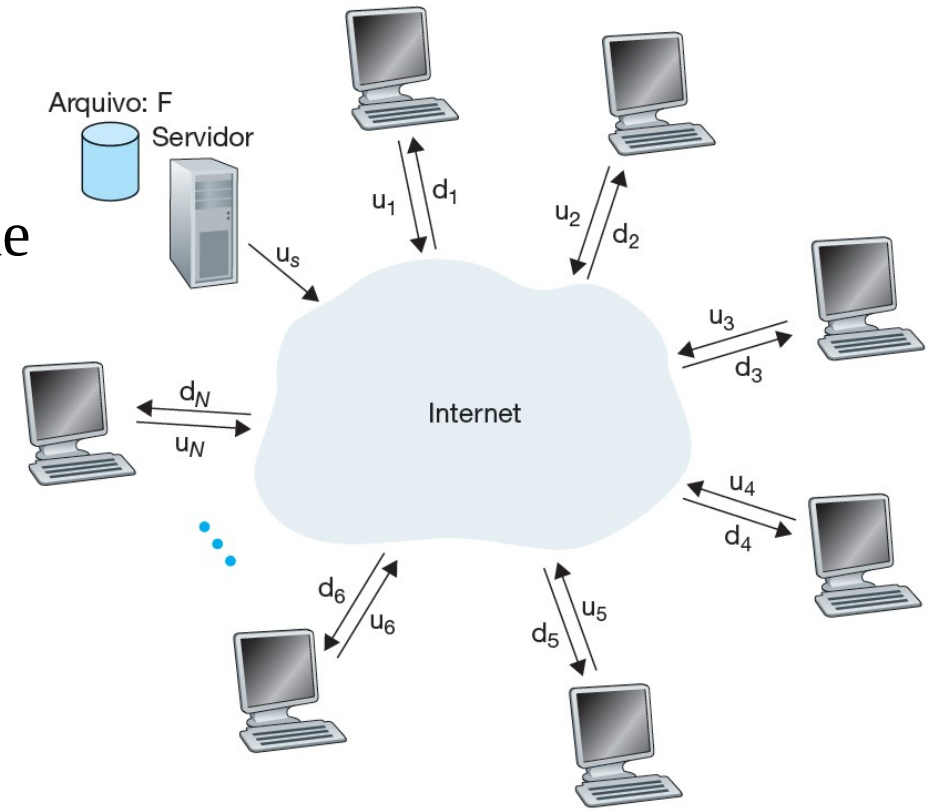
Distribuição de arquivos P2P

- Na distribuição de arquivos P2P, cada par pode redistribuir qualquer parte do arquivo recebido para outros pares, auxiliando, assim, o servidor no processo de distribuição.
- O tempo de distribuição é o tempo necessário para que todos os *N* pares obtenham uma cópia do arquivo.
- O BitTorrent é um protocolo P2P popular para distribuição de arquivos.
- Diferentemente da arquitetura cliente/servidor, aplicações P2P tem dependência mínima ou nenhuma de um servidor que fique sempre ligado.

Aplicações P2P

Distribuição de arquivos P2P

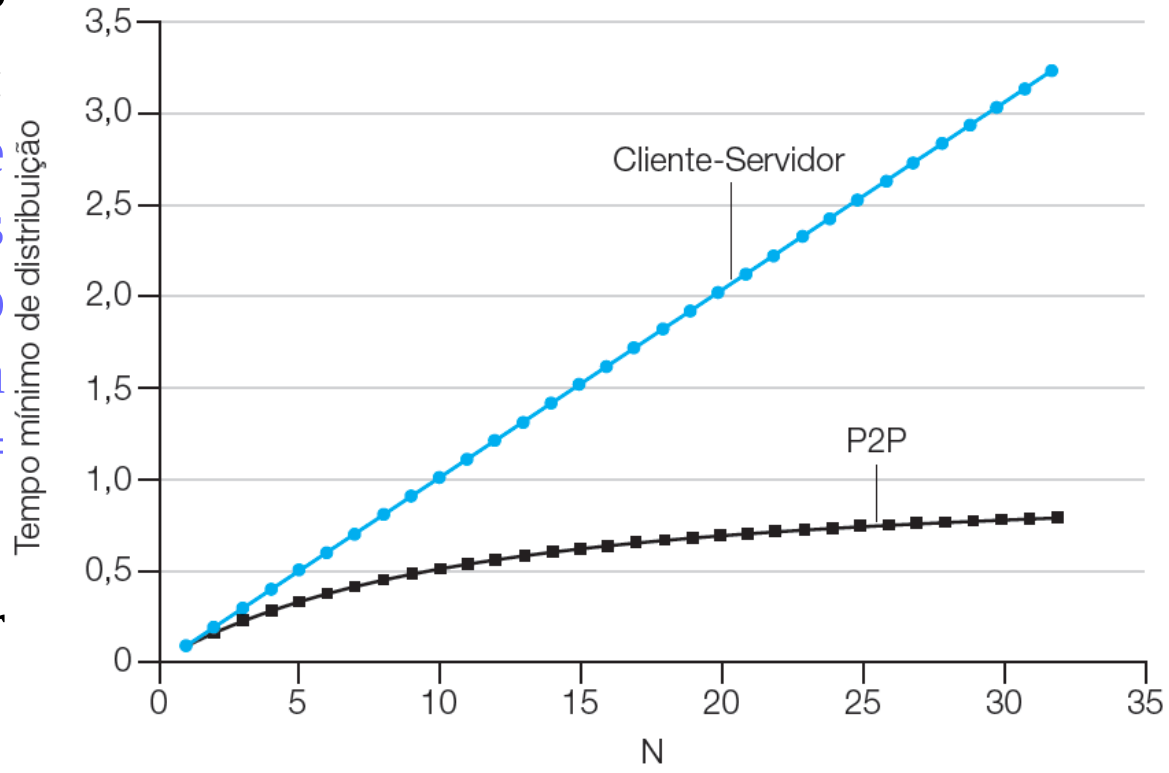
Um problema ilustrativo de distribuição de arquivo



Aplicações P2P

Distribuição de arquivos P2P

- Tempo de distribuição para arquiteturas P2P e cliente-servidor: troca entre um cliente e servidor 10x mais rápido quando comparado na troca entre dois pares (u_s = 10 x u_i).
- P2P – sempre menor que 1 hora
- Alta escalabilidade

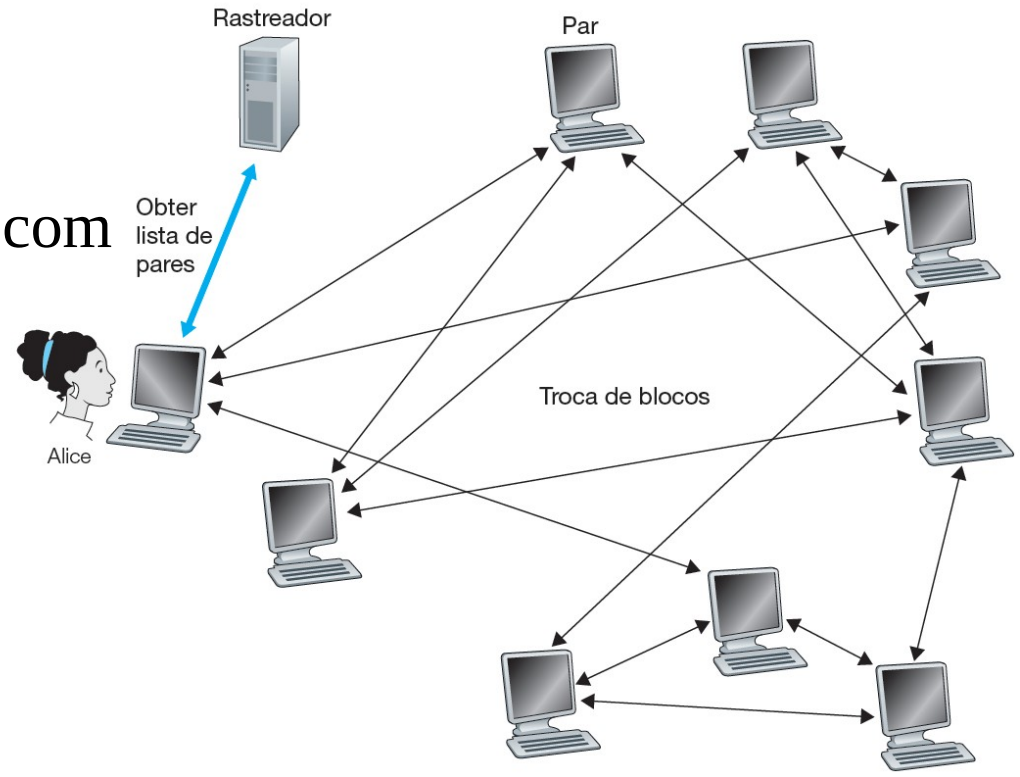


Aplicações P2P

Distribuição de arquivos P2P

Distribuição de arquivos com

- BitTorrent



BitTorrent – Par típico

1. Obtém a lista de pares
2. i) Quando um par entra no *torrent* ele não tem nenhum bloco.
   ii) A medida que o tempo passa ele vai acumulando blocos.
   iii) Enquanto faz o download de blocos faz também upload para outros pares (bloco típico 256 KB).
3. Cada par, ao entrar no *torrent* ganha um rastreador e periodicamente informa que está "vivo".

- Na figura acima, quando Alice entra, ganha um subconjunto de pares (IP), por exemplo 50.

4. Alice inicia o download de todas as partes do arquivo.
5. De tempos em tempos Alice pede qual os blocos que seus pares tem e verifica quais não tem.

BitTorrent

6.Rarest first: escolhe baixar os mais raros primeiro:

- aumenta o número de cópias dos raros.

6.Alice prioriza *upload* para os 4 vizinhos dos quais baixa com maior taxa: pares não sufocados (*unchocked*).

- O efeito é que pares capazes de fazer *upload* em taxas compatíveis se encontrem.
- Mecanismo denominado olho por olho.

6.A cada 30 s ela escolhe um vizinho adicional, ao acaso, e envia blocos a ele: otimisticamente não sufocado.

- Permite que novatos recebam blocos e tenham algo para trocar.