

Aplicação e Transporte em Redes de Computadores

Prof. Miguel Elias Mitre Campista

<http://www.gta.ufrj.br/~miguel>

Roteiro Resumido

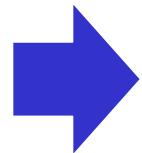
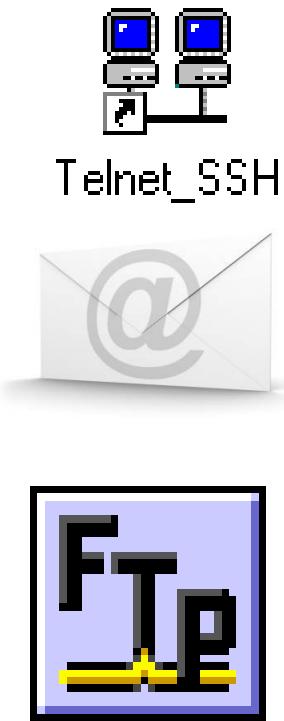
- Princípios básicos da Internet
- Princípios básicos de comunicação em redes
- Descrição das diferentes camadas de protocolos
 - Camada de aplicação e os seus protocolos
 - Camada de transporte e os seus protocolos
 - Camada de rede
 - Camada de enlace

Parte III

Camada de Aplicação e seus Protocolos

Aplicações: O Que Mudou?

- Número e características das aplicações
 - Poucas → muitas e com diferentes requisitos



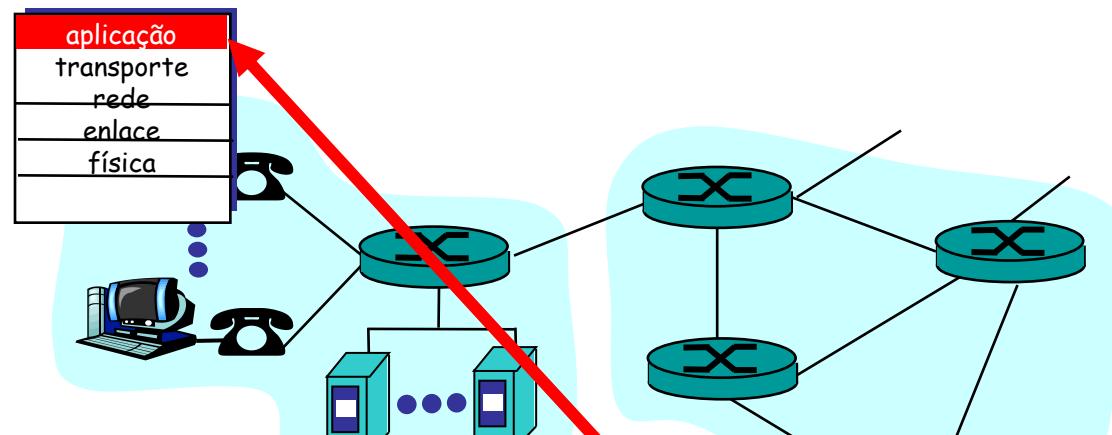
Importância das Aplicações

- Razão de ser das redes de computadores
 - Sem aplicações úteis, não haveria protocolos de rede para suportá-las
- Popularidade crescente
 - Do correio eletrônico, evoluiu para aplicações web incluindo o IPTV
 - Aumento das redes de acesso ajudou no aumento do número de aplicações

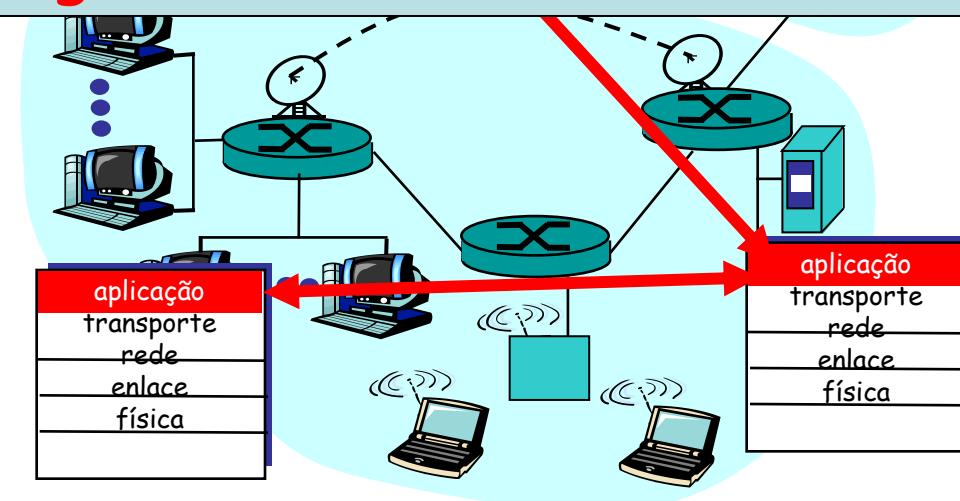
Aplicações: O Que São?

- Programas que
 - Executam em diferentes **sistemas finais**
 - Comunicam-se através da rede
 - Ex: servidor Web se comunica com um navegador
- Importante:
 - Dispositivos do núcleo da rede não executam aplicações de usuários
 - Aplicações nos sistemas finais permite rápido desenvolvimento e disseminação

Aplicações: O Que São?



Inteligência nas bordas e núcleo simples!

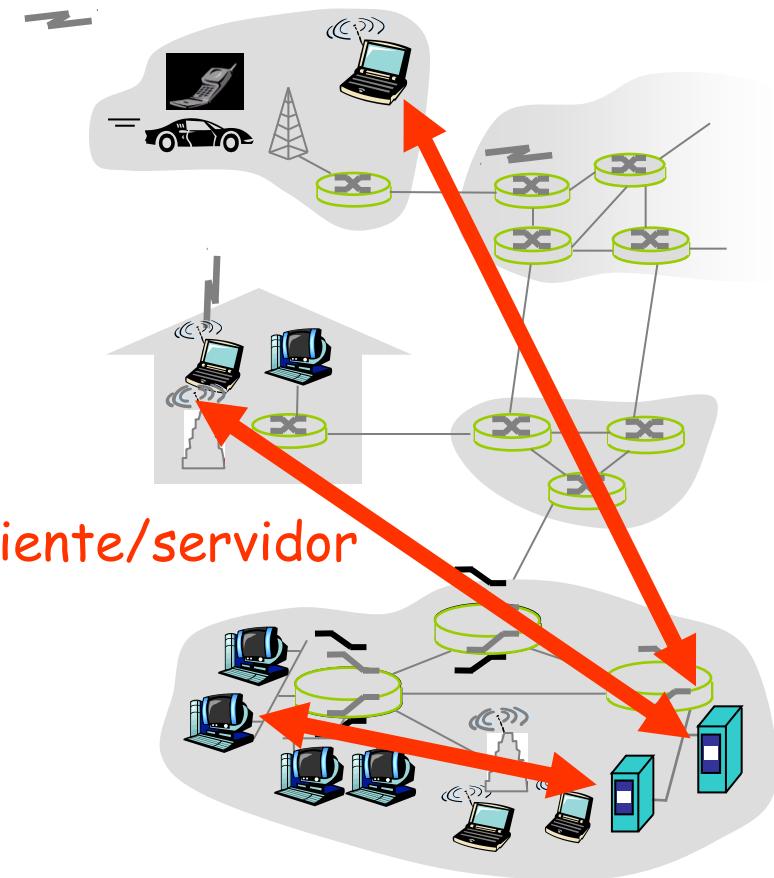


Arquiteturas de Aplicações

- Define como a aplicação está organizada nos sistemas finais
- Três básicas
 - Cliente-servidor
 - Par-a-par (P2P - *peer-to-peer*)
 - Híbrida

Cliente-Servidor

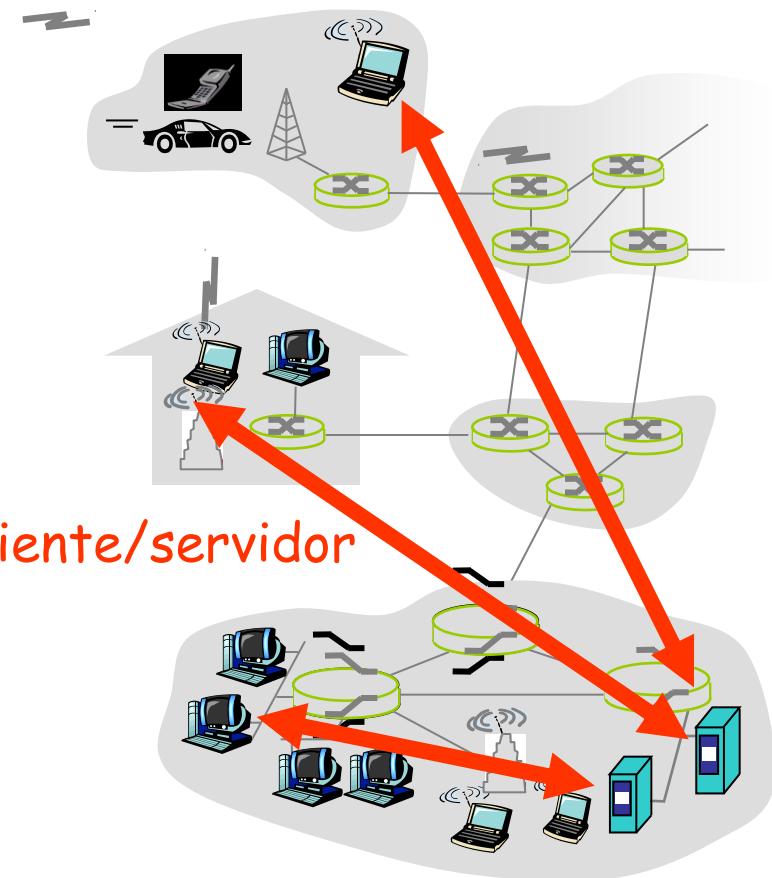
- Servidor
 - É um nó “especial”
 - Possui algum serviço de interesse
 - Recebe requisições dos clientes
 - Sempre ligado
 - **Disponibilidade**
 - Endereço conhecido
 - **Facilmente alcançável**



Cliente-Servidor

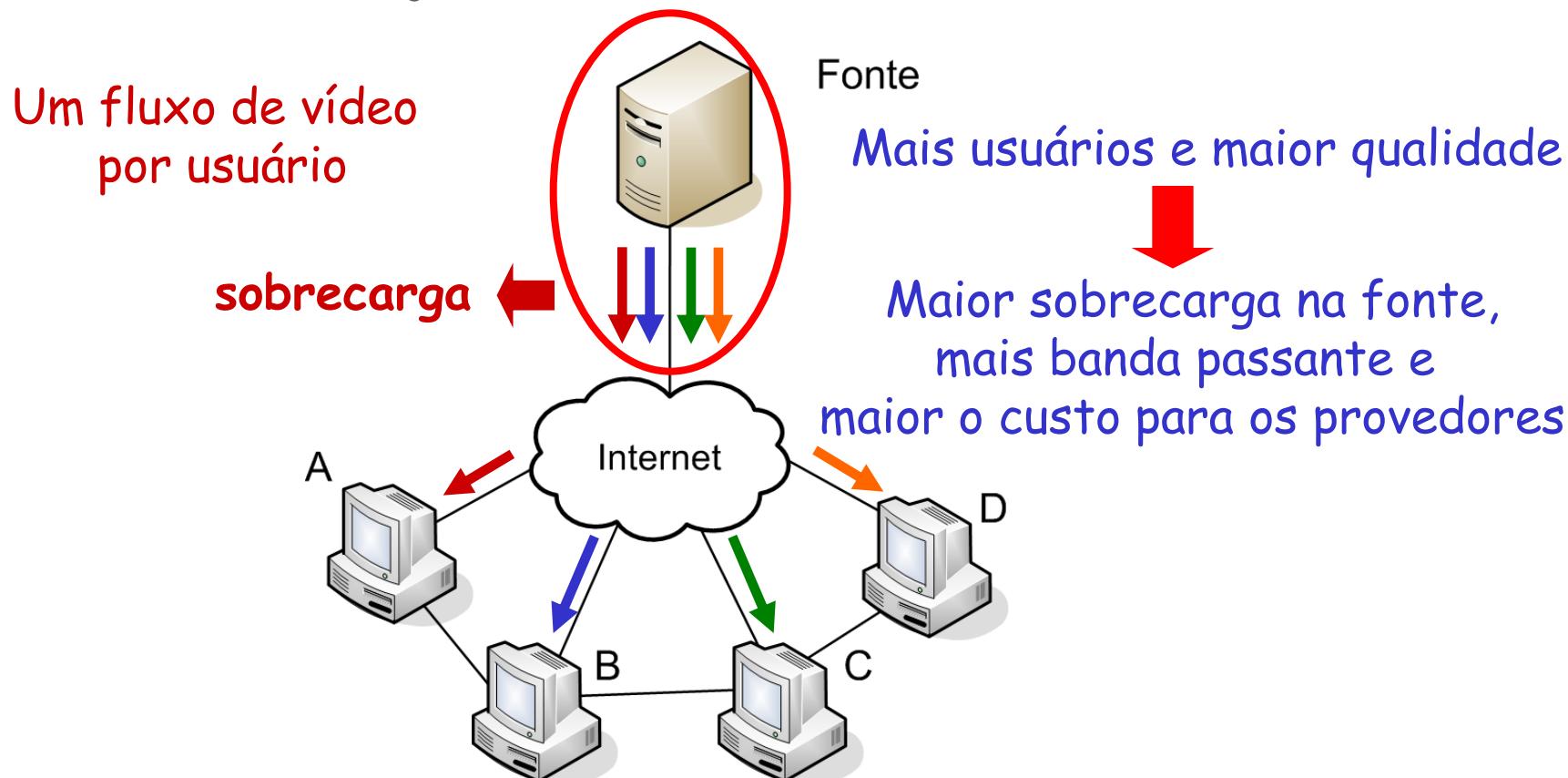
- **Cliente**

- Faz requisições ao servidor
- Não está necessariamente sempre ligado
- Endereço pode ser dinâmico
- **Não se comunica diretamente com outros clientes**



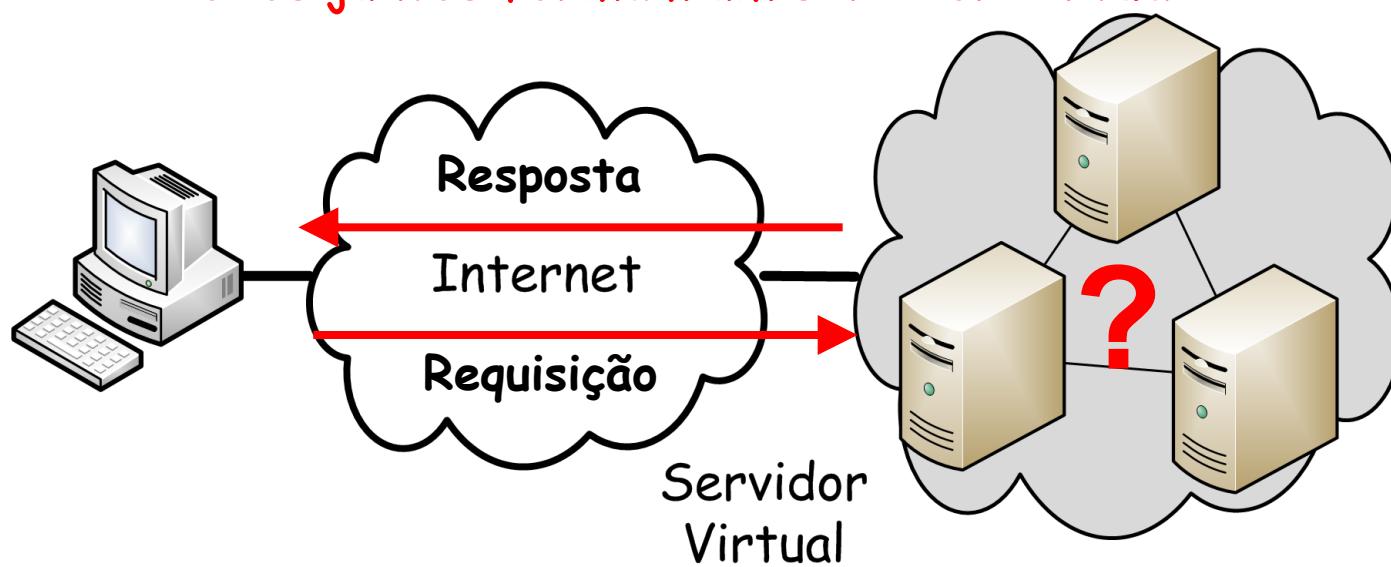
Cliente-Servidor

- Comunicação ponto-a-ponto
 - Ex.: distribuição de vídeo



Cliente-Servidor

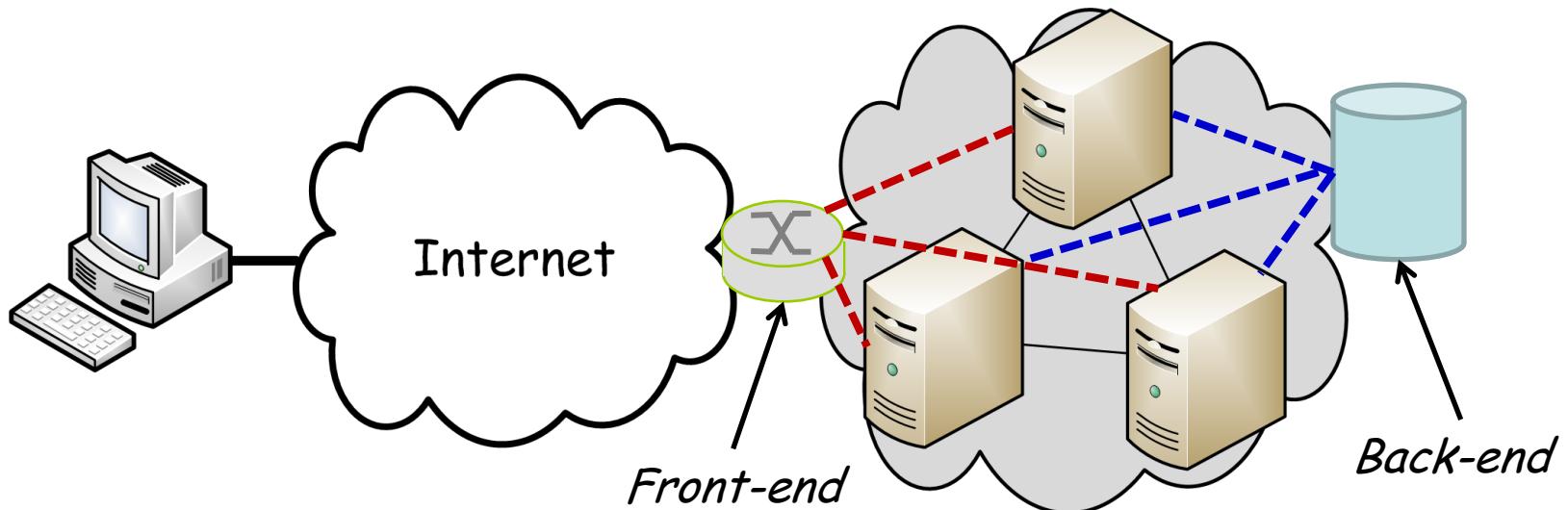
- Único servidor pode ficar saturado de requisições...
 - Emprego de um parque de servidores para atender múltiplas requisições
 - Todos juntos formam um **servidor virtual**



Torna o modelo cliente-servidor mais escalável...

Cliente-Servidor

- Servidor virtual
 - Deve parecer único para os clientes
 - Presença de um *front-end* para balanceamento de carga + banco de dados *back-end* para sincronismo dos dados
- Espalhamento através do DNS



Cliente-Servidor

- Espalhamento através do DNS: Servidores google

```
itaqua:~> traceroute www.google.com.br
traceroute to www.google.com.br (74.125.234.55), 30 hops max, 60 byte packets
 1  192.168.1.1 (192.168.1.1)  0.255 ms  0.354 ms  0.416 ms
 2  angra.gta.ufrj.br (146.164.69.129)  0.841 ms  1.806 ms  1.851 ms
 3  146.164.6.193 (146.164.6.193)  1.849 ms  2.026 ms  2.073 ms
 4  rt-ufrj.ufrj.br (146.164.1.193)  2.066 ms  2.134 ms  2.133 ms
 5  giga-bgp-cbpf.rederio.br (200.20.94.58)  17.819 ms  17.868 ms  17.946 ms
 6  xe-0-3-1.ar2.gig1.gblx.net (64.214.61.249)  17.813 ms  16.222 ms  15.186 ms
 7  * * *
 8  google-1.ar5.gru1.gblx.net (64.208.110.102)  332.760 ms  330.756 ms *
 9  209.85.243.200 (209.85.243.200)  22.436 ms ** *
10  209.85.251.99 (209.85.251.99)  23.143 ms  33.551 ms  33.544 ms
11  gru03s06-in-f23.1e100.net (74.125.234.55)  22.363 ms  22.921 ms  22.897 ms
itaqua:~> traceroute www.google.com.br
traceroute to www.google.com.br (74.125.234.63), 30 hops max, 60 byte packets
 1  192.168.1.1 (192.168.1.1)  0.288 ms  0.357 ms  0.414 ms
 2  angra.gta.ufrj.br (146.164.69.129)  1.772 ms  1.818 ms  1.817 ms
 3  146.164.6.193 (146.164.6.193)  1.885 ms  1.937 ms  1.936 ms
 4  rt-ufrj.ufrj.br (146.164.1.193)  1.980 ms  2.046 ms  2.191 ms
 5  giga-bgp-cbpf.rederio.br (200.20.94.58)  15.799 ms  15.809 ms  16.280 ms
 6  xe-0-3-1.ar2.gig1.gblx.net (64.214.61.249)  15.796 ms  15.026 ms  14.641 ms
 7  po5.asrl.GRU1.gblx.net (67.16.130.58)  21.295 ms po3.asrl.GRU1.gblx.net (67.16.139.166)  21.302 ms po5.asrl.GRU1.gblx.net (67.16.130.58)  20.895 ms
 8  google-1.ar5.gru1.gblx.net (64.208.110.102)  344.980 ms  345.133 ms  344.953 ms
 9  209.85.243.200 (209.85.243.200)  21.319 ms  21.416 ms  21.240 ms
10  209.85.251.99 (209.85.251.99)  22.359 ms  20.818 ms  21.050 ms
11  gru03s06-in-f31.1e100.net (74.125.234.63)  20.027 ms  21.183 ms  21.396 ms
```

Cliente-Servidor

- Espalhamento através do DNS: Servidores google
 - Nome está relacionado ao endereço de servidores distintos

```
itaqua:~> dig www.gta.ufrj.br

; <>> DiG 9.7.3 <>> www.gta.ufrj.br
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3575
;; Flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 7, ADDITIONAL: 6

;; QUESTION SECTION:
;www.gta.ufrj.br.      IN      A

;; ANSWER SECTION:
www.gta.ufrj.br.    86400   IN      CNAME   recreio.gta.ufrj.br.
recreio.gta.ufrj.br. 86400   IN      A       146.164.69.2
```

```
itaqua:~> dig www.google.com.br

; <>> DiG 9.7.3 <>> www.google.com.br
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7740
;; Flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;www.google.com.br.      IN      A

;; ANSWER SECTION:
www.google.com.br.    44      IN      A       74.125.234.56
www.google.com.br.    44      IN      A       74.125.234.63
www.google.com.br.    44      IN      A       74.125.234.55
```

Redes de Distribuição de Conteúdo

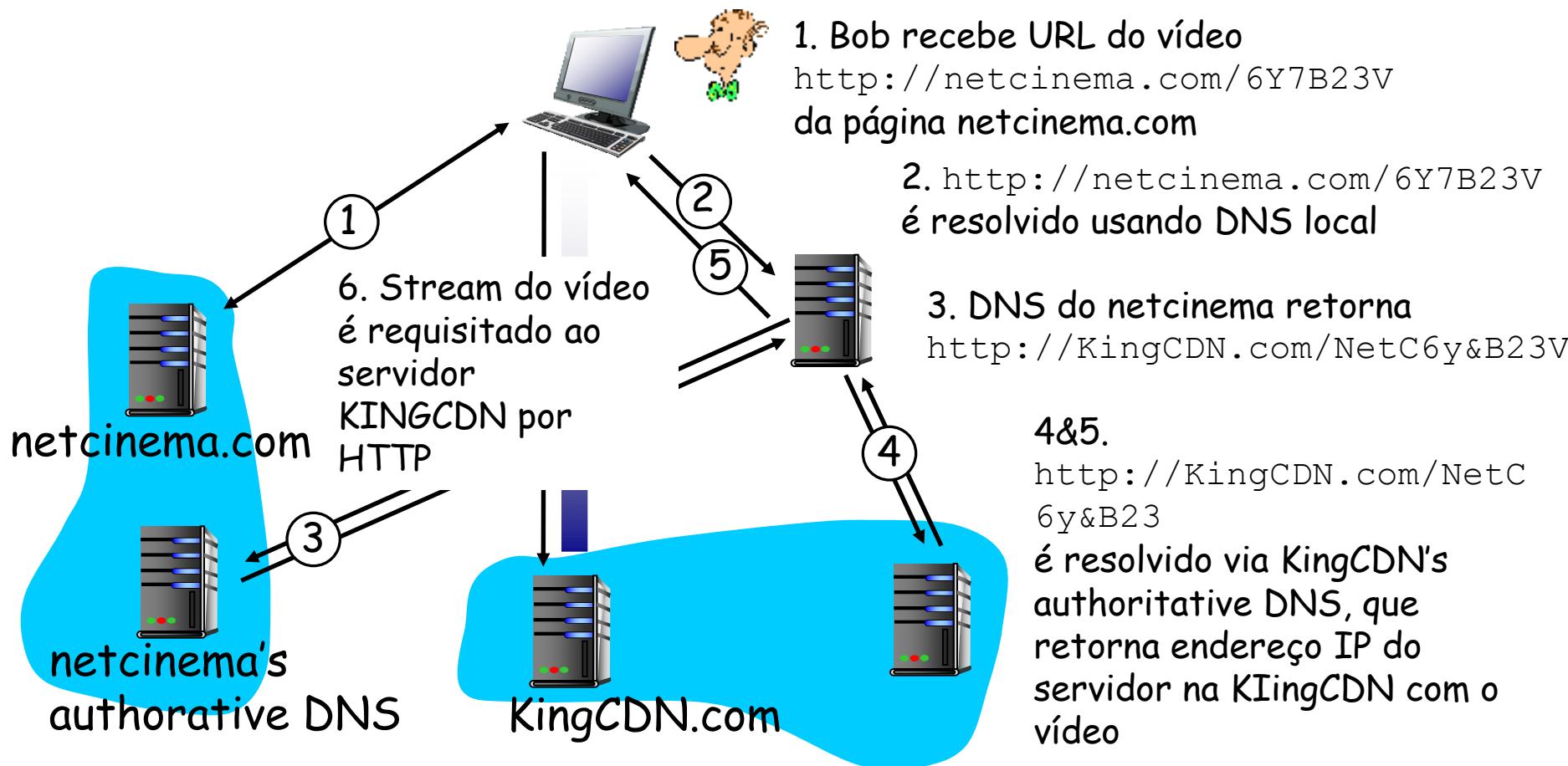
- Tornam o modelo cliente-servidor mais eficiente e escalável
 - Parque de servidores não é suficiente para sites muito populares
 - Exemplo: site de distribuição de vídeo
- Definem conjunto de servidores auxiliares (espelhos)
 - Replicação do conteúdo
 - Espalhados geograficamente
 - Pertencem a diferentes *backbones*

Redes de Distribuição de Conteúdo

- Interceptam a requisição do cliente
 - Determinação do servidor CDN mais apropriado para o cliente naquele momento
 - Redirecionamento da requisição para o servidor escolhido
- Operação preferencialmente transparente para o cliente
 - Há a possibilidade do cliente escolher o servidor auxiliar de uma lista estática recebida como resposta

Redes de Distribuição de Conteúdo

- Operação baseada no uso de DNS (figura Kurose)
 - Interceptação e redirecionamento de requisições

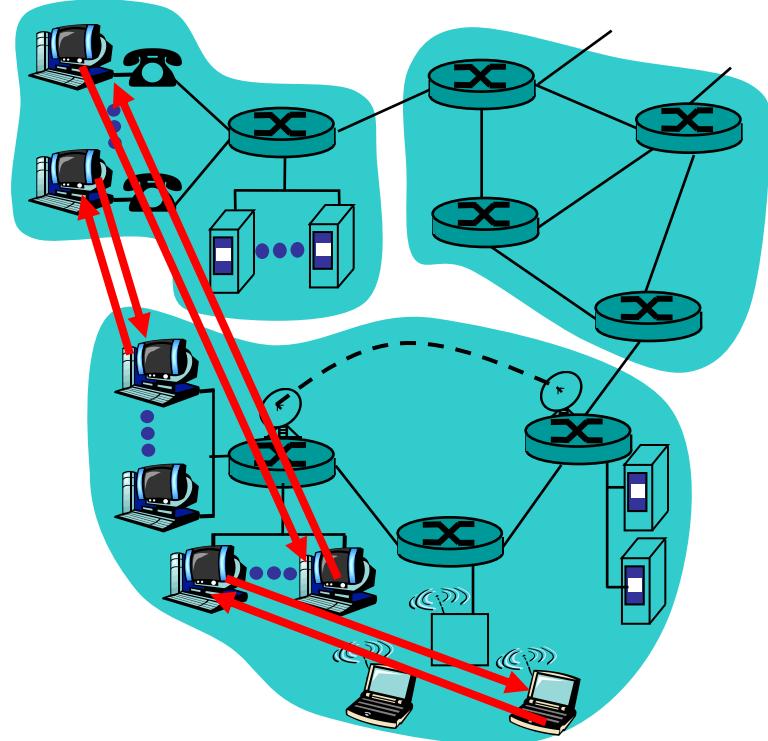


Redes de Distribuição de Conteúdo

- Desafios
 - Encaminhamento da requisição
 - Escolha do servidor de réplica
 - Replicação do conteúdo
- Desvantagem
 - Eficiência depende do número de servidores auxiliares
 - Alto custo
- Exemplo: Akamai
 - 19 mil servidores na Internet
 - Transmissão do concerto Live Earth
 - 237 mil usuários simultâneos e 15 milhões de fluxos no total

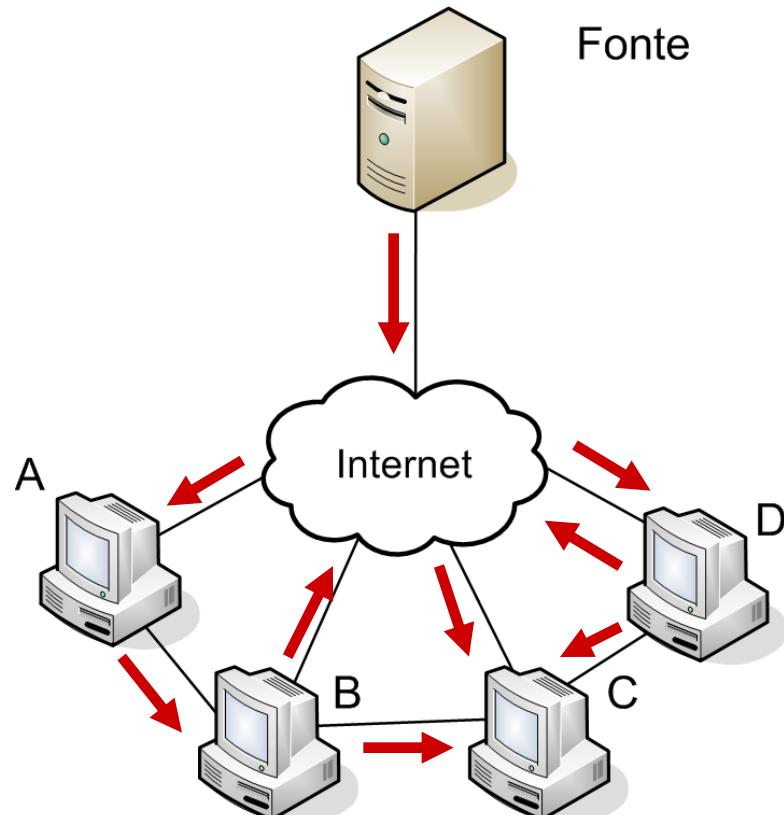
Par-a-Par

- Não requer funcionamento permanente de servidores
 - Comunicação direta entre sistemas finais
 - Sistemas finais não são propriedade dos provedores de serviço
 - Sistemas finais são controlados por usuários



Par-a-Par

- Participantes colaboram para o funcionamento e manutenção do sistema



Par-a-Par

- Participantes colaboram para o funcionamento e manutenção do sistema
 - Compartilhamento de recursos
 - Banda passante, processamento e armazenamento
 - Mais participantes → maior a capacidade
 - Escalabilidade
- Problemas: gerenciamento
 - Não há um elemento dedicado
 - Não há garantia de continuidade do serviço
 - Pares estão conectados间断地 e mudam de endereços IP

Híbrida

- Arquitetura par-a-par com uso de servidores auxiliares
 - Skype
 - Aplicação par-a-par de voz sobre IP
 - Localização do endereço do parceiro remoto: servidor
 - Conversação é direta: cliente-cliente
- Mensagem instantânea
 - Conversação é direta: cliente-cliente
 - Localização e detecção de presença são centralizadas
 - Usuários registram o seu endereço IP junto ao servidor central quando ficam *online*
 - Usuários consultam o servidor central para encontrar endereços IP dos contatos

Desafios da Arquitetura Par-a-Par

- Provedor de serviço amigável
 - Provedores residenciais oferecem taxas maiores para downstream
 - Aplicações usam igualmente banda para upstream
- Segurança
 - Aplicações são distribuídas e os dados são expostos
 - Participação direta dos usuários no funcionamento
- Incentivos
 - Usuários devem compartilhar recursos
 - Funcionamento do sistema depende dessa participação

Comunicação entre Processos

- **Processo:** programa que executa num sistema final
 - Processos no mesmo sistema final se comunicam usando comunicação interprocessos definida pelo sistema operacional
 - Processos em sistemas finais distintos se comunicam trocando mensagens pela rede

Processo cliente:

processo que inicia a comunicação

Processo servidor:

processo que espera ser contatado

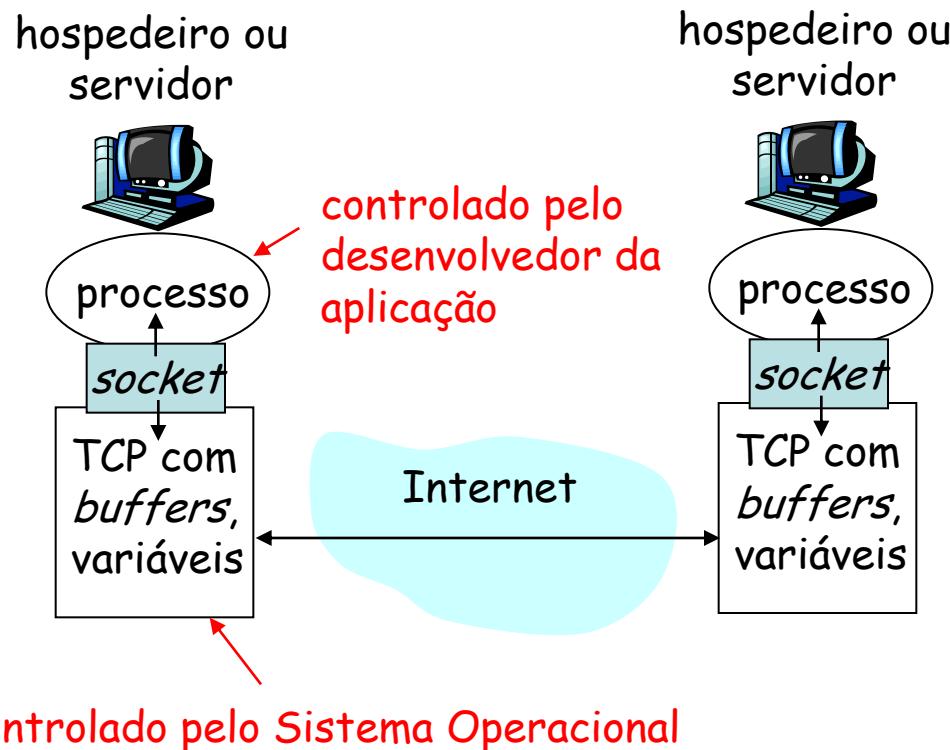
Nota: aplicações com arquiteturas P2P possuem processos clientes e processos servidores

Socket

- Os processos enviam/recebem mensagens para/dos seus sockets
- Um socket é análogo a uma porta
 - Processo transmissor envia a mensagem através da porta
 - O processo transmissor assume a existência da infraestrutura de transporte no outro lado da porta que faz com que a mensagem chegue ao socket do processo receptor

Socket

- API: (1) escolha do protocolo de transporte; (2) habilidade para fixar alguns parâmetros (p.ex. tamanho máximo do buffer)



Requisitos das Aplicações

- Transferência confiável de dados
 - Algumas aplicações podem tolerar perdas
 - Ex.: áudio e vídeo não-codificados
 - Outras requerem transferência 100% confiável
 - Transferência de arquivos, email, SSH, etc.

Vídeo Codificado em MPEG-4

99,0 %



97,0 %



92,0 %



Requisitos das Aplicações

- Banda passante
 - Algumas aplicações exigem uma quantidade mínima de banda para funcionarem
 - Aplicações multimídias
 - Outras aplicações se adaptam a banda disponível
 - Aplicações elásticas
 - Web, email, transferência de arquivos, etc.

Requisitos das Aplicações

- Atraso
 - Algumas aplicações exigem um atraso máximo para funcionarem
 - Aplicações interativas em tempo real
 - Outras aplicações toleram o atraso
 - Quanto menor melhor, mas não há limites de atraso fim-a-fim

Requisitos das Aplicações

- Segurança
 - Autenticação
 - Controle de acesso
 - Integridade
 - Não-repúdio
 - Confidencialidade

Requisitos das Aplicações

Aplicação	Perda	Banda passante	Atraso
Transferência de arquivos	sem perdas	elástica	tolerante
Email	sem perdas	elástica	tolerante
Web	sem perdas	elástica	tolerante
Áudio/vídeo em tempo real	tolerante	áudio: 5kb-1Mb vídeo:10kb-5Mb	centenas de miliseg.
Áudio/vídeo gravado	tolerante	Idem	poucos seg.
Jogos interativos	tolerante	até 10 kbps	centenas de miliseg.
Mensagens instantâneas	sem perdas	elástica	sim/não (?)

Serviços de Transporte

- Serviço oferecido pelo TCP
 - Orientado a conexão
 - Estabelecimento de conexão fim-a-fim
 - Mensagens de controle antes da troca de mensagens da aplicação
 - Transporte confiável
 - Entre processos emissor e receptor
 - Controle de fluxo
 - Emissor não irá sobrecarregar o receptor
 - Controle de congestionamento
 - A taxa de envio do emissor depende da carga da rede
 - Não provê garantias temporais ou de banda mínima

Serviços de Transporte

- Serviço oferecido pelo UDP
 - Transferência de dados não-confiável
 - Entre processos emissor e receptor
 - Não provê
 - Estabelecimento da conexão
 - Confiabilidade
 - Controle de fluxo
 - Controle de congestionamento
 - Garantias temporais ou de banda mínima

Requisitos das Aplicações

Aplicação	Protocolo de aplicação	Protocolo de transporte
Email	SMTP	TCP
Acesso remoto	Telnet, SSH	TCP
Web	HTTP	TCP
Transferência de arquivos	FTP	TCP
Distribuição multimídia	HTTP, RTP	TCP ou UDP
Telefonia na Internet	SIP, RTP, proprietário (Skype)	tipicamente UDP

Protocolos de Camada de Aplicação

Protocolos de Aplicação

- Definem:
 - Tipos de mensagens trocadas
 - Ex.: mensagens de requisição e resposta
 - Sintaxe das mensagens
 - Campos presentes nas mensagens e como são identificados
 - Semântica das mensagens
 - Significado da informação carregada por cada campo
 - Regras para quando os processos enviam e respondem às mensagens

Protocolos de Aplicação

- Domínio público
 - Definidos geralmente por RFCs (Request for Comments)
 - Documentos de responsabilidade do IETF (Internet Engineering Task Force)
 - Drafts são versões ainda em aberto
- Proprietários
 - Código-fonte fechado
 - Ex.: Skype

Conceitos Web

- 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
 - Hiperlinks
 - Cada objeto é endereçável por uma URL (*Uniform Resource Locator*)
 - URL contém o nome do hospedeiro e o caminho do objeto

www.gta.ufrj.br/~miguel/courses.html

Conceitos Web

- Tipos de páginas web
 - Estática
 - Mesmo documento é apresentado a cada exibição
 - Dinâmica
 - Documentos são gerados sob demanda por um programa ou contêm um programa
 - Pode se apresentar de forma diferente a cada acesso
- Navegador
 - Programa que exibe páginas web e captura cliques de mouse em itens da página exibida

Cliente Web

- Passos após o clique sobre um hiperlink:
 1. Navegador determina URL (verifica o selecionado)
 2. Navegador pergunta ao DNS o endereço IP do servidor (parte da URL entre o protocolo e a / seguinte)
 - Pergunta: Qual o endereço IP de **www.gta.ufrj.br/**?
 3. DNS responde com o endereço IP do servidor
 - Resposta: **146.164.69.2**
 4. Navegador estabelece conexão TCP com o servidor
 - Usa a porta 80 - Porta executando o serviço HTTP

Cliente Web

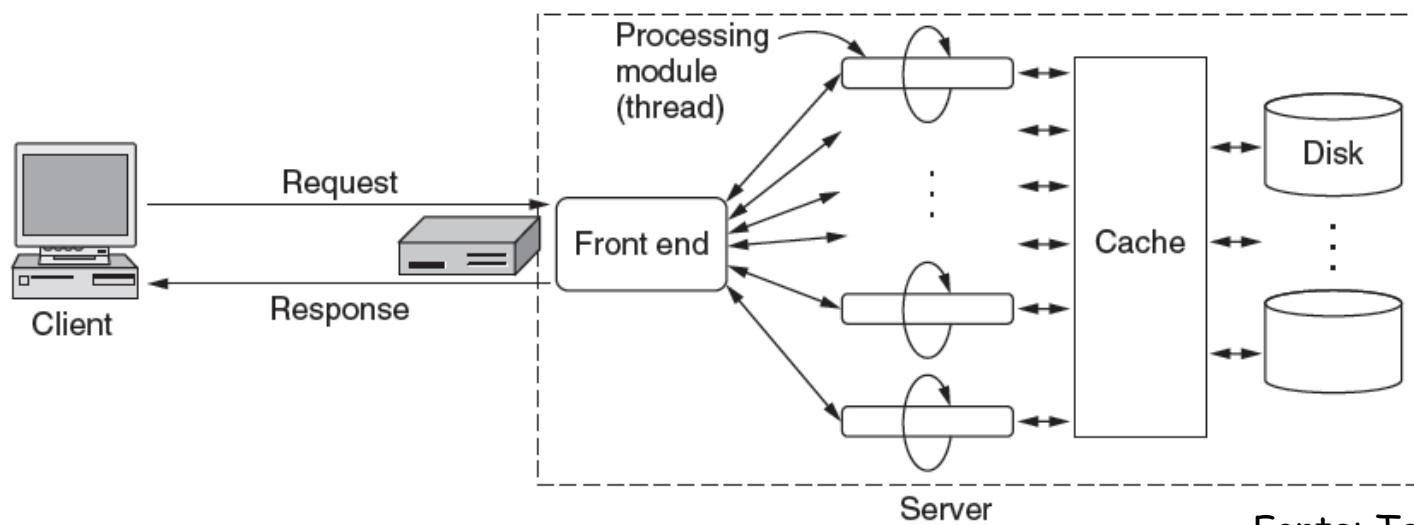
- Passos após o clique sobre um hiperlink:
 5. Navegador envia requisição HTTP solicitando a página index.html
 6. Servidor envia uma resposta HTTP contendo a página solicitada
 - Caso a página possua outras URLs, a operação é repetida
 - As URLs podem incluir diferentes tipos de objetos
 7. Navegador exibe a página
 8. Conexões TCP são fechadas

Servidor Web

- Passos gerais executados em seu loop principal:
 1. Aceita conexão TCP de um cliente (navegador)
 2. Obtém o caminho até a página (nome do arquivo solicitado)
 3. Obtém o arquivo do disco (Gargalo!)
 - Conteúdo pode ser armazenado em cache
 4. Envia o conteúdo do arquivo ao cliente
 5. Encerra a conexão TCP

Servidor Web

- Escalabilidade: Servidor pode ser *multi-threaded*
 - Processo composto por um módulo *front-end* que aceita as solicitações recebidas + n threads de processamento
 - Todas as threads têm acesso ao cache
 - *Front-end* encaminha a solicitação para a thread que busca o objeto em cache ou em disco



Fonte: Tanenbaum

Servidor Web

- Podem desempenhar ainda...
 - Tradução de nomes
 - Ex. expande `http://www.gta.ufrj.br` para `http://www.gta.ufrj.br/index.html`
 - Controle de acesso
 - Determinados arquivos são de acesso restrito
 - Requerem identidade do cliente
 - Requerem endereço IP de origem conhecidos
 - Verificação de cache
 - Objetos em cache podem estar desatualizados
 - etc.

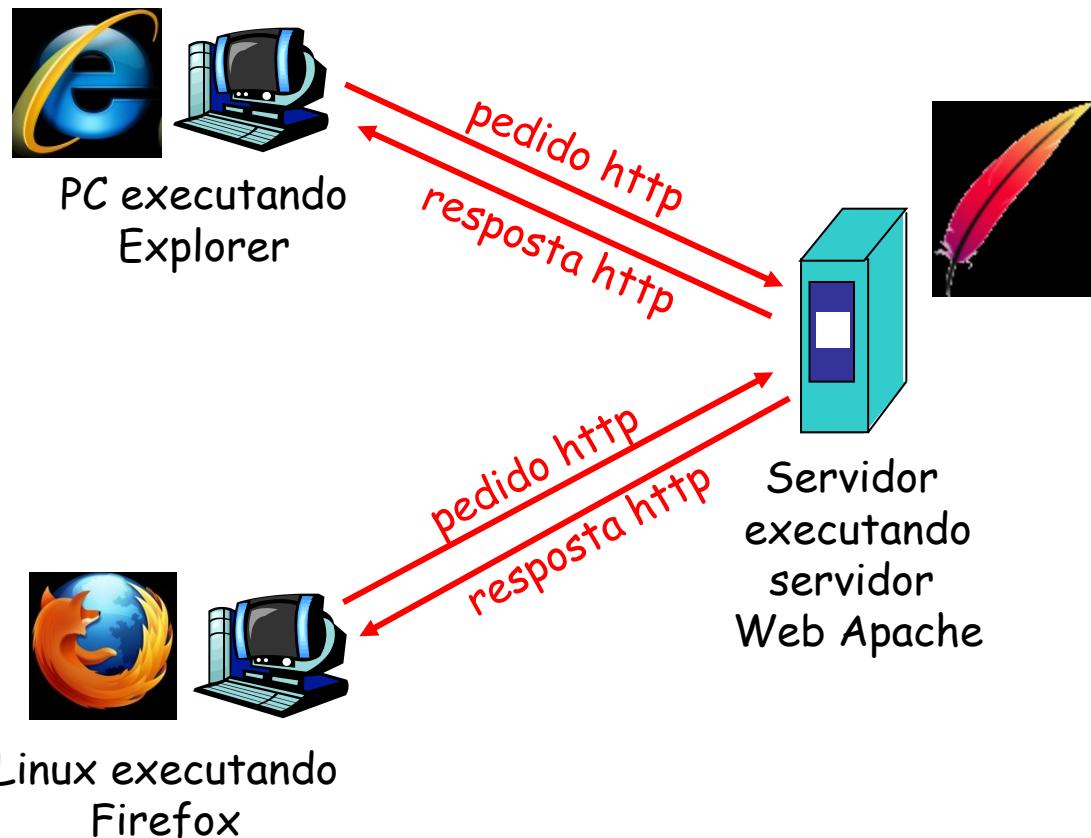
HyperText Transfer Protocol (HTTP)

Protocolo HTTP

- Aplicação: **navegação Web**
 - Diferente de outras aplicações, a web permite a obtenção de conteúdo **sob demanda e de forma interativa**
- Modelo cliente/servidor
 - Cliente
 - Navegador que pede, recebe e “visualiza” os objetos Web
 - Servidor
 - Servidor Web envia objetos em resposta a pedidos

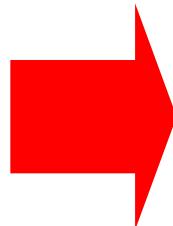
Protocolo HTTP

- Aplicação: navegação Web
- Modelo cliente/servidor



Protocolo HTTP

- Usa o TCP como protocolo de transporte
 - Cliente inicia conexão TCP com o servidor
 - Geralmente na porta 80
 - Servidor aceita conexão TCP do cliente
 - Mensagens HTTP trocadas entre o navegador (cliente HTTP) e o servidor Web (servidor HTTP)
 - Cliente encerra a conexão TCP



Assegurar uma transmissão
confiável é tarefa do TCP

Protocolo HTTP

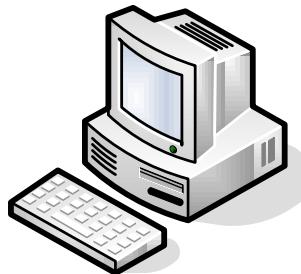
- É um protocolo **sem estado**
 - Servidor não mantém informação sobre pedidos anteriores do cliente
 - Um mesmo objeto pedido pela segunda vez é reenviado
- Observação
 - Protocolos que mantêm "estado" são complexos
 - Estados passados tem que ser guardados
 - Consumo de memória
 - Caso servidor/cliente caia, suas visões do "estado" podem ficar inconsistentes e devem ser atualizadas

Protocolo HTTP

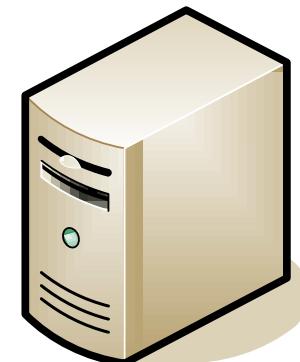
- Dois tipos de conexão
 - Não persistente (versão 1.0)
 - Uma requisição/resposta por conexão TCP
 - Pensada para páginas contendo um único objeto (p.ex. página web que só contém texto)
 - Persistente (versão 1.1)
 - Mais de uma requisição/resposta por conexão TCP
 - Evolução dado que as páginas web possuem cada vez mais objetos

Conexão Não-Persistente

Usuário digita a URL **www.gta.ufrj.br**



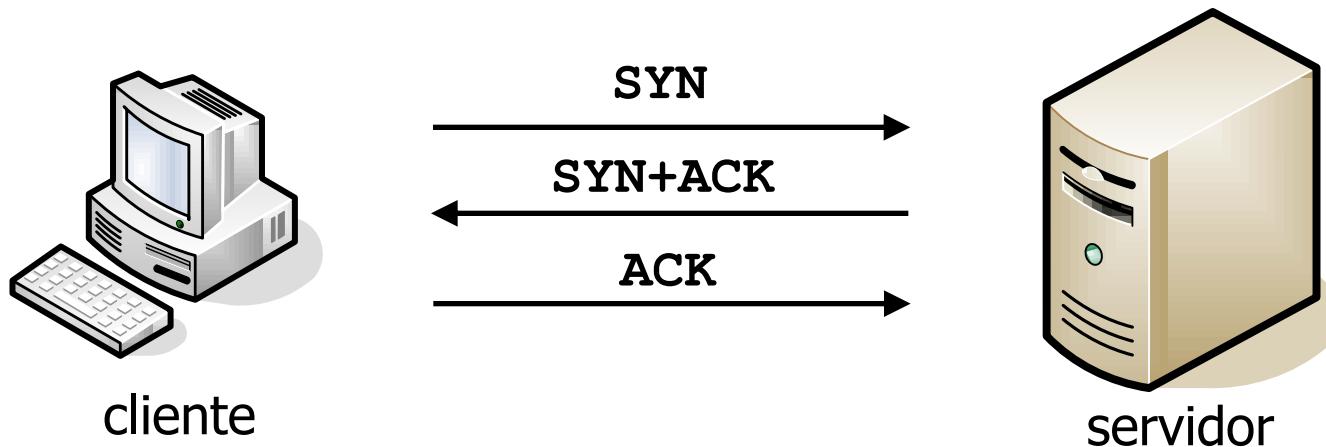
cliente



servidor

Conexão Não-Persistente

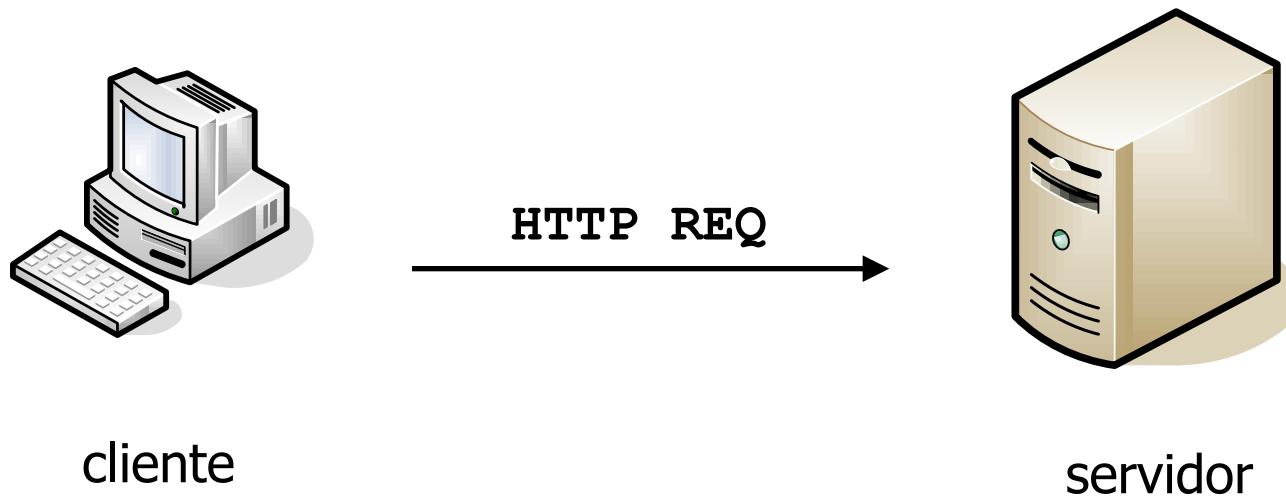
Usuário digita a URL `www.gta.ufrj.br`



1. Cliente HTTP inicia conexão TCP a servidor HTTP (processo) a `www.gta.ufrj.br` pela porta padrão 80

Conexão Não-Persistente

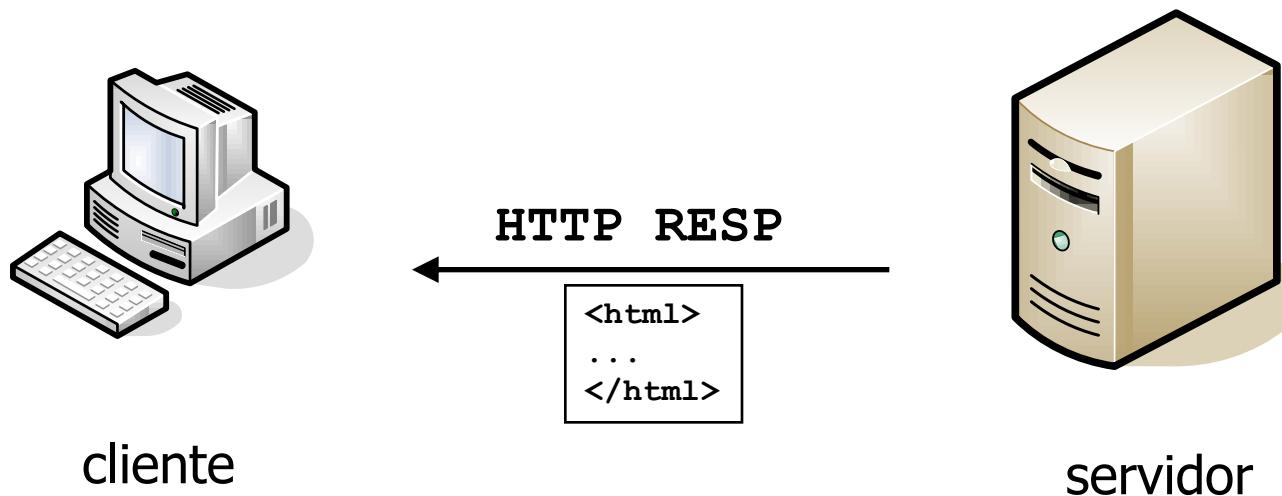
Usuário digita a URL `www.gta.ufrj.br`



2. Cliente HTTP envia mensagem de pedido de HTTP (contendo URL) através da conexão TCP. A mensagem indica que o cliente deseja receber o objeto `www.gta.ufrj.br/index.html`

Conexão Não-Persistente

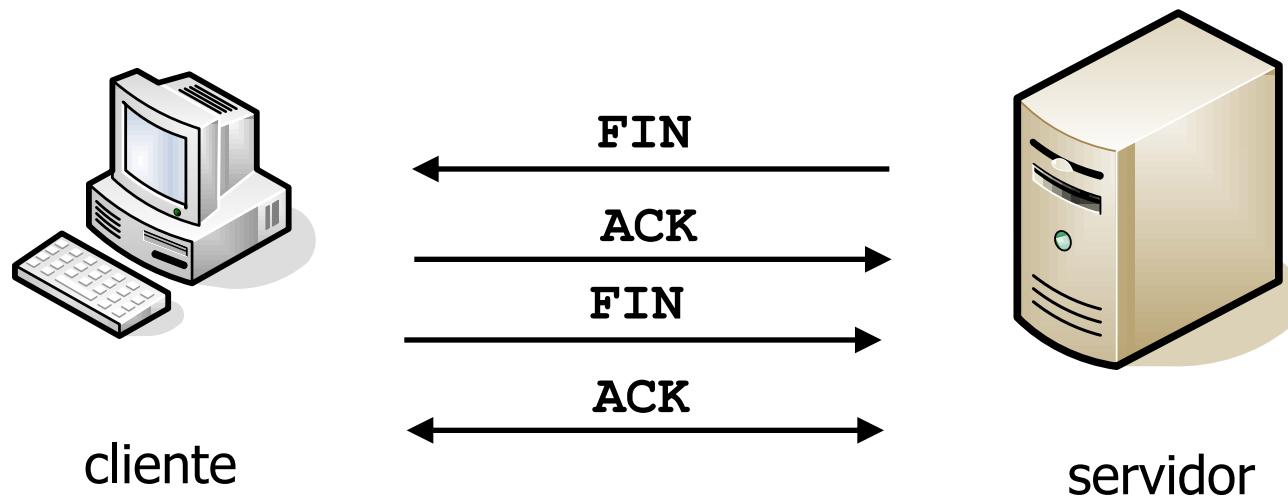
Usuário digita a URL `www.gta.ufrj.br`



3. Servidor HTTP recebe mensagem de pedido, formula mensagem de resposta contendo objeto solicitado e envia a mensagem

Conexão Não-Persistente

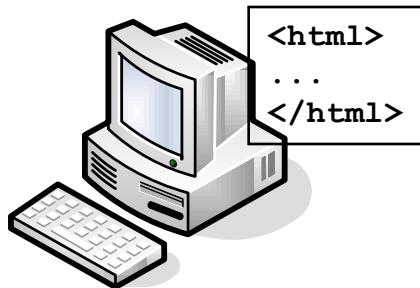
Usuário digita a URL `www.gta.ufrj.br`



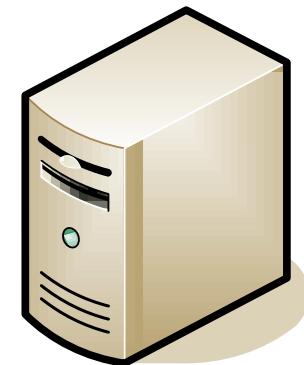
4. Servidor HTTP encerra a conexão TCP

Conexão Não-Persistente

Usuário digita a URL `www.gta.ufrj.br`



cliente

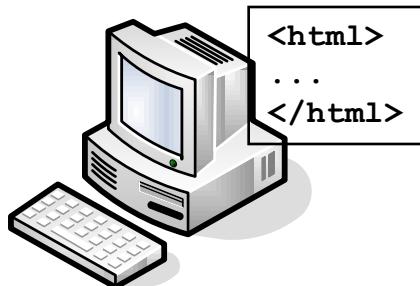


servidor

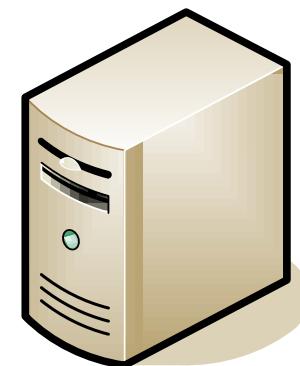
5. Cliente HTTP recebe mensagem de resposta contendo arquivo HTML e visualiza o HTML. Analisando o arquivo, encontra diversos objetos JPEG referenciados

Conexão Não-Persistente

Usuário digita a URL `www.gta.ufrj.br`



cliente

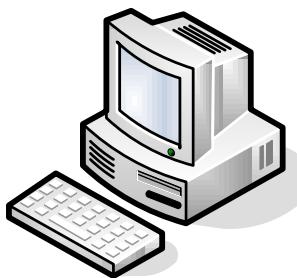


servidor

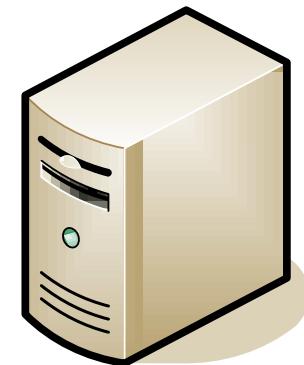
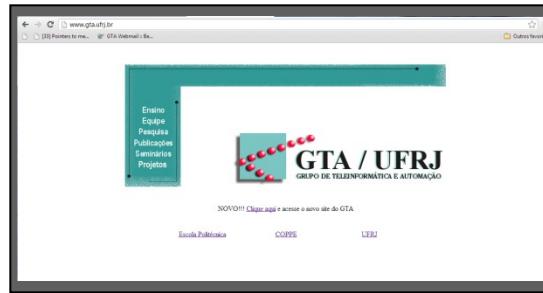
Repete os passos de 1 a 5 para cada objeto encontrado

Conexão Não-Persistente

Usuário digita a URL `www.gta.ufrj.br`



cliente

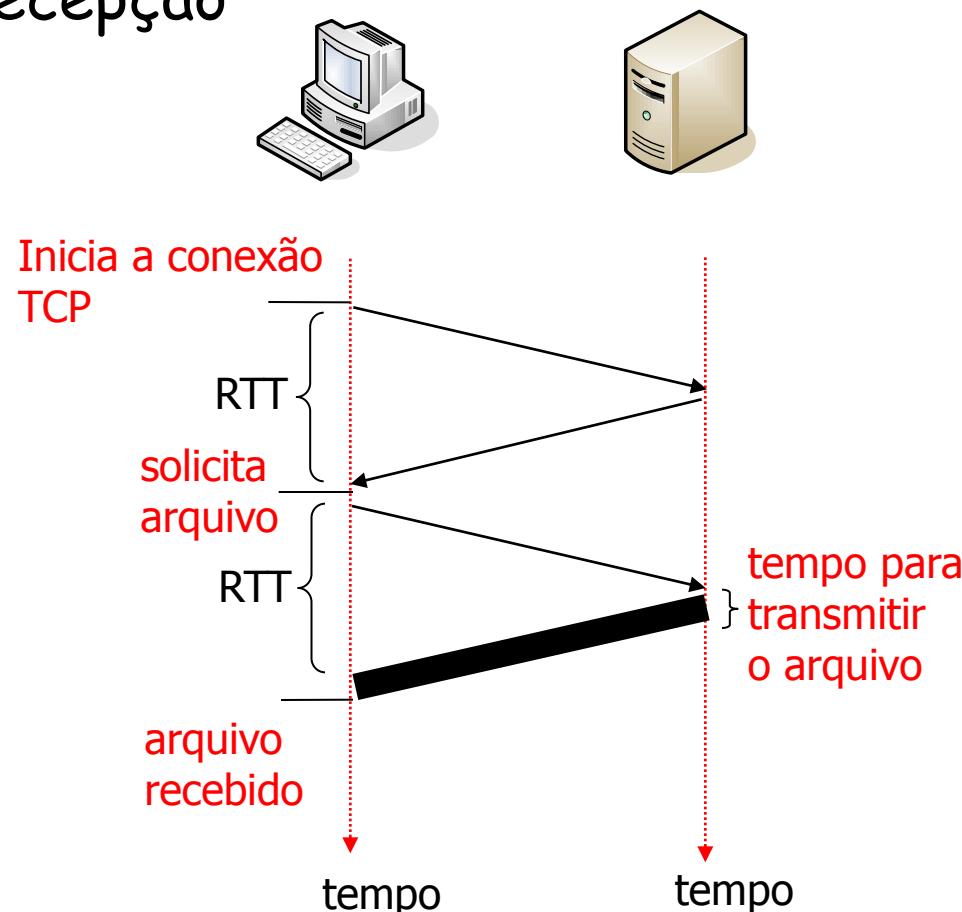


servidor

Visualiza a página com todos os seus objetos

Conexão Não-Persistente

- **Tempo de resposta:** tempo entre um pedido de um objeto e sua recepção



Conexão Não-Persistente

- **Tempo de resposta:** tempo entre um pedido de um objeto e sua recepção
 - Um RTT para iniciar a conexão TCP
 - Three-way handshake
 - Um RTT para o pedido HTTP e o retorno dos primeiros bytes da resposta HTTP
 - Tempo total de transmissão do objeto
 - Total = $2 * \text{RTT} + \text{tempo para transmitir o arquivo}$

Conexão Não-Persistente

- Prós
 - Os navegadores frequentemente abrem conexões TCP paralelas para recuperar os objetos referenciados
 - Múltiplas conexões TCP com o mesmo destinatário podem competir entre elas e prejudicar o desempenho global
- Contras
 - Requer 2 RTTs para cada objeto
 - Sistema Operacional aloca recursos do hospedeiro para cada conexão TCP

Conexão Persistente

- Presente na versão 1.1
- 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 mais o tempo para transmitir os arquivos
 - Os objetos são solicitados em sequência, sem esperar a resposta à solicitação anterior

Conexão Persistente

Estabeleci-
mento de
conexão
TCP

Envio da
página

```
root@itaqua:/home/miguel/submissões/revistas/comnet11# tshark 'tcp port 80'  
Running as user "root" and group "root". This could be dangerous.  
Capturing on eth0  
0.000000 192.168.1.100 -> 146.164.69.2 TCP 41869 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=217325283 TSER=0 WS=7  
0.000489 146.164.69.2 -> 192.168.1.100 TCP http > 41869 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=121360163 TSER=217325283 WS=7  
0.000500 192.168.1.100 -> 146.164.69.2 TCP 41869 > http [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=217325283 TSER=121360163  
0.000517 192.168.1.100 -> 146.164.69.2 HTTP GET / HTTP/1.1  
0.001088 146.164.69.2 -> 192.168.1.100 TCP http > 41869 [ACK] Seq=1 Ack=606 Win=7040 Len=0 TSV=121360164 TSER=217325283  
0.004361 146.164.69.2 -> 192.168.1.100 TCP [TCP segment of a reassembled PDU]  
0.004371 192.168.1.100 -> 146.164.69.2 TCP 41869 > http [ACK] Seq=606 Ack=1449 Win=8832 Len=0 TSV=217325284 TSER=121360164  
0.004424 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 200 OK (text/html)  
0.004433 192.168.1.100 -> 146.164.69.2 TCP 41869 > http [ACK] Seq=606 Ack=2534 Win=11648 Len=0 TSV=217325284 TSER=121360164  
0.015767 192.168.1.100 -> 146.164.69.2 HTTP GET /img/botao-ensino-nao-selecionado.gif HTTP/1.1  
0.016607 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified  
0.016812 192.168.1.100 -> 146.164.69.2 HTTP GET /img/botao-equipe-selecionado.gif HTTP/1.1  
0.017014 192.168.1.100 -> 146.164.69.2 TCP 41870 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=217325288 TSER=0 WS=7  
0.017034 192.168.1.100 -> 146.164.69.2 TCP 41871 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=217325288 TSER=0 WS=7  
0.017525 146.164.69.2 -> 192.168.1.100 TCP http > 41870 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=121360167 TSER=217325288 WS=7  
0.017533 192.168.1.100 -> 146.164.69.2 TCP 41870 > http [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=217325288 TSER=121360167  
0.017550 192.168.1.100 -> 146.164.69.2 HTTP GET /img/botao-ensino-selecionado.gif HTTP/1.1  
0.017889 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified  
0.017898 146.164.69.2 -> 192.168.1.100 TCP http > 41871 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=121360167 TSER=217325288 WS=7  
0.017905 192.168.1.100 -> 146.164.69.2 TCP 41871 > http [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=217325288 TSER=121360167  
0.017990 192.168.1.100 -> 146.164.69.2 HTTP GET /img/botao-equipe-nao-selecionado.gif HTTP/1.1  
0.018058 192.168.1.100 -> 146.164.69.2 HTTP GET /img/botao-publicacoes-nao-selecionado.gif HTTP/1.1  
0.018283 192.168.1.100 -> 146.164.69.2 TCP 41872 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=217325288 TSER=0 WS=7  
0.018305 192.168.1.100 -> 146.164.69.2 TCP 41873 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=217325288 TSER=0 WS=7  
0.018441 146.164.69.2 -> 192.168.1.100 TCP http > 41870 [ACK] Seq=1 Ack=736 Win=7296 Len=0 TSV=121360168 TSER=217325288  
0.018777 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified  
0.018785 192.168.1.100 -> 146.164.69.2 TCP 41870 > http [ACK] Seq=736 Ack=175 Win=6912 Len=0 TSV=217325288 TSER=121360168  
0.018789 146.164.69.2 -> 192.168.1.100 TCP http > 41871 [ACK] Seq=1 Ack=740 Win=7296 Len=0 TSV=121360168 TSER=217325288  
0.018829 192.168.1.100 -> 146.164.69.2 HTTP GET /img/botao-seminarios-nao-selecionado.gif HTTP/1.1  
0.019125 146.164.69.2 -> 192.168.1.100 TCP http > 41872 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=121360168 TSER=217325288 WS=7  
0.019135 192.168.1.100 -> 146.164.69.2 TCP 41872 > http [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=217325288 TSER=121360168  
0.019140 146.164.69.2 -> 192.168.1.100 TCP http > 41873 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=121360168 TSER=217325288 WS=7  
0.019146 192.168.1.100 -> 146.164.69.2 TCP 41873 > http [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=217325288 TSER=121360168  
0.019165 192.168.1.100 -> 146.164.69.2 HTTP GET /img/botao-pesquisa-nao-selecionado.gif HTTP/1.1  
0.019173 192.168.1.100 -> 146.164.69.2 HTTP GET /img/botao-pesquisa-selecionado.gif HTTP/1.1  
0.019257 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified  
0.019264 192.168.1.100 -> 146.164.69.2 TCP 41871 > http [ACK] Seq=740 Ack=175 Win=6912 Len=0 TSV=217325288 TSER=121360168  
0.019268 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified  
0.019324 192.168.1.100 -> 146.164.69.2 HTTP GET /img/botao-seminarios-selecionado.gif HTTP/1.1  
0.019833 192.168.1.100 -> 146.164.69.2 TCP 41874 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=217325288 TSER=0 WS=7  
0.020007 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified  
0.020202 146.164.69.2 -> 192.168.1.100 TCP http > 41872 [ACK] Seq=1 Ack=742 Win=7296 Len=0 TSV=121360168 TSER=217325288  
0.020210 146.164.69.2 -> 192.168.1.100 TCP http > 41873 [ACK] Seq=1 Ack=738 Win=7296 Len=0 TSV=121360168 TSER=217325288  
0.020436 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified  
0.020443 192.168.1.100 -> 146.164.69.2 TCP 41872 > http [ACK] Seq=742 Ack=175 Win=6912 Len=0 TSV=217325288 TSER=121360168  
0.020494 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified  
0.020500 192.168.1.100 -> 146.164.69.2 TCP 41873 > http [ACK] Seq=738 Ack=175 Win=6912 Len=0 TSV=217325288 TSER=121360168  
0.020550 146.164.69.2 -> 192.168.1.100 TCP http > 41874 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=121360168 TSER=217325288 WS=7
```

Conexão Persistente

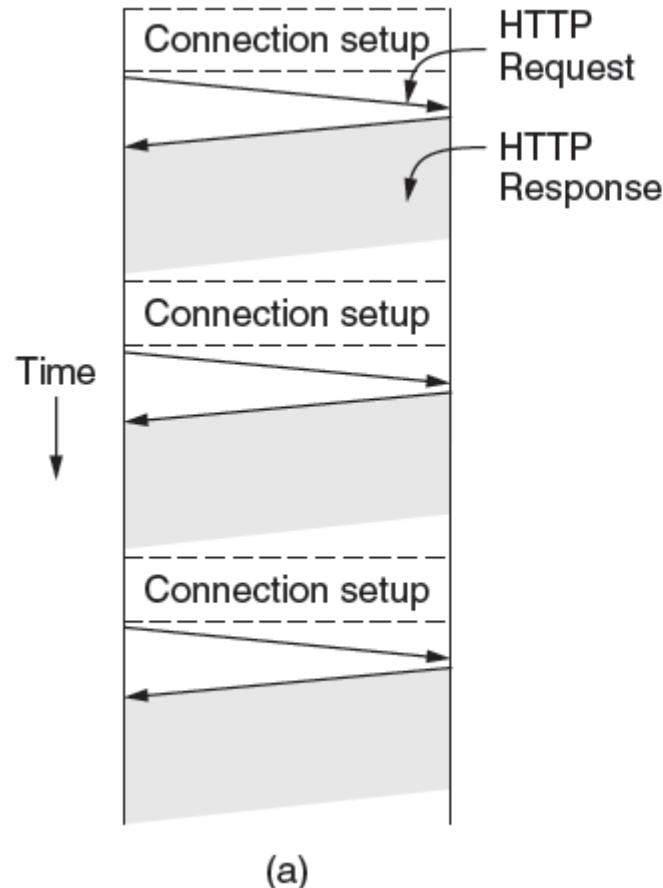
Envio da
página

```
0.022572 192.168.1.100 -> 146.164.69.2 HTTP GET /img/menu-lateral-topo-entrada.gif HTTP/1.1
0.022620 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified
0.022921 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified
0.023232 192.168.1.100 -> 146.164.69.2 HTTP GET /img/botao-projetos-nao-selecionado.gif HTTP/1.1
0.023258 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified
0.023577 192.168.1.100 -> 146.164.69.2 HTTP GET /img/logo-GTA-grande.gif HTTP/1.1
0.023897 192.168.1.100 -> 146.164.69.2 HTTP GET /img/menu-lateral-base.gif HTTP/1.1
0.023918 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified
0.024271 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified
0.024550 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified
0.032164 192.168.1.100 -> 146.164.69.2 HTTP GET /img/botao-projetos-selecionado.gif HTTP/1.1
0.032848 146.164.69.2 -> 192.168.1.100 HTTP HTTP/1.1 304 Not Modified
0.060546 192.168.1.100 -> 146.164.69.2 TCP 41869 > http [ACK] Seq=4301 Ack=3399 Win=26112 Len=0 TSW=217325299 TSER=121360169
0.060551 192.168.1.100 -> 146.164.69.2 TCP 41870 > http [ACK] Seq=2952 Ack=695 Win=10240 Len=0 TSW=217325299 TSER=121360169
0.060553 192.168.1.100 -> 146.164.69.2 TCP 41874 > http [ACK] Seq=1482 Ack=348 Win=8064 Len=0 TSW=217325299 TSER=121360169
0.060838 192.168.1.100 -> 146.164.69.2 TCP 41871 > http [ACK] Seq=2228 Ack=521 Win=9088 Len=0 TSW=217325299 TSER=121360169
0.064852 192.168.1.100 -> 146.164.69.2 TCP 41872 > http [ACK] Seq=2216 Ack=521 Win=9088 Len=0 TSW=217325300 TSER=121360169
0.074722 192.168.1.100 -> 146.164.69.2 TCP 41873 > http [ACK] Seq=2216 Ack=521 Win=9088 Len=0 TSW=217325302 TSER=121360171
15.037081 146.164.69.2 -> 192.168.1.100 TCP http > 41871 [FIN, ACK] Seq=521 Ack=2228 Win=10368 Len=0 TSW=121363922 TSER=217325299
15.037140 146.164.69.2 -> 192.168.1.100 TCP http > 41869 [FIN, ACK] Seq=3399 Ack=4301 Win=14464 Len=0 TSW=121363922 TSER=217325299
15.037197 146.164.69.2 -> 192.168.1.100 TCP http > 41874 [FIN, ACK] Seq=348 Ack=1482 Win=8832 Len=0 TSW=121363922 TSER=217325299
15.037203 146.164.69.2 -> 192.168.1.100 TCP http > 41870 [FIN, ACK] Seq=695 Ack=2952 Win=11776 Len=0 TSW=121363922 TSER=217325299
15.037266 146.164.69.2 -> 192.168.1.100 TCP http > 41872 [FIN, ACK] Seq=521 Ack=2216 Win=10368 Len=0 TSW=121363922 TSER=217325300
15.045012 146.164.69.2 -> 192.168.1.100 TCP http > 41873 [FIN, ACK] Seq=521 Ack=2216 Win=10240 Len=0 TSW=121363925 TSER=217325302
15.076550 192.168.1.100 -> 146.164.69.2 TCP 41871 > http [ACK] Seq=2228 Ack=522 Win=9088 Len=0 TSW=217329053 TSER=121363922
15.076558 192.168.1.100 -> 146.164.69.2 TCP 41869 > http [ACK] Seq=4301 Ack=3400 Win=26112 Len=0 TSW=217329053 TSER=121363922
15.076561 192.168.1.100 -> 146.164.69.2 TCP 41874 > http [ACK] Seq=1482 Ack=349 Win=8064 Len=0 TSW=217329053 TSER=121363922
15.076564 192.168.1.100 -> 146.164.69.2 TCP 41870 > http [ACK] Seq=2952 Ack=696 Win=10240 Len=0 TSW=217329053 TSER=121363922
15.076567 192.168.1.100 -> 146.164.69.2 TCP 41872 > http [ACK] Seq=2216 Ack=522 Win=9088 Len=0 TSW=217329053 TSER=121363922
15.0844545 192.168.1.100 -> 146.164.69.2 TCP 41873 > http [ACK] Seq=2216 Ack=522 Win=9088 Len=0 TSW=217329055 TSER=121363925
19.955841 192.168.1.100 -> 146.164.69.2 TCP 41873 > http [FIN, ACK] Seq=2216 Ack=522 Win=9088 Len=0 TSW=217330272 TSER=121363925
19.955871 192.168.1.100 -> 146.164.69.2 TCP 41870 > http [FIN, ACK] Seq=2952 Ack=696 Win=10240 Len=0 TSW=217330272 TSER=121363922
19.955882 192.168.1.100 -> 146.164.69.2 TCP 41874 > http [FIN, ACK] Seq=1482 Ack=349 Win=8064 Len=0 TSW=217330272 TSER=121363922
19.955893 192.168.1.100 -> 146.164.69.2 TCP 41872 > http [FIN, ACK] Seq=2216 Ack=522 Win=9088 Len=0 TSW=217330272 TSER=121363922
19.955903 192.168.1.100 -> 146.164.69.2 TCP 41871 > http [FIN, ACK] Seq=2228 Ack=522 Win=9088 Len=0 TSW=217330272 TSER=121363922
19.955913 192.168.1.100 -> 146.164.69.2 TCP 41869 > http [FIN, ACK] Seq=4301 Ack=3400 Win=26112 Len=0 TSW=217330272 TSER=121363922
19.956370 146.164.69.2 -> 192.168.1.100 TCP http > 41873 [ACK] Seq=522 Ack=2217 Win=10240 Len=0 TSW=121365153 TSER=217330272
19.956468 146.164.69.2 -> 192.168.1.100 TCP http > 41870 [ACK] Seq=696 Ack=2953 Win=11776 Len=0 TSW=121365153 TSER=217330272
19.956476 146.164.69.2 -> 192.168.1.100 TCP http > 41874 [ACK] Seq=349 Ack=1483 Win=8832 Len=0 TSW=121365153 TSER=217330272
19.956644 146.164.69.2 -> 192.168.1.100 TCP http > 41872 [ACK] Seq=522 Ack=2217 Win=10368 Len=0 TSW=121365153 TSER=217330272
```

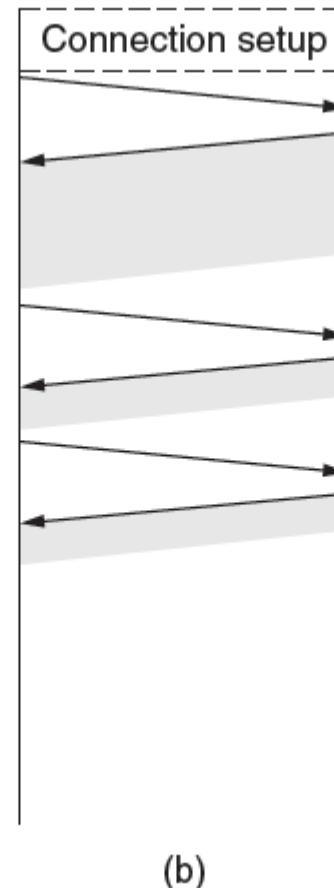
Encerramento
da conexão

Tipos de Conexão

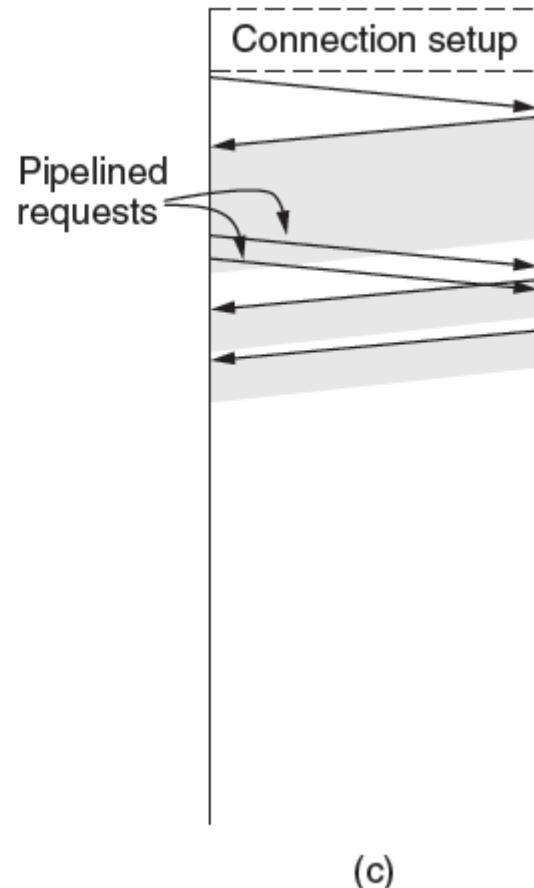
Fonte: Tanenbaum



Conexão não-persistente



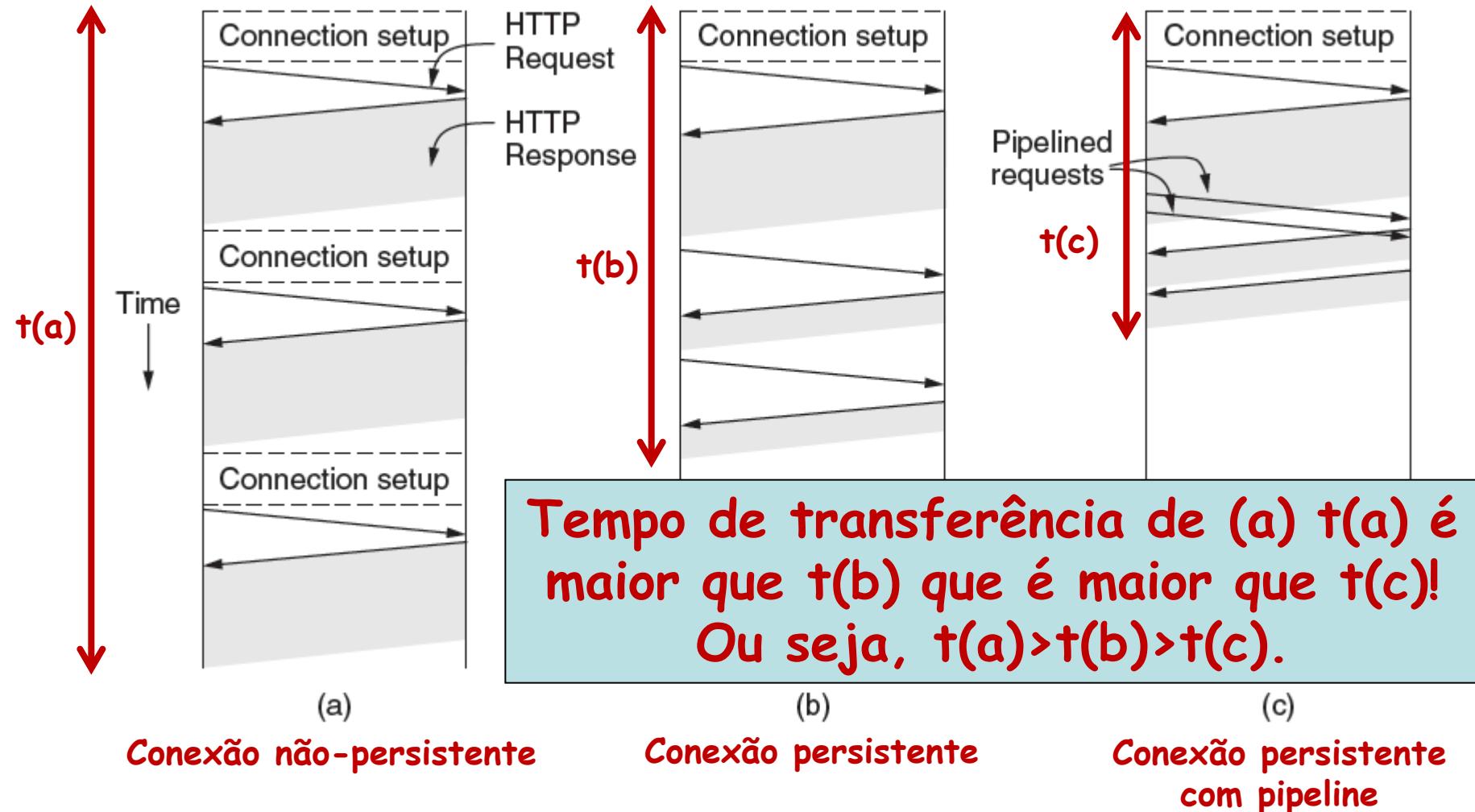
Conexão persistente



Conexão persistente com pipeline

Tipos de Conexão

Fonte: Tanenbaum



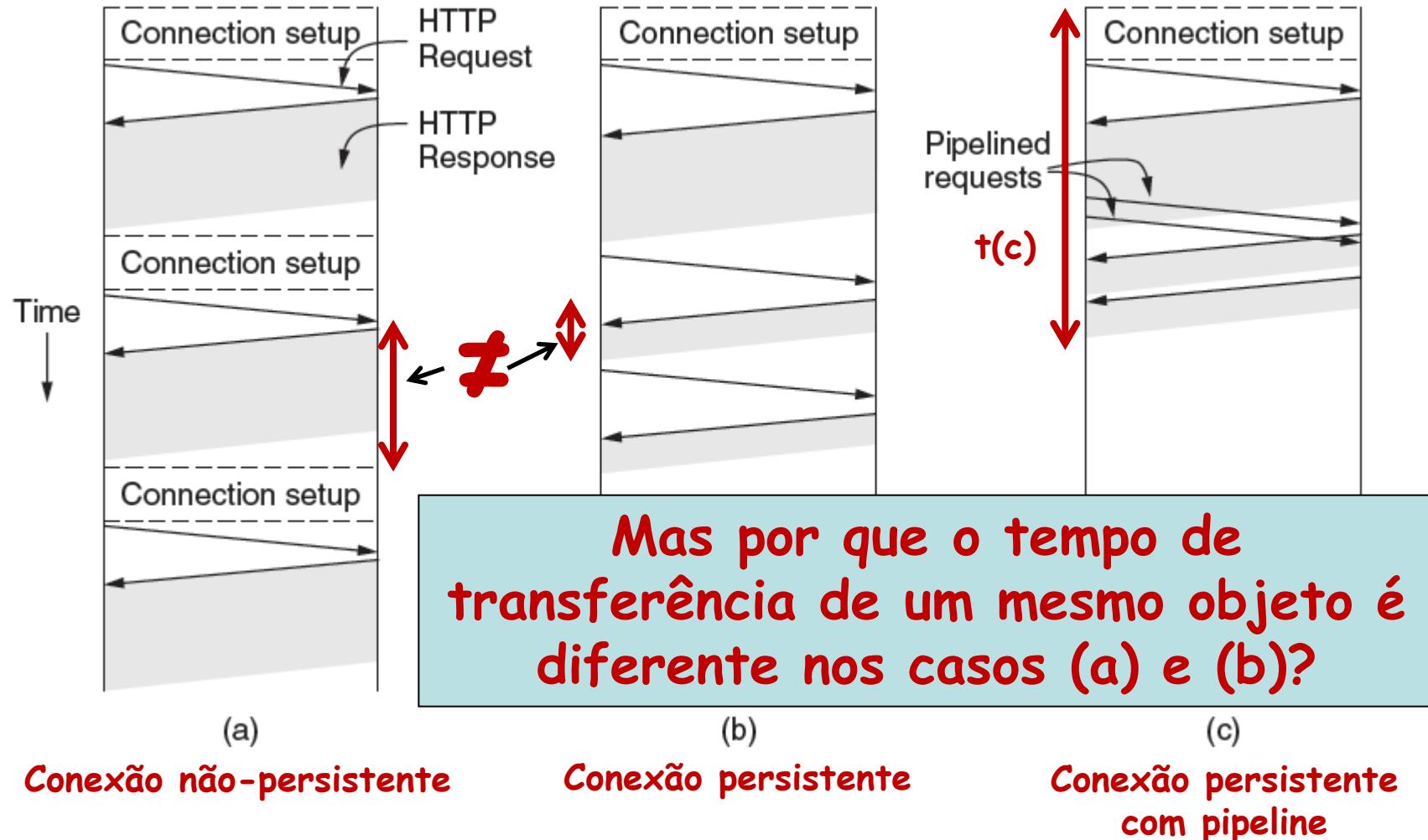
Conexão não-persistente

Conexão persistente

Conexão persistente
com pipeline

Tipos de Conexão

Fonte: Tanenbaum



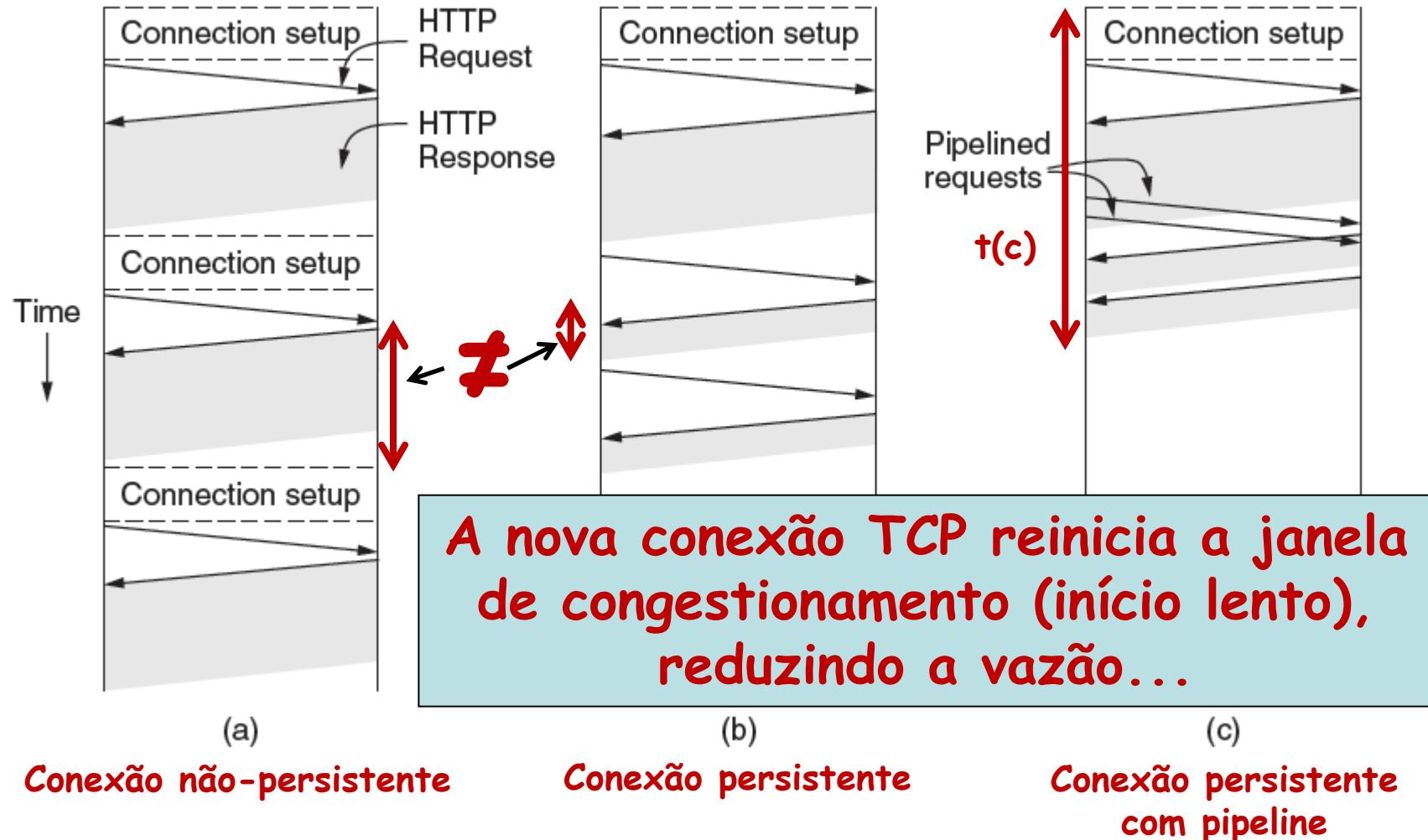
Conexão não-persistente

Conexão persistente

Conexão persistente com pipeline

Tipos de Conexão

Fonte: Tanenbaum



Conexão não-persistente

Conexão persistente

Conexão persistente com pipeline

Formato das Mensagens HTTP

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

linha da requisição

(comandos GET, POST, HEAD;
URL e versão do HTTP)

linhas de
cabeçalho

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

Carriage return,
line feed
indicam fim
de mensagem

(carriage return (CR),
line feed(LF) adicionais)

Formato das Mensagens HTTP

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

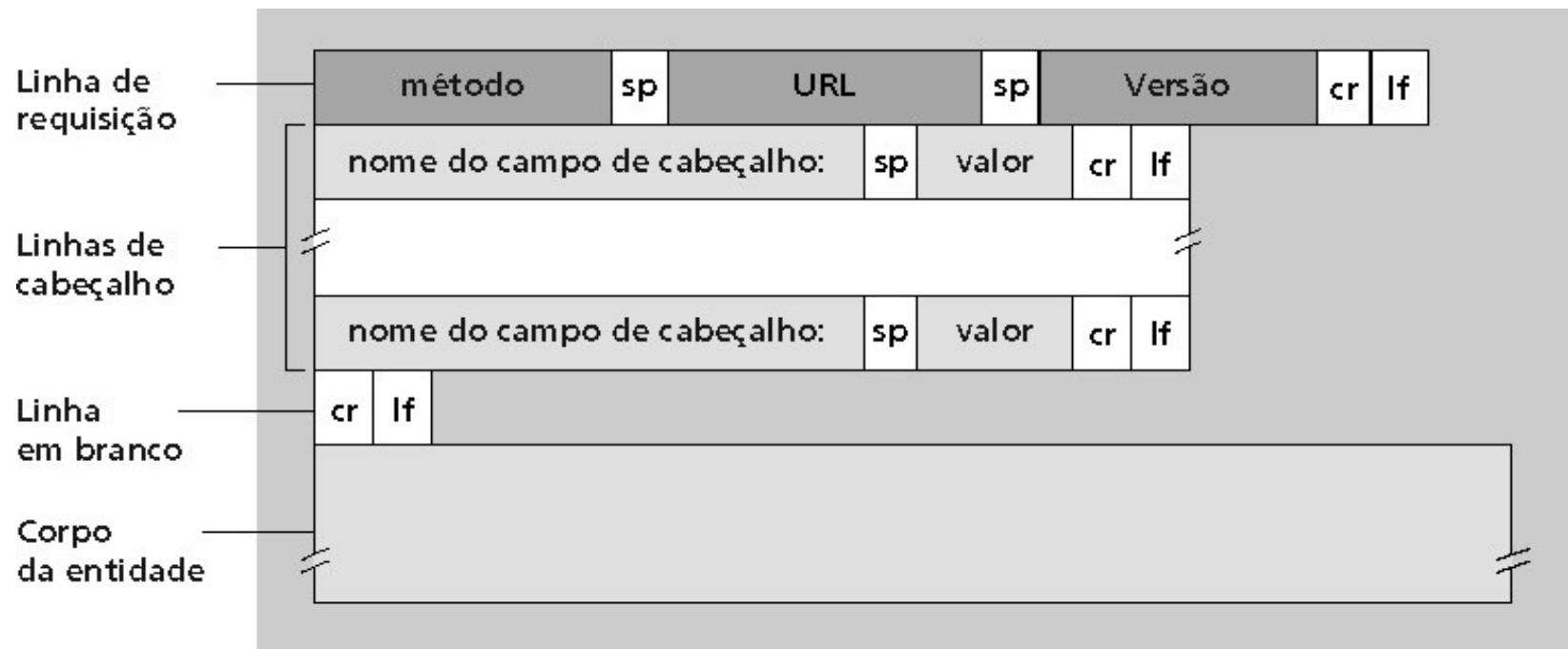
Mesmo usando a versão 1.1, a conexão pode ser fechada por objeto usando a opção
Connection: close

GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr

(carriage return (CR),
line feed(LF) adicionais)

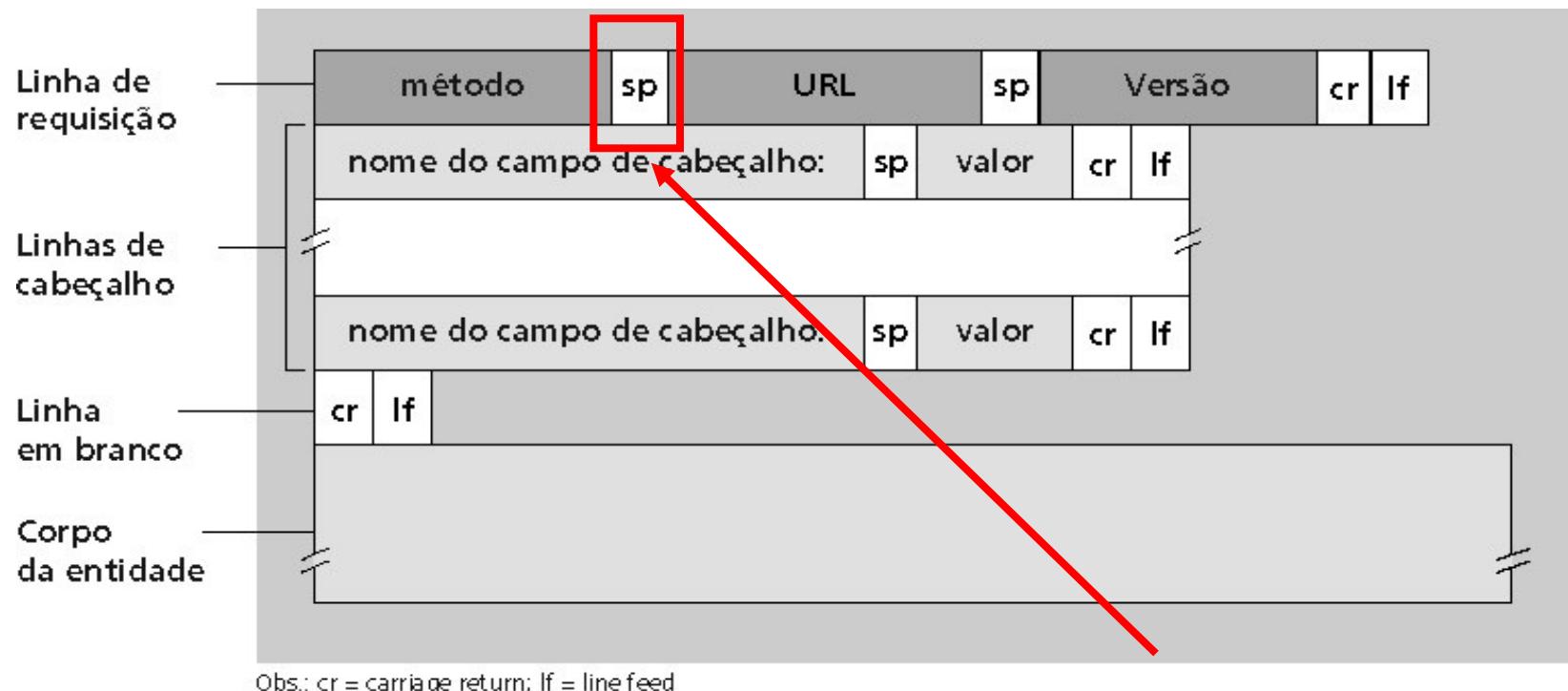
Formato das Mensagens HTTP

- Mensagem de requisição HTTP



Formato das Mensagens HTTP

- Mensagem de requisição HTTP



Space: significa que a linha ainda continua. Em oposição ao cr/lf.

Métodos do HTTP

- Uso do HTTP é principalmente para aplicações web
 - Entretanto, o HTTP possui projeto mais geral com vistas a aplicações futuras
- Métodos determinam o que o servidor deve fazer com a URL fornecida pela requisição de um recurso
 - Oito métodos no HTTP 1.1
 - **GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT**

Detalhes na RFC 2616

Método GET

- Representa a grande maioria das mensagens de requisição HTTP
 - Solicita algum objeto ao servidor e o identifica a partir de uma URL
 - Formato padrão: GET nome-arquivo HTTP/1.1
 - Ex.:

GET /index.html HTTP/1.1

Método HEAD

- Semelhante ao GET
 - Usado para depuração de servidores HTTP
 - Resposta não contém objeto requisitado

Envio de Formulários

- Método POST
 - Páginas Web frequentemente contêm um formulário de entrada
 - Conteúdo é enviado para o servidor no corpo da mensagem
- Método URL
 - Usa o método GET
 - Conteúdo é enviado para o servidor no campo URL

www.somesite.com/animalsearch?key=monkeys&max=10

Formato das Respostas HTTP

linha de estado
(versão do protocolo,
código de estado,
mensagem de estado)

linhas de
cabeçalho

Corpo da entidade
(dados, p.ex., arquivo
html solicitado)

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 ...

Códigos de Estado da Resposta HTTP

- Primeira linha da mensagem de resposta

Código	Significado	Exemplos
1xx	Informação	100 = servidor concorda em tratar da solicitação do cliente
2xx	Sucesso	200 = solicitação com sucesso; 204 = nenhum conteúdo presente
3xx	Redirecionamento	301 = página movida; 304 = página em cache ainda válida
4xx	Erro no cliente	403 = página proibida; 404 = página não localizada
5xx	Erro no servidor	500 = erro interno do servidor; 503 = tente novamente mais tarde

Mais Alguns Códigos de Estado da Resposta HTTP

- Respostas típicas recebidas pelos clientes:

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

Experimento

1. Use cliente telnet:

```
telnet www.gta.ufrj.br 80
```

Abre conexão TCP para a porta 80 (porta padrão do servidor http). Qualquer coisa digitada é enviada para a porta 80 do www.gta.ufrj.br

2. Digite um pedido GET HTTP:

```
GET /~miguel/index.html HTTP/1.1  
Host:gta.ufrj.br
```

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 !

Experimento

```
itaqua:~> telnet gta.ufrj.br 80
Trying 146.164.69.2...
Connected to gta.ufrj.br.
Escape character is '^]'.
GET /~miguel/index.html HTTP/1.1
Host:gta.ufrj.br

HTTP/1.1 200 OK
Date: Mon, 18 Mar 2013 19:55:16 GMT
Server: Apache
Last-Modified: Tue, 13 Sep 2011 21:35:34 GMT
ETag: "141e0f4-c5f-4acd96c6ff580"
Accept-Ranges: bytes
Content-Length: 3167
Vary: Accept-Encoding
Content-Type: text/html

<HTML>
<HEAD>
<title>Miguel Elias Mitre Campista's Home Page</title>
<meta name="KeyWords" content="Miguel Campista, Miguel Elias Campista, COPPE, UFRJ, Miguel Elias M. Campista, Miguel Campista, Campista, M. E. M., Miguel E. M. Campista">
<meta name="Description" content="Miguel Elias Mitre Campista's Home Page">

<table border="0">
<tr>
<td rowspan="3" width="38%"></td>
<td><p style="font-size:18.0pt; font-family:Times New Roman; font-weight:bold" id="topo">Grupo de Teleinform&aacute;tica e Automa&ccedil;&atilde;o - GTA</p></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><p style="font-size:12.0pt; font-family:Times New Roman"><b>Universidade Federal do Rio de Janeiro - UFRJ</b></p></td>
</tr>
</table>
```

• • •

Experimento

• • •

```
<hr size=1 width=80% align=left color=red>

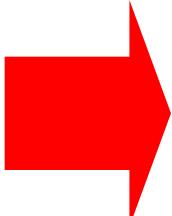
<table border=0 width=80%>
<tr>
<td><p style="font-size:9.0pt; font-style:italic">
    "I am enough of an artist to draw freely upon my imagination. Imagination is
    more important than knowledge. Knowledge is limited. <br>Imagination encircles
    the world." <font style="font-style:normal">Albert Einstein</font></p></td>
</tr>
<tr><td align="center">
<p style="font-size:10.0pt">| <a href="#topo">TOP</a> |
</td></tr>
</table>

</BODY>

</HTML>
Connection closed by foreign host.
itagua:~> █
```

HTTP/2

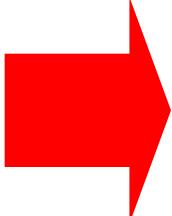
- HTTP/1.0
 - Uma requisição por conexão TCP
- HTTP/1.1
 - Sequência de requisições por conexão
 - Melhora em relação ao 1.0, mas ainda pode ter problemas de bloqueio na cabeceira da fila



Cientes que precisam fazer muitas requisições acabam usando múltiplas conexões com o servidor para conseguir paralelismo e, consequentemente, redução do atraso

HTTP/2

- Além disso...
 - Cabeçalho HTTP é verboso
 - Janela de congestionamento do TCP enche rapidamente
 - Latência aumenta rapidamente com múltiplas requisições
- HTTP/2
 - Permite a intercalação de requisições e respostas na mesma conexão
 - Usa codificação do cabeçalho HTTP



HTTP/2 se torna mais amigável à rede já que um número menor de conexões TCP é necessário

Cookies

- Uma maneira de **guardar estados**
 - HTTP não armazena estados
 - Simplificação do projeto do servidor
 - Reduz problemas de escalabilidade
 - Um conteúdo solicitado duas vezes é enviado duas vezes
- Usado por quase todos os sítios Web
 - Identificação dos usuários
 - Seja para restringir acesso
 - O usuário já se registrou anteriormente?
 - Seja para personalizar a apresentação do conteúdo
 - O usuário já colocou itens em seu carrinho de compras?

Cookies

- Como armazenar estados?
 - Através da observação dos endereços IP dos clientes?

Cookies

- Como armazenar estados?
 - Através da observação dos endereços IP dos clientes?
 - Isso não funcionaria...
 - Endereço IP não identifica o usuário, mas a máquina
 - Endereço IP pode mudar de tempos em tempos, caso o usuário use DHCP
 - Endereço IP pode ser privado, caso o usuário esteja atrás de um NAT

A solução que vem sendo usada é o uso de cookies

Navegação com Cookies

- Quatro componentes principais:
 - Linha de cabeçalho do cookie na mensagem de resposta HTTP
 - Linha de cabeçalho do cookie na mensagem de pedido HTTP
 - Arquivo do cookie mantido na estação do usuário e gerenciado pelo navegador do usuário
 - Banco de Dados de re guarda no sítio Web

Navegação com Cookies

- Arquivo cookie
 - Arquivo pequeno (~4kB) de strings

Domínio	Caminho	Conteúdo	Expira	Seguro
rio-casino.com	/	Cliente=D4829302	15-10-17 17:00	Sim
free-store.com	/	Cart=1-3452;1-35345	10-08-18 8:00	Não
portal.com	/	Prefs=Stk:Cisco	1-05-17 13:00	Não
esperto.com	/	UserID=5283754	15-07-20 16:00	Não

Origem do cookie

Diretórios no servidor que podem usar o cookie (/ significa todos)

Informações armazenadas no formato nome=valor

Navegação com Cookies

- Arquivo cookie
 - Arquivo pequeno (~4kB) de strings

Domínio	Caminho	Conteúdo	Expira	Seguro
rio-casino.com	/	Cliente=D4829302	15-10-17 17:00	Sim
free-store.com	/	Cart=1-3452;1-35345	10-08-18 8:00	Não
portal.com	/	Prefs=Stk:Cisco	1-05-17 13:00	Não
esperto.com	/	UserID=5283754	15-07-20 16:00	Não

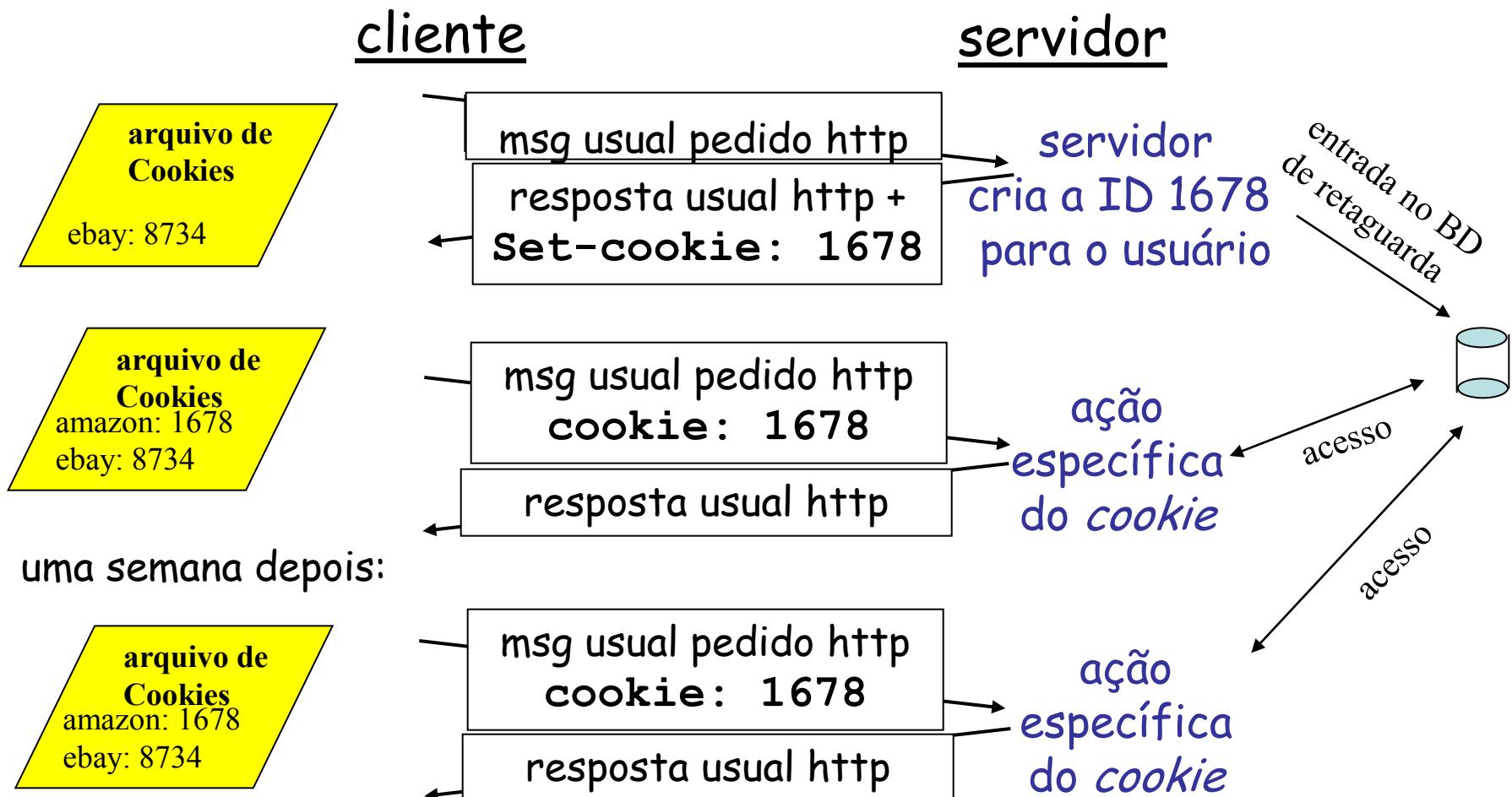
Data de expiração do cookie (cookie persistente). Cookies não persistentes são descartados com a saída do navegador

Indica se o navegador pode retornar o cookie para o servidor usando transporte seguro (SSL/TSL) : Não significa que requer segurança

Navegação com Cookies

- Exemplo
 - Suzana acessa a Internet sempre do mesmo PC
 - Ela visita um sítio específico de comércio eletrônico pela primeira vez
 - Antes da visita, porém, o navegador foi verificar se já existia um cookie para o domínio correspondente
 - Quando os pedidos iniciais HTTP chegam no sítio, o sítio cria
 - Uma ID única
 - Uma entrada para a ID no Banco de Dados de re guarda

Navegação com Cookies



Cookies

- O que os cookies podem obter:
 - Autorização
 - Carrinhos de compra
 - Sugestões
 - Estado da sessão do usuário (*Webmail*)
- Como manter o “estado”:
 - Pontos finais do protocolo: mantêm o estado no transmissor/receptor para múltiplas transações
 - *Cookies*: mensagens http transportam o estado

Cookies e Privacidade

- Cookies permitem que sítios aprendam sobre o usuário
 - Rastreamento das ações do usuário
 - Usuário visita uma página que possui um dado objeto (Cookie com id único é instalado a partir do download de um anúncio em <http://www.qqcoisa.com/file.gif>)
 - Usuário ao se deparar com o mesmo objeto em outra página reenvia o id do cookie recebido anteriormente
- Você pode fornecer nome e e-mail para os sítios
 - Nome e e-mail complementam informações que possivelmente já são conhecidas!
 - P.ex. endereço IP do usuário

Web Caches (Proxies)

- **Meta:** atender pedido do cliente sem envolver servidor de origem
 - Pessoas tendem a retornar a páginas já visitadas ou acessar páginas populares
 - Objetos de uma página não mudam tão rapidamente

Seria necessário obter todos os objetos a partir do servidor de origem?

Web Caches (Proxies)

- Presença do cache diminui o número de requisições ao servidor
 - Usuário configura navegador: acessos Web via proxy
 - Também existem *proxies transparentes e locais* (na própria máquina do cliente)
 - 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

Web Caches (Proxies)

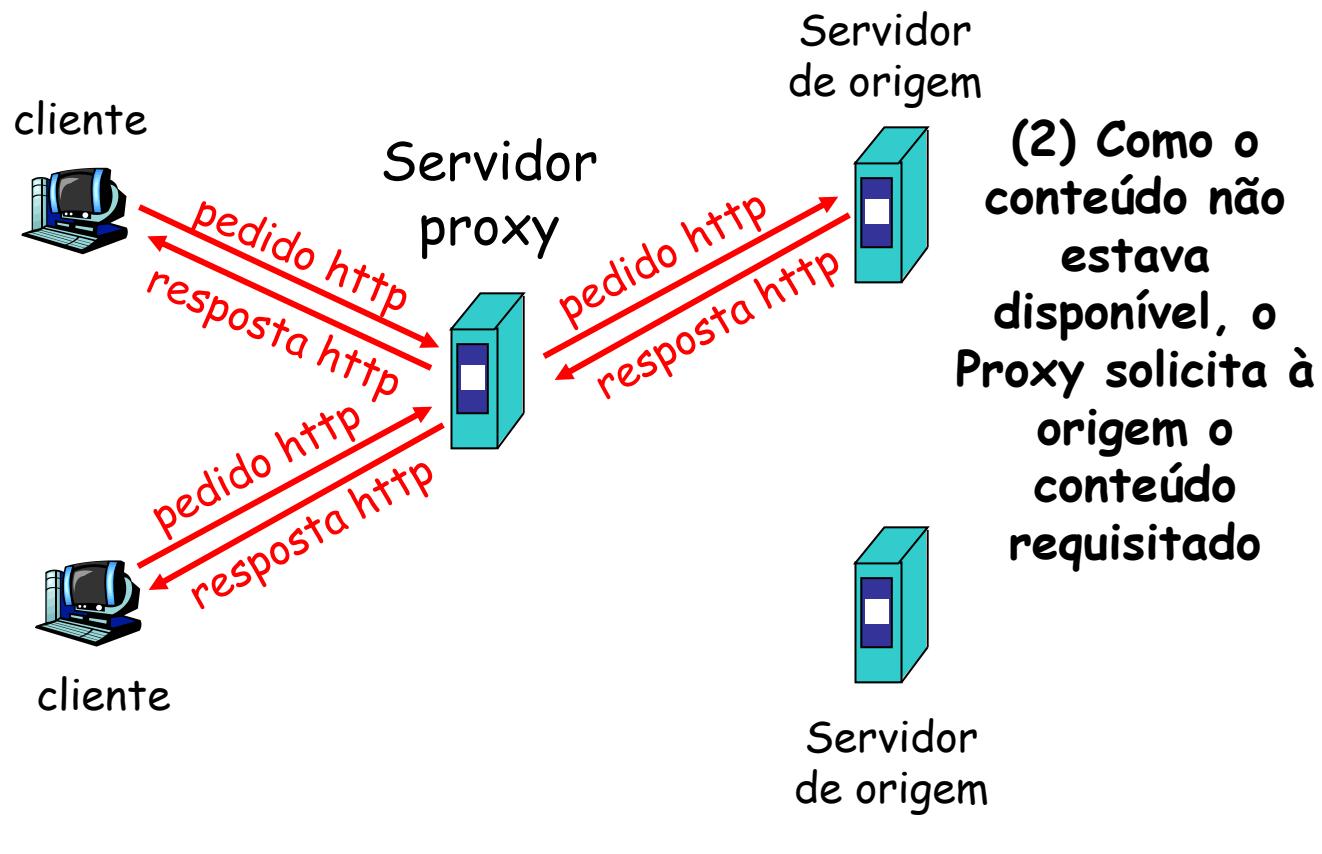
- Quais as vantagens do cache?
 - Redução do tempo de resposta para os pedidos do cliente
 - Redução do tráfego no canal de acesso de uma instituição

Desempenho depende da taxa de acerto (*hit ratio*)!

- Cache atua tanto como cliente quanto como servidor
 - Tipicamente, o cache é instalado por um ISP (universidade, empresa, ISP residencial)
 - ISP também quer reduzir o tráfego que sai da própria rede...

Web Caches (Proxies)

(1) Cliente pede conteúdo que não está no proxy

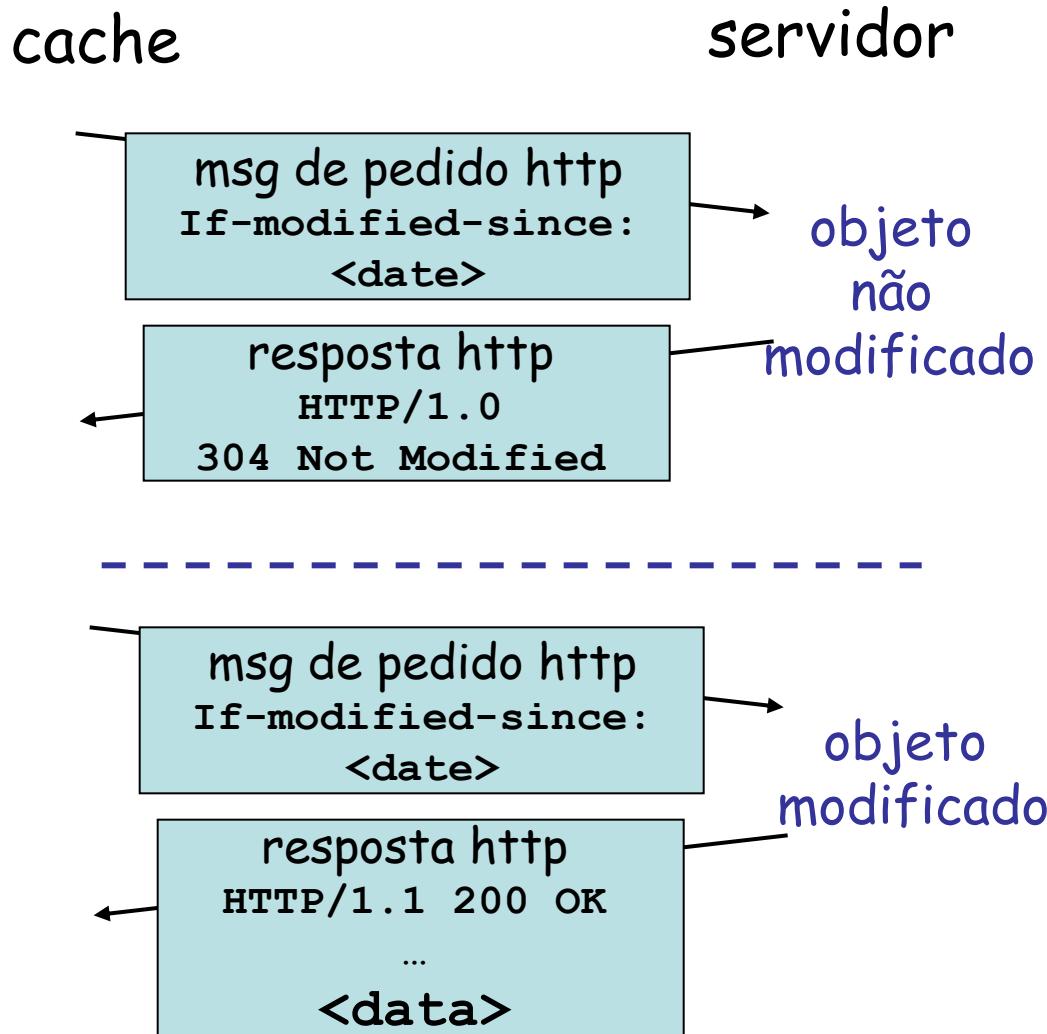


(3) Proxy atende diretamente a nova requisição ao mesmo conteúdo

Método GET Condicional

- **Objetivo:** não enviar objeto se cliente já tem (no cache) versão atual
 - **cache:** especifica data da cópia no cache no pedido http
If-modified-since: <date>
 - **servidor:** resposta não contém objeto se cópia no cache é atual:
HTTP/1.0 304 Not Modified

Método GET Condicional



Busca na Web

- Aplicação web mais bem sucedida até o momento
 - Google, Yahoo!, Bing, etc.
- Algoritmos de busca
 - Podem contar o número de vezes que a palavra-chave aparece na página
 - Página que tiver mais aparições da palavra-chave é a mais importante
 - Podem contar o número de vezes que a página é apontada por outras
 - Página mais apontada é a mais importante

Busca na Web

- Processo de busca:
 1. Submissão da palavra-chave
 2. Execução da busca através de consultas a um banco de dados
 3. Retorno do resultado como uma página dinâmica
- Processo de busca: *Web crawling*
 - A partir das páginas no banco de dados, a busca segue através dos hiperlinks encontrados
 - Páginas dinâmicas são mais difíceis de serem investigadas e possuem conteúdo oculto (*deep Web*)

Desafios da Busca na Web

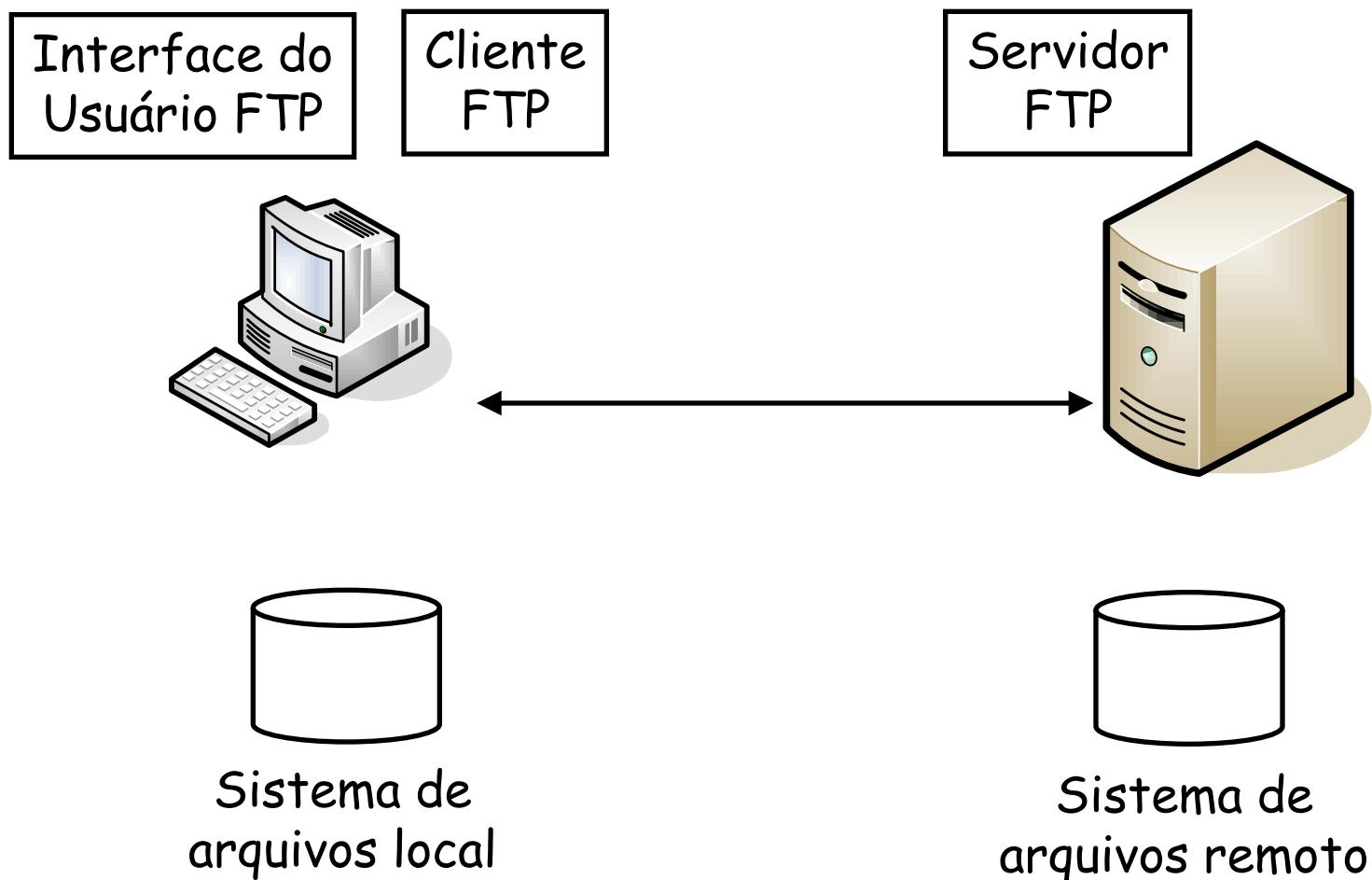
- Quantidade de dados armazenados
 - Bancos de dados armazenam cópias de todas as páginas da parte visível da Web
 - ~20 petabytes (tamanho médio das páginas é 320 kB)
- Formatação dos dados
 - Estrutura de dados pode não ser conhecida
 - Uso do XML para como forma de estruturação
 - Significado dos dados pode ser ainda um problema
- Mudança para o paradigma “orientado a conteúdo”
 - Ao invés do tradicional “centrado no usuário”

File Transfer Protocol (FTP)

Protocolo FTP

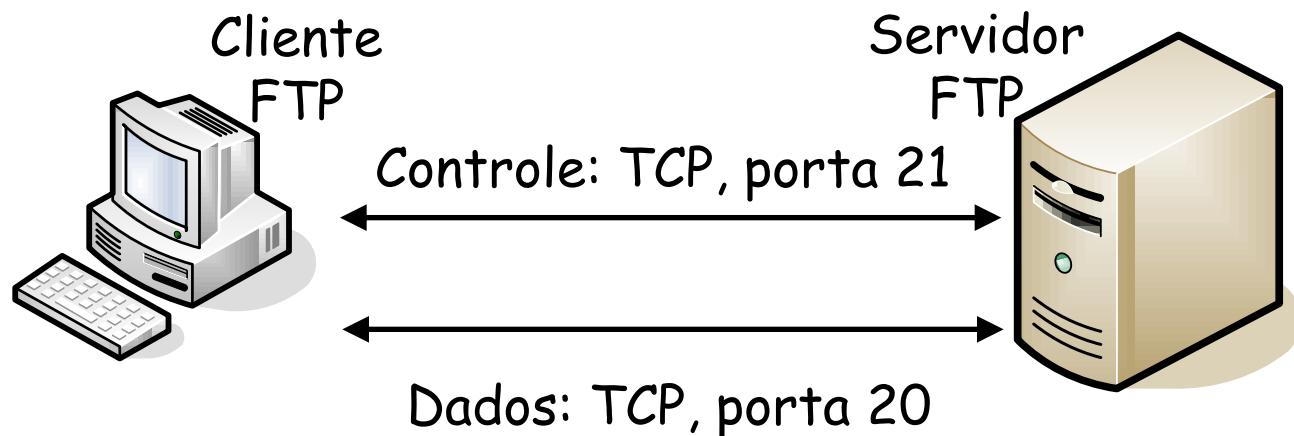
- Transferir um arquivo
 - De um hospedeiro remoto
 - Para um hospedeiro remoto
- Modelo cliente-servidor
 - Cliente
 - Lado que inicia a transferência
 - Pode ser de ou para o sistema remoto
 - Servidor:
 - Hospedeiro remoto

Protocolo FTP



Protocolo FTP

- Conexões separadas
 - Uma para controle
 - Identificação de usuário, senha, comandos para trocar diretório remoto e comandos para pegar e inserir um arquivo
 - Uma para dados
 - Envio do arquivo



Protocolo FTP

- Observações...
 - Para transferir outro arquivo (mesmo se o anterior ainda não tiver terminado)
 - O servidor abre uma segunda conexão TCP
 - Conexão de controle: fora da banda

Funcionamento do FTP

- **Passo 1:** Cliente FTP contata servidor FTP na porta
 - Especifica o TCP como protocolo de transporte
- **Passo 2:** Cliente obtém autorização através da conexão de controle
- **Passo 3:** Cliente consulta o diretório remoto
 - Envia comandos através da conexão de controle
- **Passo 4:** Quando o servidor recebe um comando para a transferência de um arquivo (download ou upload)
 - Ele abre uma conexão de dados TCP para o cliente
- **Passo 5:** Após a transmissão de um arquivo
 - Servidor fecha a conexão

Funcionamento do FTP

- Comandos
 - Enviados em texto ASCII pelo canal de controle
 - USER nome
 - PASS senha
 - LIST
 - Servidor devolve lista de arquivos no atual diretório remoto
 - RETR arquivo
 - Recupera (lê) arquivo no diretório atual do hospedeiro remoto
 - STOR arquivo
 - Armazena (escreve) arquivo no diretório atual do hospedeiro remoto

Funcionamento do FTP

- Códigos de retorno
 - Código e frase de estado (como para o HTTP)
 - 331 Username OK, password required
 - 125 data connection already open; transfer starting
 - 425 Can't open data connection
 - 452 Error writing file

FTP X HTTP

- Protocolos de aplicação usados para troca de arquivos
 - Conexões TCP
 - FTP usa duas conexões em paralelo: Dados e controle
 - HTTP usa apenas uma
 - Informações de controle
 - FTP envia fora da banda
 - HTTP envia na banda
 - Manutenção de estados dos usuários
 - FTP mantém estados
 - Associa uma conexão de controle a um usuário, impactando a escalabilidade do serviço
 - HTTP não mantém estados

Correio Eletrônico na Internet

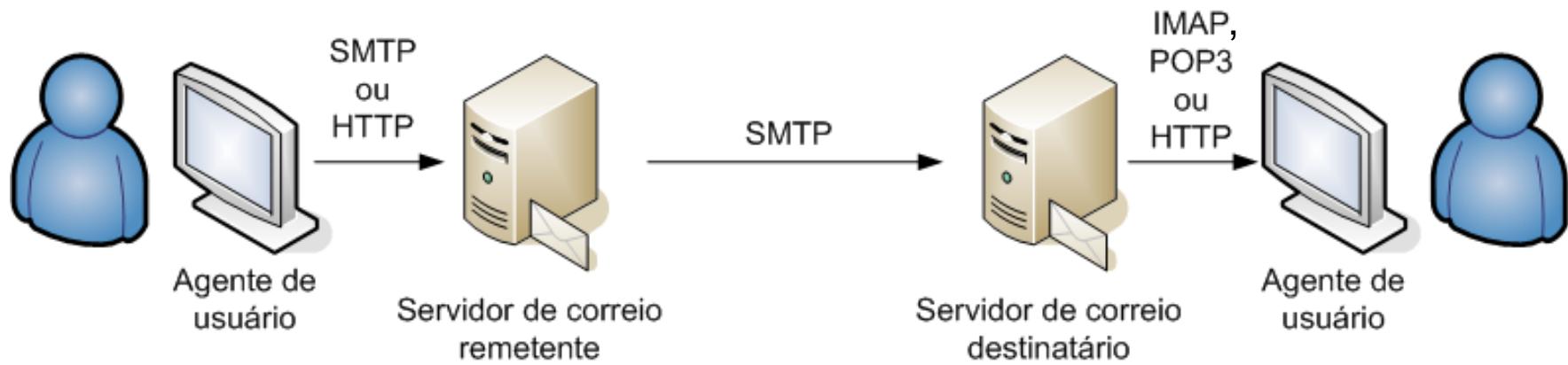
Sistema de Correio da Internet

- Composto por:
 - Agentes de usuário
 - Servidores de correio ou agentes de transferência de mensagens
 - Protocolo simples de transferência de correio
 - *Simple Mail Transfer Protocol (SMTP)*
 - Protocolos de acesso a correio

Sistema de Correio da Internet

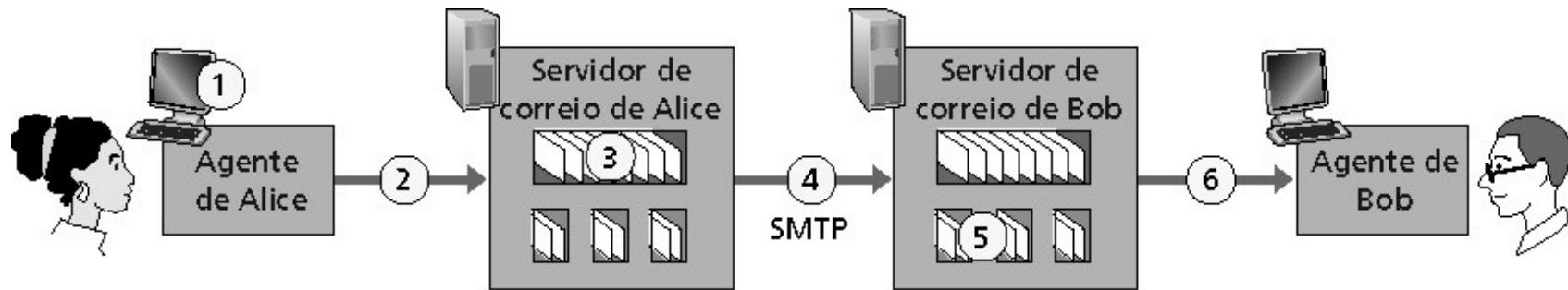
- Agentes de usuário
 - Permitem que usuários leiam, respondam, encaminhem, salvem e editem mensagens
 - Ex.: **Outlook, Eudora, Thunderbird, Mutt**
- Servidores de correio
 - Armazenam as mensagens
 - Se comunicam para realizar a transferência das mensagens
- SMTP
 - Transfere mensagens entre servidores de correio
- Protocolos de acesso a correio
 - Transferem mensagens do servidor de correio para o agente de usuário

Sistema de Correio da Internet



Exemplo: Envio de Mensagem de Alice para Bob

- **Passo 1:** Alice usa o agente de usuário para compor uma mensagem “para” bob@someschool.edu
- **Passo 2:** O agente de usuário de Alice envia a mensagem para o seu servidor de correio
 - A mensagem é colocada na fila de mensagens
- **Passo 3:** O lado cliente do SMTP abre uma conexão TCP com o servidor de correio de Bob



Legenda:



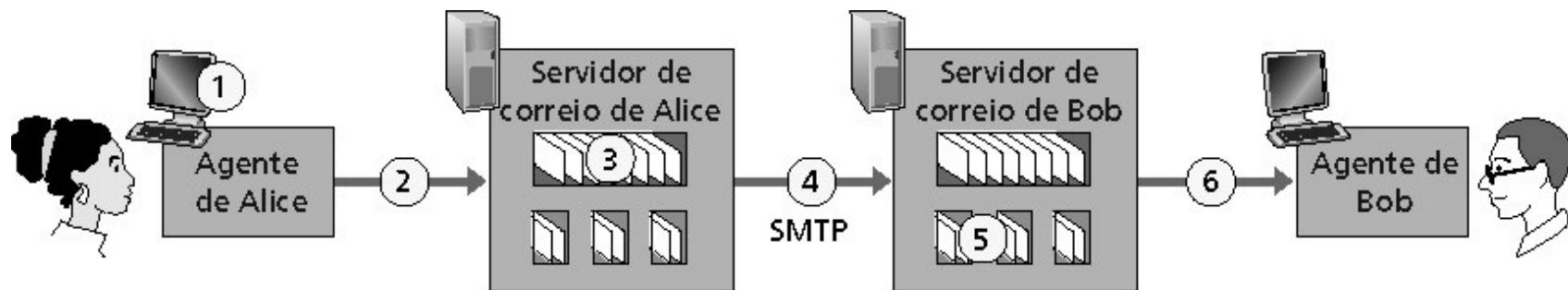
Fila de mensagens



Caixa postal do usuário

Exemplo: Envio de Mensagem de Alice para Bob

- **Passo 4:** O cliente SMTP envia a mensagem de Alice através da conexão TCP
- **Passo 5:** O servidor de correio de Bob coloca a mensagem na caixa de entrada de Bob
- **Passo 6:** Bob chama o seu agente de usuário para ler a mensagem



Legenda:



Fila de mensagens



Caixa postal do usuário

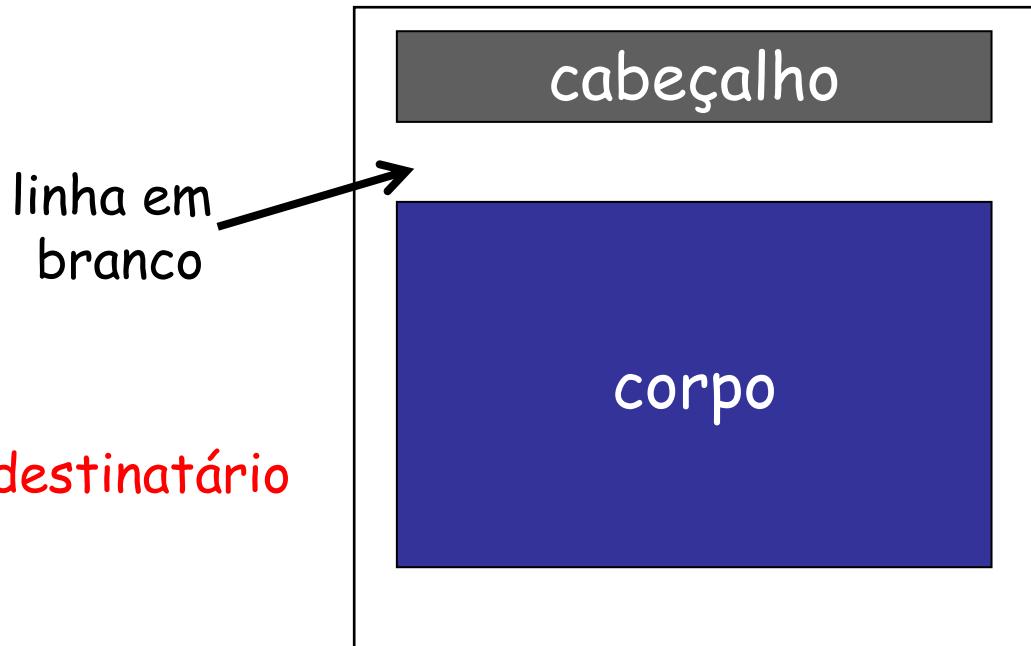
Formato das Mensagens

- Correio eletrônico formado por:
 - Envelope
 - Encapsula a mensagem
 - Contém as informações necessárias para o transporte da mensagem
 - Mensagem
 - Composta de cabeçalho e corpo

Agente cria a mensagem e repassa para o agente de transferência. O agente emprega campos do cabeçalho da mensagem para criar o envelope (mistura entre mensagem e envelope)

Formato das Mensagens

- Mensagem
 - Campos de cabeçalho
 - Linha de texto ASCII contendo o nome do campo, dois pontos e valor, exs.:
 - **From:**
 - **To:**
 - **Subject:**
 - **Received:**
 - Corpo
 - Só diz respeito ao destinatário



Formato das Mensagens

Cabeçalho	Significado
To:	O(s) endereço(s) de correio eletrônico do(s) destinatário(s) principal(is)
Cc:	O(s) endereço(s) de correio eletrônico do(s) destinatário(s) secundário(s)
Cco:	O(s) endereço(s) de correio eletrônico do(s) destinatário(s) oculto(s)
From:	A(s) pessoa(s) que criou(aram) a mensagem
Sender:	O endereço de e-mail do remetente
Received:	A linha incluída por cada agente de transferência ao longo da rota
Return-Path:	Pode ser usado para identificar um caminho de volta ao remetente
...	...

SMTP

- Descrito na RFC 2821
- Usa conexões persistentes
 - Usa o TCP e a porta 25
- Comunicação entre um cliente SMTP (transmissor) e um servidor SMTP (receptor)

SMTP

- Mensagens enviadas
 - Em ASCII (7 bits)
 - Servidor SMTP usa CRLF.CRLF para reconhecer o final da mensagem
 - Uso de extensão ou de codificação para 8 bits
- Mensagens não entregues
 - Gera relatório de erro contendo primeira parte da mensagem
 - Relatório é enviado ao remetente

SMTP

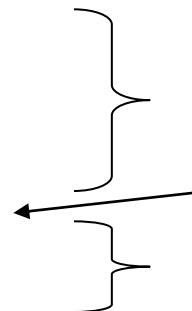
- Utiliza comandos para fazer a comunicação entre servidores
 - Exemplos
 - HELO
 - MAIL FROM
 - RCPT TO
 - DATA
 - QUIT
 - VRFY
- Respostas do servidor são numéricas

Exemplo de Interação Usando telnet

```
S: 220 servidor.br
C: HELO cliente.br
S: 250 Hello cliente.br, pleased to meet you
C: MAIL FROM: <usuario@cliente.br>
S: 250 usuario@cliente.br... Sender ok
C: RCPT TO: <usuario@servidor.br>
S: 250 usuario@servidor.br ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: From: usuario@cliente.br
C: To: usuario@servidor.br
C: Subject: Teste
C:
C: Teste de envio de correio.
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 servidor.br closing connection
```

Exemplo de Interação Usando telnet

```
S: 220 servidor.br
C: HELO cliente.br
S: 250 Hello cliente.br, pleased to meet you
C: MAIL FROM: <usuario@cliente.br>
S: 250 usuario@cliente.br... Sender ok
C: RCPT TO: <usuario@servidor.br>
S: 250 usuario@servidor.br ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: From: usuario@cliente.br
C: To: usuario@servidor.br
C: Subject: Teste
C:
C: Teste de envio de correio.
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 servidor.br closing connection
```



Cabeçalho
Linha em branco
Corpo

Exemplo de Interação Usando telnet

```
itaqua:~> telnet recreio 25
Trying 146.164.69.2...
Connected to recreio.gta.ufrj.br.
Escape character is '^]'.
220 gta.ufrj.br ESMTP Postfix (Debian/GNU)
HELO miguel.br
250 gta.ufrj.br
MAIL FROM: <didi@gta.ufrj.br>
250 2.1.0 Ok
RCPT TO: <miguel@gta.ufrj.br>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>, <CR><LF>
Teste
.
250 2.0.0 Ok: queued as 0F44113E0008
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
itaqua:~>
```

Note que o formato ASCII facilita o envio da mensagem manualmente... Será que isso é por acaso?

Exemplo de Interação Usando telnet

```
itaqua:~> telnet recreio 25
Trying 146.164.69.2...
Connected to recreio.gta.ufrj.br.
Escape character is '^]'.
220 gta.ufrj.br ESMTP Postfix (Debian/GNU)
HELO miguel.br
250 gta.ufrj.br
MAIL FROM: <didi@gta.ufrj.br>
250 2.1.0 Ok
RCPT TO: <miguel@gta.ufrj.br>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>, <CR><LF>
Teste
.
250 2.0.0 Ok: queued as 0F44113E008
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
itaqua:~>
```

Dando uma olhada agora no remetente, ele parece estranho...

Como será que o agente de email deve reagir?

Exemplo de Interação Usando telnet

The screenshot shows the GTA / UFRJ webmail interface. The left sidebar lists various email folders: Caixa de entrada (2817), Rascunhos, Enviados, Spam (791), Lixeira, Archives, 2011, Imap, Drafts, Sent, Trash, mail, Drafts, Mail, Drafts, Sent, Trash, and spam. The main area displays a list of emails in a table format. The columns are Assunto (Subject), Remetente (Sender), Data (Date), and Tamanho (Size). The table includes several entries, with the first two being spam messages from 'didi@gta.ufrj.br' and 'Aldri Santos'. Below the table, a preview pane shows the details of a selected spam message from 'didi@gta.ufrj.br' received 'Hoje 10:35' with the subject '***SPAM***' and the body 'Teste'.

Assunto	Remetente	Data	Tamanho
SPAM	didi@gta.ufrj.br	19.03.2013 10:35	1 KB
Re: [CIA2] Data para o último entregável - 27/0...	Aldri Santos	Hoje 10:28	15 KB
Re: Formulário Copyright	Junior	Hoje 10:08	43 KB
Re: Formulário Copyright	Jose Geraldo Ribeiro Junior	Hoje 10:05	2 KB
Formulário Copyright	Junior	Hoje 09:58	1 KB
Fwd: Call for paper	Luis Henrique M. K. Costa	Hoje 09:35	30 KB
Re: [CIA2] Data para o último entregável - 27/0...	Paulo Pires	Hoje 09:34	8 KB
SPAM The Smart Money is Loading Up	Dick Herrera	Hoje 09:33	3 KB

Exemplo de Interação Usando telnet

- De uns anos pra cá, o servidor de e-mail evoluiu...

```
itaqua:~> telnet recreio 25
Trying 146.164.69.2...
Connected to recreio.gta.ufrj.br.
Escape character is '^]'.
220 gta.ufrj.br ESMTP
HELO didi.br
250 gta.ufrj.br
MAIL FROM: <didi@gta.ufrj.br>
250 2.1.0 Ok
RCPT TO: <miguel@gta.ufrj.br>
550 5.1.0 <didi@gta.ufrj.br>; Sender address rejected: User unknown in local recipient table
```

Exemplo de Interação Usando telnet

- Mudando o exemplo para dar certo...

```
itaqua:~> telnet recreio 25
Trying 146.164.69.2...
Connected to recreio.gta.ufrj.br.
Escape character is '^>'.
220 gta.ufrj.br ESMTP
HELO miguel.br
250 gta.ufrj.br
MAIL FROM: <miguel@gta.ufrj.br>
250 2.1.0 Ok
RCPT TO: <miguel@gta.ufrj.br>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>,<CR><LF>
From: Miguel Elias Mitre Campista <miguel@gta.ufrj.br>
To: miguel@gta.ufrj.br
Subject: Teste
Teste.
.
250 2.0.0 Ok: queued as BC5C3214009F
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

Exemplo de Interação Usando telnet

- Mudando o exemplo para dar certo...

Inbox - Icedove Mail/News

Address Book Tag ▾ From filter

Subject	From	Date
SBRC 2019 - orçamento	Alessandra Leitão	27-03-2017 17:09
Re: Seção 2: Funcionalidades	cruz	27-03-2017 17:26
Re: Seção 2: Funcionalidades	Rodrigo de Souza Couto	27-03-2017 17:37
Re: Seção 2: Funcionalidades	Luis Henrique M. K. Costa	27-03-2017 17:39
Re: Seção 2: Funcionalidades	Luis Henrique M. K. Costa	27-03-2017 17:49
Re: Seção 2: Funcionalidades	cruz	27-03-2017 17:50
Re: Fwd: SBRC 2019 - orçamento	Igor Monteiro Moraes	27-03-2017 17:59
Re: Fwd: SBRC 2019 - orçamento	Igor Monteiro Moraes	27-03-2017 18:56
Re: Seção 2: Funcionalidades	Roberto Gonçalves	27-03-2017 19:36
Re: Seção 2: Funcionalidades	Miguel Elias M. Campista	08:28
[Professores] Fwd: [Coordenacao] Fwd: CAPES-COFECUB	Luis Henrique M K Costa	09:12
Re: Seção 2: Funcionalidades	Luis Henrique M K Costa	09:23
Re: Seção 2: Funcionalidades	Luis Henrique M K Costa	09:23
[Usuariosdacoppe] HOJE - SEMINÁRIO CONTEÚDO LOCAL DA INDÚSTRIA DO PETRÓLEO	Luiz Pinguelli Rosa	09:29
Rascunho gráfico rho btw - random walk vs alpha (powerlaw)	Dianne Scherly	09:34
Re: Rascunho gráfico rho btw - random walk vs alpha (powerlaw)	Dianne Scherly	10:10
Teste	Miguel@gta.ufrj.br	10:14
The Institute Online: How U.S. Immigration Policies Will Impact Engineering Education	IEEE The Institute	10:15
Teste	miguel@gta.ufrj.br	10:17
Teste	Miguel Elias Mitre Campista	10:19

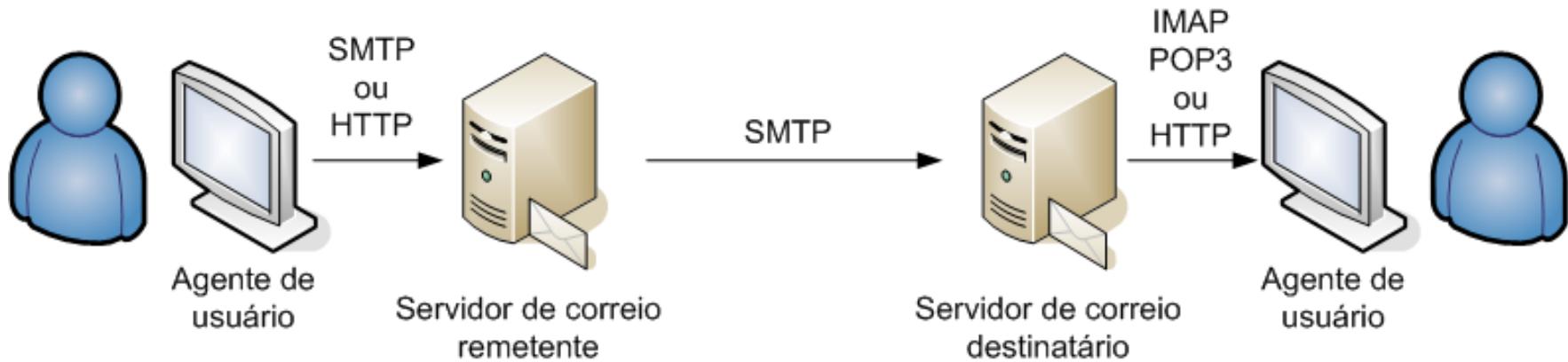
from You★
subject Teste
to You★
Teste.

reply forward archive

SMTP x HTTP

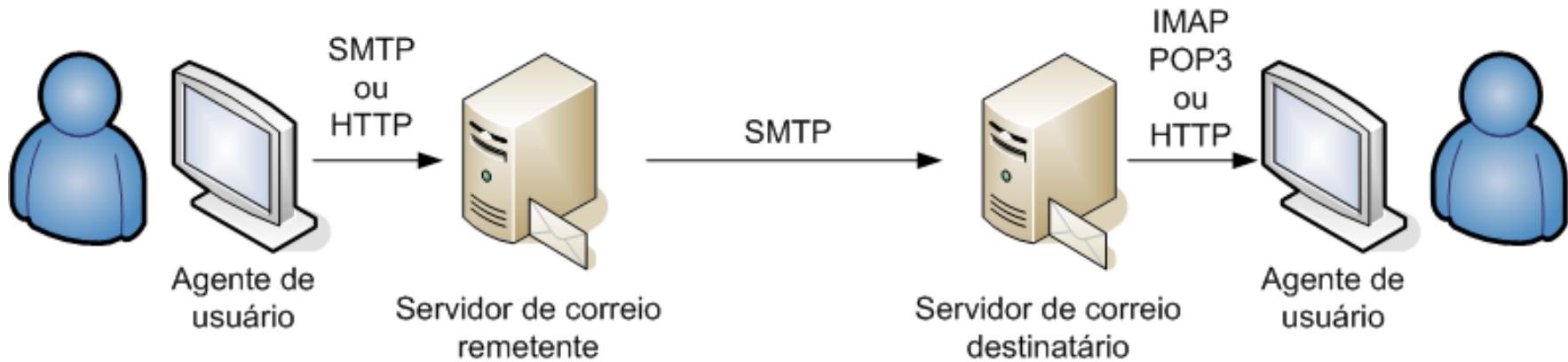
- HTTP: Recupera os dados (*pull*)
- SMTP: Envia os dados (*push*)
- Ambos têm interação comando/resposta e códigos de estado em ASCII
- HTTP
 - Cada objeto é enviado em sua própria mensagem de resposta
- SMTP
 - Múltiplos objetos enviados numa mensagem de múltiplas partes
 - *Tipo multipart, possível conteúdo MIME (Multipurpose Internet Mail Extensions)*

Protocolos de Acesso ao Correio



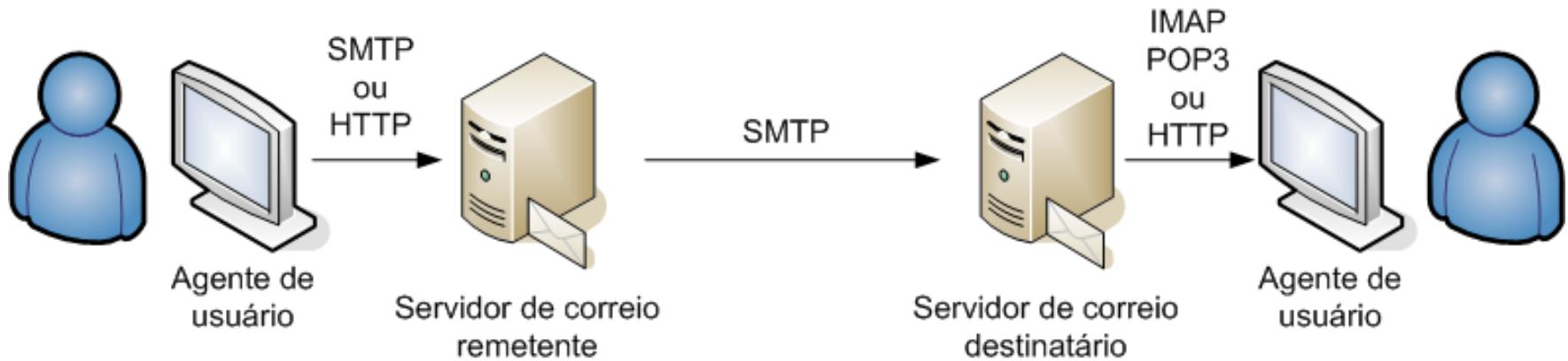
- **SMTP**
 - Entrega/armazena no servidor do receptor
- **Protocolo de acesso ao correio**
 - Recupera do servidor

Protocolos de Acesso ao Correio



- Pergunta:
 - Um servidor SMTP poderia ser executado na própria máquina do cliente?
 - Sim, mas o serviço teria que estar disponível 24/7...
 - Isso não é prático, o que leva ao uso dos protocolos de acesso ao correio!
 - Ainda, o usuário pode querer ter acesso aos e-mails remotamente

Protocolos de Acesso ao Correio



- Pergunta:
 - Um remetente poderia enviar seu e-mail diretamente ao servidor de correio do destinatário?
 - Sim, mas se algum problema ocorrer com servidor do destinatário, o remetente pode não conseguir retransmitir...
 - Isso não é prático, o que leva ao uso dos protocolos de acesso ao correio!

Protocolos de Acesso ao Correio

- SMTP é um protocolo para envio de dados (*push*)
 - Logo, é necessário algum protocolo para recuperação de dados...
 - POP: Post Office Protocol [RFC 1939]
 - Autorização (agente <-->servidor) e transferência
 - IMAP: Internet Mail Access Protocol [RFC 1730]
 - Mais comandos (mais complexo)
 - Manuseio de mensagens **armazenadas no servidor**
 - HTTP
 - Hotmail , Yahoo! Mail, Webmail, etc.

Protocolo POP versão 3 (POP3)

fase de autorização

- comandos do cliente:
 - **user**: declara nome
- **pass**: senha
- servidor responde
 - +OK
 - -ERR

fase de transação, cliente:

- **list**: lista números das msgs
- **retr**: recupera msg por número
- **dele**: apaga msg
- **quit**

```
S: +OK POP3 server ready
C: user ana
S: +OK
C: pass faminta
S: +OK user successfully logged on

C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3

- O exemplo anterior usa o modo “*download e delete*”
 - Bob não pode reler as mensagens se mudar de cliente
- “*Download-e-mantenha*”: copia as mensagens em clientes diferentes
 - Bob pode reler as mensagens se mudar de cliente
- POP3 não mantém estado entre conexões

Internet Message Access Protocol (IMAP)

- Protocolo de acesso ao correio assim como o POP3
 - Mais poderoso que o POP3, porém mais complexo
- Associa cada uma das mensagens a uma pasta
 - Permite ao usuário organizar as mensagens
 - POP3 não possui essa facilidade
 - Quando uma mensagem chega, ela é associada a pasta INBOX
- Mantém o estado do usuário entre sessões:
 - Nomes das pastas e respectivas mensagens estão associadas

Webmail

- Envio e recuperação de mensagens entre remetente e servidor de correio do remetente:
 - Realizado com HTTP
- Comunicação entre o servidor de correio do remetente e do destinatário:
 - Realizado com SMTP

Aumenta a acessibilidade ao correio eletrônico visto que não é necessário a presença de um agente de usuário específico

Domain Name System (DNS)

Identificadores

- Uma pessoa qualquer...
 - Possui várias formas de identificação
 - Nome
 - Carteira de identidade
 - CPF
 - Carteira de motorista
 - Etc.

A identificação usada é a mais adequada a um dado contexto

Identificadores

- Estações e roteadores na Internet

- Endereço IP (ex.: 146.164.69.2)

- Conjunto de bits
 - Tamanho fixo
 - Estrutura hierárquica
 - Pouco intuitivo para os usuários

Bom para
uma máquina

- Nome (ex.: www.gta.ufrj.br)

- Tamanho variável
 - Intuitivo para os usuários
 - Independente da máquina

Bom para
um humano

O que fazer?

Identificadores

- Estações e roteadores na Internet

- Endereço IP (ex.: 146.164.69.2)

- Conjunto de bits
 - Tamanho fixo
 - Estrutura hierárquica
 - Pouco intuitivo para os usuários

Bom para
uma máquina

- Nome (ex.: www.gta.ufrj.br)

- Tamanho variável
 - Intuitivo para os usuários
 - Independente da máquina

Bom para
um humano

Mapeamento

DNS (*Domain Name System*)

- Mapeamento entre nomes de domínio e endereços IP
 - Também faz o inverso: DNS reverso
- É composto por:
 - Base de dados distribuída entre diferentes servidores
 - Organização hierárquica
 - Protocolo da camada de aplicação
 - Nós se comunicam para resolver nomes
 - Utiliza UDP e porta 53
- Mais um exemplo do princípio da Internet
 - Complexidade na borda da rede

DNS (*Domain Name System*)

- Serviços
 - Traduz um nome para um endereço IP
 - Permite o uso de “apelidos” para os nós (*aliasing*)
 - Servidores, estações, roteadores, etc.
 - Mapeamento de nomes canônicos e apelidos
 - Distribuição de carga
 - Conjunto de endereços IP mapeados em apenas um nome
 - Ex.: servidores Web replicados

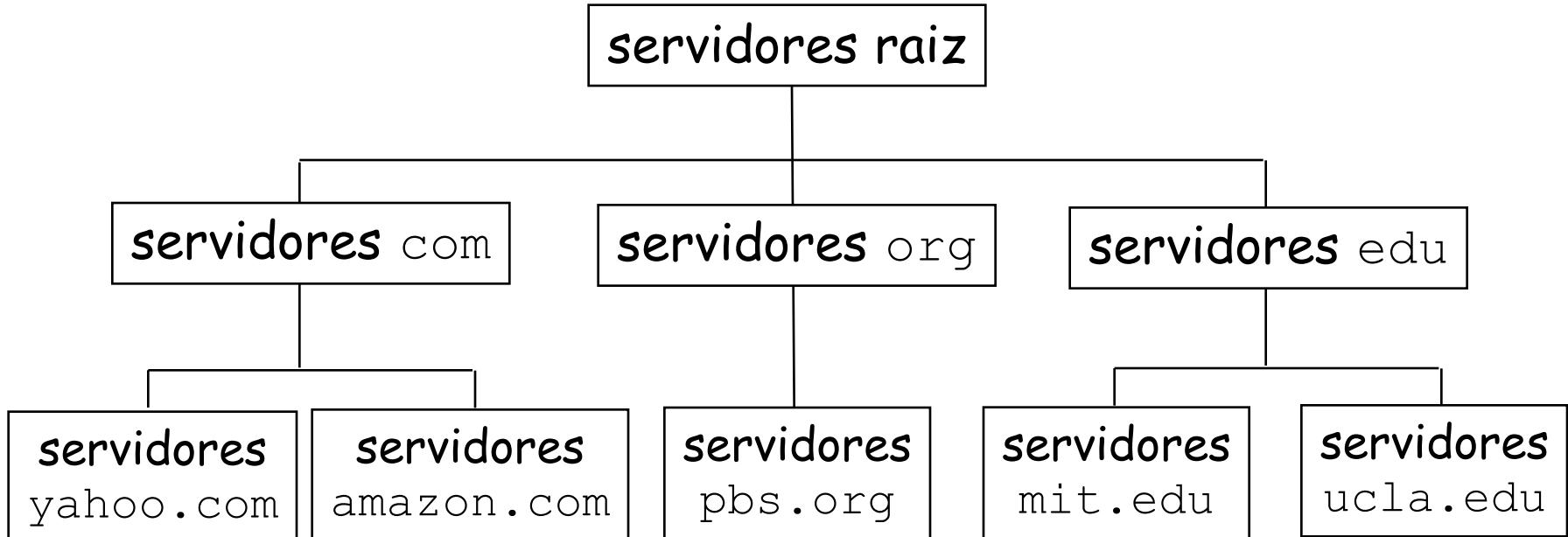
DNS (*Domain Name System*)

- ARPANET: Tradução com um único arquivo (*hosts.txt*)
 - Todos os nomes e endereços IP eram listados
- Por que a base de dados **centralizada** foi substituída?
 - Ponto único de falha
 - Volume de tráfego
 - Requisições e respostas
 - Distância para um usuário
 - Maior tempo de resposta caso o usuário esteja em um ponto distante do planeta
 - Manutenção
 - Como parar o sistema de DNS?

Problemas de escalabilidade

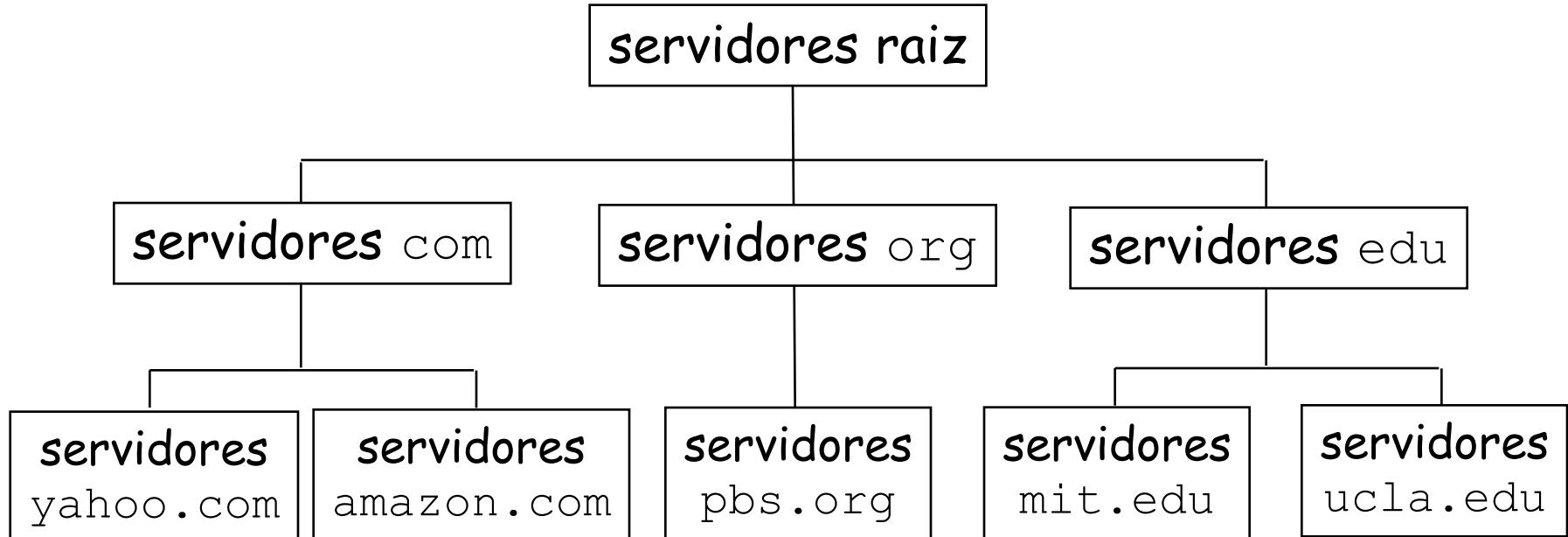
DNS (*Domain Name System*)

- Solução adotada: base de dados **distribuída** e **hierárquica**



DNS (*Domain Name System*)

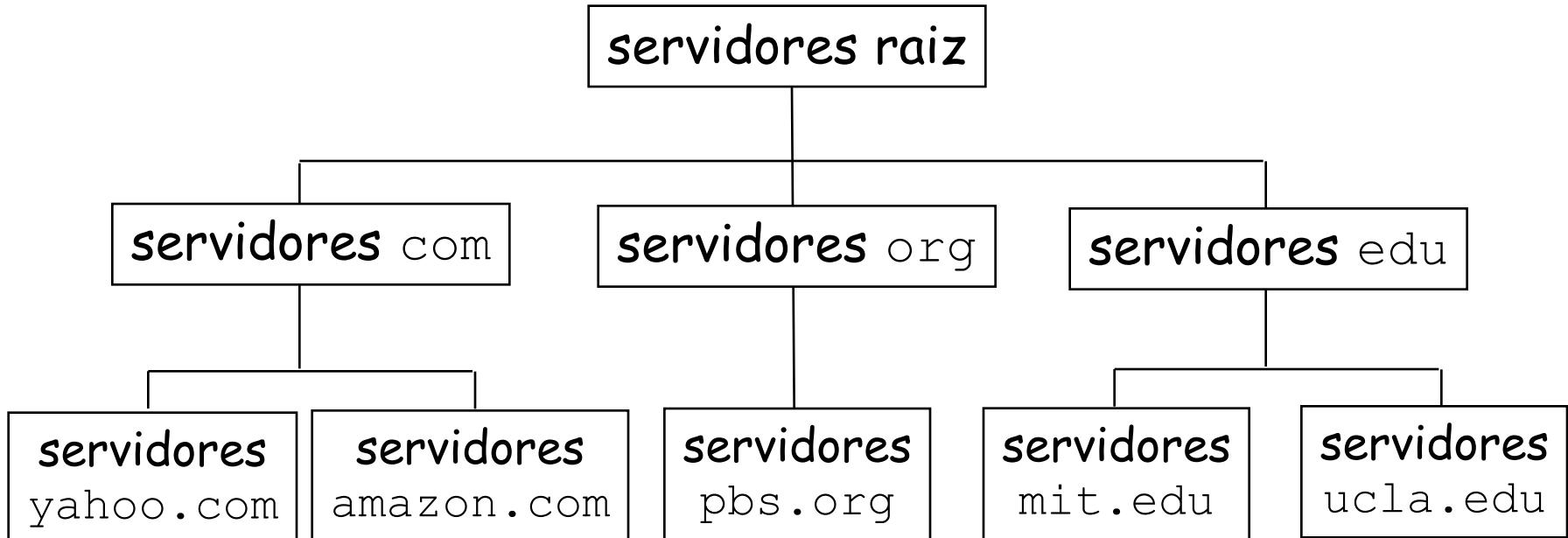
- Solução adotada: base de dados **distribuída** e **hierárquica**



Cliente quer acessar amazon.com

DNS (*Domain Name System*)

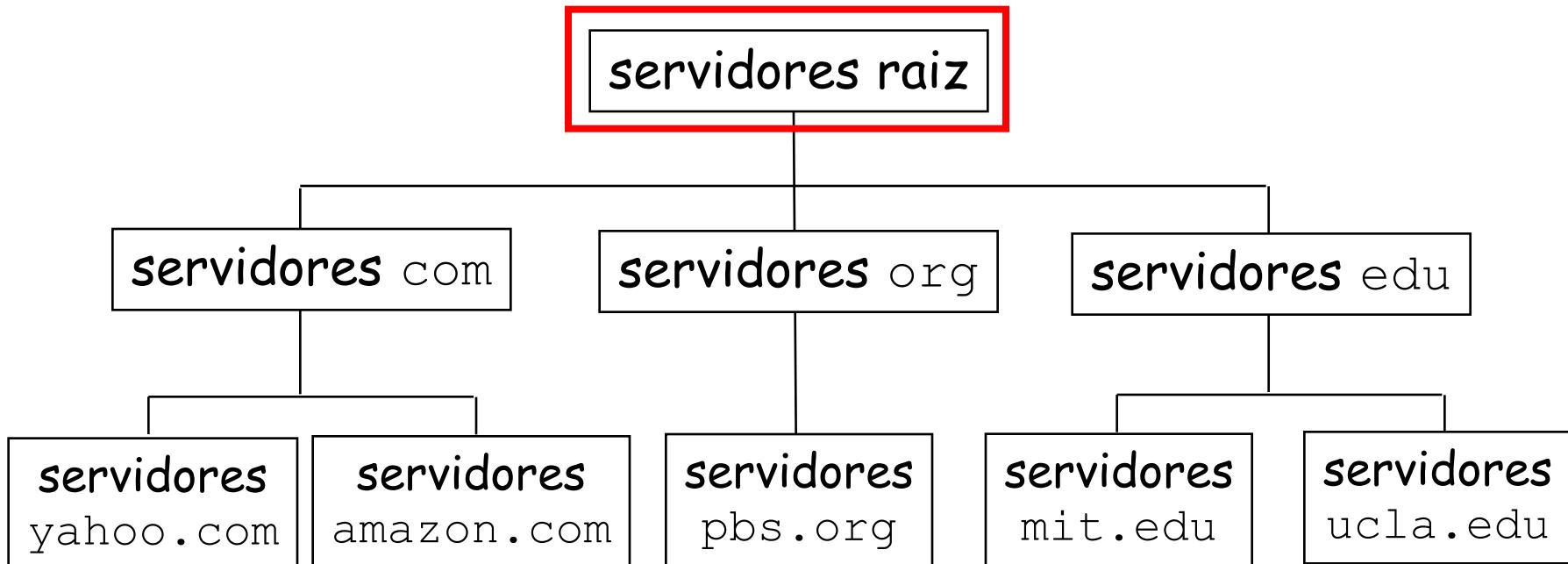
- Solução adotada: base de dados **distribuída** e **hierárquica**



Descobrir o endereço IP de `amazon.com`

DNS (*Domain Name System*)

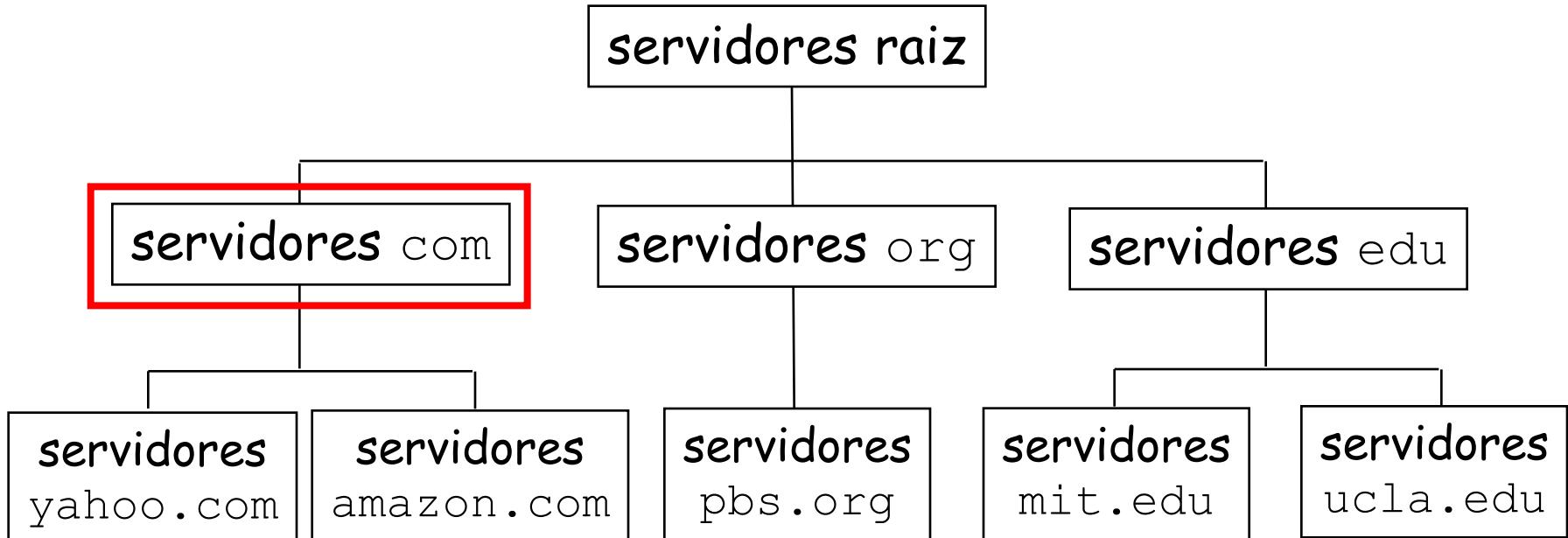
- Solução adotada: base de dados **distribuída** e **hierárquica**



Consulta ao servidor raiz para descobrir o servidor .com

DNS (*Domain Name System*)

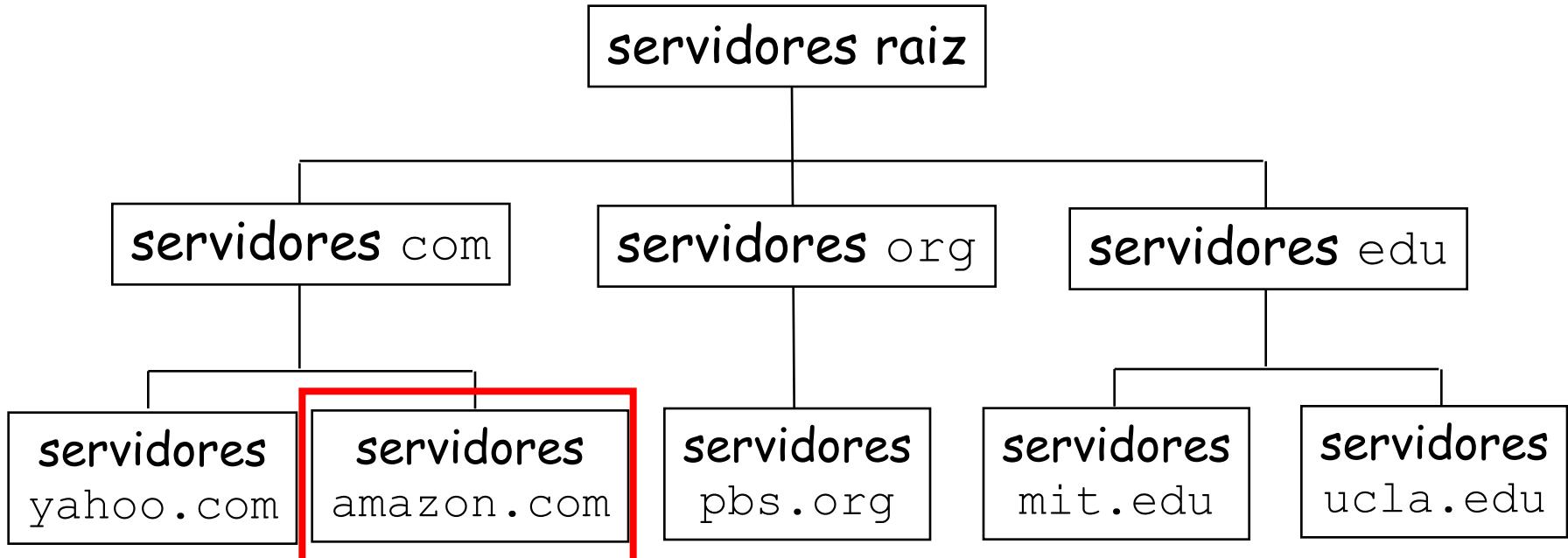
- Solução adotada: base de dados **distribuída** e **hierárquica**



**Consulta ao servidor .com para descobrir o servidor
amazon.com**

DNS (*Domain Name System*)

- Solução adotada: base de dados **distribuída** e **hierárquica**



Cliente consulta servidor DNS do domínio `amazon.com` para obter endereço IP de `www.amazon.com`

Servidores Raiz

- Controlados pelo ICANN (Internet Corporation for Assigned Names and Numbers)
- Ao receber uma consulta
 - Procura o servidor responsável pelo mapeamento no nível imediatamente inferior
 - Esse procedimento é realizado de maneira recursiva até que o servidor oficial que conheça o mapeamento seja encontrado

Servidores Raiz

- 13 ao redor do mundo
 - 10 somente nos EUA



Servidores Raiz

- 13 ao redor do mundo
 - Um pode ter várias réplicas espalhadas



Servidores de Domínio de Alto Nível

- Servidores TLD (*Top-level Domain*)
 - Controlados pelos registradores apontados pelo ICANN
- Responsáveis por:
 - Domínios como com, org, net, edu, ...
 - Todos os domínios de países como br, uk, fr, ca, jp
- Network Solutions manteve servidores para domínio .com
 - Monopólio até 1999
- NIC.br (Registro .br) para domínio .br

Servidores Oficiais (Authoritative)

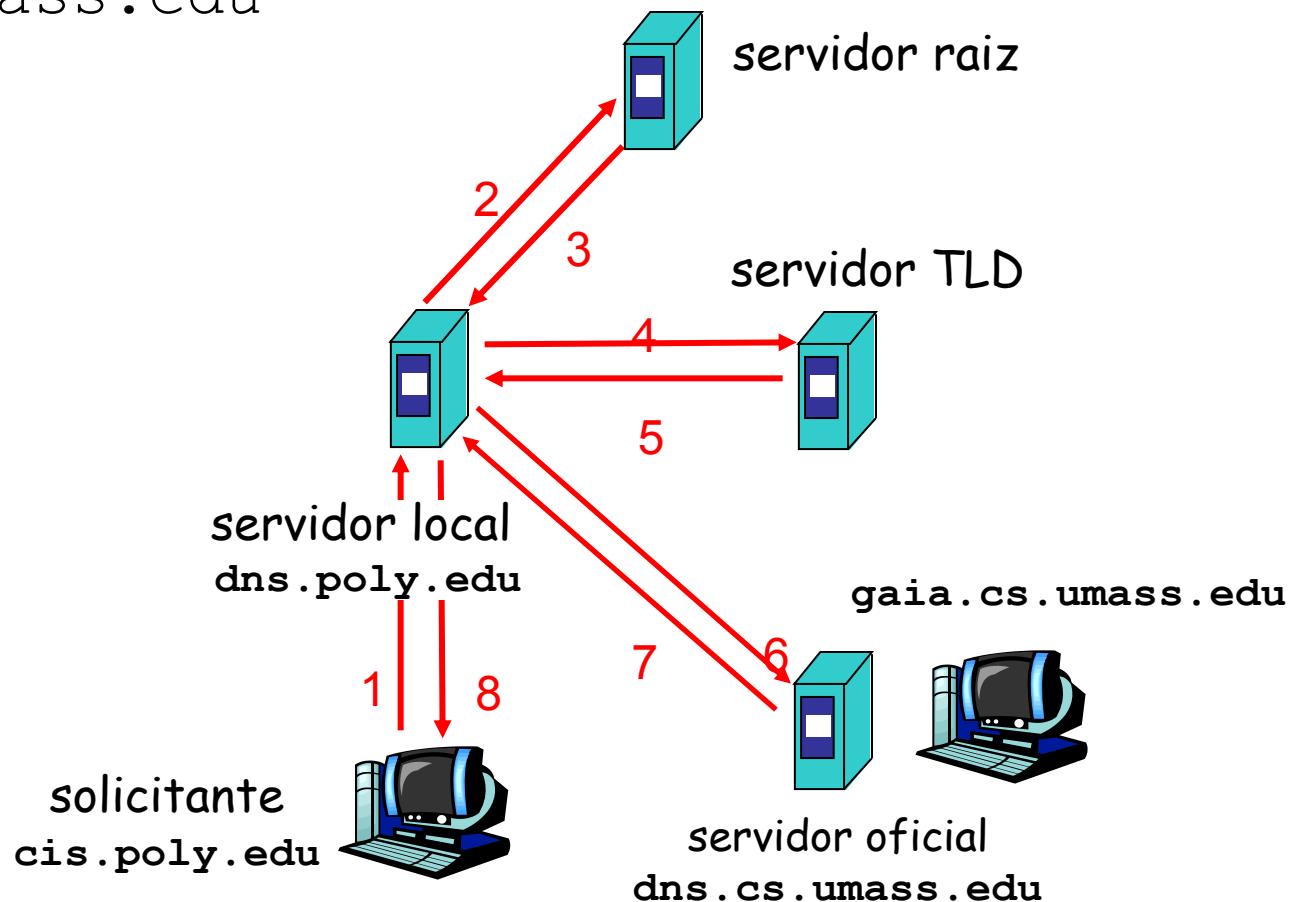
- São os servidores de DNS das organizações
 - Mapeamentos oficiais entre nomes e endereços IP
 - Inclusive para outros servidores da organização (ex., Web e correio)
 - Podem ser mantidos pelas organizações ou pelo provedor de acesso

Servidor de Nomes Local

- Não pertence necessariamente à hierarquia
- Cada “provedor” possui um
 - ISP residencial, empresa, universidade, etc.
 - Também chamado de **servidor de nomes padrão**
- Quando uma estação faz uma consulta DNS
 - Ela é primeiro enviada para o seu **servidor local**
 - Atua como um intermediário
 - O servidor local é quem consulta os demais servidores da hierarquia

Exemplo de Resolução de Nome pelo DNS

- Estação em cis.poly.edu quer endereço IP para gaia.cs.umass.edu



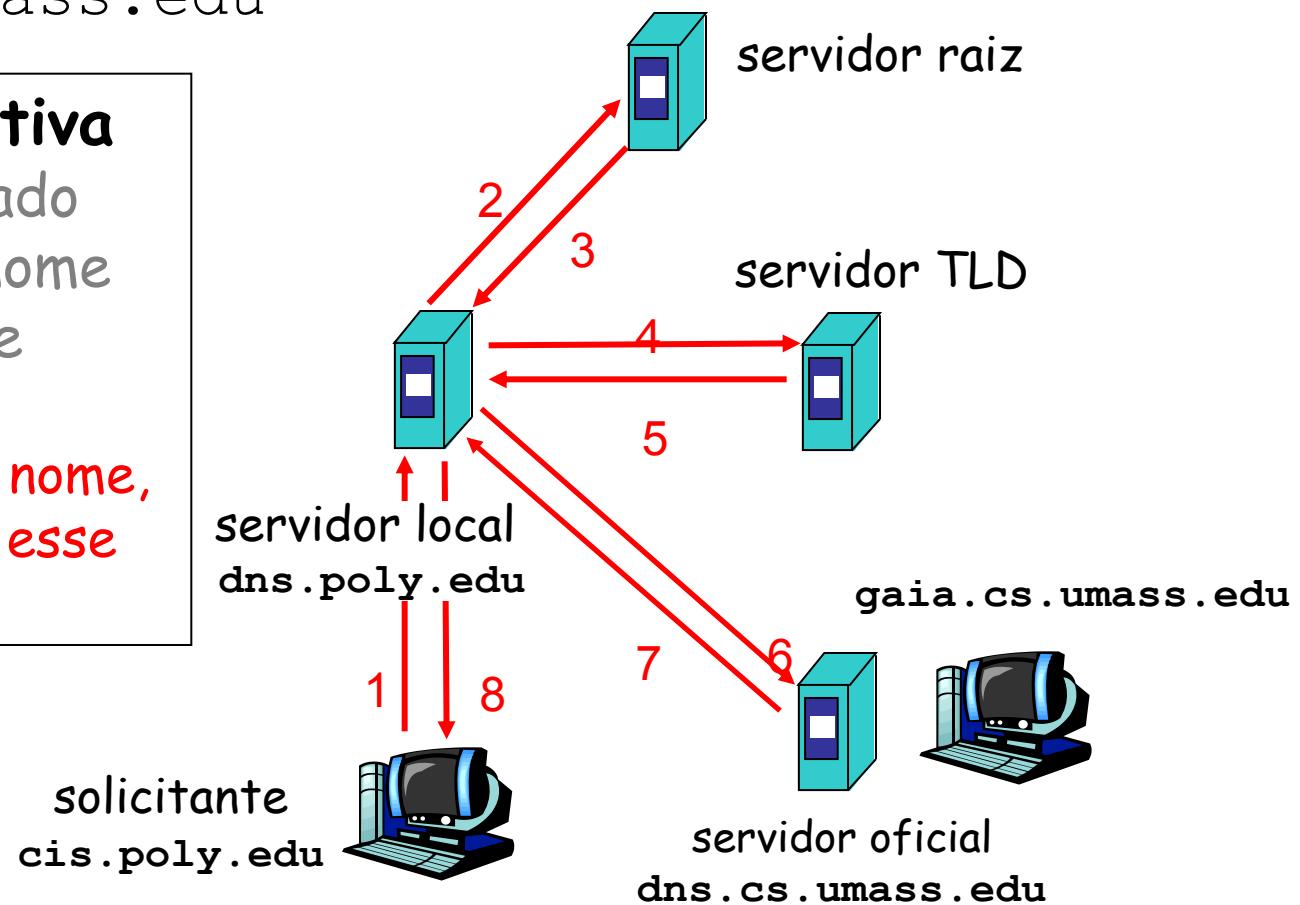
Exemplo de Resolução de Nome pelo DNS

- Estação em cis.poly.edu quer endereço IP para gaia.cs.umass.edu

Consulta interativa

Servidor consultado responde com o nome de um servidor de contato

"Não conheço este nome, mas pergunte para esse servidor"

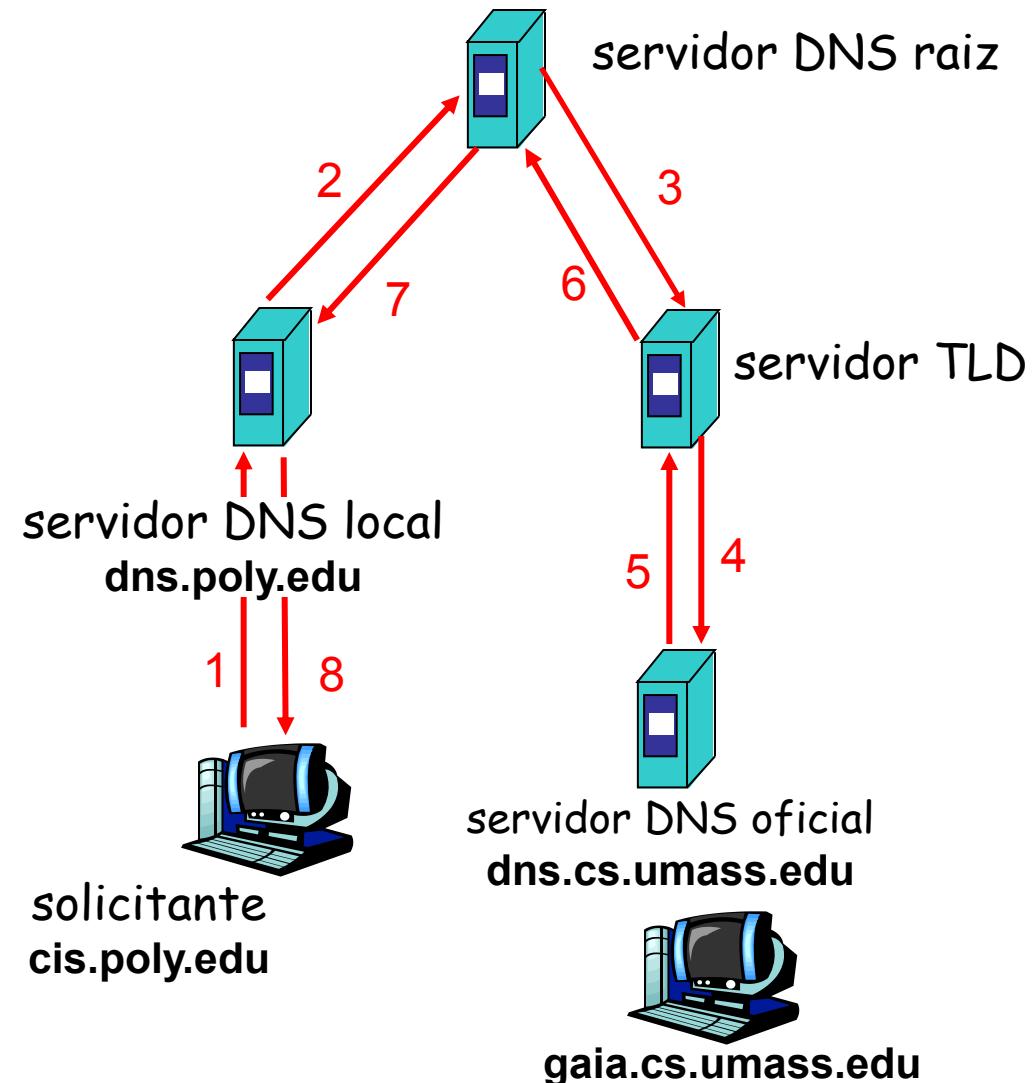


Exemplo de Resolução de Nome pelo DNS

Consulta recursiva

Transfere a responsabilidade de resolução do nome para o servidor de nomes contatado

Maior carga em servidores de maior altura



Modos de Resolução de Nomes

- Interativo X Recursivo
 - Interativo: Respostas são retornadas ao servidor de DNS local
 - Adotado na Internet
 - Recursivo: Cada servidor trata a requisição como sendo própria até receber a resposta



Em ambos os casos o procedimento completo envolve muitas requisições e respostas

Uso do Cache

- Uma vez que um servidor qualquer aprende um mapeamento, ele o coloca em um *cache* local
 - Evita a consulta a servidores de nível hierárquico mais alto
 - Entradas no *cache* são sujeitas a temporização
 - Desaparecem depois de um certo tempo
 - Geralmente, 2 dias
- Endereços dos servidores TLD
 - Armazenados no cache dos servidores de nomes locais
 - Servidores raiz acabam não sendo visitados com muita frequência

Registros

- O DNS é uma base distribuída composta por **registros de recursos (RR)**

RR: (nome, TTL, classe, tipo, valor)

- Significado de cada campo depende do tipo
 - Tipo A
 - **nome** é nome de uma estação
 - **valor** é o seu endereço IP
 - Tipo NS
 - **nome** é domínio (p.ex. foo.com.br)
 - **valor** é nome do servidor oficial de nomes para este domínio

Registros

- O DNS é uma base distribuída composta por **registros de recursos (RR)**

RR: (nome, TTL, classe, tipo, valor)

- Significado de cada campo depende do tipo
 - Tipo CNAME
 - **nome** é o "apelido" (*alias*) para algum nome "canônico" (*verdadeiro*)
 - **valor** é o nome canônico
 - Tipo MX
 - **nome** é o domínio
 - **valor** é nome do servidor de correio para este domínio

Registros

- Servidor de e-mail e de arquivos podem ter o mesmo apelido
 - Ao fazer a requisição do nome canônico (verdadeiro), o cliente escolhe o servidor pelo tipo do registro
 - **CNAME** quando quer o endereço do servidor de arquivos
 - **MX** quando quer o endereço do servidor de e-mail

Mensagens

- O DNS é protocolo baseado em mensagens de pedido e resposta
 - As duas possuem o mesmo formato

Identificação	Flags	
Número de perguntas	Número de RRs de resposta	12 bytes
Número de RRs com autoridade	Número de RRs adicionais	
Perguntas (número variável de perguntas)		Nome, campos de tipo para uma consulta
Respostas (número variável de registros de recursos)		RRs de resposta à consulta
Autoridade (número variável de registros de recursos)		Registros para servidores com autoridade
Informação adicional (número variável de registros de recursos)		Informação adicional 'útil', que pode ser usada

Mensagens

```
recreio:~> host -v www.gta.ufrj.br
Trying "www.gta.ufrj.br"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34704
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 7, ADDITIONAL: 3

;; QUESTION SECTION:
www.gta.ufrj.br.          IN      A

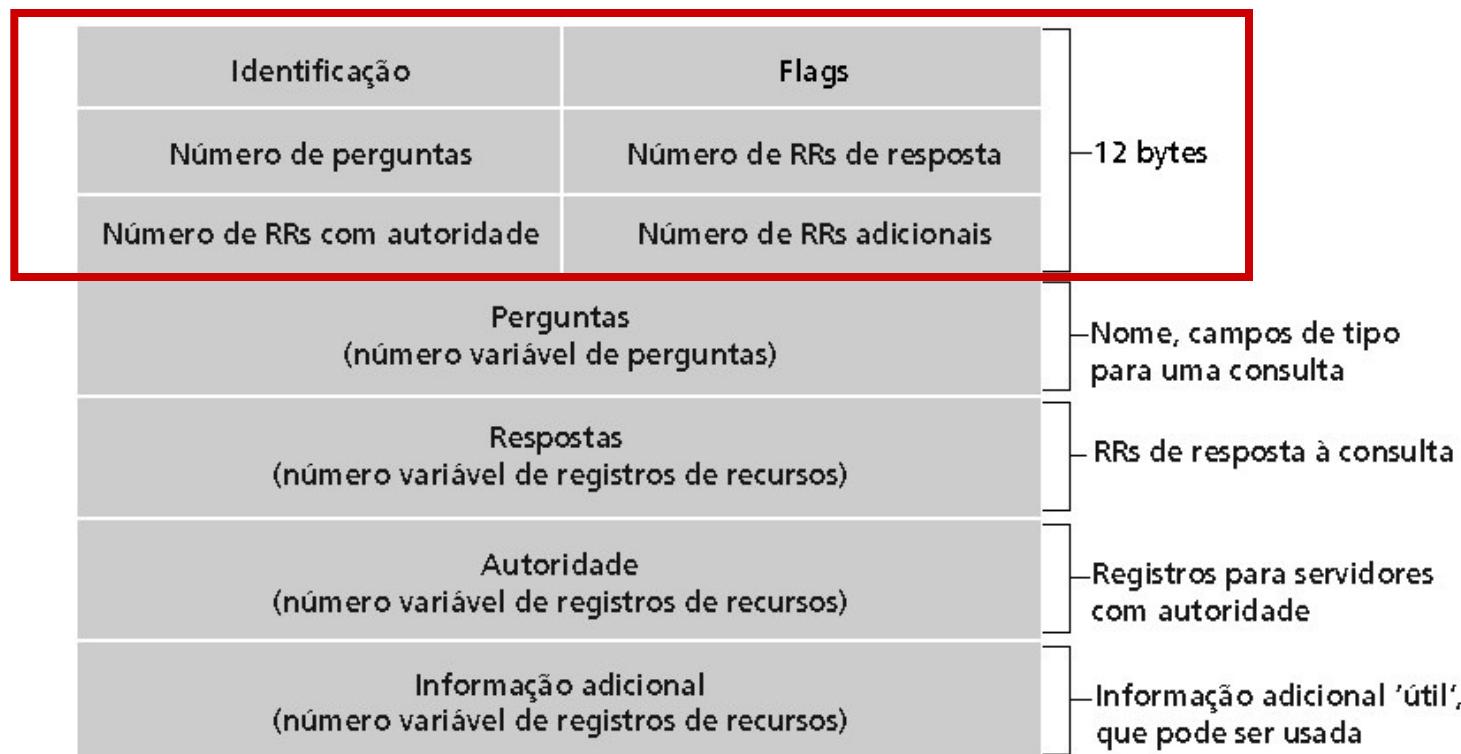
;; ANSWER SECTION:
www.gta.ufrj.br.          86400   IN      CNAME  recreio.gta.ufrj.br.
recreio.gta.ufrj.br.       86400   IN      A       146.164.69.2

;; AUTHORITY SECTION:
gta.ufrj.br.               86400   IN      NS      ns.gta.ufrj.br.
gta.ufrj.br.               86400   IN      NS      ns1.gta.ufrj.br.
gta.ufrj.br.               86400   IN      NS      ns.coppe.ufrj.br.
gta.ufrj.br.               86400   IN      NS      ns.coe.ufrj.br.
gta.ufrj.br.               86400   IN      NS      ceopl.rederio.br.
gta.ufrj.br.               86400   IN      NS      vidigal.nce.ufrj.br.
gta.ufrj.br.               86400   IN      NS      ns.ades.gta.ufrj.br.

;; ADDITIONAL SECTION:
ns.gta.ufrj.br.            86400   IN      A       146.164.69.2
ns.coppe.ufrj.br.          70305   IN      A       146.164.63.4
ns1.gta.ufrj.br.           86400   IN      A       146.164.69.28
```

Mensagens

- O DNS é um protocolo baseado em mensagens de pedido e resposta
 - As duas possuem o mesmo formato



Mensagens

```
recreio:~> host -v www.gta.ufrj.br
```

```
Trying "www.gta.ufrj.br"
```

```
; ; >>>HEADER<<< opcode: QUERY, status: NOERROR, id: 34704  
; ; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 7, ADDITIONAL: 3
```

```
; ; QUESTION SECTION:
```

```
;www.gta.ufrj.br. IN A
```

```
; ; ANSWER SECTION:
```

```
www.gta.ufrj.br. 86400 IN CNAME recreio.gta.ufrj.br.  
recreio.gta.ufrj.br. 86400 IN A 146.164.69.2
```

```
; ; AUTHORITY SECTION:
```

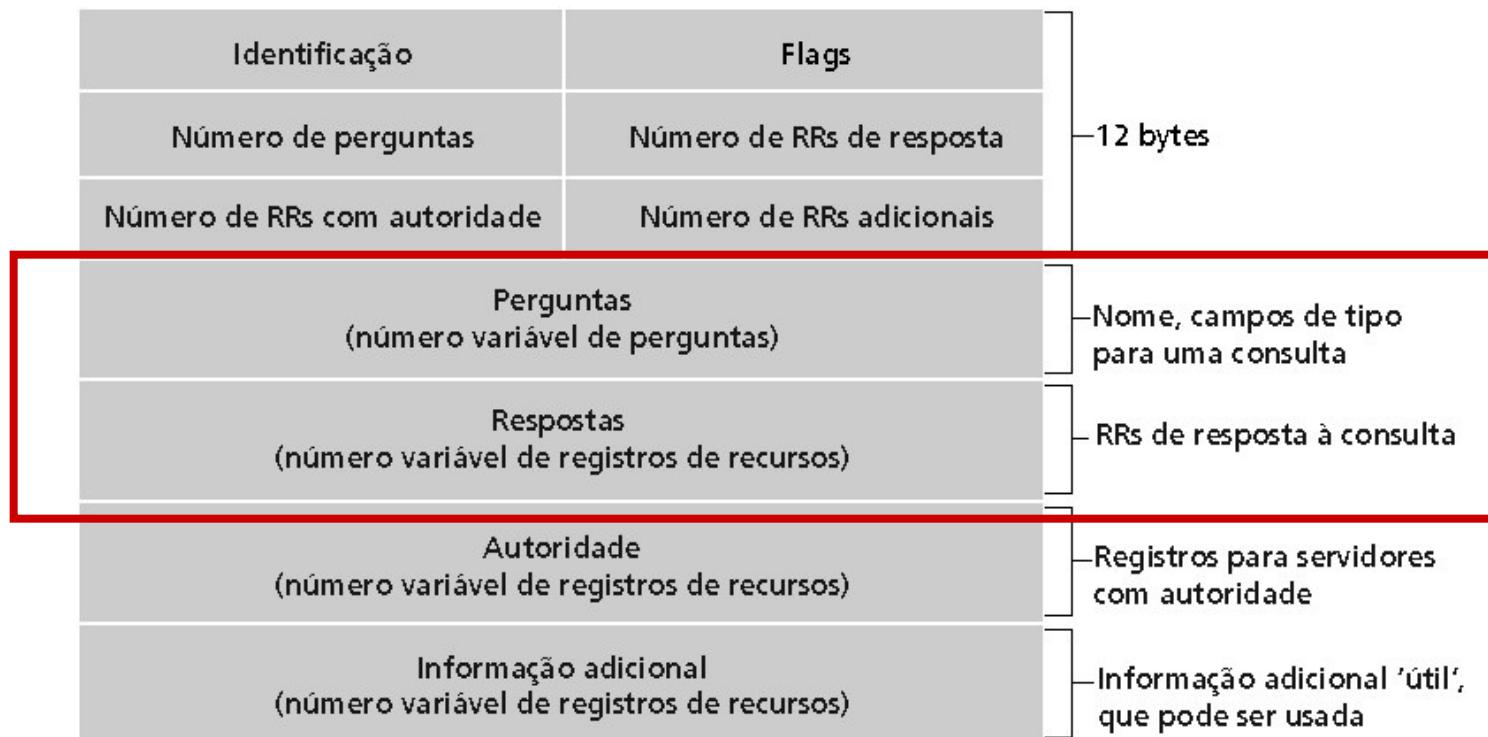
```
gta.ufrj.br. 86400 IN NS ns.gta.ufrj.br.  
gta.ufrj.br. 86400 IN NS ns1.gta.ufrj.br.  
gta.ufrj.br. 86400 IN NS ns.coppe.ufrj.br.  
gta.ufrj.br. 86400 IN NS ns.coe.ufrj.br.  
gta.ufrj.br. 86400 IN NS ceopl.rederio.br.  
gta.ufrj.br. 86400 IN NS vidigal.nce.ufrj.br.  
gta.ufrj.br. 86400 IN NS ns.ades.gta.ufrj.br.
```

```
; ; ADDITIONAL SECTION:
```

```
ns.gta.ufrj.br. 86400 IN A 146.164.69.2  
ns.coppe.ufrj.br. 70305 IN A 146.164.63.4  
ns1.gta.ufrj.br. 86400 IN A 146.164.69.28
```

Mensagens

- O DNS é um protocolo baseado em mensagens de pedido e resposta
 - As duas possuem o mesmo formato



Mensagens

```
recreio:~> host -v www.gta.ufrj.br
Trying "www.gta.ufrj.br"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34704
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 7, ADDITIONAL: 3
```

```
;; QUESTION SECTION:
```

```
;www.gta.ufrj.br.          IN      A
```

```
;; ANSWER SECTION:
```

www.gta.ufrj.br.	86400	IN	CNAME	recreio.gta.ufrj.br.
recreio.gta.ufrj.br.	86400	IN	A	146.164.69.2

```
;; AUTHORITY SECTION:
```

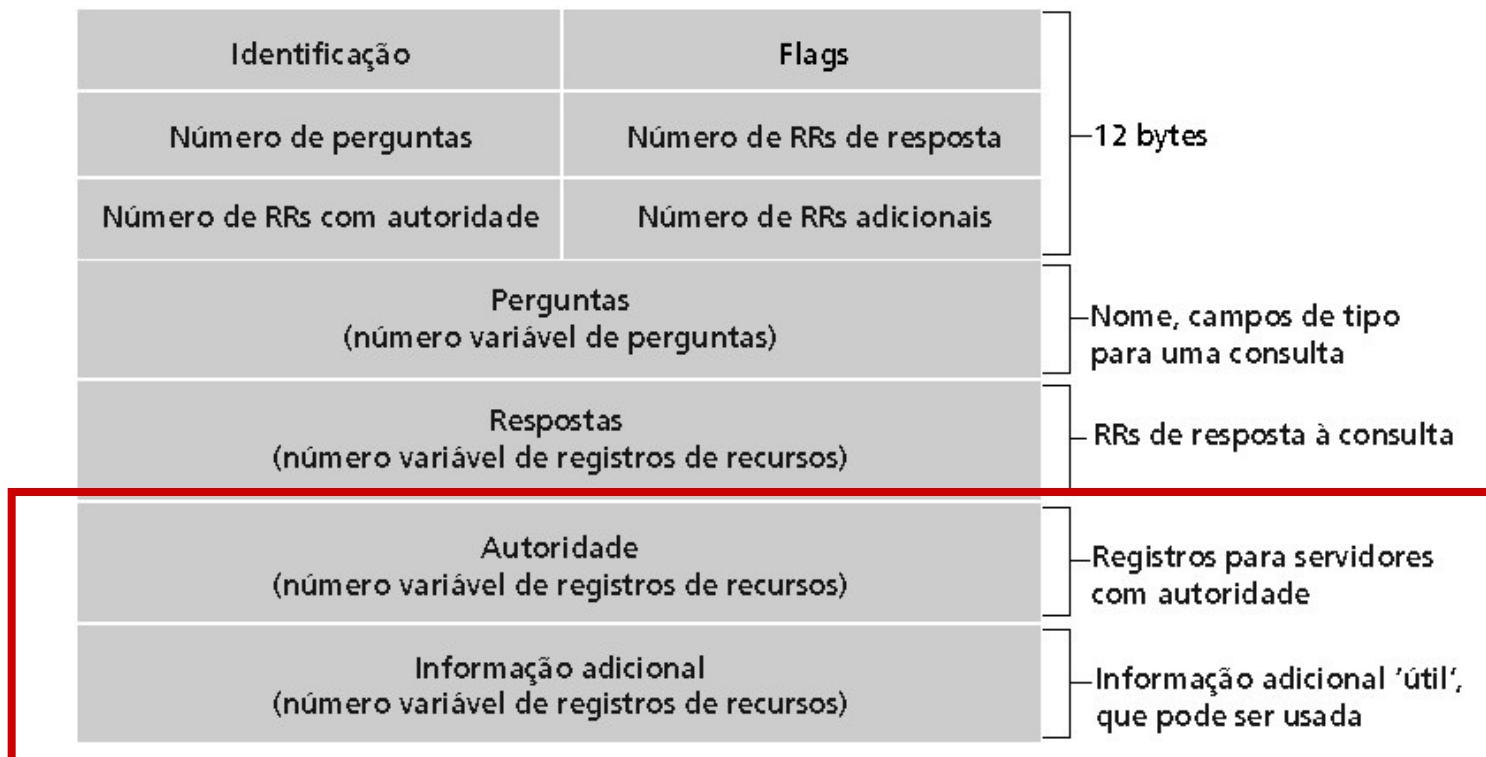
gta.ufrj.br.	86400	IN	NS	ns.gta.ufrj.br.
gta.ufrj.br.	86400	IN	NS	ns1.gta.ufrj.br.
gta.ufrj.br.	86400	IN	NS	ns.coppe.ufrj.br.
gta.ufrj.br.	86400	IN	NS	ns.coe.ufrj.br.
gta.ufrj.br.	86400	IN	NS	ceopl.rederio.br.
gta.ufrj.br.	86400	IN	NS	vidigal.nce.ufrj.br.
gta.ufrj.br.	86400	IN	NS	ns.ades.gta.ufrj.br.

```
;; ADDITIONAL SECTION:
```

ns.gta.ufrj.br.	86400	IN	A	146.164.69.2
ns.coppe.ufrj.br.	70305	IN	A	146.164.63.4
ns1.gta.ufrj.br.	86400	IN	A	146.164.69.28

Mensagens

- O DNS é um protocolo baseado em mensagens de pedido e resposta
 - As duas possuem o mesmo formato



Mensagens

```
recreio:~> host -v www.gta.ufrj.br
Trying "www.gta.ufrj.br"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34704
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 7, ADDITIONAL: 3

;; QUESTION SECTION:
;www.gta.ufrj.br.          IN      A

;; ANSWER SECTION:
www.gta.ufrj.br.      86400   IN      CNAME   recreio.gta.ufrj.br.
recreio.gta.ufrj.br.    86400   IN      A       146.164.69.2

;; AUTHORITY SECTION:
gta.ufrj.br.          86400   IN      NS      ns.gta.ufrj.br.
gta.ufrj.br.          86400   IN      NS      ns1.gta.ufrj.br.
gta.ufrj.br.          86400   IN      NS      ns.coppe.ufrj.br.
gta.ufrj.br.          86400   IN      NS      ns.coe.ufrj.br.
gta.ufrj.br.          86400   IN      NS      ceopl.rederio.br.
gta.ufrj.br.          86400   IN      NS      vidigal.nce.ufrj.br.
gta.ufrj.br.          86400   IN      NS      ns.ades.gta.ufrj.br.

;; ADDITIONAL SECTION:
ns.gta.ufrj.br.        86400   IN      A       146.164.69.2
ns.coppe.ufrj.br.      70305   IN      A       146.164.63.4
ns1.gta.ufrj.br.       86400   IN      A       146.164.69.28
```

Inserção de Registros no DNS

- Exemplo: Criação da empresa “Network Utopia”
 - Primeiro: Registra-se o nome netutopia.com.br em uma **entidade registradora** (e.x., Registro.br)
 - Tem que prover para a registradora os nomes e endereços IP dos servidores DNS oficiais (primário e secundário)
 - Registradora insere dois RRs no servidor TLD .br:
(netutopia.com.br, dns1.netutopia.com.br, NS)
(dns1.netutopia.com.br, 212.212.212.1, A)
 - Por fim: Configura no servidor oficial um registro do tipo A para www.netutopia.com.br e um registro do tipo MX para netutopia.com.br

Sistemas Par-a-Par

Modelo de Aplicações

Rede orientada
ao **usuário**



Rede orientada
ao **conteúdo**

um usuário quer contatar
outro usuário

acesso a terminal remoto (telnet),
transferência de arquivos (FTP) e
correo eletrônico (SMTP)

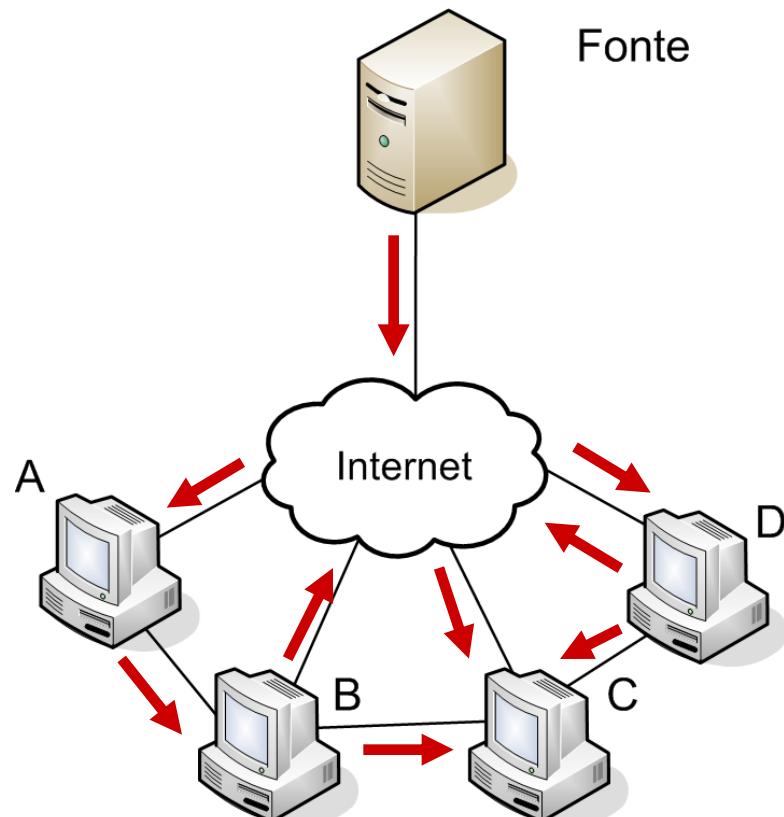
um usuário quer acessar
um serviço ou dado
específico

Não importa onde (em que
estação) esse serviço ou dado
está localizado

Sistemas par-a-par
(BitTorrent),
redes de distribuição de
conteúdo (Akamai)

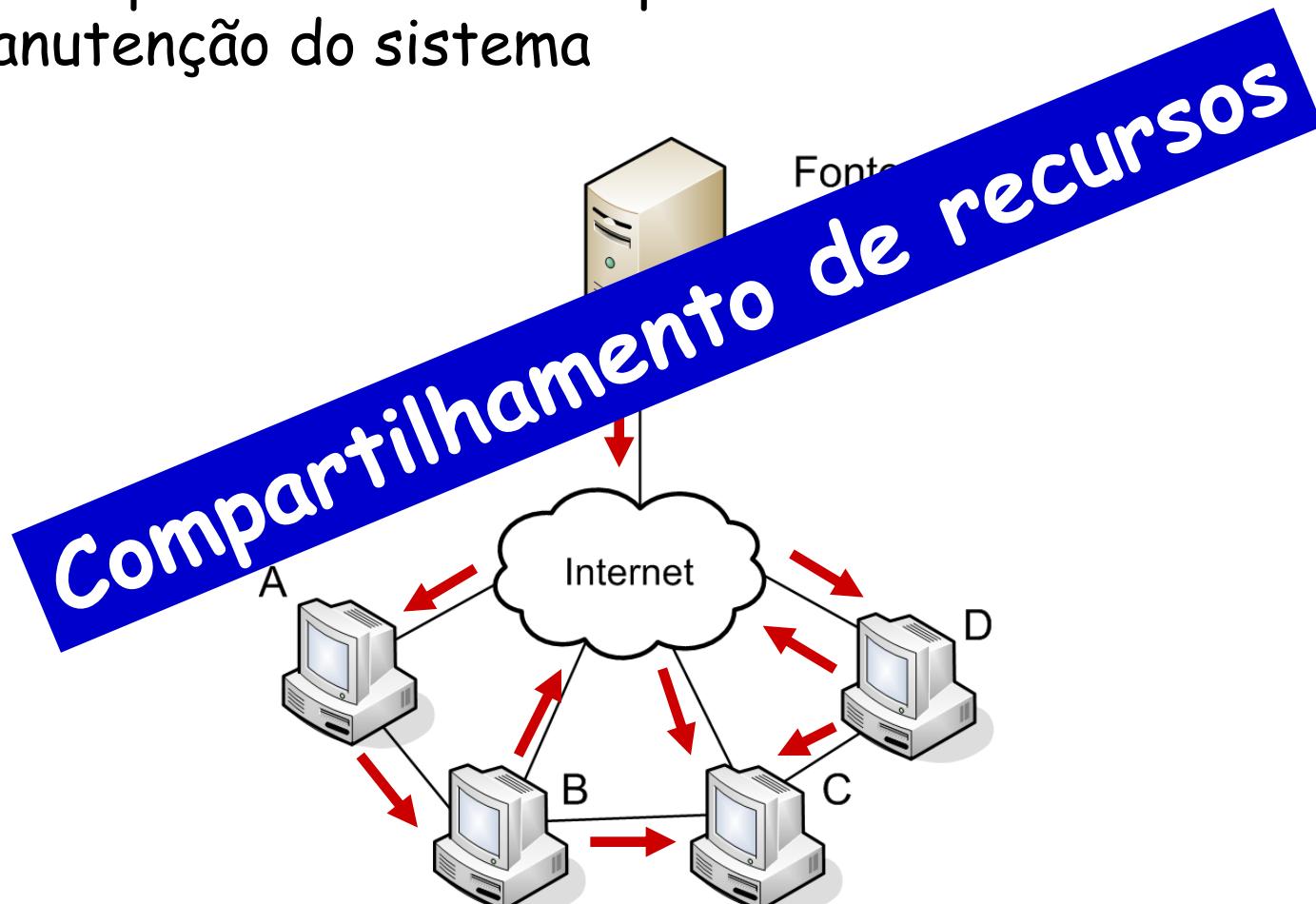
Sistemas Par-a-Par

- Participantes colaboram para o funcionamento e manutenção do sistema



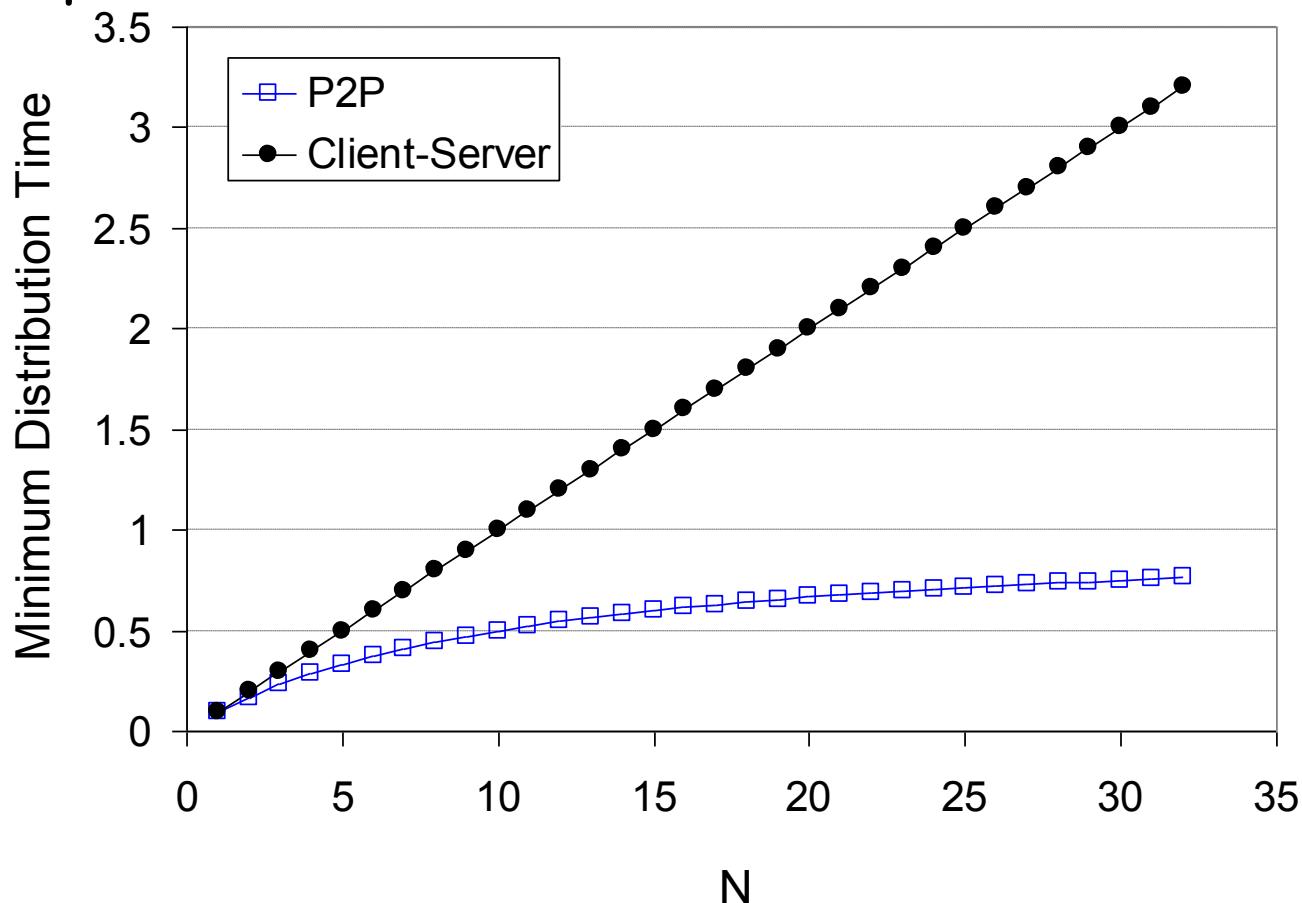
Sistemas Par-a-Par

- Participantes colaboram para o funcionamento e manutenção do sistema



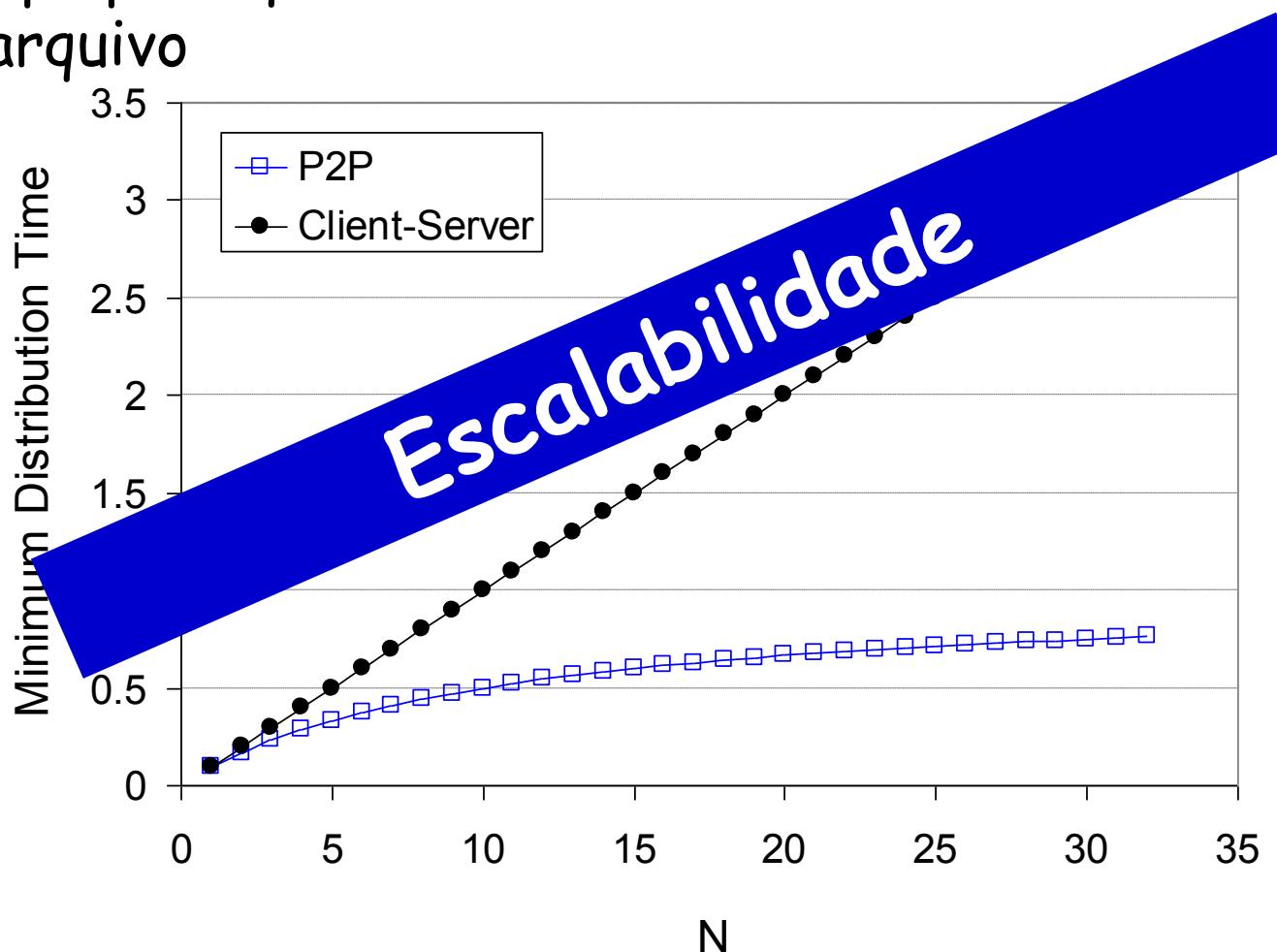
Cliente Servidor X P2P

- Tempo para que todos os usuários recebam uma cópia do arquivo



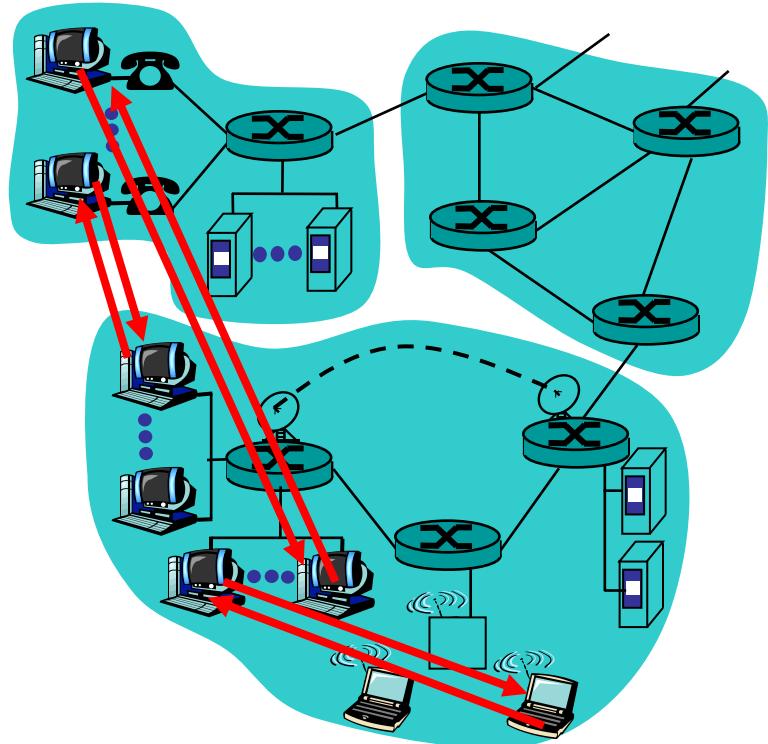
Cliente Servidor X P2P

- Tempo para que todos os usuários recebam uma cópia do arquivo



Arquiteturas dos Sistemas Par-a-Par

- “Pura”
 - Comunicação direta entre sistemas finais
- Híbrida
 - Uso de servidores auxiliares
 - Ex.: Skype, BitTorrent, etc.



Compartilhamento de Arquivos

- Ideia
 - Alice executa aplicação cliente P2P no seu notebook
 - Busca a música: "Hey Jude"
 - Aplicação apresenta uma lista de outros parceiros que possuem uma cópia de "Hey Jude"
 - Alice escolhe um dos parceiros: Bob
 - O arquivo é copiado do PC do Bob para o notebook da Alice
 - Enquanto Alice está baixando a música, outros usuários podem pegar arquivos do seu computador
 - Bob é tanto um cliente quanto um servidor Web temporário

Compartilhamento de Arquivos

- Ideia
 - Alice executa aplicação cliente P2P no seu notebook
 - Busca a música: "Hey Jude"
 - Aplicação apresenta uma lista de usuários que possuem uma cópia da música
 - Alice escolhe o usuário Bob
 - O arquivo é baixado do computador Bob para o notebook da Alice
- Um dos problemas críticos: busca por arquivos**
- Alice está baixando a música, outros usuários podem estar trocar arquivos do seu computador
- Bob é tanto um cliente quanto um servidor Web temporário

Busca

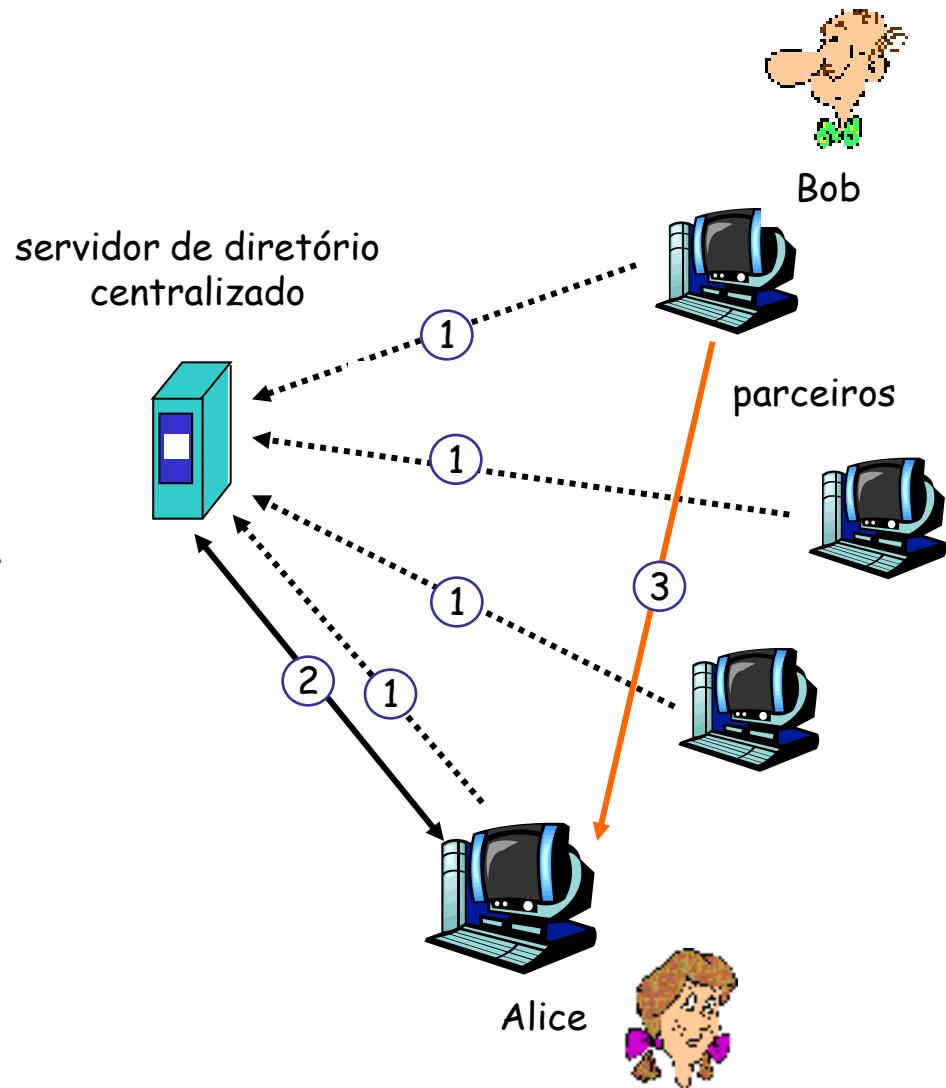
- Índice em um sistema par-a-par
 - Mapeia informação à localização de um par
 - Registra dinamicamente as localizações dos arquivos compartilhados pelos pares
- Pares devem informar o índice dos conteúdos que possuem
- Pares buscam no índice para descobrir onde podem encontrar os arquivos

Como construir o índice?

Diretório Centralizado

- Napster

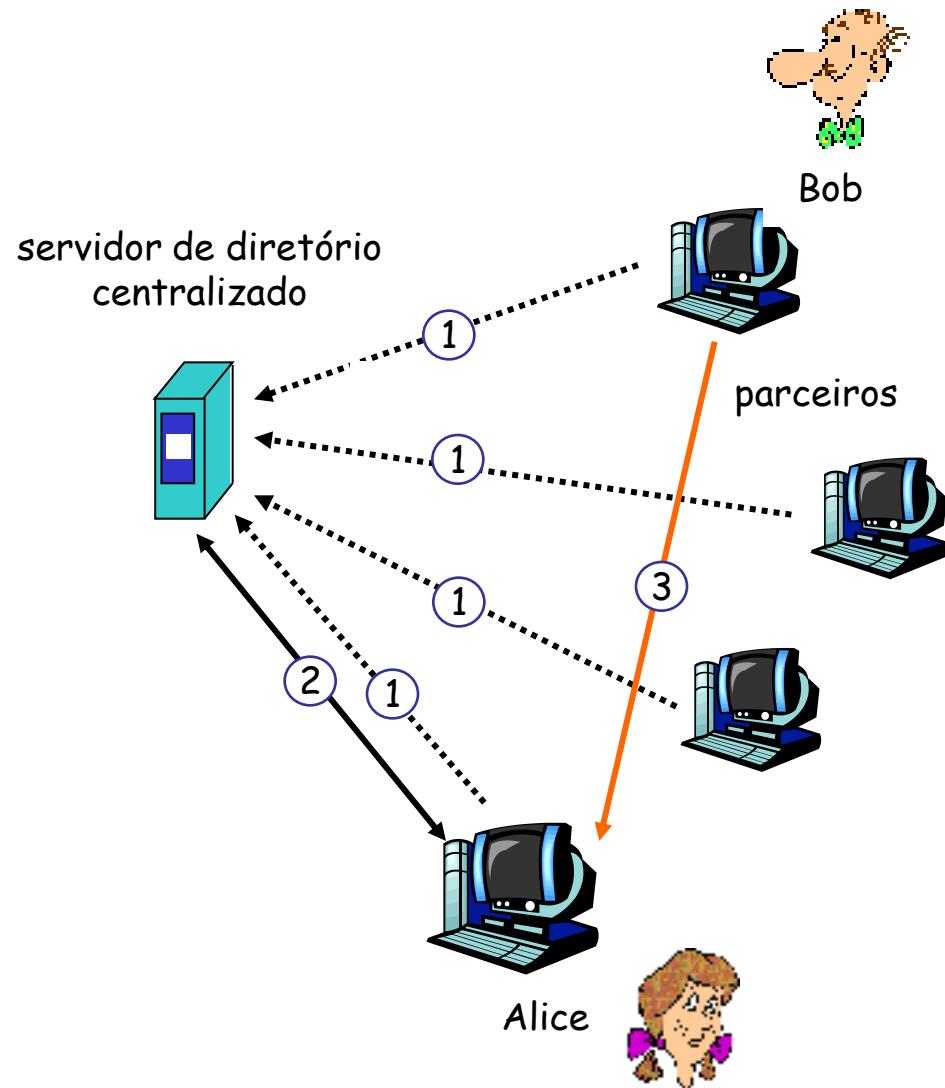
- Passo 1: Quando um parceiro conecta, ele informa ao servidor central o seu:
 - Endereço IP
 - Conteúdo
- Passo 2: Alice consulta o servidor central sobre a música "Hey Jude"
- Passo 3: Alice solicita o arquivo a Bob



Diretório Centralizado

- **Problemas**

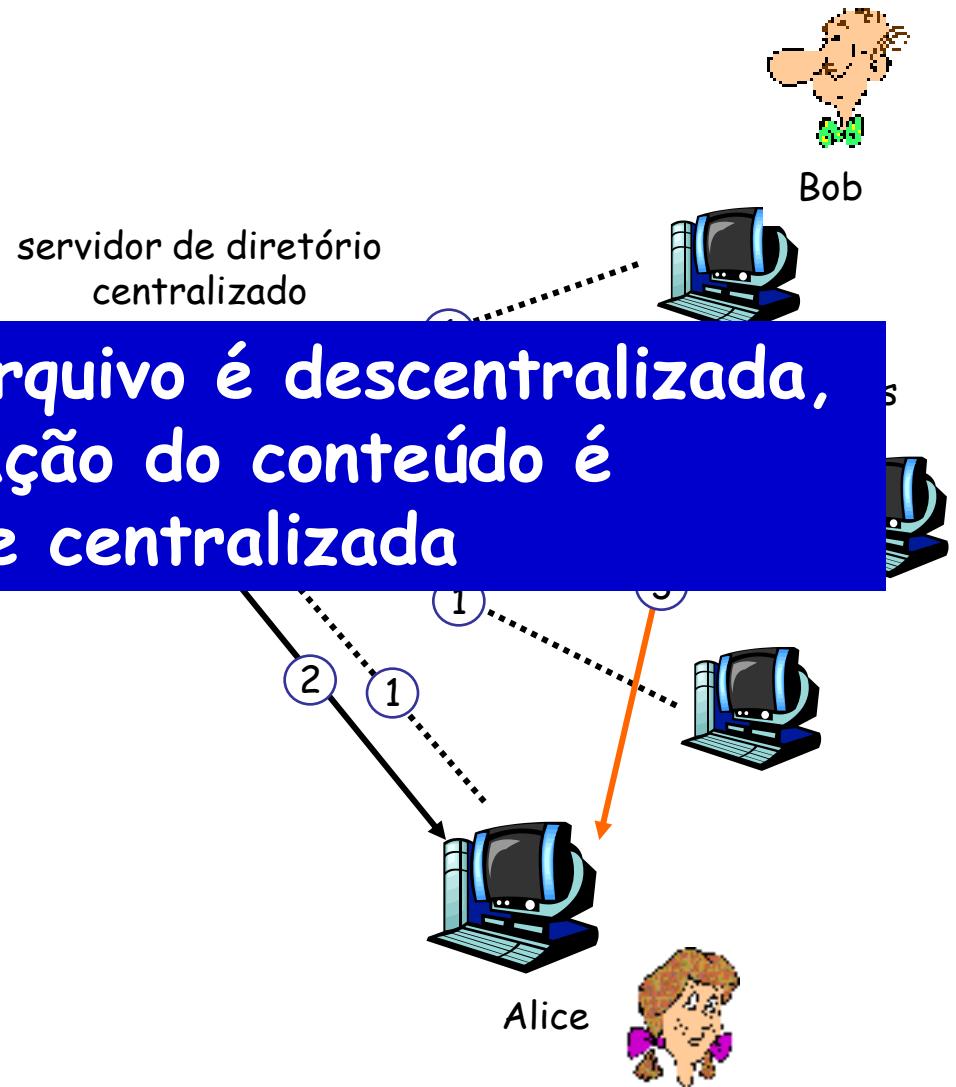
- Ponto único de falha
- Gargalo de desempenho no servidor de diretório
- Violação de Direitos Autorais



Diretório Centralizado

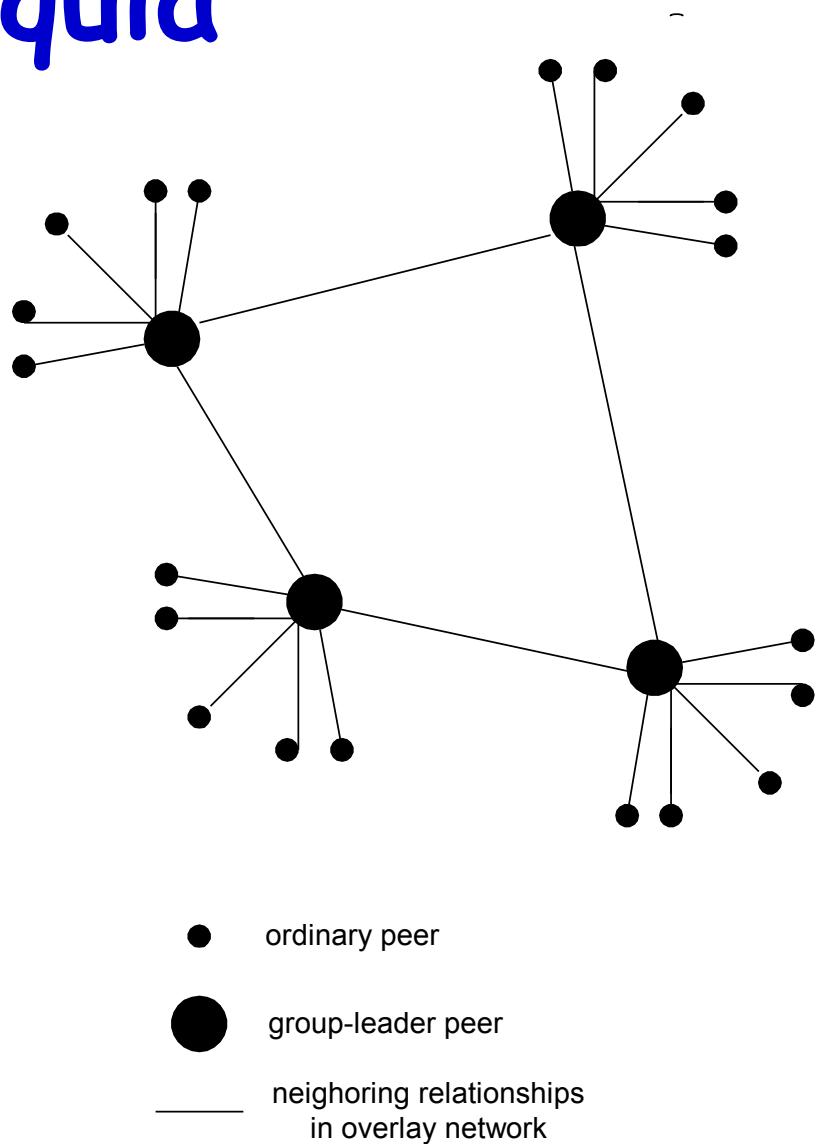
- **Problemas**

- Ponto único de falha
- Gargalo de desempenho no



Hierarquia

- Cada parceiro é um líder de grupo ou está alocado a um líder de grupo
 - Conexão TCP entre cada par e o seu líder de grupo
 - Conexões TCP entre alguns pares de líderes de grupos
- O líder de um grupo mantém registro sobre o conteúdo de todos os seus filhos



Tabelas Hash Distribuídas (*Distributed Hash Tables* - DHTs)

- Informações representadas por um par (**chave, valor**)
 - (1980, José)
 - (Led Zeppelin IV, 192.168.2.1)
- Aplicar uma função *hash* em elementos do par
 - Pequena probabilidade de colisão → identificação única
 - Espalhamento
 - **Distribuição de carga**
 - Algumas garantem que não é possível a partir de um valor de *hash* retornar à informação original
 - **Irreversibilidade**

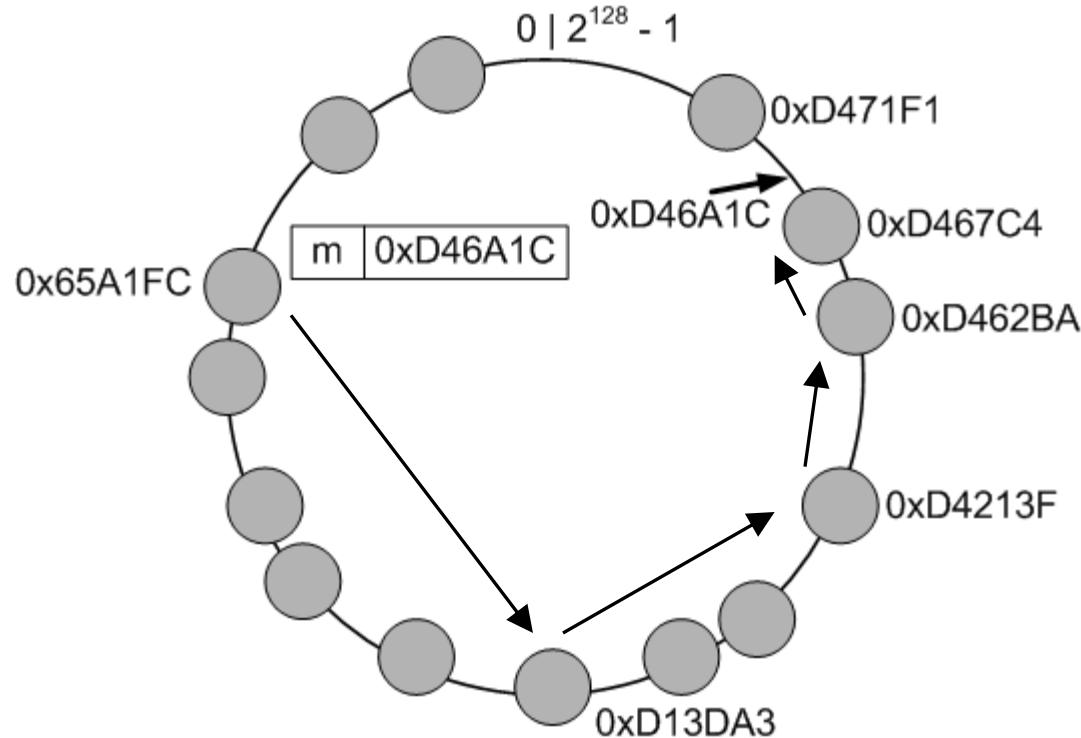
Exemplo: Pastry

- Recomendado para **construção da rede sobreposta**
- Estima a proximidade entre os nós para construir os enlaces da rede sobreposta
 - Nós são capazes de medir sua distância para outro nó de endereço IP conhecido
 - Construção de caminhos próximos aos da camada de rede
 - **Característica desejada**
- Distância
 - Número de saltos (traceroute)
 - Tempo de ida-e-volta (ping)
 - Vazão (par de pacotes)

Funcionamento do Pastry

- Para cada nó → um identificador de 128 bits
 - Função *hash* do endereço IP ou da chave pública do nó
 - Conjunto de identificadores uniformemente distribuído
- Para cada objeto → uma chave de 128 bits
- Dada uma mensagem e uma chave
 - A mensagem é encaminhada para o nó com identificador numericamente mais próximo da chave

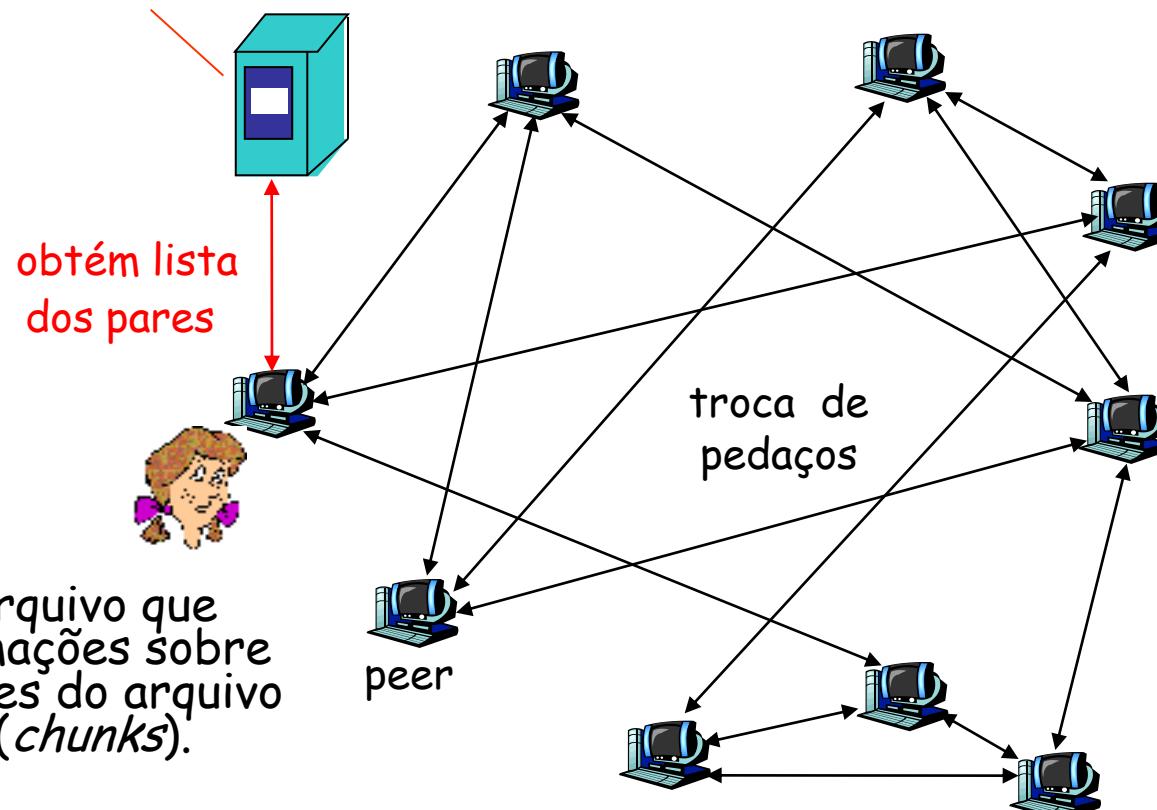
Funcionamento do Pastry



BitTorrent

Tracker: registra pares
Participantes de um torrent

Enxame (swarm): grupo de
pares trocando
pedaços de um arquivo



Torrent: Arquivo que
contém informações sobre
tracker e partes do arquivo
desejado (*chunks*).

BitTorrent

- BitTorrent Trackers
 - Auxiliam a comunicação entre os pares que estão trocando pedaços do mesmo arquivo
 - Pares que participam do mesmo swarm (enxame)
 - Estruturas do tipo DHT são usadas por clientes que não utilizam os trackers
- BitTorrent Indexer
 - Usado para listar detalhes dos arquivos compartilhados
 - Detalhes obtidos de um ou mais trackers
 - Arquivos acessíveis através do protocolo BitTorrent
 - Normalmente são páginas web
 - Muitos indexers são também trackers

BitTorrent

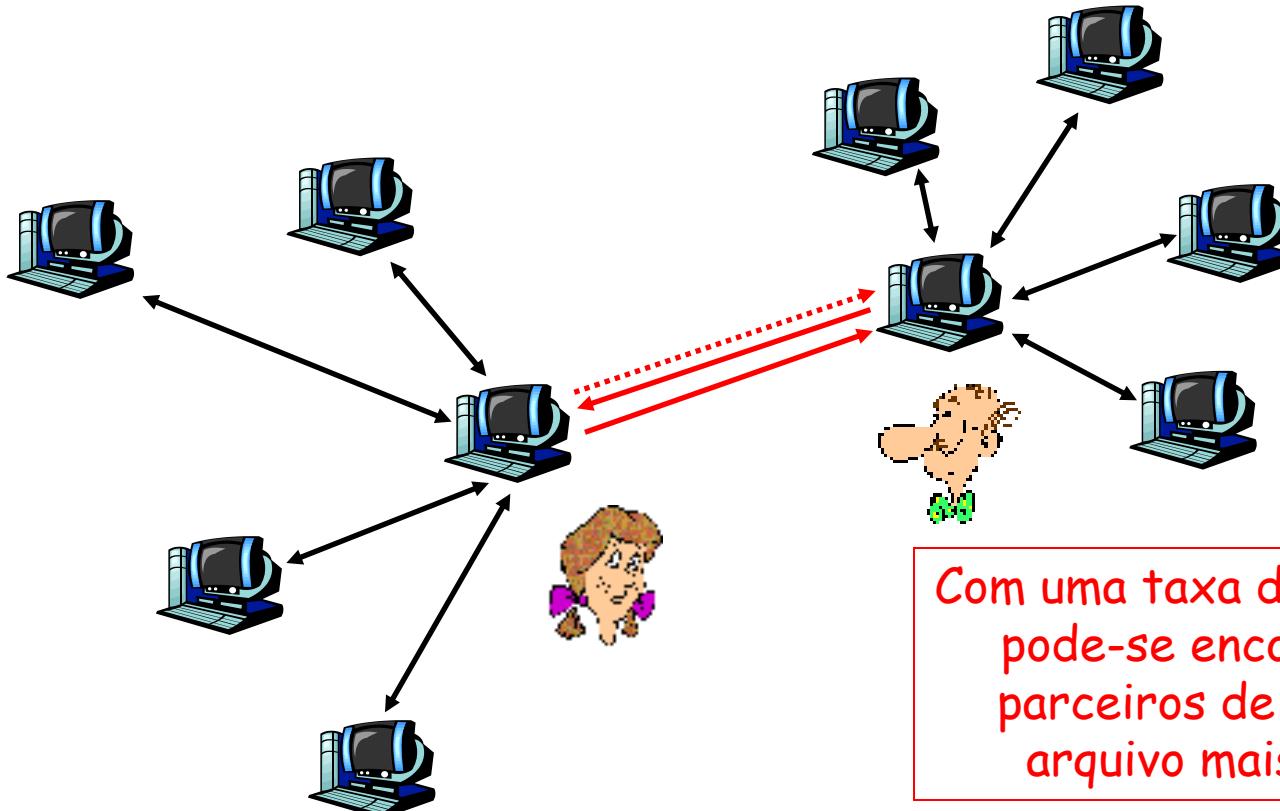
- Arquivo dividido em **pedaços** (*chunks*) de 64 a 512 kB
- Ao se unir ao enxame, o par:
 - Não tem nenhum pedaço, mas irá acumulá-los com o tempo
 - Registra com o *tracker* para obter lista dos pares, conecta a um subconjunto de pares ("vizinhos")
- Enquanto faz o *download*, par carrega pedaços para outros pares
- Pares podem entrar e sair
- Ao obter o arquivo, o par pode (egoisticamente) sair ou (altruisticamente) permanecer

BitTorrent

- Num determinado instante, pares distintos possuem diferentes subconjuntos dos pedaços do arquivo
 - Pares que possuem todos os pedaços são chamados de *seeders*
- Periodicamente, um par (Alice) recebe de seus vizinhos a lista de pedaços que eles possuem
- Alice envia pedidos para os pedaços que ainda não tem
 - Primeiro os mais raros

BitTorrent

- (1) Alice “optimistically unchoke” Bob
- (2) Alice se torna um dos quatro melhores provedores de Bob;
Bob age da mesma forma
- (3) Bob se torna um dos quatro melhores provedores de Alice

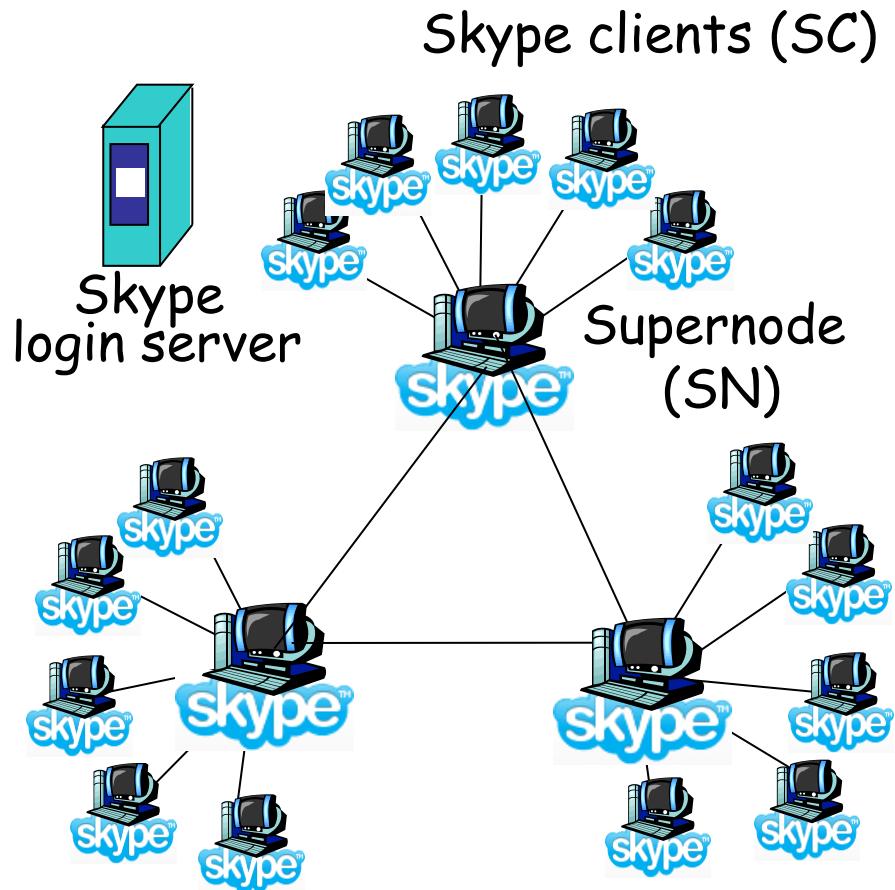


BitTorrent

- Olho-por-olho (*tit-for-tat*)
 - Alice envia pedaços para quatro vizinhos que estejam lhe enviando pedaços na taxa mais elevada
 - Reavalia os 4 a cada 10 s
 - Seleciona aleatoriamente outro par, começa a enviar pedaços
 - A cada 30 s
 - “*optimistically unchoke*”
 - Não sabe se o novo par será melhor

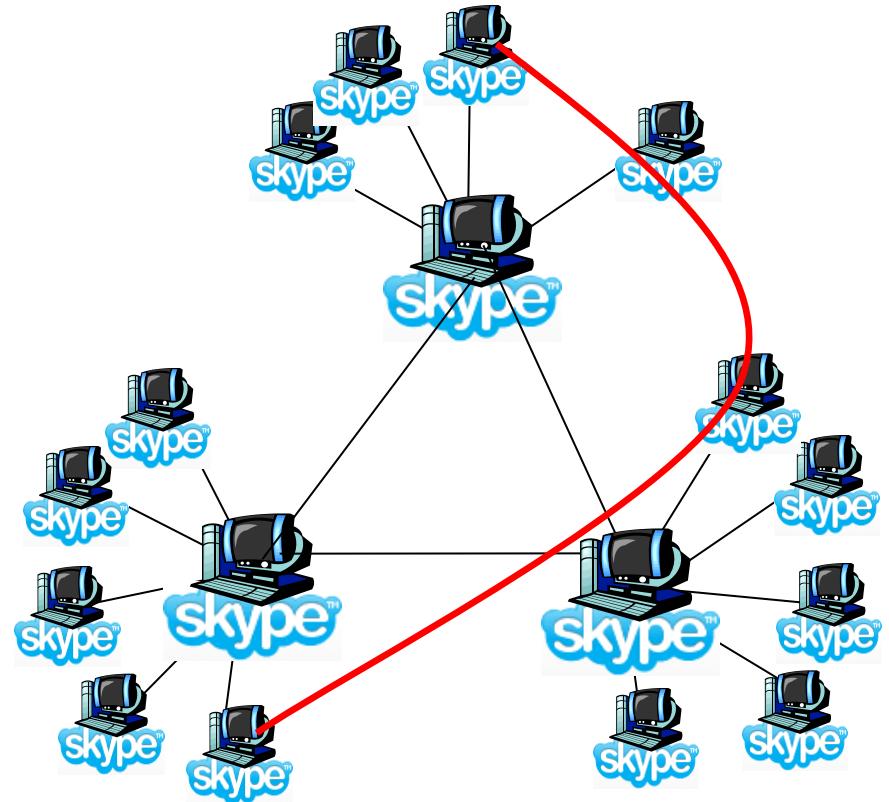
Skype

- Protocolo proprietário da camada de aplicação
 - Funcionamento estimado através de engenharia reversa
- Comunicação entre pares de usuários é P2P
- Overlay hierárquico com super-nós (SNs)
- Índice mapeia nomes dos usuários em endereços IP
 - Distribuído através dos SNs



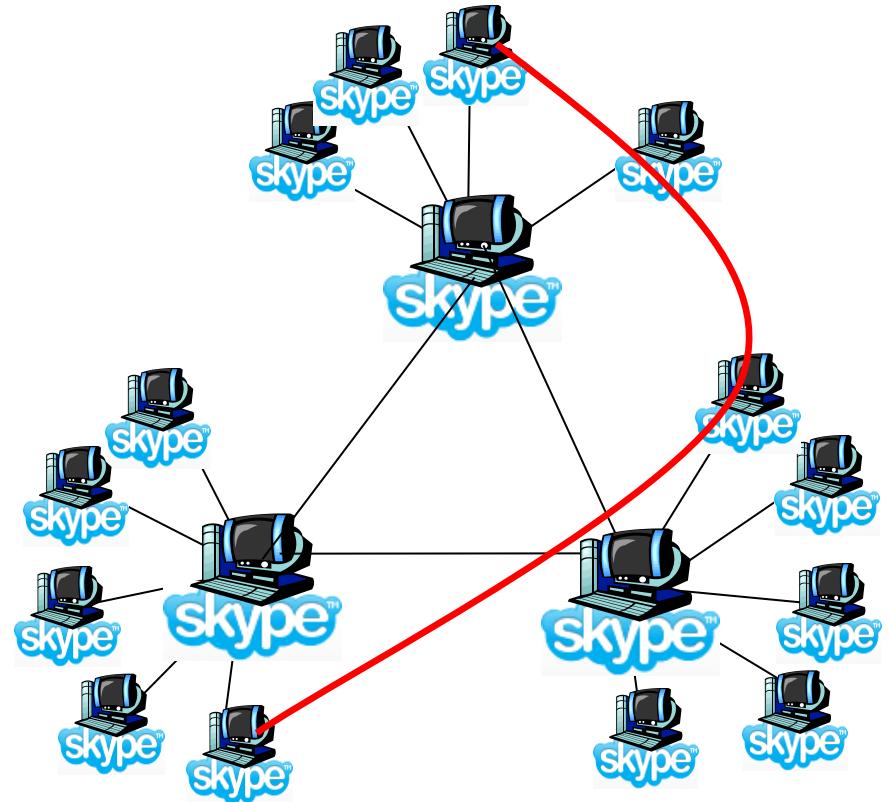
Skype

- Problema quando tanto Alice quanto Bob estão atrás de "NATs"
 - O NAT impede que um par externo inicie uma chamada com um par interno



Skype

- Solução
 - Usuário mantém conexão com super-nó para controle
 - Um terceiro nó (intermediário) é escolhido pelos SNs de Alice e Bob
 - Intermediário não pode estar atrás de um NAT
 - Cada par inicia sessão com o intermediário
 - Pares podem se comunicar mesmo atrás de NATs usando o nó intermediário para triangulação



Sistemas de Vídeo Par-a-Par

- Sucesso do compartilhamento de arquivos
 - Indicativo do potencial para distribuição de vídeo

Compartilhamento de arquivos	Distribuição de vídeo
Longas transferências sem restrições de tempo	Requisitos estritos de banda passante e tempo
Indexação e busca eficientes	Comunicação eficiente
Arquivos disponíveis a partir da publicação	Exibição durante um período de tempo

Sistemas de Vídeo Par-a-Par

- Usuários simultâneos
 - Característica da distribuição de vídeo
 - Audiência de um programa
 - Mais usuários
 - Mais recursos compartilhados
 - É possível atender os requisitos das aplicações de vídeo

Arquiteturas de Distribuição

- Duas arquiteturas:
 - Em árvore
 - Em malha

Arquiteturas de Distribuição

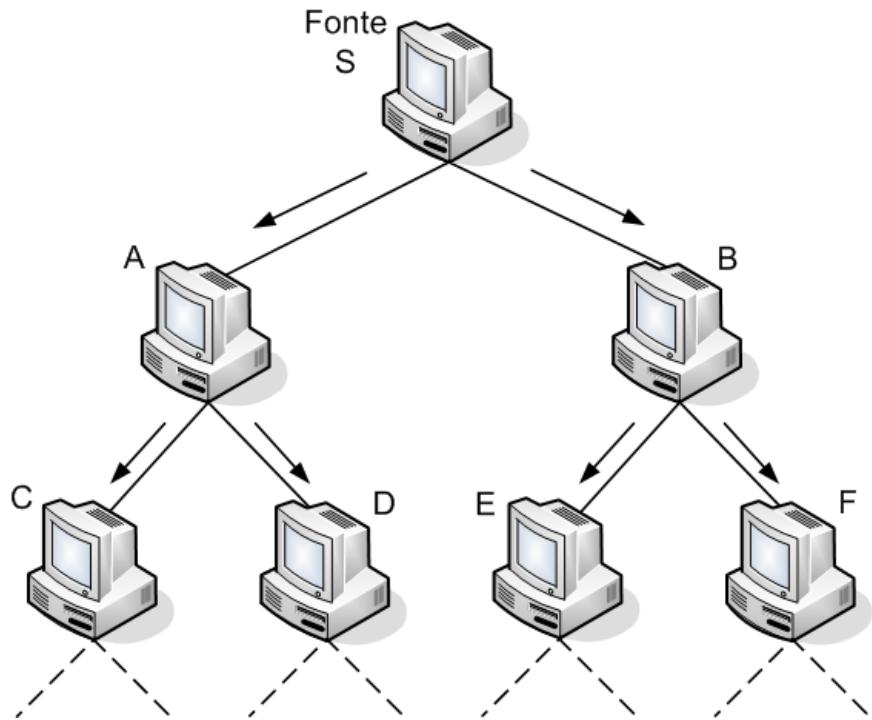
- Árvore
 - Uma ou múltiplas árvores
 - A fonte é a raiz
 - Relações de pai e filho
 - Um pai encaminha os dados somente para os filhos
 - Um participante deve se inscrever na árvore
 - Vídeo recebido sem novas requisições

Arquiteturas de Distribuição

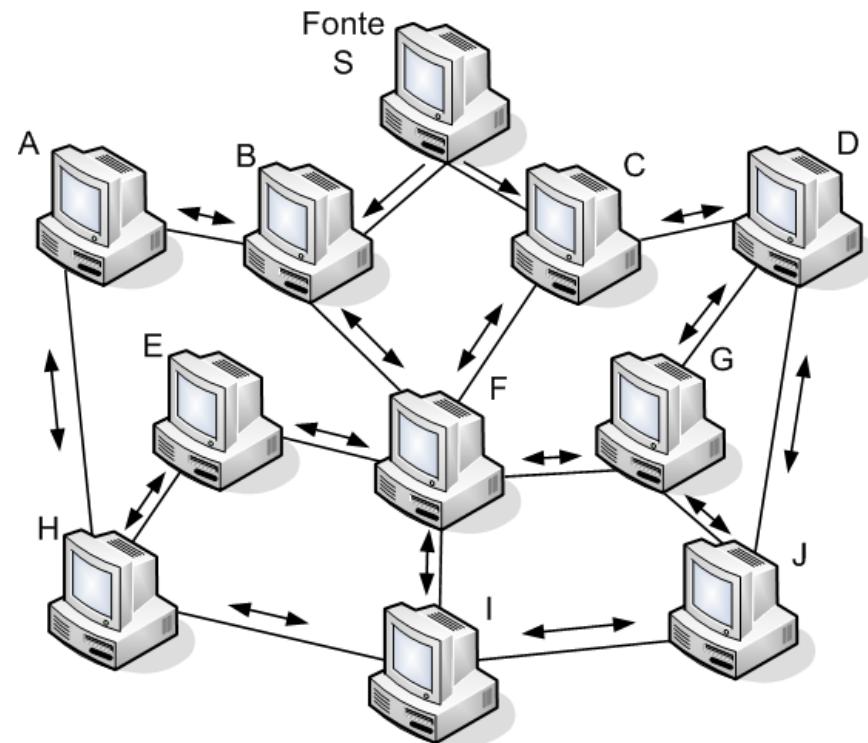
- Malha
 - Malha de distribuição
 - Participantes não possuem funções específicas
 - Receber e encaminhar para quaisquer nós
 - Sem uma organização hierárquica
 - Vídeo é dividido em pedaços (chunks)
 - Espalhados pelos participantes
 - Localização dos pedaços
 - Uma requisição por pedaço

Arquiteturas de Distribuição

Árvore



Malha



Arquiteturas de Distribuição

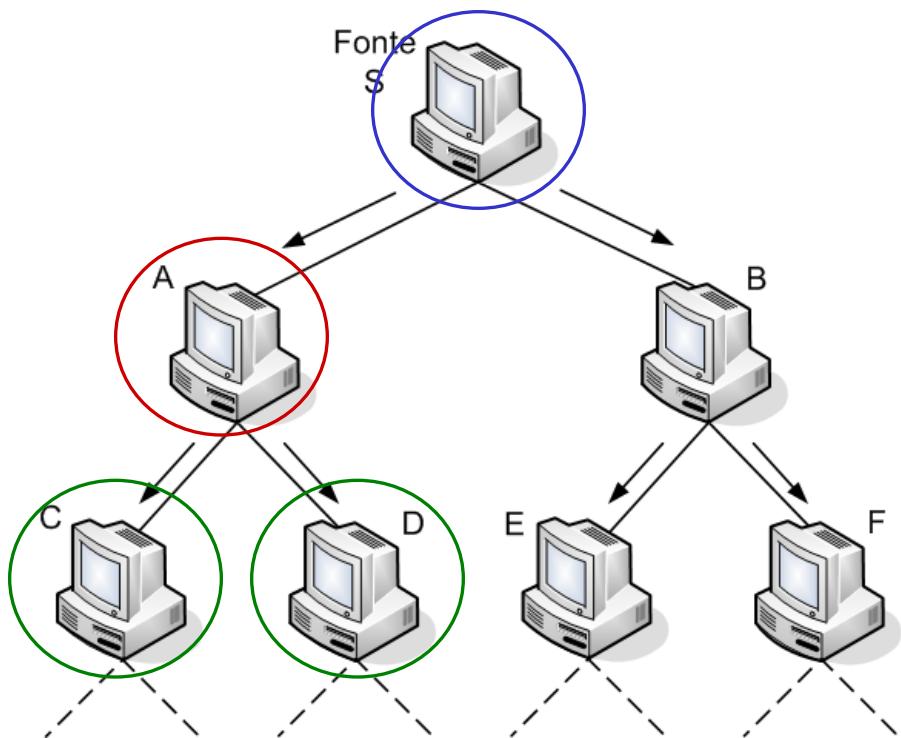
Árvore	Malha
Comunicação multidestinatária	Divisão do vídeo em pedaços
Hierarquia no encaminhamento	Sem hierarquia no encaminhamento
Uma requisição à fonte	Uma requisição a cada pedaço
Aumentar a eficiência do encaminhamento	Aumentar a robustez à dinâmica dos participantes

Arquiteturas de Distribuição

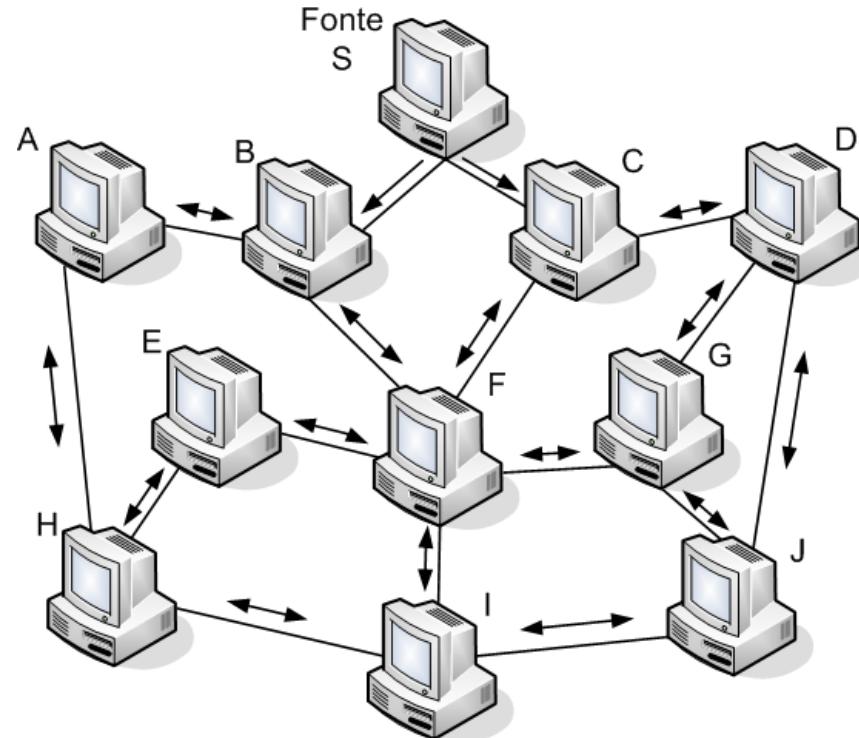
Árvore	Malha
Comunicação multidestinatária	Divisão do vídeo em pedaços
Hierarquia no encaminhamento	Sem hierarquia no encaminhamento
Uma requisição à fonte	Uma requisição a cada pedaço
Aumentar a eficiência do encaminhamento	Aumentar a robustez à dinâmica dos participantes

Arquiteturas de Distribuição

Árvore

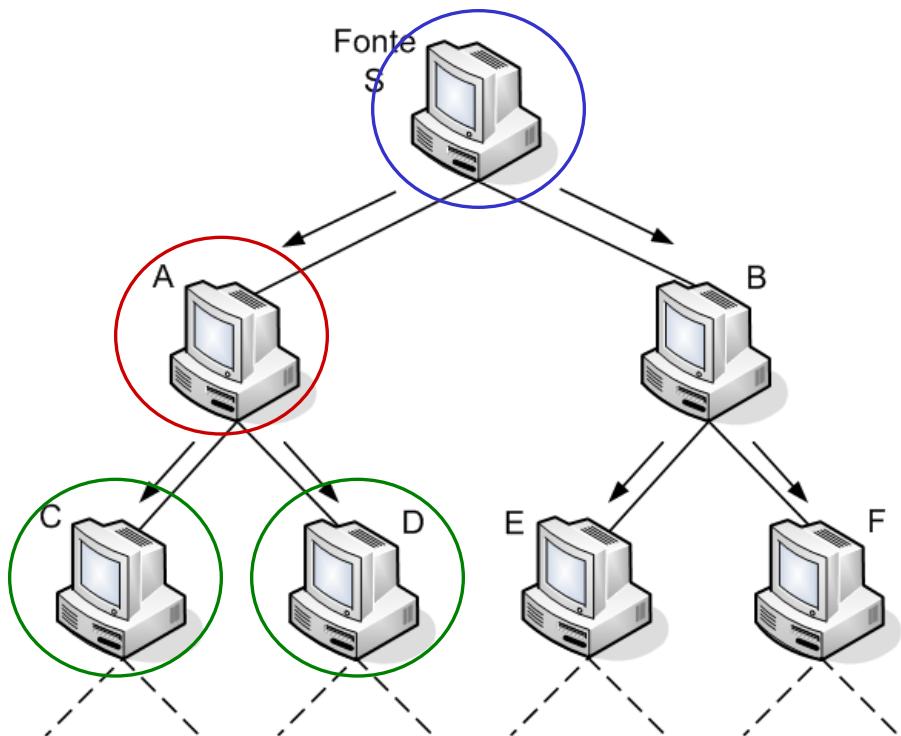


Malha

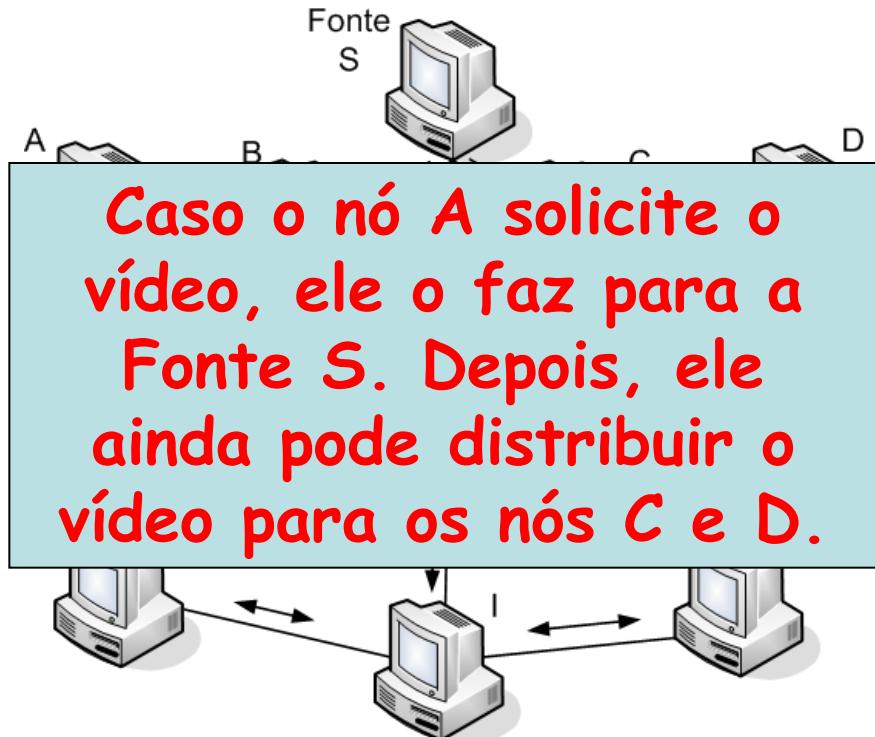


Arquiteturas de Distribuição

Árvore

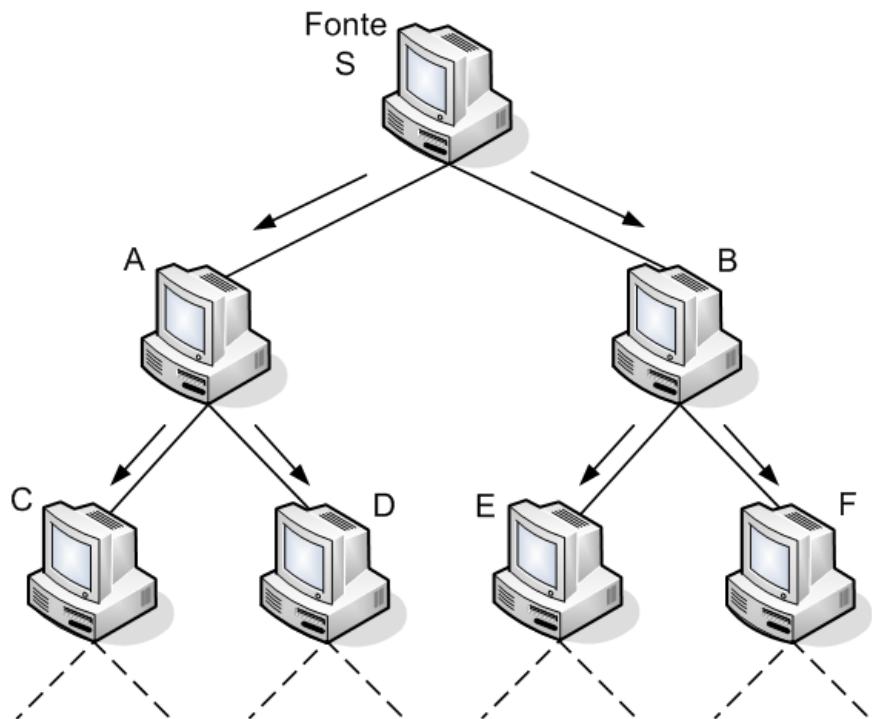


Malha

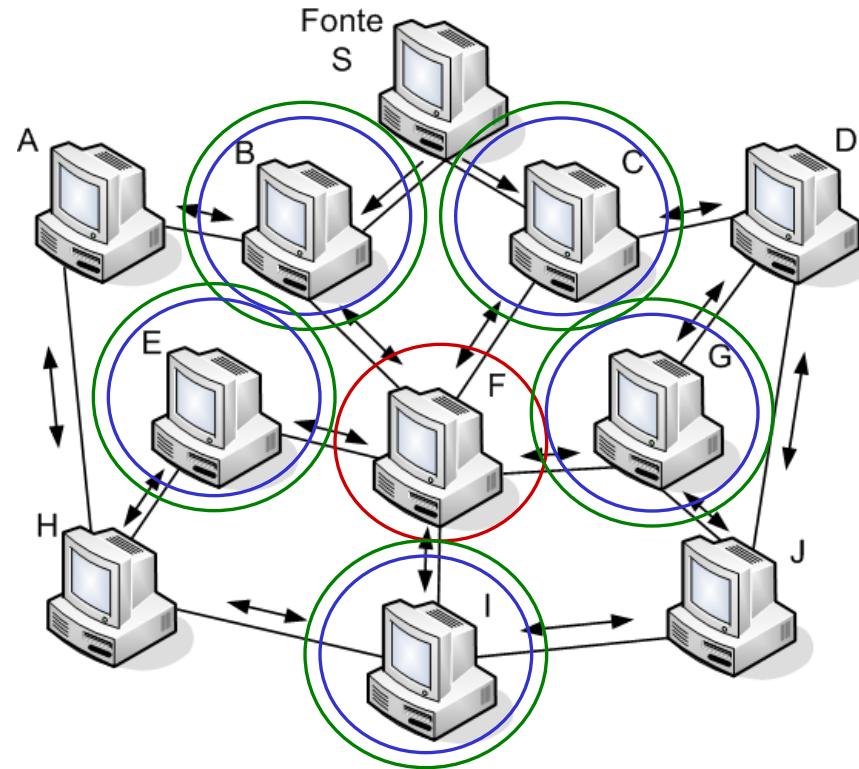


Arquiteturas de Distribuição

Árvore

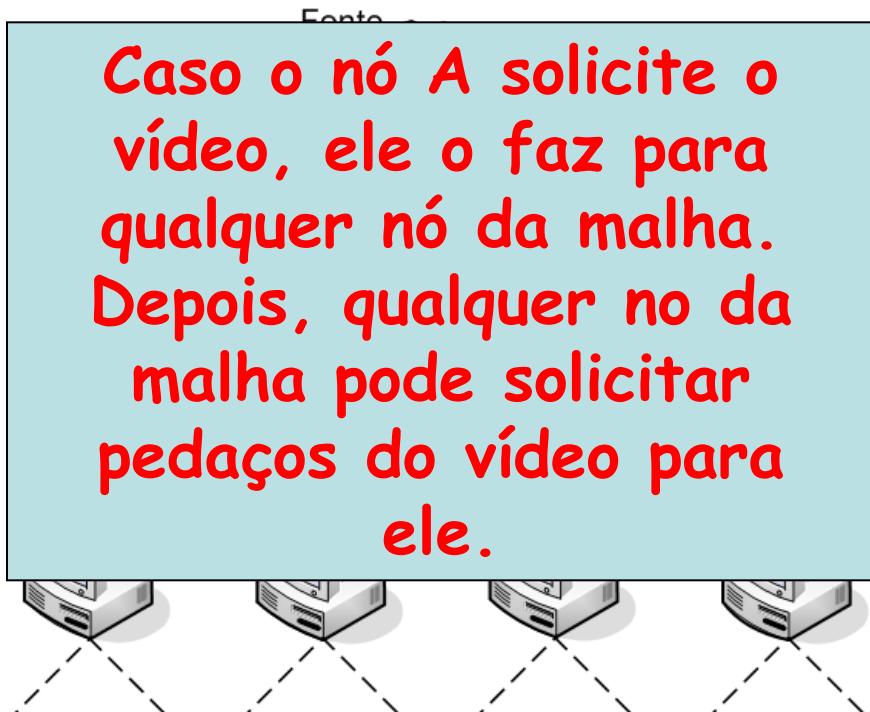


Malha

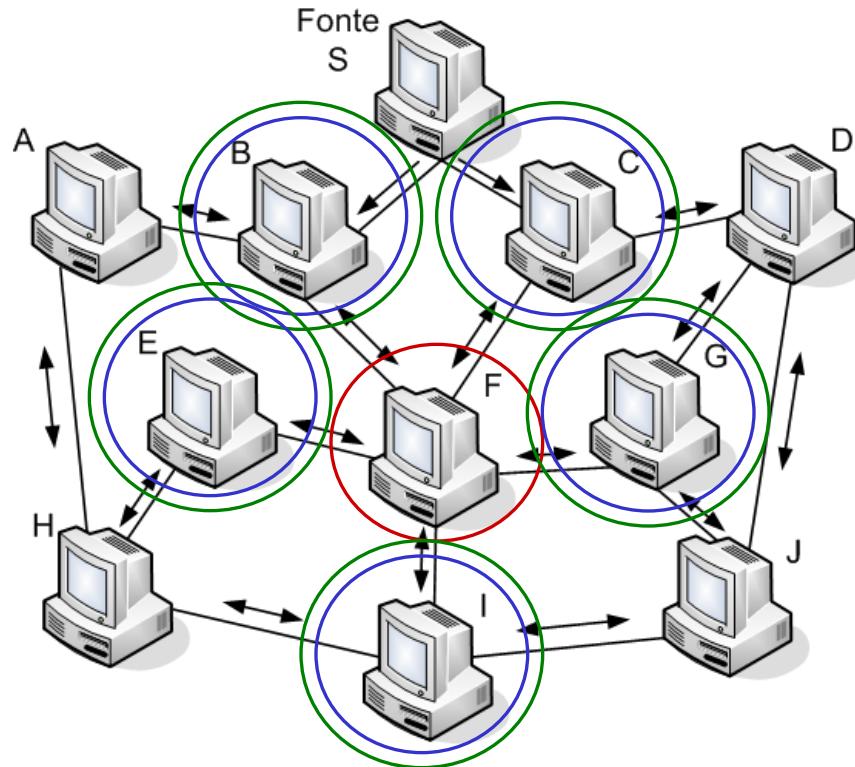


Arquiteturas de Distribuição

Árvore



Malha



Arquiteturas de Distribuição

Árvore	Malha
Comunicação multidestinatária	Divisão do vídeo em pedaços
Hierarquia no encaminhamento	Sem hierarquia no encaminhamento
Uma requisição à fonte	Uma requisição a cada pedaço
Aumentar a eficiência do encaminhamento	Aumentar a robustez à dinâmica dos participantes

Arquiteturas de Distribuição

Árvore	Malha
Comunicação multidestinatária	Divisão do vídeo em pedaços
Hierarquia no encaminhamento	Sem hierarquia no encaminhamento
Uma requisição à fonte	Uma requisição a cada pedaço
Aumentar a eficiência do encaminhamento	Aumentar a robustez à dinâmica dos participantes

Arquitetura em Árvore

- Comunicação multidestinatária na camada de aplicação
 - IP Multicast
 - Um nó se inscreve na fonte
 - Pais encaminham cópias dos pacotes para os filhos
 - Sem requisição
 - Conteúdo é empurrado para os participantes
 - Procedimento do tipo **PUSH**

Arquitetura em Árvore

- Desempenho afetado pela dinâmica dos participantes
- Se entrada e saída de pares não for frequente
 - Baixa latência e baixa sobrecarga de controle
 - Árvore construída → somente encaminhamento de dados
- Se saída de pares for frequente
 - Aumento da sobrecarga de controle
 - Reconstrução da árvore
 - Interrupção do fluxo
 - Descendentes podem deixar de receber o vídeo

Arquitetura em Árvore

- Pacotes de um fluxo → mesmo caminho até um receptor
 - Balancear o número de filhos
 - Evitar congestionamentos
- Maioria dos participantes são folhas
 - Não contribuem com recursos
 - Não possuem filhos
- Heterogeneidade dos receptores
 - A capacidade do pai influencia a recepção nos filhos

Arquitetura em Árvore

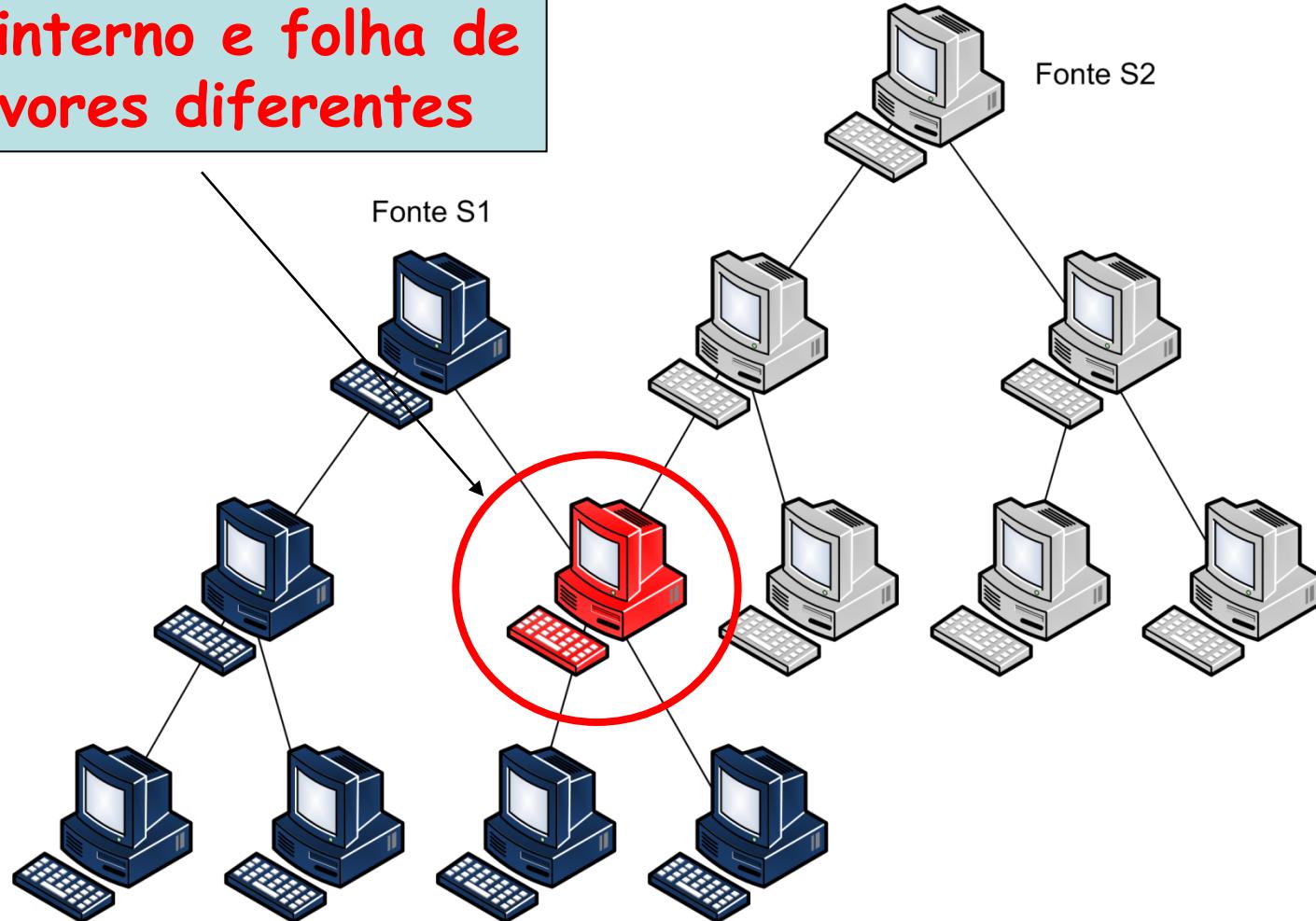
- Múltiplas árvores
 - Maior robustez em relação a dinâmica dos participantes
 - Menos sensível a heterogeneidade dos receptores
- Ideia
 - Receptores com diferentes capacidades → vídeos de diferentes qualidades
 - Usar a codificação em camadas ou *MDC (Multiple Description Coding)*
 - MDC: Divisão do vídeo em subfluxos
 - Cada subfluxo encaminhado em uma árvore diferente
 - Um nó se inscreve em um dado número de árvores
 - Mais árvores → mais qualidade

Arquitetura em Árvore

- Em arquiteturas com múltiplas árvores: Um nó só é interno em uma das árvores
 - Nas demais, é folha
 - Minimizar os efeitos da saída de um nó
 - A saída de um antecessor em uma das árvores não acarreta na interrupção do vídeo
 - Apenas pode reduzir a qualidade do vídeo
 - Maximizar a utilização da banda passante compartilhada
 - Todos os nós contribuem com recursos
 - Não podem ser folhas em todas as árvores

Arquitetura em Árvore

Nó interno e folha de árvores diferentes



Arquitetura em Malha

- Divisão do vídeo em pedaços (*chunks*)
 - Pedaços espalhados pelos nós participantes
 - Não há uma estrutura explícita de comunicação
 - Não é eficiente manter uma estrutura fixa
 - Como na arquitetura em árvore
 - É mais eficiente disseminar a disponibilidade dos pedaços
 - Conjunto de parceiros
 - Uma requisição para cada pedaço
 - Conteúdo é puxado pelos participantes
 - Procedimento do tipo **PULL**
 - Comunicação ponto-a-ponto na camada de aplicação

Arquitetura em Malha

- Participantes não possuem funções específicas
 - Recebem de qualquer nó
 - Encaminham para qualquer nó
 - Sem uma organização hierárquica
- Menos suscetível à dinâmica dos participantes
 - Pedaços disponíveis em vários nós
 - Pedaços recebidos de diferentes parceiros
 - Não somente de um nó pai



Reducem a probabilidade de descontinuidade

Arquitetura em Malha

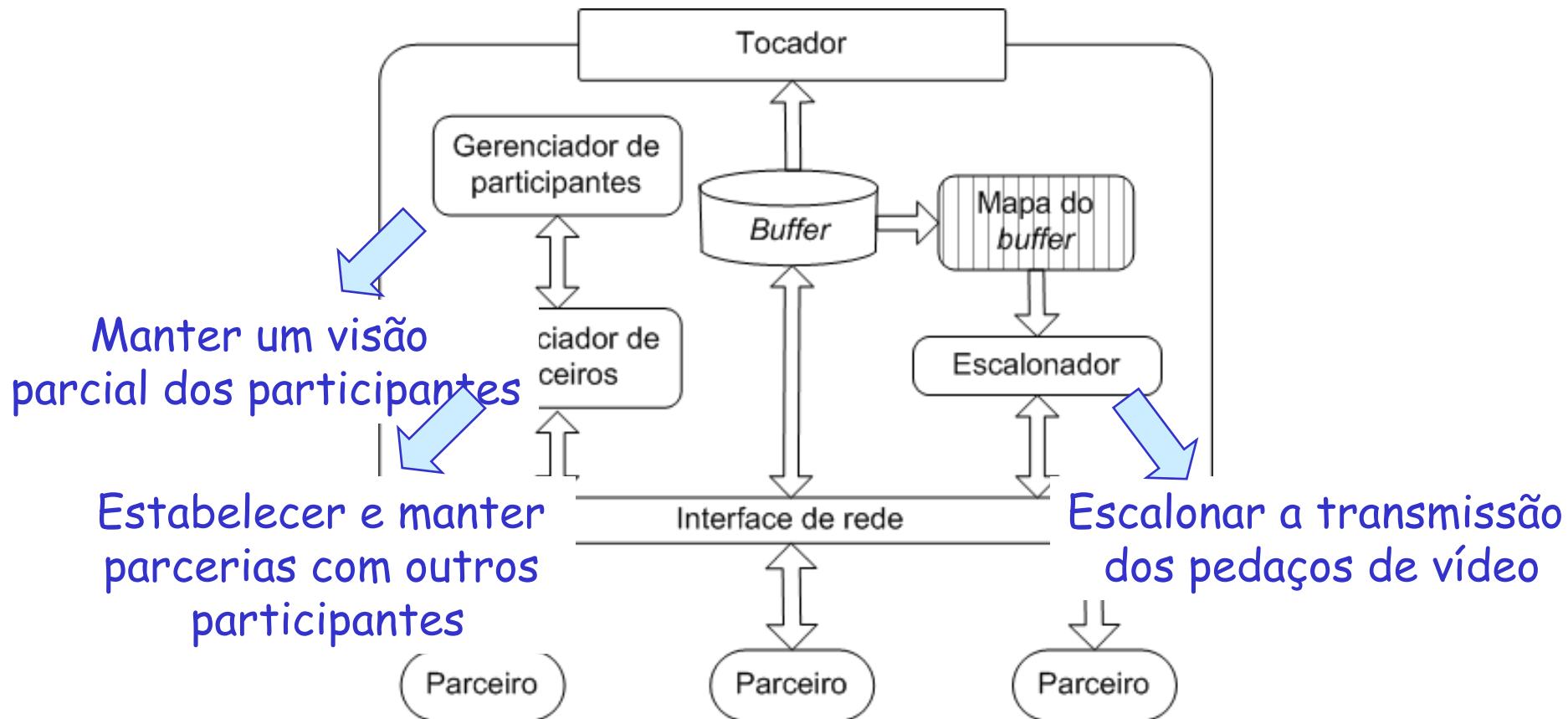
- Maior sobrecarga de controle
 - Trocar informações sobre a disponibilidade dos pedaços
 - Uma requisição por pedaço
- Maiores atrasos
 - Inicialização e encaminhamento do vídeo
 - Não há uma estrutura explícita de distribuição
 - Caminhos não são otimizados

Arquitetura em Malha

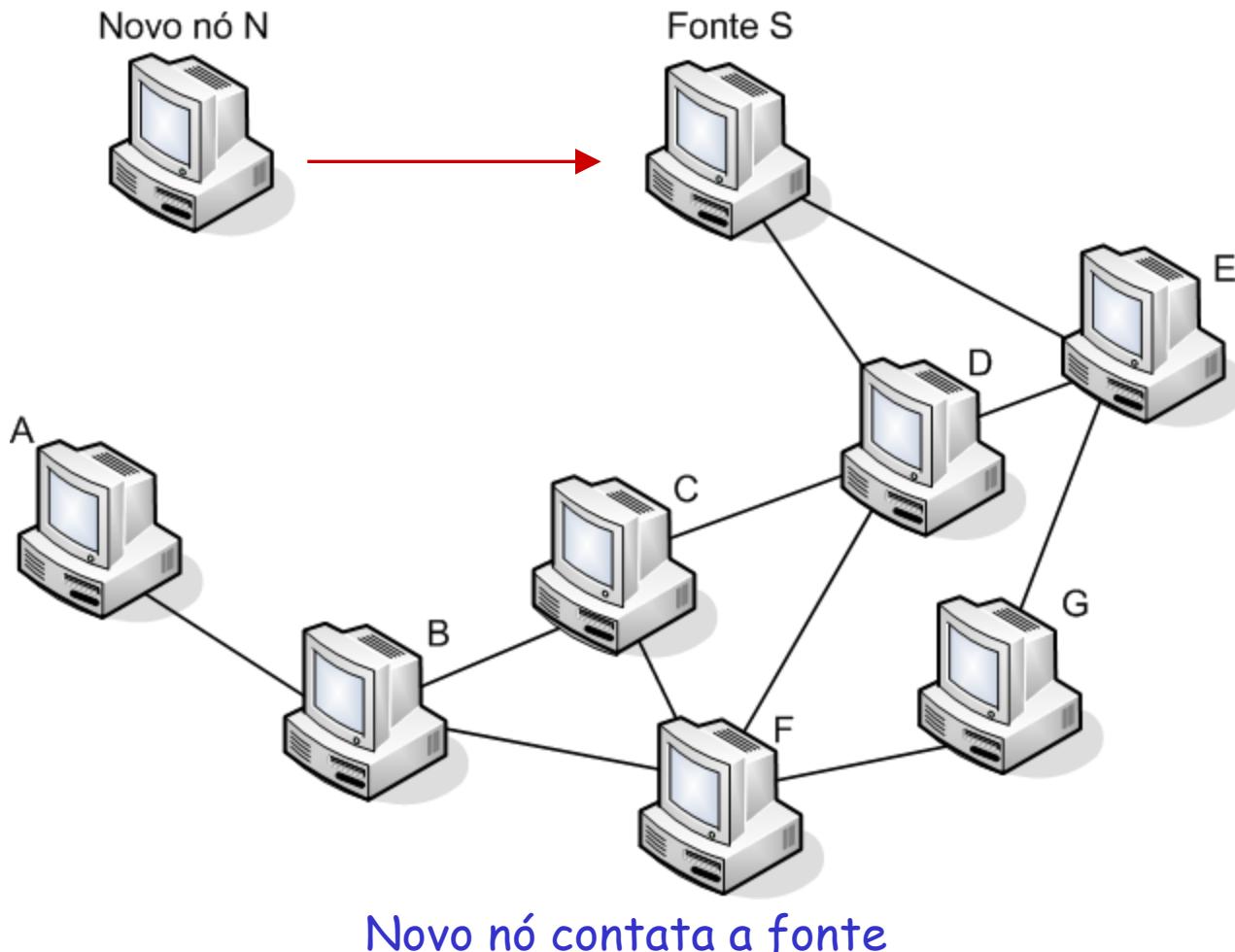
- Desempenho depende do tamanho dos *buffers*
 - Mais pedaços podem ser armazenados
 - Maior disponibilidade
 - Pedaços fora de ordem devem ser reordenados
 - Mais fácil se o *buffer* for maior

CoolStreaming/DONet

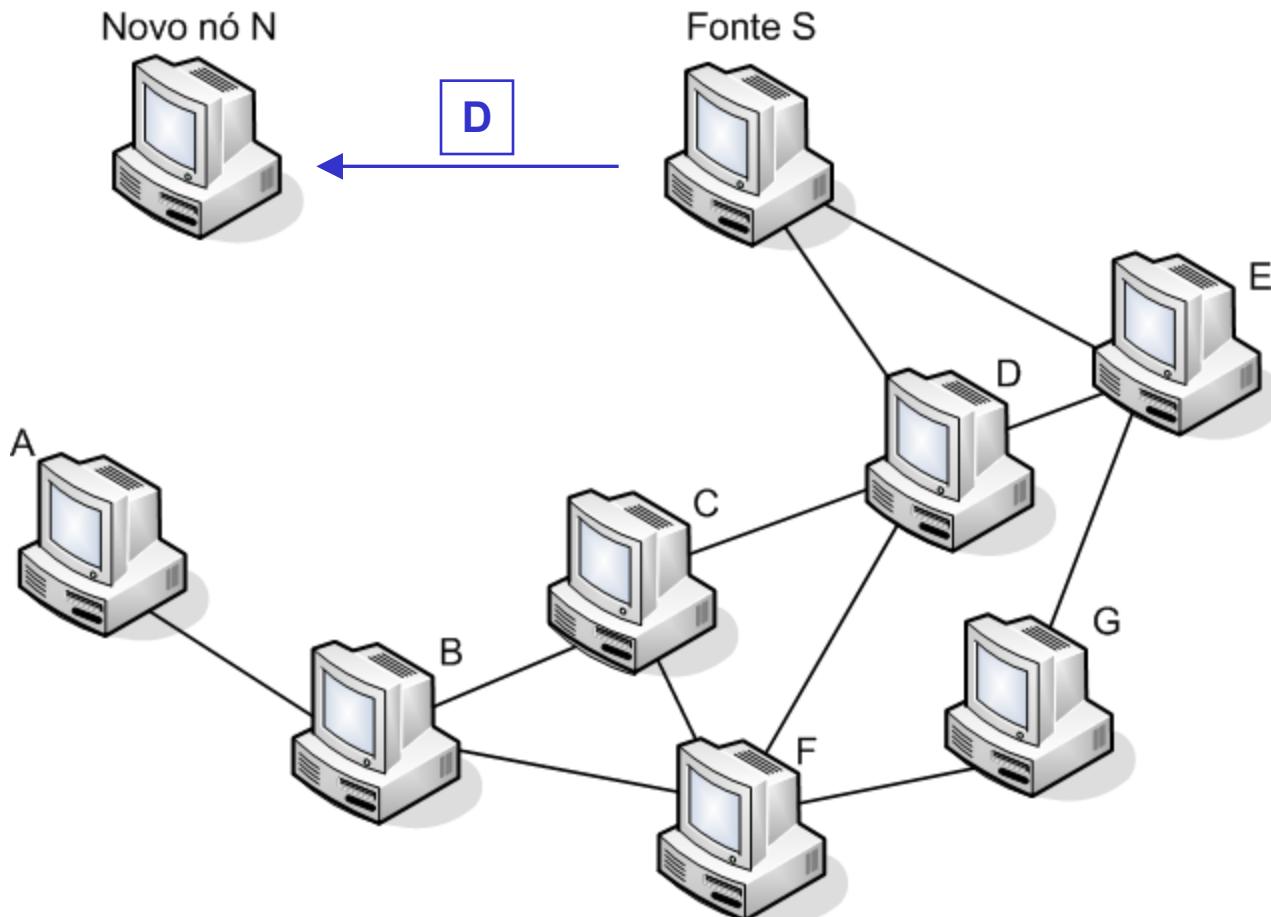
- Estrutura de um nó



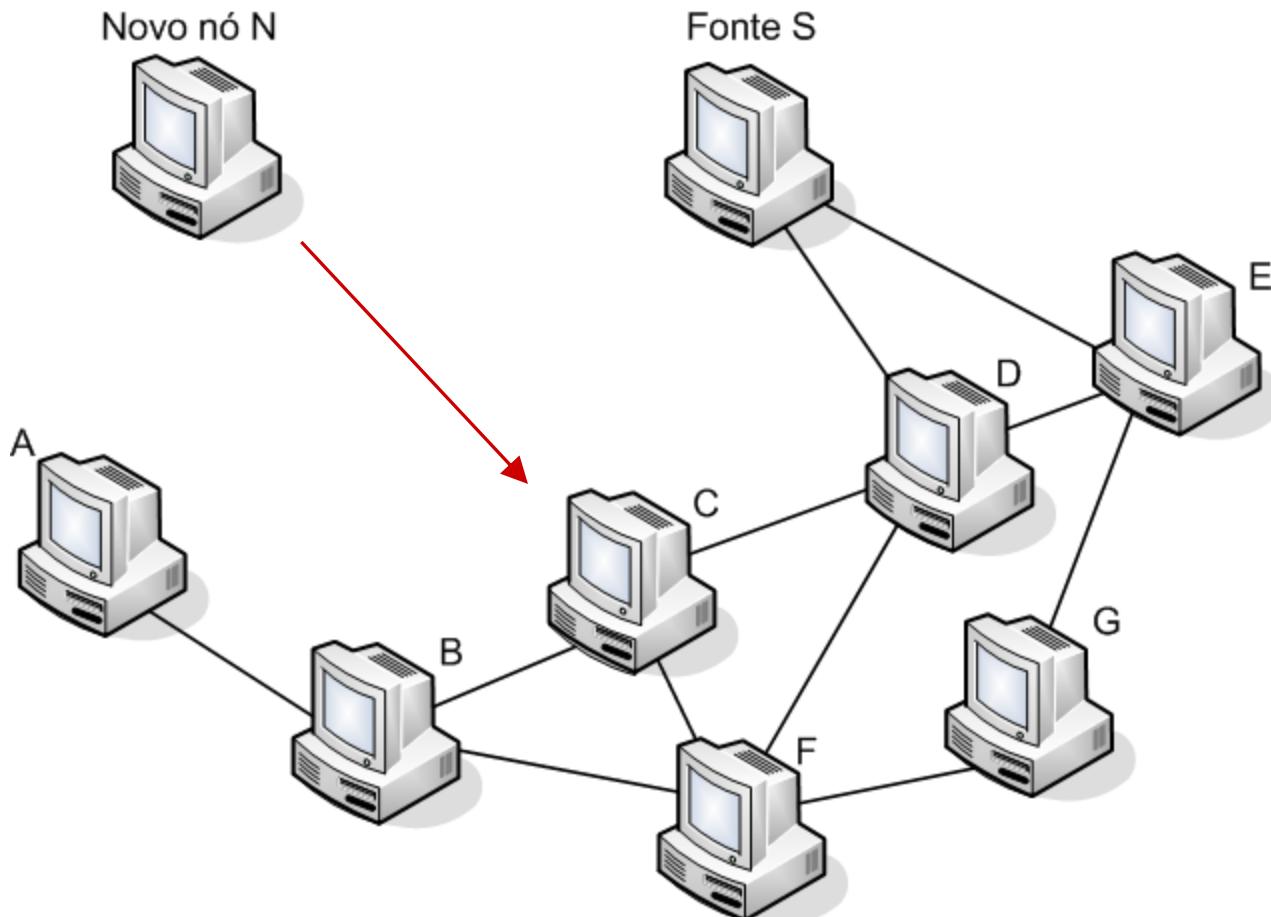
Construção da Malha



Construção da Malha

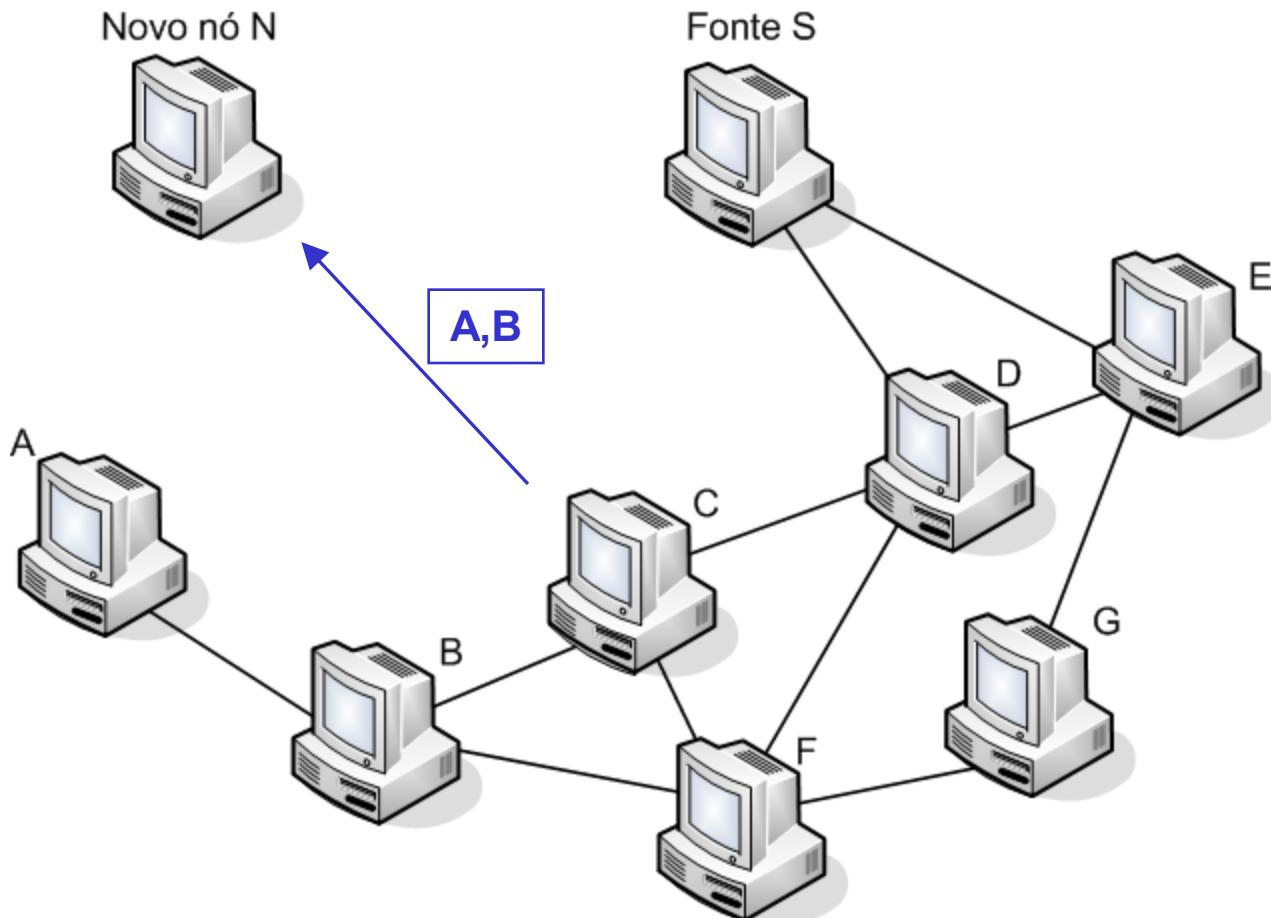


Construção da Malha



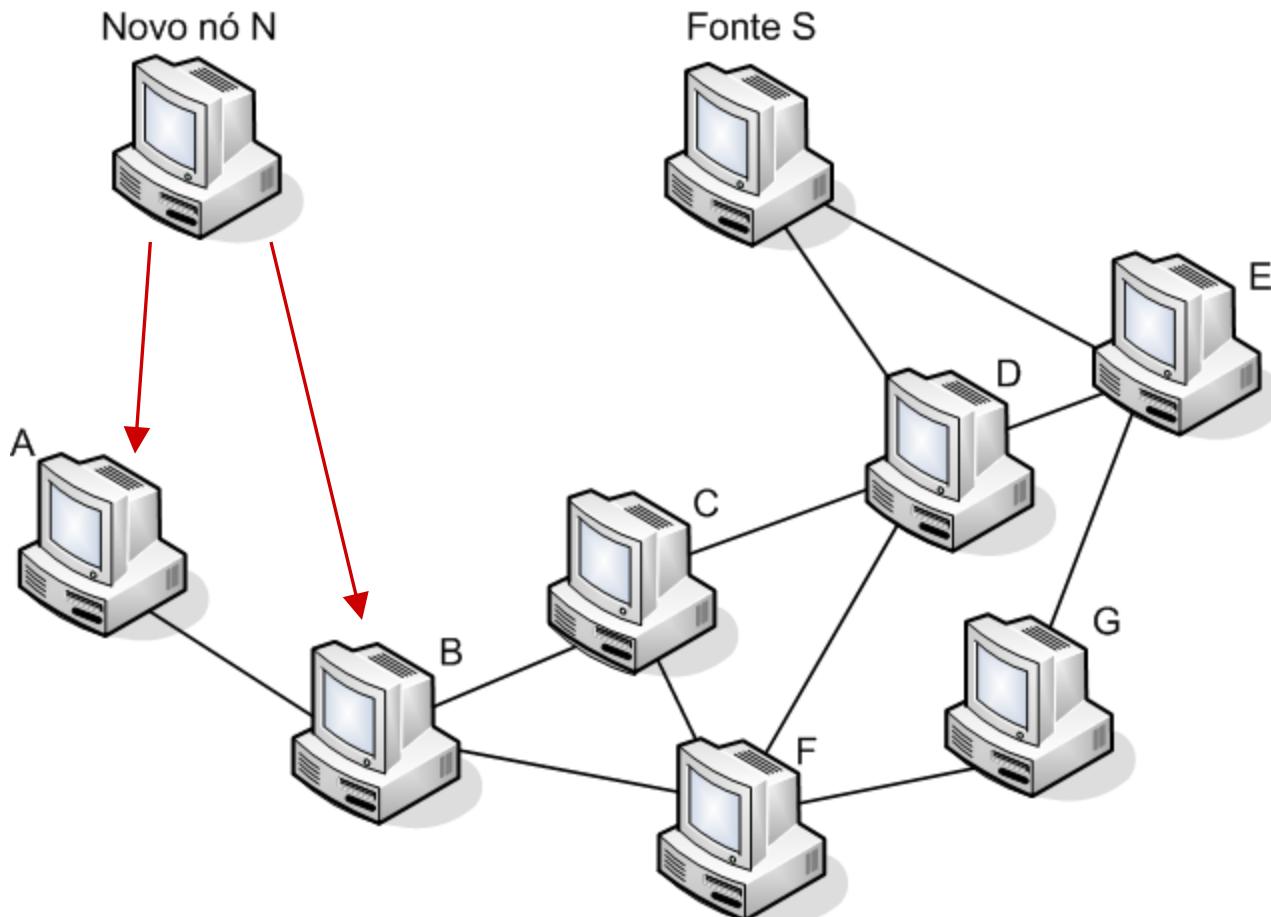
Novo nó contata o adjunto

Construção da Malha



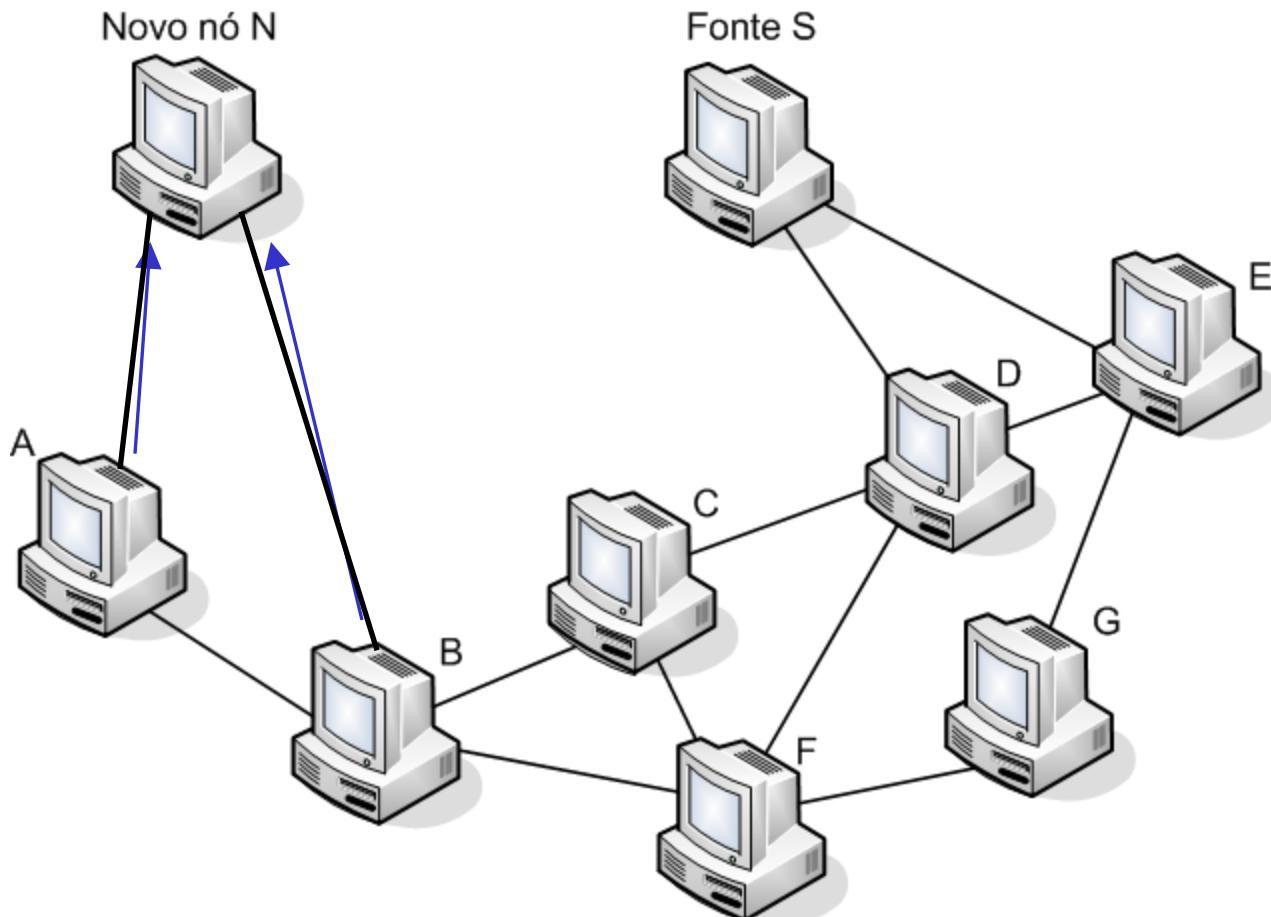
Adjunto responde com a lista de candidatos

Construção da Malha



Novo nó envia uma mensagem para estabelecer parcerias com os candidatos

Construção da Malha



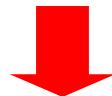
Em caso de resposta positiva, os enlaces são criados

Construção da Malha

- Por que usar um nó adjunto?
 - Reduzir a carga da fonte
 - Tornar a seleção de parceiros mais uniforme

Escalonamento de Pedaços

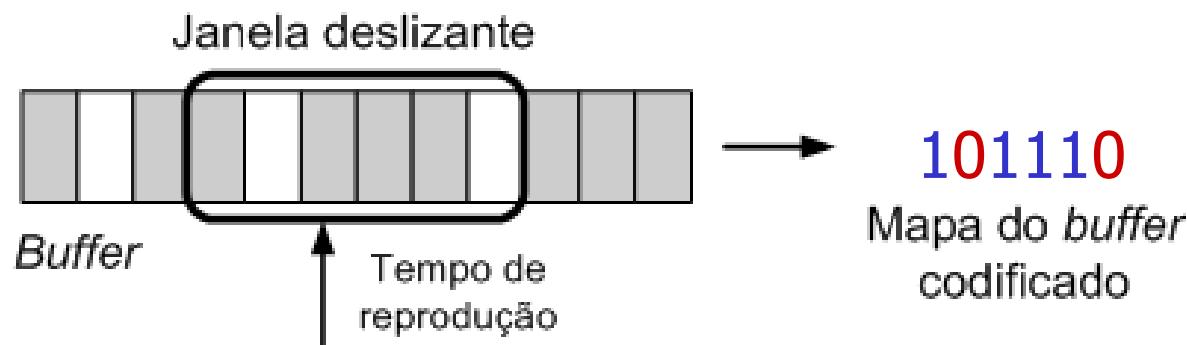
- Fundamental para garantir os requisitos de QoS do vídeo
 - Pedaços devem ser recebidos antes do tempo de reprodução
 - Podem ser recebidos fora de ordem
 - Progresso de reprodução fortemente sincronizado
 - Na difusão, não é possível controlar a reprodução
 - Interesse pelo conteúdo em um dado período
 - Intervalo de trechos reproduzidos por participantes: 1 minuto



Diferença para o compartilhamento de arquivos

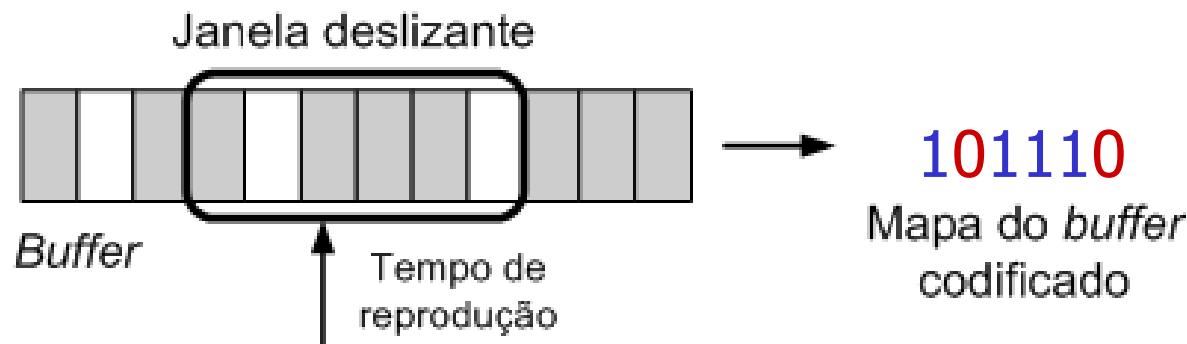
Escalonamento de Pedaços

- Pedaços de tamanho uniforme
- A disponibilidade é representada pelo mapa de *buffer* (BM)
 - Janela deslizante
 - Pedaços só são úteis se forem recebidos antes do tempo
 - No exemplo: capacidade do *buffer* = 12 e tamanho do BM=6



Escalonamento de Pedaços

- Configuração padrão
 - Pedaços de 1 segundo de vídeo
 - Janela de 120 pedaços
- Codificação
 - Se o pedaço está disponível → 1
 - Se o pedaço **não está disponível** → 0



Escalonamento de Pedaços

- Identificação dos pedaços
 - Número de sequência de 2 bytes
 - Somente o número do primeiro pedaço da janela é armazenado
 - É possível identificar os pedaços na janela em um período
 - Número de sequência é incremental
 - Tamanho da janela é fixo

Escalonamento de Pedaços

- Difundir a disponibilidade dos pedaços
 - Troca periódica de mapas de *buffer* entre parceiros
 - Um nó sabe quais pedaços seus parceiros possuem
- Escalonador de pedaços
 - Definir de qual parceiro e quando vai requisitar um pedaço
 - Lidar com
 - Restrições do tempo de reprodução de cada pedaço
 - Heterogeneidade dos parceiros



Escalonamento de máquinas paralelas → NP-completo

Escalonamento de Pedaços

- Heurística
 - Simples e de resposta rápida às variações da malha
 - Requisitos do vídeo
 - Baseada no:
 - Número de emissores potenciais de um pedaço
 - Capacidade de saída de cada possível emissor
 - Suposição
 - Pedaços com menos emissores potencias
 - Maior chance de recepção após o tempo de reprodução
 - Ações
 - Priorizar pedaços com menos emissores
 - Mais de um emissor
 - Escolha do de maior banda de saída

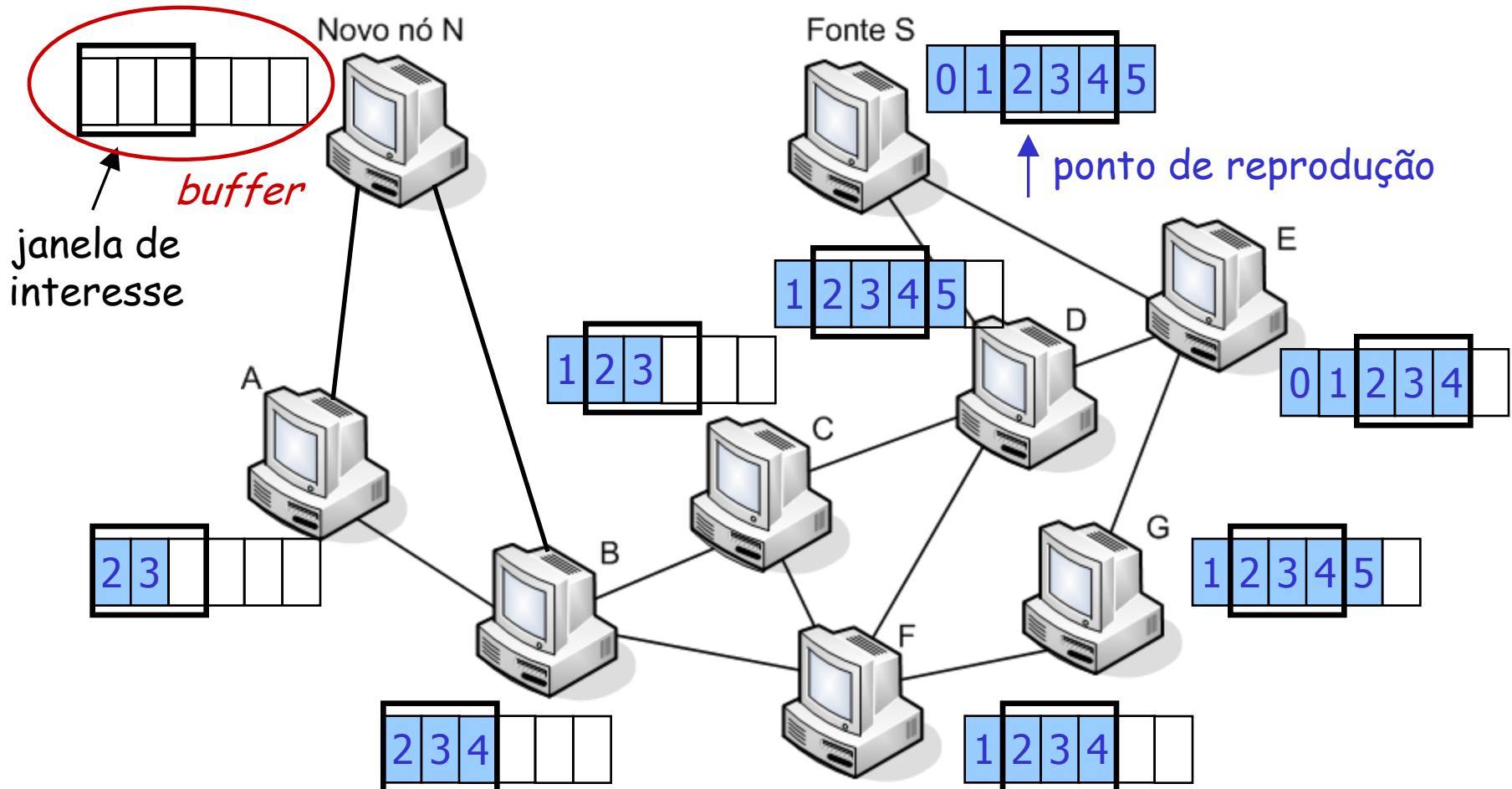
Escalonamento de Pedaços

- Escalonador
 - Executado periodicamente
 - Definir a escala de pedaços a serem requisitados
 - Uma para cada parceiro
 - Representada por um sequência de bits como o BM
 - Enviar a escala para o parceiro correspondente
- Ao receber a escala
 - Parceiro envia os pedaços requisitados ordenadamente

Escalonamento de Pedaços

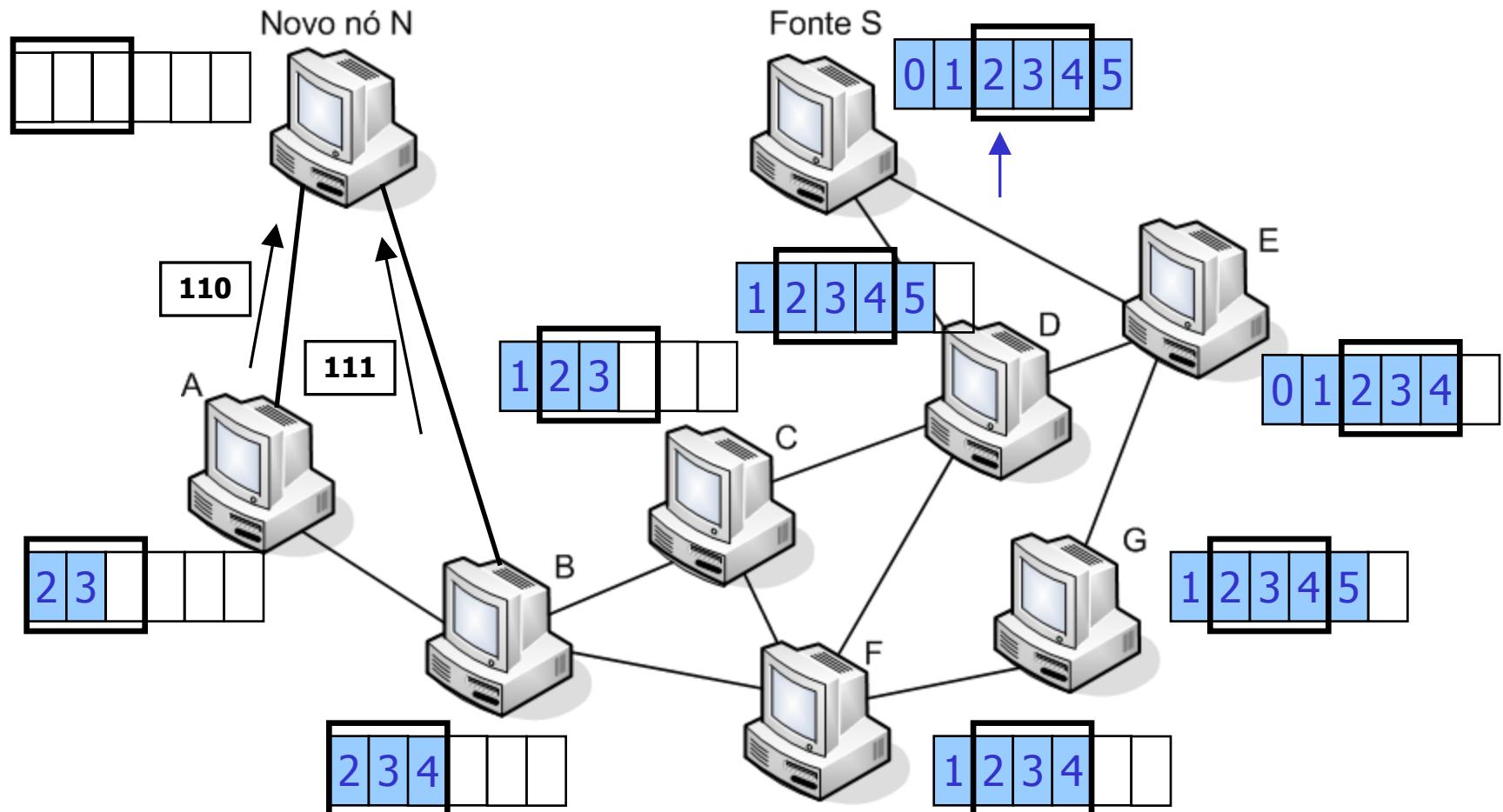
- **Fonte**
 - Possui todos os pedaços
- **Escalonamento adaptativo**
 - Implementado pela fonte
 - Evitar a sobrecarga de requisições dos parceiros
 - Difundir um mapa de *buffer conservativo*
 - Nem todos os pedaços disponíveis
 - Bits em zero
 - Parceiros deixam de requisitá-los à fonte

Sistemas de Difusão



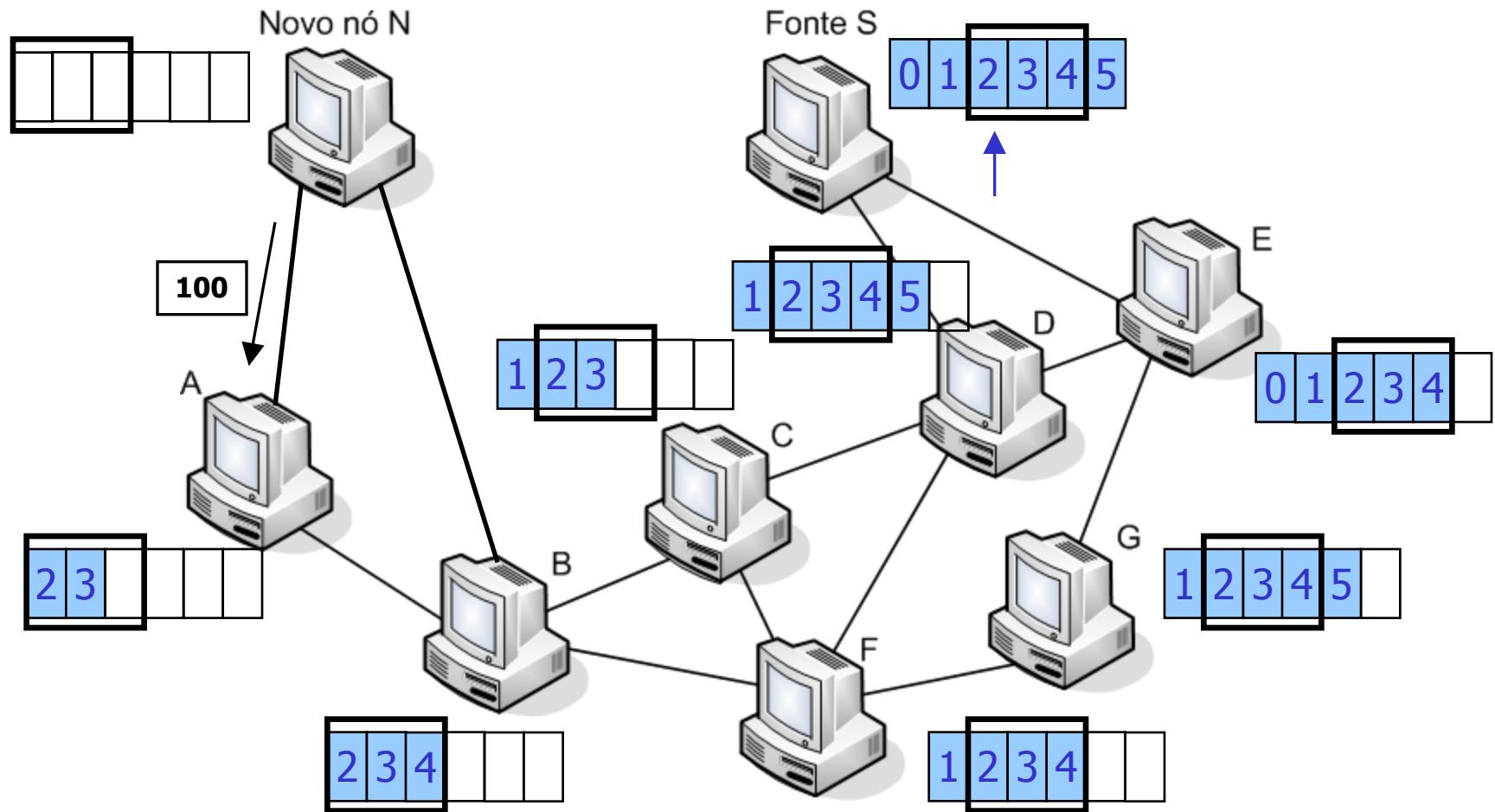
A reprodução começa a partir do ponto atual da fonte

Sistemas de Difusão

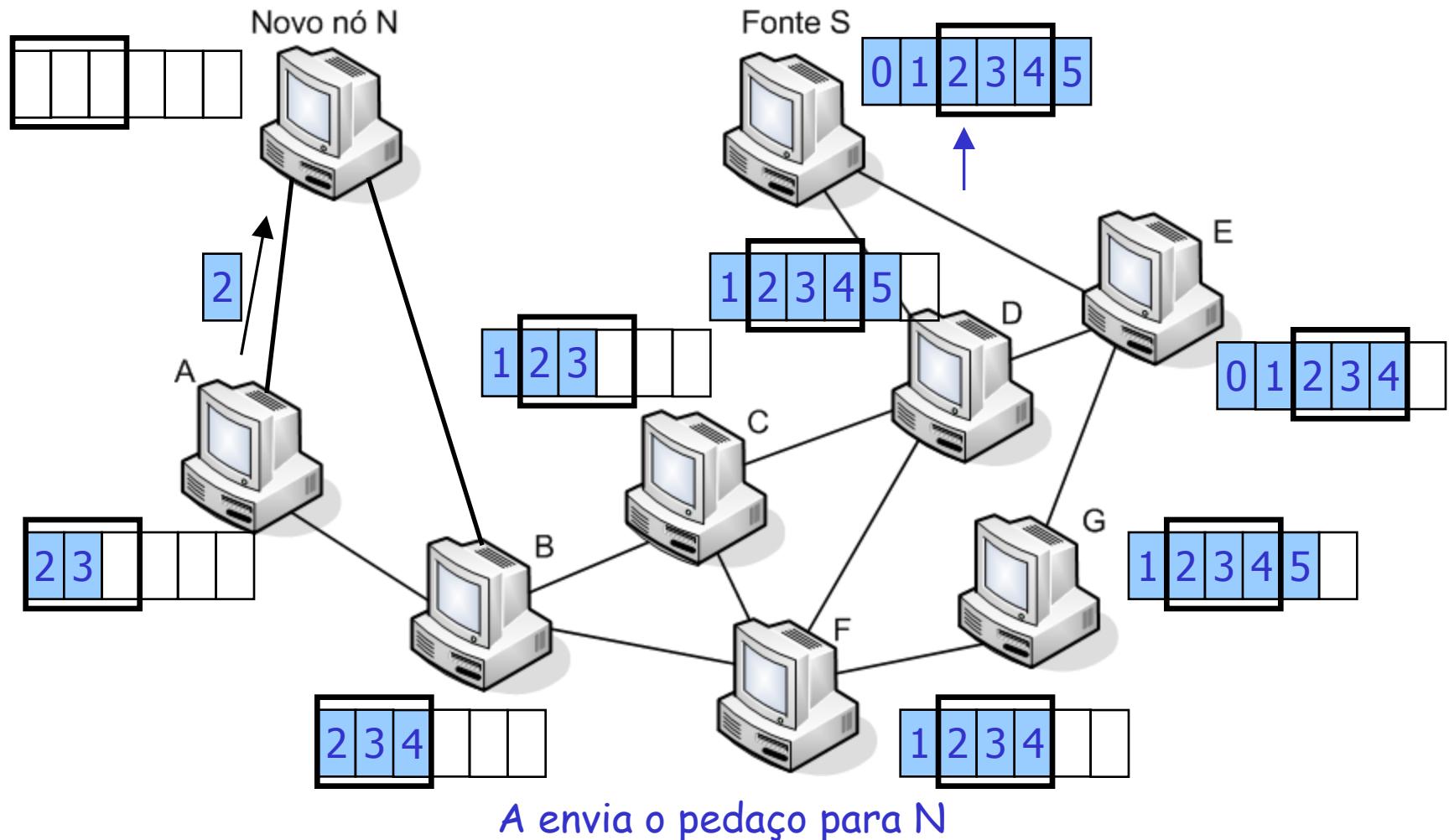


O nó N recebe os BMs dos seus parceiros

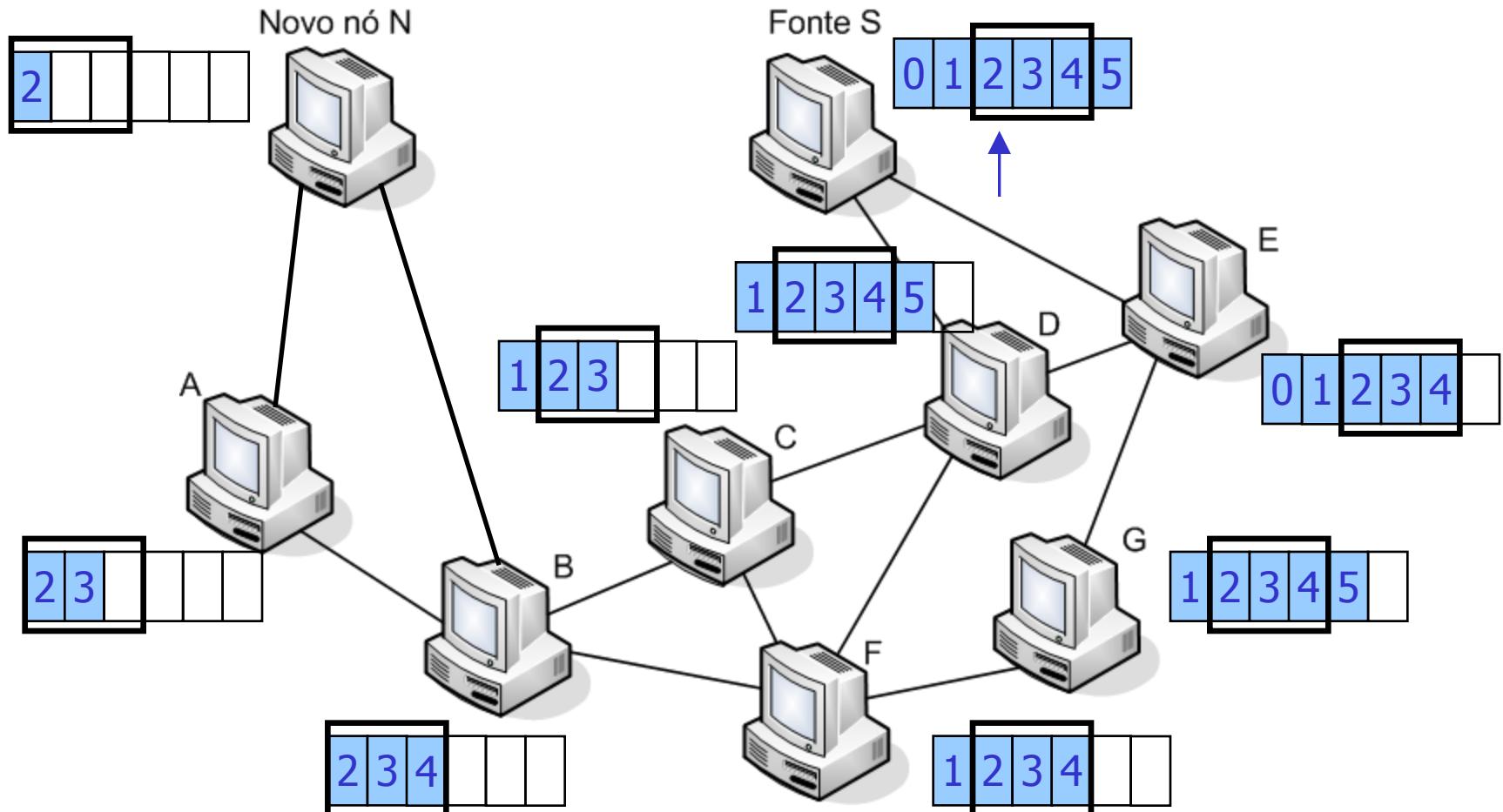
Sistemas de Difusão



Sistemas de Difusão



Sistemas de Difusão



O pedaço é armazenado no *buffer*

Árvore x Malha

- Arquitetura em árvore
 - Reduz a latência do encaminhamento do vídeo
 - Implementar a comunicação multidestinatária na camada de aplicação
 - Possui problemas
 - Instabilidade provocada pela saída de participantes
 - Sobrecarga de controle para manter a malha conectada
 - Subutilização de banda passantes
 - Nós folhas
 - Escolha ineficiente de nós pai

Árvore x Malha

- Arquitetura em malha
 - Aumenta a disponibilidade do conteúdo
 - Tornar o sistema mais robusto à dinâmica dos participantes
 - Possui problemas
 - Encaminhamento menos eficiente
 - Comunicações ponto-a-ponto entre os parceiros
 - Compromisso entre latência e sobrecarga de controle
 - Notificação de recepção de cada pedaço → maior sobrecarga
 - Enviar os BMs com as requisições → menor sobrecarga, mas maior latência já que não se sabe quem tem os pedaços desejados

Material Utilizado

- Notas de aula do Prof. Igor Monteiro Moraes,
disponíveis em
<http://www2.ic.uff.br/~igor/cursos/redespg>

Leitura Recomendada

- Capítulo 2 do Livro "*Computer Networking: A Top Down Approach*", 5a. Ed., Jim Kurose and Keith Ross, Pearson, 2010
- Capítulo 7 do Livro "*Computer Networks*", Andrew S. Tanenbaum e David J. Wetherall, 5a. Ed., Pearson, 2011
- Moraes, I. M., Campista, M. E. M., Moreira, M. D. D., Rubinstein, M. G., Costa, L. H. M. K., and Duarte, O. C. M. B. - "Distribuição de Vídeo sobre Redes Par-a-Par: Arquiteturas, Mecanismos e Desafios", in Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC'2008, pp. 115-171, Rio de Janeiro, RJ, Brazil, May 2008