

Parte 1

Camada de Aplicação

- 1 Fundamentos
- 2 A Web e o HTTP
- 3 DNS: O Serviço de Diretório da Internet
- 4 Correio Eletrônico na Internet
- 5 Transferência de Arquivos: FTP
 - Fundamentos
 - Comandos e Respostas FTP
- 6 Aplicações Peer-to-Peer
 - Fundamentos
 - Distribuição de Arquivos P2P
 - Escalabilidade da Arquitetura P2P
 - Protocolo BitTorrent

Aplicações de Redes



A camada de Aplicação fornece a conexão para a rede.

Aplicações de Redes

- As aplicações são a **razão de ser** de uma rede de computadores;
- As aplicações de redes (Web, e-mail, compartilhamento de arquivos, etc) fornecem **interface à rede subjacente**, permitindo que enviamos e recebemos informações de forma relativamente fácil;
- Para profissionais de rede, é importante saber como uma aplicação pode **formatar, transmitir e interpretar mensagens** enviadas e recebidas através da rede;

Aplicações de Redes



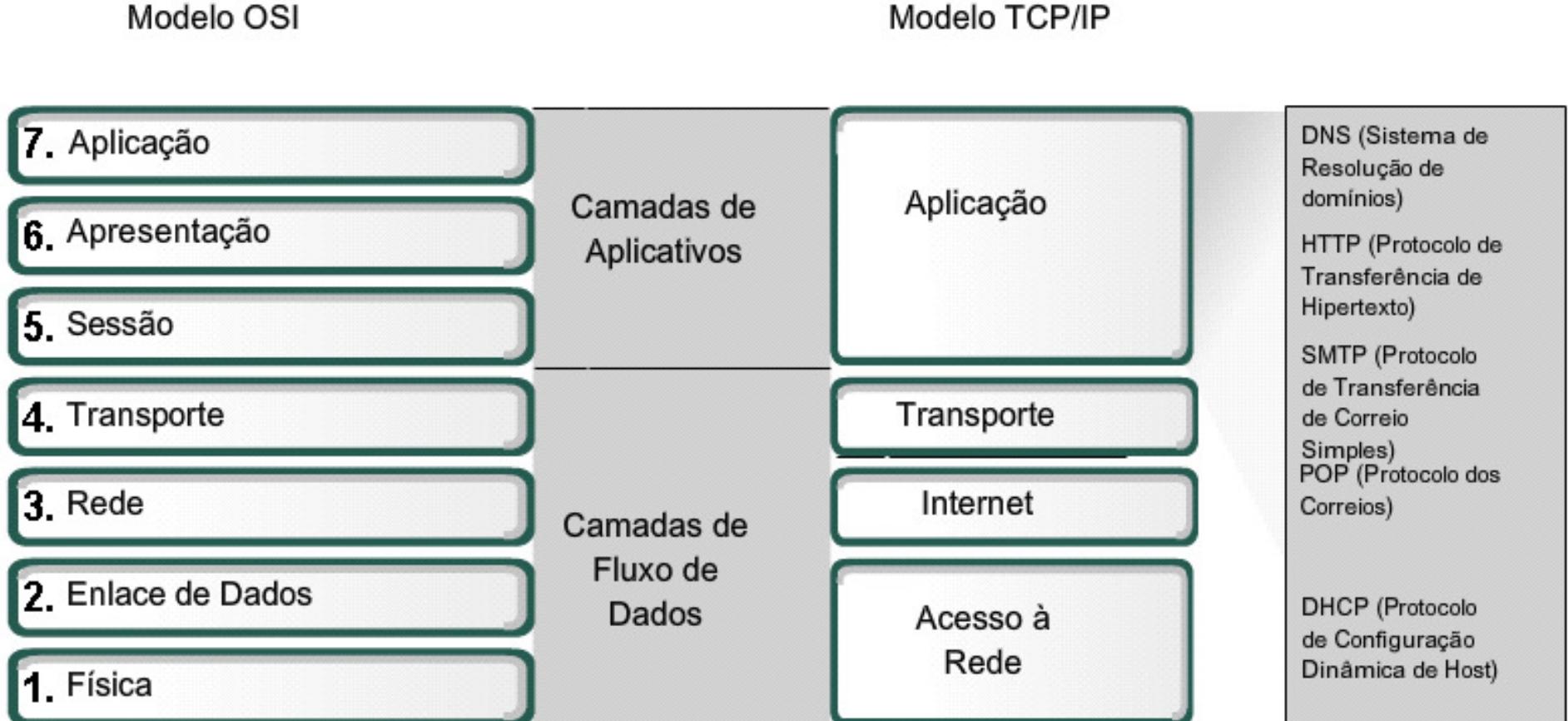
A camada de Aplicação prepara comunicação humana para transmissão através da rede de dados.



Aplicações de Redes



Modelo OSI e TCP/IP



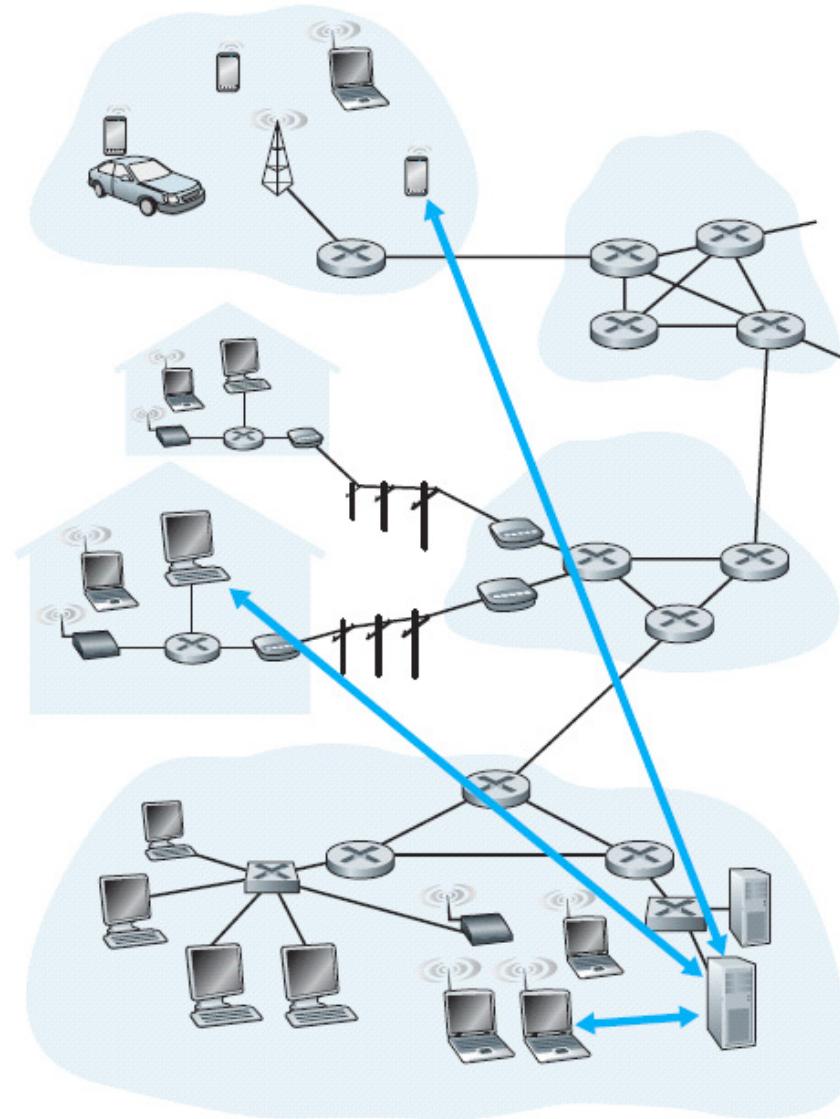
Arquiteturas de Aplicação de Rede

- A arquitetura da aplicação é projetada pelo desenvolvedor e determina **como a aplicação é organizada** nos vários sistemas finais (hosts);
- Duas arquiteturas mais utilizadas em aplicações modernas de redes:
 - **Arquitetura cliente-servidor;**
 - **Arquitetura P2P (Peer-to-Peer).**

Programas Clientes e Servidores

- Em redes modernas, um host pode agir como um **cliente**, um **servidor**, ou ambos. O software instalado no host determina qual papel ele desempenha na rede;
- Servidores são hosts que têm software instalado que os permite fornecer informação e serviços, como e-mail ou páginas web, a outros hosts na rede;
- Clientes são hosts que têm software instalado que os permite solicitar e exibir as informações obtidas do servidor;
- Muitas aplicações são, cada vez mais, **peer-to-peer (P2P)**, nas quais os hosts interagem e executam programas que apresentam funções de servidor e de cliente.

A Arquitetura Cliente/Servidor



A Arquitetura Cliente/Servidor

- O dispositivo que **solicita** as informações é chamado de **cliente**;
- O dispositivo que **responde** à solicitação é chamado de **servidor**;
- Os processos de cliente e servidor são considerados como estando na camada de Aplicação.

A Arquitetura Cliente/Servidor

- Além da transferência real de dados, esse intercâmbio também pode exigir informações de controle, como autenticação de usuário e identificação de um arquivo de dados a ser transferido;
- A transferência de dados de um cliente para um servidor é conhecida como **upload**;
- A transferência de dados de um servidor para um cliente é conhecida como **download**.

A Arquitetura Cliente/Servidor

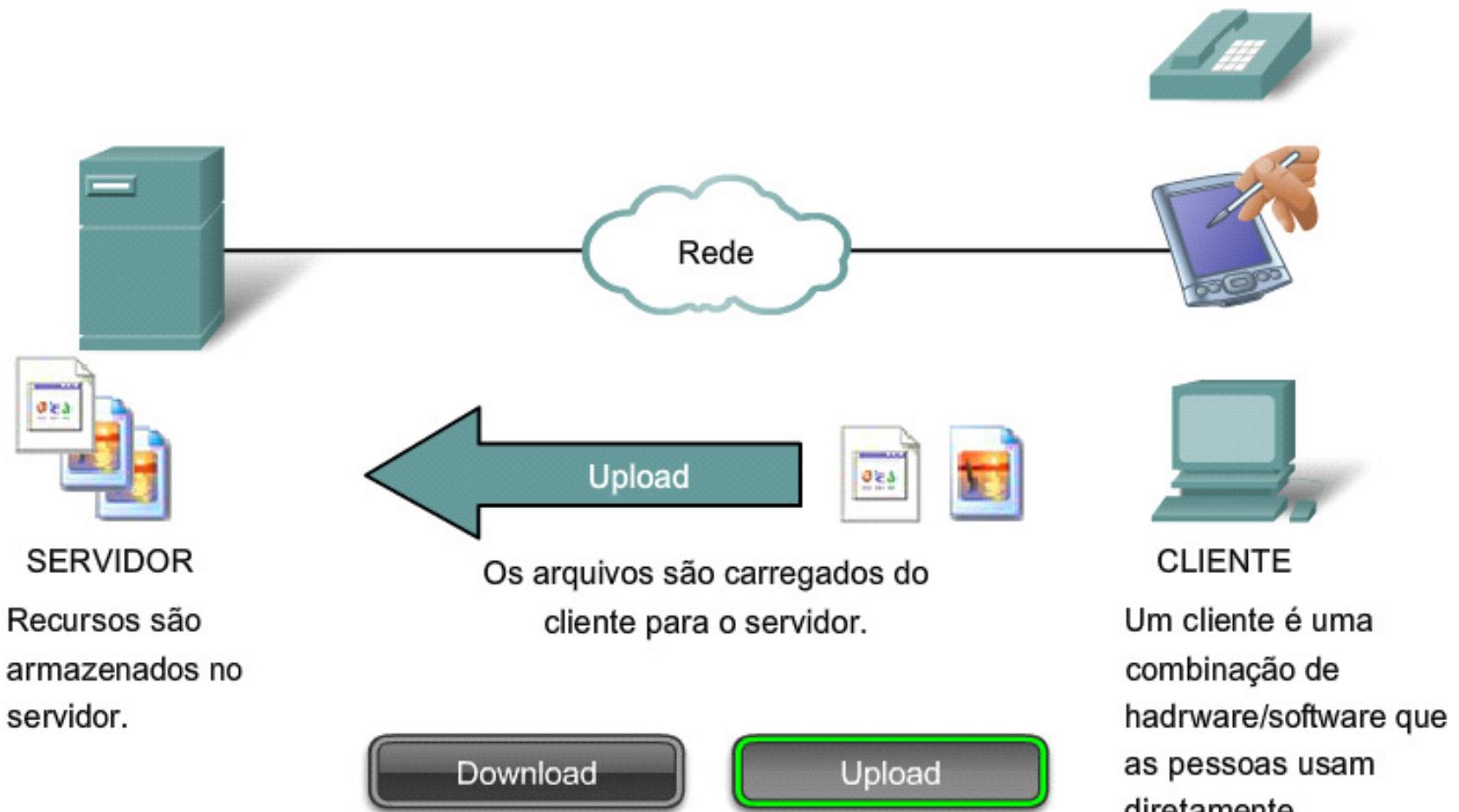
Modelo Cliente/Servidor

Os arquivos são descarregados do sevidor para o cliente.



A Arquitetura Cliente/Servidor

Modelo Cliente/Servidor

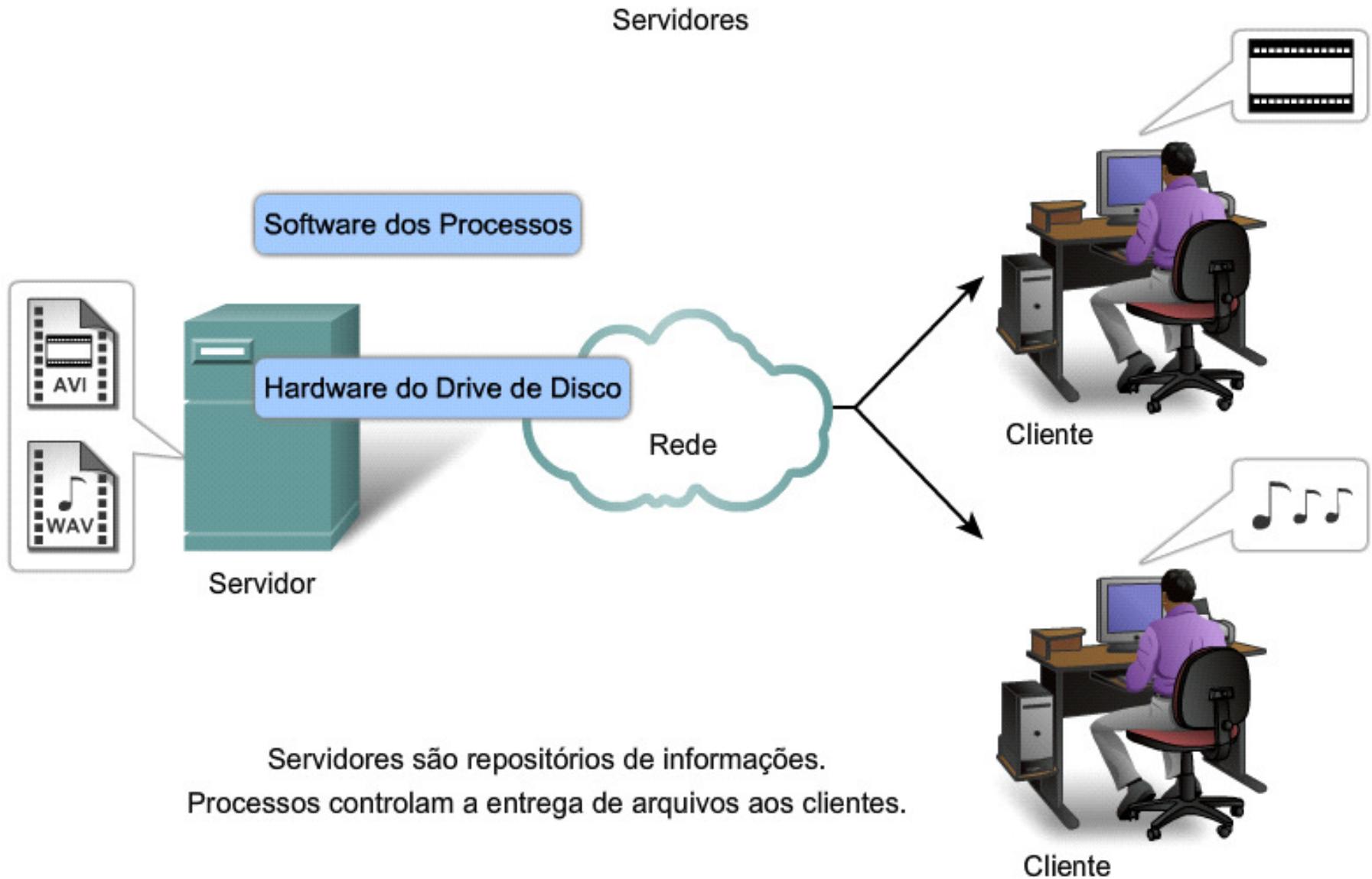


A Arquitetura Cliente/Servidor

Em um contexto geral de rede, qualquer dispositivo que responda a solicitações de aplicações de clientes funciona como um servidor.

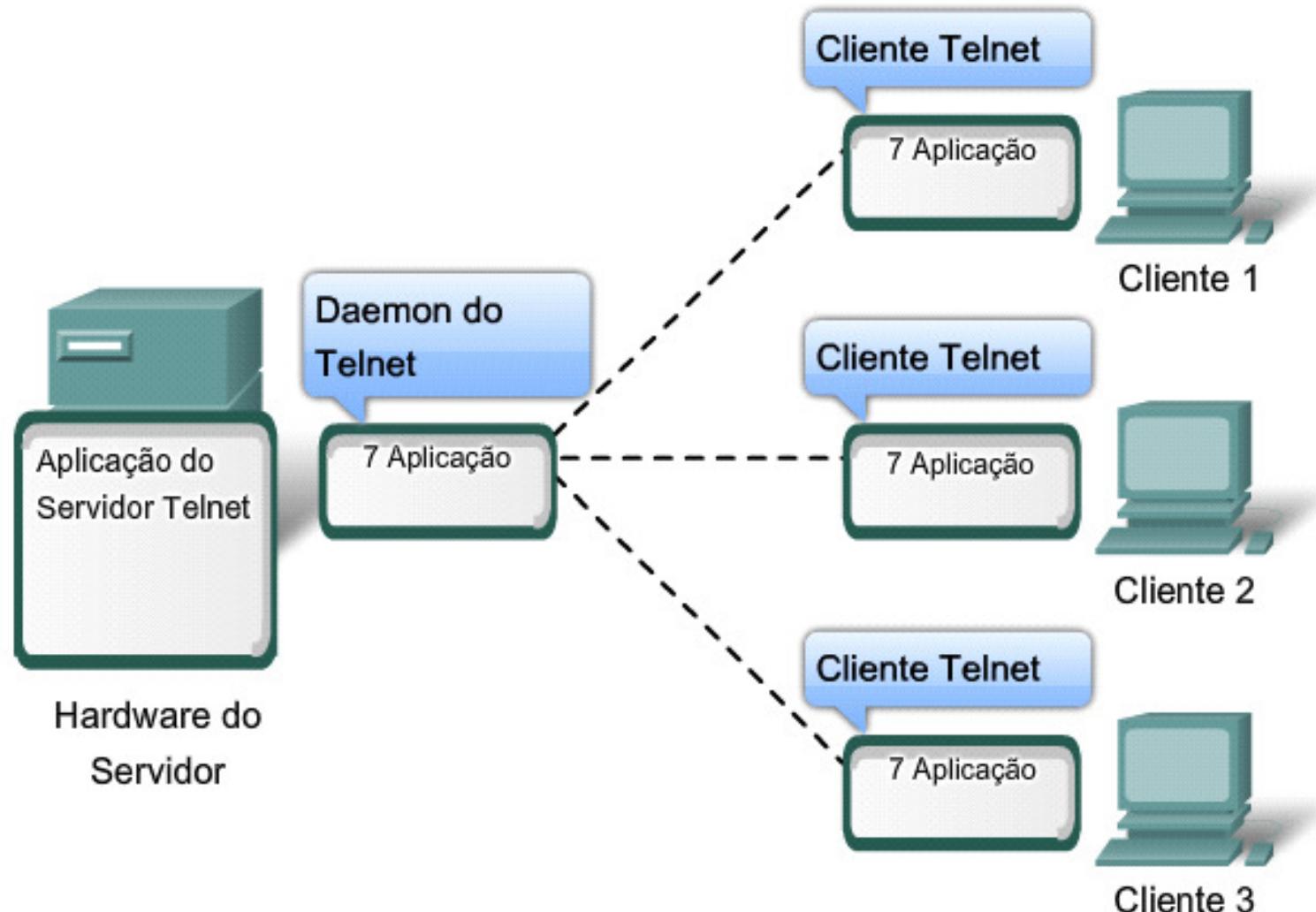
- Um servidor normalmente é um computador que contém **informações a serem compartilhadas** com muitos sistemas cliente;
- Em uma rede cliente/servidor, o servidor executa um serviço, ou processo, às vezes chamado de **daemon** de servidor;
- Daemons são descritos como “**ouvintes**” de uma solicitação de um cliente, porque são programados para responder sempre que o servidor recebe uma solicitação para o serviço fornecido pelo daemon.

A Arquitetura Cliente/Servidor



A Arquitetura Cliente/Servidor

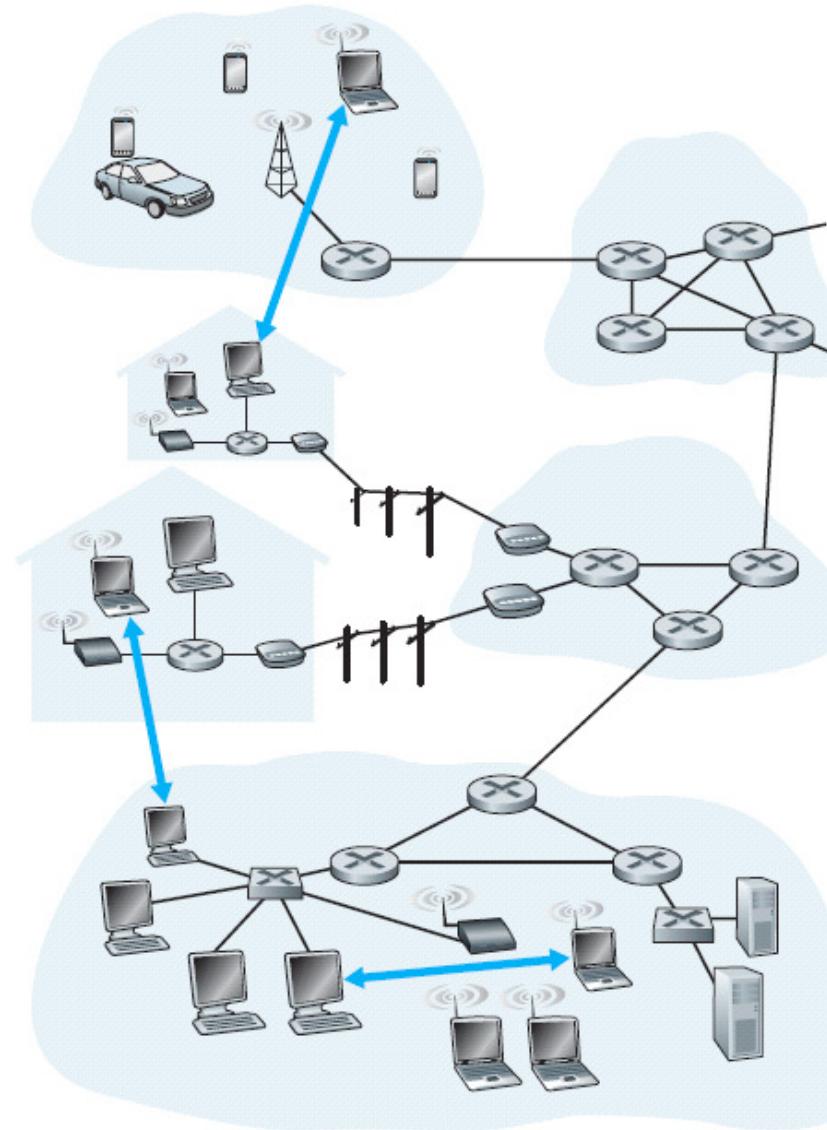
Os processos do servidor podem suportar múltiplos clientes.



A Arquitetura Cliente/Servidor

- Muitas vezes, um único host servidor é incapaz de atender a todas as requisições de seus clientes;
- Por essa razão, um grande conjunto de hosts (**data center**) é usado para criar um servidor virtual poderoso;
- Os serviços de aplicação baseados na arquitetura cliente-servidor são geralmente de **infraestrutura intensiva**;
- Exemplos: Mecanismos de busca (Google), comércio via Internet (Amazon, Mercado Livre), e-mail baseado na Web (Gmail, Hotmail), rede social (Facebook, Instagram), compartilhamento de vídeo (YouTube, Netflix), etc;

A Arquitetura P2P



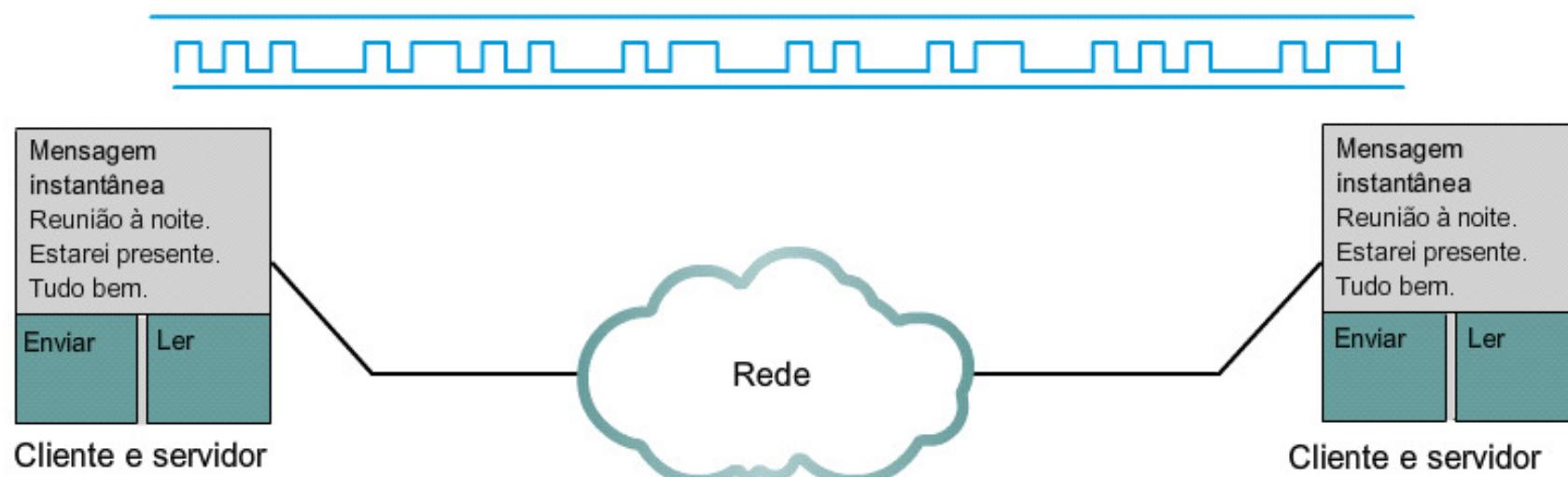
A Arquitetura P2P

- A aplicação utiliza a **comunicação direta** entre pares de hosts conectados alternadamente, denominados **pares (peers)**;
- Os pares não são de propriedade dos provedores de serviços, mas são controlados por usuários finais;
- Os pares se comunicam sem passar por nenhum servidor dedicado;
- **Autoescalabilidade**;
- **Boa relação custo-benefício**;
- Exemplos: Distribuição de arquivos (BitTorrent), compartilhamento de arquivo (eMule, Gnutella, Kazaa), telefonia por Internet (Skype), televisão por Internet IPTV (PPLive, Kodi), etc;

Aplicações P2P

Aplicações ponto-a-ponto

Cliente e servidor na mesma comunicação



Ambos os clientes:

- Iniciam uma mensagem
- Recebem uma mensagem

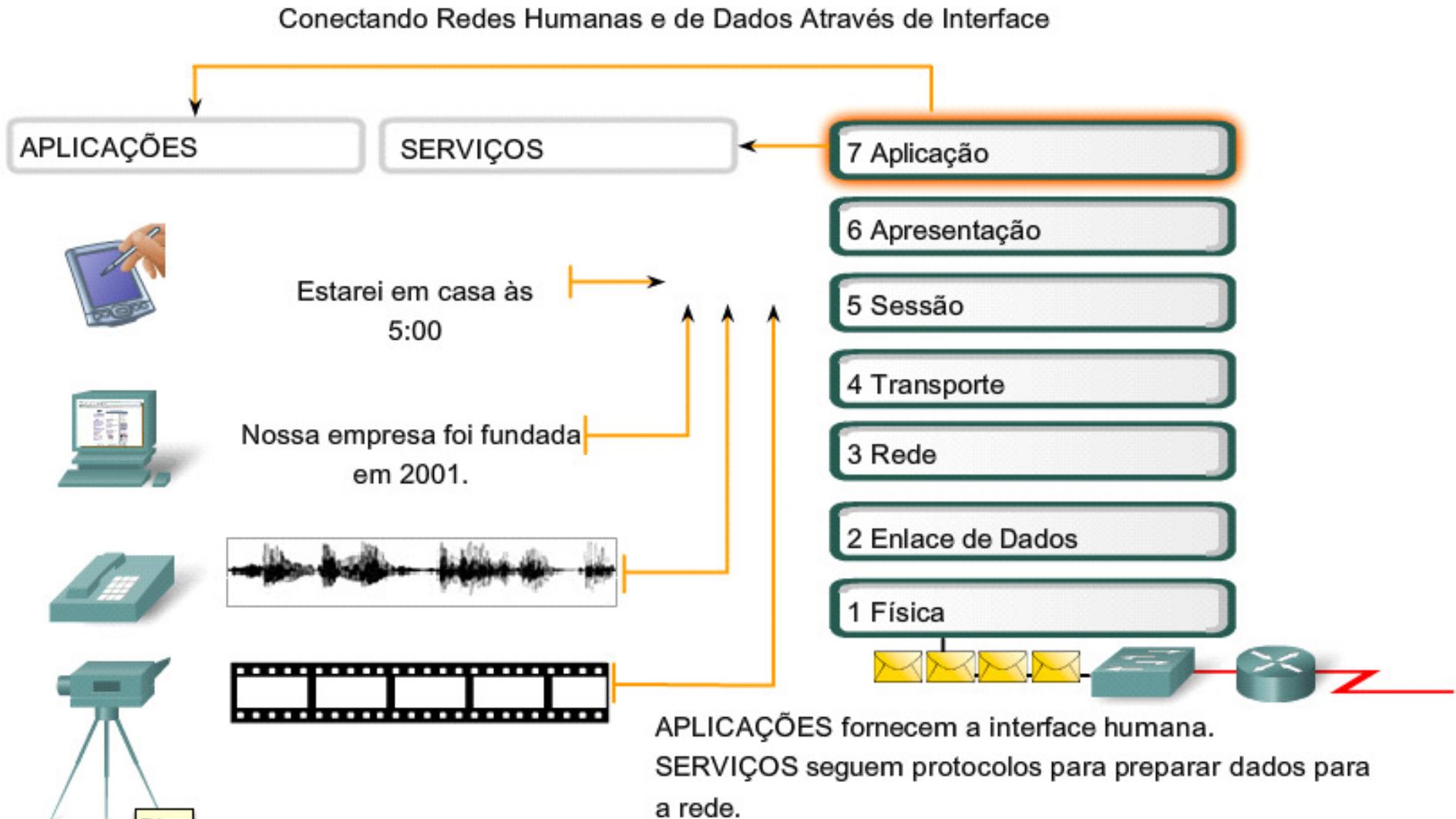
Ambos os clientes
simultaneamente:

- Enviam
- Recebem

Elementos da Camada de Aplicação

- Elementos da camada de aplicação:
 - **Aplicação (Processos)**: oferecem uma maneira de criar mensagens;
 - **Serviços (Sockets)**: estabelecem uma interface com a rede;
 - **Protocolos**: Fornecem as regras e formatos que regem como os dados são tratados.
- Todos os três componentes podem ser utilizados por um único programa executável e até mesmo usar o mesmo nome;
- Por exemplo, ao discutir “Telnet”, podemos nos referir à aplicação, ao serviço ou ao protocolo.

Aplicações de Usuário, Serviços e Protocolos da Camada de Aplicação



Comunicação entre Processos

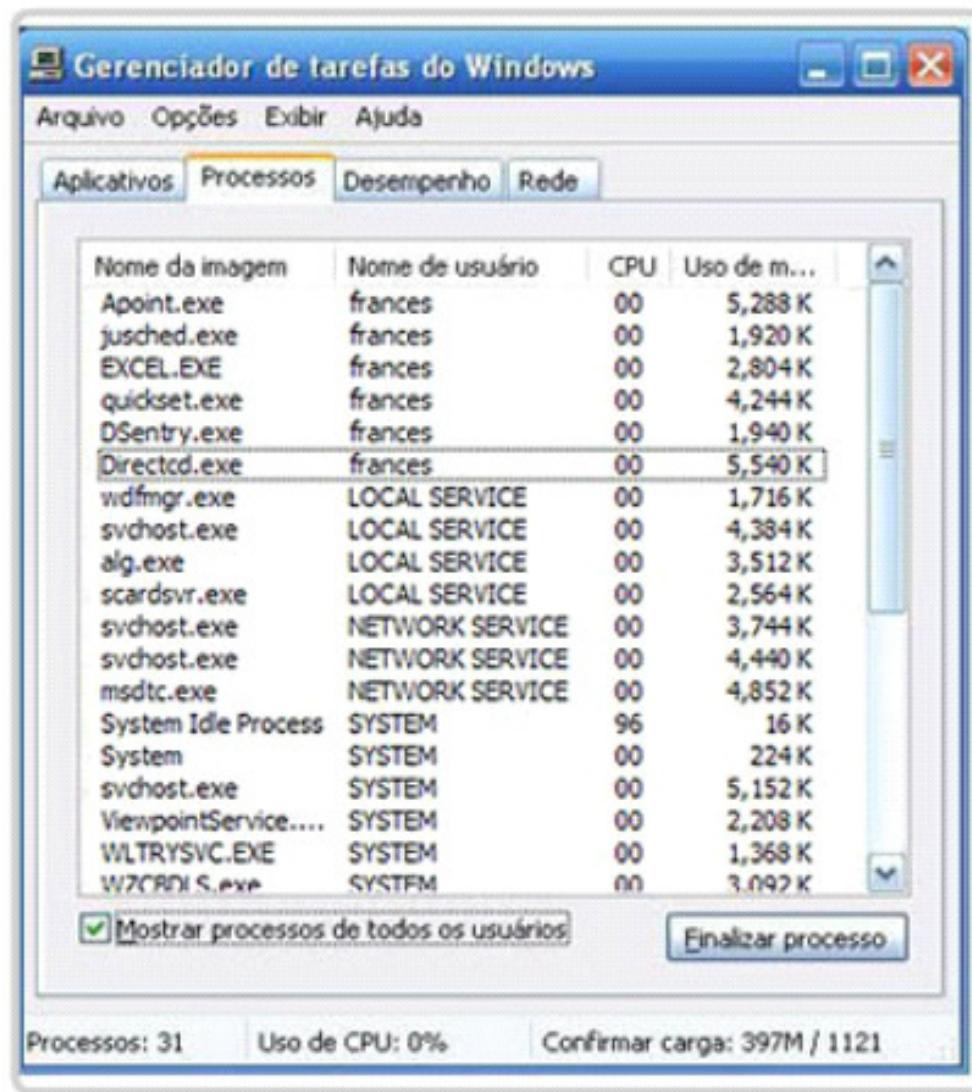
- No jargão de sistemas operacionais, são os **processos que se comunicam**, e não programas;
- Processos podem ser pensados como **programas que rodam em hosts diferentes** (com potencialmente sistemas operacionais diferentes);
- Processos em dois hosts diferentes se comunicam entre si através da **troca de mensagens** através da rede de computadores.

Processos Clientes e Processos Servidores

O processo que inicia a comunicação (inicialmente contacta o outro processo no início da sessão) é chamado **cliente**. O processo que espera para ser contactado para iniciar a sessão é o **servidor**.

Comunicação entre Processos

Processos de Software



Processos são programas individuais de software executando ao mesmo tempo.

Processos podem ser

1 Aplicações

2 Serviços

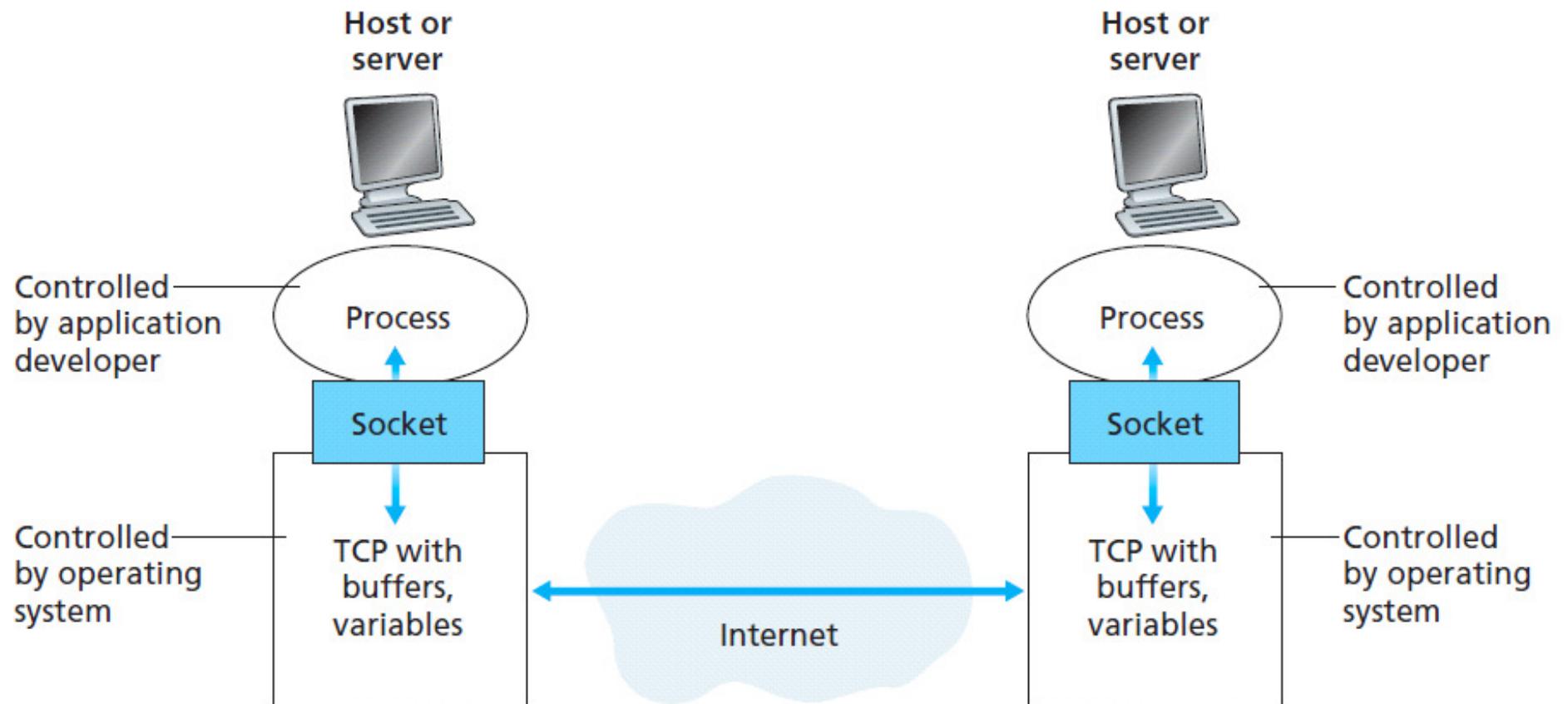
3 Operações do sistema

4 Um programa pode estar sendo executado muitas vezes, cada uma no seu próprio processo.

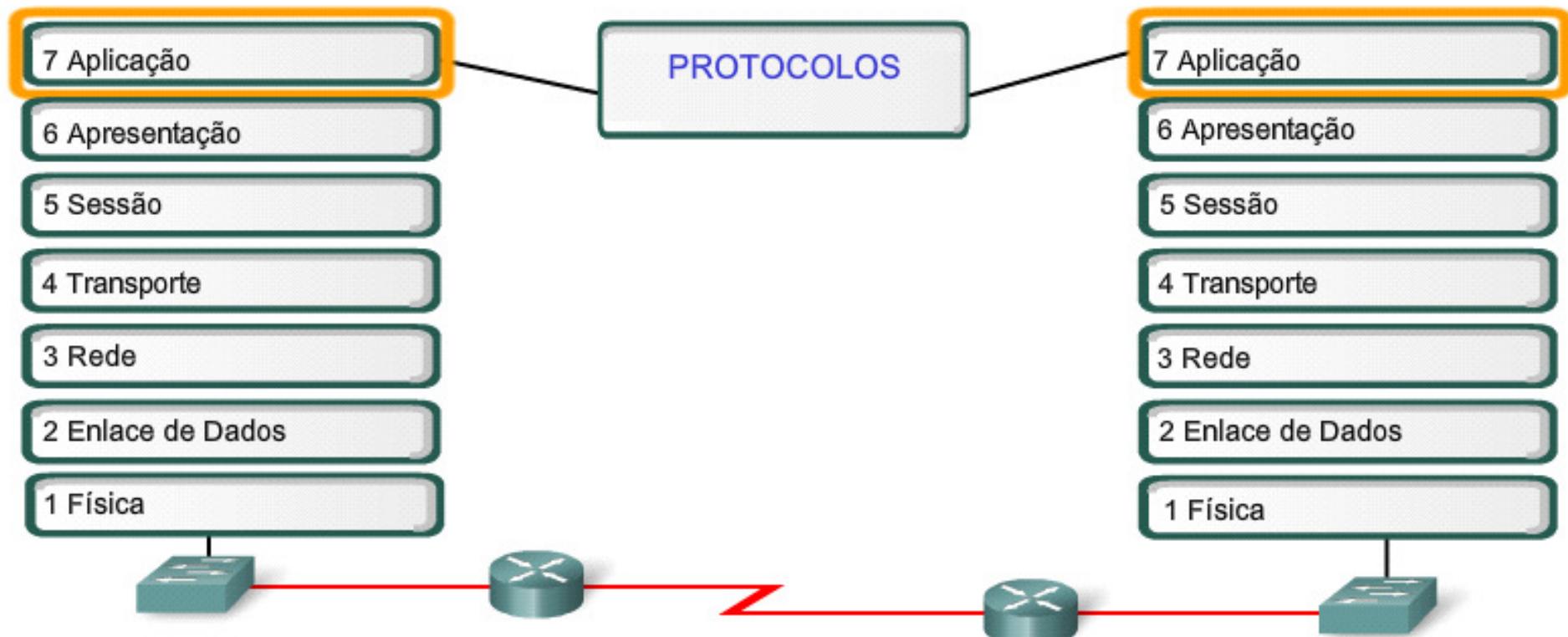
A Interface entre os Processos e a Rede de Computadores

- Qualquer mensagem enviada de um processo para outro deve passar pela rede subjacente;
- Um processo envia mensagens para a rede, e recebe mensagens da rede, através de uma **interface de software** chamada **socket**;
- Analogia: **Processo <-> casa / socket <-> porta**;
- Um socket é a interface entre a camada de aplicação e a camada de transporte em um host;
- Também nos referimos ao socket como uma **API (Application Programming Interface)** entre a aplicação e a rede.

A Interface entre os Processos e a Rede de Computadores



Funções de Protocolo da Camada de Aplicação



Os protocolos da camada de Aplicações fornecem as regras para a comunicação entre os Aplicações.

Funções de Protocolo da Camada de Aplicação

- Os protocolos da camada de aplicação definem:
 - Os tipos de mensagens trocadas, por exemplo, as mensagens de requisição e resposta;
 - A sintaxe dos vários tipos de mensagens, tais como os campos na mensagem e como os campos são delineados;
 - A semântica dos campos, isto é, o significado da informação nos campos;
 - Regras para determinar quando e como um processo envia mensagens e responde às mensagens.

Serviços de Transporte Disponíveis para as Aplicações

- A aplicação do lado remetente envia mensagens através do socket;
- Do outro lado do socket, o protocolo de camada de transporte tem a responsabilidade de levar as mensagens pela rede até a “porta” do socket destinatário;
- Possíveis serviços que um protocolo da camada de transporte pode oferecer às aplicações:
 - Transferência confiável de dados;
 - Vazão garantida;
 - Garantias de temporização (atraso);
 - Segurança.

Serviços de Transporte Disponíveis para as Aplicações

- **Transferência confiável de dados**

- **Os pacotes podem se perder** dentro de uma rede de computadores;
- Para muitas aplicações a perda de dados pode ter **consequências devastadoras**;
- Se um protocolo fornecer um **serviço de recebimento de dados garantidos**, ele fornecerá uma transferência confiável de dados;
- O processo remetente pode então passar seus dados para um socket e saber com absoluta confiança que os **dados chegarão sem erro** ao processo destinatário.

Serviços de Transporte Disponíveis para as Aplicações

● Vazão garantida

- A vazão disponível na rede pode **oscilar com o tempo**;
- Um protocolo da camada de transporte pode oferecer uma **vazão disponível garantida** a uma taxa específica;
- Aplicações que possuam necessidade de vazão são conhecidas como **aplicações sensíveis à largura de banda** (ex. aplicações multimídia);
- As **aplicações elásticas** podem fazer uso de qualquer quantidade mínima ou máxima de largura de banda que por acaso esteja disponível (ex. correio eletrônico, transferência de arquivos e Web).

Serviços de Transporte Disponíveis para as Aplicações

- **Garantia de temporização**

- Um protocolo de camada de transporte pode garantir, por exemplo, que cada bit que o remetente insira no socket chegue ao socket destinatário em menos de X milissegundos depois;
- Esse serviço seria atrativo para **aplicações interativas em tempo real**, que exigem restrições de temporização no envio de dados para garantir eficácia;

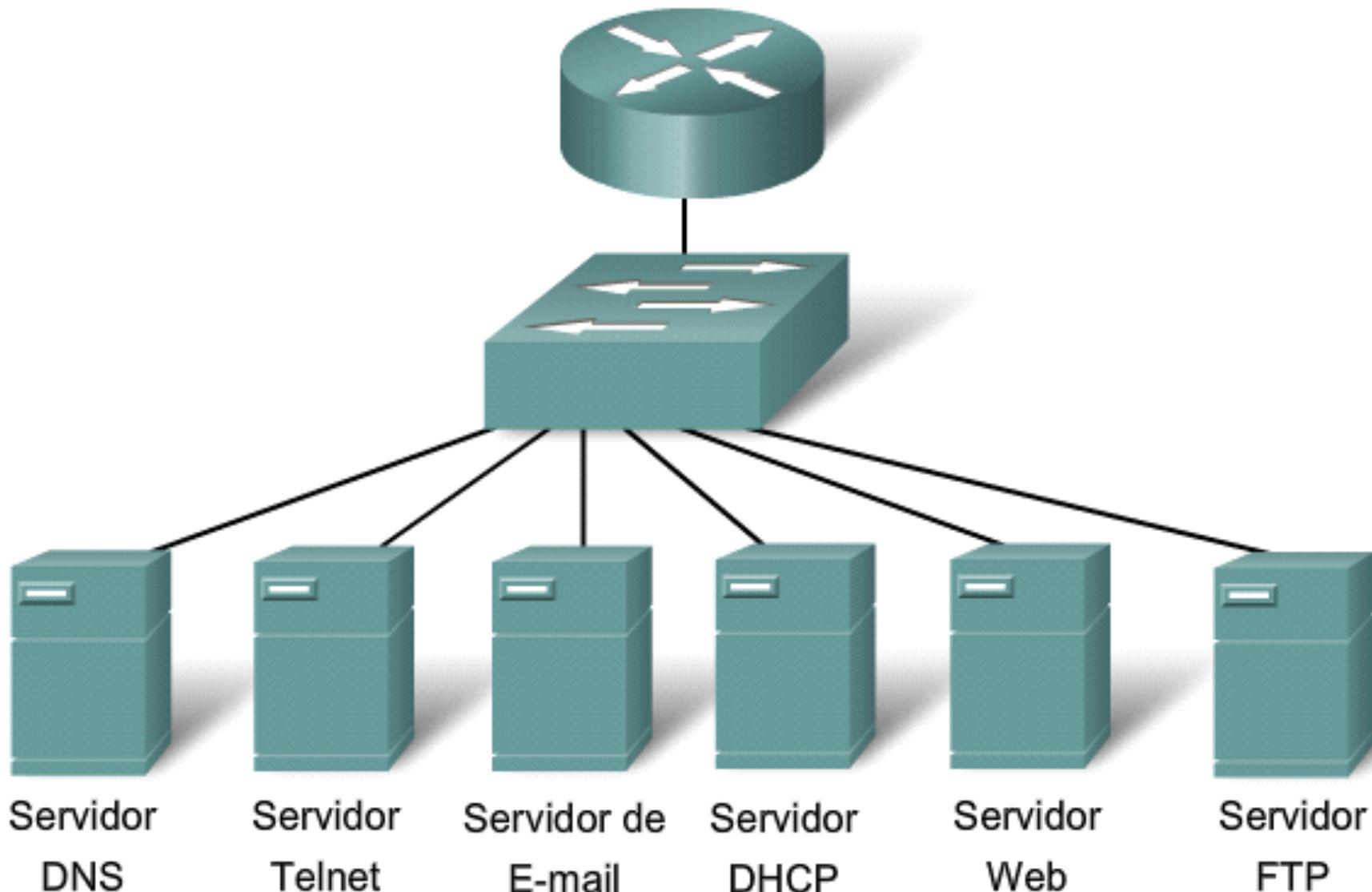
- **Segurança**

- Um protocolo da camada de transporte pode oferecer **sigilo dos dados** através de codificação/decodificação;
- Outros serviços de segurança que podem ser ofertados: **integridade dos dados** e **autenticação do ponto terminal**.

Requisitos de Aplicações

Application	Data Loss	Throughput	Time-Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/ Video conferencing	Loss-tolerant	Audio: few kbps–1 Mbps Video: 10 kbps–5 Mbps	Yes: 100s of msec
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of msec
Instant messaging	No loss	Elastic	Yes and no

Tipos de Aplicações



Tipos de Aplicações

- **Sistema de Resolução de Domínios - DNS**
 - Serviço que fornece o endereço IP de um site ou nome de domínio para que um host possa se conectar a ele.
- **Telnet/SSH**
 - Serviços que permitem que os administradores se conectem a um host em uma localização remota e controlem o host como se eles estivessem conectados localmente.
 - Telnet não é seguro. O SSH sim.

Tipos de Aplicações

- **E-mail**
 - Usa o protocolos SMTP, POP3 ou IMAP
 - Usado para enviar mensagens de e-mail de clientes para servidores através da Internet;
 - Destinatários são especificados usando o formato usuário@xyz.
- **Protocolo de Configuração Dinâmica de Host - DHCP**
 - Serviço que designa o gateway padrão da máscara de sub-rede do endereço IP e outras informações aos clientes.

Tipos de Aplicações

- **Web**
 - Protocolo HTTP;
 - Usado para transferir informações entre clientes e servidores Web;
- **FTP**
 - Serviço que permite o download e upload de arquivos entre um cliente e servidor.

Tipos de Aplicações

- A camada de Transporte utiliza um esquema de endereçamento chamado número de porta;
- Os números de porta identificam aplicações e serviços da camada de Aplicação que são a origem e o destino dos dados;
- Portas que algumas aplicações populares usam:
 - **Domain Name System (DNS)** - Porta TCP/UDP 53
 - **Hypertext Transfer Protocol (HTTP)** - Porta TCP 80
 - **Simple Mail Transfer Protocol (SMTP)** - Porta TCP 25
 - **Protocolo POP** - Porta UDP 110
 - **SSH** - Porta TCP 22
 - **Telnet** - Porta TCP 23
 - **Dynamic Host Configuration Protocol (DHCP)** - Porta UDP 67
 - **File Transfer Protocol (FTP)** - Portas TCP 20 e 21

O Serviço WWW

- Quando um **endereço Web** é digitado em um navegador Web, este estabelece uma conexão com o serviço Web executado no servidor utilizando o **protocolo HTTP**;
- A URL `http://www.cisco.com/index.html` é um exemplo de URL que se refere a um recurso específico - uma página Web nomeada `index.html` em um servidor identificado como `cisco.com`;
- Uma **página Web** (também denominada documento) é constituída de objetos;
- Um **objeto** é apenas um arquivo (ex. arquivo HTML, figura JPEG, applet Java, clipe de vídeo, etc) que se pode acessar com um único URL.

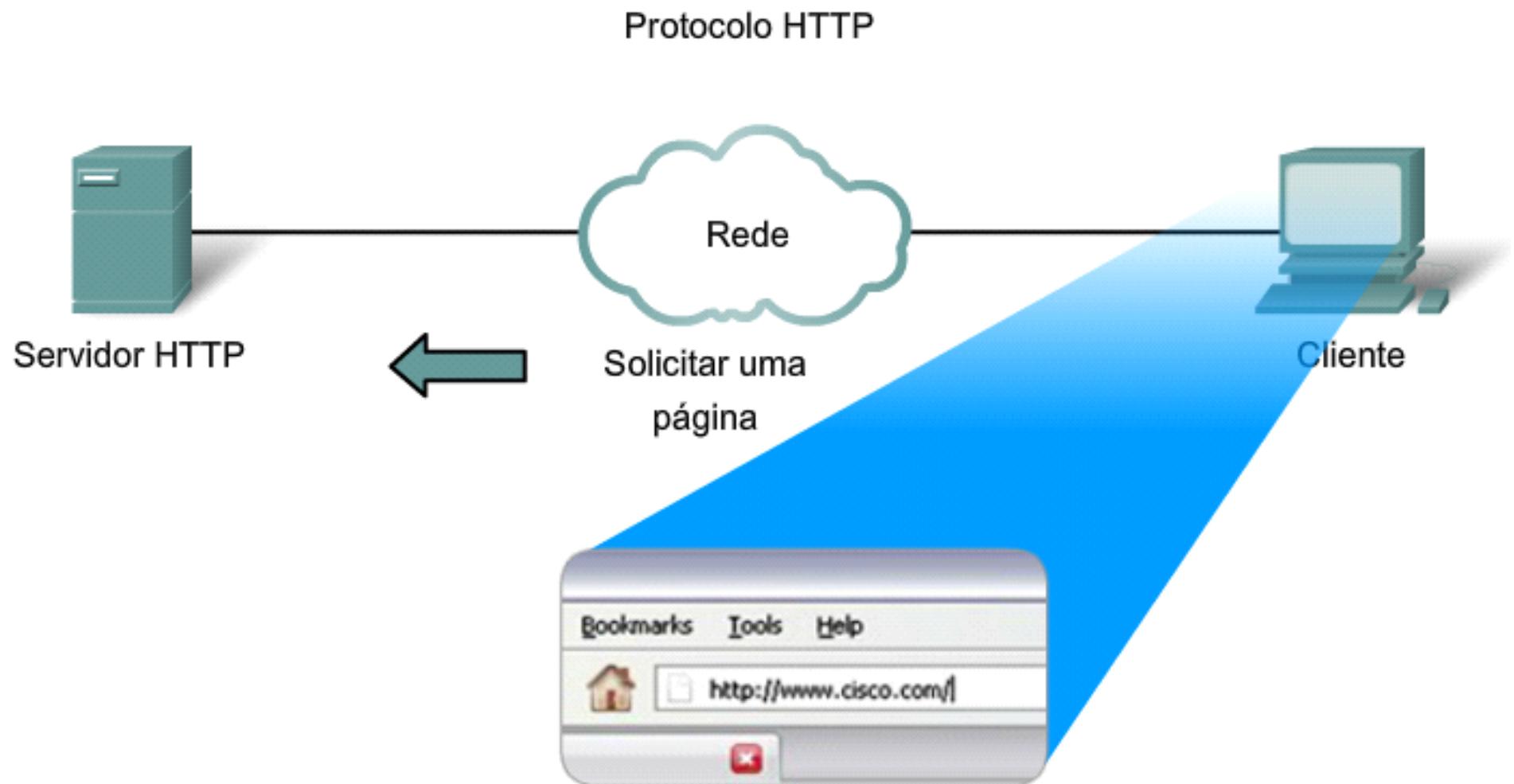
O Serviço WWW

- Os clientes Web fazem conexões ao servidor e solicitam os recursos desejados. O servidor responde com os recursos e, no recebimento, o navegador interpreta os dados e os apresenta ao usuário;
- Os navegadores podem interpretar e apresentar muitos tipos de dados, como texto simples ou **Hypertext Markup Language** (HTML, linguagem na qual as páginas Web são construídas);
- Outros tipos de dados podem exigir outro serviço ou programa, normalmente mencionado como **plug-ins** ou **add-ons**;

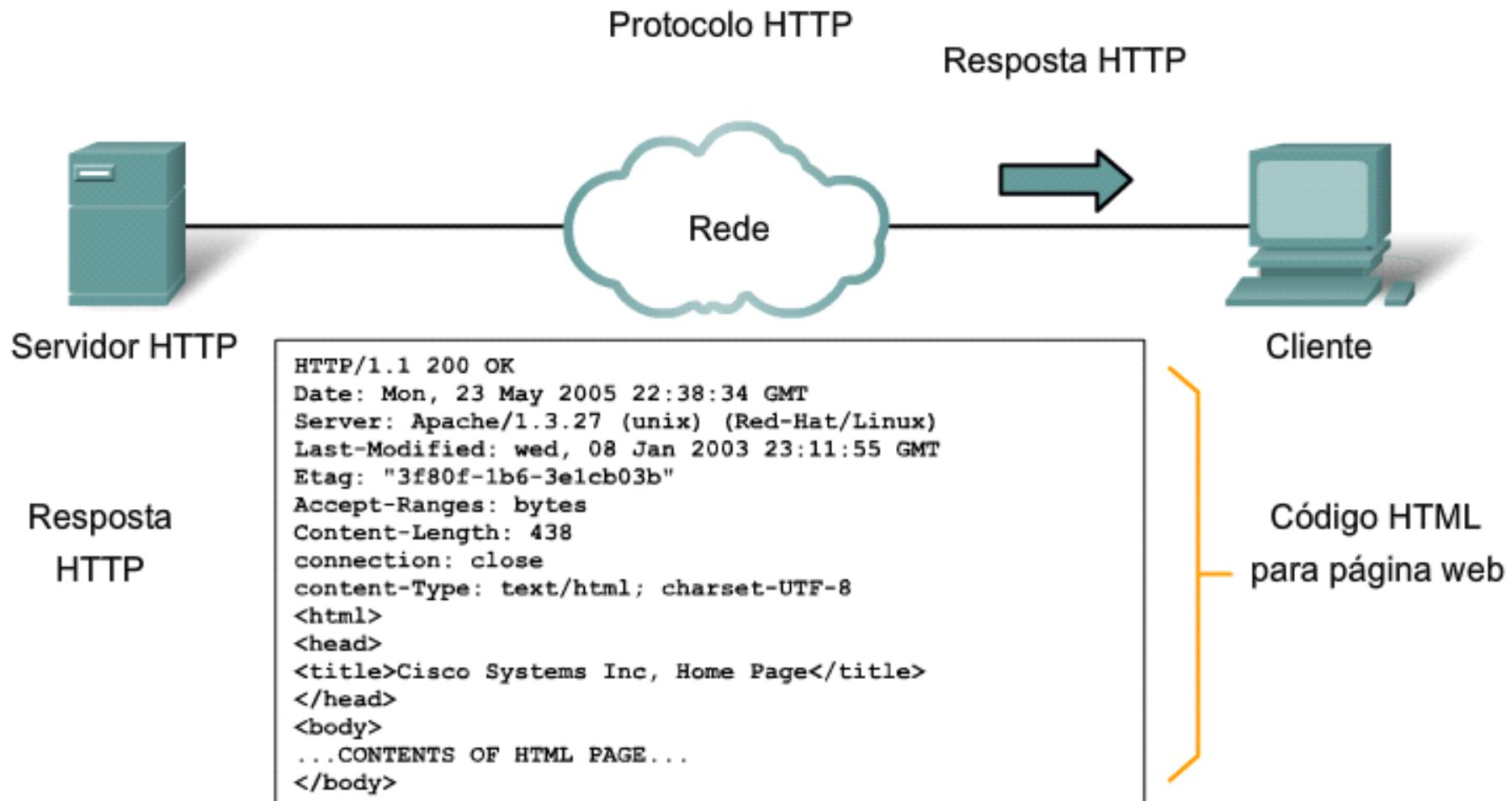
Exemplo: <http://www.cisco.com/web-server.htm>

- ① O navegador interpreta as três partes da URL:
 - ① **http**: protocolo ou esquema
 - ② **www.cisco.com**: nome do servidor
 - ③ **web-server.htm**: nome do arquivo específico solicitado
- ② O navegador consulta um **servidor de nomes** para converter **www.cisco.com** em um endereço numérico, que utiliza para se conectar ao servidor;
- ③ Utilizando os requerimentos do protocolo HTTP, o navegador envia uma **solicitação GET** ao servidor e pede o arquivo **web-server.htm**;
- ④ O servidor, por sua vez, envia o **código HTML** para esta página Web ao navegador;
- ⑤ O navegador decifra o código HTML e formata a página para a janela do navegador.

Solicitação de uma página



Resposta HTTP



Em resposta à solicitação, o servidor HTTP retorna um código para uma página Web.

Exibição da página



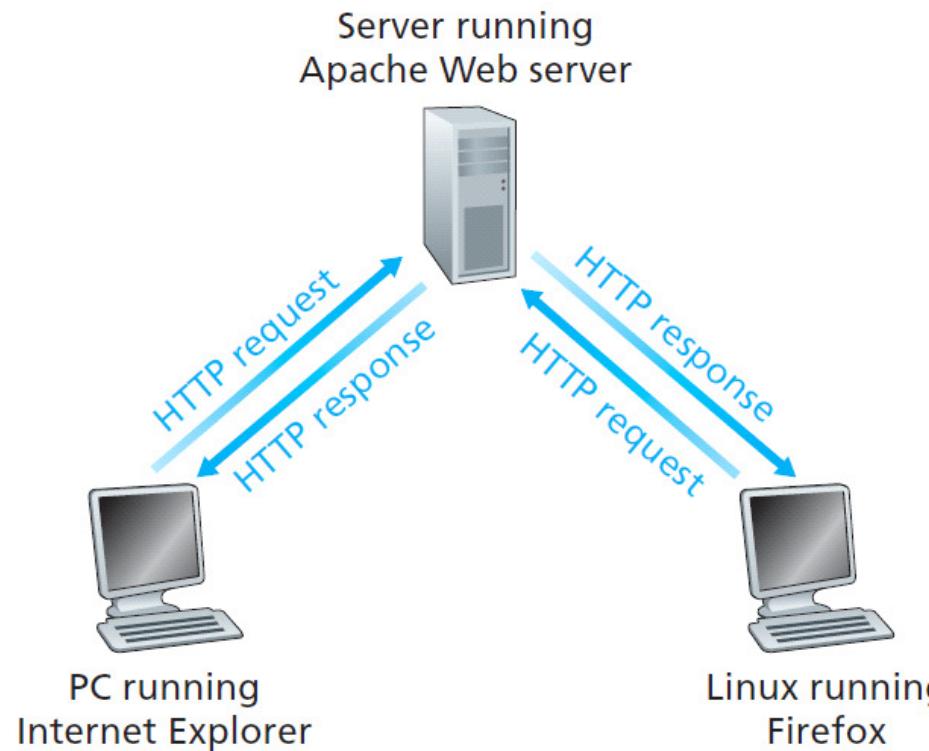
O navegador interpreta o código HTML e exibe uma página Web.

O Protocolo HTTP

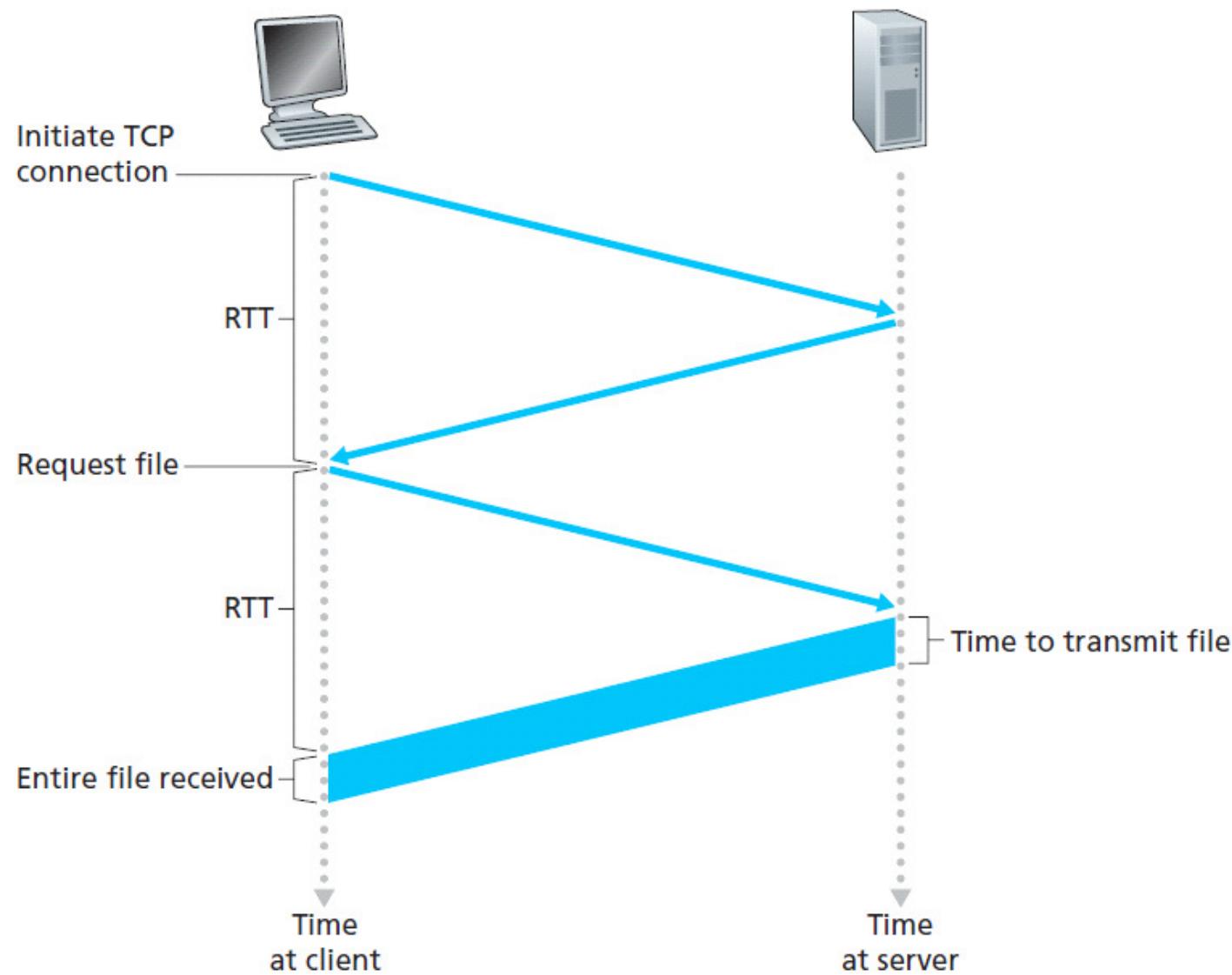
- O HTTP (HyperText Transfer Protocol) é utilizado na World Wide Web para **transferência de dados** e é um dos protocolos de aplicação mais usados;
- O HTTP usa o **TCP** como seu protocolo de transporte subjacente (em vez de rodar em UDP);
- O cliente HTTP primeiramente inicia uma conexão TCP com o servidor; uma vez estabelecida a conexão, os processos do browser e do servidor acessam o TCP por meio de suas interfaces sockets;
- O HTTP não precisa se preocupar com dados perdidos, ou como a perda de dados é recuperada, ou como os dados são reordenados dentro da rede; essa é tarefa do TCP e dos protocolos das camadas inferiores.

O Protocolo HTTP

- O HTTP especifica um **protocolo de requisição/resposta**;
- O protocolo HTTP define os **tipos de mensagem** que o cliente utiliza para solicitar a página Web e também os tipos de mensagem que o servidor usa para responder;



Tempo necessário para solicitar e receber um arquivo HTML



Conexões Persistentes e Não Persistentes

Cada par de requisição/resposta deve ser enviado por uma conexão TCP **distinta** ou todas as requisições e suas respostas devem ser enviadas por uma **mesma** conexão TCP?

- Exemplo: Uma página Web consiste em 01 arquivo-base HTML e em 10 imagens JPEG; todos os 11 objetos residem no mesmo servidor.
- **Conexões não persistentes**
 - Cada conexão TCP transporta exatamente uma mensagem de requisição e uma mensagem de resposta;
 - Cada conexão TCP é encerrada após o servidor enviar o objeto - a conexão não persiste para outros objetos;
 - São geradas 11 conexões TCP.

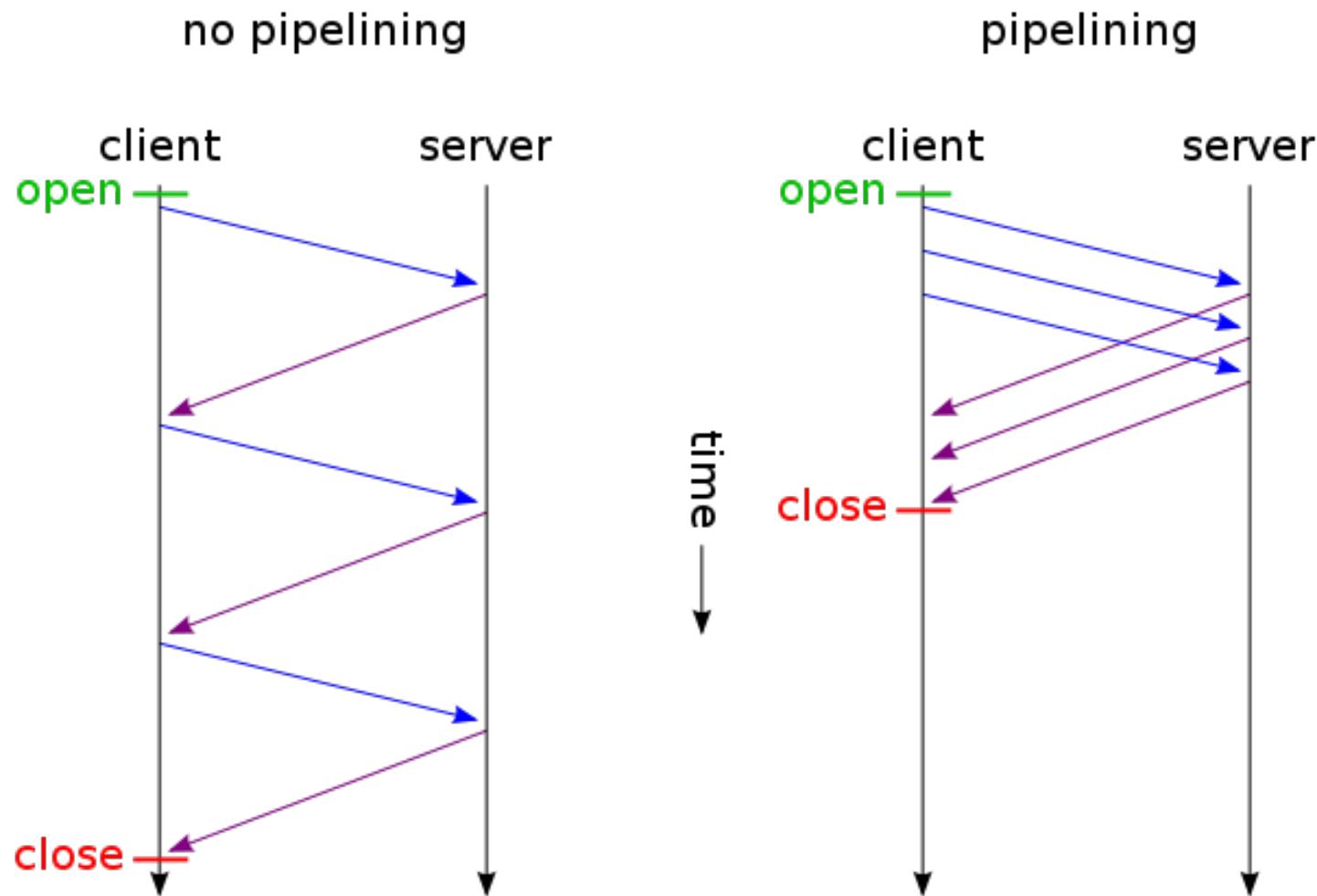
Conexões Persistentes e Não Persistentes

- **Conexões persistentes**

- O servidor deixa a conexão TCP aberta após enviar resposta;
- Requisições e respostas subsequentes entre os mesmos cliente e servidor podem ser enviadas por meio da mesma conexão.
- O modo default to HTTP usa conexões persistentes com pipelining (requisições por objetos podem ser feitas consecutivamente sem ter de esperar por respostas a requisições pendentes).

Conexões Persistentes e Não Persistentes

Pipelining



Mensagens de Requisição HTTP

- **GET**

- É usado quando o browser requisita um objeto e este é identificado no campo do URL.

- **POST**

- É usado quando o usuário preenche um formulário;
- O usuário continua solicitando uma página Web ao servidor, mas o conteúdo específico dela depende do que o usuário escreveu nos campos do formulário.

Mensagens de Requisição HTTP

• **HEAD**

- Semelhante ao GET;
- Quando um servidor recebe uma requisição com este método, responde com uma mensagem HTTP, mas deixa de fora o objeto requisitado;
- Usado pelos desenvolvedores de servidores HTTP para depuração.

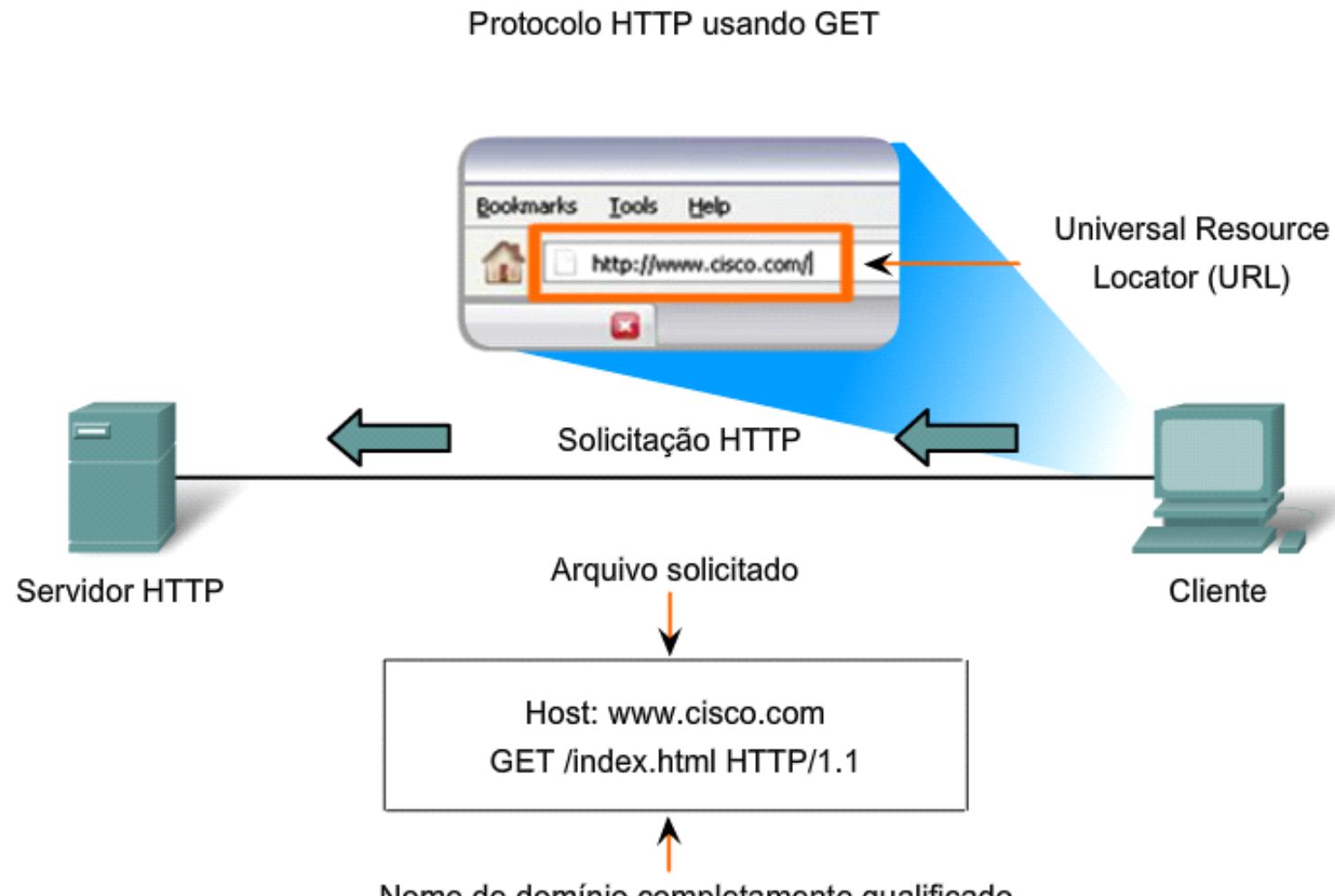
• **PUT**

- Permite que um usuário carregue um objeto para um caminho específico (diretório) em um servidor Web específico.

• **DELETE**

- Permite que um usuário, ou uma aplicação, elimine um objeto em um servidor Web.

Mensagens de Requisição HTTP



A inserção de 'http://www.cisco.com' na barra de endereços de um web browser gera uma Mensagem HTTP 'GET'.

Formato de Mensagens de Requisição HTTP

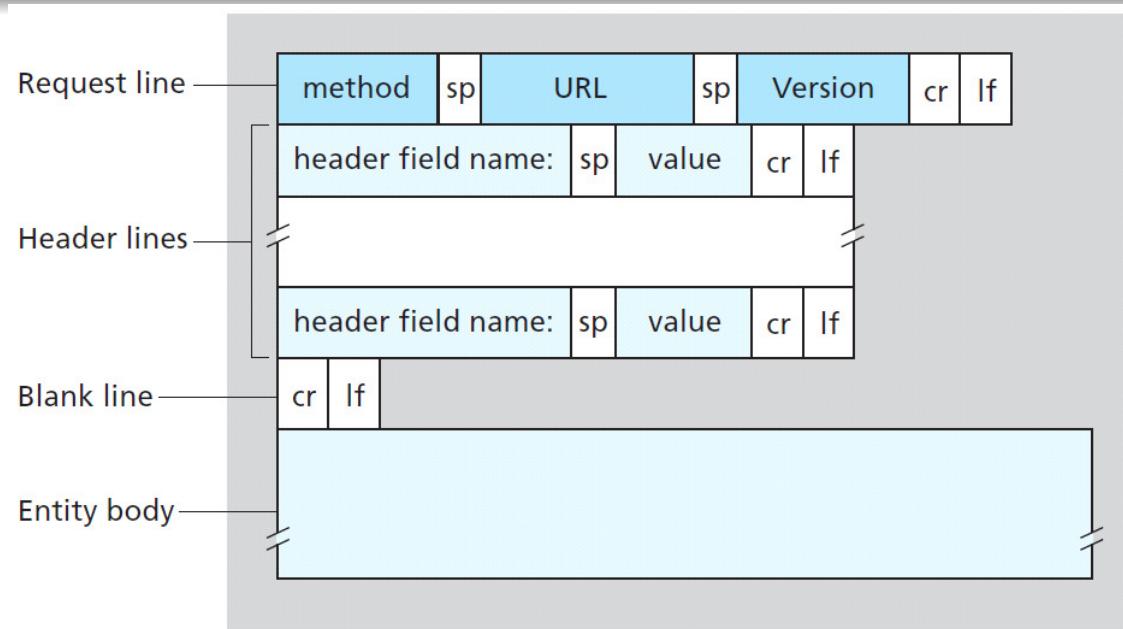
```
GET /somedir/page.html HTTP/1.1
```

```
Host: www.someschool.edu
```

```
Connection: close
```

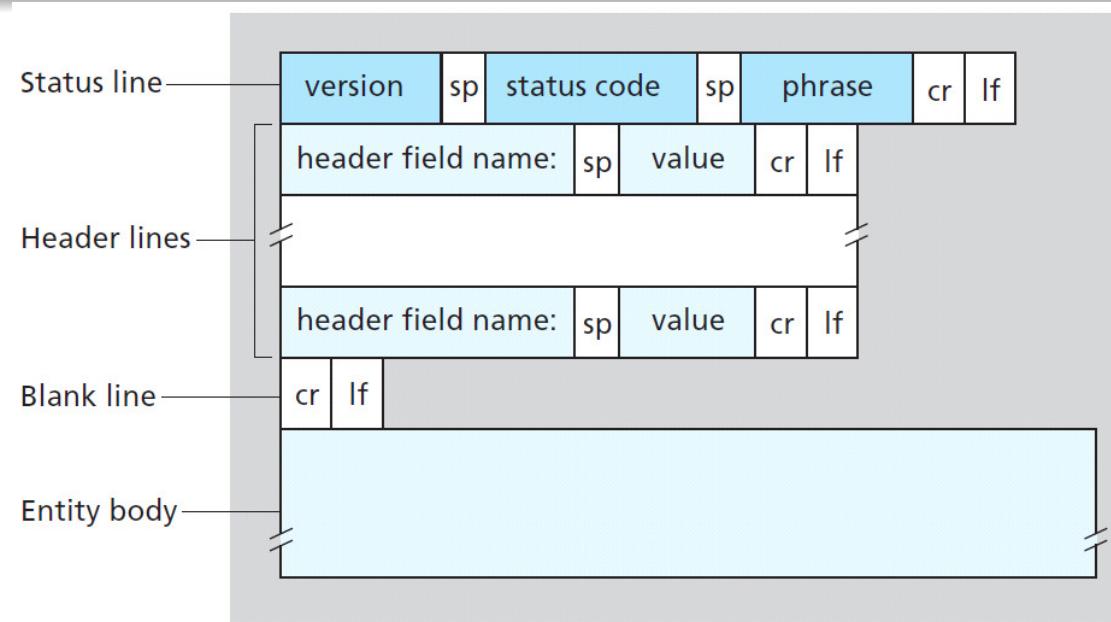
```
User-agent: Mozilla/4.0
```

```
Accept-language: fr
```



Formato de Mensagens de Resposta HTTP

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



Formato de Mensagens de Resposta HTTP

- Códigos de estado e frases associadas comuns:
 - **200 OK**: Requisição bem-sucedida e a informação é entregue com a resposta;
 - **301 Moved Permanently**: objeto requisitado foi removido permanentemente; novo URL é especificado no cabeçalho **Location**: da mensagem de resposta; o software do cliente recuperará automaticamente o novo URL;
 - **400 Bad Request**: código genérico de erro que indica que a requisição não pôde ser entendida pelo servidor;
 - **404 Not Found**: o documento requisitado não existe no servidor;
 - **505 HTTP Version Not Supported**: a versão do protocolo HTTP requisitada não é suportada pelo servidor.

HTTP Seguro

- Embora seja notavelmente flexível, o **HTTP não é um protocolo seguro**;
- As mensagens POST fazem upload de informações ao servidor em texto simples que podem ser **interceptadas e lidas**;
- As respostas do servidor, normalmente páginas HTML, também são **não-criptografadas**;
- o protocolo **HTTP Seguro (HTTPS)** pode utilizar **autenticação e criptografia** para proteger os dados que trafegam entre o cliente e o servidor;
- O HTTPS especifica regras adicionais para a passagem de dados entre a camada de Aplicação e a de Transporte.

Fundamentos

- A maioria das pessoas tem dificuldade em **lembrar o endereço IP** numérico que identificam os dispositivos;
- Os **nomes de domínio** foram criados para converter o endereço numérico em um nome simples e reconhecível;
- Se uma empresa decidir alterar o endereço numérico, isso será **transparente para o usuário**, já que o nome de domínio continuará sendo o mesmo;
- À medida que as redes começaram a crescer e o número de dispositivos aumentou, o sistema manual de mapeamento entre nomes e endereços ficou inviável.

Fundamentos

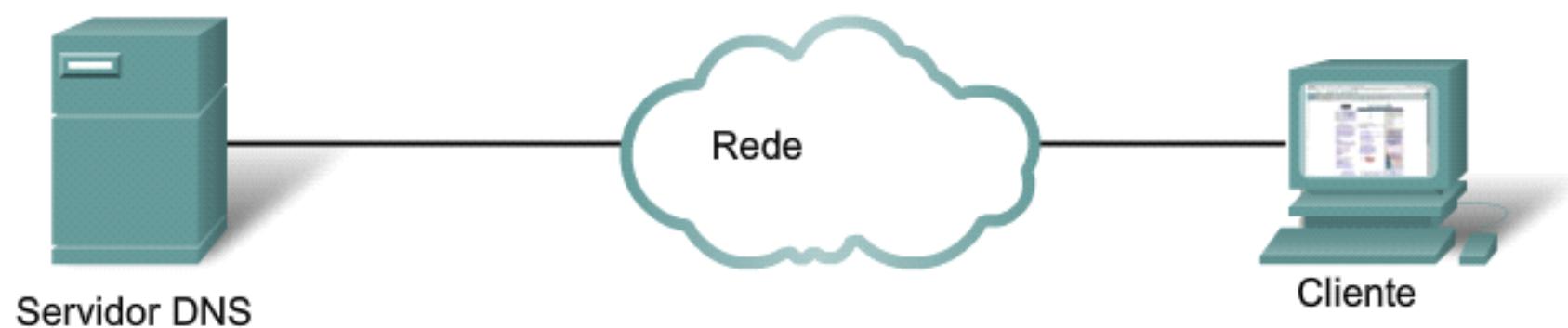
- O DNS utiliza um **conjunto distribuído de servidores** para definir os nomes associados a tais endereços numéricos;
- O protocolo DNS define um **serviço automatizado** que alia os nomes de recursos com o endereço de rede numérico necessário;
- O DNS é um serviço **cliente/servidor**;
- Enquanto outros serviços utilizam um cliente que é uma aplicação (como navegador Web, cliente de e-mail), o cliente DNS é **executado como um serviço**;
- Ao configurar um dispositivo de rede, geralmente fornecemos um ou mais endereços de **Servidor DNS** que o cliente DNS pode utilizar para resolução de nome.

Fundamentos

- Outros serviços importantes providos pelo DNS:
 - **Apelidos de hosts**
 - Um host com nome complicado pode ter um ou mais **apelidos**;
 - Exemplo: relay1.west-coast.enterprise.com (**nome canônico**) → enterprise.com (apelido) ou www.enterprise.com (apelido);
 - **Apelidos de servidores de correio**
 - Endereços de e-mail devem ser fáceis de lembrar;
 - Exemplo: relay1.west-coast.hotmail.com (nome canônico) → hotmail.com (apelido);
 - **Distribuição de carga**
 - Sites movimentados como cnn.com são replicados em varios servidores, cada um com um endereço IP diferente;
 - Um conjunto de endereços IP fica associado a um único nome canônico e contido no banco de dados do DNS;
 - O DNS realiza um **rodízio** da ordem dos endereços dentro de cada resposta.

Resolução de Endereços

Resolução de Endereços (DNS)

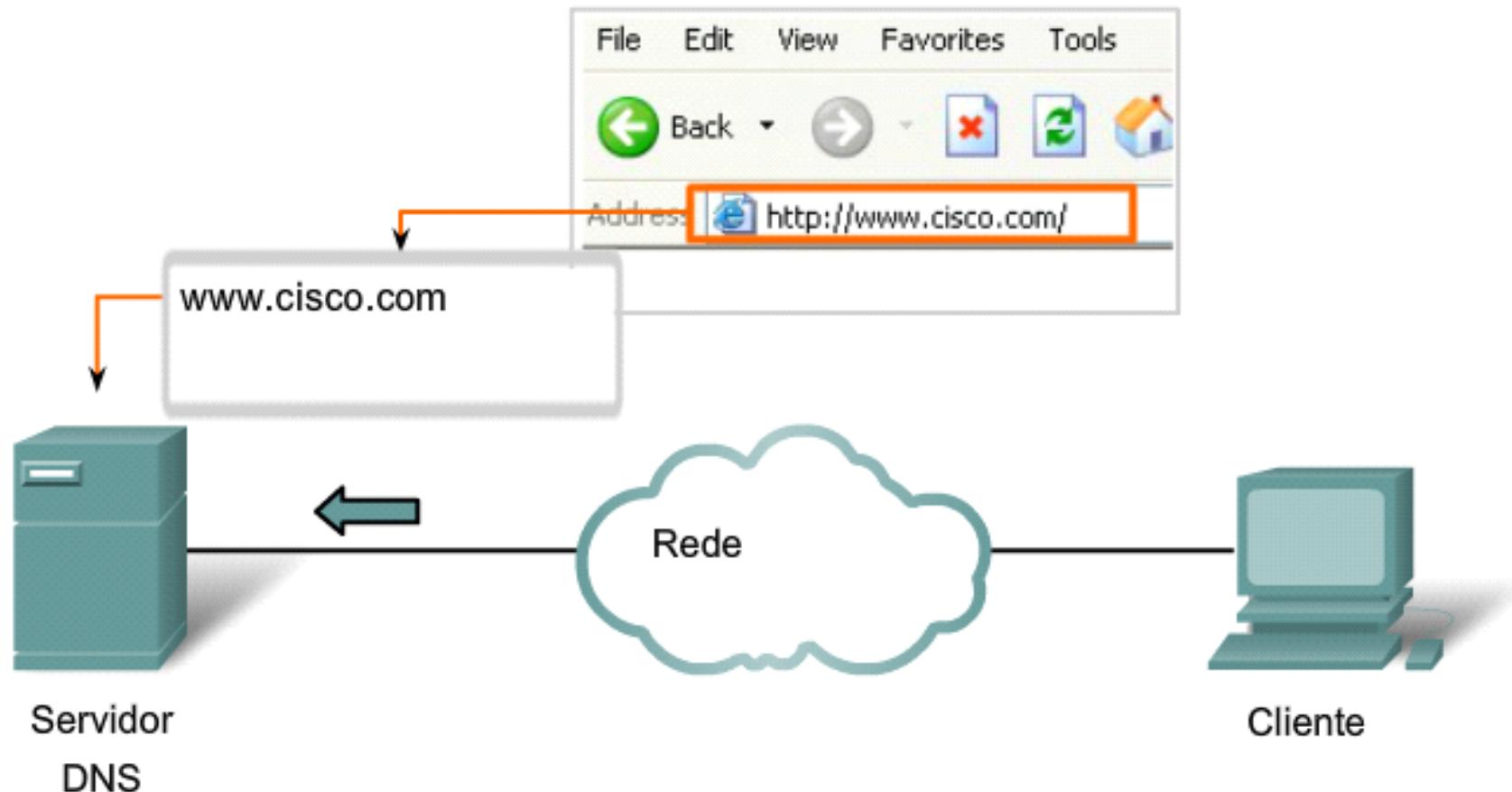


Resolução de Endereços



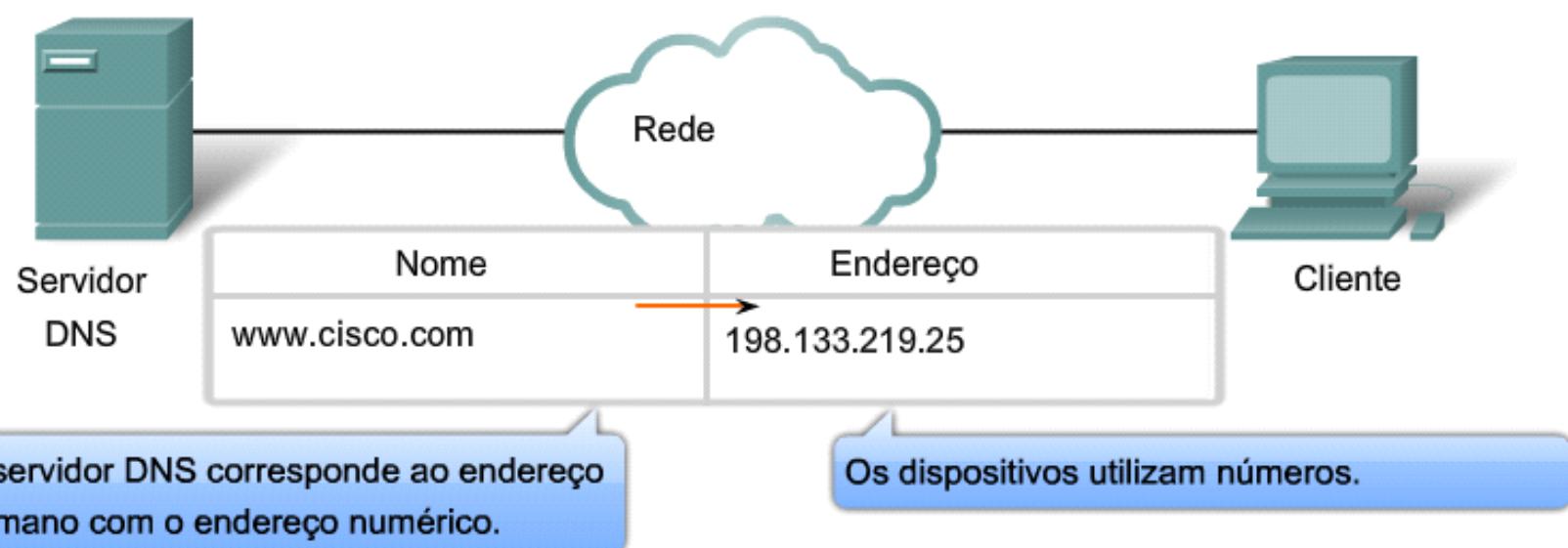
Resolução de Endereços

Resolução de Endereços (DNS)



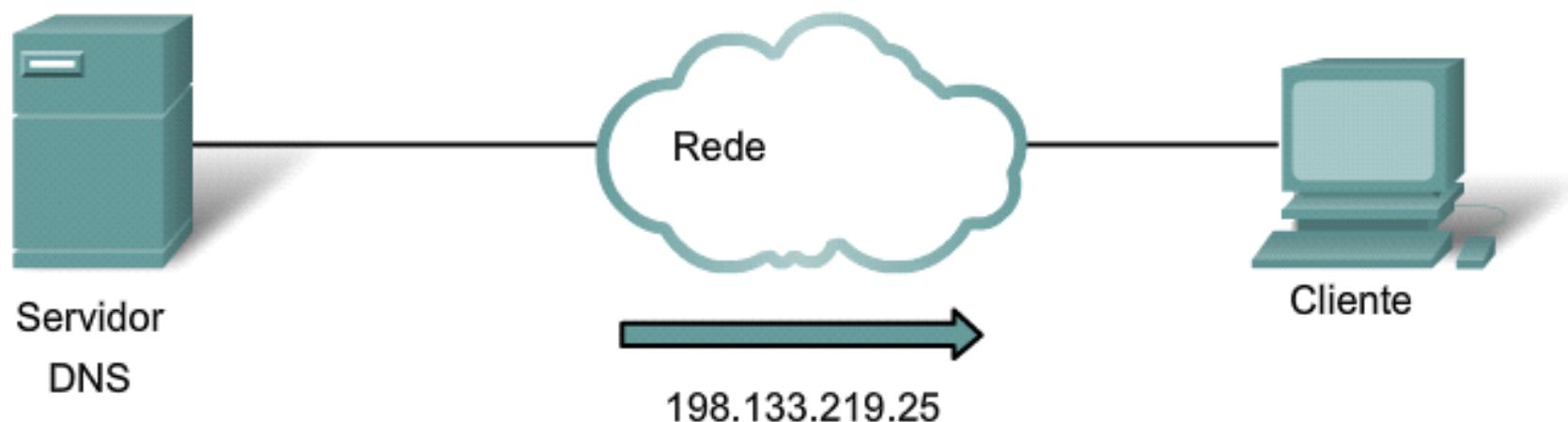
Resolução de Endereços

Resolução de Endereços (DNS)



Resolução de Endereços

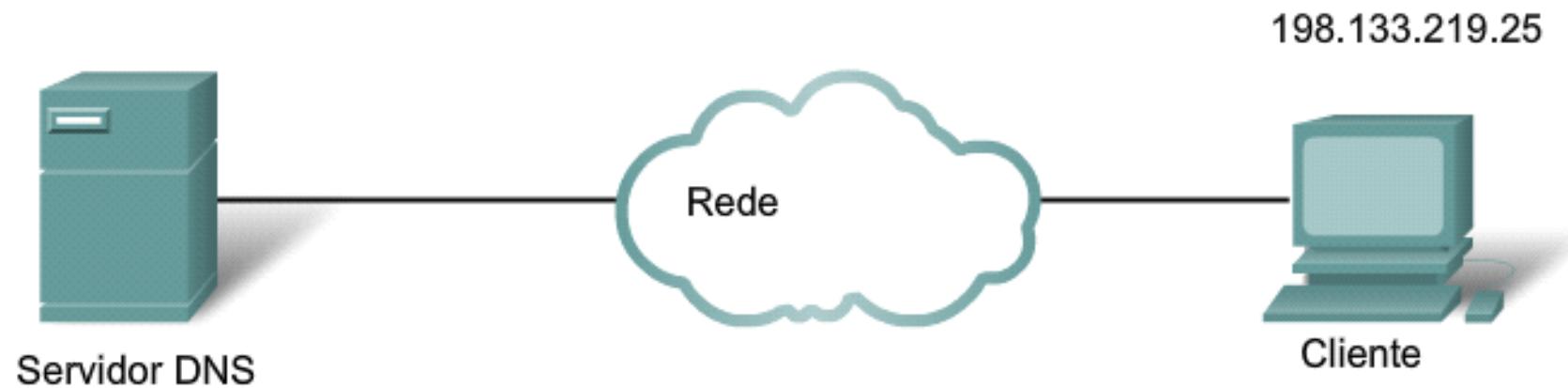
Resolução de Endereços (DNS)



O número volta para o cliente para uso ao fazer
solicitações do servidor.

Resolução de Endereços

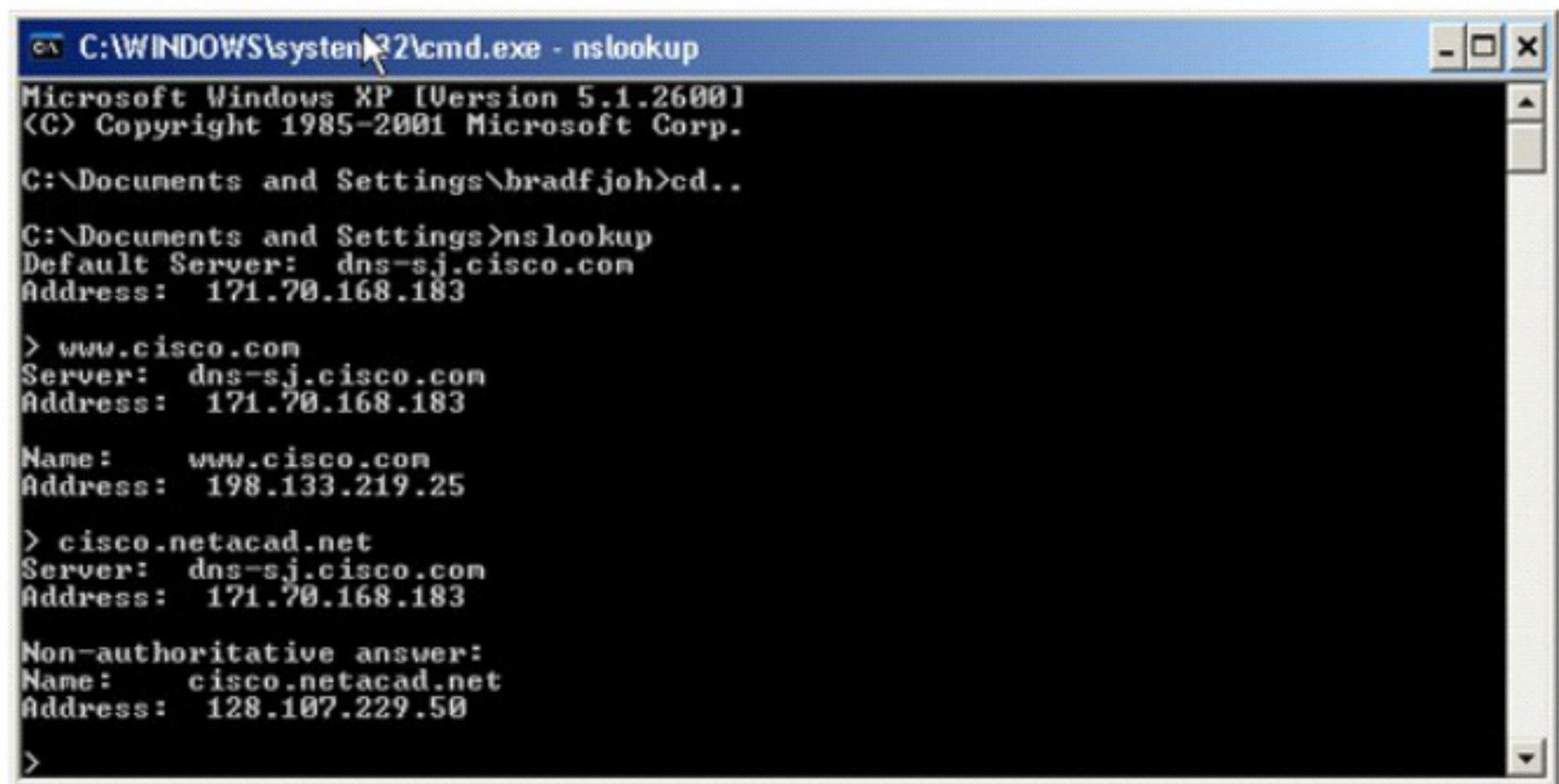
Resolução de Endereços (DNS)



Um nome humano legível é resolvido a seu endereço de dispositivo de rede numérico pelo protocolo DNS.

Uso de nslookup

- Os sistemas operacionais têm um utilitário chamado **nslookup** que permite que o usuário consulte manualmente os servidores de nome para decidir um nome de host.



```
C:\WINDOWS\system32\cmd.exe - nslookup
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\bradfjoh>cd..

C:\Documents and Settings>nslookup
Default Server: dns-sj.cisco.com
Address: 171.70.168.183

> www.cisco.com
Server: dns-sj.cisco.com
Address: 171.70.168.183

Name: www.cisco.com
Address: 198.133.219.25

> cisco.netacad.net
Server: dns-sj.cisco.com
Address: 171.70.168.183

Non-authoritative answer:
Name: cisco.netacad.net
Address: 128.107.229.50

>
```

Hierarquia de Servidores DNS

- O DNS utiliza um **sistema hierárquico descentralizado** para criar um **banco de dados distribuído**;

Quais os problemas com uma abordagem centralizada?

Hierarquia de Servidores DNS

- O DNS utiliza um **sistema hierárquico descentralizado** para criar um **banco de dados distribuído**;

Quais os problemas com uma abordagem centralizada?

- Problemas com uma abordagem centralizada:
 - Um único **ponto de falha**;
 - Imenso **volume de tráfego**;
 - Banco de dados centralizado **distante**;
 - Alto custo de **manutenção**;
 - Em resumo, **não é escalável**.

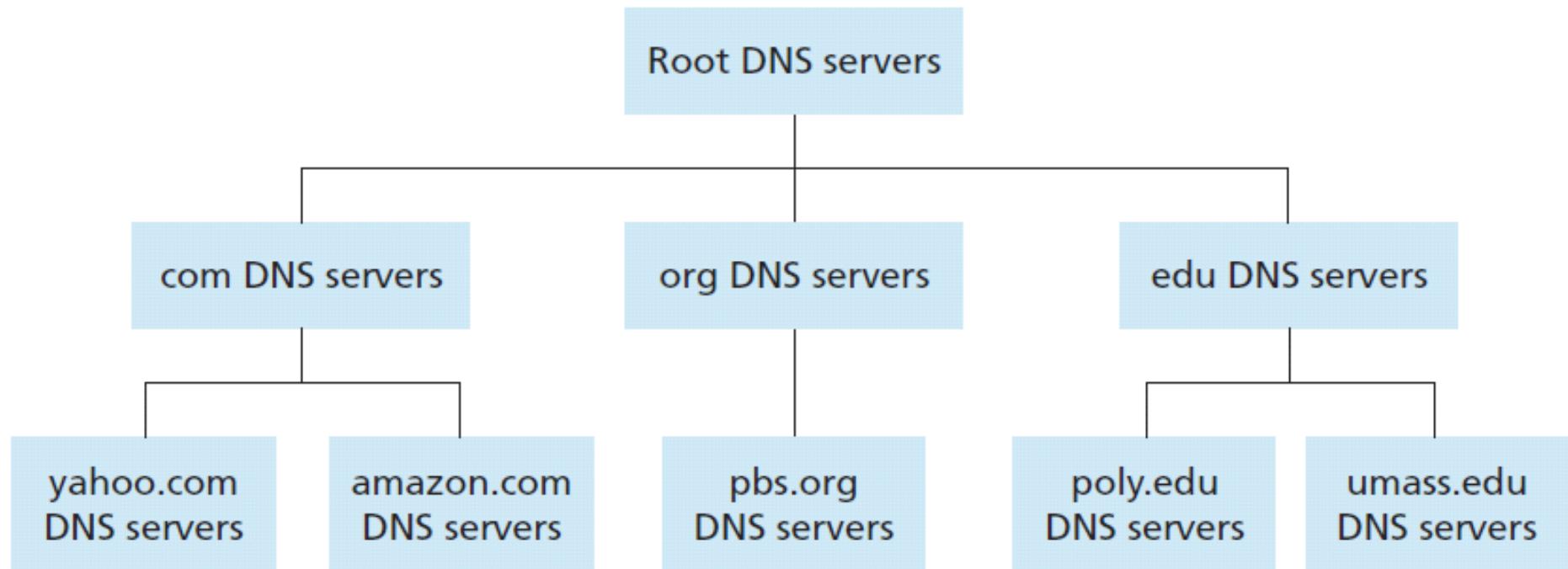
Hierarquia de Servidores DNS

- Quando um cliente faz uma consulta, o daemon DNS do servidor procura em seus **próprios registros** primeiro para ver se pode decidir o nome;
- Se não puder decidir o nome utilizando os seus registros armazenados, entra em **contato com outros servidores** para resolver o nome;
- Quando uma correspondência é encontrada e retornada ao servidor solicitante original, o servidor temporariamente armazena o endereço que corresponde ao nome em **cache**.

Hierarquia de Servidores DNS

- Existem **três classes** de servidores de nomes:
 - Servidores de nomes raiz;
 - Servidores de nomes de Domínio de Alto Nível (Top-Level Domain - TLD);
 - Servidores de nomes com autoridade.
- **Exemplo:** `www.amazon.com`
 - O cliente contatará um dos servidores raiz, que retornará endereços IP dos servidores TLD para o domínio de alto nível `.com`;
 - O cliente contatará um desses servidores TLD, que retornará o endereço IP de um servidor com autoridade para `amazon.com`;
 - O cliente contatará um dos servidores com autoridade para `amazon.com`, que retornará o endereço IP para o nome de host `www.amazon.com`.

Hierarquia de Servidores DNS

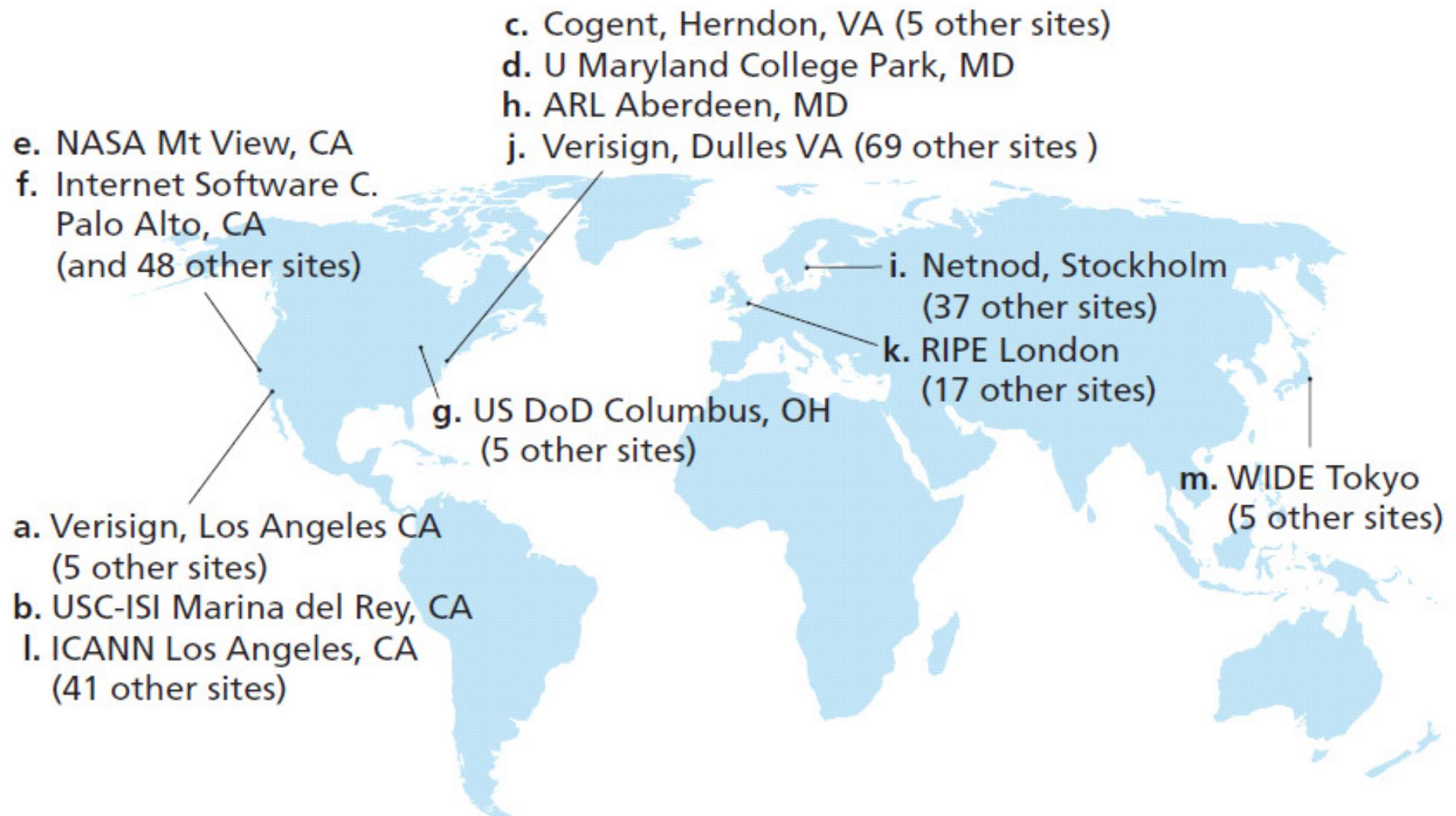


Hierarquia de Servidores DNS

- **Servidores de nome raiz**

- Na Internet há **13 servidores** de nomes raiz (denominados de A a M);
- A maior parte deles está localizada na América do Norte;
- Cada um dos 13 servidores é na realidade um **conglomerado de servidores replicados**, para fins de segurança e confiabilidade.

Hierarquia de Servidores DNS



Hierarquia de Servidores DNS

● Servidores de nomes TLD

- Os diferentes servidores TLD representam o **tipo de organização ou país de origem**. Exemplos:
 - **.br** - Brasil
 - **.au** - Austrália
 - **.jp** - Japão
 - **.com** - uma empresa ou setor
 - **.org** - uma organização sem fins lucrativos
 - **.edu** - universidades
 - **.gov** - governos

Hierarquia de Servidores DNS

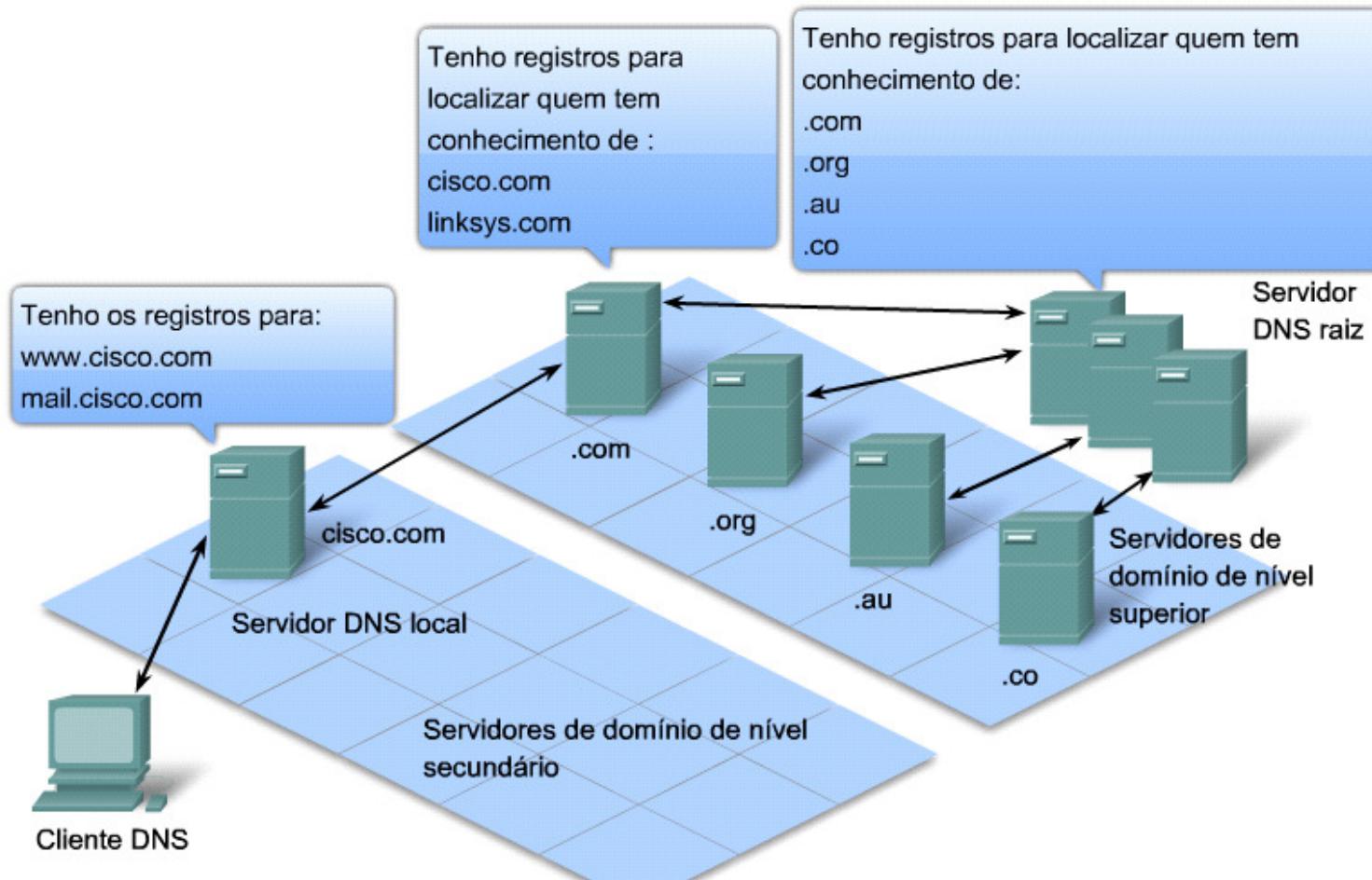
- **Servidores de nomes com autoridade**

- Toda organização que tiver hosts que podem ser acessados publicamente deve ter um servidor DNS com autoridade também acessível publicamente;
- As organizações podem implementar seu próprio servidor DNS ou pagar à uma ISP.

- **Servidores DNS locais**

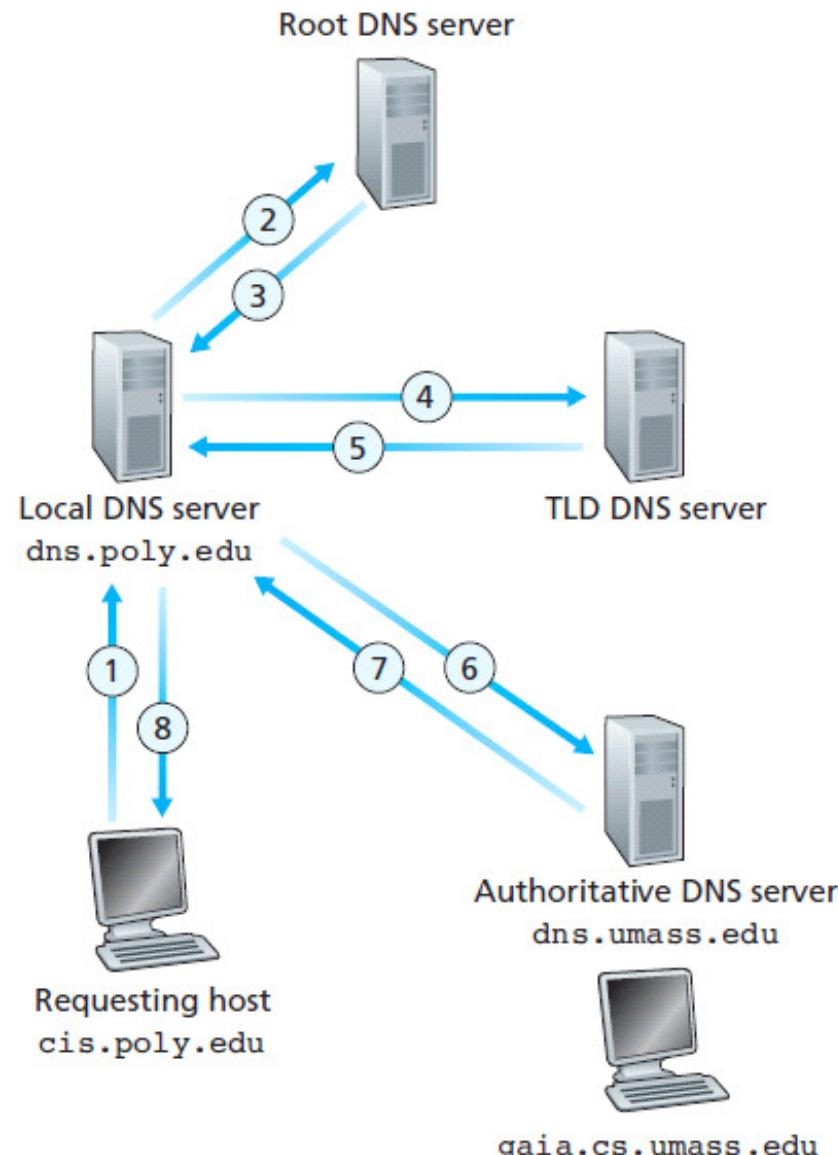
- Esse tipo de servidor não pertence, estritamente, à hierarquia de servidores, mas é central para a arquitetura DNS;
- Cada ISP tem um servidor de nomes local (servidor de nomes default);
- O servidor de nomes local de um host normalmente está “próximo” dele.

Hierarquia de Servidores DNS



Hierarquia de servidores DNS que contêm os registros dos recursos correspondentes aos nomes com endereços.

Interação entre Servidores DNS



Vídeo

Como funciona a Internet - Parte 3 - DNS (nic.br)

Vídeo

O que é DNS?

Vídeo

DNS Explained

Registros DNS

- Cada mensagem de resposta DNS carrega um ou mais **registros de recursos**;
- Um registro de recurso é uma **tupla de quatro elementos** com os seguintes campos: **(Name, Value, Type, TTL)**;
- TTL determina quando um recurso deve ser removido de um cache (**tempo de vida útil**);

Registros DNS

- Os significados de Name e Value dependem de Type:
 - Se Type=A, então Name é um nome de host e Value é o endereço IP do host. Ex: **(relay1.bar.foo.com, 145.37.93.126, A)**.
 - Se Type=NS, então Name é um domínio e Value é o nome de um servidor DNS com autoridade. Ex: **(foo.com, dns.foo.com, NS)**.
 - Se Type=CNAME, então Name é um apelido do host e Value é um nome canônico do host. Ex: **(foo.com, relay1.bar.foo.com, CNAME)**.
 - Se Type=MX, então Name é um apelido do servidor de correio e Value é um nome canônico deste servidor de e-mail. Ex: **(foo.com, mail.bar.foo.com, MX)**.

Formato de Mensagem DNS

- As comunicações do protocolo DNS utilizam um único formato, chamado de **mensagem**. Este formato de mensagem é utilizado para:
 - Consultas de cliente;
 - Respostas de servidor;
 - Mensagens de erro;
 - Transferência de informações de registro de recursos entre servidores.

Inserção de Registros no Banco de Dados do DNS

Imagine que você criou uma empresa chamada Network Utopia. O que você precisa fazer para colocar a página Web e o serviço de e-mail da sua empresa acessíveis publicamente na Internet?

Inserção de Registros no Banco de Dados do DNS

Imagine que você criou uma empresa chamada Network Utopia. O que você precisa fazer para colocar a página Web e o serviço de e-mail da sua empresa acessíveis publicamente na Internet?

- ① Registrar o nome de domínio networkutopia.com em uma entidade registradora;
- ② Informar os nomes e endereços IP dos seus servidores DNS com autoridade.
Ex: dns.networkutopia.com → 212.212.212.1;
- ③ A entidade registradora inserirá nos servidores TLD do domínio .com os seguintes registros de recurso:
 - (networkutopia.com, dns.networkutopia.com, NS);
 - (dns.networkutopia.com, 212.212.212.1, A);
- ④ Você deverá inserir nos servidores de nome com autoridade os seguintes registros de recurso:
 - (www.networkutopia.com, 212.212.212.2, A);
 - (mail.networkutopia.com, 212.212.212.3, MX).

Vídeo

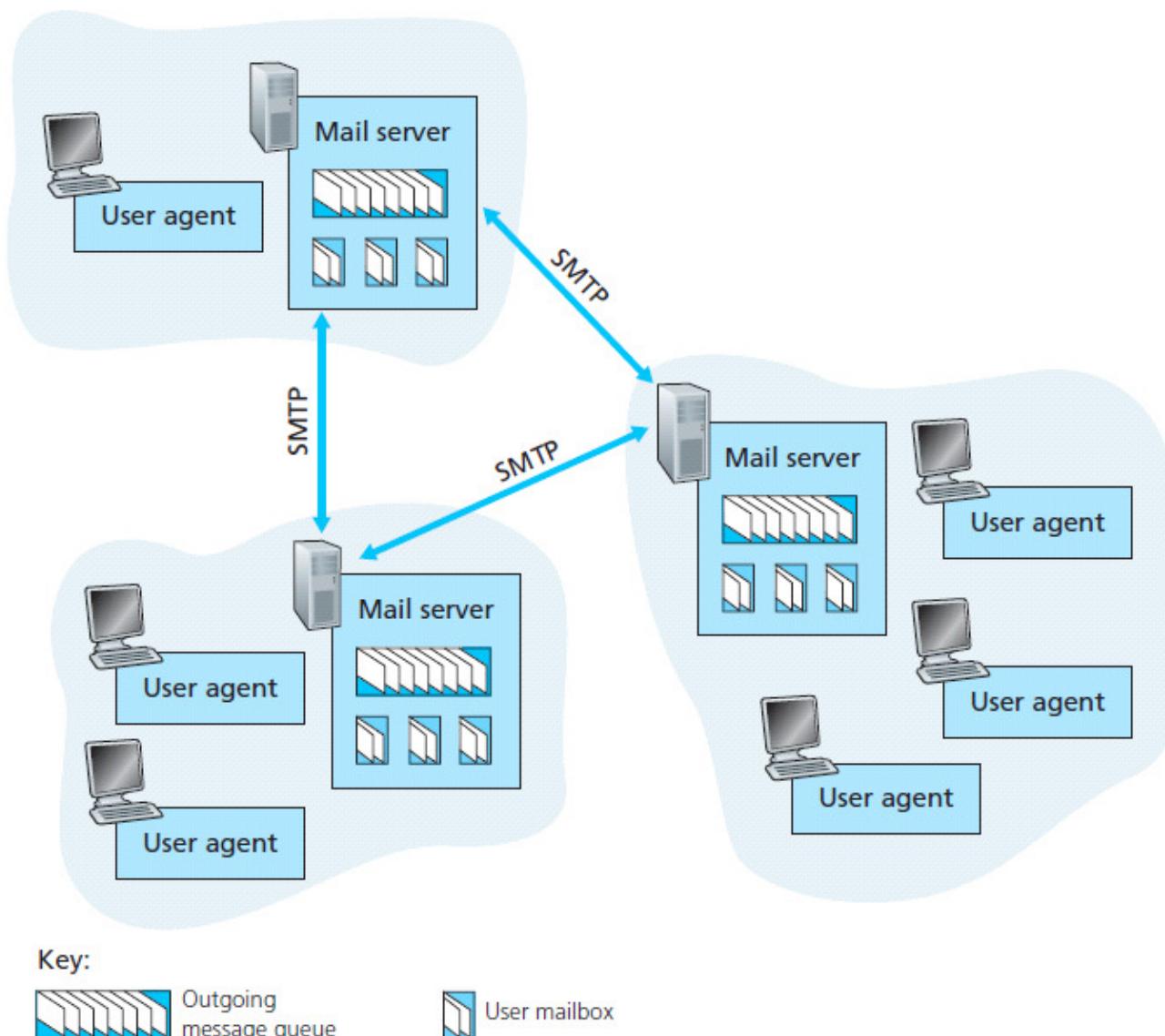
O que é um domínio?

(nic.br)

Fundamentos

- O e-mail, o serviço de rede **mais popular**, revolucionou a forma como as pessoas se comunicam graças a sua simplicidade e velocidade;
- Utiliza o modelo **cliente-servidor**;
- O sistema de correio da Internet é formado por três componentes principais:
 - **Agentes de usuário**;
 - **Servidores de correio**;
 - **SMTP**;

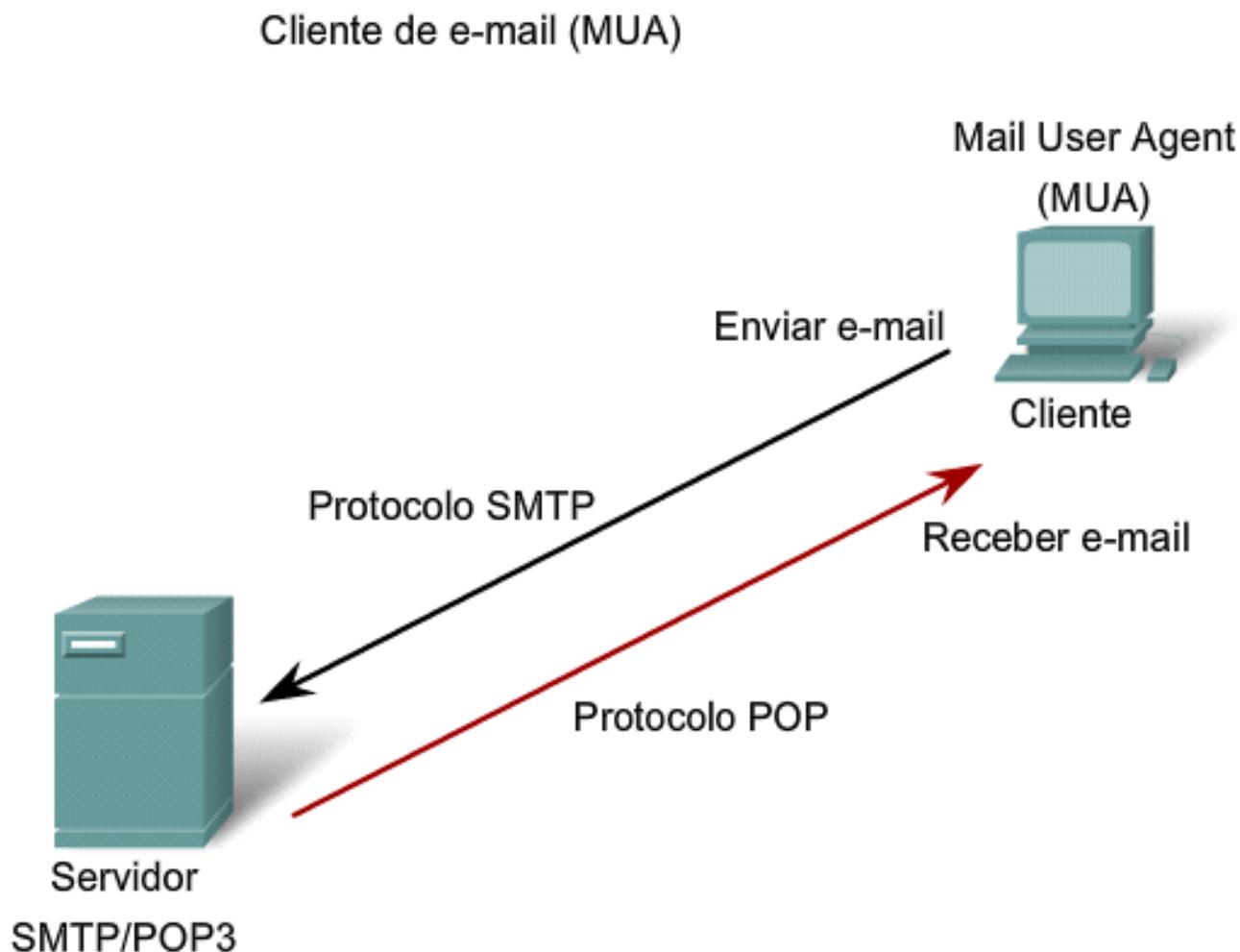
Fundamentos



Agentes de Usuário

- Cliente de e-mail → **Mail User Agent (MUA)** - Agente do Usuário de Correio;
- Agentes de usuários permitem que usuários leiam, respondam, transmitam, salvem e componham mensagens;
- O processo cliente **envia** e-mails ao servidor de e-mail através do protocolo **SMTP** (Simple Mail Transfer Protocol), e os **recebe** através do protocolo **POP** (Post Office Protocol);
- Como outra alternativa, os computadores que não têm um MUA ainda podem se conectar a um serviço de correspondência em um **navegador Web** para recuperar e enviar mensagens desta forma.

Agentes de Usuário



Os clientes enviam e-mails ao servidor usando SMTP e recebem e-mails com POP3.

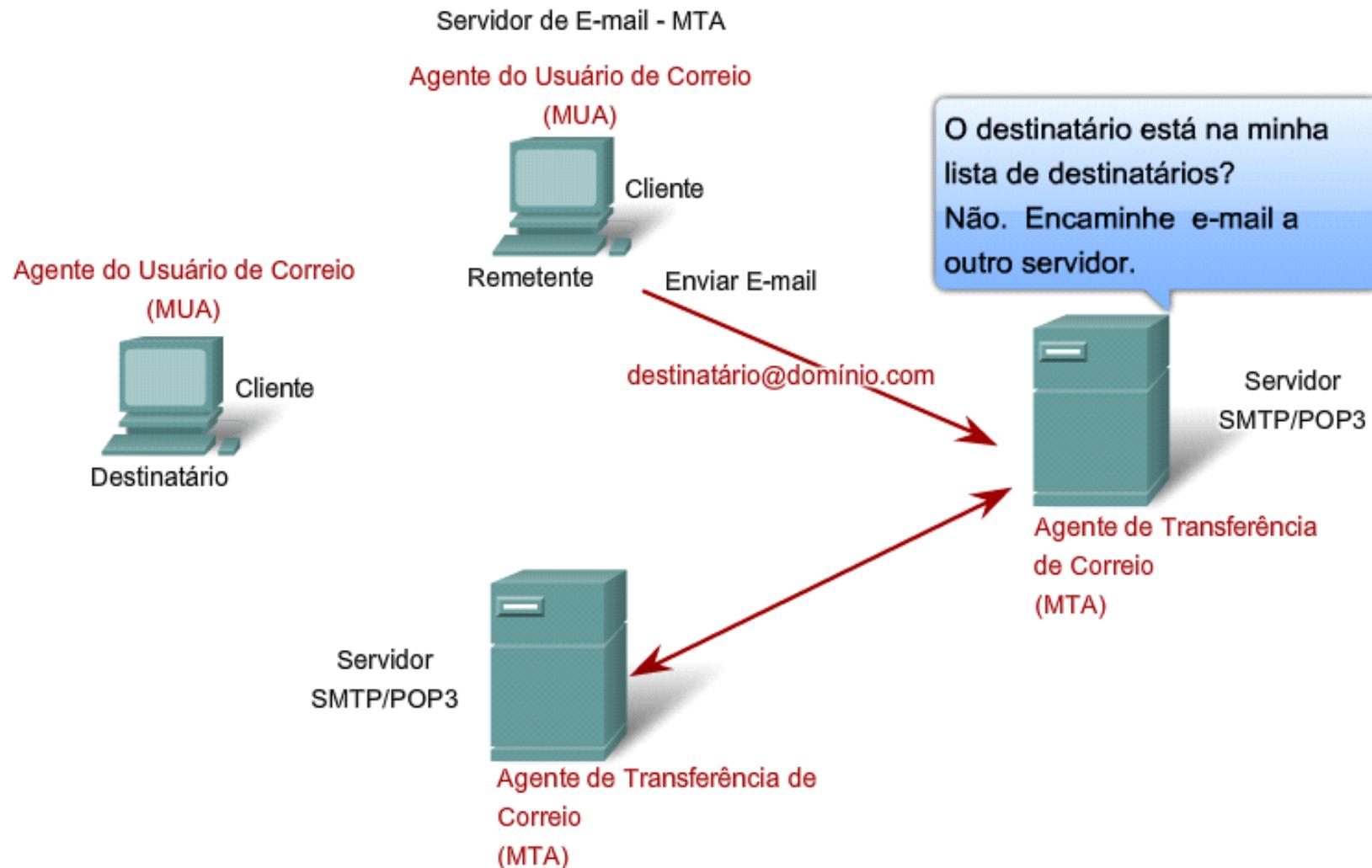
Servidores de Correio

- Servidores de correio formam o **núcleo da infraestrutura** do e-mail;
- Cada destinatário tem uma **caixa postal** localizada em um dos servidores de correio;
- Se o servidor de correio do remetente não puder entregar a correspondência ao servidor do destinatário, manterá a mensagem em uma **fila de mensagens** e tentará transferi-la mais tarde;
- O servidor de e-mail opera dois processos (agentes) separados:
 - **Mail Transfer Agent (MTA)** - Agente de Transferência de Correio;
 - **Mail Delivery Agent (MDA)** - Agente de Entrega de Correio;

Agente de Transferência de Correio

- O processo MTA é utilizado para **encaminhar** e-mail.
- O MTA recebe mensagens do MUA ou de outro MTA em outro servidor de e-mail;
- Se a correspondência for endereçada a um usuário cuja caixa de correio fique no servidor local, ela será passada para o MDA;
- Se o e-mail for para um usuário fora do servidor local, o MTA o encaminha para o MTA no servidor em questão.

Agente de Transferência de Correio

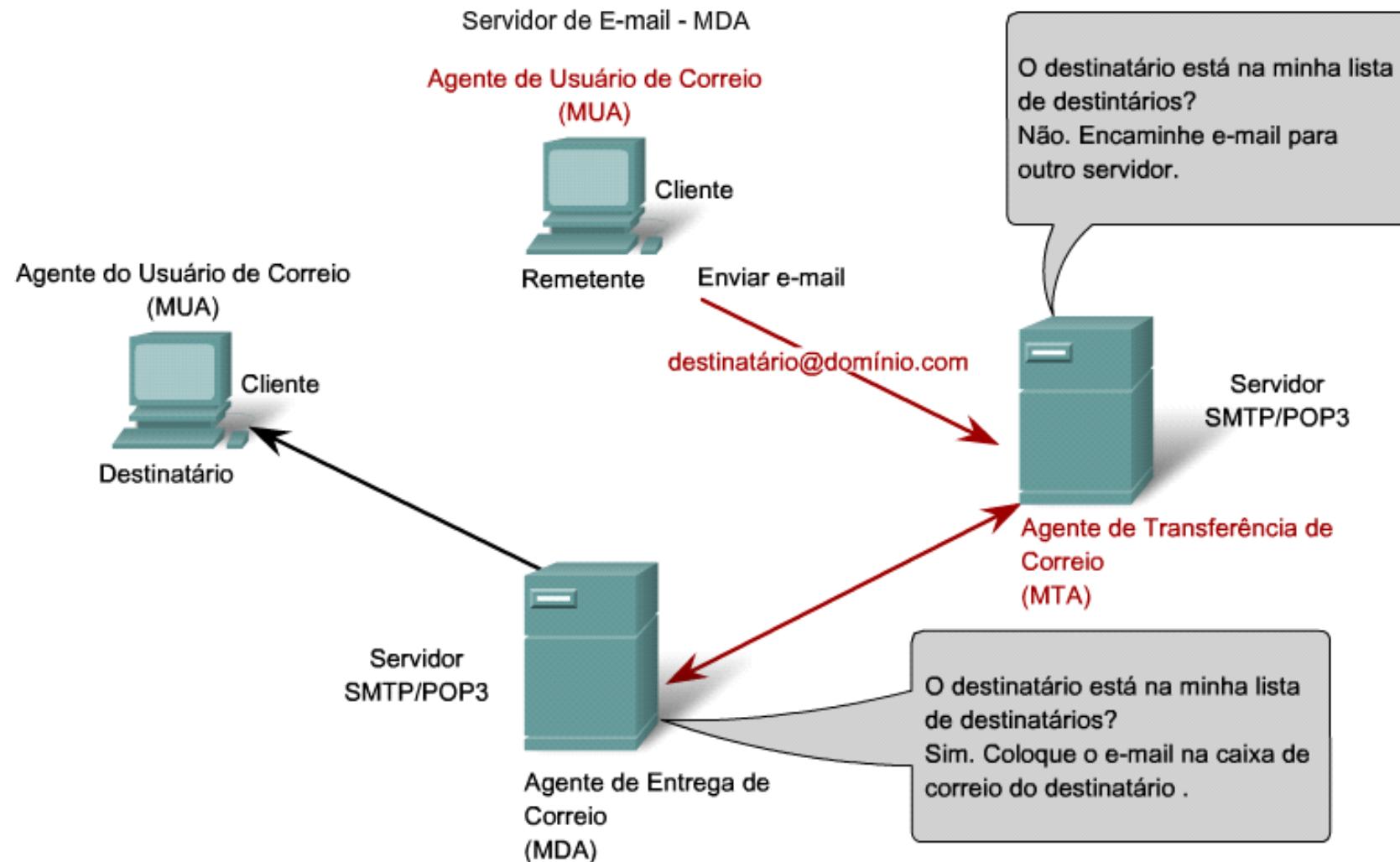


O processo do Agente de Transferência de Correio administra a manipulação de e-mail entre servidores e servidores.

Agente de Entrega de Correio

- O MDA aceita um e-mail de um MTA e faz a **entrega real**, i.e. coloca nas caixas de correio dos usuários adequados;
- O MDA também pode solucionar problemas de entrega final, como varredura de **vírus**, filtragem de **spam** e tratamento de **recebimento de retorno**.

Agente de Entrega de Correio



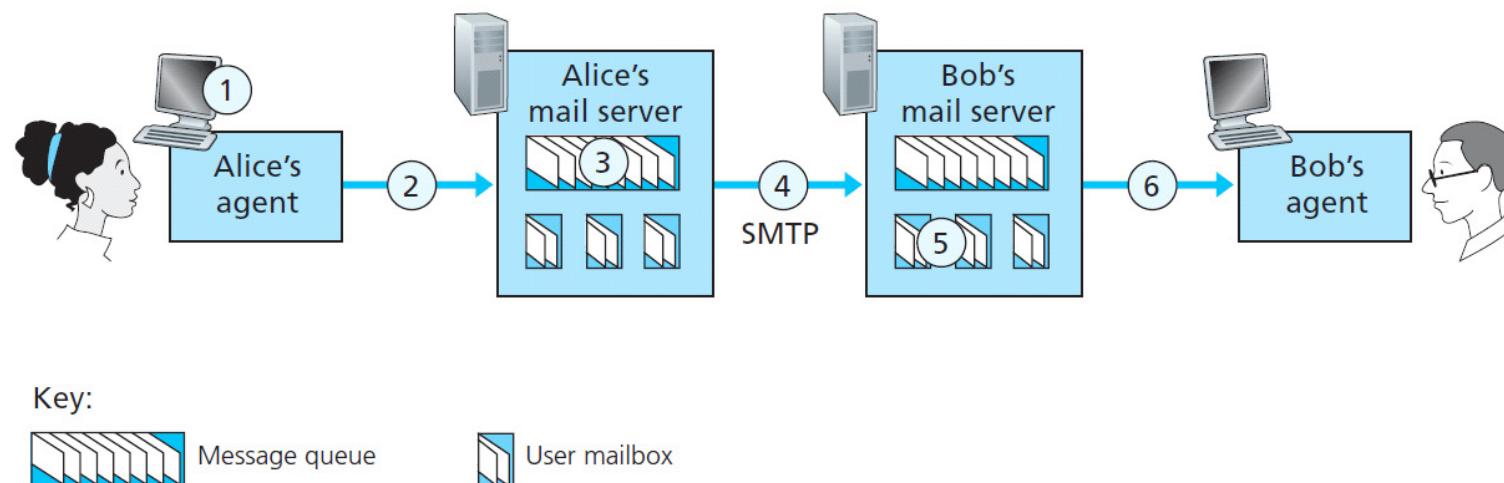
O processo do Agente de Entrega de Correio administra a entrega de e-mail entre servidores e clientes.

Protocolo SMTP

- SMTP é o **principal protocolo** do serviço de e-mail na Internet;
- Ele utiliza o serviço de transferência confiável de dados do **TCP** para realizar as seguintes transferências de e-mail:
 - Cliente de e-mail do remetente → Servidor de e-mail do remetente;
 - Servidor de e-mail do remetente → Servidor de e-mail do destinatário.
- Utiliza o modelo **cliente-servidor**;
- SMTP tem muitas **ótimas qualidades**, o que justifica o seu amplo uso na Internet, mas ele possui certas **características arcaicas**;
- Por exemplo, ele restringe o corpo de todas as mensagens a um simples **código ASCII** de 7 bits;
- Isso poderia fazer sentido no começo dos anos 80, mas não é compatível com os e-mails com **anexos multimídia** de hoje em dia. Para isso foi proposto o formato **MIME** (*Multipurpose Internet Mail Extensions*).

Protocolo SMTP

- ① Alice chama o seu MUA, fornece o endereço de Bob, compõe uma mensagem e instrui o seu MUA a enviar a mensagem;
- ② O MUA de Alice envia a mensagem para seu servidor de correio, onde ela é colocada em uma fila de mensagens;
- ③ O lado cliente do SMTP (servidor de Alice), abre uma conexão TCP para um servidor SMTP (servidor de Bob);
- ④ O cliente SMTP envia a mensagem para dentro da conexão TCP;
- ⑤ No servidor de correio de Bob, o lado servidor do SMTP recebe a mensagem e a coloca na caixa postal dele;
- ⑥ Bob chama seu MUA para ler a mensagem.



Protocolo SMTP

- O protocolo SMTP utiliza um conjunto rígido de comandos e respostas;
- Tais comandos suportam os procedimentos utilizados no SMTP;
- Comandos especificados no protocolo SMTP:
 - **HELO** - identifica o processo do cliente SMTP para o processo do servidor SMTP;
 - **EHLO** - é uma versão mais recente do HELO, que inclui extensões de serviços;
 - **MAIL FROM** - identifica o remetente;
 - **RCPT TO** - identifica o destinatário;
 - **DATA** - identifica o corpo da mensagem;
 - **CRLF.CRLF** (CR - Carriage Return, LF - Line Feed)) - indica o final da mensagem;
 - **QUIT** - finaliza a conexão.

Protocolo SMTP

- Exemplo: **alice@crepes.fr** envia e-mail para **bob@hamburger.edu**

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr ... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Comparação SMTP x HTTP

Quais as semelhanças entre o SMTP e o HTTP?

Comparação SMTP x HTTP

Quais as semelhanças entre o SMTP e o HTTP?

- Ambos os protocolos são usados para **transferir arquivos** de um host para outro;
 - O HTTP transfere arquivos (objetos) de um servidor Web para um cliente Web (browser);
 - O SMTP transfere arquivos (mensagens de e-mail) de um servidor de correio para outro.
- Ao transferir arquivos, o HTTP persistente e o SMTP usam **conexões TCP persistentes**.

Comparação SMTP x HTTP

Quais as diferenças entre o SMTP e o HTTP?

Comparação SMTP x HTTP

Quais as diferenças entre o SMTP e o HTTP?

- Recuperação x envio de informações

- O HTTP é um **protocolo de recuperação de informações** (*pull protocol*);
- No HTTP alguém carrega as informações em um servidor Web e os usuários utilizam o HTTP para recuperá-las do servidor quando quiserem;
- O SMTP é um **protocolo de envio de informações** (*push protocol*);
- No SMTP, o servidor de correio remetente envia o arquivo para o servidor de correio destinatário.

Comparação SMTP x HTTP

Quais as diferenças entre o SMTP e o HTTP?

- **Codificação do corpo da mensagem**

- O SMTP exige que cada mensagem (cabeçalho e corpo) esteja no formato ASCII de 7 bits;
- O HTTP não exige a codificação ASCII, dados binários (ex. imagens) podem ser transmitidos tais como estão.

- **Documentos que contém texto e mídias**

- O HTTP encapsula cada objeto em sua própria mensagem HTTP;
- O SMTP coloca todos os objetos em uma única mensagem.

Formatos de Mensagem de Correio

- Quando uma mensagem de e-mail é enviada, um **cabeçalho** contendo informações periféricas antecede o **corpo da mensagem** em si;
- As linhas de cabeçalho e o corpo da mensagem são separados por uma linha em branco (CRLF);
- As linhas de cabeçalho são diferentes dos comandos SMTP; os comandos SMTP fazem parte do protocolo de apresentação SMTP, enquanto as linhas de cabeçalho fazem parte da própria mensagem de correio.

Exemplo de cabeçalho

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

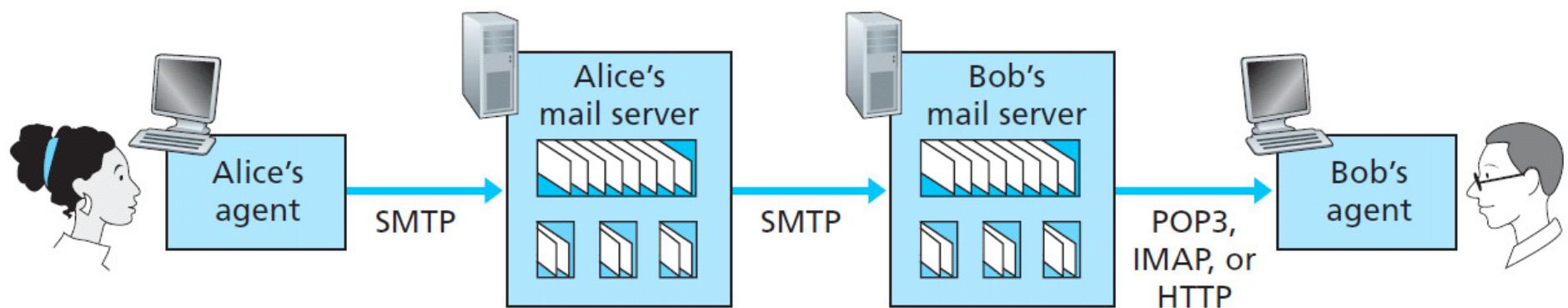
Protocolos de Acesso ao Correio

- Protocolos de acesso ao correio transferem mensagens do servidor de correio do destinatário para a máquina local do destinatário;
- Esses protocolos realizam uma operação de **recuperação** (*pull*) de mensagens;
- Exemplos de protocolos populares de acesso ao correio:
 - **POP3** (Post Office Protocol version 3);
 - **IMAP** (Internet Mail Access Protocol);
 - **HTTP** (Web mail).

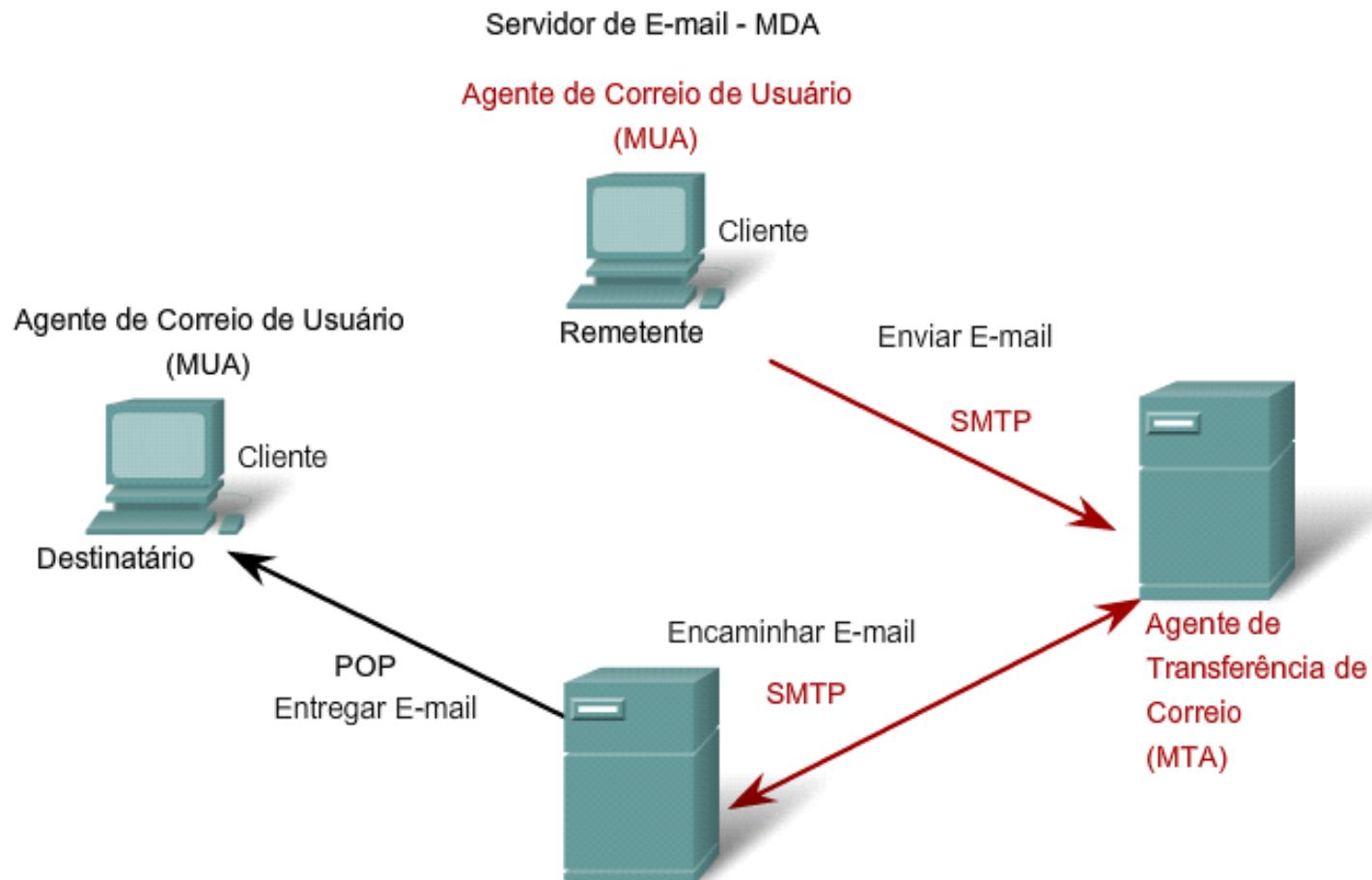
Protocolos de Acesso ao Correio

Por que Bob não instala um servidor de correio em sua própria máquina, de forma que o servidor de correio de Alice dialogasse diretamente com o PC de Bob?

Por que o agente de usuário de Alice não se comunica diretamente com o servidor de correio de Bob?



Protocolos SMTP/POP



O protocolo SMTP é usado para enviar e-mail.

O protocolo POP é usado para receber e-mail.

Agente de Entrega de Correio (MDA)

Agente de Transferência de Correio (MTA)

Protocolo POP3

- O POP3 é um protocolo de acesso de correio **extremamente simples**;
- O POP3 começa quando o agente de usuário (o cliente) abre uma **conexão TCP com o servidor** de correio (o servidor) na porta 110;
- Com a conexão TCP ativada, o protocolo passa por 3 fases:
 - **Autorização**: autenticação de usuário;
 - **Transação**: recuperação de mensagens, marcação de mensagens e obtenção de estatísticas;
 - **Atualização**: ocorre após o cliente ter dado o comando `quit` que encerra a sessão POP3. Nesse momento o servidor apaga as mensagens marcadas.

Protocolo POP3

- Comandos da fase de autorização:
 - user <user name>;
 - pass <password>.

Exemplo da fase de autorização

```
telnet mailServer 110
+OK POP3 server ready
user bob
+OK
pass hungry
+OK user successfully logged on
```

- Se um comando é escrito errado, o POP3 responderá com uma mensagem -ERR.

Protocolo POP3

- O POP3 possui 2 modos de operação: “**ler-e-apagar**” e “**ler-e-guardar**”.
- Comandos da fase de transação:
 - list;
 - retr;
 - dele.

Exemplo da fase de transação

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: (blah blah ...
S: .....
S: .....blah)
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off

C: retr 2
S: (blah blah ...
S: .....
S: .....blah)
S: .
C: dele 1
```

Protocolo IMAP

- O protocolo IMAP é significativamente **mais complexo** que o POP3;
- O IMAP é adequado para **usuários nomâdes**, que gostariam de manter uma **hierarquia de pastas em um servidor remoto** que possa ser acessado de qualquer computador;
- Um servidor IMAP associa cada mensagem a uma pasta;
- O IMAP provê comandos que permitem que os usuários criem pastas e transfiram mensagens de uma para outra.

E-mail pela Web

- Os usuários enviam e acessam e-mails por meio de seus **browsers Web**;
- O agente de usuário é um browser Web comum e o usuário se comunica com sua caixa postal remota via **HTTP**;
- Exemplo: Alice quer enviar um e-mail a Bob utilizando a Web:
 - O browser de Alice envia a mensagem para o seu servidor de correio via HTTP, e não SMTP;
 - O servidor de correio de Alice envia a mensagem ao servidor de correio de Bob via SMTP;
 - O servidor de correio de Bob envia a mensagem ao browser de Bob via HTTP, e não POP3 ou IMAP.

Conteúdo

- 1 Fundamentos
- 2 A Web e o HTTP
- 3 DNS: O Serviço de Diretório da Internet
- 4 Correio Eletrônico na Internet
- 5 Transferência de Arquivos: FTP
 - Fundamentos
 - Comandos e Respostas FTP
- 6 Aplicações Peer-to-Peer
 - Fundamentos
 - Distribuição de Arquivos P2P
 - Escalabilidade da Arquitetura P2P
 - Protocolo BitTorrent

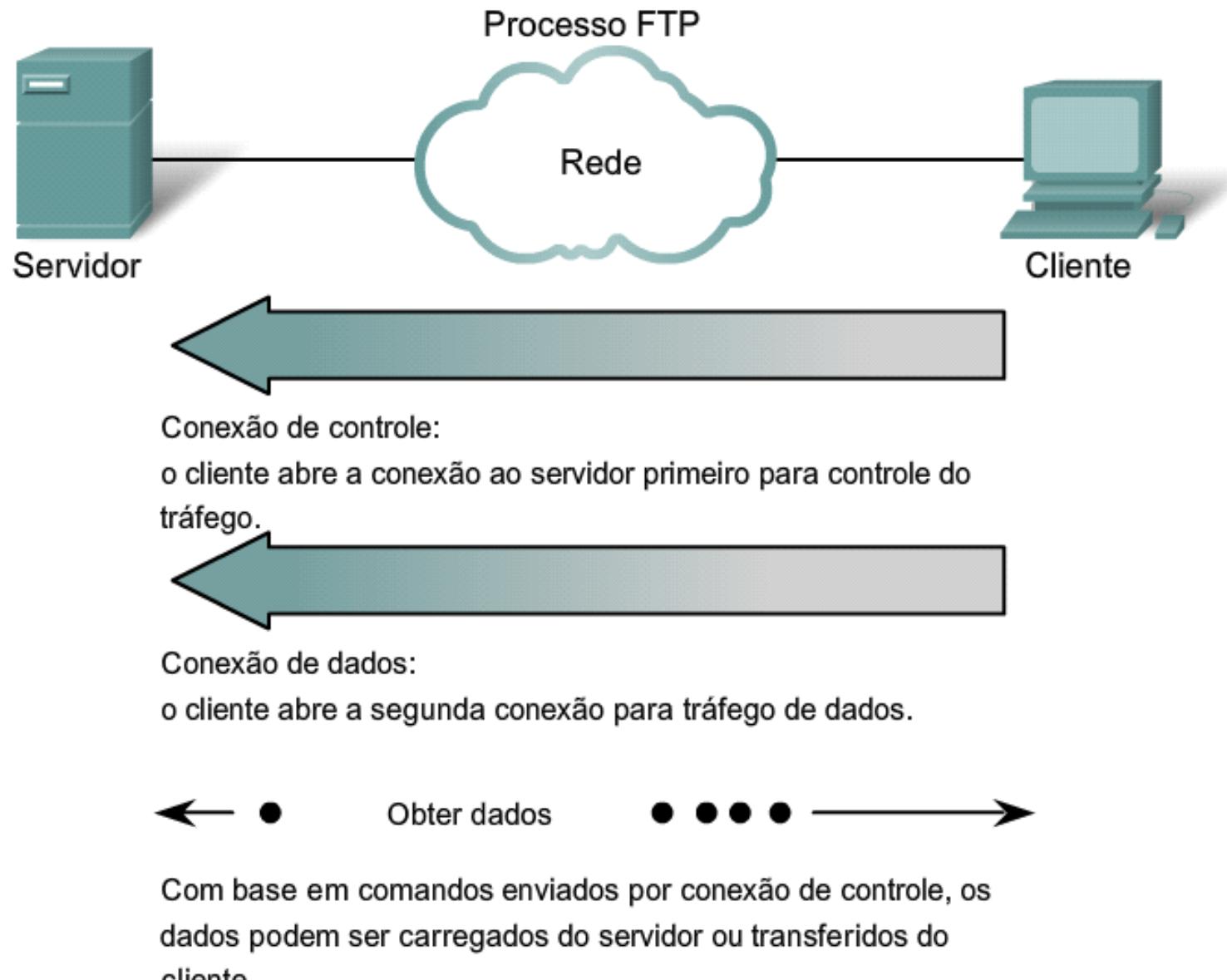
Fundamentos

- O FTP (File Transfer Protocol) foi desenvolvido para possibilitar **transferências de arquivos** entre um cliente e um servidor;
- Um **cliente FTP** é uma aplicação que roda em um computador e utilizado para carregar e baixar arquivos de um servidor que executa o **daemon FTP** (FTPD);
- A transferência de arquivos pode acontecer em **ambas as direções**;

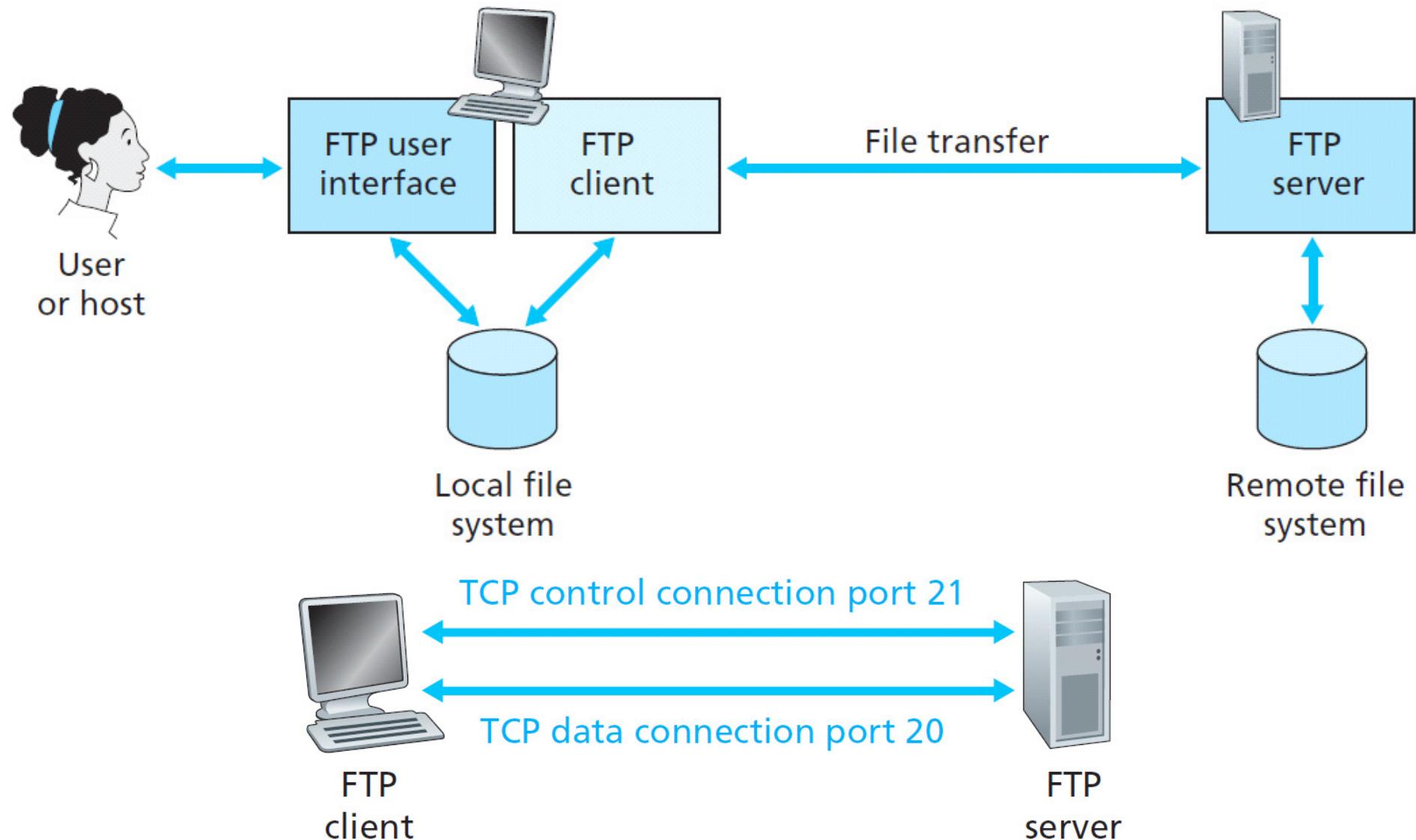
Fundamentos

- Para transferir os arquivos com sucesso, o FTP precisa de **duas conexões** entre o cliente e o servidor:
 - Conexão para **comandos e respostas**
 - Utiliza a porta TCP 21;
 - Utilizada para controlar o tráfego, consistindo de comandos do cliente e respostas do servidor.
 - Conexão para a real **transferência do arquivo**
 - Utiliza a porta TCP 20;
 - É criada toda vez que houver um arquivo transferido.

Transferência de Arquivos: FTP



Transferência de Arquivos: FTP



Modo de Operação

- ① O usuário informa ao agente de usuário FTP o nome do host remoto;
- ② O processo cliente FTP na máquina local estabelece uma conexão TCP de controle (porta 21) com o processo servidor FTP no host remoto;
- ③ O usuário informa o nome de usuário e senha, os quais são enviados pela conexão TCP de controle como parte dos comandos FTP;
- ④ O servidor FTP autoriza o usuário;

Modo de Operação

- ⑤ O cliente FTP envia por essa conexão de controle comandos para mudar o diretório remoto;
- ⑥ O cliente FTP envia um comando de transferência de arquivos pela conexão TCP de controle;
- ⑦ O servidor FTP inicia uma conexão TCP de dados (porta 20) com o cliente FTP;
- ⑧ O usuário copia um ou mais arquivos armazenados no sistema de arquivos local para o sistema de arquivos remoto (ou vice versa). Para cada arquivo é criada uma nova conexão TCP de dados (conexão não-persistente).

Comparação FTP x HTTP

Quais as semelhanças e diferenças entre o FTP e o HTTP?

Comparação FTP x HTTP

Quais as semelhanças e diferenças entre o FTP e o HTTP?

- **Semelhanças**
 - Os dois protocolos rodam em cima do TCP;
- **Diferenças**
 - O FTP usa duas conexões TCP paralelas, enquanto o HTTP usa somente uma;
 - O FTP envia informações de controle **out-of-band**, enquanto o HTTP envia **in-band**;
 - O servidor FTP precisa **rastrear o estado dos usuários** conectados, enquanto o HTTP não precisa.

Conteúdo

- 1 Fundamentos
- 2 A Web e o HTTP
- 3 DNS: O Serviço de Diretório da Internet
- 4 Correio Eletrônico na Internet
- 5 Transferência de Arquivos: FTP
 - Fundamentos
 - Comandos e Respostas FTP
- 6 Aplicações Peer-to-Peer
 - Fundamentos
 - Distribuição de Arquivos P2P
 - Escalabilidade da Arquitetura P2P
 - Protocolo BitTorrent

Comandos FTP

- Os comandos são enviados do cliente para o servidor na conexão TCP de controle, e são codificados no formato **ASCII de 7 bits**;
- Alguns dos comandos mais comuns:
 - **USER username**: usado para enviar a identificação do usuário para o servidor;
 - **PASS password**: usado para enviar a senha do usuário para o servidor;
 - **LIST**: usado para pedir ao servidor que envie uma lista de todos os arquivos no diretório remoto atual. A lista dos arquivos é enviada sobre uma (nova e não-persistente) conexão de dados, ao invés de numa conexão TCP de controle;

Comandos FTP

- Alguns dos comandos mais comuns (cont.):
 - **RETR filename**: usado para recuperar (download) um arquivo do diretório atual do host remoto. Este comando faz com que o host remoto inicie uma conexão TCP de dados e envie o arquivo requisitado sobre esta conexão;
 - **STOR filename**: Usado para armazenar (upload) um arquivo no diretório atual do host remoto.

Respostas FTP

- Cada comando do cliente é seguido de uma resposta do servidor;
- Alguns respostas típicas:
 - **331 Username OK, password required;**
 - **125 Data connection already open; transfer starting;**
 - **425 Can't open data connection;**
 - **452 Error writing file;**

Conteúdo

- 1 Fundamentos
- 2 A Web e o HTTP
- 3 DNS: O Serviço de Diretório da Internet
- 4 Correio Eletrônico na Internet
- 5 Transferência de Arquivos: FTP
 - Fundamentos
 - Comandos e Respostas FTP
- 6 Aplicações Peer-to-Peer
 - Fundamentos
 - Distribuição de Arquivos P2P
 - Escalabilidade da Arquitetura P2P
 - Protocolo BitTorrent

Fundamentos

- Numa arquitetura P2P, existe uma mínima (ou nenhuma) confiança em servidores de infraestrutura sempre conectados;
- Ao invés disso, **pares de hosts conectados intermitentemente** (*peers*), se comunicam diretamente uns com os outros;
- Os peers não pertencem a nenhum provedor de serviço, pois eles são desktops e notebooks **controlados pelos usuários**.

Conteúdo

- 1 Fundamentos
- 2 A Web e o HTTP
- 3 DNS: O Serviço de Diretório da Internet
- 4 Correio Eletrônico na Internet
- 5 Transferência de Arquivos: FTP
 - Fundamentos
 - Comandos e Respostas FTP
- 6 Aplicações Peer-to-Peer
 - Fundamentos
 - Distribuição de Arquivos P2P
 - Escalabilidade da Arquitetura P2P
 - Protocolo BitTorrent

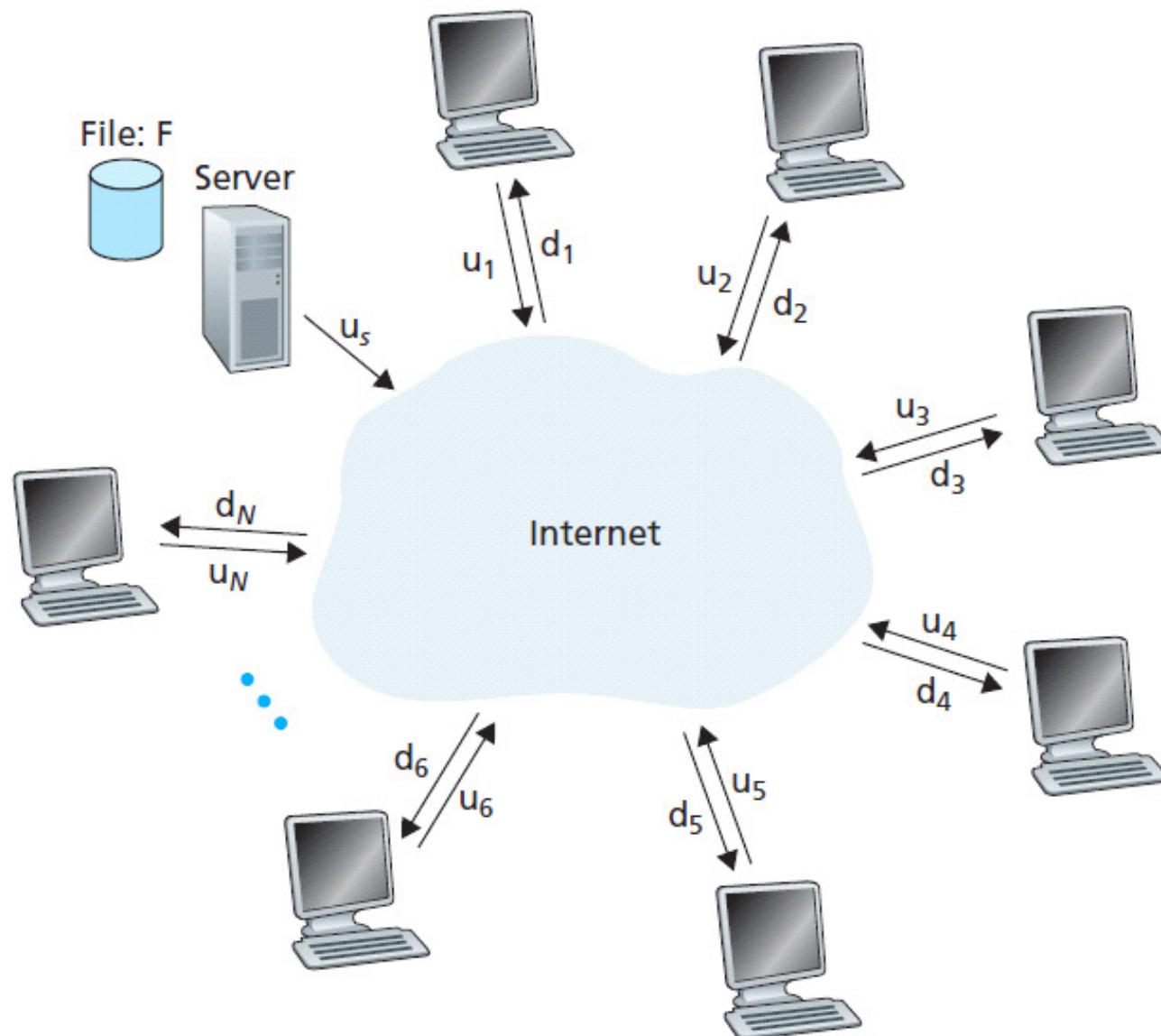
Distribuição de Arquivos P2P

- Distribuição de um arquivo grande de um servidor único para um grande número de hosts (peers)
 - **Cliente-servidor**
 - O servidor deve enviar uma cópia do arquivo para cada um dos peers;
 - Enorme sobrecarga no servidor;
 - Consumo de grande quantidade de largura de banda do servidor.
 - **Peer-to-peer**
 - Cada peer pode redistribuir qualquer parte do arquivo que ele tenha recebido de outros peers;
 - O servidor é ajudado pelos peers no processo de distribuição dos arquivos.
 - Atualmente, o protocolo de distribuição de arquivos P2P mais popular é o **BitTorrent**.

Conteúdo

- 1 Fundamentos
- 2 A Web e o HTTP
- 3 DNS: O Serviço de Diretório da Internet
- 4 Correio Eletrônico na Internet
- 5 Transferência de Arquivos: FTP
 - Fundamentos
 - Comandos e Respostas FTP
- 6 Aplicações Peer-to-Peer
 - Fundamentos
 - Distribuição de Arquivos P2P
 - Escalabilidade da Arquitetura P2P
 - Protocolo BitTorrent

Problema Ilustrativo de Distribuição de Arquivos



Problema Ilustrativo de Distribuição de Arquivos

- Um arquivo de tamanho F bits deve ser distribuído para N peers;
- O **tempo de distribuição** é o tempo necessário para todos os N peers receberem a sua cópia do arquivo;
- Assumimos que o núcleo da Internet possui uma largura de banda abundante, portanto os gargalos estão localizados nas redes de acesso;
- Comparaçāo **cliente-servidor x P2P**.

Problema Ilustrativo de Distribuição de Arquivos

● Cliente-servidor

- O servidor deve transmitir uma cópia do arquivo para cada um dos N peers. Portanto, ele deve transmitir NF bits;
- Como a taxa de upload do servidor é u_s , o tempo para o servidor distribuir o arquivo deve ser pelo menos NF/u_s ;
- $d_{min} = \min\{d_1, d_2, \dots, d_N\}$ é a taxa de download mínima dentre todos os peers. Dessa forma, temos que o tempo mínimo de distribuição por parte dos peers deve ser pelo menos F/d_{min} ;
- Dessa forma, temos que o tempo de distribuição na arquitetura cliente-servidor é:

$$D_{cs} = \max \left\{ \frac{NF}{u_s}, \frac{F}{d_{min}} \right\}. \quad (1)$$

- Para um valor de N suficientemente grande, temos que o tempo de distribuição cresce linearmente com o do número de peers N .

Problema Ilustrativo de Distribuição de Arquivos

● Peer-to-peer

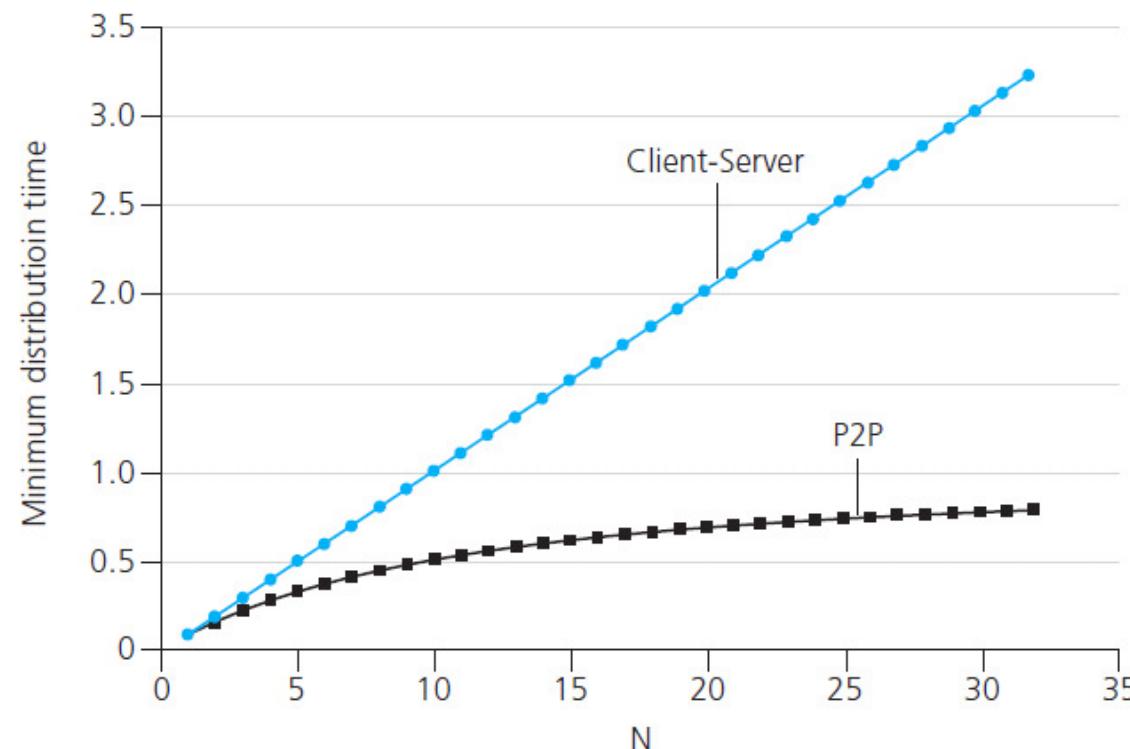
- No começo da distribuição, somente o servidor tem o arquivo. Para disponibilizar o arquivo para a comunidade de peers, o servidor deve enviar cada bit do arquivo pelo seu link de acesso pelo menos uma vez;
- Portanto, o tempo de distribuição mínimo é pelo menos F/u_s ;
- Como no caso do cliente-servidor, o tempo de distribuição mínimo na perspectiva dos peers é F/d_{min} ;
- A capacidade de upload total do sistema como um todo é dada por $u_{total} = u_s + u_1 + \dots + u_N$;
- O sistema deve entregar (upload) F bits para cada um dos N peers, o que resulta em um tempo de distribuição mínimo de NF/u_{total} ;
- Dessa forma, temos que o tempo de distribuição na arquitetura P2P é:

$$D_{p2p} = \max \left\{ \frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i} \right\}. \quad (2)$$

Problema Ilustrativo de Distribuição de Arquivos

- **Cliente-servidor x Peer-to-peer**

- Assumimos que todos os peers têm a mesma taxa de upload u ;
- Assumimos também que: $F/u = 1$ hora, $u_s = 10u$, $d_{min} \geq u_s$;
- Pode-se concluir que a arquitetura P2P é **auto-escalável**; a escalabilidade é consequência direta do fato de que os peers são ao mesmo tempo **redistribuidores e consumidores** de bits.



Conteúdo

- 1 Fundamentos
- 2 A Web e o HTTP
- 3 DNS: O Serviço de Diretório da Internet
- 4 Correio Eletrônico na Internet
- 5 Transferência de Arquivos: FTP
 - Fundamentos
 - Comandos e Respostas FTP
- 6 Aplicações Peer-to-Peer
 - Fundamentos
 - Distribuição de Arquivos P2P
 - Escalabilidade da Arquitetura P2P
 - Protocolo BitTorrent

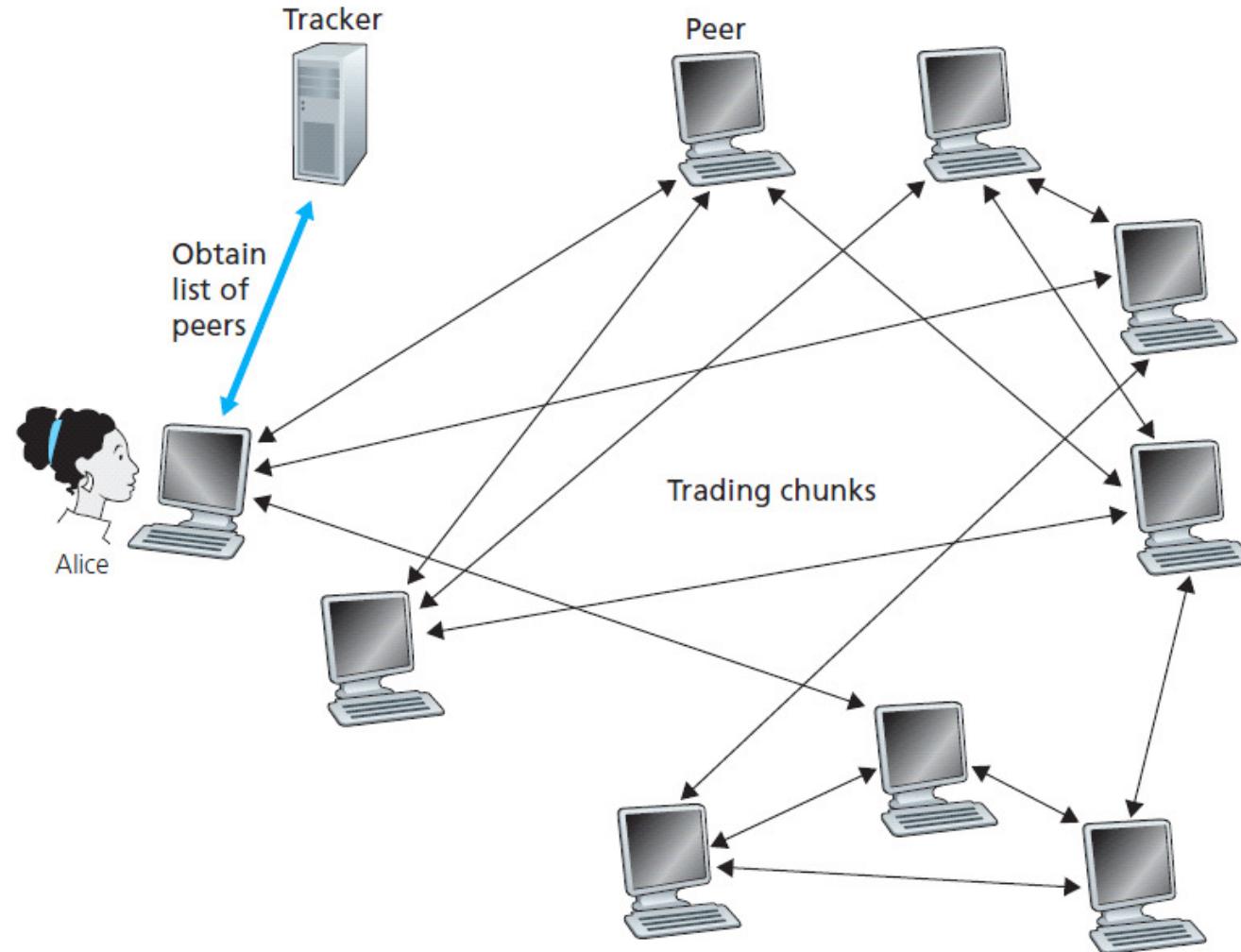
Protocolo BitTorrent

- A coleção de todos os pares que participam da distribuição de um determinado arquivo é chamada de **torrent**;
- Os peers em um torrent fazem o download de **blocos** de tamanho igual do arquivo entre si, com um tamanho típico de bloco de 256 KBytes;
- Enquanto um peer faz o download de blocos, faz também upload de blocos para outros pares;
- Uma vez que o peer adquire todo o arquivo, ele pode de forma **egoísta** sair do torrent, ou de forma **altruista** permanecer no torrent;
- Qualquer peer pode sair do torrent a qualquer momento com apenas um subconjunto de blocos, e depois voltar;

Protocolo BitTorrent

- Cada torrent tem um nó de infraestrutura chamado **rastreador**;
- Quando um peer chega em um torrent, ele se registra com o rastreador; dessa forma o rastreador mantém um registro dos peers que participam do torrent;
- Quando um novo peer (Alice) chega no torrent, o rastreador seleciona aleatoriamente um subconjunto de N peers do conjunto de pares participantes, e envia os endereços IP desses N peers para Alice;
- Com a **lista de peers**, Alice tenta estabelecer **conexões TCP simultâneas** com todos os peers da lista;
- Os peers vizinhos de um peer podem flutuar com o tempo.

Protocolo BitTorrent



Protocolo BitTorrent

- Periodicamente, Alice pedirá a cada um de seus peers vizinhos (nas conexões TCP) a lista de quais blocos eles têm;
- A qualquer momento, Alice terá um subconjunto de blocos e saberá quais blocos seus vizinhos têm;
- Alice terá duas decisões importantes a fazer:

Quais blocos ela deve solicitar primeiro de seus vizinhos?

A quais vizinhos ela deve enviar os blocos solicitados?

Protocolo BitTorrent

- Periodicamente, Alice pedirá a cada um de seus peers vizinhos (nas conexões TCP) a lista de quais blocos eles têm;
- A qualquer momento, Alice terá um subconjunto de blocos e saberá quais blocos seus vizinhos têm;
- Alice terá duas decisões importantes a fazer:

Quais blocos ela deve solicitar primeiro de seus vizinhos?

Os mais raros.

A quais vizinhos ela deve enviar os blocos solicitados?

Aos que fornecem seus dados com a maior taxa.