Fundamentos de Redes de Computadores

Tiago Alves

Faculdade UnB Gama
Universidade de Brasília

UnB Gama
O novo endereço da tecnologia

Sumário

Camada de Aplicação

UnB Gama
O novo endereço da tecnologia

Aplicações de rede são as razões de ser de uma rede de computadores.

Aplicações

- modo texto: e-mail, acesso remoto a computadores, transferência de arquivos, grupos de discussão (notícias)
- WWW, P2P
- VoIP, videoconferências (Skype)
- Multimídia: YouTube, VoD (NetFlix)
- Games: Second Life, WoW
- Redes sociais: Orkut, Facebook, Twitter

Princípios de aplicações de rede

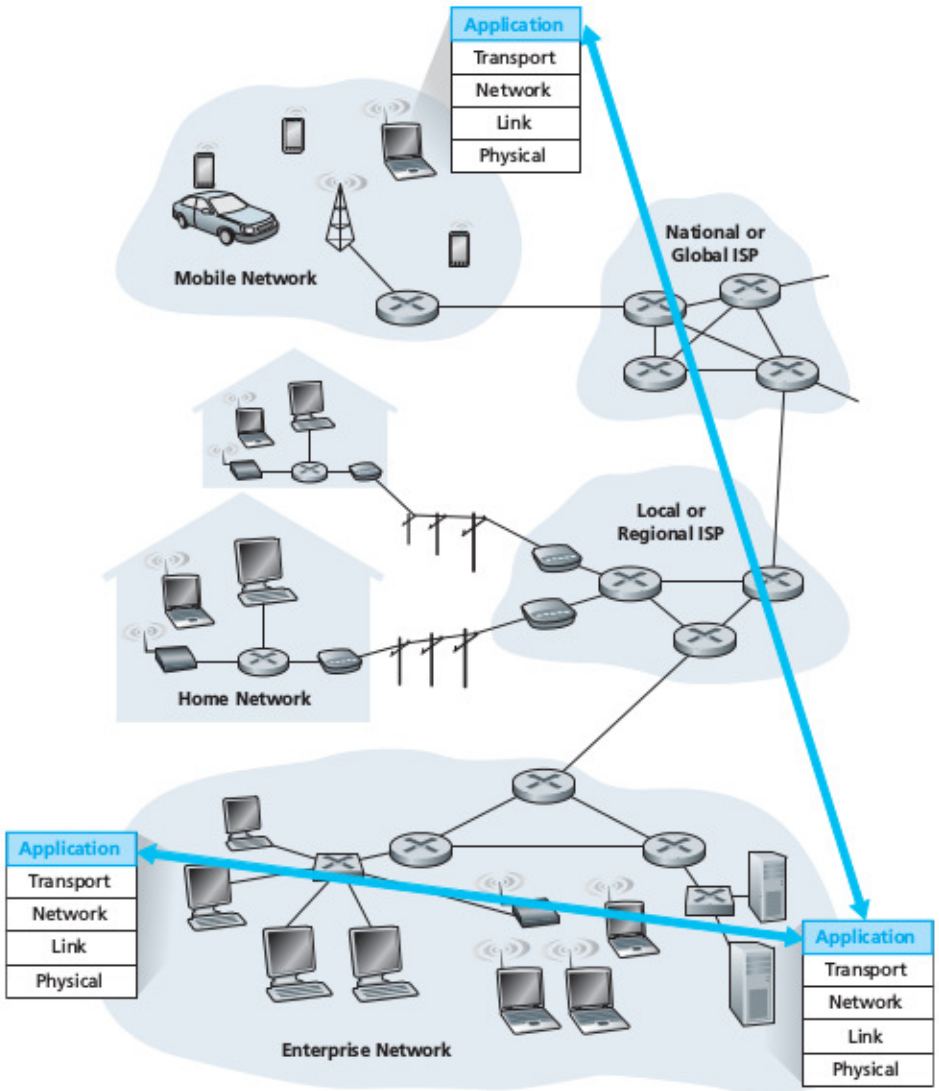
Princípio básico do desenvolvimento de aplicações de redes: escrever programas que executam em diferentes end systems, comunicando uns com os outro através da rede.

- Página da Web: browser (cliente) e o servidor (em alguma server farm do planeta).
- P2P: programa para cada host participante da comunidade de compartilhamento de arquivos. É mais difícil distinguir cliente e servidor.

Como são implementadas: linguagens mais comuns (C, Java, Ruby, Python), *firmware*, ...

Encapsulamento:

- Você não precisa se preocupar em escrever software que execute em ativos do núcleo da rede, como roteadores ou switches.
- Premissa: esses dispositivos proverão adequadamente os serviços originalmente propostos em suas respectivas camadas.

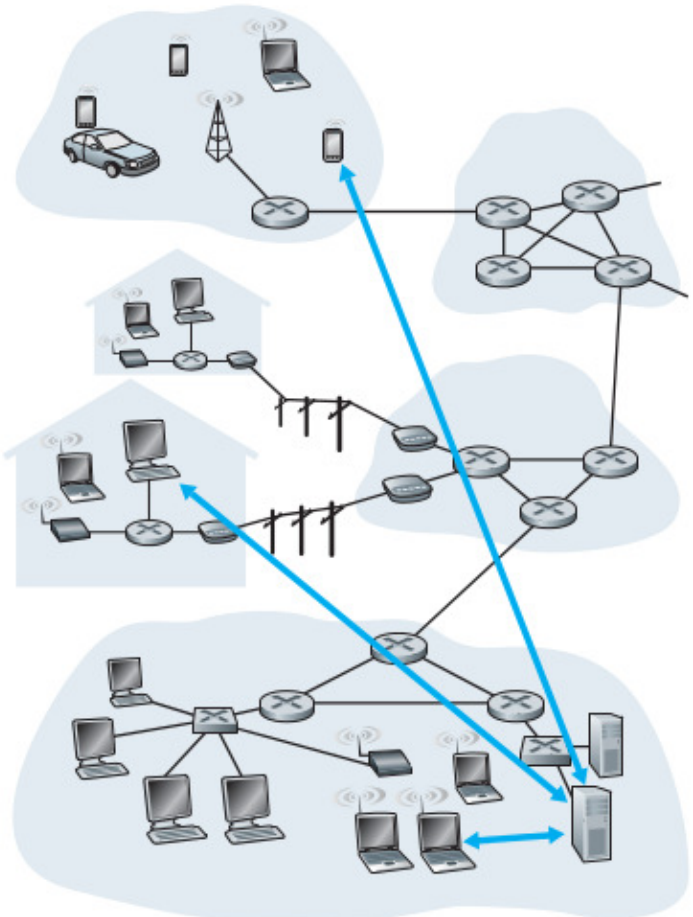

Application
Transport
Network
Link
Physical
Mobile Network
National or
Global ISP
Local or
Regional ISP
Home Network
Application
Transport
Network
Link
Physical
Enterprise Network
Application
Transport
Network
Link
Physical

Arquiteturas de aplicações de rede

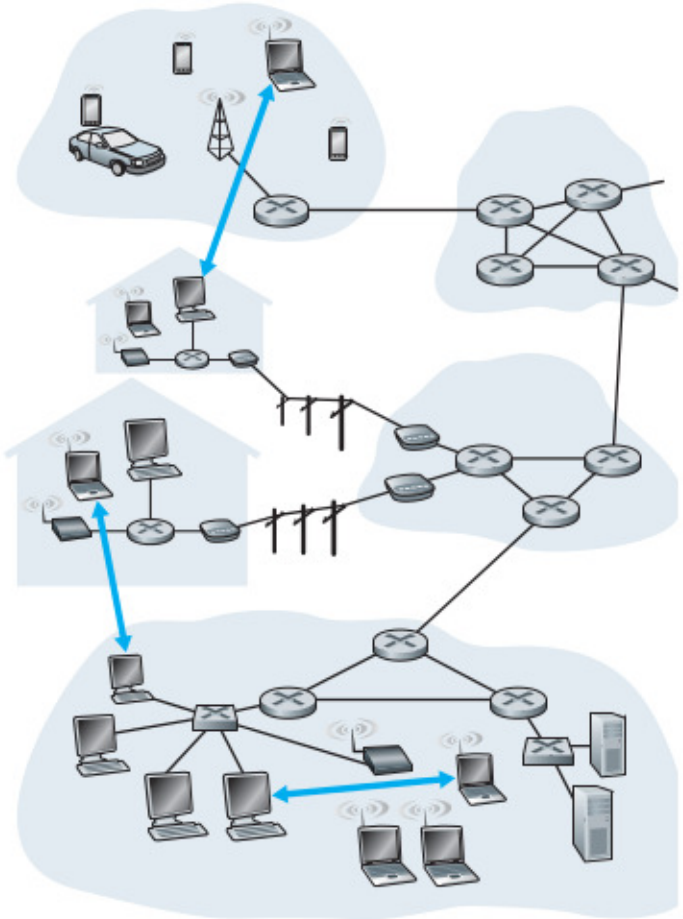
A arquitetura de uma aplicação não deve ser confundida com a arquitetura da rede.

- Esta é fixa é já provê um conjunto específicos de serviços às aplicações.
- A arquitetura de uma aplicação é projetada pelo desenvolvedor da aplicação e descreve como a aplicação é estruturada entre os diferentes end systems.

Paradigmas arquiteturais: cliente/servidor ou peer-to-peer.



a. Client-server architecture

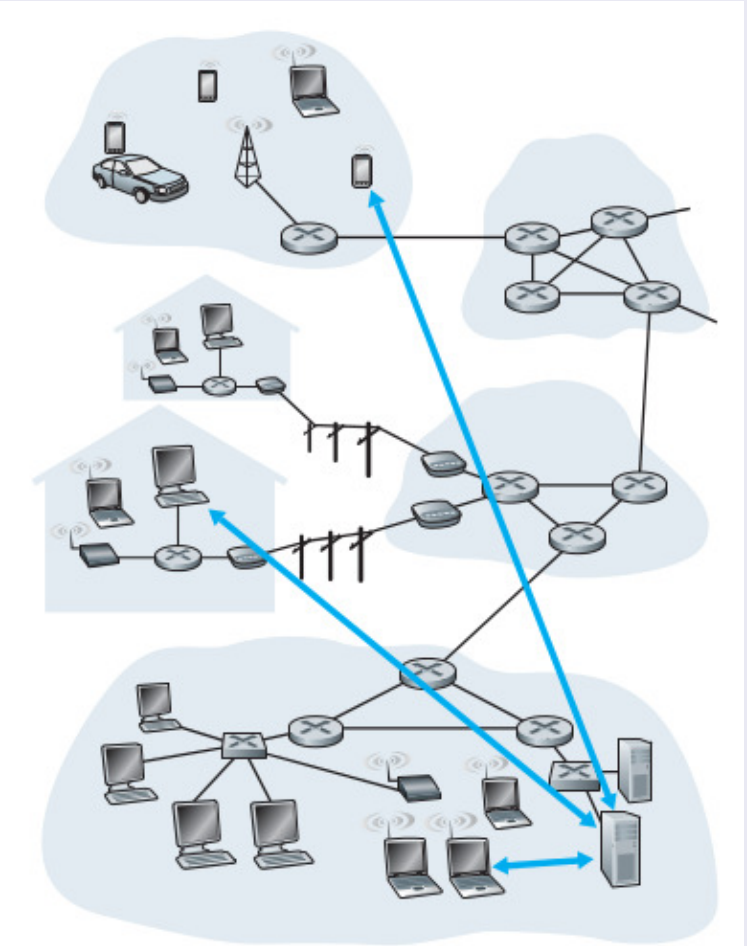


b. Peer-to-peer architecture

Cliente-servidor

Há sempre um host ativo, **o servidor**, que atende as requisições dos muitos outros hosts, **os clientes**.

- WWW: quando um servidor Web recebe requisições de browsers, responde enviando o *objeto* demandado para o host cliente.
- Os clientes não se comunicam diretamente entre si.
- Os servidores costumam possuir um endereço de IP (ou um conjunto deles) fixo.
- Exemplos: Web, FTP, SSH, Telnet, e-mail...

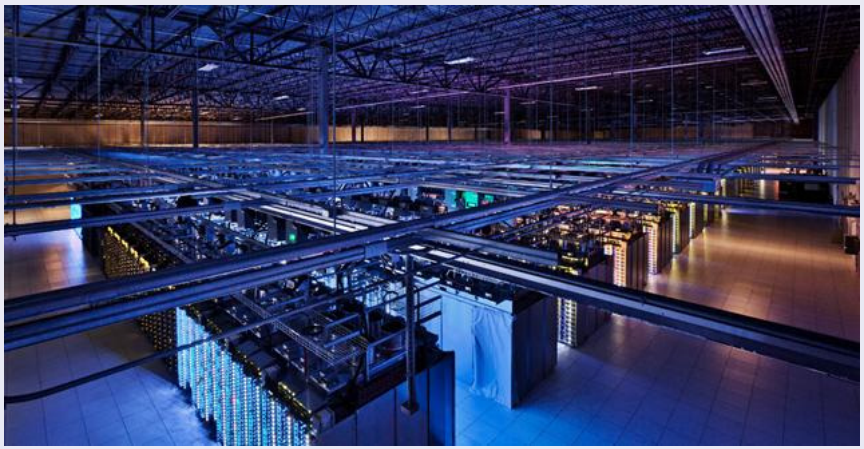


a. Client-server architecture

Cliente-servidor

Um **único** servidor pode ser incapaz de atender às requisições de seus clientes:

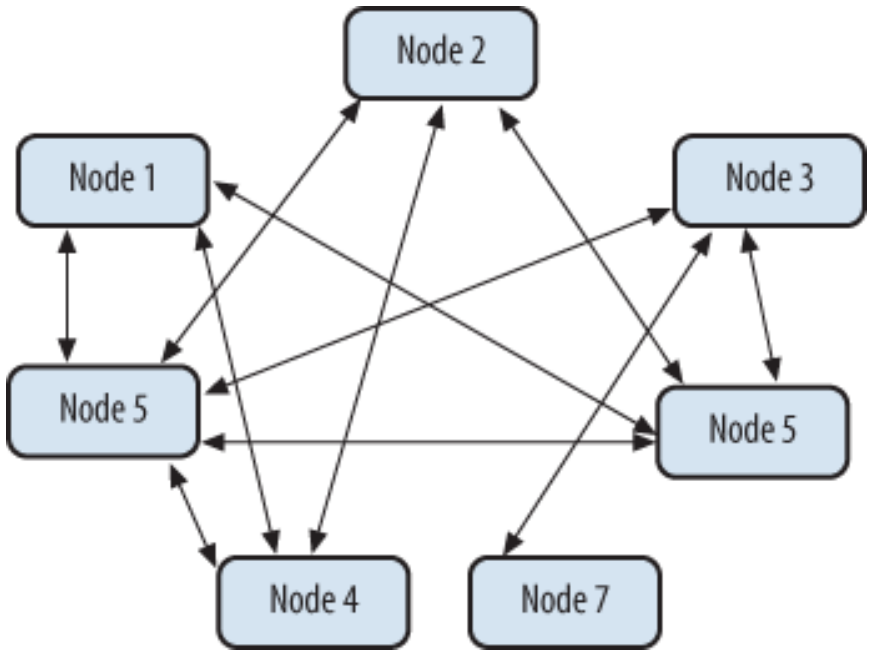
- Redes sociais: data centers (um servidor virtual poderoso).
- Os serviços mais conhecidos da Internet operam em data centers: Google, Bing, Amazon, e-Bay, Gmail, Yahoo Mail, Facebook, Twitter.
- Um data center pode agregar entre centenas e milhares de servidores, que devem ser alimentados (**energia** e **rede**) e mantidos.



P2P

Há dependência mínima (ou mesmo inexistente) em servidores providos por data centers. Em vez disso, aplicações exploram comunicações diretas entre os pares de hosts intermitentemente conectados: *peers*.

Exemplos: BitTorrent, Xunlei, Skype, Kankan e PPstream.



UnB Gama
O novo endereço da tecnologia

Vantagens

Auto escalabilidade: No caso de uma aplicação para compartilhamento de arquivos P2P, cada peer gera uma carga de trabalho através da requisição de arquivos, porém, cada peer é capaz, também, de prover capacidade de serviço ao sistema distribuindo arquivos a outros peers.

P2P: Desafios

- **Coexistência com ISPs**:
  - ISPs domésticos (acesso) costumam prover serviços de conectividade assimétricos.
  - Aplicações P2P para compartilhamento de arquivos e de vídeos costumam demandar enlaces simétricos: fator de estresse para ISPs de acesso.
- **Segurança**: garantir algumas premissas de segurança é muito complexo para um sistema distribuído.
- **Política de incentivos**: depende do compartilhamento de banda, armazenamento e de recursos de computação. Isso, também, apresenta interessantes desafios de projeto.

Arquiteturas Híbridas

Instant Messengers.

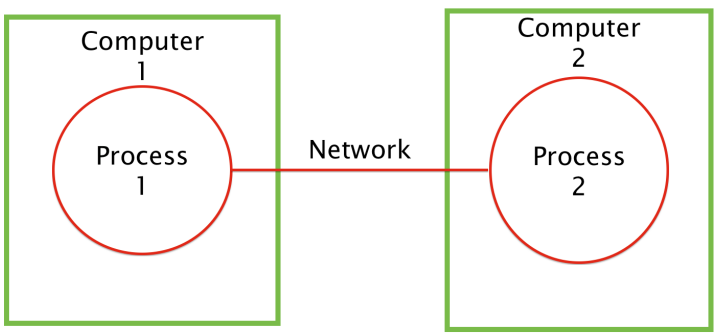
Comunicação entre processos

Um processo pode ser entendido como um programa que executa em um end system.

Se os processos comunicantes pertencem a um mesmo sistema, é possível usar de mecanismos de comunicação entre processos usando regras governadas pelo sistema operacional hospedeiro. **Esse não é o objetivo de nosso curso**.

Processos em diferentes hosts podem comunicar entre si pela troca de mensagens através da rede de computadores

- O processo emissor cria e envia mensagens pela rede.
- O processo receptor recebe essas mensagens e possivelmente responde enviando mensagens de resposta.
- Isso tudo acontece na camada de aplicação.



UnB Gama

Processos Cliente e Servidor

Uma aplicação de rede consiste em um par de processos que trocam mensagens entre si pela rede.

- Web: browser e servidor HTTP.
- P2P de compartilhamento de arquivos: um arquivo é transmitido de um processo de um peer para um processo em outro peer.

Em algumas aplicações, um processo pode ser tanto cliente como servidor ao mesmo tempo.

Definição

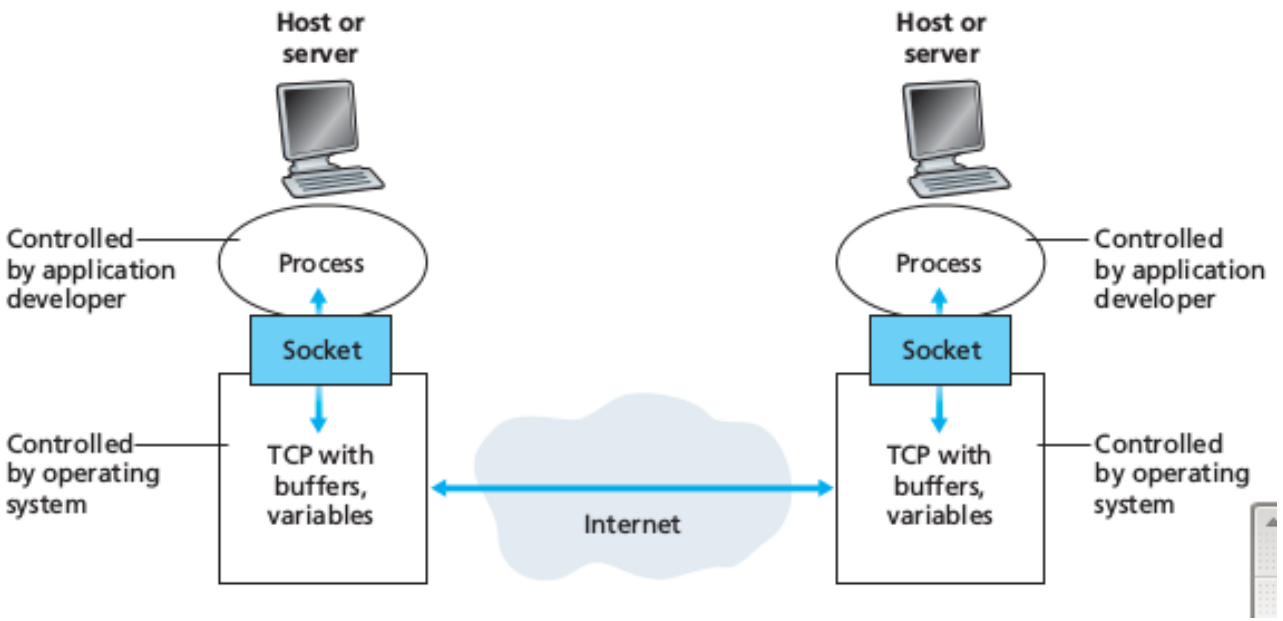
No contexto de uma sessão de comunicação entre um par de processos, **o processo que inicia a comunicação** (aquele que inicialmente contacta o outro processo no começo da sessão) é chamado de **cliente**.

O processo que espera por um contato que implicará o início de uma sessão é chamado de **servidor**.

Interface entre o Processo e a Rede de Computadores

Cada mensagem enviada por um processo deve atravessar a rede de interligação para alcançar o outro processo.

- Interface de software (API) usada para envio e recepção de um mensagens: *socket*.
- Analogias: casa e arquivo.

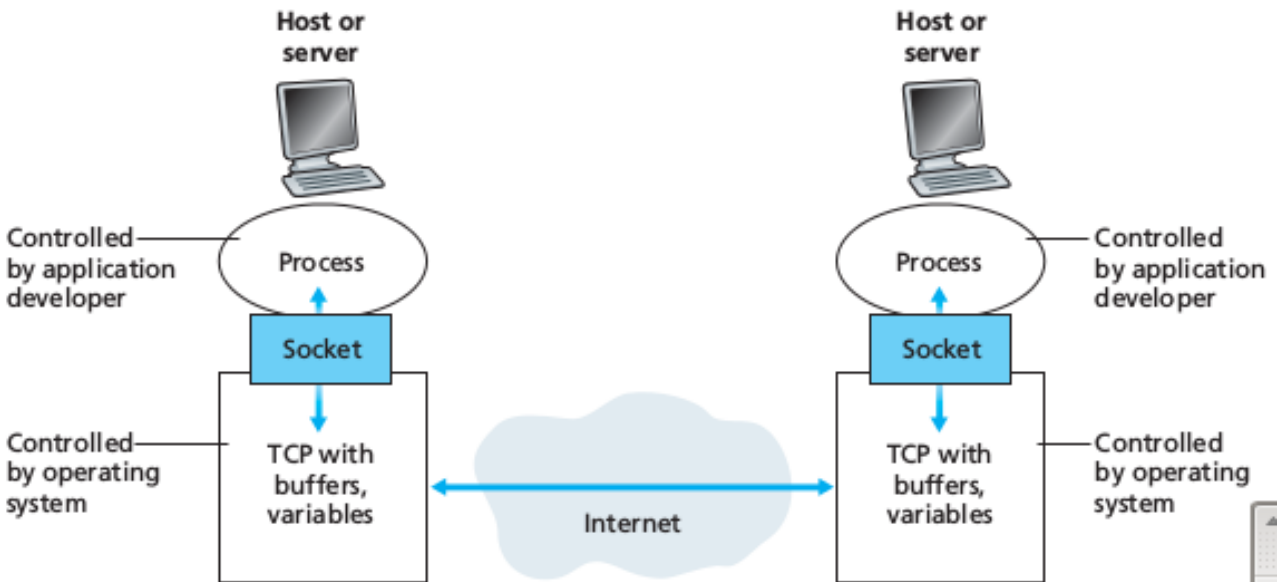


Socket (TCP)

O desenvolvedor da aplicação possui controle de tudo do lado da camada de aplicação do socket mas possui pouco controle sobre a camada de transporte a partir do socket.

Controles viáveis sobre a camada de transporte:

1. escolha do protocolo de transporte e
2. a habilidade de fixar alguns parâmetros da camada de transporte tais como tamanho máximo de buffer e tamanho máximo de segmento.



Endereçamento de processos

Para identificar um processo receptor, duas peças de informação devem ser determinadas: (1) endereço do host de destino e (2) o indentificador que especifica o processo receptor no host de destino

- Um host é identificado por um **endereço de IP**: Endereço de 32 bits
- Um processo é identificado por um número de **porta de destino**. Há uma lista de portas padrões previamente associadas a aplicações mais conhecidas: 80 - HTTP, 25 - SMTP. (/etc/services)

Serviços de transporte disponíveis a aplicações

Muitas redes, inclusive a Internet, provêem mais de um protocolo de camada de transporte.

Q: Como escolhê-los?

A: Depende dos requisitos das aplicações: escolha o protocolo com os serviços mais adequados para as necessidades de suas aplicações!

Requisito: Transmissão confiável de dados

Estamos tratando de redes de melhor esforço: pacotes podem ser perdidos na rede de computadores.

- **Overflow** no buffer de um roteador (**drops**), descarte no roteador ou host de destino no caso em que há dados corrompidos nos pacotes.
- Para algumas aplicações, a perda de pacotes pode ocasionar consequências devastadoras: aplicações financeiras.
- Se um protocolo é capaz de garantir um serviço de entrega de dados, ele é chamado de transmissão confiável.
- Um serviço importante que um protocolo de camada de transporte pode prover a uma aplicação é a **transmissão confiável** processo a processo.

Q: Há situações em que se aceitam perdas de pacotes?

A: Sim, há aplicações/serviços que toleram a perda de pacotes: aplicações multimídia

Requisito: Vazão (mínima)

Alguns protocolos podem ser capazes de prover **garantias** de vazão: throughput.

- aplicações sensíveis a largura de banda: aplicações de telefonia sobre IP, por exemplo
- aplicações *elásticas*: e-mail, FTP, HTTP.

Requisito: Temporização/Latência/Atraso (máximo)

Alguns protocolos podem prover garantias de temporização:

- **aplicações em tempo real**

Requisito: Segurança

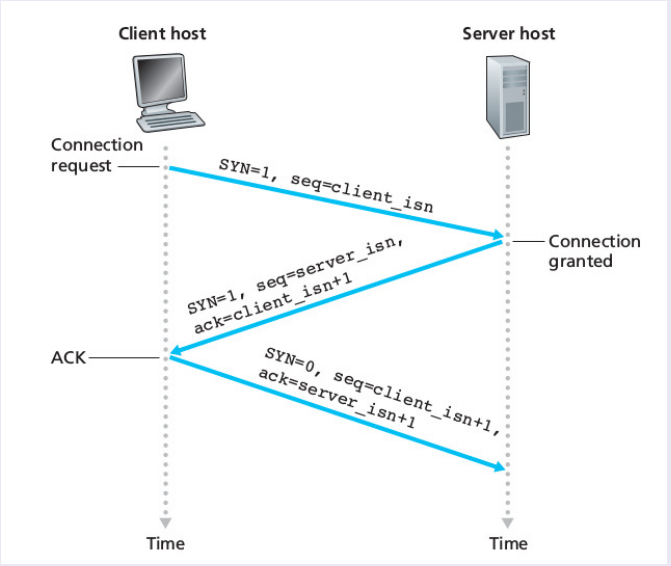
Contramedidas para garantia de: sigilo, integridade, autenticação (mútua ou não) nas comunicações.

Serviços de transporte providos pela Internet

TCP e UDP

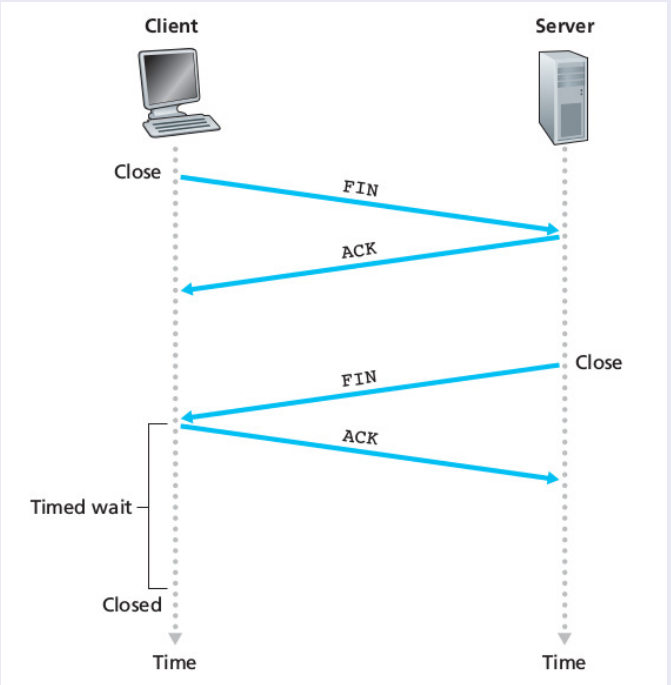
TCP: Serviço orientado a conexão

- TCP troca informações de controle da camada de transporte entre cliente e servidor antes das mensagens da camada de aplicação começarem a serem trocadas $\Longrightarrow$ handshaking.
- Depois do handshaking, uma conexão TCP é dita **estabelecida** entre os sockets dos dois processos.



TCP: Serviço orientado a conexão

- As conexões são **full-duplex**.
- Quando a aplicação se encerra, deve ocorrer o fechamento adequado da conexão.



TCP: Serviço de transmissão confiável de dados

- Os processos que utilizam o TCP podem confiar que todos os dados serão enviados sem erros e na ordem correta.
- TCP também inclui um mecanismo de controle de congestionamento:
  - modula o processo emissor quando se percebe que a rede entre o emissor e o receptor passa por congestionamento.
  - tenta limitar cada conexão TCP à sua **partilha justa da banda passante disponível da rede**.

UDP

- Protocolo sem conexão: não há handshaking entre os dois processos para que se inicie uma conexão.
- Protocolo não confiável: melhor esforço $\Longrightarrow$ mensagens podem não chegar ou mesmo chegar fora de ordem
- Não possui mecanismo de controle de congestionamento: flooding!

Serviços _não_ providos pelos protocolos da camada de transporte da Internet

- Garantias de temporização e vazão.
- Pacotes UDP sendo filtrados por *firewalls*: aplicações de telefonia por Internet passam a usar TCP como protocolo de contingência.

Protocolos de camada de aplicação

Definem como processos de uma aplicação, cada um executando em diferentes end systems, trocam mensagens entre si:

- **o tipo** de mensagens trocadas: requisições e respostas;
- **sintaxe** dos vários tipos de mensagens: campos das mensagens e como os campos são delimitados/preenchidos;
- **a semântica** dos campos;
- **regras** para determinar quando e como um processo **envia** mensagens e **responde** a mensagens: máquina de estados.

Protocolos de camada de aplicação

Alguns protocolos de camada de aplicação são especificados em RFCs: HTTP.

Há protocolos que seguem padrões proprietários: Skype.

Distinção entre aplicações de rede e protocolos de camada de aplicação:

- um protocolo de camada de aplicação é apenas uma peça do quebra-cabeças de uma aplicação de rede.

Web

Composição:

- formato padrão para documentos (HTML);
- Web browsers, Web servers;
- protocolo de camada de aplicação (HTTP).

E-mail

Composição:

- servidor de e-mail (mailboxes),
- clientes de e-mail,
- padrões que definem a estrutura de uma mensagem de e-mail,
- protocolo de camada de aplicação que define como as mensagens são trocadas entre servidores (SMTP), como as mensagens são trocadas entre clientes e servidores (POP3, IMAP) e como os conteúdos dos cabeçalhos das mensagens podem ser interpretados.

Protocolos de Camada de Aplicação a serem estudados nessa disciplina

- Web,
- FTP,
- E-mail,
- DNS.

World Wide Web: Bernes-Lee, 1994

- Primórdios da Internet: e-mail, terminais remotos, transferência de arquivos, publicação de notícias ...
- Quebra de paradigma WWW: em vez de receber passivamente conteúdo em suas casas (sistema de difusão como rádio ou TV), a Internet permite agora que os usuários escolham o conteúdo que desejam consumir.
  - Ficou fácil publicar conteúdos (na Internet): baixo custo!
  - Hyperlinks conduziam a navegação entre os objetos publicados e viabilizavam a indexação de um volume enorme de conteúdo.
  - Gráficos, animações, vídeos, áudios passam a compor o documento digital!

Visão geral do HTTP

- HTTP é o protocolo da camada de aplicação considerado o coração da Web. Definido pelas RFC 1945 e 2616.
- Cliente e servidor conversam através da troca de mensagens HTTP.
- HTTP define a estrutura dessas mensagens e como o cliente e o servidor trocam essas mensagens.

Definições

- Página da Web: documento, agrupamento de objetos.
- **Objeto**: arquivo HTML, JPEG, Applet Java, video, arquivo. Qualquer elemento capaz de ser endereçado através de um URL: Uniform Resource Locator.
- Cada URL possui dois componentes: hostname e o caminho para um objeto.

URL

http://www.unb.br/noticias/galeria/images/...

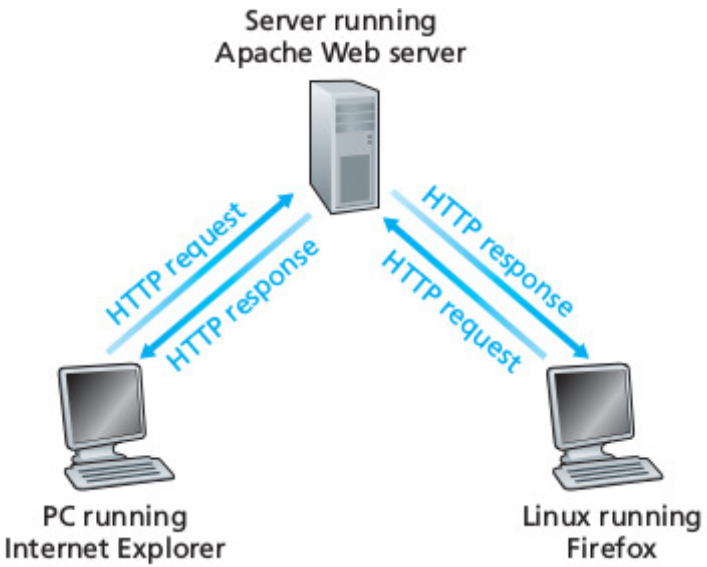
... INSTITUCIONAL/PARCERIA/2015/Jul/17/antuerpia/thumbs/pqantuerpia1.jpg

Definições

- Web browsers: clientes.
- Web servers: Apache, IIS, Nginx, ...

HTTP usa TCP como protocolo de transporte

- Usa a interface socket para interagir com a camada de transporte.
- O protocolo TCP é responsável por se recuperar de perdas de pacotes e reordenamento de dados nas camadas mais baixas.
- O servidor envia os objetos requisitados pelos clientes sem, necessariamente, armazenar qualquer informação de estado sobre o cliente: requisições repetidas a um mesmo objeto gerarão mensagens diferentes em atendimento às demandas.
- Protocolo **sem estado**!



Conexões persistentes e não persistentes

Persistente: envia todos os pedidos através de uma mesma conexão TCP estabelecida

HTTP possui conexões persistentes como modo padrão.

Conexões não persistentes

Usuário requisita a seguinte URL no seu browser:

http://www.someSchool.edu/someDepartment/home.index

1. O cliente HTTP inicia uma conexão TCP com o servidor www.someSchool.edu e na porta 80 (padrão http). Associado à conexão, haverá um socket no cliente e no servidor
2. O cliente HTTP envia uma mensagem de requisição HTTP ao servidor através de seu socket. A mensagem de requisição inclui o caminho do recurso someDepartment/home.index
3. O processo do servidor HTTP recebe a mensagem de requisição por seu socket, busca pelo objeto someDepartment/home.index em sua memória de persistência, encapsula o objeto em uma resposta HTTP e envia a mensagem de resposta ao cliente através de um socket

Conexões não persistentes

Usuário requisita a seguinte URL no seu browser:

http://www.someSchool.edu/someDepartment/home.index

4. O processo do servidor HTTP diz ao TCP que a conexão deve ser encerrada. (Isso acontecerá, de fato, apenas quando o cliente confirmar a recepção adequada da mensagem de resposta)
5. O cliente HTTP recebe a mensagem de resposta. A conexão TCP se encerra. A mensagem indica que o objeto encapsulado é um arquivo HTML. O cliente extrai o arquivo da mensagem de resposta, examina o arquivo HTML e encontra referências a 10 objetos JPEG.
6. Os primeiro quatro passos acima serão repetidos para cada um dos JPEGs referenciados no arquivo HTML.

Conexões não persistentes

Usuário requisita a seguinte URL no seu browser:

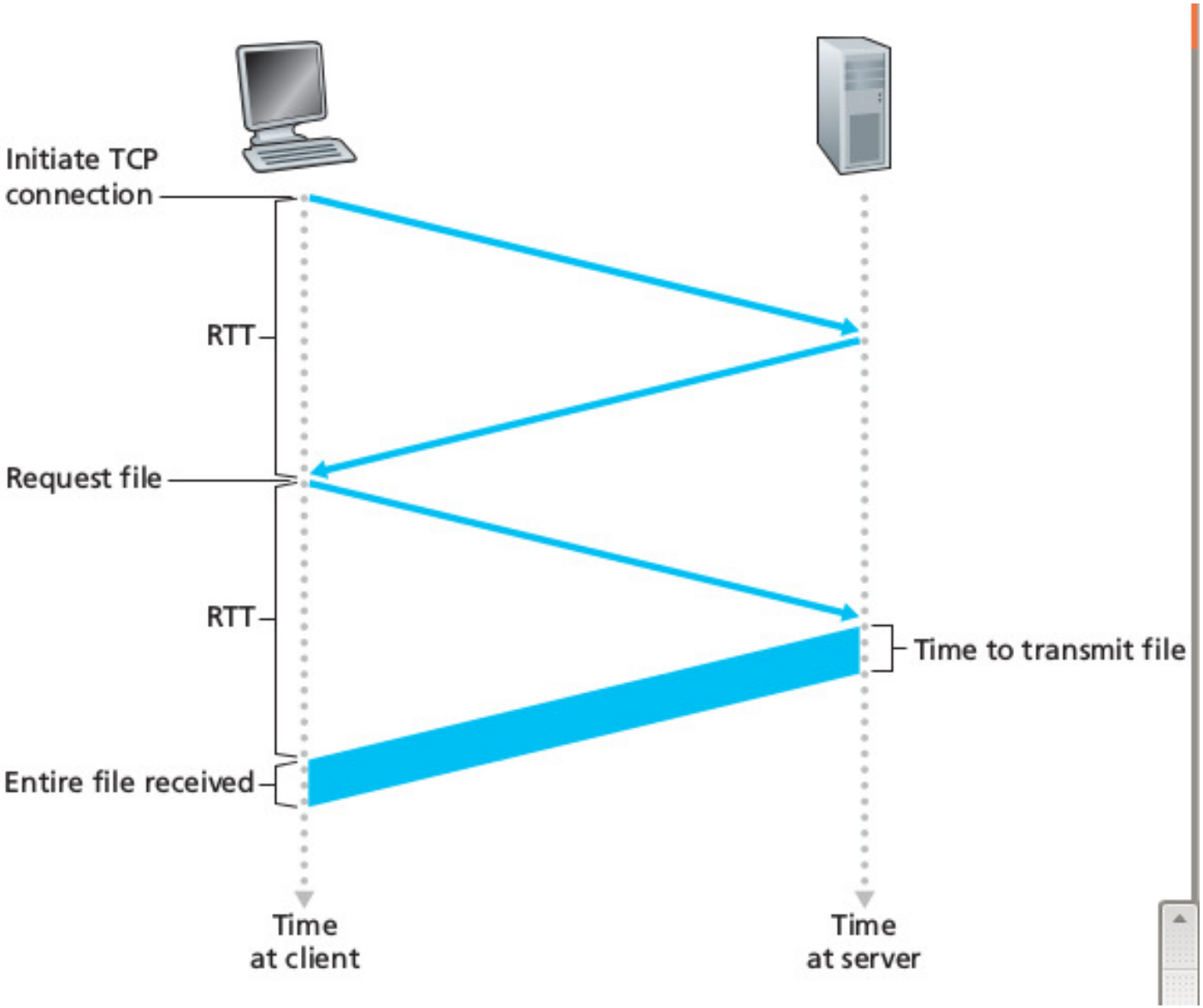
http://www.someSchool.edu/someDepartment/home.index

- Assim que o browser recebe a página da Web, ele a disponibiliza ao usuário.
- Browsers diferentes podem interpretar de formas diferentes a página da Web: Isso não é uma questão a ser tratada pelo protocolo HTTP: o protocolo HTTP define apenas o protocolo de camada de aplicação entre o programa HTTP cliente e o programa HTTP servidor.
- Os passos acima descritos ilustram o uso de conexões não persistentes: são estabelecidas conexões TCP diferentes para o tratamento de cada um dos objetos envolvidos na comunicação
- **Há a possibilidade de paralelismo**

RTT: Round-Trip Time

Definição: tempo levado por um pequeno pacote viajar de um cliente para um servidor e, em seguida, voltar para o cliente.

RTT inclui atrasos da propagação do pacote, enfileiramento do pacote e do processamento do pacote.


Initiate TCP connection
RTT
Request file
RTT
Time to transmit file
Entire file received
Time at client
Time at server

HTTP com conexões persistentes

Persistente: envia todos os pedidos através de uma mesma conexão TCP estabelecida
$\Longrightarrow$ HTTP possui conexões persistentes como modo padrão.

Problemas com conexões não persistentes:

- Nova conexão TCP para cada objeto demandado.
- Atrasos de 2 RTTs para a recepção de cada um dos objetos demandado.

Configurações do servidor HTTP: timeout para encerramento de conexão (preferência por pipelining).

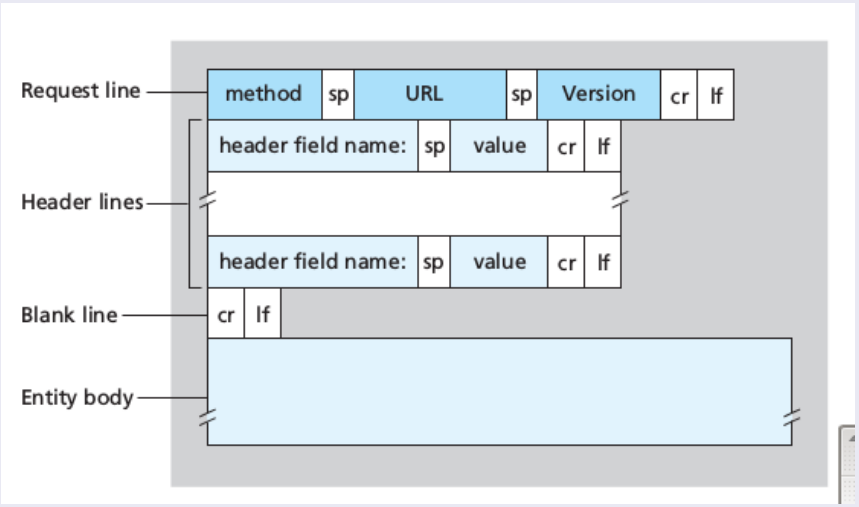
Formato de mensagens HTTP

HTTP request:

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

Formato de mensagens HTTP: HTTP request

- Mensagem codificada em ASCII, 5 linhas separadas por CRLF, última linha seguida por CRLF.
- Primeira linha: linha de requisição
  - Método: GET, POST, HEAD, PUT e DELETE
  - URL
  - campo de versão HTTP
- Outras linhas: cabeçalho
  - **host**: especifica o host em que o objeto reside. Necessárias a proxy caches.
  - **Connection**: close $\Longrightarrow$ não há necessidade de conexão persistente
  - **User-agent**: informações sobre o tipo de browser. Servidor capaz de prover conteúdos baseados no browser que apresenta a requisição
  - **Accept-language**: apenas um dos muitos cabeçalhos de negociação de conteúdo disponibilizados no HTTP



Formato de mensagens HTTP: HTTP request

- Entity body: vazio com GET, mas pode vir preenchido com outros métodos (POST).
- É possível enviar informações usando o método GET: www.somesite.com/animalsearch?monkeys&bananas
- Método HEAD, similar a método GET: recebe resposta, mas servidor não envia objeto solicitado. Usado em depuração!
- Método PUT: upload de objetos nos web servers
- Método DELETE: apagamento de objetos nos web servers

Formato de mensagens HTTP

HTTP response:

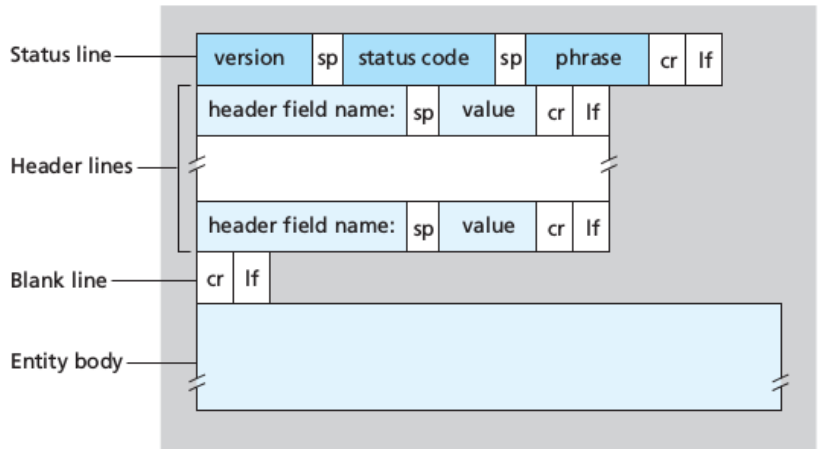
```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data ...)
```

Formato de mensagens HTTP: HTTP response

Dividido em 3 seções: Linha de estado, Linhas de cabeçalho e Entity Body.

Linha de estado:

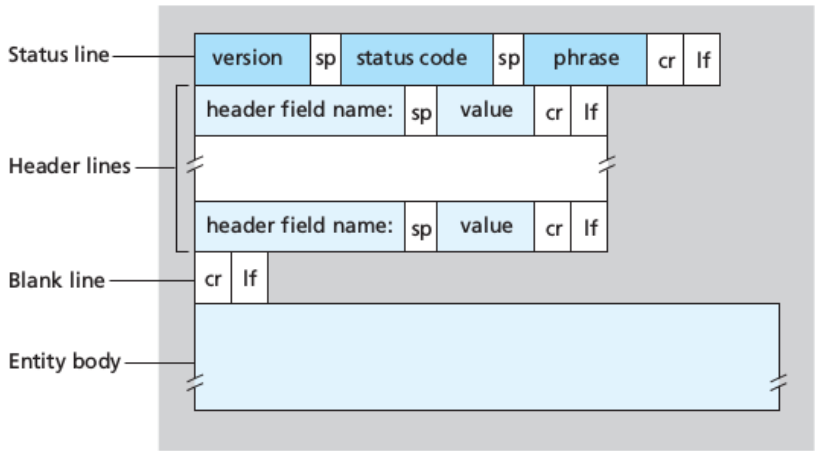
- campo de versão do protocolo: HTTP/1.1
- código de estado: 200
- mensagem de estado: OK



Formato de mensagens HTTP: HTTP response

Linhas de cabeçalho:

- Connection: close, o cliente irá fechar a conexão TCP depois de enviar a mensagem.
- Date: indica a data e a hora em que a resposta HTTP foi criada e enviada pelo servidor
- Server: indica o servidor usado para prover o serviço HTTP

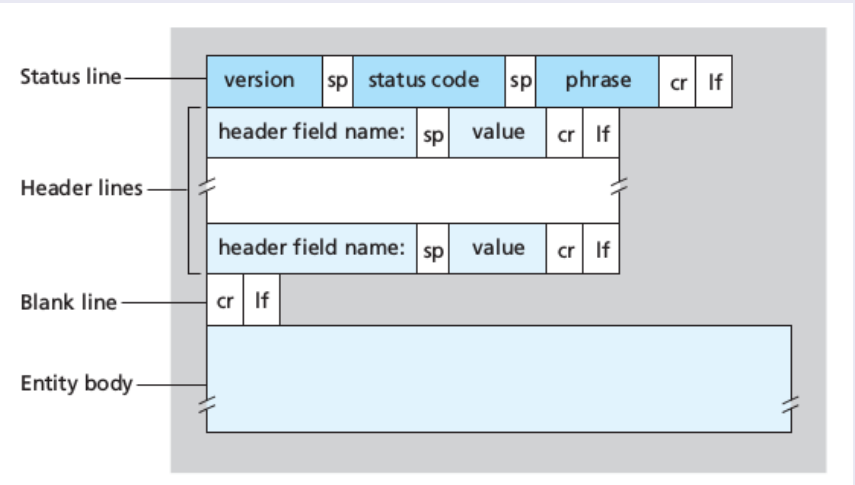


Formato de mensagens HTTP: HTTP response

Linhas de cabeçalho:

- Last-Modified: indica a data e a hora em que o objeto foi criado ou modificado pela última vez $\Longrightarrow$Crítico para cacheamento de objetos (lado cliente e lado servidor, proxy)
- Content-Length: indica o número de bytes demandados para representar o objeto que está sendo enviado
- Content-Type: indica o tipo de objeto que será encaminhado. Nesse caso, um texto HTML

Entity Body: Conteúdo da mensagem, contém o objeto requisitado (data data data ...)



Formato de mensagens HTTP

Código de estado:

- 200 OK
- 301 Moved Permanently: o objeto requisitado foi permanentemente movido e a nova URL em que o mesmo se encontra está especificada no campo Location: da mensagem de resposta. Um cliente devidamente implementado deverá obter automaticamente a nova URL
- 400 Bad Request: erro genérico que indica que a requisição não pode ser entendida pelo servidor
- 404 Not Found: documento não existe no servidor
- 505 HTTP Version Not Supported: a versão de protocolo especificada na requisição não é suportada pelo servidor

Hands on

```
telnet URL 80
GET /~some_path HTTP/1.1
Host URL
```

- trocar GET por HEAD
- trocar /~some_path por /~another_path

Cookies

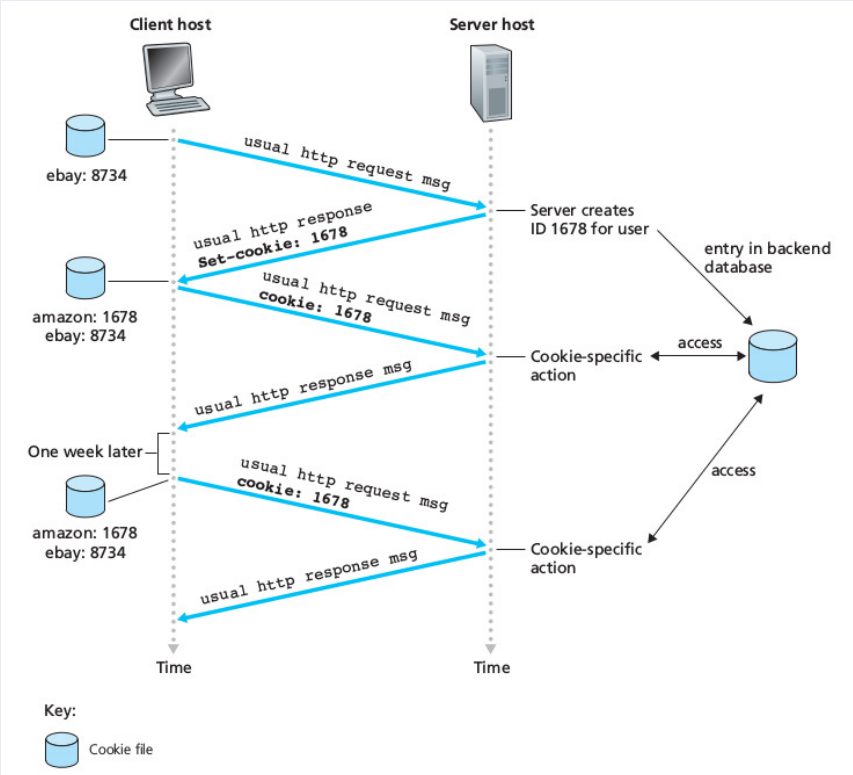
O protocolo HTTP foi declarado **sem estado**. Isso simplifica o projeto dos servidores e permitiu engenheiros construirem servidores Web de alto desempenho e capazes de atender milhares de conexões TCP simultâneas

Q: Como identificar um usuário? (restringir acesso, autorização, carrinho de compras?)

Cookies

A: Cookies: RFC 6263.

- linha de cabeçalho cookie na mensagem de resposta: Set-cookie: XXXX
- linha de cabeçalho cookie na mensagem de requisição. Depois de criada entrada cookie no arquivo: Cookie: XXXX
- arquivo cookie mantido no host do cliente e gerenciado pelo browser (aplicação) do cliente
- banco de dados de back-end no servidor.



Cookies

Cookies permitem o provimento de serviços como carrinho de compras.

- Browsers mantém os cookies em persistência por um longo período de tempo. A partir de cookies, webstores podem indicar produtos ajustados ao perfil de seu usuário, por exemplo.
- É possível identificar univocamente o usuário, indicando páginas personalizadas com o nome do indivíduo
- Permitem criar uma **camada de sessão (OSI)** a partir da aplicação HTTP (que não possui estado).
- São questionados em função da quebra de privacidade: Cookie Central.

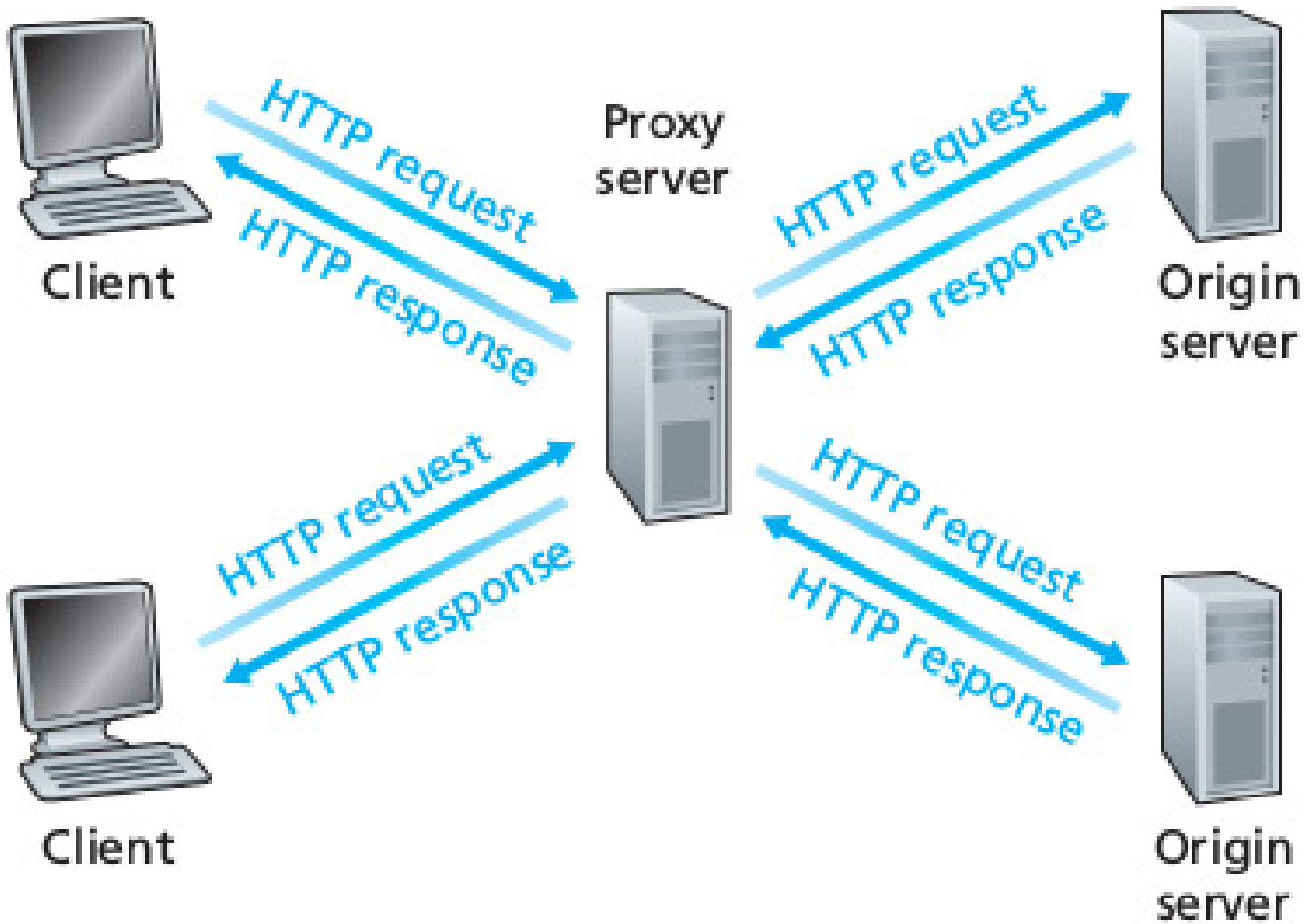
Web Caching

Proxy: intermediário de requisições Web.

É possível configurar browsers para requisitarem objetos inicialmente a um Web cache.

1. O browser estabelece uma conexão TCP com o cache Web e envia o pedido HTTP ao cache
2. O cache da Web verifica se possui uma cópia do objeto requisitado armazenada localmente. Se a possui, retorna uma mensagem com o objeto demandado ao cliente
3. Se o cache da Web não possui o objeto, o cache da Web abre uma conexão TCP com o servidor de origem (que provê, de fato, o objeto). O Web cache envia uma requisição HTTP pelo objeto através da conexão aberta.
4. Quando o Web cache recebe o objeto, o mesmo é armazenado localmente (é feita uma cópia local) e, também, é encaminhada uma cópia à mensagem de resposta destinada ao cliente que demandou o objeto.

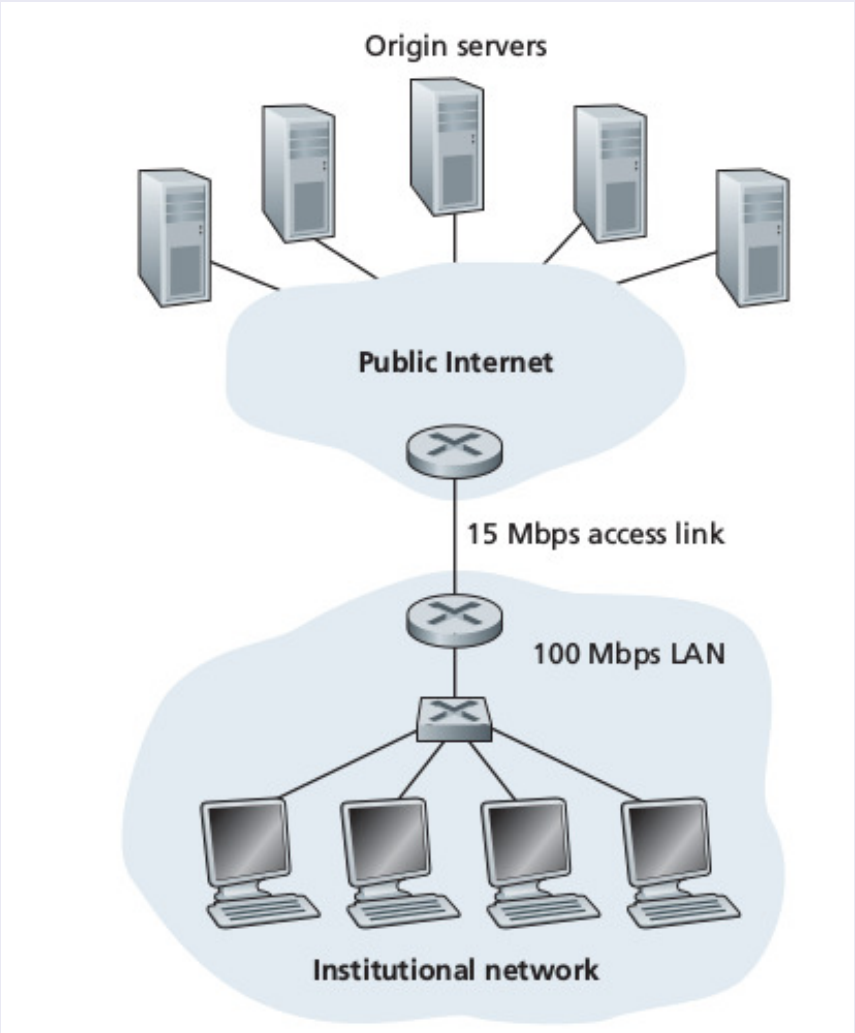
Note que o proxy é cliente e servidor ao mesmo tempo!

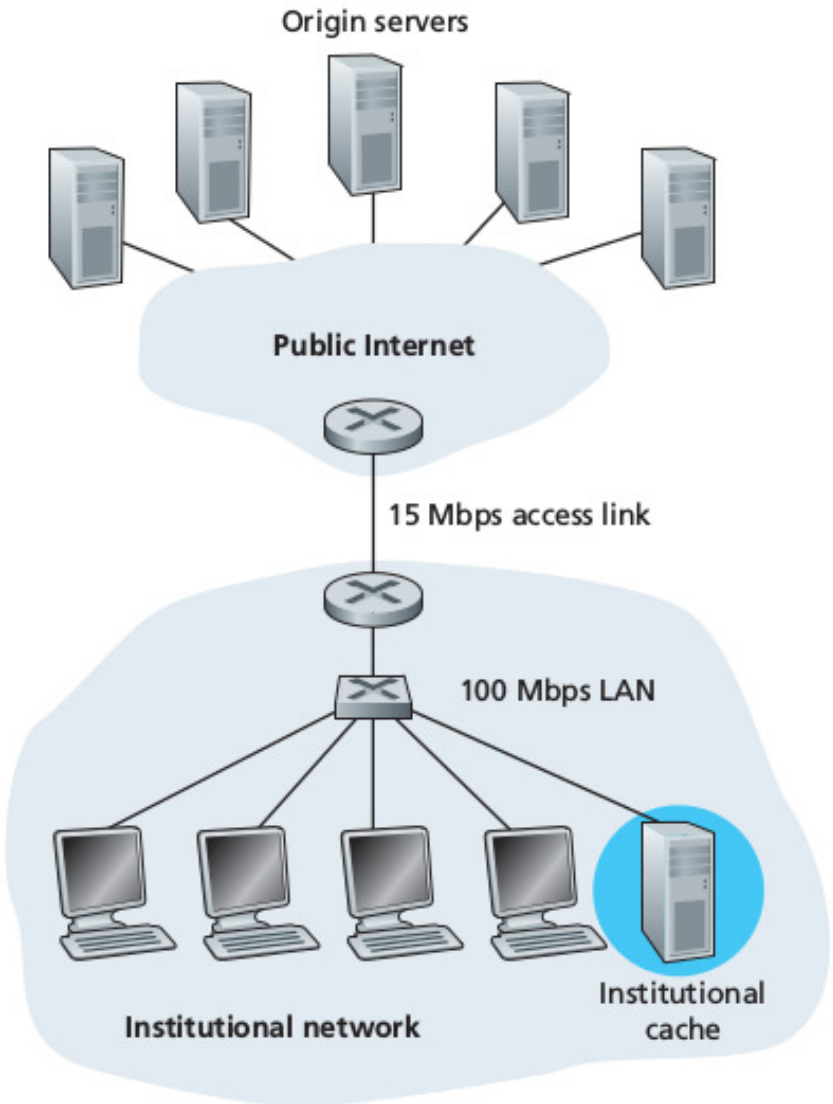

Client
HTTP request
HTTP response
Proxy server
HTTP request
HTTP response
Origin server
Client
HTTP request
HTTP response
HTTP request
HTTP response
Origin server

Web Caching

Um Web cache pode servir para economizar largura de banda e reduzir o tempo de resposta de requisições.

Cenário: enlace de saída possui uma largura de banda menor (**gargalo**) que o enlace que liga o cliente ao Web cache.




Origin servers
Public Internet
15 Mbps access link
100 Mbps LAN
Institutional network
Institutional cache

GET condicional

O que acontece se o objeto em cache não estiver na mesma versão que no sítio provedor? (Stale)

HTTP tem um mecanismo que permite ao cache verificar se os objetos em cache estão atualizados: GET condicional:

1. a requisição da mensagem usa método GET
2. a mensagem de requisição inclui uma linha de cabeçalho **If-Modified-Since**:

GET condicional

Requisição:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
```

Resposta:

```
HTTP/1.1 200 OK
Date: Sat, 8 Oct 2011 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 7 Sep 2011 09:23:24
Content-Type: image/gif

(data data data data data ...)
```

GET condicional

Requisição, uma semana depois através de um Web cache:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 7 Sep 2011 09:23:24
```

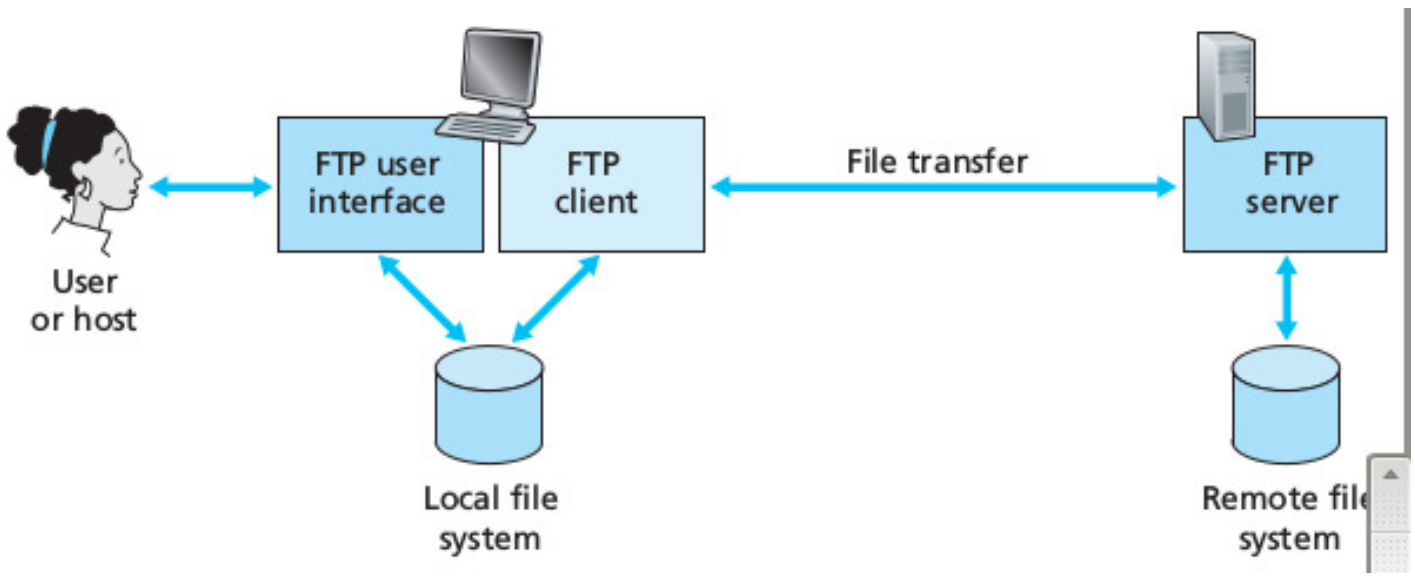
Resposta, se não houver modificação:

```
HTTP/1.1 304 Not Modified
Date: Sat, 15 Oct 2011 15:39:29
Server: Apache/1.3.0 (Unix)
(empty entity body)
```

Introdução

Usuário e servidores são equipamentos (hosts) diferentes que desejam transferir arquivos entre si:

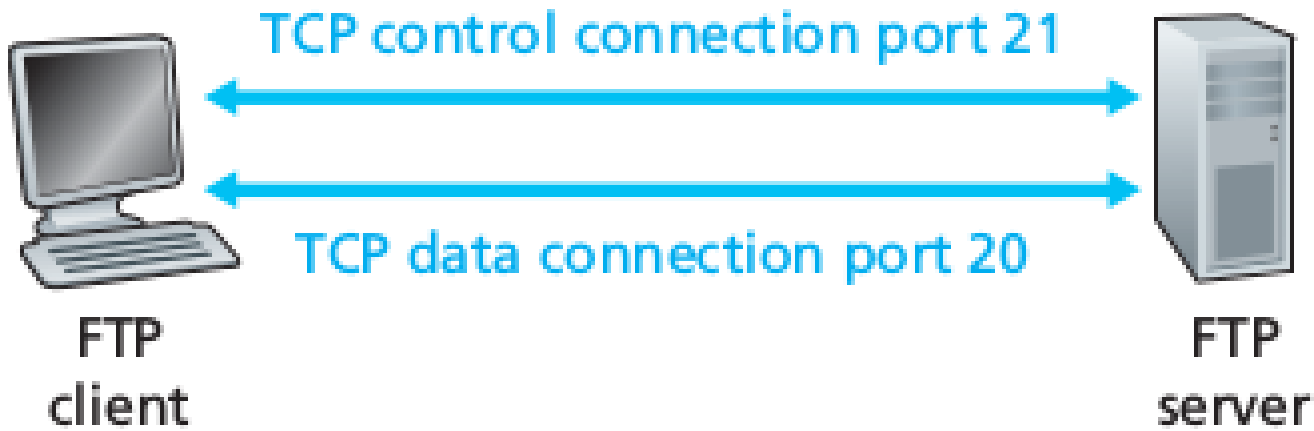
- Usuário deve passar por um processo de autorização: identificação e senha.
- Desse ponto em diante, o usuário pode enviar arquivos para o servidor ou baixar arquivos do servidor
- O protocolo de transporte é TCP



Introdução

FTP usa duas conexões:

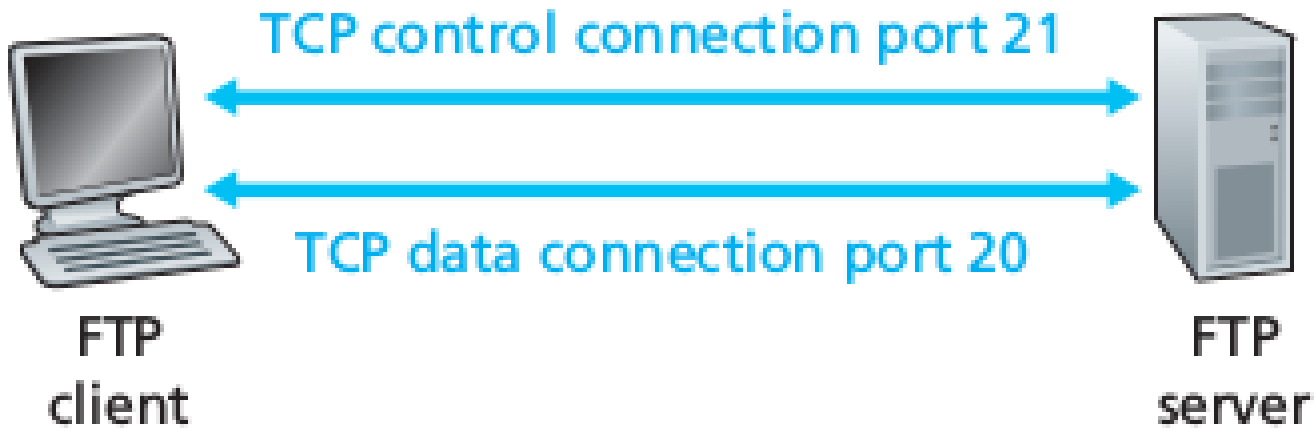
- controle: usada para enviar informações de controle entre os dois hosts (identificação de usuário, senha, comandos para sistemas de arquivos, put e get): controle fora de banda (out-of-band). HTTP in-band.
- dados: usada para transmissão dos arquivos



FTP (RFC 959)

Fluxo de conexão

1. Usuário inicia conexão TCP de controle com o servidor na porta 21.
2. O host cliente envia identificação de usuário e senha através da conexão de controle. Conexão de controle receberá, também, comandos típicos para transações sobre sistemas de arquivos
3. Um comando de transferência de arquivo despachado pelo cliente e recebido pelo servidor, implica no servidor o início de uma conexão TCP de dados para o cliente.
   - FTP envia apenas um arquivo em cada conexão de dados e fecha a conexão de dados. Conexões não persistentes.
   - A conexão de controle permanece aberta pela duração da sessão de usuário.



Fluxo de conexão

O servidor FTP deve manter o estado do usuário.

Isso restringe significativamente o número total de conexões que um servidor típico FTP é capaz de manter simultaneamente.

Comandos FTP e respostas

Comandos são emitidos pelo cliente ao servidor.

Respostas, do servidor ao cliente.

A conexão de controle é codificada em formato ASCII de 7 bits

- São comandos perfeitamente legíveis por humanos.
- Comandos separados por CRLF
- Cada comando consiste de 4 caracteres ASCII em caixa alta e alguns argurmentos (opcionais).

Comandos FTP e respostas

Comandos

- USER username: usado para enviar a identificação do usuário ao servidor.
- PASS password: usado para enviar a senha do usuário ao servidor
- LIST: usado para enviar solicitar ao servidor a lista de todos os arquivos contidos pelo diretório atual. A lista de arquivos é enviada através de uma nova conexão de dados (não persistente), em vez de usar a conexão TCP de controle
- RETR filename: usado para demandar (get) um arquivo contido no diretório atual do servidor. Cria uma nova conexão de dados para enviar o arquivo solicitado.
- STOR filename: usado para gravar (put) um arquivo no diretório atual do servidor (ou host remoto).

Comandos FTP e respostas

Respostas: Números de 3 dígitos, com mensagem opcional seguindo o número. Similar à estrutura usada nos códigos e linhas de estado do HTTP

Algumas respostas típicas

- 331 Username OK, password required.
- 125 Data connection already open; transfer starting
- 425 Can't open data connection
- 452 Error writing file

Introdução

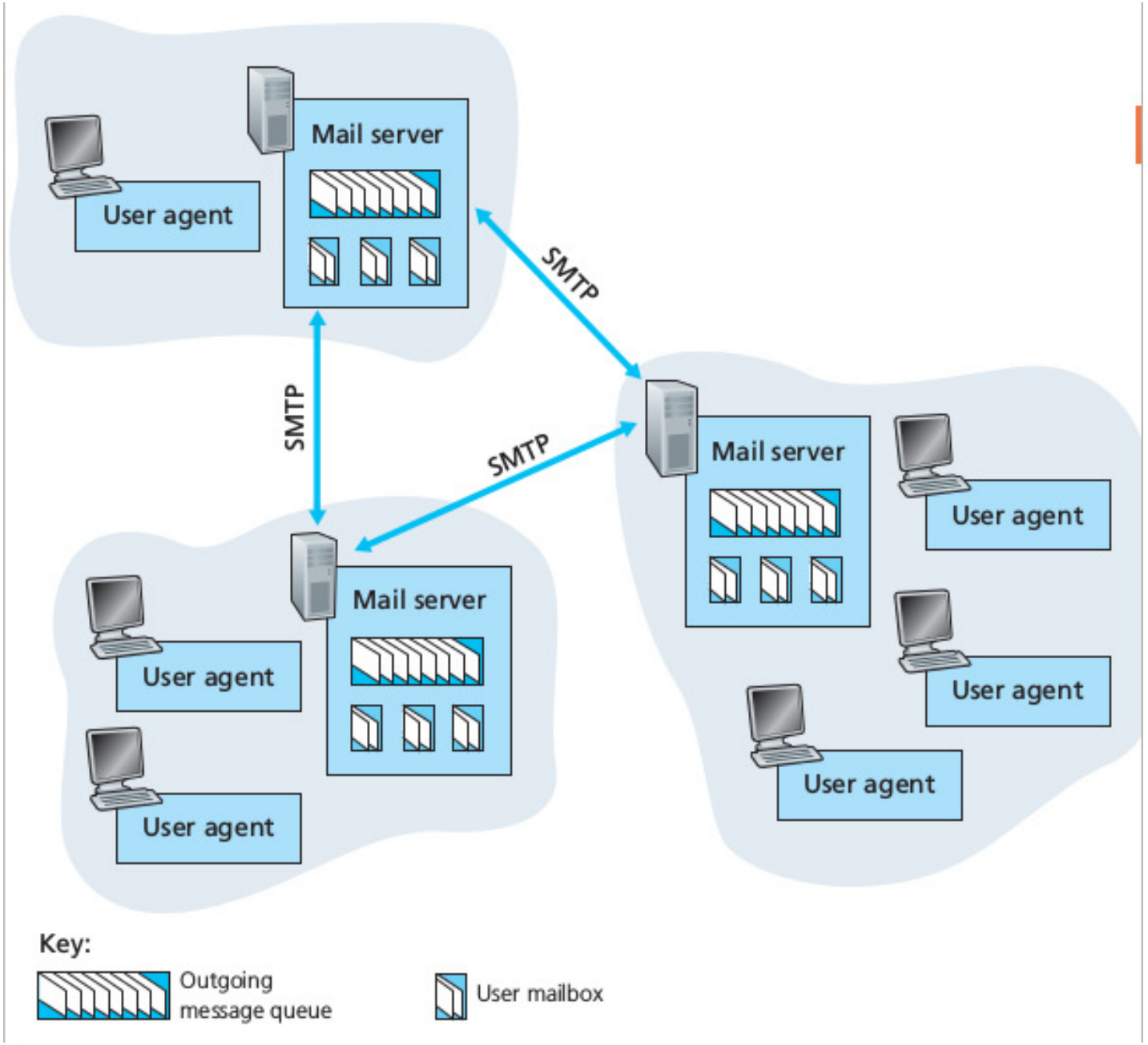
Já foi a aplicação mais popular quando a Internet estava em sua infância.

Permanece como uma das mais importantes e utilizadas aplicações

E-mail é um meio de comunicação assíncrono: pessoas enviam e lêem mensagens quando é conveniente:

- É rápido, fácil de distribuir e barato.
- Possui recursos modernos: mensagens com anexos, hyperlinks, textos formatados em HTML e fotos embutidas

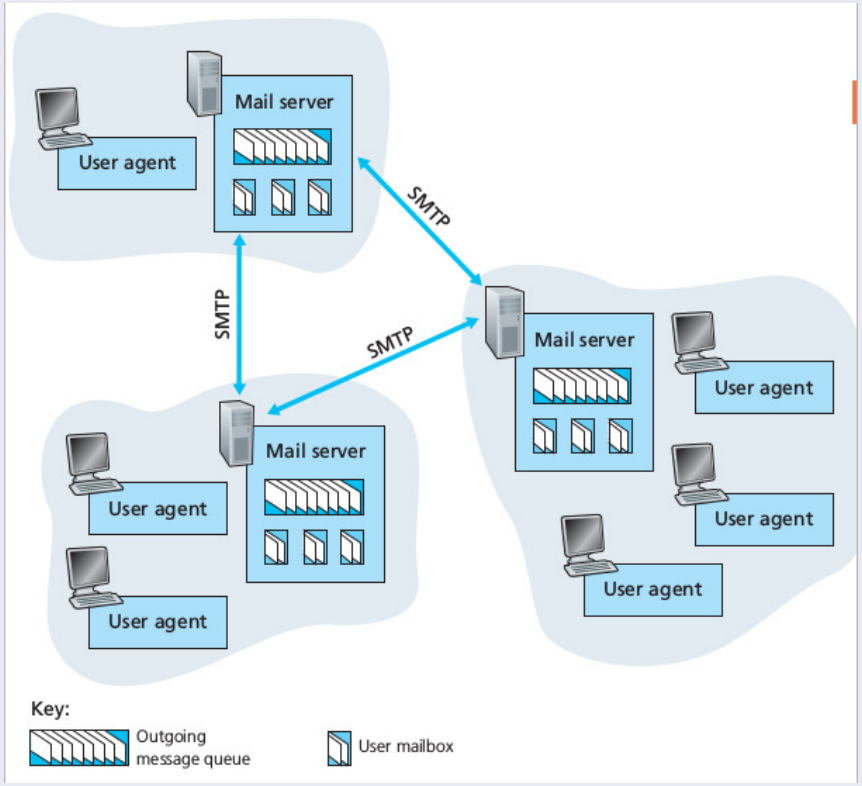
E-mail e Internet



Componentes: Alice envia e-mail a Bob

Agentes de usuários (clientes de e-mail):

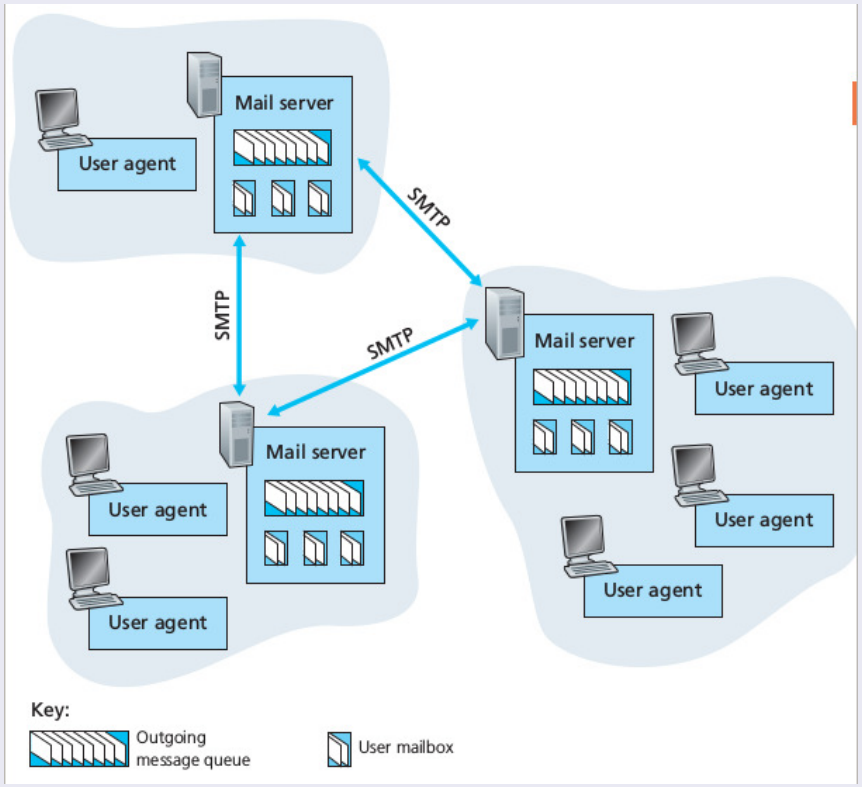
- usados para ler, responder, encaminhar, salvar e compor mensagens.
- Thunderbird, Icedove, Outlook, Apple Mail.



Componentes: Alice envia e-mail a Bob

Servidores de e-mail:

- recebe mensagens de e-mail criadas pelos agentes de usuários: as mensagens são enfileiradas na fila de saída do servidor.
- formam o núcleo da infraestrutura de e-mail
- cada destinatário (Bob) possui um depósito de mensagens, chamado de **mailbox**: quando Bob deseja ter acesso às suas mensagens em sua mailbox, o servidor de e-mail que provê sua mailbox autentica Bob (autoriza, com usuário e senha)



SMTP: RFC 5321

Mais antigo que o HTTP. A RFC é de 1982.

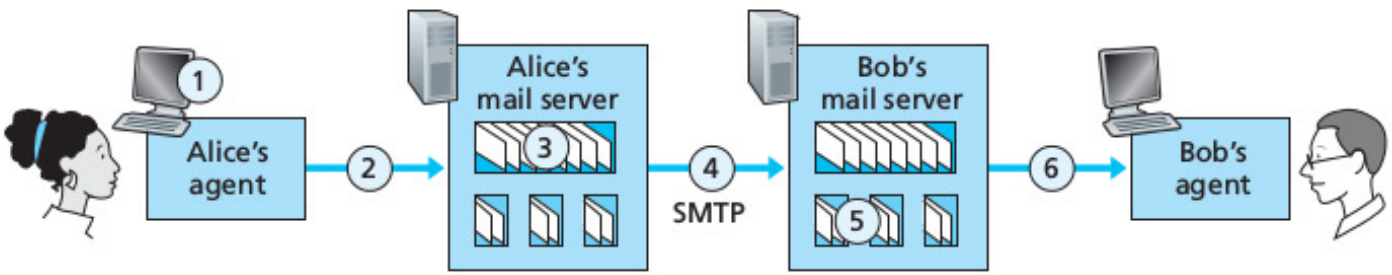
Características legadas: codificação ASCII 7 bits.

- Isso provoca a necessidade de se recodificar dados, em geral encaminhados como anexos, para (e de) ASCII 7 bits
- ASCII 7 bits (do SMTP e do FTP) era uma forma de se usar 8 bits com código de correção (paridade).

SMTP: RFC 5321

Fluxo básico de operação: Alice envia mensagem de e-mail para Bob

1. Alice usa o cliente de e-mail (agente de e-mail), indica qual o endereço de e-mail de Bob (bob@someschool.edu), escreve a mensagem para Bob e instrui o cliente de e-mail enviar a mensagem.
2. O cliente de e-mail de Alice manda a mensagem recém criada para o servidor de e-mail de Alice. Nesse servidor, a mensagem é colocada em uma fila de mensagens.
3. O lado (a seção/parte) cliente do servidor SMTP de Alice vê a mensagem na fila. Em seguida, ele abre uma conexão TCP com o servidor SMTP que provê a mailbox de Bob.

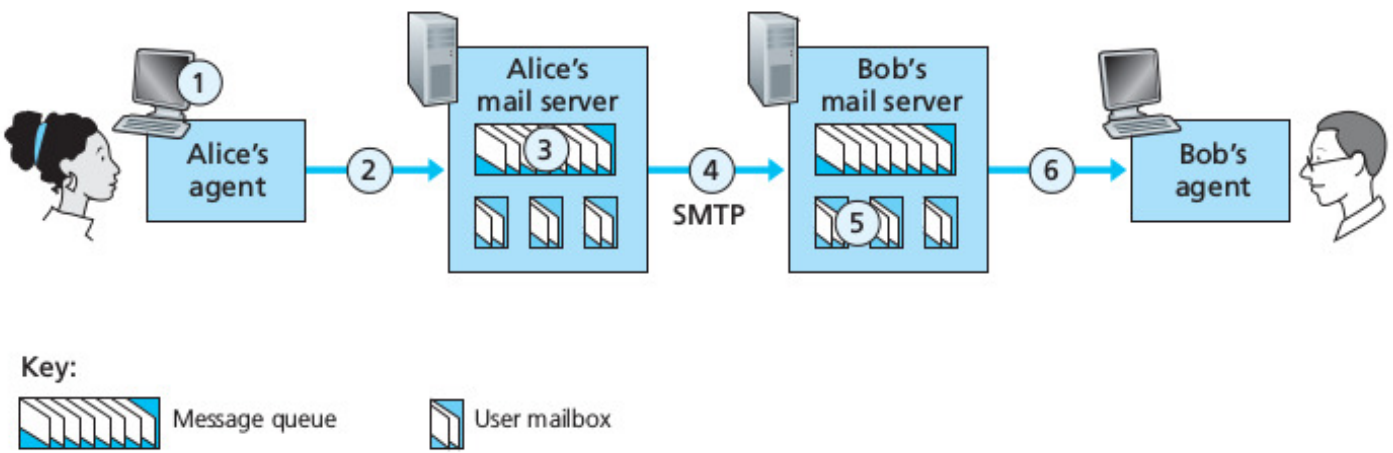


SMTP: RFC 5321

Fluxo básico de operação: Alice envia mensagem de e-mail para Bob

4. Depois de um handshaking inicial, o lado cliente SMTP de Alice envia a mensagem enfileirada pela conexão TCP.
5. No servidor de e-mail de Bob, o lado servidor SMTP recebe a mensagem para Bob. Esse componente, em seguida, põe a mensagem na mailbox de Bob.
6. Em algum momento oportuno, Bob usa seu cliente de e-mail para verificar se há novas mensagens e lê-las.

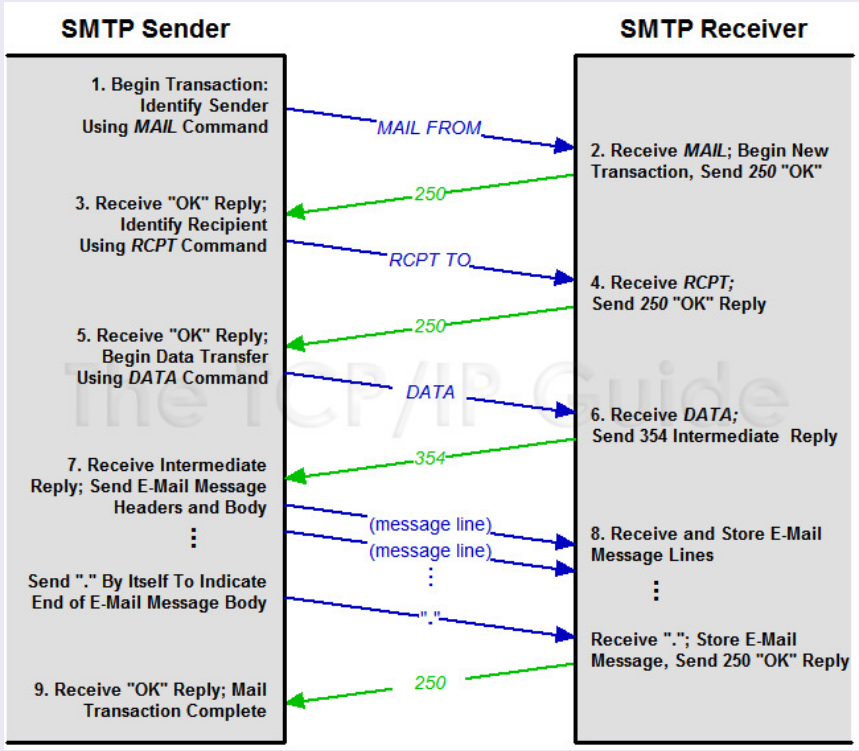
Não existe, normalmente, a figura de um servidor intermediário de e-mail.



SMTP: RFC 5321

Transferência de mensagens SMTP.

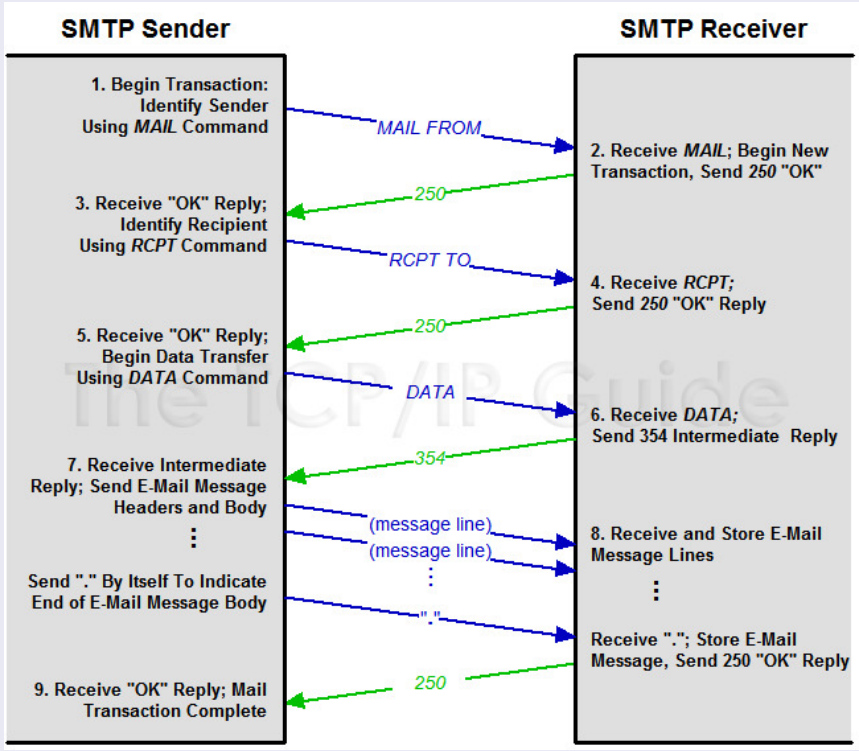
- O SMTP cliente estabelece uma conexão na porta 25 do SMTP servidor. Se o servidor de destino estiver desativado, o cliente tenta novamente mais tarde. (Camada de Transporte)
- Uma vez a conexão estabelecida, o servidor e o cliente realizam um handshaking de camada de aplicação: o cliente indica o endereço de e-mail do emissor e o do destinatário.



SMTP: RFC 5321

Transferência de mensagens SMTP.

- Depois do handshaking, acontece, de fato, o envio da mensagem: o procotolo TCP garante serviço de transporte confiável
- Se houver mais de uma mensagem, o processo pode ser repetido através da conexão. Caso contrário, o cliente solicita o fechamento de conexão TCP. (Camada de Transporte)



SMTP: RFC 5321

Transmissão SMTP (exemplo):

```
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
```

SMTP: RFC 5321

Comandos:

- HELO, MAIL FROM, RCPT TO, DATA e QUIT.
- Fim de mensagem: CRLF.CRLF
- Código de resposta (com comentário)

Para aproveitar a conexão SMTP, basta o cliente começar o processo com novo comando MAIL FROM:

Isso indica, implicitamente, o final da mensagem anterior.

Hands on

```
telnet servername 25
HELO
```

HTTP vs. SMTP

HTTP é protocolo do tipo PULL: alguém disponibiliza informações num servidor Web e os usuários baixam (PULL) a informação do servidor.

- A conexão TCP é iniciada pela máquina que quer receber o objeto.
- Cada objeto requisitado segue em mensagem de resposta distinta.

SMTP é um protocolo do tipo PUSH: o servidor de e-mail emissor envia a mensagem para o servidor de e-mail receptor.

- A conexão TCP é iniciada pela máquina que quer enviar o arquivo.
- Cada mensagem SMTP deve ser codificada em formato ASCII 7 bits
- SMTP encapsula mensagens de e-mail que consistem em mais de um objeto em uma única mensagem.

Formatos de mensagens de e-mail

RFC 5322: informações periféricas (linhas de cabeçalho)

- cabeçalho e corpo de mensagem são separados por linha (CRLF).
- cada linha de cabeçalho especificada pela RFC contém texto legível, constituindo de uma palavra chave seguida de ":" que, por sua vez, é seguido de um valor
- algumas palavras chave são obrigatórias enquanto outras são opcionais: From:, To:, Subject:
- Não confundir linhas de cabeçalho com os comandos SMTP!

Cabeçalho típico

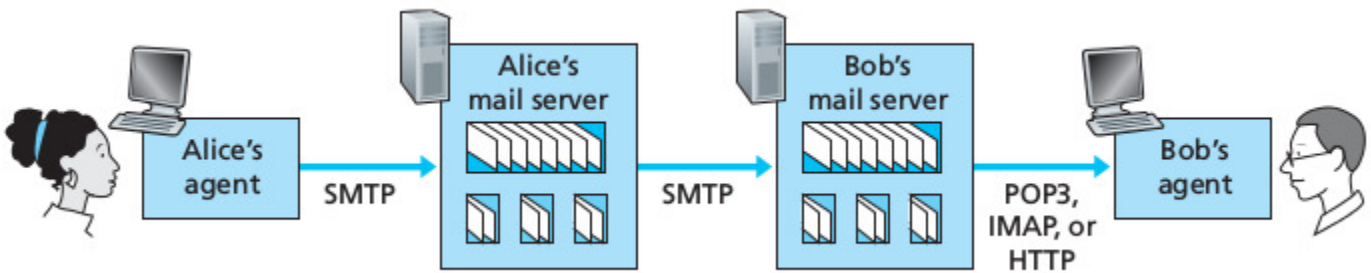
```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Searching for the meaning of life.
```

Procolos de acesso a e-mail

Uma vez terminados os processos de encaminhamento de e-mail, a mensagem enviada por Alice chega à mailbox de Bob

- Bob usa um **cliente de e-mail** para logar em sua mailbox e interagir com seu servidor de e-mail
- O cliente de e-mail pode estar localizado no mesmo host do servidor de e-mail (Pine) como em outro host

O **encaminhamento** ocorre, tipicamente, em dois passos: **relay**.



Procolos de acesso a e-mail

Q: Como o destinatário coleta e lê suas mensagens?

- Seria usando o mesmo SMTP (em modo PUSH)? Não. É necessário um protocolo PULL.
- **A:** POP3: Post Office Protocol - Version 3
- **A:** IMAP: Internet Mail Access Protocolo
- **A:** Webmail (HTTP)

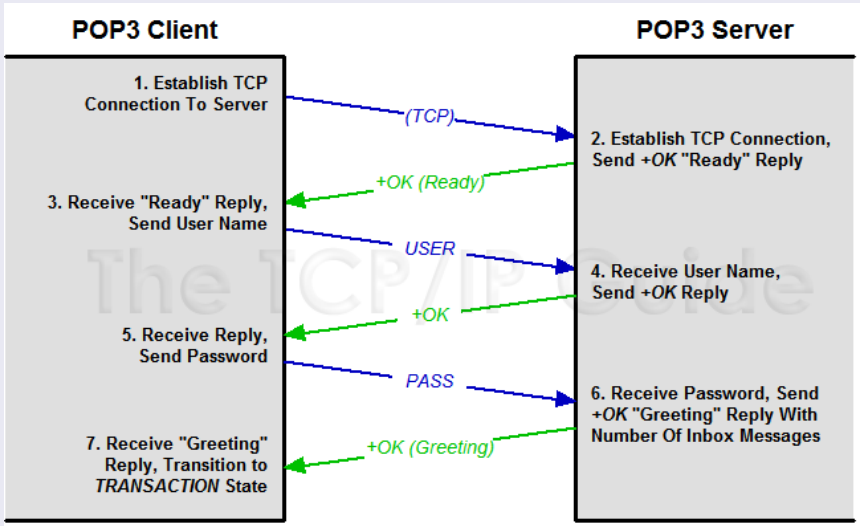
POP3

Procolo para acesso a e-mail extremamente simples. Definido na RFC 1939.

POP3 inicia quanto o cliente de e-mail abre uma conexão TCP com o servidor de e-mail na porta 110.

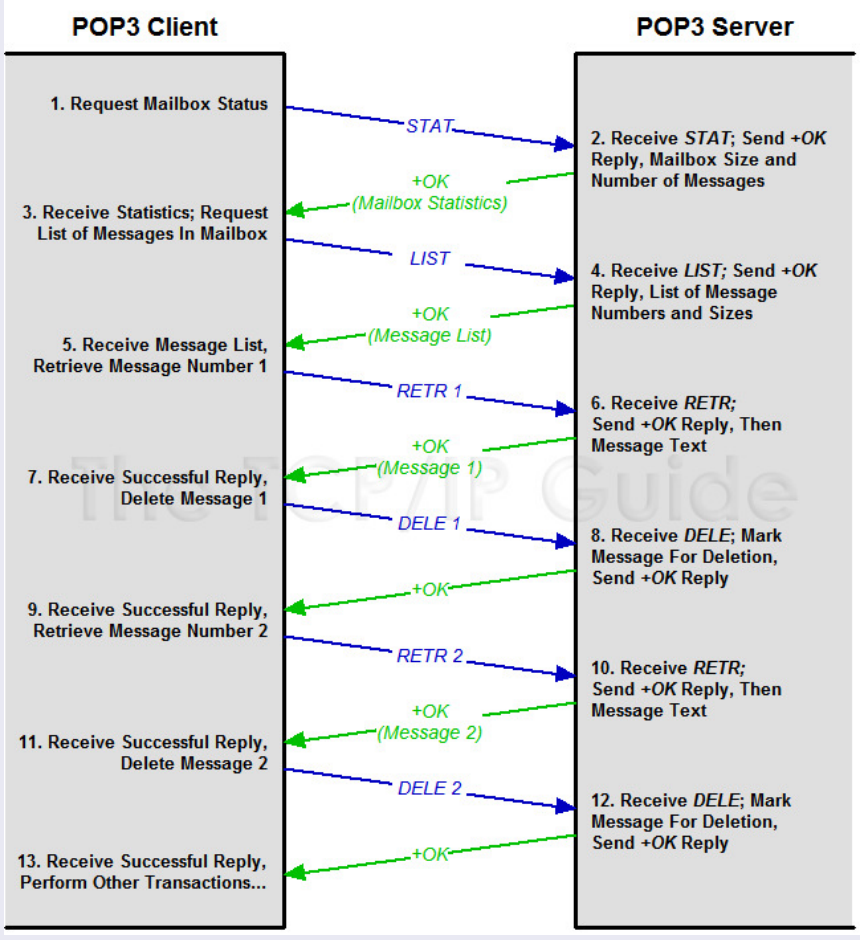
Com a conexão TCP estabelecida, POP3 atravessa 3 fases:

1. Autorização: o cliente de e-mail envia o usuário e a senha (ambos em claro) para autenticar (autorizar) o usuário.



POP3

2. Na segunda fase, transação, o cliente de e-mail obtém as mensagens. O cliente de e-mail pode marcar mensagens para que sejam apagadas, remover marcações de mensagens escolhidas para apagamento e obter estatísticas de e-mail.
3. A terceira fase, update, acontece depois que o cliente de e-mail envia o comando **quit**, que encerra a sessão POP3. Nesse momento, o servidor de e-mail deleta todas as mensagens marcadas para apagamento.



POP3

Numa transação POP3, o cliente de e-mail envia comandos e o servidor responde a cada comando com uma resposta.

- +OK, às vezes seguida pelos dados que são respondidos pelo servidor em função de um comando do cliente.
- -ERR, algo de errado aconteceu com o comando anterior.

A fase de autorização possui dois comandos:

- user <username>
- pass <password>

POP3

Autorização

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

POP3

Fase transação $\Longrightarrow$ baixar e apagar / baixar e manter.

```
C: list # comando de listagem: duas mensagens <.>
S: 1 498 # identificador e tamanho da mensagem armazenada
S: 2 912
S: .
C: retr 1 # obter mensagem 1
S: (blah blah ...
S: ..................
S: ..........blah)
S: .
C: dele 1 # marcar mensagem 1 para remoção
C: retr 2 # obter mensagem 2
S: (blah blah ...
S: ..................
S: ..........blah)
S: .
C: dele 2 # marcar mensagem 2 para remoção
C: quit
S: +OK POP3 server signing off
```

POP3

Fase transação

- Problema com o usuário nômade.
- Protocolo que mantém estado apenas dentro da conexão.

IMAP

Resolve o problema do usuário nômade

RFC 3501: possui mais funcionalidades que o POP3, porém é mais complexo.

Um servidor IMAP associará cada mensagem a uma pasta

- mensagem que chega: INBOX
- usuário pode criar outras pastas para onde pode movimentar as mensagens que recebe, pode ler a mensagem, deletar a mensagem, etc

IMAP

Funcionalidades:

- comandos: criar pastas, mover mensagens entre pastas, buscar em pastas remotas por mensagens usando determinado critério
- mantém estado do usuário entre sessões
- permite filtrar mensagens de forma a exibir os componentes das mensagens: pouca largura de banda.

Web-based e-mail

Webmail: aplicação Web comunica-se diretamente com a mailbox.

Identificação de hosts

Um host pode ser identificado por um hostname: cnn.com, www.yahoo.com, www.fga.unb.br

- São mnemônicos, melhores tratados por seres humanos
- Provêem pouca informação acerca da localização na rede Internet. (.br, Brasil)
- Alternativa: endereço IP.

IPv4

- número composto por quatro bytes
- possui rígida estrutura hierárquica: pela leitura da esquerda para a direita, obtém-se mais informações sobre onde o host está localizado na Internet.

Serviços providos pelo DNS

Alternativas de identificação de um host na Internet: hostname ou IP.

DNS (Domain Name System) serve auxiliando o mapeamento entre hostname e IP

O que é?

1. banco de dados distribuído implementado em uma hierarquia de servidores DNS
2. protocolo da camada de aplicação que permite aos hosts consultarem o banco de dados distribuído

Aplicação: BIND (Berkeley Internet Name Domain)

Usa como protocolo de transporte o UDP, na porta 53.

Identificação de hosts

Usado por outros protocolos de camada de aplicação (HTTP, SMTP e FTP) na tradução de hostnames para IPs

Cenário: Browser tentando acessar `www.someschool.edu/index.html`

Para abrir um socket TCP, deve-se saber o IP de `www.someschool.edu`

1. Na máquina do usuário do browser, é executada a aplicação cliente DNS
2. O browser extrai o hostname `www.someschool.edu` da URL e passa o hostname para a aplicação cliente DNS
3. A aplicação cliente DNS realiza uma consulta contendo o hostname ao servidor DNS
4. O cliente DNS **eventualmente** recebe uma resposta, que inclui o endereço de IP do host consultado (hostname)
5. Uma vez que o browser recebe o endereço de IP a partir da consulta DNS bem sucedida, o browser inicia uma conexão TCP com o servidor HTTP, provido na porta 80 do host servidor.

Atrasos

DNS adiciona um atraso adicional às aplicações que dele dependem.

Caches próximos de DNS podem reduzir esse efeito, melhorando a sensação de navegabilidade!

Serviços importantes providos pelo DNS

Host aliasing

- Um host com um hostname complicado pode possuir um ou mais aliases: www.unb.br e unb.br ambos apontam para 164.41.101.33
- Nome canônico: nome comprido

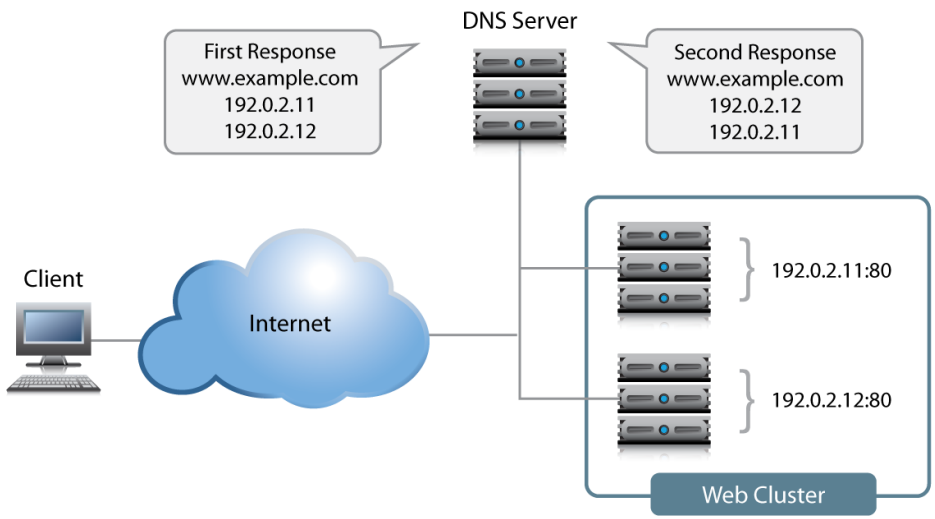
Mail server aliasing

- é desejável que endereços de e-mail sejam mnemônicos.
- DNS pode ser usado por uma aplicação de e-mail para obter o hostname canônico para um alias de hostname bem como o endereço IP daquele host
- registro MX permite ao servidor de e-mail e ao servidor Web de uma corporação terem o mesmo hostname (alias).

Serviços importantes providos pelo DNS

Distribuição/Balanceamento de carga

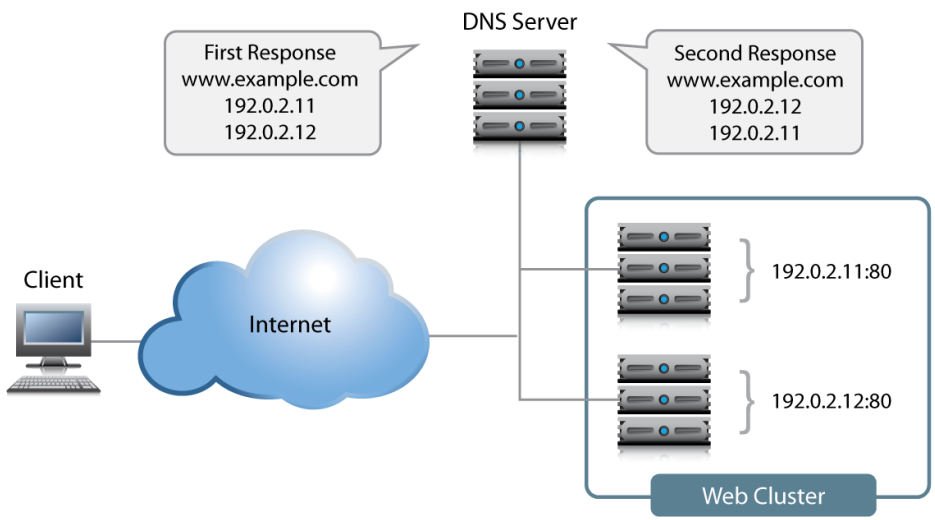
- DNS também é usado para realizar balanceamento de carga entre servidores replicados (Web servers, por exemplo)
- Sites responsáveis por elevada demanda (cnn.com), são replicados entre diferentes servidores, cada um executado por um equipamento diferente e, também, com endereço de IP diferente.
- Um conjunto de endereços de IP é associado com um hostname canônico. O banco de dados DNS possui esse conjunto de IPs



Serviços importantes providos pelo DNS

Distribuição/Balanceamento de carga

- Quando um cliente faz uma consulta para um nome que possui, associado à sua entrada no DNS, um conjunto de IPs, o servidor responde apresentando a lista completa de endereços de IP, mas rotaciona a ordem dos endereços em cada resposta.
  - Clientes tipicamente estabelecem conexões enviando pedidos de conexão para o primeiro endereço IP listado no conjunto.
  - Isso permite fazer balanceamento/distribuição de tráfego.



Visão geral de como o DNS funciona

DNS é especificado nas RFCs 1034 e 1035, e atualizado em várias outras RFCs

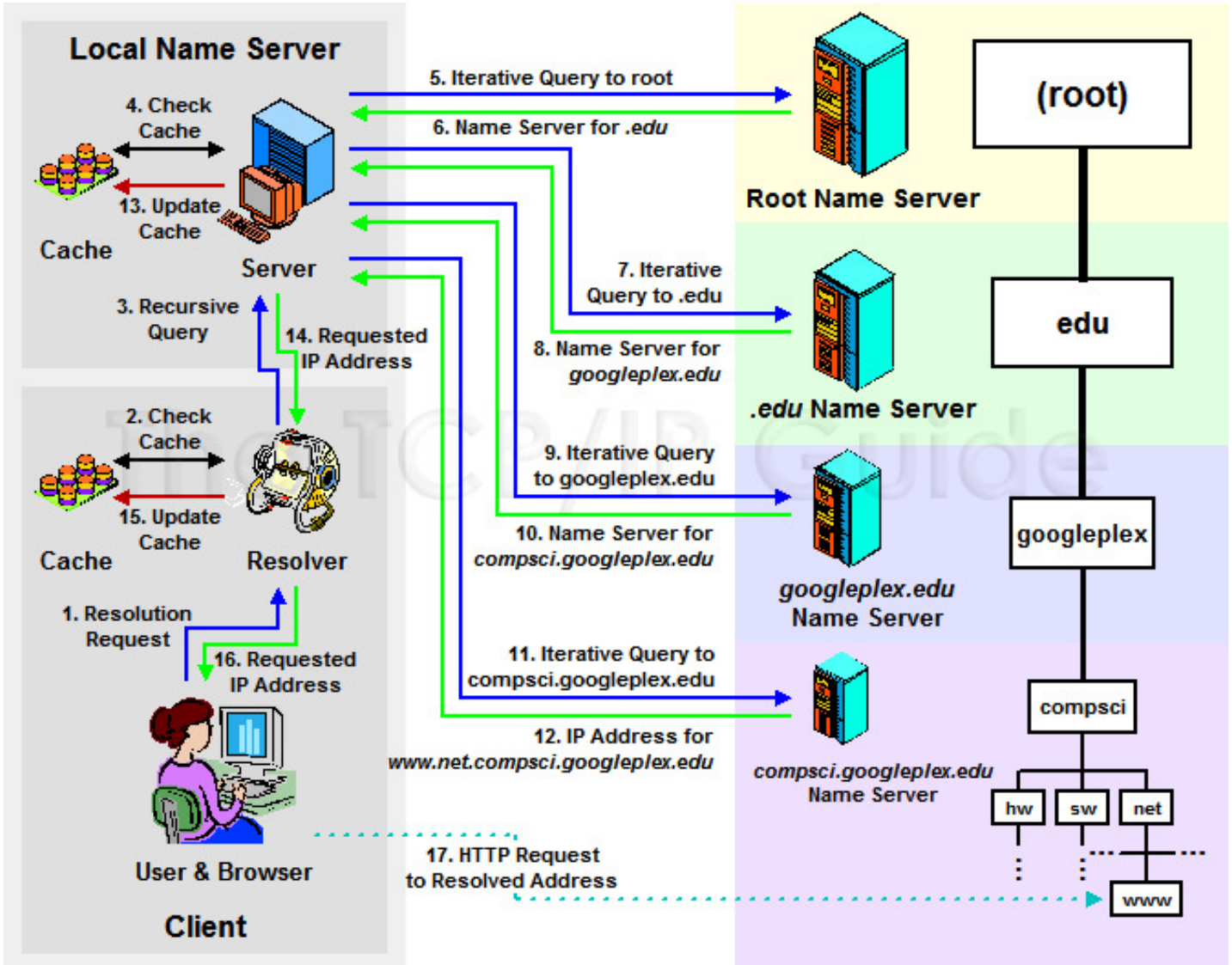
Livros: Albitz e Liu, 1993

Visão geral de como o DNS funciona

Suponhamos que uma aplicação que executa no host do cliente necessita traduzir um hostname para IP

- A aplicação irá invocar o lado cliente do serviço DNS, indicando o hostname que precisa ser traduzido: gethostbyname() é a primitiva de sistema provida pelo UNIX
- O DNS no host do cliente prossegue, enviando uma mensagem de consulta pela rede: todos os pedidos de consultas e respostas são enviados em datagramas UDP na porta 53.
- Depois de determinado atraso, o host do cliente recebe a mensagem de resposta que provê o mapeamento solicitado.
- O mapeamento é encaminhado para a aplicação que demandou o processo de consulta DNS

DNS: O serviço de diretório da Internet



Visão geral de como o DNS funciona

Visão caixa preta: um serviço de tradução hostname para IP.

- Por trás dessa visão simplificada do sistema, há um serviço complexo.
- Grande número de servidores DNS distribuídos geograficamente
- Protocolo de camada de aplicação que especifica como servidores DNS e hosts que demandam consultas devem interagir

Visão geral de como o DNS funciona

Uma proposta de arquitetura simples:

- Um servidor DNS com todos os mapeamentos. Modelo centralizado.
- Todos os clientes encaminham consultas ao único servidor DNS, o servidor DNS responde diretamente aos clientes demandantes
- **Problema**: Modelo inapropriado para a Internet atual
  - Único ponto de falha: o desligamento de um equipamento provoca interrupção global do serviço
  - Elevado volume de tráfego: um servidor DNS responsável por responder a todas as consultas DNS
  - Banco de dados centralizado e, eventualmente, distante: um único servidor DNS não pode estar simultaneamente próximo a todos os seus clientes
  - Manutenção: o único servidor DNS deveria manter todos os registros para todos os hosts conectados à Internet
- **Essa proposta não escala!**

Visão geral de como o DNS funciona

Uma proposta de arquitetura simples:

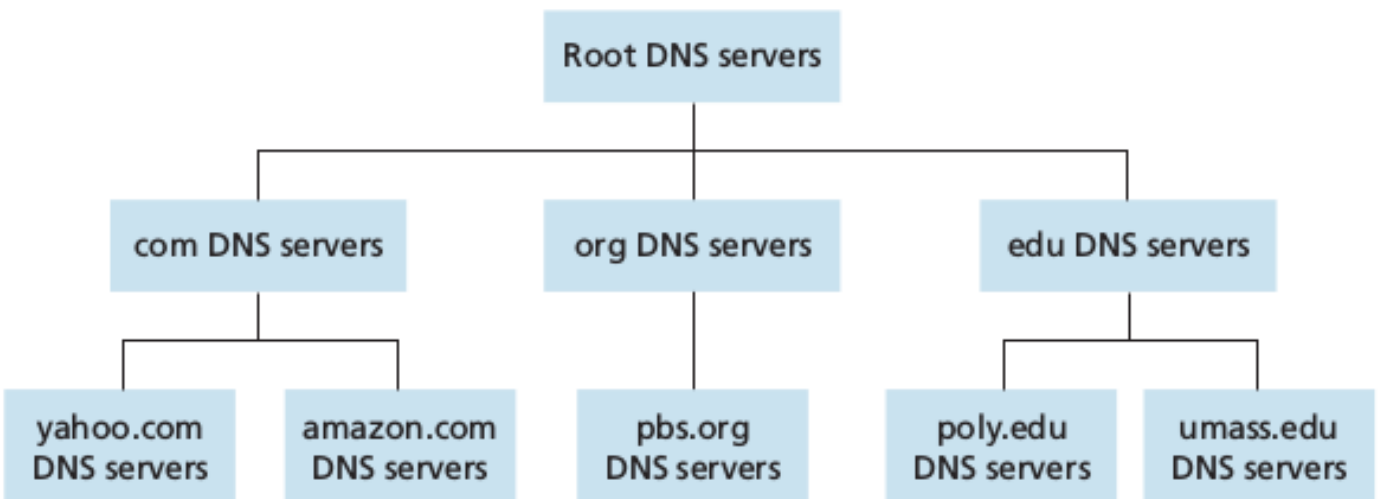
- Vida real:
  - Banco de dados distribuído e hierárquico: trata melhor escala. Realização de DNS usa um grande número de servidores organizados de forma hierárquica e distribuídos pelo mundo
  - Nenhum servidor DNS possui todas as entradas para todos os hosts da Internet: mapeamento é distribuído entre os servidores DNS

Visão geral de como o DNS funciona

Primeira aproximação $\Longrightarrow$ 3 classes de servidores DNS: DNS root, DNS top-level domain (TLD) e authoritative DNS.

Cenário: um cliente deseja acessar o site da Amazon e começa por pedir o IP mapeado para o nome www.amazon.com

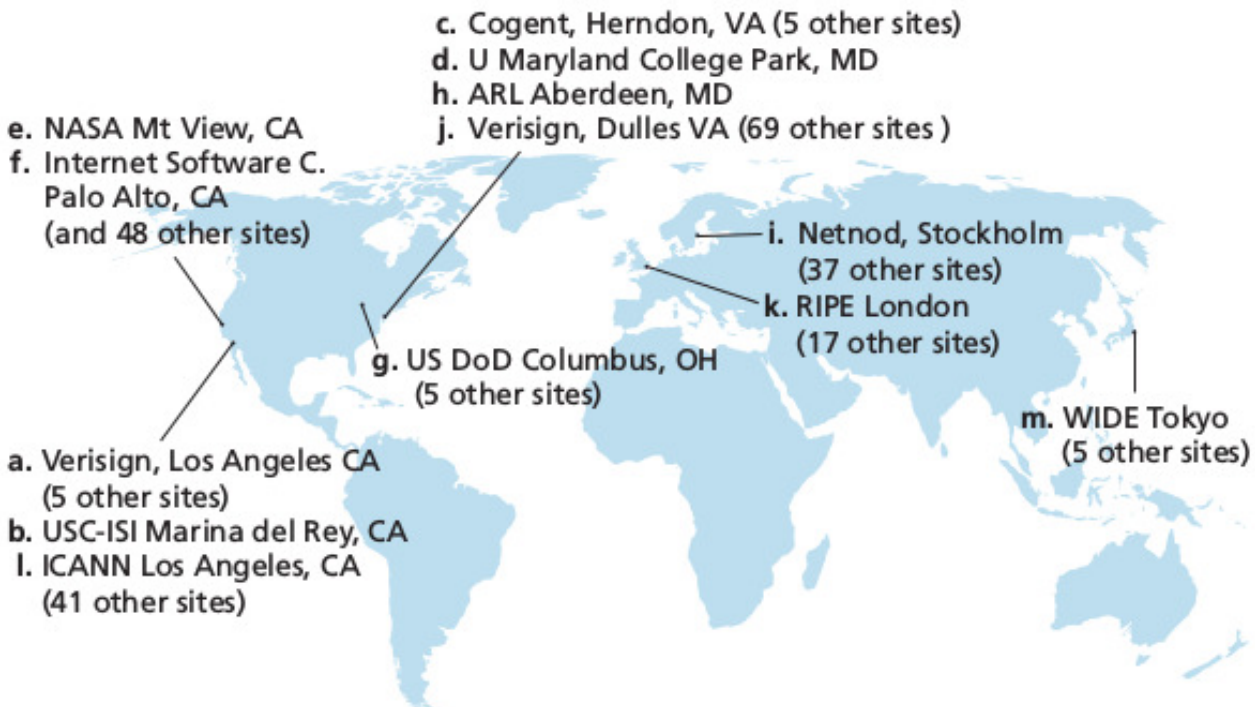
- o cliente contacta um dos root, que retorna IPs para os TLD do TLD .com
- o cliente contacta um dos TLD, que retorna o endereço IP para um authoritative DNS para amazon.com
- finalmente, o cliente contacta um dos authoritative DNS para amazon.com, que retorna o IP do hostname www.amazon.com



Visão geral de como o DNS funciona

Classes em detalhes:

- DNS raiz/root: na Internet há 13 servidores raiz (A a M), a maioria deles localizada na América do Norte. $\Longrightarrow$ Na verdade, cada um dos 13 nós raiz é implementado por uma rede de servidores replicados: segurança e confiabilidade. (Até 2011, eram 247 hosts provendo os 13 nós raiz.)



Visão geral de como o DNS funciona

Classes em detalhes:

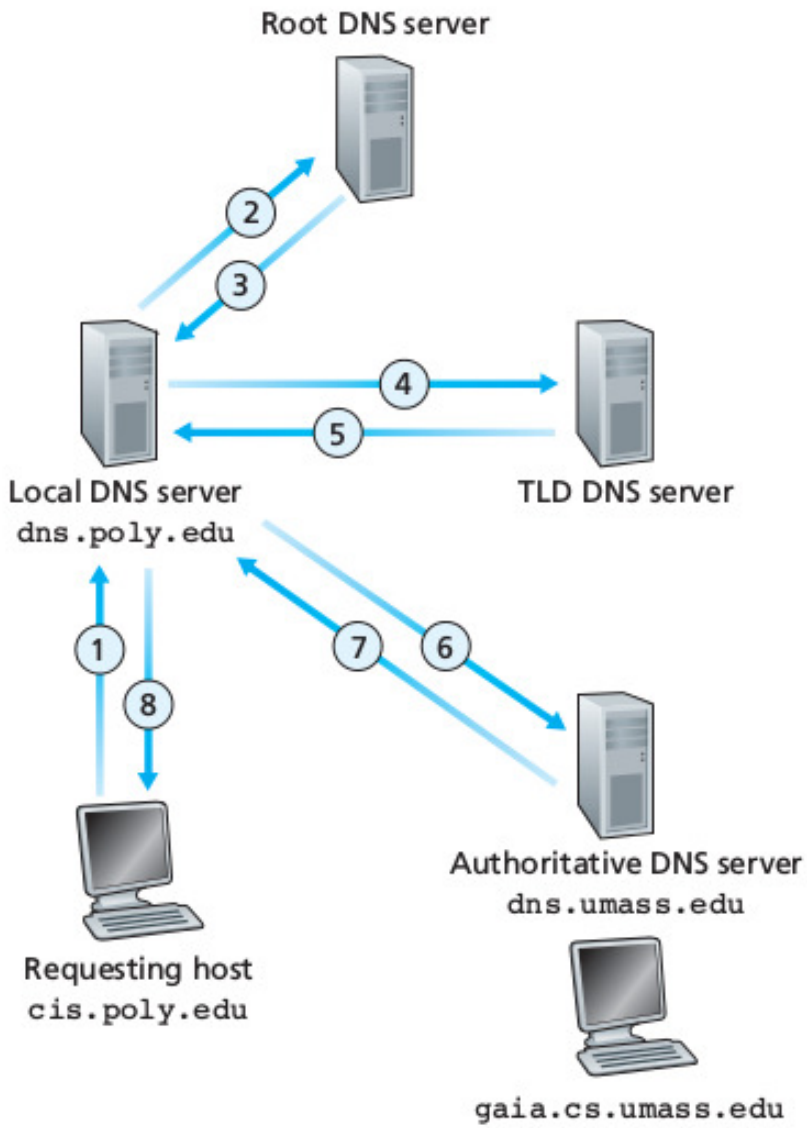
- **Top-level domain**: Esses são os servidores responsáveis pelos domínios altos como `com`, `org`, `net`, `edu` e `gov`, e por todos os domínios de países como br, uk, fr, ca e jp.
- **Authoritative DNS servers**: cada organização com host acessíveis publicamente (Web servers e mail servers) deve prover registros de DNS publicamente acessíveis que mapeiem os nomes dos hosts para endereços de IP. Um authoritative DNS da organização provê esses registros DNS. Há duas opções para uma organização:
  - Implementar seu próprio authoritative DNS para armazenar os registros.
  - Pagar para um authoritative DNS server

Visão geral de como o DNS funciona

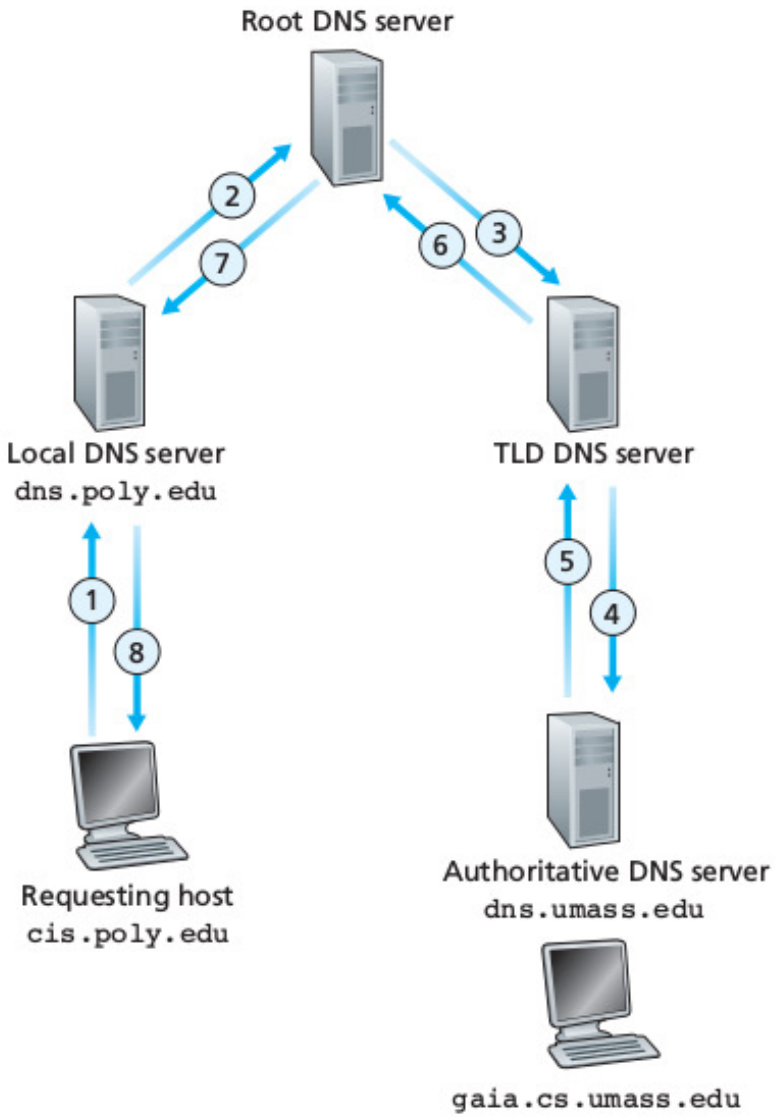
Além do root, TLD e authoritative DNS, há também o servidor de DNS local: ou mais de seus DNS locais (DHCP, cenas dos próximos capítulos!)

- Esse servidor não pertence à hierarquia de servidores, porém é elemento fundamental da arquitetura DNS
- Cada ISP (universidade, departamento acadêmico, empresa ou provedor residencial) possui um servidor de DNS local: Quando um cliente se conecta à sua rede de acesso, o ISP provê ao host o endereço de IP de um desses servidores.
- O DNS local não fica afastado do host que dele depende: quando um host realiza uma consulta, a consulta é enviada ao servidor de DNS local. Esse, por sua vez, age como um proxy, encaminhando as consultas para a hierarquia de servidores DNS.

DNS: O serviço de diretório da Internet



DNS: O serviço de diretório da Internet



UnB Gama
O novo endereço da tecnologia

Visão geral de como o DNS funciona

DNS caching

- Melhora do desempenho pela redução do atraso provocado pela sequência de consultas e, também reduz a sequência de iterações necessárias na resolução de nomes.
- Quando um servidor DNS recebe uma resposta DNS (mapeando um IP a um hostname), ele pode cachear a informação na memória local
  - Devido ao fato de os mapeamentos entre hostnames e endereços de IP não serem permanentes, servidores DNS descartam informações cacheadas depois de um período de tempo.
  - Load balancing pode ser afetado pelo caching.

Registros e Mensagens DNS

Resource Records (RRs)

- Cada mensagem de resposta DNS leva um ou mais resource records
- RFCs 1034 e 1035
- Estrutura: (Name, Value, Type, TTL)
  - TTL (time to live): determina quando um recurso deve ser removido do cache
  - Type=A, Name é hostname e Value é endereço de IP: (relay1.bar.foo.com, 145.37.93.126, A)
  - Type=NS, Name é um domínio e Value é o hostname do authoritative DNS server que sabe como obter os endereços de IP para os hosts no domínio.
    - Tipo de registro usado para rotear consultas DNS através da cadeia de consultas.
    - (foo.com, dns.foo.com, NS)
  - Type=CNAME, Value é o nome canônico para o alias do hostname Name. Esse registro pode prover aos hosts consultantes o nome canônico de um hostname: (foo.com, relay1.bar.foo.com, CNAME)
  - Type=MX, Value é o nome canônico para o servidor de e-mail que possui como alias o hostname Name: (foo.com, mail.bar.foo.com, MX)

Registros e Mensagens DNS

Resource Records (RRs)

- Se um servidor DNS é authoritative para um hostname em particular, o servidor DNS conterá um registro do Type A para o hostname. (Mesmo que não seja authoritative, é possível ter um registro Type A em seu cache)
- Se o servidor não for authoritativo para um hostname, o servidor conterá um registro Type NS para o domínio que contém o hostname. Também conterá um registro Type A que provê o endereço IP do servidor DNS no campo Value do registro NS.

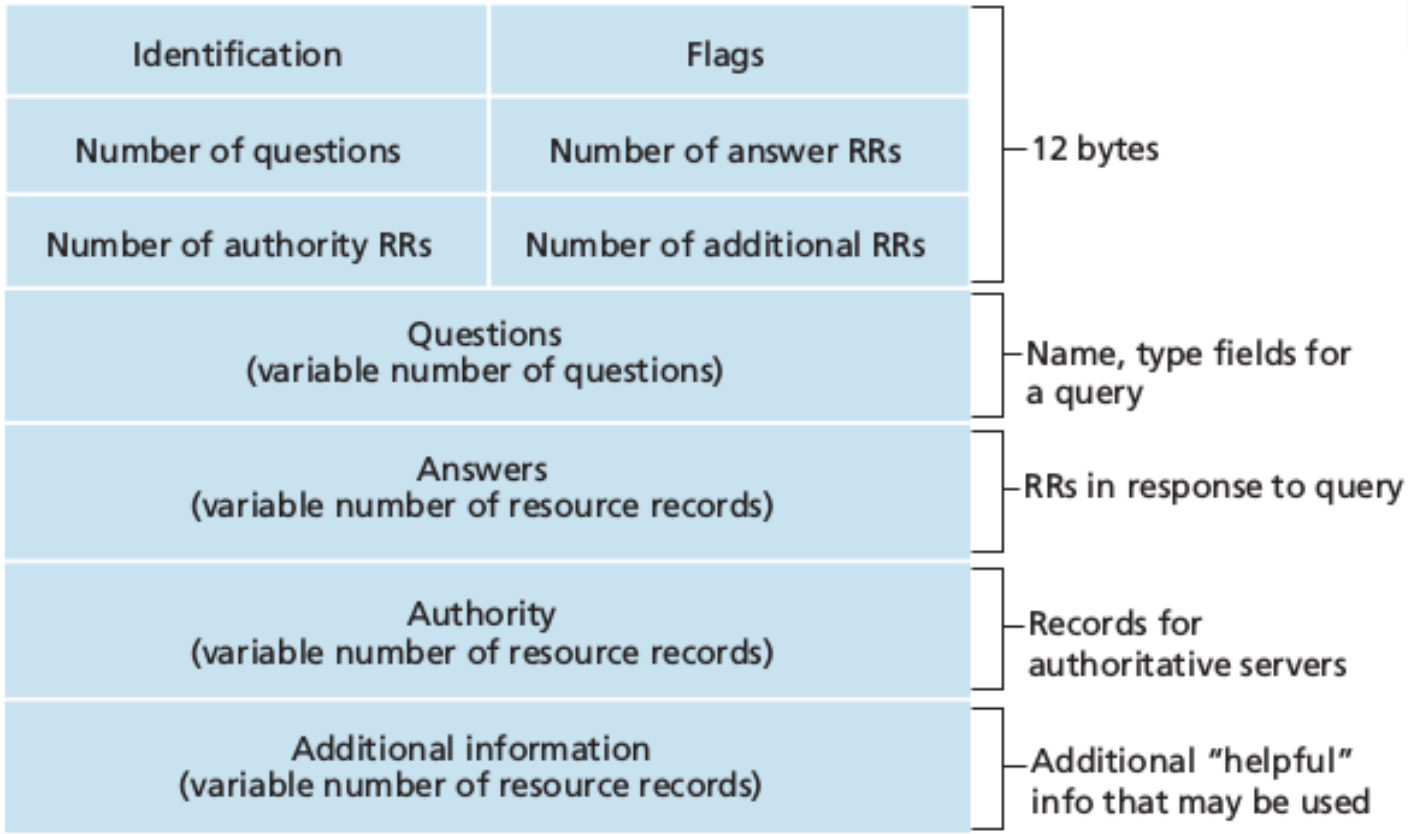
Registros e Mensagens DNS

Mensagens de consulta e de resposta possuem o mesmo formato

Campos:

- primeiros 12 bytes compõem a seção de cabeçalho
  1. campo de 16 bits que identifica a consulta. Copiado na mensagem de resposta, permitindo o cliente DNS relacionar as respostas com as consultas. $\Longrightarrow$ campo de flags:
     - 1 bit (query-reply flag) indica se a mensagem é uma consulta ou uma resposta
     - 1 bit (authoritative flag) indica que a resposta provém de um DNS authoritative para o nome consultado
     - 1 bit (recursion-desired flag) indica se o cliente (host ou servidor DNS) deseja que o servidor DNS realize recursão quando não possui o registro em sua base de dados
     - 1 bit (recursion-available field) indica (na resposta) se o servidor DNS suporta recursão.
  2. seção de consulta, contém informação sobre a consulta que está sendo feita.
     1. contém campo Name que contém o nome que está sendo consultado
     2. contém campo Type que indica o tipo de consulta apresentada

DNS: O serviço de diretório da Internet



UnB Gama
O novo endereço da tecnologia

Registros e Mensagens DNS

Mensagens de consulta e de resposta possuem o mesmo formato

Campos:

- primeiros 12 bytes compõem a seção de cabeçalho
  4. seção de resposta, contém os resource records para o nome que foi originalmente consultado: Type, Value e TTL. $\Longrightarrow$ uma resposta pode retornar múltiplos RR, pois a um hostname podem estar associados vários endereços IPs.
  5. Seção authority, que contém registros de outros servidores authoritatives
  6. Seção adicional, contém outros registros úteis

Hands on

`nslookup`

Registros e Mensagens DNS

Inserindo registros no banco de dados DNS:

- RFCs 2136 e 3007: updates dinâmicos de DNS.
- **Registrar**: entidade comercial que verifica a unicidade do nome do domínio, insere o nome de domínio no banco de dados DNS e coleta pequena taxa pelos serviços prestados.
- É necessário apresentar os nomes e endereços de IP para os seus servidores DNS authoritative primário e secundário
  - **Registrar** irá inserir entradas NS e A nos servidores TLD no banco de dados

    `(networkutopia.com, dns1.networkutopia.com, NS)`
    `(dns1.networkutopia.com, 212.212.212.1, A)`

  - É necessário que você insira registros Type A para seu Web server e MX para seu servidor de e-mail em seus authoritative DNS.

UnB Gama
O novo endereço da tecnologia

Vulnerabilidades do DNS

DDoS bandwidth-flooding $\Longrightarrow$ 21 de Outubro de 2002.

- botnet enviou cargas massivas de mensagens ICMP ping para cada um dos 13 root
- poucos danos: filtros de pacotes bloquearam a chegada de pacotes ICMP ping a alguns dos root; cache local permitiu funcionamento em contingência por um bom período.

Dilúvio de consultas DNS para servidores TLD

Man-in-the-middle: envenenamento

DDoS usando um DNS: apontamento oportunístico direcionando tráfego para servidor alvo, provocando queda de serviço.