



INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA  
RIO GRANDE DO NORTE

---

# Camada de Aplicação

---

---



# Camada de aplicação

---

- 2.1 Princípios de aplicações de rede
- 2.2 Web e HTTP
- 2.3 FTP
- 2.4 Correio eletrônico
  - SMTP, POP3, IMAP
- 2.5 DNS



# Camada de aplicação

---

## Nossos objetivos:

- Conceitual, aspectos de implementação de protocolos de aplicação de redes
  - Modelos de serviço da camada de transporte
  - Paradigma cliente-servidor
  - Paradigma **peer-to-peer**
  - Aprender sobre protocolos examinando protocolos da camada de aplicação populares:
    - HTTP
    - FTP
    - SMTP/ POP3/ IMAP
      - DNS

# Algumas aplicações de rede

---

- E-mail
- Web
- Mensagem instantânea
- Login remoto
- P2P file sharing
- Jogos de rede multi-usuário
- Streaming stored vídeos
- Telefonia via Internet
- Videoconferência em tempo real
- Computação paralela massiva

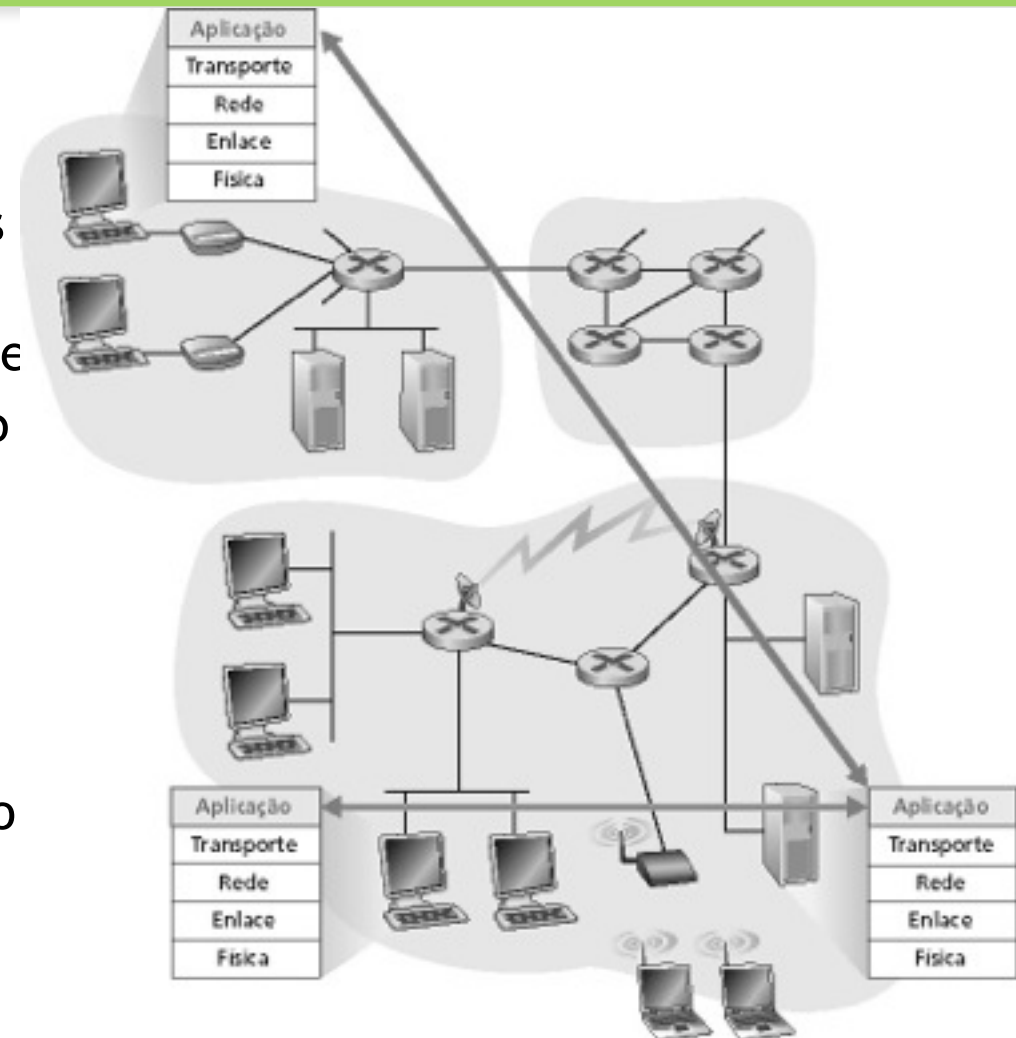
# Criando uma nova aplicação de rede

## Escrever programas que

- Executem sobre diferentes sistemas finais e
- Se comuniquem através de uma rede
- Ex.: Web - software de servidor Web se comunicando com software do browser.

## Nenhum software é escrito para dispositivos no núcleo da rede

- Dispositivos do núcleo da rede não trabalham na camada de aplicação
- Esta estrutura permite um rápido desenvolvimento de aplicação





# Camada de aplicação

---

- 2.1 Princípios de aplicações de rede
- 2.2 Web e HTTP
- 2.3 FTP
- 2.4 Correio eletrônico SMTP, POP3, IMAP
- 2.5 DNS



# Arquiteturas de aplicação

---

- Cliente-servidor
- **Peer-to-peer (P2P)**
- Híbrida de cliente-servidor e P2P



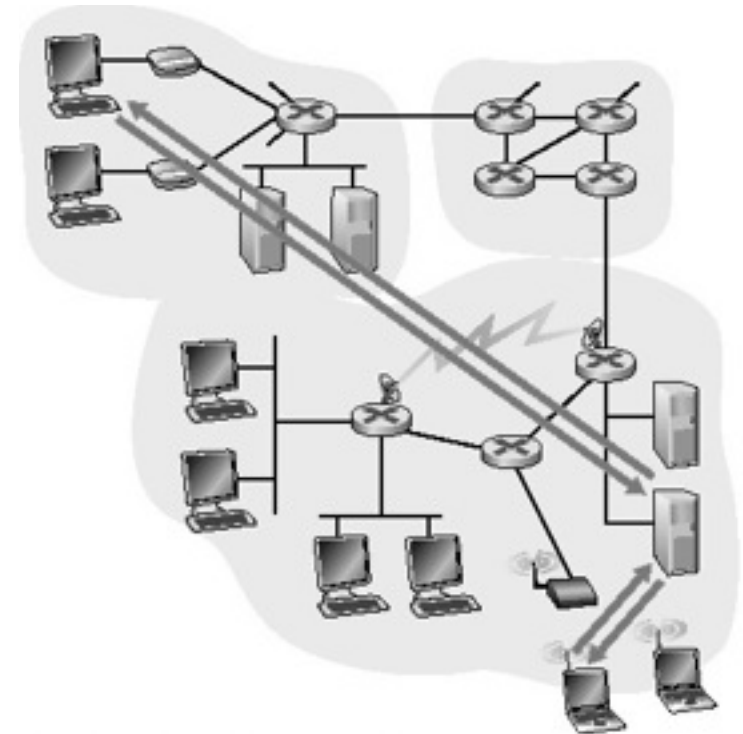
# Arquitetura cliente-servidor

## Servidor:

- Sempre ativo
- Endereço IP permanente
- Fornece serviços solicitados pelo cliente

## Clientes:

- Comunicam-se com o servidor
- Pode ser conectado intermitentemente
- Pode ter endereço IP dinâmico
- Não se comunicam diretamente uns com os outros



a. Aplicação cliente-servidor

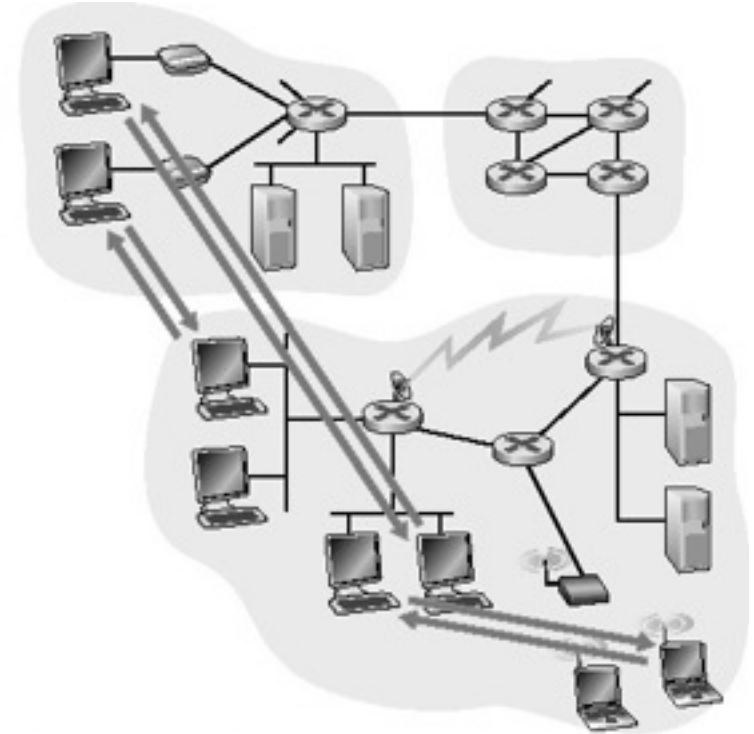




# Arquitetura P2P pura

- Nem sempre no servidor
- Sistemas finais arbitrários comunicam-se diretamente
- Pares são intermitentemente conectados e trocam endereços IP
- Ex.: Gnutella

Altamente escaláveis mas difíceis de gerenciar



b. Aplicação P2P



# Híbrida de cliente-servidor e P2P

---

## Napster

- Transferência de arquivo P2P
- Busca centralizada de arquivos:
  - Conteúdo de registro dos pares no servidor central
  - Consulta de pares no mesmo servidor central para localizar o conteúdo

## Instant messaging

- Bate-papo entre dois usuários é P2P
- Detecção/localização centralizada de presença:
  - Usuário registra seu endereço IP com o servidor central quando fica on-line
  - Usuário contata o servidor central para encontrar endereços IP dos vizinhos



# Comunicação de processos

---

**Processo:** programa executando num hospedeiro

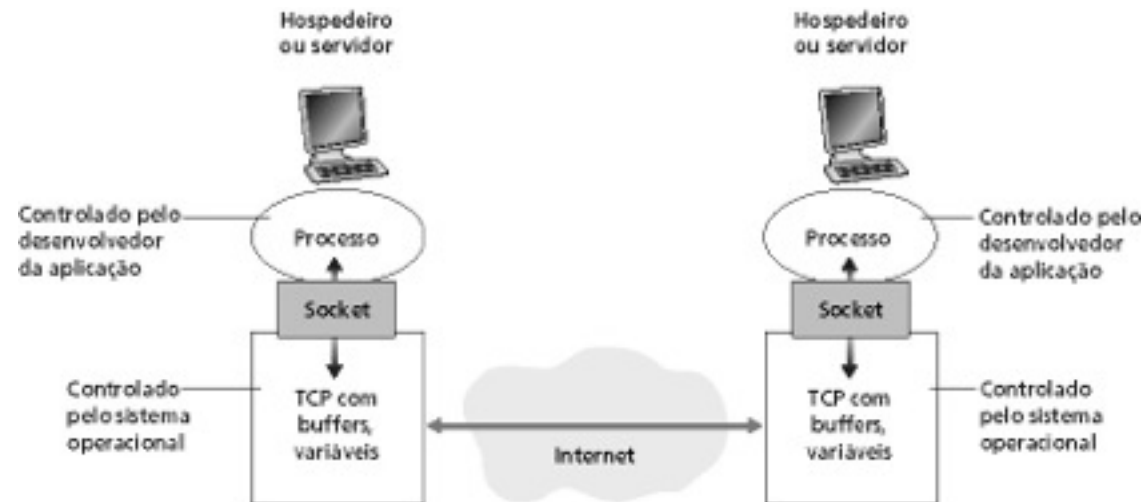
- Dentro do mesmo hospedeiro: dois processos se comunicam usando **comunicação interprocesso** (definido pelo OS)
- Processos em diferentes hospedeiros se comunicam por meio de troca de **mensagens**
- **Processo cliente:** processo que inicia a comunicação
- **Processo servidor:** processo que espera para ser contatado

**Nota:** aplicações com arquiteturas P2P possuem processos cliente e processos servidor



# Sockets

- Um processo envia/recebe mensagens para/de seu **socket**
- O socket é análogo a uma porta
  - O processo de envio empurra a mensagem para fora da porta
  - O processo de envio confia na infra-estrutura de transporte no outro lado da porta que leva a mensagem para o socket no processo de recepção.
- API: (1) escolha do protocolo de transporte; (2) habilidade para fixar poucos parâmetros (**será explicado mais tarde**)





# Processos de endereçamento

---

- Para um processo receber mensagens, ele deve ter um identificador
- Um hospedeiro possui um único endereço IP de 32 bits
- **P.:** O endereço IP do hospedeiro onde o processo está executando é suficiente para identificar o processo?
- **R.:** Não, muitos processos podem estar em execução no mesmo hospedeiro.
- O identificador inclui o endereço IP e o **número da porta** associada ao processo no hospedeiro
- Exemplos de números de porta:
  - Servidor HTTP: 80
  - Servidor de Correio: 25
- (mais detalhes serão mostrados mais tarde)



# O protocolo de aplicação define

---

- Tipo das mensagens trocadas, mensagens de requisição e resposta
- Sintaxe dos tipos de mensagem: os campos nas mensagens e como são delineados
- Semântica dos campos, ou seja, significado da informação nos campos
- Regras para quando e como os processos enviam e respondem às mensagens

## Protocolos de domínio público:

- Definidos nas RFCs
- Recomendados para interoperabilidade
- Ex.: HTTP, SMTP

## Protocolos proprietários:

- Ex.: KaZaA

# De qual serviço de transporte uma aplicação necessita?

---

## Perda de dados

- Algumas aplicações (ex.: áudio) podem tolerar alguma perda
- Outras aplicações (ex.: transferência de arquivos, telnet) exigem transferência de dados 100% confiável

## Temporização

- Algumas aplicações (ex.: telefonia Internet, jogos interativos) exigem baixos atrasos para serem “efetivos”

## Banda passante

- Algumas aplicações (ex.: multimídia) exigem uma banda mínima para serem “efetivas”
- Outras aplicações (“aplicações elásticas”) melhoram quando a banda disponível aumenta”



# Requisitos de transporte de aplicação comuns

<b>Aplicação</b>	<b>Perdas</b>	<b>Banda</b>	<b>Sensível ao atraso</b>
file transfer	sem perdas	elástica	não
e-mail	sem perdas	elástica	não
Web documents	tolerante	elástica	não
real-time áudio/vídeo	tolerante	áudio:5Kb-1 Mb vídeo:10Kb-5 Mb	sim, 100's mseg
stored áudio/video	tolerante	igual à anterior	sim, segundos
jogos interativos	tolerante	kbps	sim, 100's mseg
e-business	sem perda	elástica	sim



# Serviços dos protocolos de transporte da Internet

---

## Serviço TCP:

- **Orientado à conexão:** conexão requerida entre processos cliente e servidor
- **Transporte confiável** entre os processador de envio e recepção
- **Controle de fluxo:** o transmissor não sobrecarrega o receptor
- **Controle de congestionamento:** protege a rede do excesso de tráfego
- **Não oferece:** garantias de temporização e de banda mínima

## Serviço UDP:

- Transferência de dados não confiável entre os processos transmissor e receptor
- Não oferece: estabelecimento de conexão, confiabilidade, controle de fluxo e de congestionamento, garantia de temporização e de banda mínima.

**P.:** Por que ambos? Por que existe o UDP?



# Aplicação e protocolos de transporte da Internet

<b>Aplicação</b>	<b>Protocolo de aplicação</b>	<b>Protocolo de transporte</b>
e-mail	smtp [RFC 821]	TCP
acesso de terminais remotos	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
transferência de arquivos	ftp [RFC 959]	TCP
streaming multimídia	RTP ou proprietário (ex.: RealNetworks)	TCP ou UDP
servidor de arquivos remoto	NSF	TCP ou UDP
telefonia Internet	RTP ou proprietário (ex.: Vocaltec)	tipicamente UDP



# Camada de aplicação

---

- 2.1 Princípios de aplicações de rede
- 2.2 Web e HTTP
- 2.3 FTP
- 2.4 Correio eletrônico
  - SMTP, POP3, IMAP
- 2.5 DNS



# Web e HTTP

---

## Primeiro alguns jargões

- **Página Web** consiste de **objetos**
- Objeto pode ser arquivo HTML, imagem JPEG, Java applet, arquivo de áudio,...
- A página Web consiste de **arquivo-HTML base** que inclui vários objetos referenciados
- Cada objeto é endereçado por uma **URL**
- Exemplo de URL:

`www.someschool.edu/someDept/`

Nome do

Nome do

# Visão geral do HTTP

## HTTP: hypertext transfer protocol

- Protocolo da camada de aplicação da Web
- Modelo cliente/servidor
  - **Cliente:** browser que solicita, recebe e apresenta objetos da Web
  - **Servidor:** envia objetos em resposta a pedidos
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068





# Visão geral do HTTP

---

## Utiliza TCP:

- Cliente inicia conexão TCP (cria socket) para o servidor na porta 80
- Servidor aceita uma conexão TCP do cliente
- mensagens HTTP (mensagens do protocolo de camada de aplicação) são trocadas entre o browser (cliente HTTP) e o servidor Web (servidor HTTP)
- A conexão TCP é fechada

## HTTP é “stateless”

- O servidor não mantém informação sobre os pedidos passados pelos clientes

## Protocolos que mantêm informações de “estado” são complexos!

- Histórico do passado (estado) deve ser mantido
- Se o servidor/cliente quebra, suas visões de “estado” podem ser inconsistentes, devendo ser reconciliadas



# Conexões HTTP

---

## HTTP não persistente

- No máximo, um objeto é enviado sobre uma conexão TCP
- O HTTP/1.0 utiliza HTTP não persistente

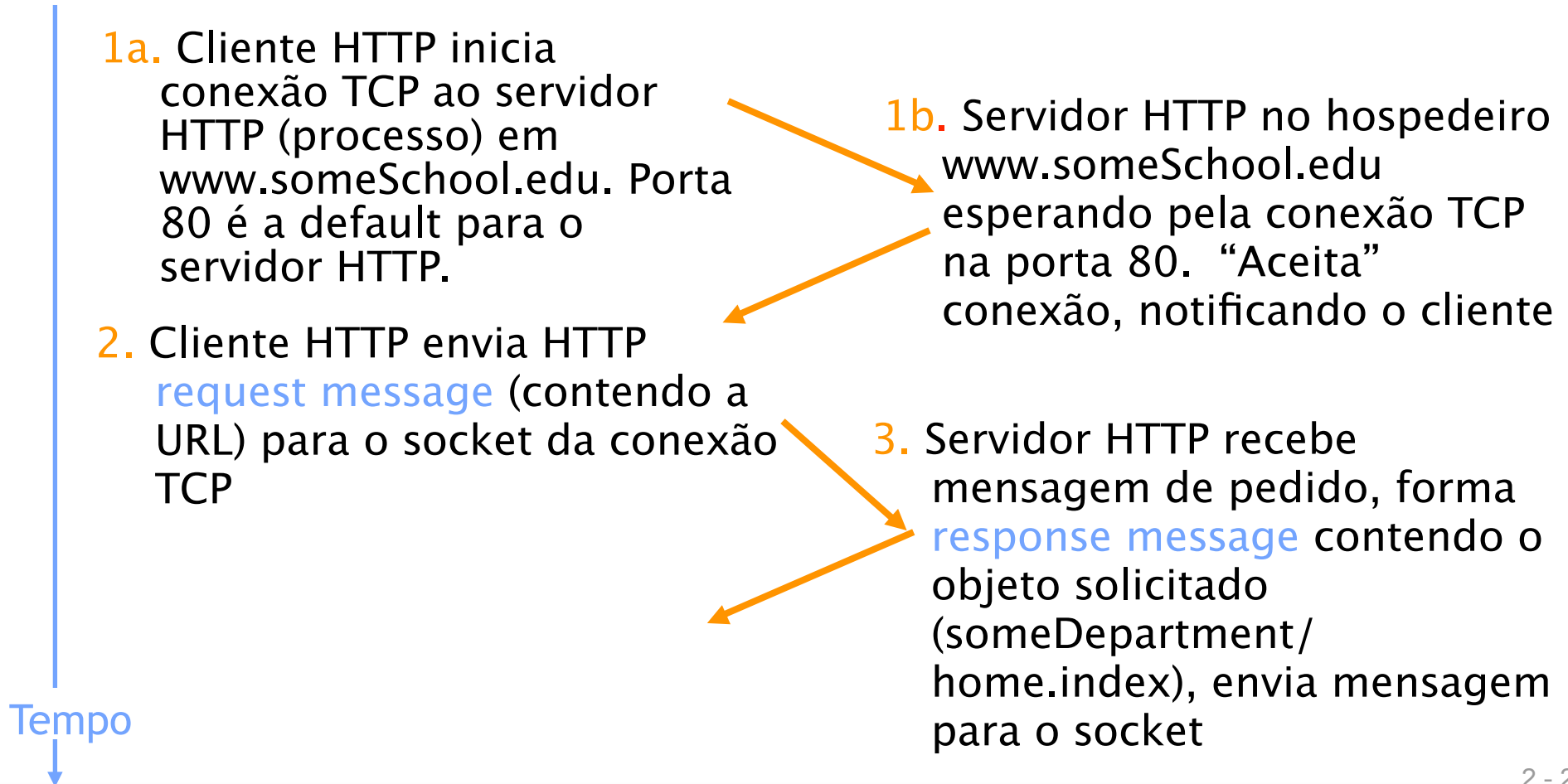
## HTTP persistente

- Múltiplos objetos podem ser enviados sobre uma conexão
- TCP entre o cliente e o servidor
- O HTTP/1.1 utiliza conexões persistentes em seu modo padrão



# HTTP não persistente

Usuário entra com a URL: `www.someSchool.edu/someDepartment/home.index`  
(contém texto, referências a 10 imagens jpeg)







# HTTP não persistente

---

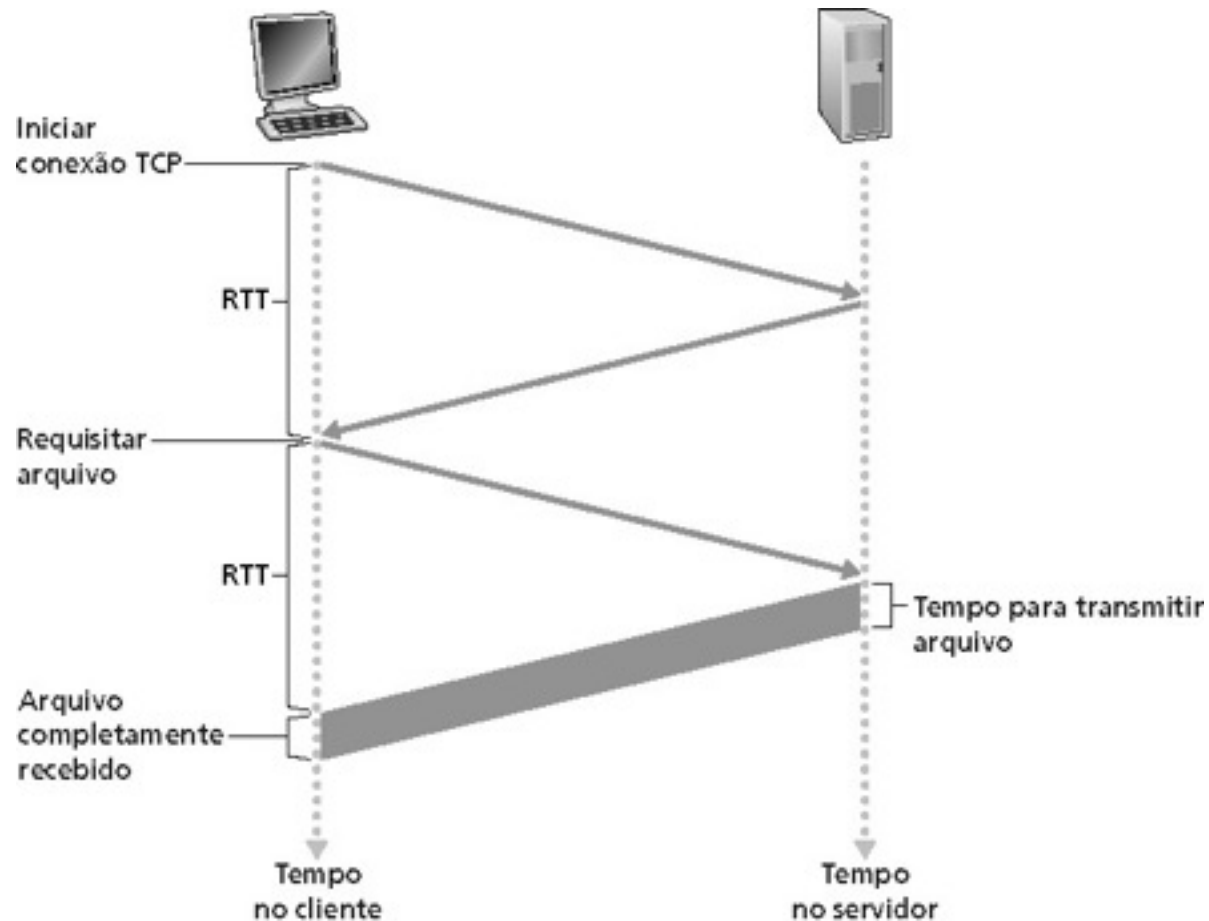
- Tempo ↓
5. Cliente HTTP recebe mensagem de resposta contendo o arquivo html, apresenta o conteúdo html. Analisando o arquivo html, encontra 10 objetos jpeg referenciados
  4. Servidor HTTP fecha conexão TCP.
  6. Passos 1–5 são repetidos para cada um dos 10 objetos jpeg.

# Modelagem do tempo de resposta

**Definição de RRT:** tempo para enviar um pequeno pacote que vai do cliente para o servidor e retorna.

## Tempo de resposta:

- Um RTT para iniciar a conexão TCP
- Um RTT para requisição HTTP e primeiros bytes da resposta
- Tempo de transmissão de arquivo



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



# HTTP persistente

---

## Características do HTTP persistente:

- Requer 2 RTTs por objeto
- OS deve manipular e alocar recursos do hospedeiro para cada conexão TCP  
Mas os browsers frequentemente abrem conexões TCP paralelas para buscar objetos referenciados

## HTTP persistente

- Servidor deixa a conexão aberta após enviar uma resposta
- Mensagens HTTP subsequentes entre o mesmo cliente/servidor são enviadas pela conexão

## Persistente sem pipelining:

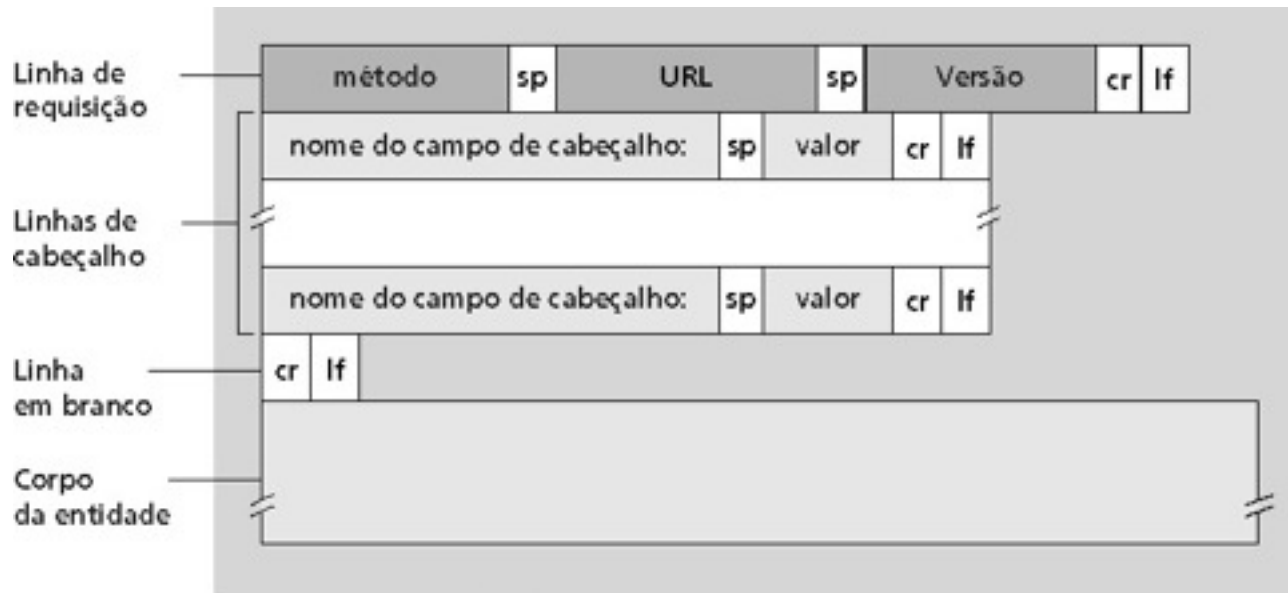
- O cliente emite novas requisições apenas quando a resposta anterior for recebida
- Um RTT para cada objeto referenciado

## Persistente com pipelining:

- Padrão no HTTP/1.1
- O cliente envia requisições assim que encontra um objeto referenciado
- Tão pequeno como um RTT para todos os objetos referenciados



# Mensagem HTTP request: formato geral



Obs.: cr = carriage return; lf = line feed



# Entrada de formulário

---

## Método Post:

- Página Web freqüentemente inclui entrada de formulário
- A entrada é enviada para o servidor no corpo da entidade

## Método URL:

- Utiliza o método GET
- A entrada é enviada no campo de URL da linha de requisição:

**[www.somesite.com/animalsearch?monkeys&banana](http://www.somesite.com/animalsearch?monkeys&banana)**



# Tipos de métodos

---

## HTTP/1.0

- GET
- POST
- HEAD
- pede para o servidor deixar o objeto requisitado fora da resposta

## HTTP/1.1

- GET, POST, HEAD
- PUT
- Envia o arquivo no corpo da entidade para o caminho especificado no campo de URL
- DELETE
- Apaga o arquivo especificado no campo de URL



# Códigos de status das respostas

---

Na primeira linha da mensagem de resposta servidor → cliente.

Alguns exemplos de códigos:

## 200 OK

- Requisição bem-sucedida, objeto requisitado a seguir nesta mensagem

## 301 Moved permanently

- Objeto requisitado foi movido, nova localização especificada a seguir nesta mensagem (Location:)

## 400 Bad request

- Mensagem de requisição não compreendida pelo servidor

## 404 Not Found

- Documento requisitado não encontrado neste servidor

## 505 HTTP version not supported



# Estado usuário-servidor: cookies

---

A maioria dos grandes Web sites utilizam cookies

## Quatro componentes:

- 1) Linha de cabeçalho do cookie na mensagem HTTP response
- 2) Linha de cabeçalho de cookie na mensagem HTTP request
- 3) Arquivo de cookie mantido no hospedeiro do usuário e manipulado pelo browser do usuário
- 4) Banco de dados **backend** no Web site

## Exemplo:

- Susan acessa a Internet sempre do mesmo PC
- Ela visita um site específico de e-commerce pela primeira vez
- Quando a requisição HTTP inicial chega ao site, este cria um ID único e uma entrada no banco de dados **backend** para este ID



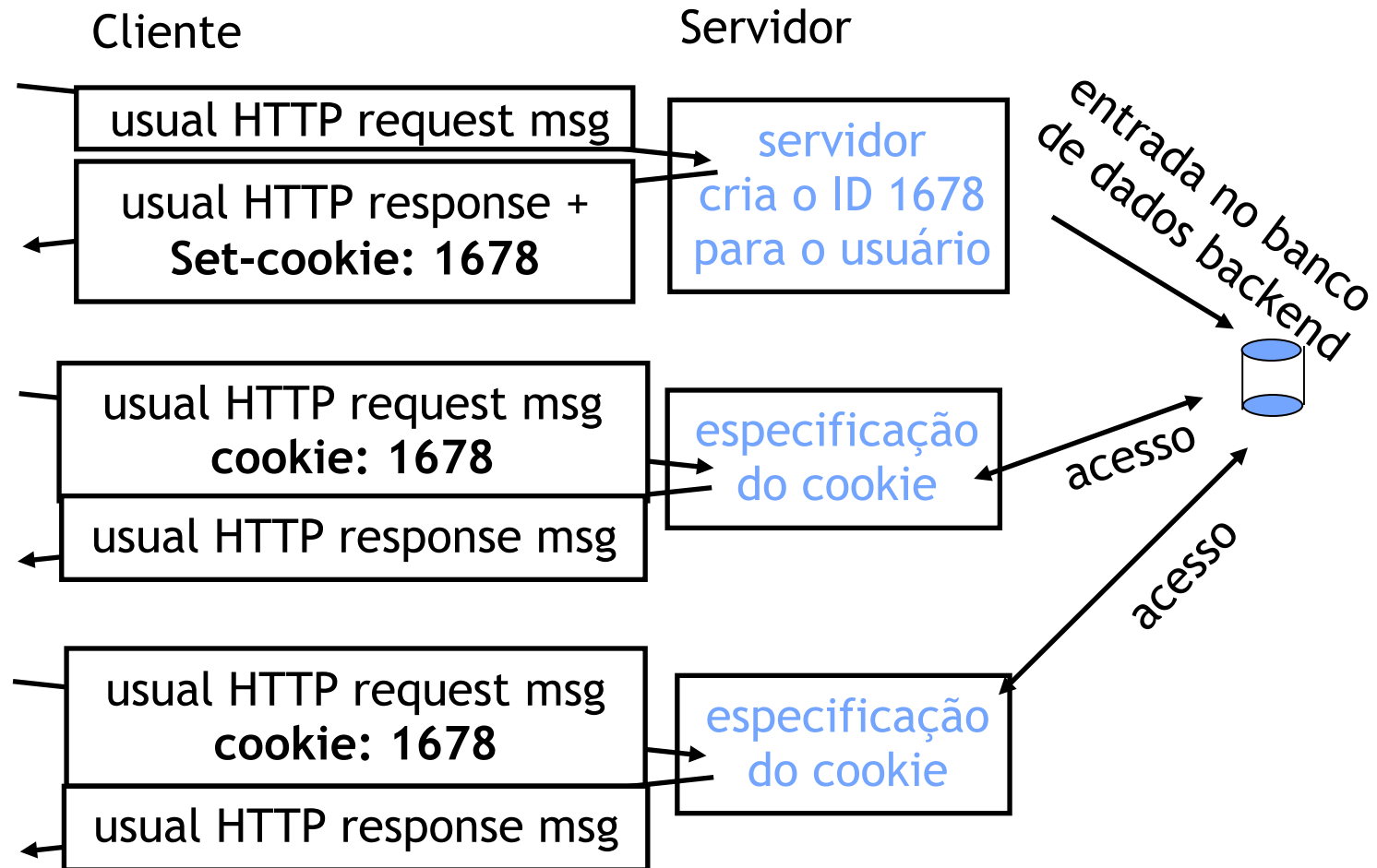
# Cookies: mantendo “estado”

**Cookie file**  
ebay: 8734

**Cookie file**  
amazon: 1678  
ebay: 8734

Uma semana depois:

**Cookie file**  
amazon: 1678  
ebay: 8734





# Cookies

---

## O que os cookies podem trazer:

- Autorização
- Cartões de compra
- Recomendações
- Estado de sessão do usuário (Web e-mail)

## ASIDE

### Cookies e privacidade:

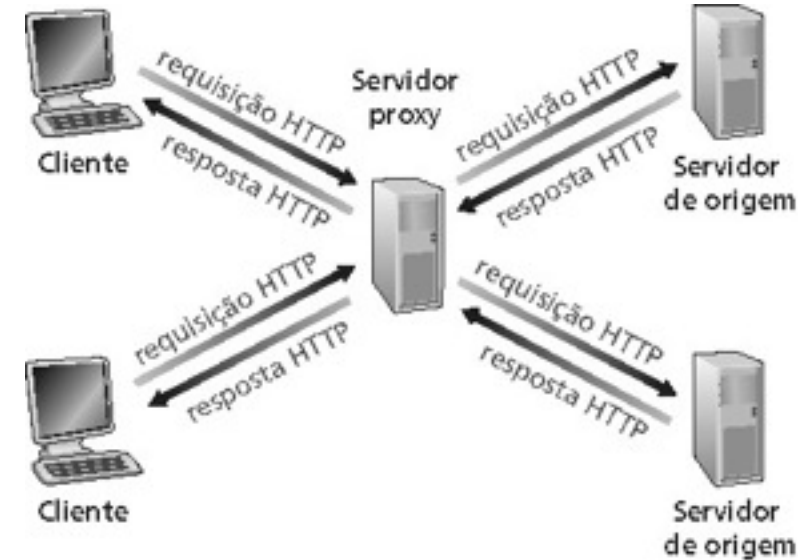
- Cookies permitem que sites saibam muito sobre você
- Você pode fornecer nome e e-mail para os sites
- Mecanismos de busca usam redirecionamento e cookies para saberem mais sobre você
- Companhias de propaganda obtêm informações por meio dos sites



# Web caches (proxy server)

**Objetivo:** atender o cliente sem envolver o servidor Web originador da informação

- Usuário configura o browser: acesso Web é feito por meio de um proxy
- Cliente envia todos os pedidos HTTP para o Web cache
  - Se o objeto existe no Web cache: Web cache retorna o objeto
  - Ou o Web cache solicita objeto do servidor original e então envia o objeto ao cliente





# Mais sobre Web caching

---

- O cache atua tanto no servidor como no cliente
- Tipicamente, o cache é instalado pelo ISP (universidade, companhia, ISP residencial)

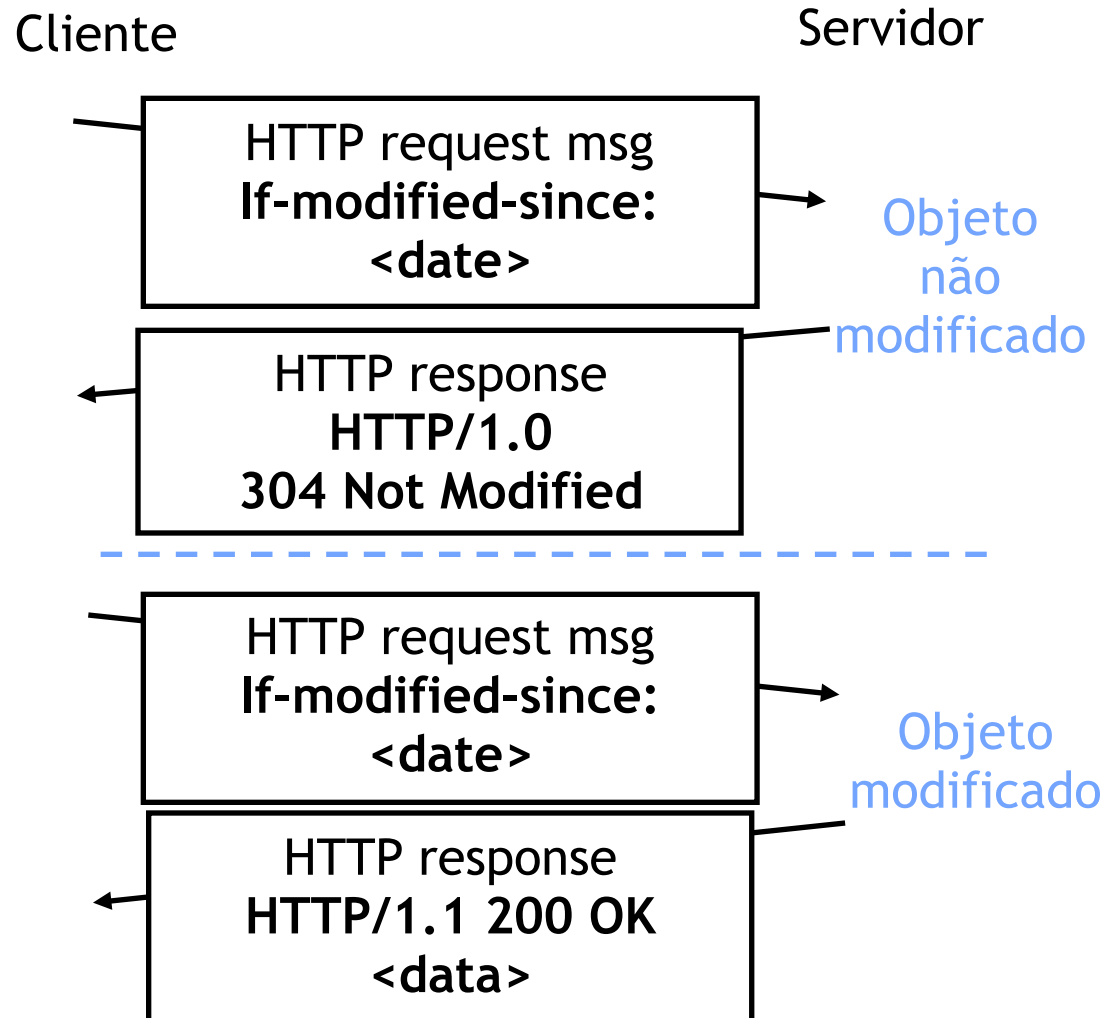
## Por que Web caching?

- Reduz o tempo de resposta para a requisição do cliente.
- Reduz o tráfego num enlace de acesso de uma instituição.
- A densidade de caches na Internet habilita os “fracos” provedores de conteúdo a efetivamente entregarem o conteúdo.



# GET condicional

- **Razão:** não enviar objeto se a versão que o cliente já possui está atualizada.
- Cliente: especifica data da versão armazenada no pedido HTTP
  - **If-modified-since: <date>**
- Servidor: resposta não contém objeto se a cópia é atualizada:  
**HTTP/1.0 304 Not Modified**





# Camada de aplicação

---

- 2.1 Princípios de aplicações de rede
- 2.2 Web e HTTP
- 2.3 FTP
- 2.4 Correio eletrônico
  - SMTP, POP3, IMAP
- 2.5 DNS



# FTP: o protocolo de transferência de arquivos

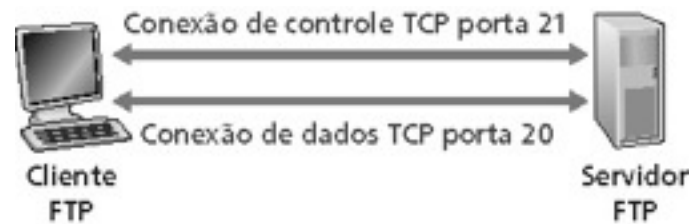


- Transferência de arquivos de e para o computador remoto
- Modelo cliente servidor
  - **Cliente:** lado que inicia a transferência (seja de ou para o lado remoto)
  - **Servidor:** hospedeiro remoto
- FTP: RFC 959
- FTP servidor: porta 21



# FTP: controle separado, conexões de dados

- Cliente FTP contata o servidor FTP na porta 21 especificando o TCP como protocolo de transporte
- Cliente obtém autorização pela conexão de controle
- Cliente procura o diretório remoto enviando comandos pela conexão de controle
- Quando o servidor recebe um comando para uma transferência de arquivo, ele abre uma conexão de dados TCP para o cliente
- Após a transferência de um arquivo, o servidor fecha a conexão
- Servidor abre uma segunda conexão de dados TCP para transferir outro arquivo
- Conexão de controle: “fora da banda”
- Servidor FTP mantém “estado”: diretório atual, autenticação anterior







# FTP comandos, respostas

---

## Exemplos de comandos:

- Envie um texto ASCII sobre canal de controle
- **USER *username***
- **PASS *password***
- **LIST** retorna listagem do arquivo no diretório atual
- **RETR filename** recupera (obtém) o arquivo
- **STOR filename** armazena o arquivo no hospedeiro remoto

## Exemplos de códigos de retorno

- Código de status e frase (como no HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**



# Camada de aplicação

---

- 2.1 Princípios de aplicações de rede
- 2.2 Web e HTTP
- 2.3 FTP
- 2.4 Correio eletrônico
  - SMTP, POP3, IMAP
- 2.5 DNS



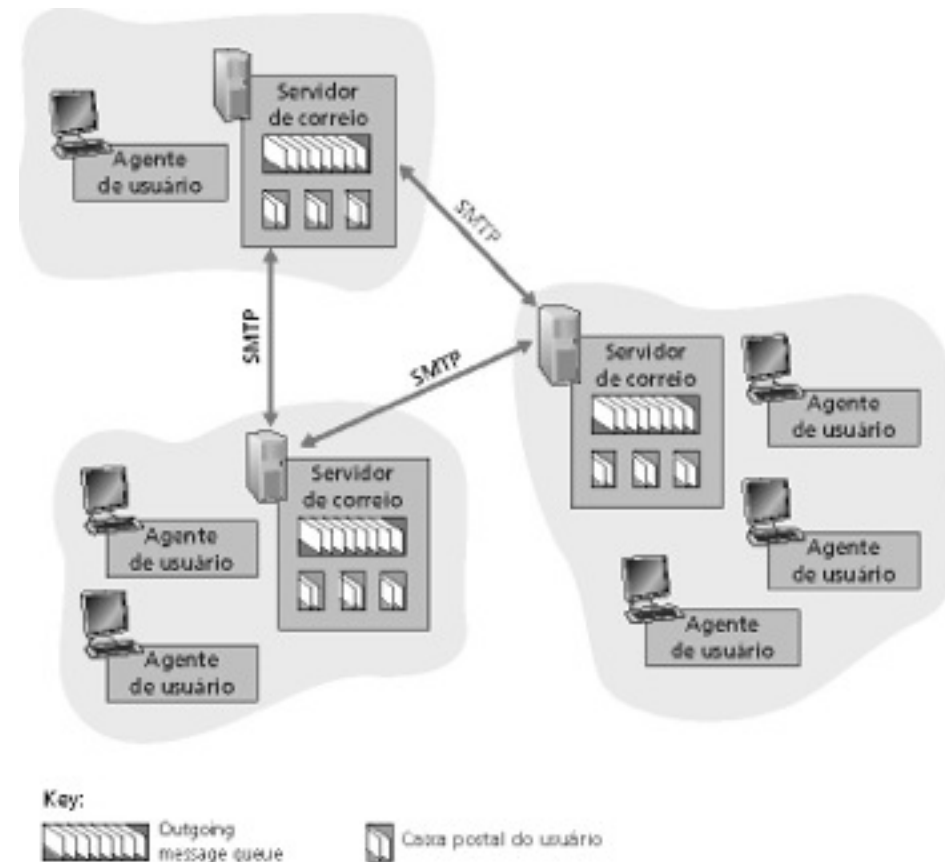
# Correio eletrônico

## Três componentes principais:

- Agentes de usuário
- Servidores de correio
- Simple mail transfer protocol: SMTP

## Agente de usuário

- “leitor de correio”
- Composição, edição, leitura de mensagens de correio
- Ex.: Eudora, Outlook, elm, Netscape Messenger
- Mensagens de entrada e de saída são armazenadas no servidor

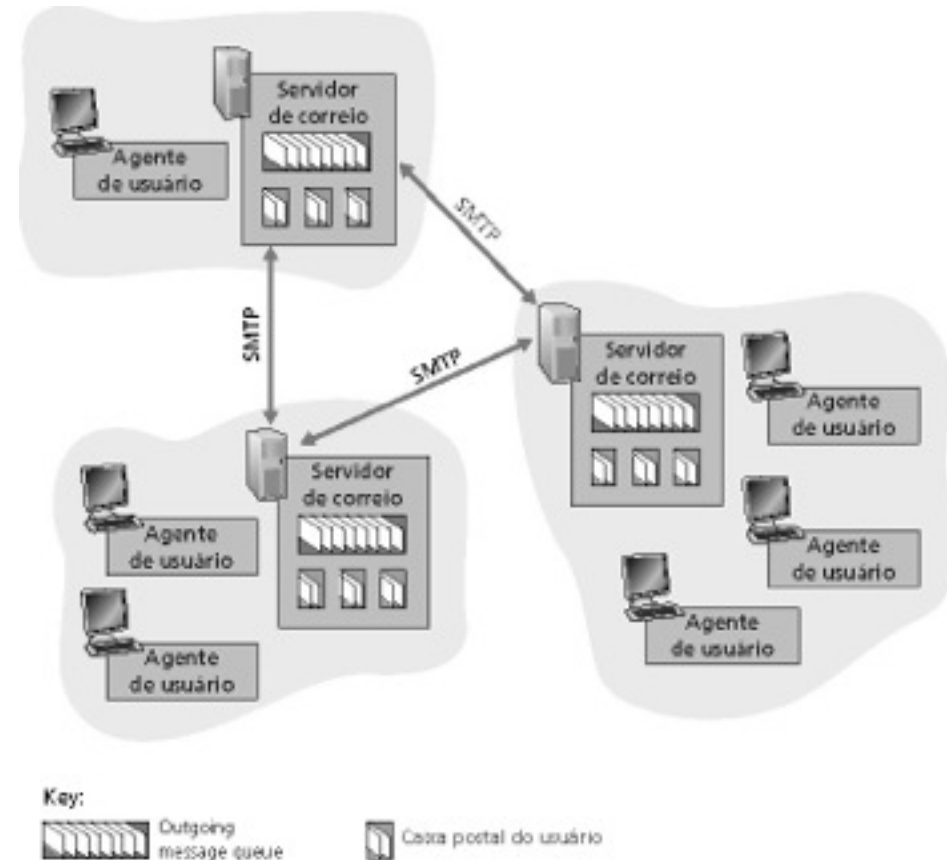


## Servidores de correio

- **Caixa postal** contém mensagens que chegaram (ainda não lidas) para o usuário
- **Fila de mensagens** contém as mensagens de correio a serem enviadas

**Protocolo SMTP** permite aos servidores de correio trocarem mensagens entre si

- **Cliente:** servidor de correio que envia
- **“servidor”:** servidor de correio que recebe





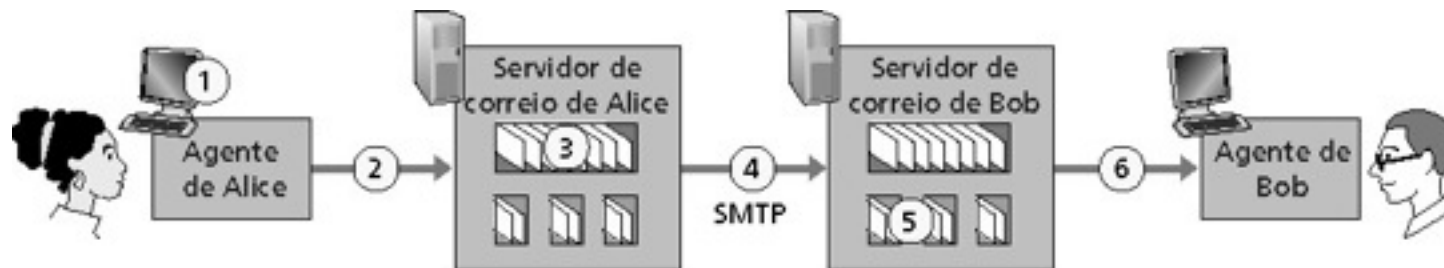
# Correio eletrônico: SMTP [RFC 821]

---

- Usa TCP para transferência confiável de mensagens de correio do cliente ao servidor, porta 25
- Transferência direta: servidor que envia para o servidor que recebe
- Três fases de transferência
  - Handshaking (apresentação)
  - Transferência de mensagens
  - Fechamento
- Interação comando/resposta
  - **Comandos:** texto ASCII
  - **Resposta:** código de status e frase
- Mensagens devem ser formatadas em código ASCII de 7 bits

# Cenário: Alice envia mensagem para Bob

- 1) Alice usa o agente de usuário (UA) para compor a mensagem e “para” bob@some school.edu
- 2) O agente de usuário dela envia a mensagem para o seu servidor de correio; a mensagem é colocada na fila de mensagens.
- 3) O lado cliente do SMTP abre uma conexão TCP com o servidor de correio do Bob.
- 4) O cliente SMTP envia a mensagem de Alice pela conexão TCP.
- 5) O servidor de correio de Bob coloca a mensagem na caixa de correio de Bob.
- 6) Bob invoca seu agente de usuário para ler a mensagem.



Legenda:



Fila de mensagens



Caixa postal do usuário



# Exemplo de interação SMTP

---

```
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: palavras finais

---

- SMTP usa conexões persistentes
- SMTP exige que as mensagens (cabeçalho e corpo) estejam em ASCII de 7 bits
- Servidor SMTP usa CRLF.CRLF para indicar o final da mensagem

## Comparação com HTTP:

- HTTP: pull
- E-mail: push
- Ambos usam comandos e respostas em ASCII, interação comando/resposta e códigos de status
- HTTP: cada objeto encapsulado na sua própria mensagem de resposta
- SMTP: múltiplos objetos são enviados numa mensagem multiparte



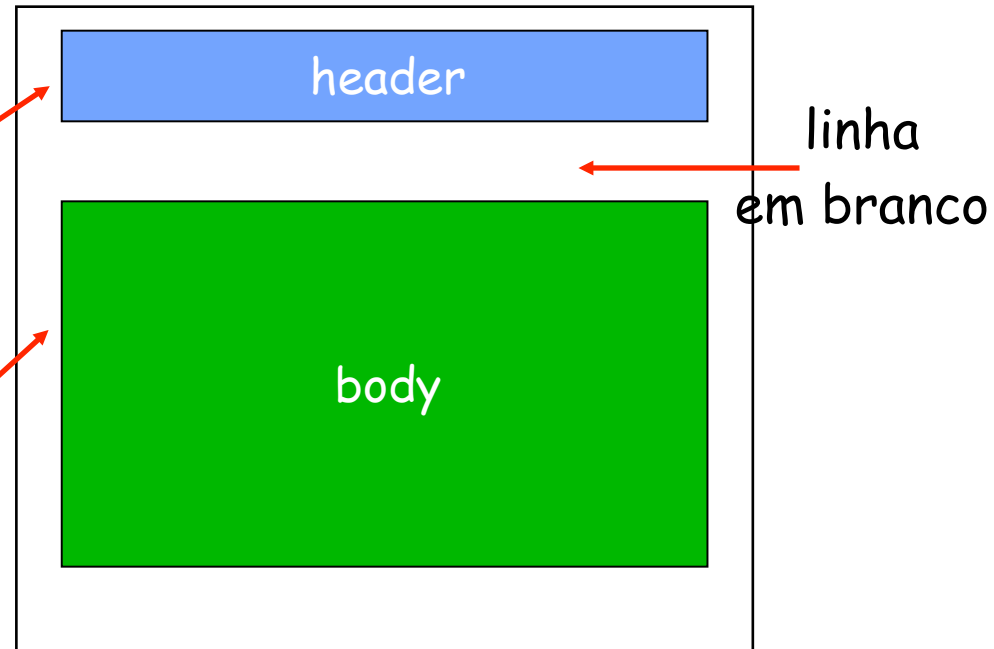


# Formato da mensagem de correio

SMTP: protocolo para trocar mensagens de e-mail

RFC 822: padrão para mensagens do tipo texto:

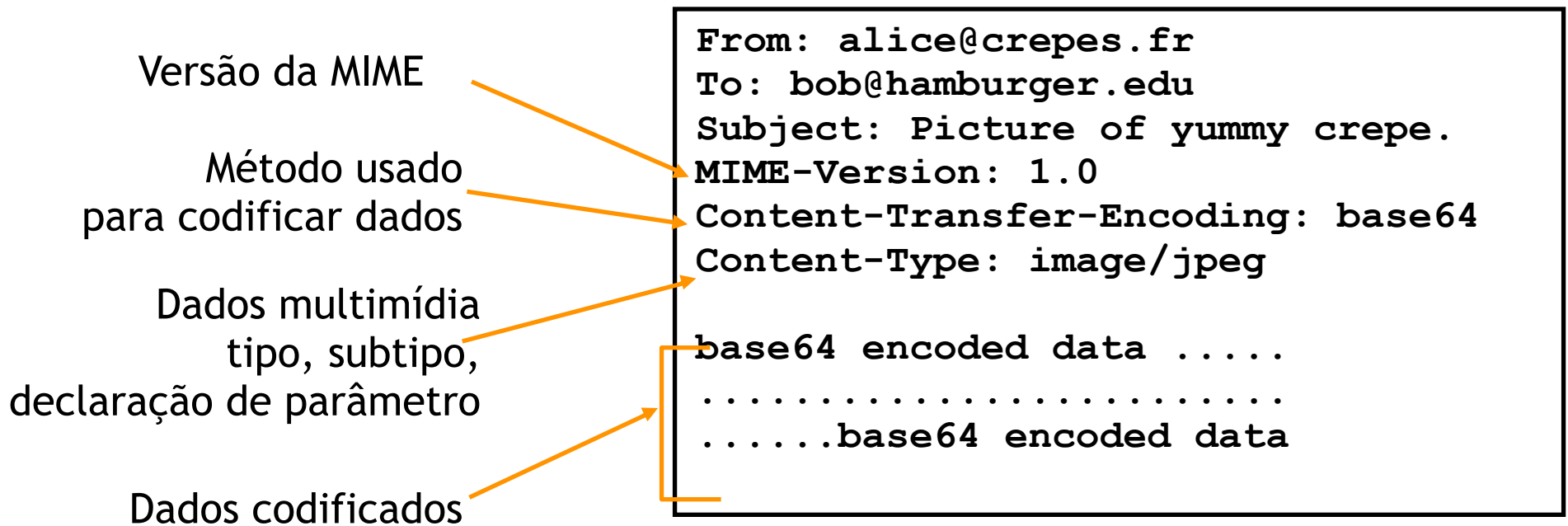
- linhas de cabeçalho, ex.:
  - To:
  - From:
  - Subject:
- diferente dos comandos HTTP
- corpo
  - a “mensagem”, ASCII somente com caracteres





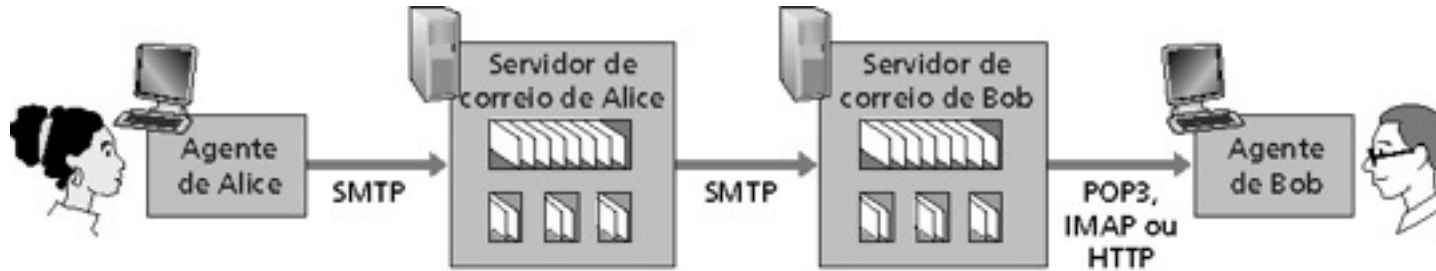
# Formato das mensagens: extensões multimídia

- MIME: multimedia mail extension, RFC 2045, 2056
- Linhas adicionais no cabeçalho declaram o tipo de conteúdo MIME





# Protocolos de acesso ao correio



- SMTP: entrega e armazena no servidor do destino
- Protocolo de acesso: recupera mensagens do servidor
  - POP: Post Office Protocol [RFC 1939]
    - Autorização (agente <-->servidor) e download
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Maiores recursos (mais complexo)
    - Manipulação de mensagens armazenadas no servidor
  - HTTP: Hotmail , Yahoo! Mail etc.



# Protocolo POP3

## Fase de autorização

- comandos do cliente:
  - **user**: declara nome do usuário
  - **pass**: password

respostas do servidor

- **+OK**
- **-ERR**

## Fase de transação, cliente:

- **list**: lista mensagens e tamanhos
- **retr**: recupera mensagem pelo número
- **dele**: apaga
- **quit**

```
S: +OK POP3 server ready
C: user alice
S: +OK
C: pass hungry
S: +OK user successfully
logged on
```

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



# POP3 (mais) e IMAP

---

## Mais sobre POP3

- O exemplo anterior usa o modo “download-and-delete”
- Bob não pode reler o e-mail se ele trocar o cliente
- “download-and-keep”: cópias das mensagens em clientes diferentes
- POP3 é stateless através das sessões

## IMAP

- Mantém todas as mensagens em um lugar: o servidor
- Permite que o usuário organize as mensagens em pastas
- IMAP mantém o estado do usuário através das sessões:
  - Nomes das pastas e mapeamentos entre os IDs da mensagem e o nome da pasta



# Camada de aplicação

---

- 2.1 Princípios de aplicações de rede
- 2.2 Web e HTTP
- 2.3 FTP
- 2.4 Correio eletrônico
  - SMTP, POP3, IMAP
- 2.5 DNS



# DNS: Dominain Name System

---

**Pessoas:** muitos identificadores:

- RG, nome, passaporte

**Internet hospedeiros, roteadores:**

- Endereços IP (32 bits) - usados para endereçar datagramas
- “nome”, ex.: gaia.cs.umass.edu - usados por humanos

**P.:** Relacionar nomes com endereços IP?

**Domain Name System:**

- **Base de dados distribuída** implementada numa hierarquia de muitos **servidores de nomes**
- **Protocolo de camada de aplicação** hospedeiro, roteadores se comunicam com servidores de nomes para **resolver** nomes (translação nome/ endereço)
  - Nota: função interna da Internet, implementada como protocolo da camada de aplicação
  - Complexidade na “borda” da rede



# DNS

---

## DNS services

- Nome do hospedeiro para tradução de endereço IP
- Hospedeiro aliasing
  - Nomes canônicos e alias
    - mail server aliasing
    - distribuição de carga
  - Servidores Web replicados: estabelece o endereço IP para um nome canônico

## Por que não centralizar o DNS?

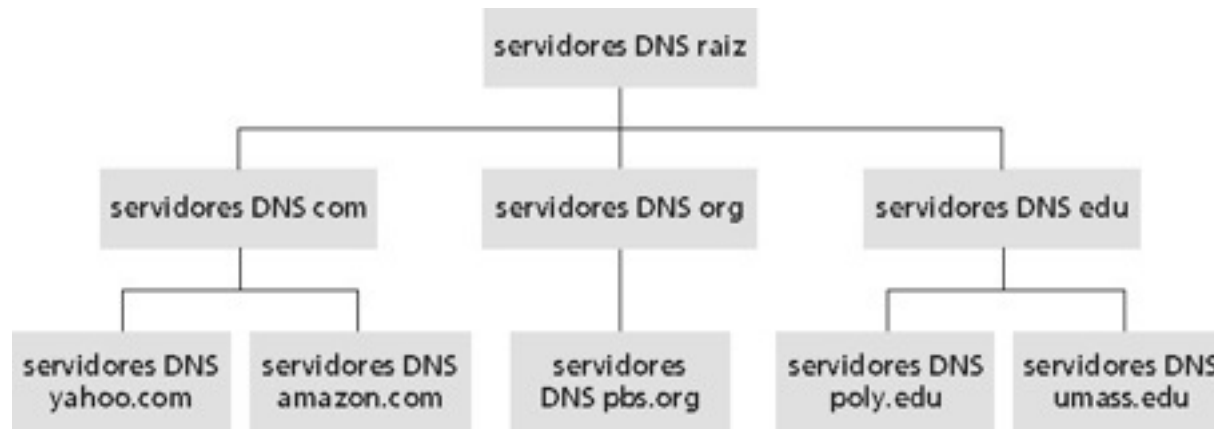
- Ponto único de falha
- Volume de tráfego
- Base centralizada de dados distante
- Manutenção

**Não é escalável!**





# Base de dados distribuída, hierárquica



Cliente quer o IP para [www.amazon.com](http://www.amazon.com); 1ª aprox.:

- Cliente consulta um servidor de raiz para encontrar o servidor DNS com
- Cliente consulta o servidor DNS com para obter o servidor DNS amazon.com
- Cliente consulta o servidor DNS amazon.com para obter o endereço IP para [www.amazon.com](http://www.amazon.com)



# DNS: servidores de nomes raiz

- São contatados pelos servidores de nomes locais que não podem resolver um nome
- Servidores de nomes raiz:
  - Buscam servidores de nomes autorizados se o mapeamento do nome não for conhecido
  - Conseguem o mapeamento
  - Retornam o mapeamento para o servidor de nomes local



Existem 13 servidores de nomes raiz no mundo

# Servidores TLD e autoritários

---

**Servidores top-level domain (TLD):** responsáveis pelos domínios com, org, net, edu etc e todos os domínios **top-level** nacionais uk, fr, ca, jp.

- Network Solutions mantém servidores para o TLD “com” TLD
- Educause para o TLD “edu”

**Servidores DNS autorizados:** servidores DNS de organizações, provêm nome de hospedeiro autorizado para mapeamentos IP para servidores de organizações (ex.: Web e mail).

- Podem ser mantidos por uma organização ou provedor de serviços



# Servidor de nomes local

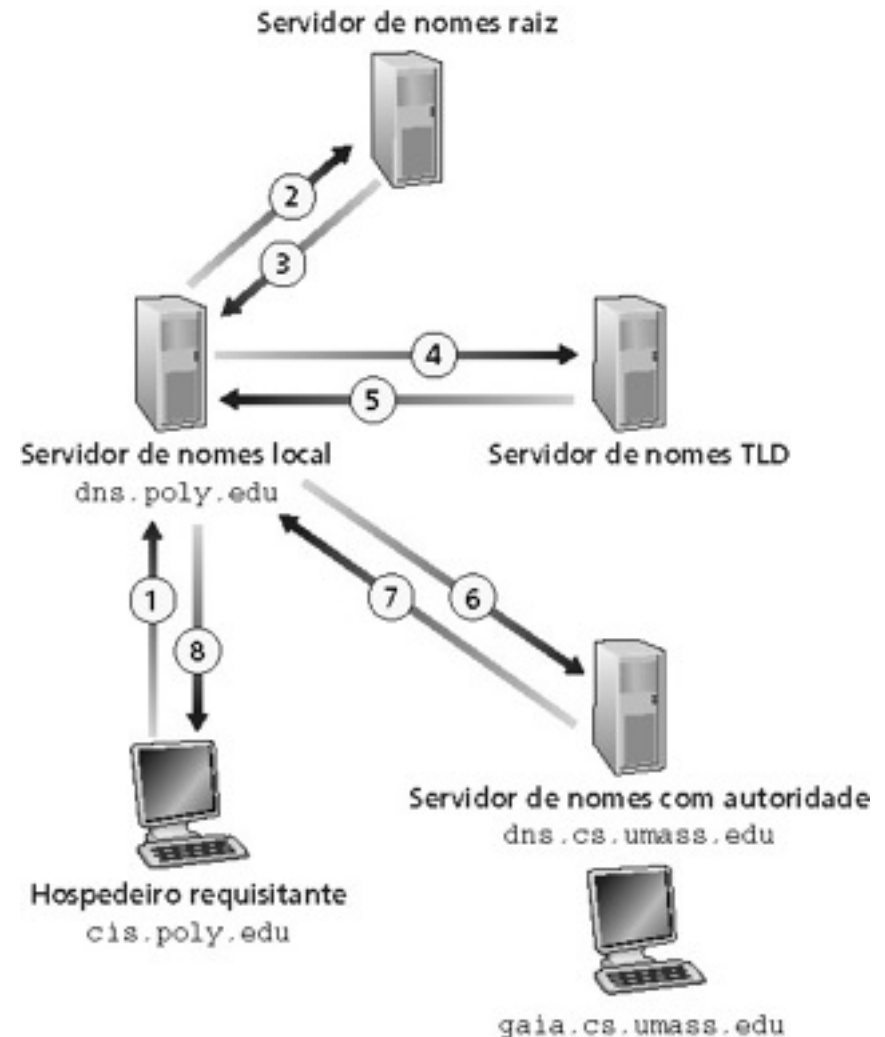
---

- Não pertence estritamente a uma hierarquia
- Cada ISP (ISP residencial, companhia, universidade) possui um
  - Também chamado de “servidor de nomes default”
- Quando um hospedeiro faz uma pergunta a um DNS, a pergunta é enviada para seu servidor DNS local
  - Age como um proxy, encaminhando as perguntas para dentro da hierarquia



# Exemplo

- O hospedeiro em cis.poly.edu quer o endereço IP para gaia.cs.umass.edu





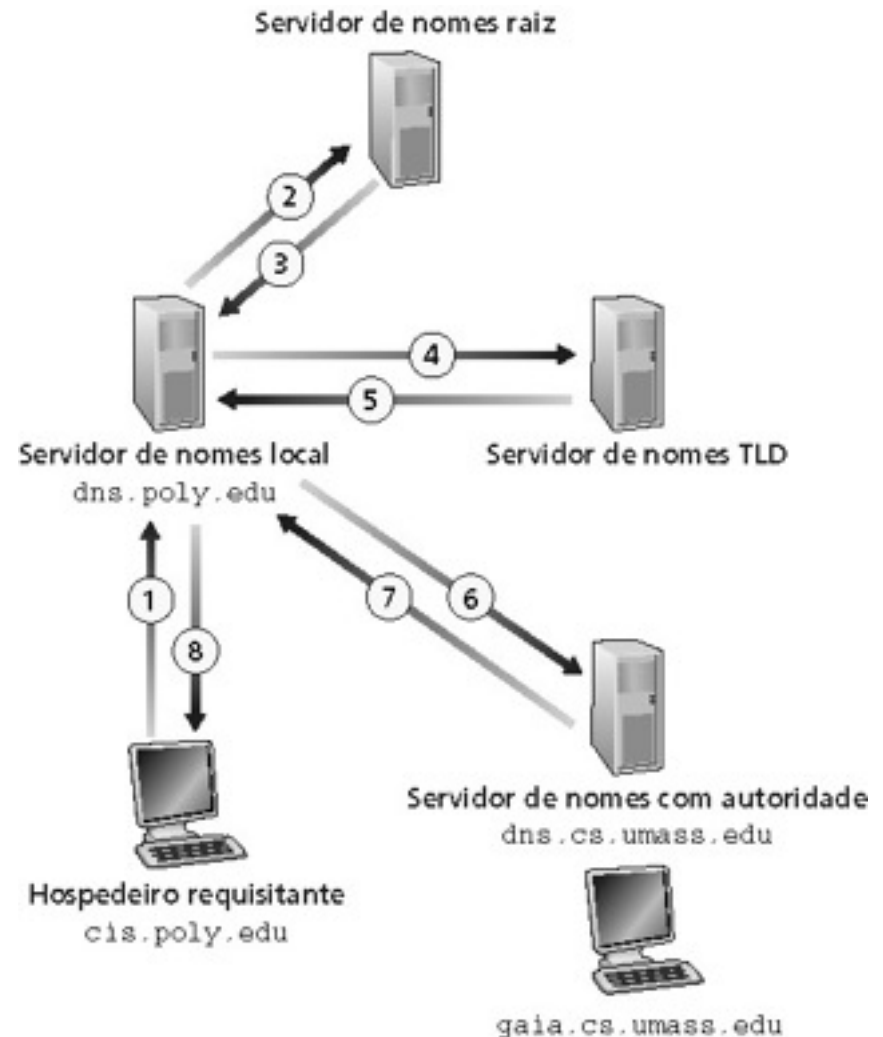
# Consultas recursivas

## Consulta recursiva:

- Transfere a tarefa de resolução do nome para o servidor de nomes consultado
- Carga pesada?

## Consulta encadeada:

- Servidor contatado responde com o nome de outro servidor de nomes para contato
- “eu não sei isto, mas pergunte a este servidor”





# DNS: armazenando e atualizando registros

---

Uma vez que um servidor de nomes apreende um mapeamento, ele armazena o mapeamento num registro do tipo **cache**

- Registro do cache tornam-se obsoletos (desaparecem) depois de um certo tempo
- Servidores TLD são tipicamente armazenados em cache nos servidores de nome locais

Mecanismos de atualização e notificação estão sendo projetados pelo IETF

- RFC 2136
- <http://www.ietf.org/html.charters/dnsind-charter.html>



# Registros do DNS

**DNS:** base de dados distribuída que armazena registros de recursos (**RR**)

formato dos RR: (name, value, type,ttl)

- Type = A
  - **name** é o nome do computador
  - **value** é o endereço IP
- Type = CNAME
  - **name** é um “apelido” para algum nome “canônico” (o nome real)  
www.ibm.com é realmente  
servereast.backup2.ibm.com
  - **value** é o nome canônico
- Type = NS
  - **name** é um domínio (ex.:  
foo.com)
  - **value** é o endereço IP do servidor de nomes autorizados para este domínio
- Type = MX
  - **value** é o nome do servidor de correio associado com **name**



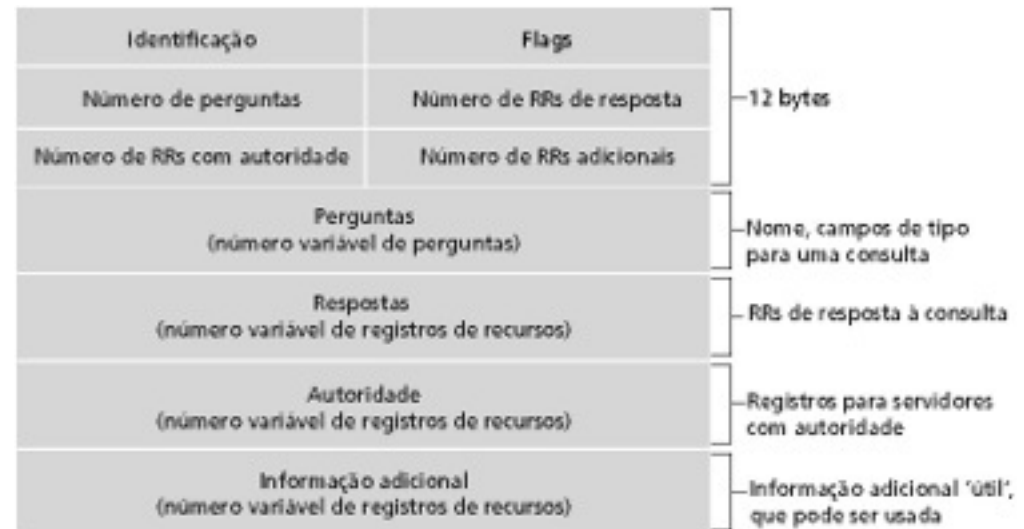


# DNS: protocolo e mensagem

**Protocolo DNS:** mensagem de **consulta** e **resposta** , ambas com o mesmo **formato de mensagem**

## Cabeçalho da msg

- **Identificação:** número de 16 bits para consulta, resposta usa o mesmo número
- **Flags:**
  - Consulta ou resposta
  - Recursão desejada
  - Recursão disponível
  - Resposta é autorizada





# Camada de aplicação

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

# Camada de aplicação

---

- Exemplo: empresa recém-criada “Network Utopia”
- Registrar o nome networkutopia.com num “registrar” (ex.: Network Solutions)
  - É necessário fornecer ao registrar os nomes e endereços IP do seu servidor nomes autorizados (primário e secundário)
  - Registrar insere dois RRs no servidor TLD do domínio com:

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

- No servidor autorizado, inserir um registro Tipo A para www.networkutopia.com e um registro Tipo MX para networkutopia.com
- Como as pessoas obtêm o endereço IP do seu Web site?

## Inserindo registros no DNS



# Resumo

---

## Nosso estudo de aplicações está completo agora!

- Arquiteturas de aplicação
  - Cliente-servidor
  - P2P
  - Híbrida
- Exigências dos serviços de aplicação:
  - Confiabilidade, banda passante, atraso
- Modelo do serviço de transporte da Internet l
  - Orientado à conexão, confiável: TCP
  - Não confiável, datagramas: UDP
- Protocolos específicos:
  - HTTP
  - FTP
  - SMTP, POP, IMAP
  - DNS



# Resumo

---

## Mais importante: características dos protocolos

- Típica troca de mensagens comando/resposta:
  - Cliente solicita informação ou serviço
  - Servidor responde com dados e código de status
- Formatos das mensagens:
  - Cabeçalhos: campos que dão informações sobre os dados
  - Dados: informação sendo comunicada
- Controle vs. dados
  - In-band, out-of-band
- Centralizado vs. descentralizado
- Stateless vs. stateful
- Transferência de mensagens confiável vs. não confiável
- “complexidade na borda da rede”