




Redes de Computadores

Camada de Aplicação

Professor: Reinaldo Gomes
reinaldo@dsc.ufcg.edu.br



Camada de aplicação

- Princípios de aplicações em rede de computadores
- Web e HTTP
- Correio eletrônico
 - SMTP, POP3, IMAP
- DNS



Algumas aplicações de rede

- Correio Eletrônico (e-mail)
- Hipertexto (<http://www...>)
- Mensagem instantânea
- Login remoto (e.g., ssh)
- Compartilhamento de Arquivos Entre-Pares (*P2P file sharing*)
- Jogos multi-usuário em Rede
- *Streaming stored videocliques*
- Telefonia via Internet (VoIP)
- Videoconferência em tempo real
- Grades Computacionais (*grid computing*)



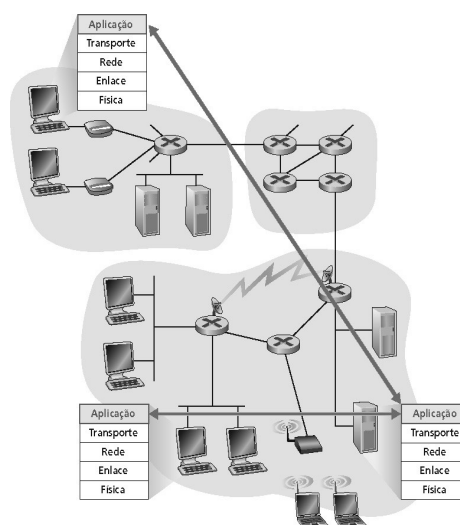
Criando uma nova aplicação de rede


Escrever programas que

- Executem sobre diferentes sistemas finais e
- Se comuniquem através de uma rede.
- Ex.: *Software* de servidor Web se comunicando com *software* do navegador (*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 em rede de computadores
- 2.2 Web e HTTP
- 2.3 FTP
- 2.4 Correio eletrônico
 - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 Compartilhamento de arquivos P2P



Arquiteturas de aplicação

6

- Cliente-servidor
- Entre-Pares (Peer-to-peer, P2P)
- Híbrida de cliente-servidor e P2P

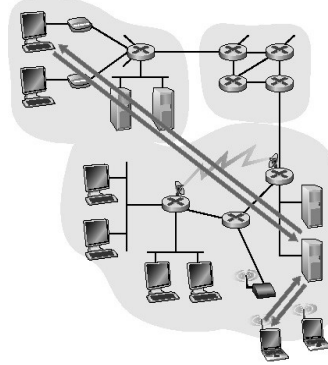
Arquitetura cliente-servidor

Servidor:

- Hospedeiro 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

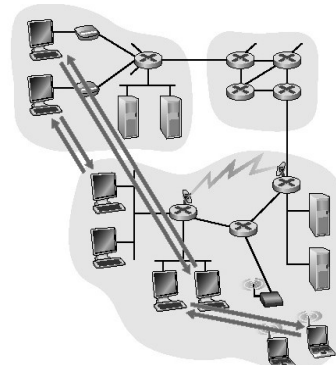


a. Aplicação cliente-servidor

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

9

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 de presença é centralizada:
 - Usuário registra seu endereço IP com o servidor central quando fica *on-line*
 - Usuário contata o servidor central para encontrar endereços IP dos “amigos”



O protocolo da camada 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, DNS, SSH

Protocolos proprietários:

- Ex.: KaZaA, BitTorrent



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, sessão remota) exigem transferência de dados 100% confiável

Temporização

- Algumas aplicações (ex.: telefonia IP, 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 aplicações comuns

12

Aplicação	Perdas	Banda	Sensível ao atraso
file transfer	sem perdas	elástica	não
e-mail	sem perdas	elástica	não
Web documents	sem perdas	elástica	não
real-time áudio/vídeo	tolerante	áudio: 5 Kb-1Mb vídeo: 10Kb-5Mb	sim, 100's mseg
stored áudio/vídeo	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 processos 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, controle de fluxo e de congestionamento, garantia de temporização e de banda mínima.



Aplicação e protocolos de transporte da Internet

14

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



Camada de aplicação

- Princípios de aplicações em rede de computadores
- Web e HTTP
- Correio eletrônico
 - SMTP, POP3, IMAP
- DNS



Visão geral do HTTP

HTTP: *hypertext transfer protocol*

- Protocolo da camada de aplicação da Web
- Modelo cliente/servidor
 - Cliente: navegador 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”

- Por *default*, o servidor não mantém informação sobre os pedidos passados pelos clientes

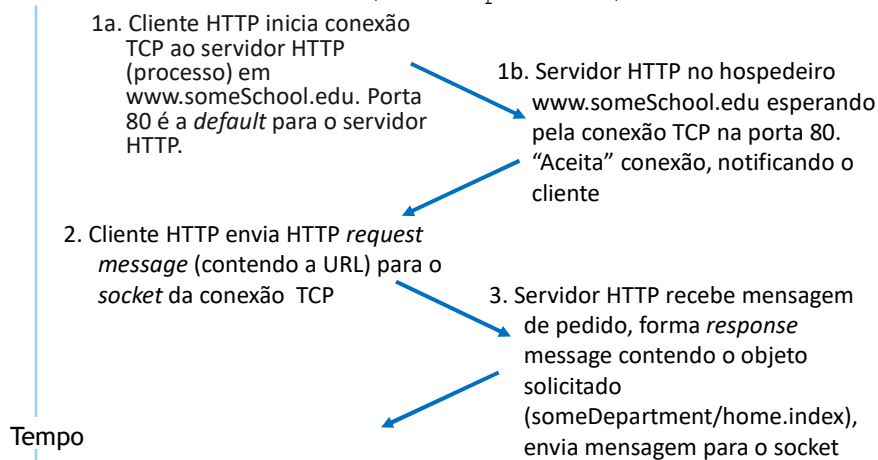
Protocolos que mantêm informações de “estado” são complexos!

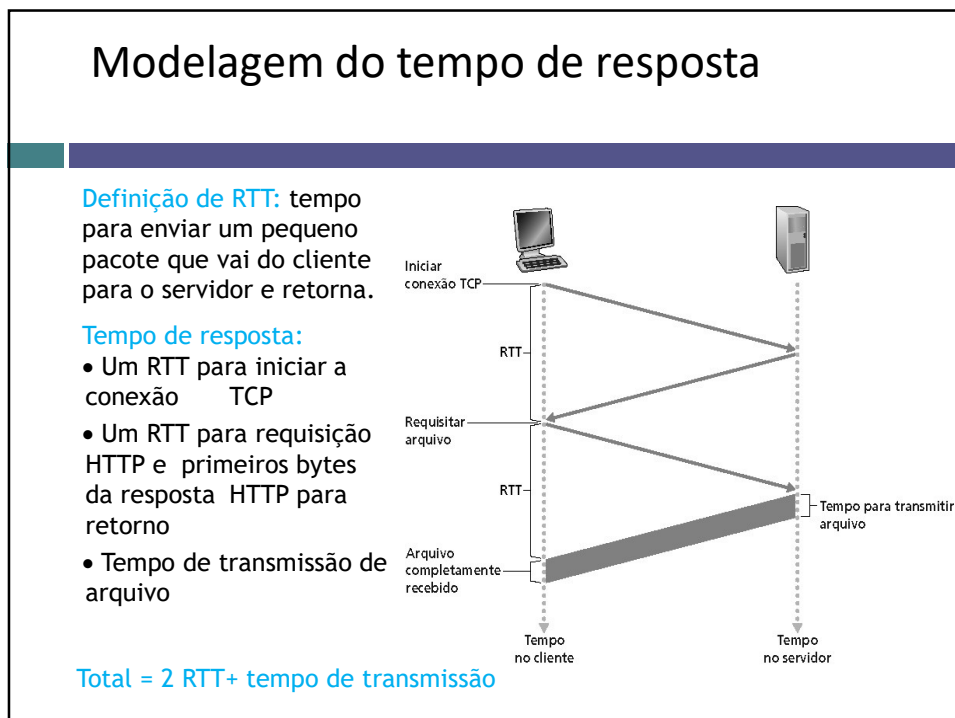
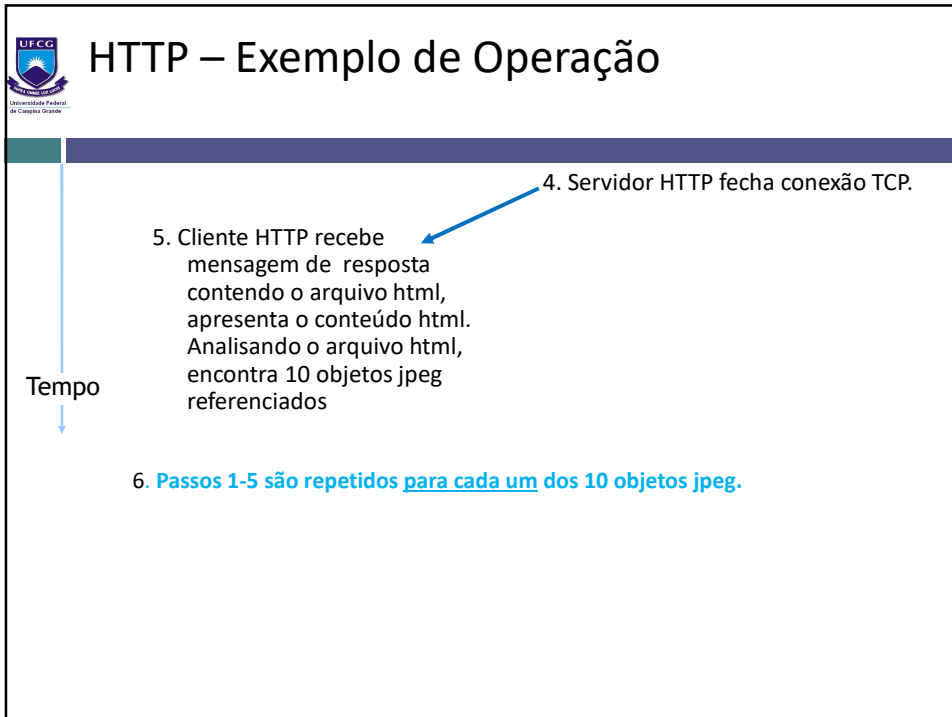
- Histórico do passado (estado) deve ser mantido
- Se o servidor/cliente quebra, suas visões de “estado” podem ser inconsistentes, devendo ser reconciliadas



HTTP – Exemplo de Operação

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







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 persistente

Características do HTTP não persistente:

- Requer 2 RTTs por objeto
- Sistema Operacional 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 curto quanto um RTT para todos os objetos referenciados



Mensagem HTTP request

- Dois tipos de mensagens HTTP: **request**, **response**
- HTTP request message:
 - ASCII (formato legível para humanos)

Linha de pedido
 (comandos GET, POST (para
 Formulários),
 HEAD (para depuração))

Linhas de
 cabeçalho

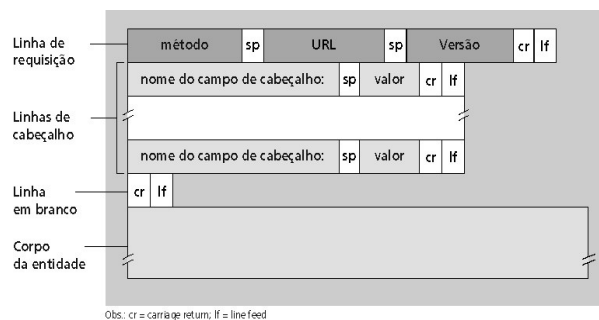
Carriage return,
 line feed
 indica fim da mensagem

GET /somedir/page.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language: fr
 (extra carriage return, line feed)



Mensagem HTTP request: formato geral

24





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`




Tipos de métodos

HTTP/1.0

- GET
- POST
- HEAD
 - Pedir 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



Mensagem HTTP response

```


HTTP/1.0 200 OK
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 .....

Content-Length: 6821
Content-Type: text/html

data data data data data ...
  
```

Diagram labels and arrows:

- Linha de status** (protocolo, código de status, frase de status) points to the first line: `HTTP/1.0 200 OK`
- Linhas de cabeçalho** points to the header lines: `Date: Thu, 06 Aug 1998 12:00:15 GMT`, `Server: Apache/1.3.0 (Unix)`, `Last-Modified: Mon, 22 Jun 1998`, `Content-Length: 6821`, and `Content-Type: text/html`
- Dados, ex.: arquivo html** points to the body: `data data data data data ...`



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 sítios Web utilizam *cookies*

Quatro componentes:

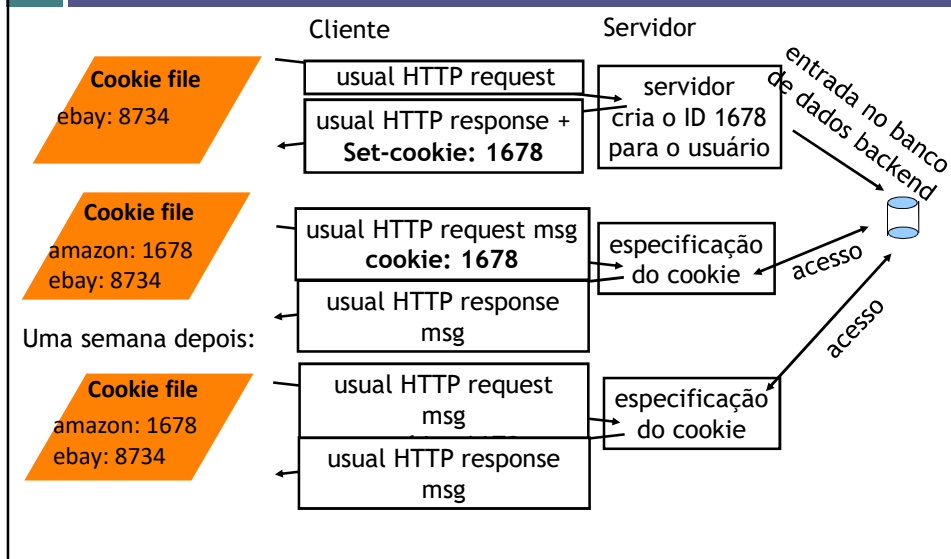
- 1) Linha de cabeçalho do *cookie* na mensagem HTTP response
- 2) Linha de cabeçalho de *cookie* na mensagem HTTP request
- 3) Arquivo de *cookie* mantido no hospedeiro do usuário e manipulado pelo *browser* do usuário
- 4) Banco de dados *backend* no sítio Web

Exemplo:

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



Cookies: mantendo “estado”





Cookies

O que os cookies podem trazer:

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

Cookies e privacidade:

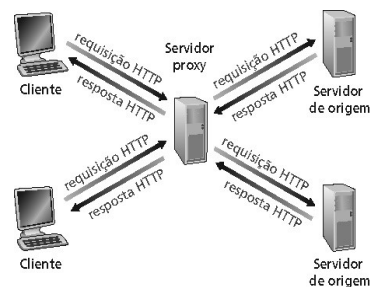
- Cookies permitem que *sites* saibam muito sobre você
- Você pode fornecer nome e e-mail para os *sites*
- Mecanismos de busca usam redirecionamento e cookies para saberem mais sobre você
- Companhias de propaganda obtêm informações por meio dos *sites*



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 procurador (*proxy*).
- Cliente envia todos os pedidos HTTP para o *proxy*
 - Se o objeto existe na *cache* do procurador: o procurador retorna o objeto
 - Caso contrario, o procurador solicita o(s) objeto(s) do servidor original, e então envia o(s) objeto(s) ao cliente





Mais sobre Web caching

- A *cache* atua tanto no servidor como no cliente
- Tipicamente, a *cache* é instalada pelo ISP (universidade, companhia, ISP residencial)

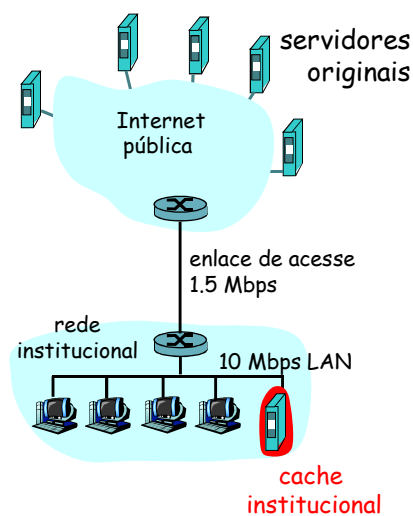
Por que Web caching?

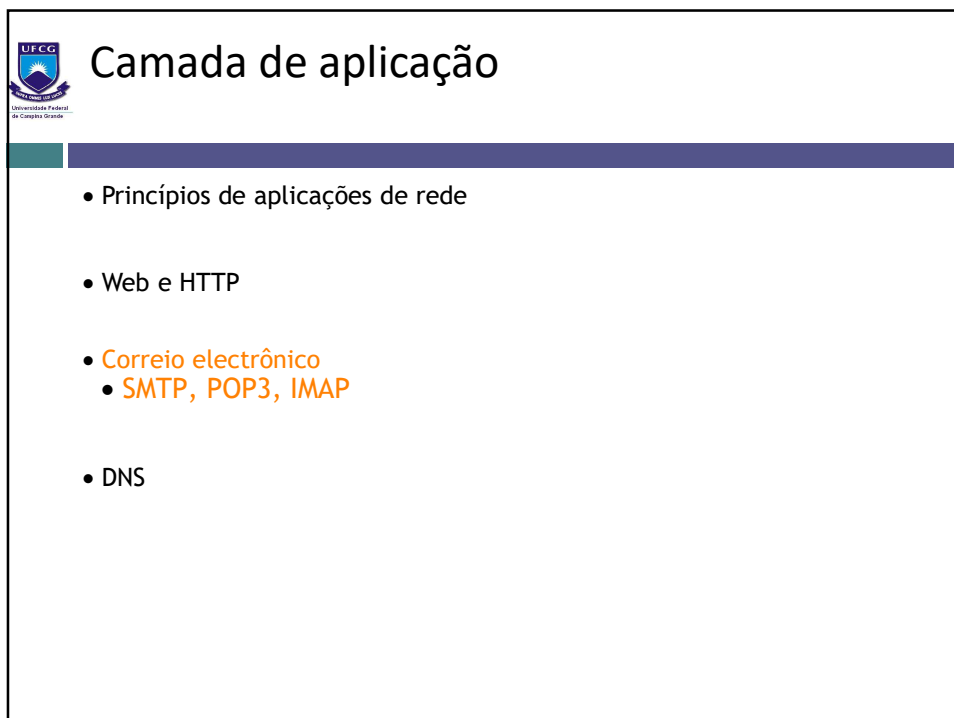
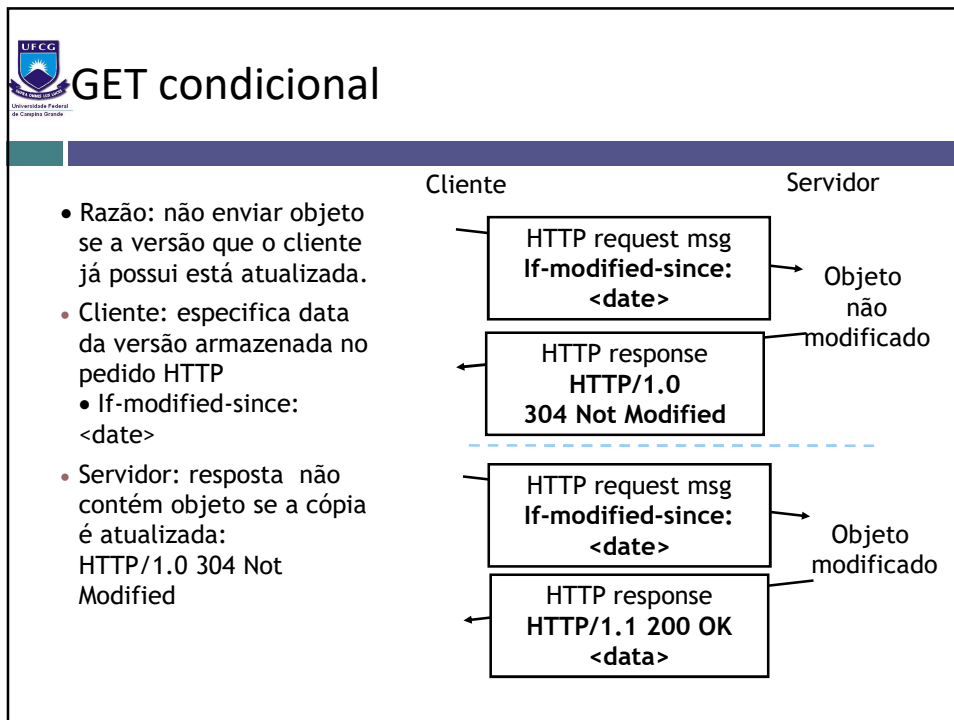
- Reduz o tempo de resposta para a requisição do cliente.
- Reduz o tráfego em um enlace de acesso de uma instituição.
- A densidade de *caches* na Internet habilita os “fracos” provedores de conteúdo a efetivamente entregarem o conteúdo (mas fazendo *P2P file sharing*)



Porque Web Caching?

- armazenamento está “perto” do cliente (ex., na mesma rede)
- menor tempo de resposta
- reduz o tráfego para servidor distante
 - ▣ links externos podem ser caros e facilmente congestionáveis







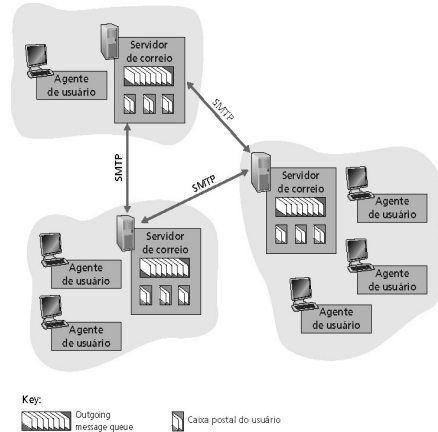
Correio eletrônico

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, Thunderbird
- Mensagens de entrada e de saída são armazenadas no servidor



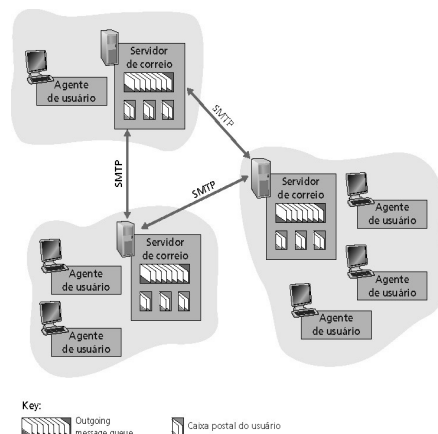
Correio eletrônico: servidores de correio

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





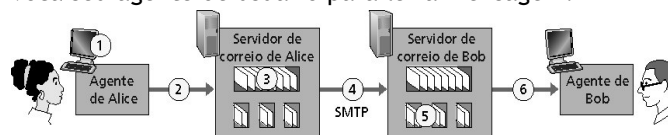
Correio eletrônico: SMTP [RFC 821]

- 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 (*User Agent*) para compor a mensagem “para” bob@someschool.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

41

```

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:

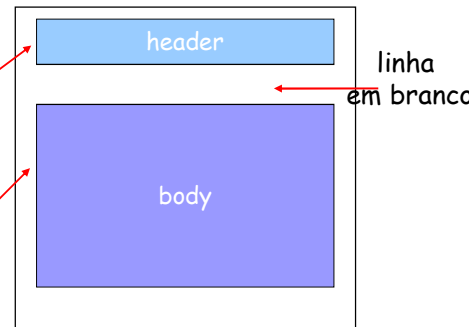
- From:

- Subject:

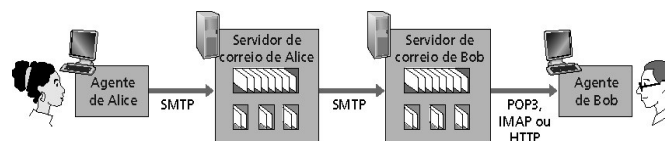
diferente *dos* comandos *HTTP*

- corpo


- a “mensagem”, ASCII somente com caracteres



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, Gmail, etc.



Protocolo POP3

Fase de autorização


- comandos do cliente:
 - user: declara nome do usuário
 - pass: *password*
- respostas do servidor
 - +OK
 - ERR

Fase de transação

- 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 (continuação) 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

- Princípios de aplicações de rede
- Web e HTTP
- Correio eletrônico
 - SMTP, POP3, IMAP
- DNS



DNS: *Dominain Name System*

Pessoas: muitos identificadores:

- RG, nome, passaporte, CPF

Internet hospedeiros, roteadores:

- Endereços IP (32 bits) - usados para endereçar datagramas
- “nome”, ex.: www.dsc.ufcg.edu.br - usados por humanos

P.: Como 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 (tradução nome/endereço)
- Nota: função interna da Internet, implementada como protocolo da camada de aplicação
- Complexidade na “borda” da rede



DNS

DNS services

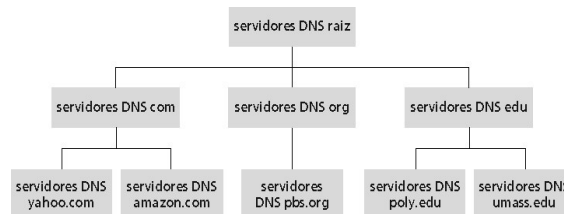
- Nome do hospedeiro para tradução de endereço IP
- Hospedeiro *aliasing* (apelido)
 - Nomes canônicos e *alias*
 - mail server aliasing
 - distribuição de carga
- Servidores Web replicados: estabelece o endereço IP para um nome canônico

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; 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**



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

52

Servidores Top-Level Domain (TLD): responsáveis pelos domínios com, org, net, edu etc e todos os domínios top-level nacionais uk, fr, ca, jp, br

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

Servidores DNS autorizados: servidores DNS de organizações, provêem 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

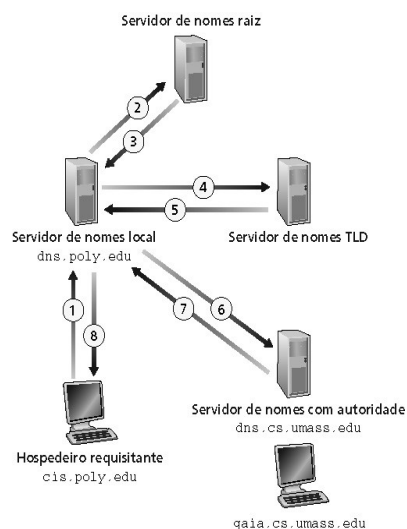
53

- 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 procurador (proxy), encaminhando as perguntas para dentro da hierarquia



Exemplo

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



UFCEG
Universidade Federal
de Campina Grande

Consultas recursivas

55

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”

Servidor de nomes raiz

Servidor de nomes local
dns.poly.edu

Servidor de nomes TLD

Servidor de nomes com autoridade
dns.cs.umass.edu

Hospedeiro requisitante
cis.poly.edu

gaia.cs.umass.edu

UFCEG
Universidade Federal
de Campina Grande

DNS: armazenando e atualizando registros

Uma vez que um servidor de nomes aprende 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**, **tll**)

- Type = A
 - **name** é o nome do computador
 - **value** é o endereço IP
- Type = NS
 - **name** é um domínio (ex.: foo.com)
 - **value** é o endereço IP do servidor de nomes autorizados para este domínio
- Type = CNAME
 - **name** é um “apelido” para algum nome “canônico” (o nome real)
www.ibm.com é realmente www.ibm.com.cs186.net
 - **value** é o nome canônico
- 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

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 'util', que pode ser usada



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

DNS: protocolo e mensagens



Camada de aplicação

60

- 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 de 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