Redes de Computadores

Camada de Aplicação

Camada de Aplicação

Nossos objetivos:

- aspectos conceituais e de implementação de protocolos de aplicação em redes
  - paradigma cliente servidor
  - modelos de serviço
- aprender sobre protocolos através do estudo de protocolos populares no nível da camada de aplicação

Outros objetivos:

- protocolos específicos:
  - http
  - ftp
  - smtp
  - pop
  - dns

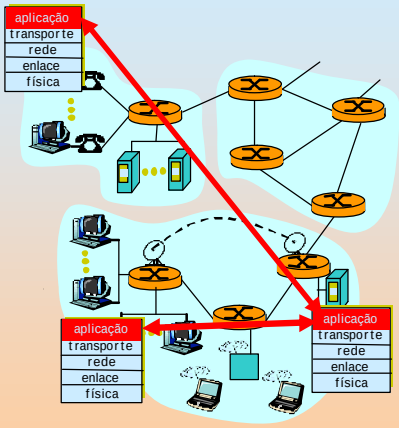
Aplicações e protocolos da camada de aplicação

Aplicação: processos distribuídos em comunicação

- executam em hospedeiros no "espaço de usuário"
- trocam mensagens para a realização da aplicação
- p.ex., correio, transf. de arquivo, WWW

Protocolos da camada de aplicação

- fazem "parte" da aplicação
- definem as mensagens trocadas e as ações tomadas
- usam serviços providos por protocolos de camadas inferiores



Aplicações de rede: jargões

- Processo: é um programa que executa num hospedeiro.
- 2 processos no mesmo hospedeiro se comunicam usando comunicação entre processos (*InterProcess Communication*) definida pelo sistema operacional (SO).
- 2 processos em hospedeiros distintos se comunicam usando um protocolo da camada de aplicação.
- Agente de usuário* (UA) processo que faz a interface entre o usuário ("em cima") e a aplicação de rede ("embaixo").
  - WWW: browser
  - Correio: leitor/compositor de mensagens
  - streaming audio/video: tocador de mídia (*media player*)

* Implementa um protocolo da camada de aplicação

Paradigma Cliente-Servidor (C/S)

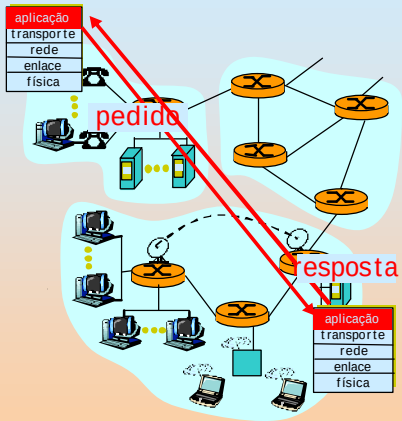
Aplicações de rede típicas têm duas partes: *cliente* e *servidor*

Cliente:

- inicia a comunicação com o servidor ("fala primeiro")
- tipicamente solicita serviços do servidor
  - WWW: cliente implementado no browser;
  - E-mail: no leitor de mensagens

Servidor:

- provê ao cliente o serviço requisitado
  - p.ex., servidor WWW envia página solicitada;
  - servidor de correio envia mensagens



Protocolos da camada de aplicação: Interface de Programação

- API (*Application Programming Interface*): interface de programação de aplicações
- define a interface entre a camada de aplicação e a de transporte
- socket: API da Internet
  - 2 processos se comunicam enviando dados para um socket ou lendo dados de um socket

P: como um processo pode "identificar"o outro processo com o qual quer se comunicar?

- endereço IP do hospedeiro do outro processo
- número de porta - permite que o hospedeiro receptor determine a qual processo local deve ser entregue a mensagem

De que serviço de transporte uma aplicação pode precisar?

Perda de dados

- algumas aplicações (p.ex., áudio) podem tolerar algumas perdas
- outras (p.ex., transf. de arquivos, telnet) requerem transferência de dados 100% confiável

Largura de banda

- algumas aplicações (p.ex., multimídia) requerem quantia de banda mínima para serem "viáveis"
- outras aplicações ("aplicações elásticas") conseguem usar qualquer quantia de banda disponível

Temporização

- algumas aplicações (p.ex., telefonia Internet, jogos interativos) requerem baixo atraso (retardo) para serem "viáveis"

Requisitos do serviço de transporte de aplicações comuns

Aplicação	Perdas	Banda	Sensibilidade temporal
transferência de arqs	sem perdas	elástica	não
correio	sem perdas	elástica	não
documentos WWW	sem perdas	elástica	não
áudio/vídeo de tempo real	tolerante	áudio: 5Kb-1Mb vídeo:10Kb-5Mb	sim, 100's mseg
áudio/vídeo gravado	tolerante	como anterior	sim, alguns segs
jogos interativos	tolerante	> alguns Kbps	sim, 100's mseg
apls financeiras	sem perdas	elástica	sim e não

Serviços providos por protocolos de transporte Internet

Serviço TCP:

- *orientado à conexão:* requerido o estabelecimento de conexão entre cliente e servidor
- *transporte confiável* de dados entre os processos transmissor e receptor
- *controle de fluxo:* remetente não vai "afogar" receptor (compatibilização de velocidades)
- *controle de congestionamento:* protege a rede do excesso de tráfego (estrangular remetente quando a rede carregada
- *não provê:* garantias de temporização ou de banda mínima

Serviço UDP:

- transferência de dados não confiável entre processos remetente e receptor
- não provê: estabelecimento de conexão, confiabilidade, controle de fluxo, controle de congestionamento, garantias temporais ou de banda mínima

P: Qual é o interesse em ter um UDP?

Aplicações Internet: seus protocolos e respectivos protocolos de transporte

Aplicação	Protocolo da camada de apl	Protocolo de transporte usado
correio eletrônico	smtp [RFC 821]	TCP
acesso a terminal remoto	telnet [RFC 854]	TCP
WWW	http [RFC 2068]	TCP
transferência de arquivos	ftp [RFC 959]	TCP
streaming multimídia	RTP ou proprietário (p.ex. RealNetworks)	TCP ou UDP
servidor de arquivo remoto	NFS	TCP ou UDP
telefonia Internet	RTP ou proprietário (p.ex., Vocaltec)	tipicamente UDP

Web (WWW): jargões

- Página WWW:
  - consiste de "objetos"
  - endereçada por uma URL
- Quase todas as páginas WWW consistem de:
  - página base HTML, e
  - vários objetos referenciados.
- URL tem duas partes: nome de hospedeiro e nome de caminho:

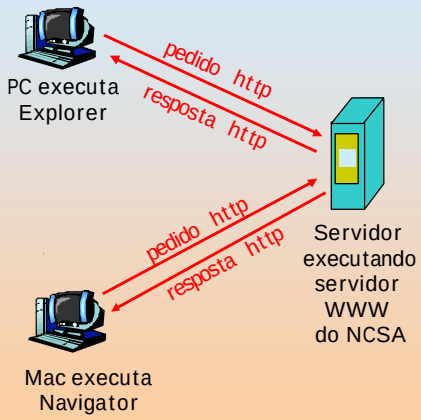
www.univ.br/algum-depto/pic.gif

- Agente de usuário para WWW se chama navegador (*browser*):
  - MS Internet Explorer
  - Netscape Communicator
- Servidor para WWW se chama "servidor WWW":
  - Apache (domínio público)
  - MS Internet Information Server (IIS)

Web: o protocolo HTTP

HTTP: HyperText Transfer Protocol

- protocolo da camada de aplicação para WWW
- modelo cliente/servidor
  - *cliente:* navegador que pede, recebe e apresenta objetos WWW
  - *servidor:* envia objetos em resposta a pedidos
- HTTP/1.0: RFC 1945
- HTTP/1.1: RFC 2068



Protocolo HTTP

http: serviço de transporte TCP

- cliente inicia conexão TCP (cria socket) com servidor na porta 80
- servidor aceita conexão TCP do cliente
- mensagens http (mensagens do protocolo da camada de aplicação) trocadas entre navegador (cliente http) e servidor WWW (servidor http)
- conexão TCP é encerrada

http é "sem estado" (*stateless*)

- servidor não mantém informação sobre pedidos anteriores do cliente

Nota

Protocolos que mantêm "estado" são complexos!

- história passada (estado) tem que ser guardada
- "quedas" do servidor ou cliente podem tornar as visões do "estado" inconsistentes (devem ser reconciliadas)

HTTP: exemplo de operação

Suponha que usuário digite a URL
www.algumaUniv.br/algumDepartmento/inicial.index

(contém texto e referências a 10 imagens jpeg)

1a. cliente http inicia conexão TCP com servidor http (processo) em www.algumaUniv.br. Porta 80 é a padrão (*default*) para servidor http.

1b. servidor http no hospedeiro www.algumaUniv.br, esperando por conexão TCP na porta 80, "aceita" conexão e notifica cliente

2. cliente http envia *mensagem de requisição* http (contendo a URL) através do socket da conexão TCP

3. servidor http recebe mensagem de requisição, formula uma *mensagem de resposta* contendo o objeto solicitado (algumDepartmento/inicial.index) e envia resposta via socket

tempo

HTTP: exemplo de operação (cont.)

4. servidor http encerra conexão TCP (HTTP/1.0).

5. cliente http recebe a mensagem de resposta contendo o arquivo html e apresenta o conteúdo html. Analisando arquivo html, encontra 10 objetos jpeg referenciados

6. Passos 1 a 5 repetidos para cada um dos 10 objetos jpeg

tempo

Conexões persistentes e não-persistentes

Não persistente

- HTTP/1.0
- servidor analisa pedido, responde e encerra conexão TCP
- 2 RTTs (*round trip time*) para obter cada objeto
  - Conexão TCP
  - Pedido/resposta do objeto
- transferência de cada objeto sofre de partida lenta
- a maioria dos navegadores 1.0 abrem várias conexões TCP paralelas

Persistente

- *default* para HTTP/1.1
- na mesma conexão TCP: servidor analisa pedido, responde, analisa novo pedido, ...
- cliente envia pedidos para todos objetos referenciados assim que recebe o HTML base
- poucos RTTs e menos partidas lentas (*slow start*)

Formato de Mensagens HTTP: requisição

- Dois tipos de mensagem http: *pedido*, *resposta*
- mensagem de pedido http:
  - ASCII (formato legível por pessoas)

linha do pedido (comandos GET, POST, HEAD)

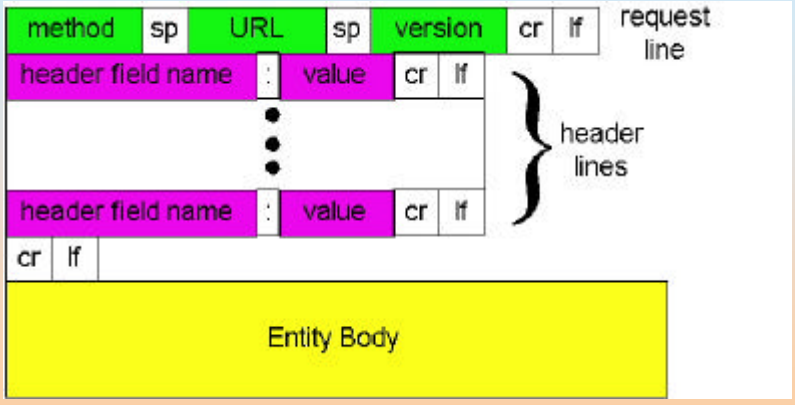
linhas de cabeçalho

```
GET /somedir/page.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif,image/jpeg
Accept-language:fr
```

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

Carriage return, line feed indica fim de mensagem

Requisição HTTP: formato geral



Formato de Mensagens HTTP: resposta

linha de status (protocolo, código de status, frase de status)

linhas de cabeçalho

dados, p.ex., arquivo html solicitado

```
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

dados dados dados dados ...
```

Códigos de status da resposta HTTP

Na primeira linha da mensagem de resposta servidor->cliente.
Alguns códigos típicos:

200 OK
- sucesso, objeto pedido segue mais adiante nesta mensagem

301 Moved Permanently
- objeto pedido mudou de lugar, nova localização especificada mais adiante nesta mensagem (Location:)

400 Bad Request
- mensagem de pedido não entendida pelo servidor

404 Not Found
- documento pedido não se encontra neste servidor

505 HTTP Version Not Supported
- versão de http do pedido não usada por este servidor

Cliente HTTP: faça você mesmo!

1. "Aponte" um cliente telnet para seu servidor WWW favorito:

```
telnet www.univ.br 80
```

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

2. Digite um pedido GET http:

```
GET /~zagari/index.html HTTP/1.0
```

Digitando isto (deve teclar ENTER duas vezes), você está enviando um pedido GET mínimo (porém completo) ao servidor http

3. Examine a mensagem de resposta enviada pelo servidor http !

HTML (HyperText Markup Language)

- HTML: uma linguagem simples para hipertexto
  - começou como versão simples de SGML
  - construção básica: cadeias de texto anotadas
- Construtores de formato operam sobre cadeias
  - .. *bold* (negrito)
  - <H1 ALIGN=CENTER> ..título centrado .. </H1>
  - <BODY bgcolor=white text=black link=red ..> .. </BODY>
- vários formatos
  - listas de *bullets*, listas ordenadas, listas de definição
  - tabelas
  - *frames*

Encadeamento de referências

- Referências <A HREF=LinkRef> ... </A>
  - a componentes do documento local
    <A HREF="importante"> clique para uma dica </A>
  - a documentos no servidor local
    <A HREF="../index.htm"> voltar ao sumário </A>
  - a documentos em outros servidores
    <A HREF="http://www.univ.br"> saiba sobre a Univ </A>
- Multimídia
  - imagem embutida: <IMG SRC="eclipse">
  - imagem externa: <A HREF="eclipse.gif"> imagem maior </A>
  - vídeo Mpeg <A HREF="ByeByeBrasil.mpg"> um bom filme </A>
  - som <A HREF="http://www.sons.br/aniv.au"> feliz niver </A>

Formulários e interação bidirecional

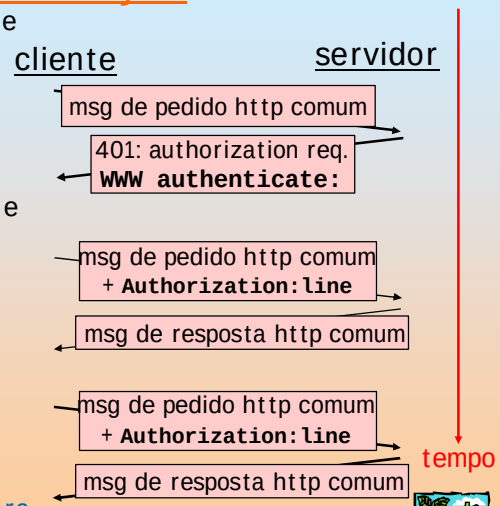
- Formulários transmitem informação do cliente ao servidor
- HTTP permite enviar formulários ao servidor
- Resposta enviada como página HTML ***dinâmica***

- Formulários processados usando *scripts* CGI (programas que executam no servidor WWW)
  - CGI - *Common Gateway Interface*
  - *scripts* CGI escondem acesso a diferentes serviços
  - servidor WWW atua como *gateway* universal

Interação usuário-servidor: autenticação

Meta da autenticação: controle de acesso aos documentos (conteúdo) do servidor

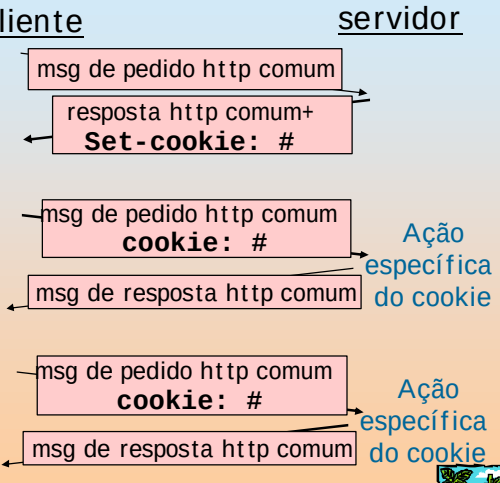
- sem estado: cliente deve apresentar autorização com cada pedido
- autorização: tipicamente nome e senha
  - **authorization:** linha de cabeçalho no pedido
  - se não for apresentada autorização, servidor nega accesso, e coloca no cabeçalho da resposta
    WWW authenticate:



Browser guarda nome e senha para evitar que sejam pedidos ao usuário a cada acesso.

Interação usuário-servidor: cookies "mantendo estado"

- servidor envia "cookie" ao cliente na msg de resposta
  Set-cookie: 1678453
- cliente apresenta cookie nos pedidos posteriores
  cookie: 1678453
- servidor casa **cookie** apresentado com a info guardada no servidor
  - autenticação
  - lembrando preferências do usuário, opções anteriores

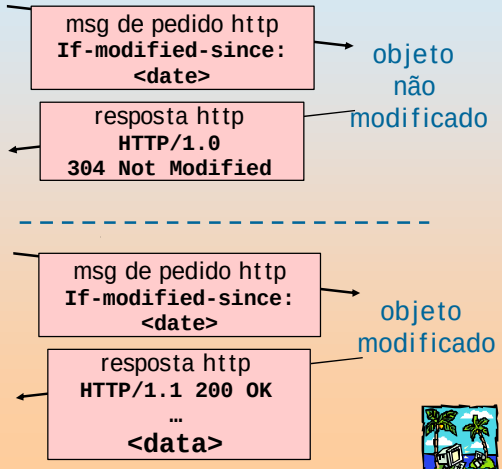


Interação usuário-servidor: GET condicional

- Objetivo: não enviar objeto se cliente já tem (na cache) versão atual
- cliente: especifica data da cópia armazenada na cache no pedido HTTP

  If-modified-since: <date>
- servidor: resposta não contém objeto se cópia na cache é atual:

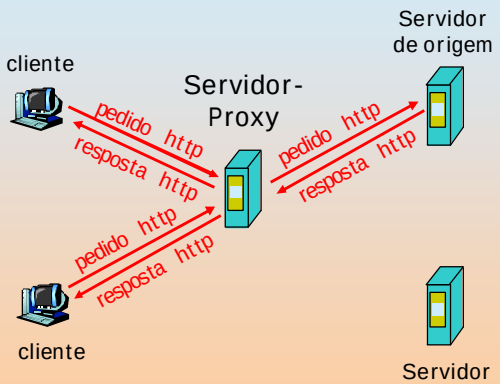
  HTTP/1.0 304 Not Modified



Web Caches (servidores proxy)

Meta: atender pedido do cliente sem envolver servidor WWW fonte da informação

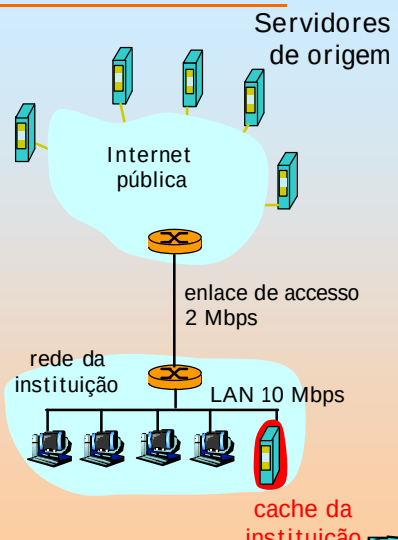
- usuário configura navegador: acesso à Web via *proxy* (procurador)
- cliente envia todos pedidos http ao *web cache*
  - se objeto existe no *web cache*, este o devolve imediatamente na resposta http
  - senão, solicita objeto do servidor de origem, depois envia objeto ao cliente na resposta http



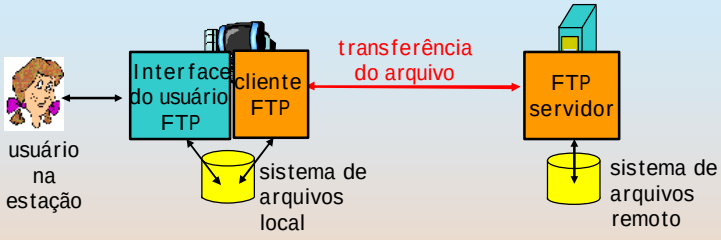
Por que usar *Web cache* ?

Suposição: cache "perto" do cliente (p.ex., na mesma rede)

- tempo de resposta menor
- reduz tráfego para servidores distantes
  - enlaces que ligam a rede da instituição ou do provedor à Internet frequentemente são gargalos



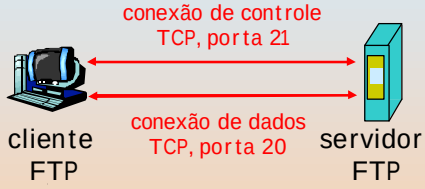
FTP: o protocolo de transferência de arquivos



- transferir arquivo de/para hospedeiro remoto
- modelo cliente/servidor
  - *cliente:* lado que inicia transferência (pode ser de ou para o sistema remoto)
  - *servidor:* hospedeiro remoto
- FTP: RFC 959
- servidor ftp: porta 21

FTP: conexões separadas para controle e dados

- cliente ftp contacta servidor ftp na porta 21, especificando TCP como protocolo de transporte
- são abertas duas conexões TCP paralelas:
  - controle: troca de comandos e respostas entre cliente e servidor.
    "controle fora da banda"
  - dados: dados do arquivo de/para servidor
- servidor ftp mantém "estado": diretório atual, autenticação realizada



FTP: comandos e respostas

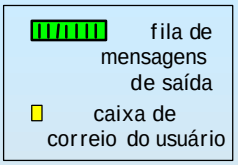
Comandos típicos:

- enviados em texto ASCII pelo canal de controle
- **USER** ***nome***
- **PASS** ***senha***
- **LIST** - devolve lista de arquivos no diretório atual
- **RETR arquivo -** recupera (obtém) arquivo remoto
- **STOR arquivo -** armazena (escreve) arquivo no hospedeiro remoto

Códigos de retorno típicos

- código e frase de status (como para http)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

Correio Eletrônico

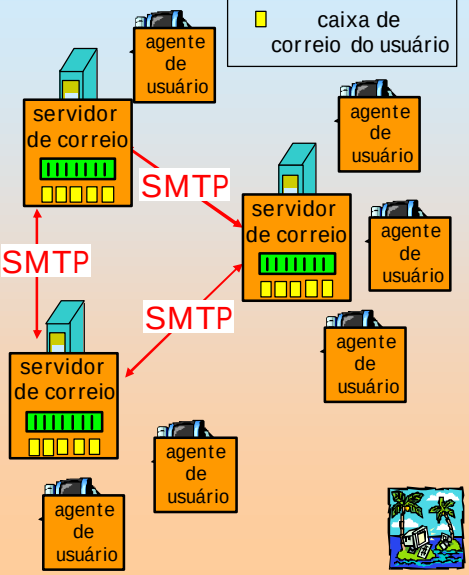


Três componentes principais:

- agentes de usuário (UA)
- servidores de correio
- simple mail transfer protocol: smtp

Agente de Usuário

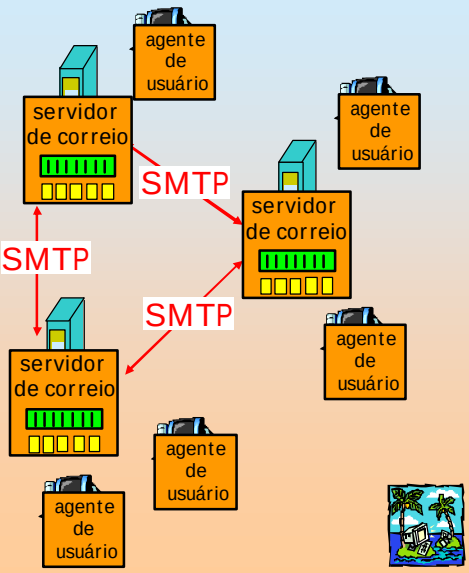
- "leitor de correio"
- compor, editar, ler mensagens de correio
- p.ex., pine, Eudora, Outlook, elm, Netscape Messenger
- mensagens de entrada e de saída são armazenadas no servidor



Correio Eletrônico: servidores de correio

Servidores de correio

- caixa de correio: contém mensagens que chegaram (ainda não lidas) p/ usuário
- fila de mensagens: contém as mensagens de saída (a serem enviadas)
- protocolo smtp: entre servidores de correio para transferir mensagens entre si
  - "cliente": servidor de correio que envia
  - "servidor": servidor de correio que recebe



Correio Eletrônico: smtp [RFC 821]

- usa TCP para a transferência confiável de mensagens de correio do cliente ao servidor - Porta 25
- transferência direta: so servidor remetente ao servidor receptor
- três fases de transferência
  - handshaking (cumprimento/apresentação)
  - transferência das mensagens
  - encerramento
- interação comando/resposta
  - comandos: texto ASCII
  - resposta: código e frase de status
- mensagens devem ser formatadas em código ASCII de 7-bits

Exemplo de Interação SMTP

```
S: 220 doces.br
C: HELO consumidor.br
S: 250  Hello consumidor.br, pleased to meet you
C: MAIL FROM: <ana@consumidor.br>
S: 250 ana@consumidor.br... Sender ok
C: RCPT TO: <bernardo@doces.br>
S: 250 bernardo@doces.br ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C:
C: Voce gosta de chocolate?
C:   Que tal sorvete?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 doces.br closing connection
```

Cliente SMTP: faça você mesmo!

- **`telnet nomedoservidor 25`**
- veja resposta 220 do servidor
- entre com os comandos HELO, MAIL FROM, RCPT TO, DATA, QUIT

- estes comandos permitem que você envie correio sem usar o agente de usuário do remetente (leitor de correio)

SMTP: últimas palavras

- SMTP usa conexões persistentes
- SMTP requer que a mensagem (cabeçalho e corpo) esteja em ASCII de 7-bits
- algumas cadeias de caracteres não são permitidas nas mensagens (p.ex., CRLF.CRLF). Logo, a mensagem pode ter que ser codificada (normalmente em base-64 ou "quoted printable")
- servidor SMTP usa CRLF.CRLF para reconhecer o final da mensagem

Comparação com HTTP

- HTTP: pull (puxar)
- email: push (empurrar)
- ambos tem interação comando/resposta e códigos de status em ASCII
- HTTP: cada objeto é encapsulado em sua própria mensagem de resposta
- SMTP: múltiplos objetos são enviados numa mensagem de múltiplas partes

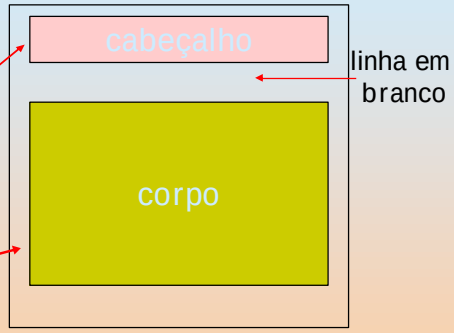
Formato de mensagens SMTP

SMTP: protocolo para trocar mensagens de correio

RFC 822: padrão para formato de mensagem de texto:

- linhas de cabeçalho, p.ex.,
  - To:
  - From:
  - Subject:

  diferentes dos comandos SMTP!
- corpo
  - a "mensagem", somente de caracteres ASCII



Formato de uma mensagem: extensões para multimídia

- MIME: multimedia mail extension, RFC 2045, 2056
- linhas adicionais no cabeçalho da msg declaram o tipo do conteúdo MIME

versão MIME

método usado para codificar dados

tipo, subtipo de dados multimídia, declaração parâmetros

Dados codificados

```
From: ana@consumidor.br
To: bernardo@doces.br
Subject: Imagem de uma bela torta
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
```

Tipos MIME

Content-Type: tipo/subtipo; parâmetros

Text

- Exemplos de subtipos: **plain, html**

Image

- Exemplos de subtipos: **jpeg, gif**

Video

- Exemplos de subtipos: **mpeg, quicktime**

Audio

- Exemplos de subtipos: **basic** (codificado 8-bit µ-law), **32kadpcm** (codificação 32 kbps)

Aplicação

- outros dados que devem ser processados pelo leitor antes de serem apresentados para "visualização"
- Exemplos de subtipos: **msword, octet-stream**

Tipo Multipart

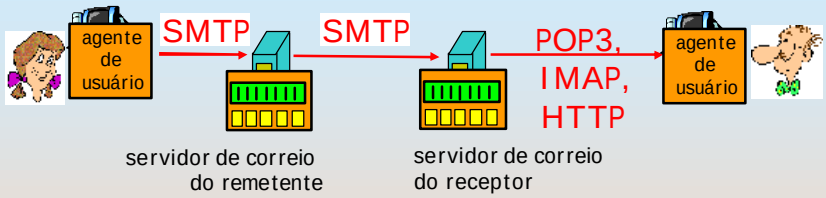
```
From: ana@consumidor.br
To: bernardo@doces.br
Subject: Imagem de uma bela torta
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

caro Bernardo,
Anexa a imagem de uma torta deliciosa.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
--98766789--
```

Protocolos de acesso ao correio



- SMTP: entrega/armazenamento no servidor do receptor (destino)
- Protocolo de acesso ao correio: recuperação de mensagens no servidor
  - POP: Post Office Protocol [RFC 1939]
    - autorização (agente <--> servidor) e transferência
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - mais recursos (mais complexo)
    - manuseio de mensagens armazenadas no servidor
  - HTTP: Hotmail , Yahoo! Mail, Webmail, etc.

Protocolo POP3

fase de autorização

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

fase de transação (cliente):

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

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

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

DNS: Domain Name System

Pessoas: muitos identificadores:
- CPF, nome, no. da Identidade

hospedeiros, roteadores Internet :
- endereço IP (32 bit) - usado p/ endereçar datagramas
- "nome" - p.ex., maquina.univ.br - usado por gente

P: como mapear entre nome e endereço IP?

Domain Name System:
- *base de dados distribuída* implementada numa hierarquia de muitos *servidores de nomes*
- *protocolo de camada de aplicação* permite que hospedeiros e roteadoresse comuniquem com servidores de nomes para *resolver* nomes (tradução nome/endereço)
  - nota: função imprescindível da Internet, implementada como protocolo de camada de aplicação
  - complexidade na "borda" da rede

DNS

- Roda sobre UDP e usa a porta 53
- Especificado nas RFCs 1034 e 1035 e atualizado em outras RFCs.
- Outros serviços:
  - apelidos para hospedeiros (aliasing)
  - apelido para o servidor de mails
  - distribuição da carga

Servidores de nomes DNS

Por que não centralizar o DNS?

- ponto único de falha
- volume de tráfego
- base de dados distante
- manutenção (da BD)

Não é escalável!

- Nenhum servidor mantém todos os mapeamentos de nomes para endereços IP

servidor de nomes local:

- ✓ cada provedor ou empresa tem seu *servidor de nomes local (default)*
- ✓ consultas DNS de hospedeiros vão primeiro ao servidor de nomes local

servidor de nomes "authoritative":

- ✓ para hospedeiro: armazena nome e endereço IP dele
- ✓ pode realizar tradução nome/endereço para aquele nome de hospedeiro

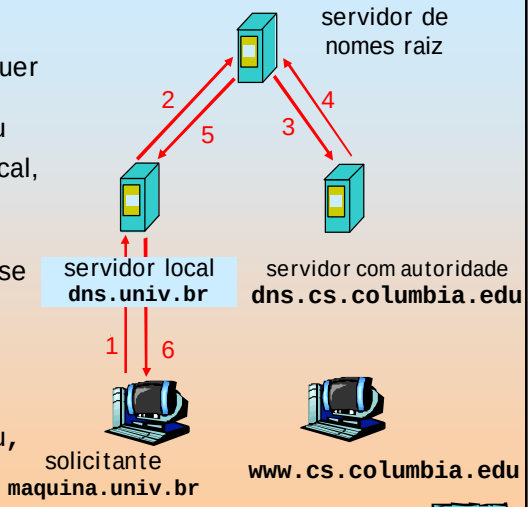
DNS: Servidores de Nome Raiz

- procurados por servidores de nomes locais que não conseguem resolver um nome
- servidor de nome raiz:
  - ✓ procuram servidores de nome com autoridade se o mapeamento do nome for desconhecido
  - ✓ obtém tradução
  - ✓ retorna mapeamento para o servidor de nomes local
- ~ uma dúzia de servidores raiz no mundo

DNS: exemplo simples

Hospedeiro `maquina.univ.br` requer endereço IP de `www.cs.columbia.edu`

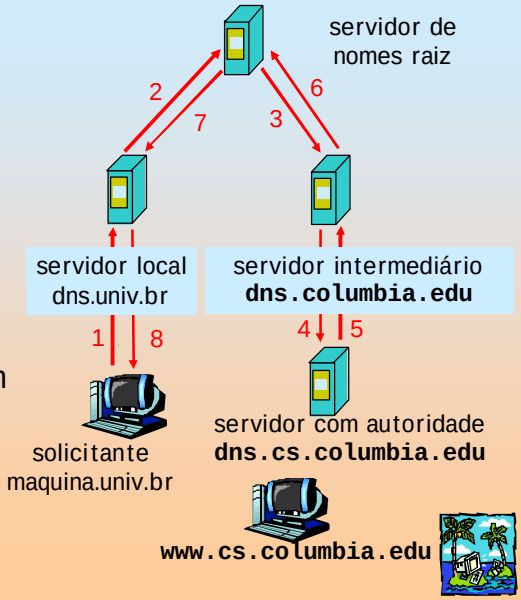
1. Contata servidor DNS local, `dns.univ.br`
2. `dns.univ.br` contata servidor de nomes raiz, se necessário
3. Servidor de nomes raiz contata servidor com autoridade `dns.cs.columbia.edu`, se necessário



DNS: exemplo

Servidor de nomes raiz:

- pode não conhecer o servidor de nomes com autoridade para um certo domínio
- pode conhecer um *servidor de nomes intermediário:* a quem contacta para descobrir o servidor de nomes com autoridade



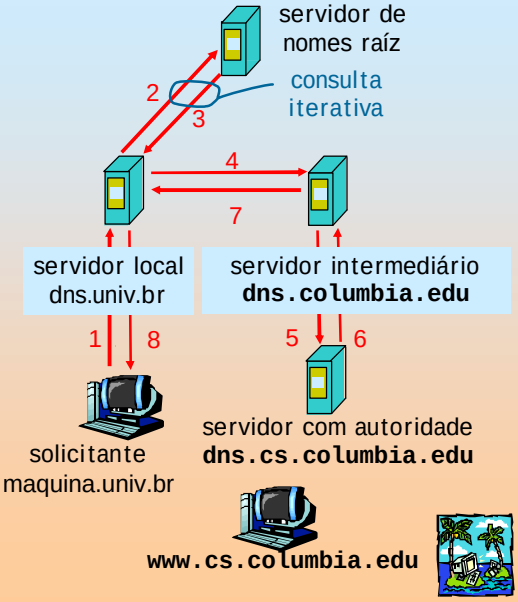
DNS: consultas iterativas

consulta recursiva:

- transfere a responsabilidade de resolução do nome para o servidor de nomes consultado
- carga pesada?

consulta iterativa:

- servidor consultado responde com o nome de outro servidor de nomes para contato
- "Não conheço este nome, mas pergunte a este servidor"



DNS: uso de cache e atualização de registros

- uma vez que um servidor qualquer aprende um mapeamento, ele o coloca numa *cache* local
  - futuras consultas são resolvidas usando dados da cache
  - entradas na cache são sujeitas a temporização (desaparecem depois de um certo tempo) ttl = time to live (sobrevida)
- mecanismos de atualização/notificação dos dados estão sendo projetados pela IETF
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

Registros DNS

DNS: BD distribuída contendo *registros de recursos* (RR)

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

- Tipo=A
  - **name** é o nome de hospedeiro
  - **value** é o seu endereço IP
- Tipo=NS
  - **name** é domínio (p.ex. xxx.com.br)
  - **value** é o endereço IP do servidor de nomes com autoridade para este domínio
- Tipo=CNAME
  - **name** é um "apelido" (alias) para algum nome "canônico" (verdadeiro), p.ex.,

  www.ibm.com é realmente

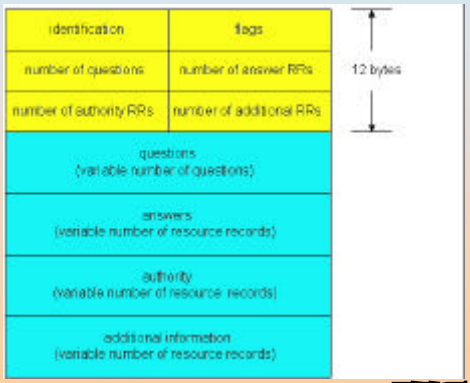
  servereast.backup2.ibm.com
  - **value** é o nome canônico
- Tipo=MX
  - **value** é o nome do servidor de correio associado com **name**

DNS: protocolo e mensagens

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

cabeçalho da msg

- identificação: ID de 16 bits para pedido. Resposta ao pedido usa mesmo ID
- flags:
  - pedido ou resposta
  - recursão desejada
  - recursão disponível
  - resposta é autoritativa



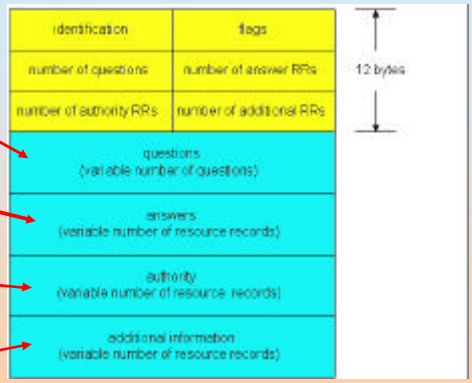
DNS: protocolo e mensagens

Campos de nome e de tipo para um pedido

RRs de resposta ao pedido

Registros para outros servidores com autoridade

informação adicional "relevante" que pode ser usada



Resumo

Terminamos nosso estudo de aplicações de rede!

- Requisitos do serviço de aplicação:
  - confiabilidade, banda, retardo
- paradigma cliente-servidor
- modelo de serviço do transporte orientado a conexão, confiável da Internet: TCP
  - não confiável, datagramas: UDP
- Protocolos específicos:
  - http
  - ftp
  - smtp, pop3
  - dns

Camada de Aplicação: Sumário

Mais importante: aprendemos sobre *protocolos*

- troca típica de mensagens pedido/resposta:
  - ✓ cliente solicita info ou serviço
  - ✓ servidor responde com dados, código de *status*
- formatos de mensagens:
  - ✓ cabeçalhos: campos com info sobre dados (metadados)
  - ✓ dados: info sendo comunicada
- msgs de controle X dados
  - ✓ na banda, fora da banda
- centralizado X descentralizado
- s/ estado X c/ estado
- transferência de msgs confiável X não confiável
- "complexidade na borda da rede"
- segurança: autenticação