Camada de Aplicação

Mário Meireles Teixeira
UFMA – DEINF
2016

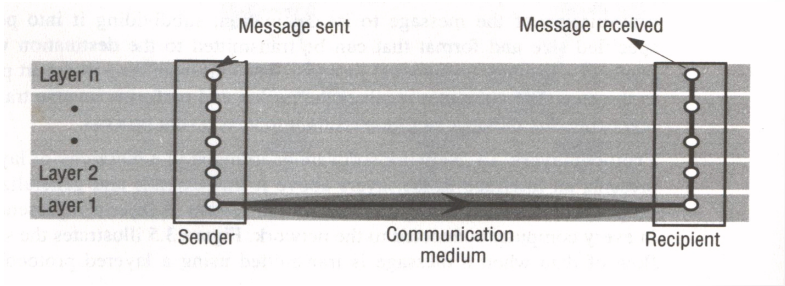
"Redes de Computadores e a Internet", 6a ed, Kurose & Ross

Protocolos

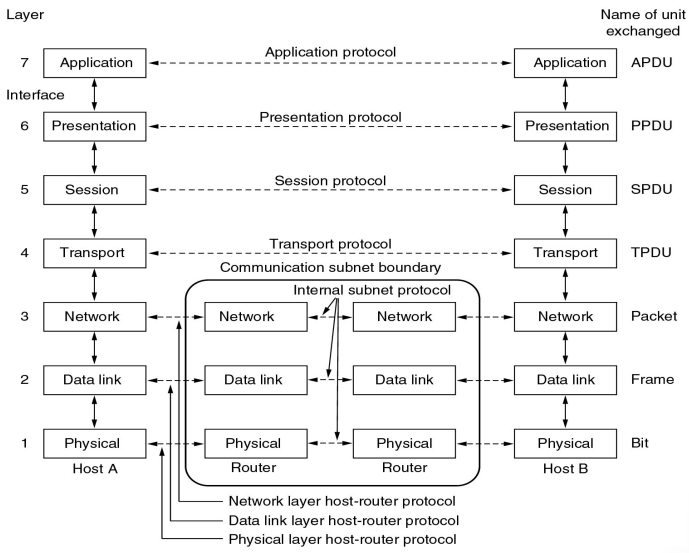
- **Protocolo:**
  - Conjunto de regras e formatos usados para comunicação entre entidades, a fim de permitir a realização de uma tarefa
- Um protocolo define:
  - Os tipos de mensagens trocadas (p.ex., de requisição e resposta)
  - A sequência das mensagens trocadas (regras que definem quando e como um processo envia/responde a mensagens)
  - A sintaxe e semântica do conteúdo das mensagens
- A existência de protocolos permite que as aplicações de rede sejam desenvolvidas de modo independente

Camadas de Protocolos

- O software de rede geralmente é organizado em um conjunto de camadas
- Cada camada é responsável por certos serviços, oferecendo à camada superior uma interface bem definida de acesso a suas operações



Modelo de Referência OSI



Modelo OSI

- **Camada Física**
  - Definição das características físicas do meio de transmissão: sinalização, cabeamento, etc.
  - Ex. de protocolos: X.21, Ethernet banda básica
- **Camada de Enlace**
  - Transmissão de *frames* de dados livres de erros entre computadores diretamente conectados
  - Ex: CSMA/CD (Ethernet), PPP, Wi-Fi, HDLC
- **Camada de Rede**
  - Responsável pelo roteamento dos *pacotes* entre os nós da rede
  - Ex: IP, X.25, IPX

Modelo OSI

- **Camada de Transporte**
  - Estabelece um enlace entre duas máquinas para transmissão de *mensagens*, não necessariamente na mesma rede
  - Serviço orientado ou não a conexão
  - Ex: TCP, UDP, SPX
- **Camada de Sessão**
  - Estabelece a comunicação entre processos em máquinas diferentes
  - Interações cliente-servidor: RPC, RMI

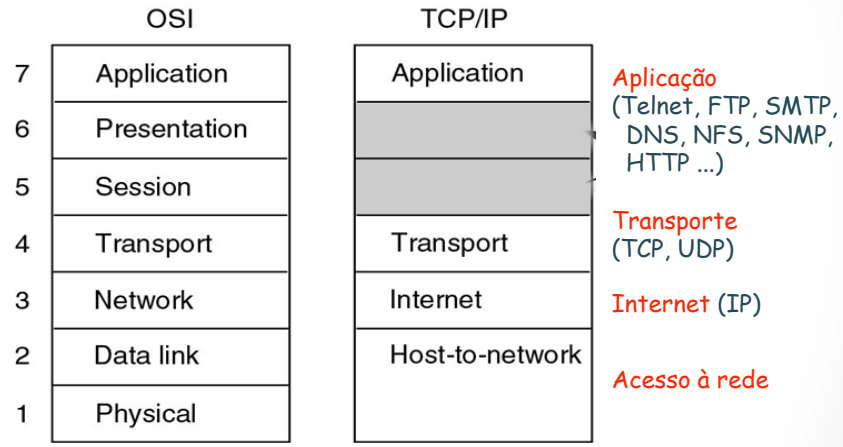
Modelo OSI

- **Camada de Apresentação**
  - Permite apresentar os dados em um formato independente dos utilizados por cada sistema individualmente
  - Criptografia
  - Ex: XDR, ASN.1, NCP
- **Camada de Aplicação**
  - Nesta camada são executadas as aplicações dos usuários e alguns protocolos utilitários de uso geral
  - Ex: ftp, telnet, ICQ, SMTP, X.400, X.500

Protocolos TCP/IP

- Surgiram com a Internet, para a interligação dos seus computadores
- Padrão de fato, em LANs e WANs
- Funcionam sobre diversas tecnologias de rede (cabos coaxiais, fibras óticas, linhas telefônicas, redes sem fio...)
- Características especiais:
  - Uso de padrões abertos
  - Independência de tecnologia de rede
  - Protocolos padronizados – RFCs (rede, transporte e aplicação)
  - Interconexão total – identificação única por dispositivo (endereço IP)
  - Confirmações (ACK) fim-a-fim

Arquitetura TCP/IP



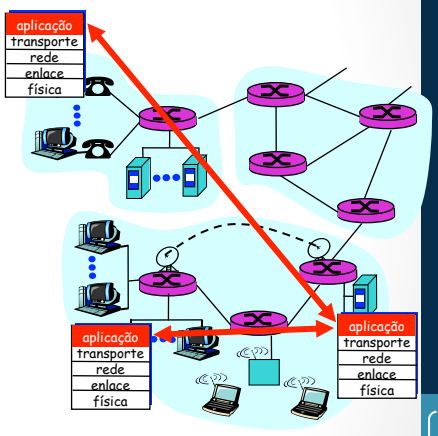
Aplicações e Protocolos de Aplicação

Aplicações: processos distribuídos comunicando-se

- executam nos computadores da rede (hosts) como programas de usuário
- trocam mensagens para realização da aplicação
- vários componentes relacionados
- ex: email, ftp, Web

Protocolos de aplicação:

- definem os tipos e sintaxe das mensagens trocadas
- definem a semântica das mensagens
- definem as ações tomadas
- usam serviços de comunicação das camadas inferiores



Aplicações de Rede

Processo: programa executando num host

- dentro do mesmo host: interprocess communication (definido pelo SO)
- processos executando em diferentes hosts se comunicam através de passagem de mensagens, obedecendo a um protocolo da camada de aplicação

Agente usuário: software que interage com o usuário, de um lado e com a rede, de outro

- implementa um protocolo da camada de aplicação
- Web: browser
- E-mail: leitor de correio
- streaming audio/video: media player

- Aplicação vs. Protocolo

Arquitetura Cliente-Servidor

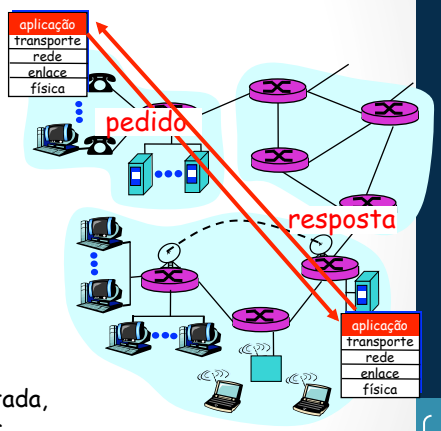
Aplicações de rede típicas têm duas partes: *cliente* e *servidor*

Cliente:

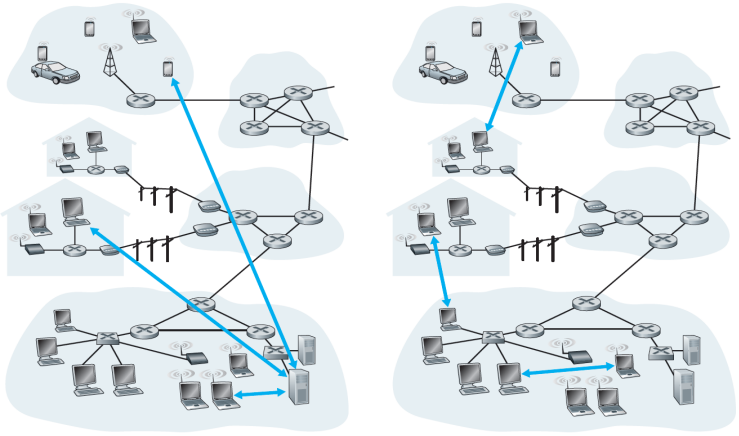
- inicia comunicação com o servidor
- tipicamente solicita serviços do servidor,
- Web: cliente implementado no browser; e-mail: leitor de correio

Servidor:

- fornece os serviços solicitados ao cliente
- ex: servidor web envia a página web solicitada, servidor de e-mail envia as mensagens, etc.
- um host pode atuar como cliente ou servidor



Arquitetura Peer-to-Peer (P2P)



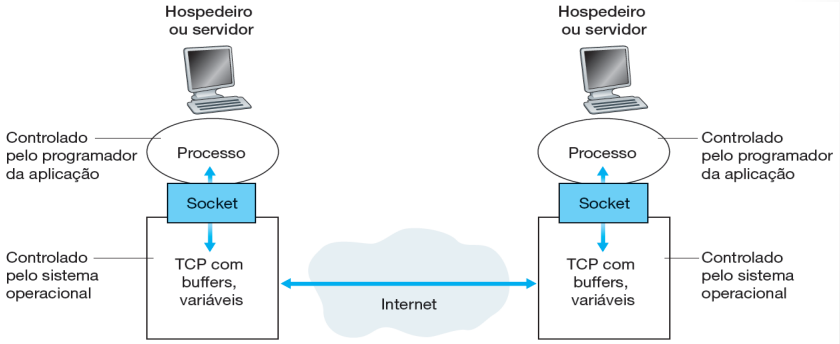
a. Arquitetura cliente-servidor

b. Arquitetura P2P

- A **arquitetura P2P** utiliza a comunicação direta entre duplas de hospedeiros conectados alternadamente, denominados *pares*.

Comunicação entre processos

- Processos de aplicação, *sockets* e protocolo de transporte subjacente.



Comunicação entre processos

- Uma aplicação de rede consiste em pares de processos que enviam mensagens uns para os outros por meio de uma rede.
- Um processo envia mensagens para a rede e recebe mensagens dela através de uma interface de software denominada ***socket***.
- Para identificar o processo receptor, duas informações devem ser especificadas:

1. o endereço do hospedeiro (end. IP)
2. um identificador que especifica o processo receptor no hospedeiro de destino (porta)

Serviços de transporte disponíveis para aplicações

- Transferência confiável de dados
- Vazão
- Temporização
- Segurança

Requisitos das Aplicações

Confiabilidade
- algumas aplicações (aúdio e vídeo) podem tolerar alguma perda de dados
- outras aplicações (transferência de arquivos, telnet, web) exigem transferência de dados 100% confiável

Temporização
- algumas aplicações (telefonia na Internet, jogos interati-vos) exigem baixos atraso e jitter para operar

Largura de Banda
- algumas aplicações (multimídia) impõem um limiar inferior de banda para funcionar (aplicações inelásticas)
- outras aplicações (aplicações elásticas: ftp, correio, web) melhoram quando a banda disponível aumenta, mas podem operar com um valor muito baixo

Requisições de Aplicações comuns da Internet

- Requisitos de aplicações de rede selecionadas:

Aplicação	Perda de dados	Vazão	Sensibilidade ao tempo
Transferência / download de arquivo	Sem perda	Elástica	Não
E-mail	Sem perda	Elástica	Não
Documentos Web	Sem perda	Elástica (alguns kbits/s)	Não
Telefonia via Internet/ videoconferência	Tolerante à perda	Áudio: alguns kbits/s – 1Mbit/s Vídeo: 10 kbits/s – 5 Mbits/s	Sim: décimos de segundo
Áudio/vídeo armazenado	Tolerante à perda	Igual acima	Sim: alguns segundos
Jogos interativos	Tolerante à perda	Poucos kbits/s – 10 kbits/s	Sim: décimos de segundo
Mensagem instantânea	Sem perda	Elástico	Sim e não

Aplicações e seus Protocolos de Transporte

Aplicação	Protocolo de camada de aplicação	Protocolo de transporte subjacente
Correio eletrônico	SMTP [RFC 5321]	TCP
Acesso a terminal remoto	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Transferência de arquivos	FTP [RFC 959]	TCP
Multimídia em fluxo contínuo	HTTP (por exemplo, YouTube)	TCP
Telefonia por Internet	SIP [RFC 3261], RTP [RFC 3550] ou proprietária (por exemplo, Skype)	UDP ou TCP

Protocolo HTTP

WORLD WIDE WEB

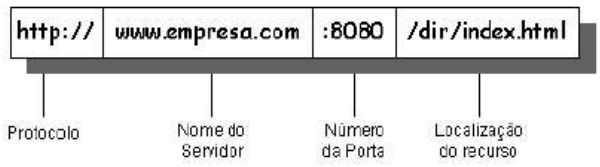
World Wide Web

- Permite o acesso a documentos interligados, espalhados pela Internet
- Tornou-se tão popular que se confunde com a própria Internet
- *1945 - Vannevar Bush:* conceito de hipertexto
- *1965 - Ted Nelson:* cunhou o termo
- *1989 - Tim Berners-Lee:* no CERN, criou a Web e o protocolo http
- *1994 – Marc Andreesen:* desenvolveu o Mosaic; links para diferentes mídias

- Em 1995, a Web tornou-se responsável pela maior parte do tráfego na Internet, porém foi ultrapassada pelas redes P2P em 2004
- Sistema hipermídia em escala global
- Sistema de nomenclatura: URLs
- Interações entre os componentes: paradigma C/S
- A Web funciona sobre dois padrões principais:
  - Linguagem HTML
  - Protocolo HTTP

Sistema de Nomenclatura – URLs

- URLs permitem que os usuários acessem páginas web e outros serviços como FTP, telnet e notícias, a partir do próprio navegador:
  - http - Hipertexto (HTML)
  - ftp - Transferência de arquivos
  - file - Acesso a arquivos locais
  - news - Grupos de notícias e artigos
  - gopher - recuperar informações pelo gopher
  - mailto - Enviar e-mail
  - telnet - Login remoto

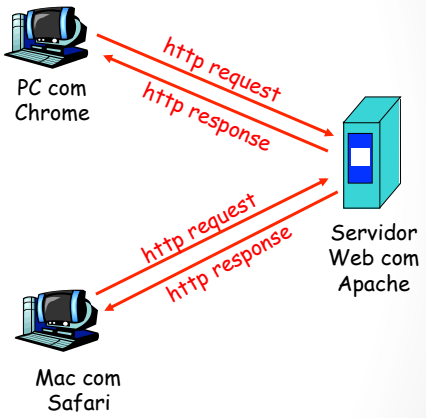


Protocolo HTTP

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 2616



A Web e o HTTP

- Uma página Web é constituída de objetos
- Um objeto é apenas um arquivo que se pode acessar com um único URL
- A maioria das páginas Web é constituída de um arquivo-base HTML e diversos objetos referenciados
- O HTTP usa o TCP como seu protocolo de transporte

Protocolo HTTP

http: protocolo de aplicação sobre TCP

- cliente inicia conexão TCP (cria socket) com o servidor na porta 80
- servidor aceita uma conexão TCP do cliente
- mensagens http são trocadas entre o browser (cliente http) e o servidor web (servidor http)
- A conexão TCP é fechada

http é um protocolo "sem estado":

- o servidor não mantém informações sobre os pedidos dos clientes

Protocolos que mantêm informações de estado são complexos:

- necessidade de organizar informações passadas
- se ocorrer uma falha, as informações podem ser perdidas ou gerar inconsistências entre o cliente e o servidor
- baixa escalabilidade

Exemplo de Operação HTTP (1)

Usuário digita a URL: www.deinf.ufma.br/prof/index.html

(referencia 10 imagens)

1a. cliente http inicia conexão TCP com o servidor http (processo) em www.deinf.ufma.br, pela porta 80 (default)

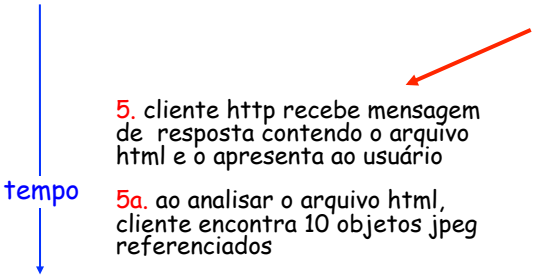
1b. servidor http no host www.deinf.ufma.br, aguardando pela conexão TCP na porta 80, aceita a conexão, notificando o cliente

2. cliente http client envia *http request*, contendo a URL, ao servidor web

3. servidor http recebe mensagem de pedido, recupera o objeto e envia uma *http response*, contendo o objeto solicitado, ao cliente

tempo

Exemplo (2)



4. servidor http fecha conexão TCP (http 1.0)

5. cliente http recebe mensagem de resposta contendo o arquivo html e o apresenta ao usuário

5a. ao analisar o arquivo html, cliente encontra 10 objetos jpeg referenciados

6. cliente repete Passos 1-5 para cada um dos 10 objetos jpeg

Conexões persistentes e não-persistentes

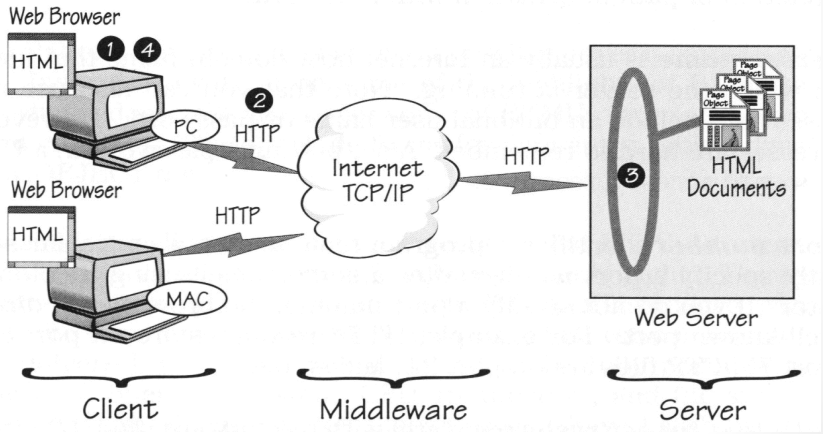
Não-persistente

- http/1.0: cada objeto deve ser enviado por uma conexão TCP distinta
- 2 RTTs para obter um objeto:
  - estabelecimento de conexão TCP
  - solicitação e transferência do objeto
- cada transferência sofre ainda por causa do mecanismo de slow-start do TCP

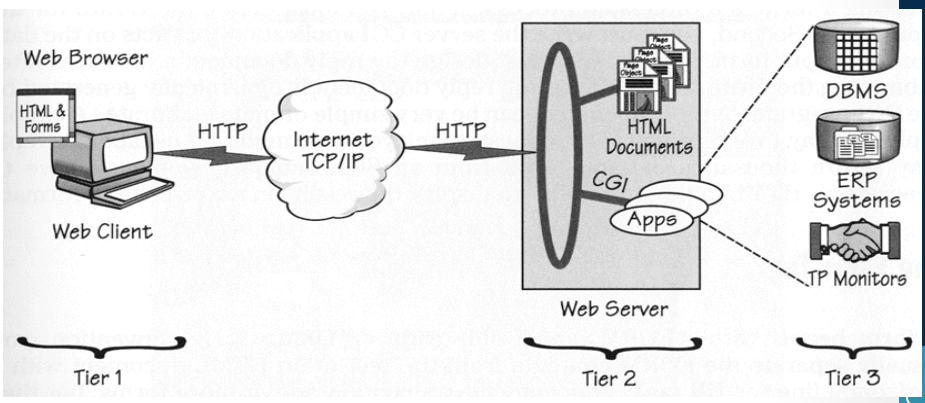
Persistente

- modo default para http/1.1
- na mesma conexão TCP, são recuperados vários objetos
- o cliente solicita todos os objetos referenciados, tão logo ele receba a página HTML básica (*pipelining*)
- poucos RTTs, menos slow start

Cliente-servidor na Web
(duas camadas)



Cliente-servidor na Web
(três camadas)



Mensagens HTTP: requisição

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

linha de pedido (comandos GET, POST, …)

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

linhas de cabeçalho

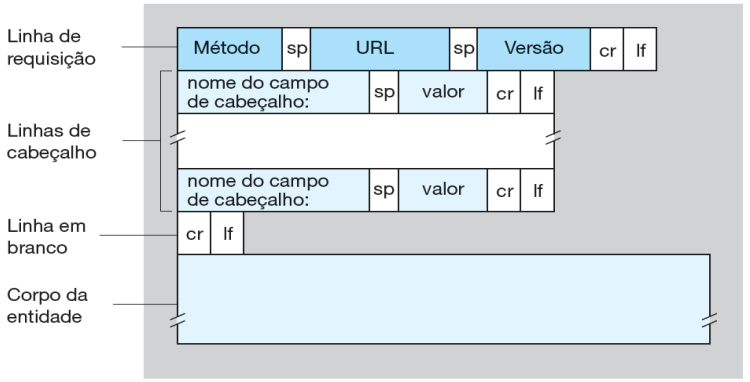
indica fim da mensagem

(linha em branco)

```
Corpo do objeto
```


Formato da mensagem HTTP

- Formato geral de uma mensagem de requisição HTTP



Mensagens HTTP: resposta

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

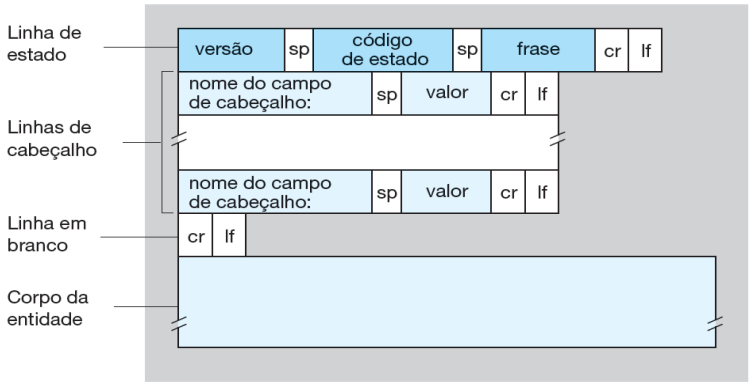
linhas de cabeçalho

dados, p.ex., arquivo html

```
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

dados dados dados dados ...
```


Formato da mensagem HTTP

- Formato geral de uma mensagem de resposta HTTP



Métodos HTTP

- GET – Solicita o objeto identificado pela URL
- HEAD – Obtém informações sobre o objeto sem que o mesmo seja retornado ao cliente (depuração)
- POST – Envia informações adicionais ao servidor web (p.ex., dados de formulários)
- OPTIONS – Obtém opções de comunicação disponíveis ou os requisitos associados ao objeto solicitado
- PUT – Cria ou modifica um objeto no servidor web
- DELETE – Remove um objeto do servidor web
- TRACE – Envia mensagem de teste (*loopback*) ao servidor
- CONNECT – Reservado para comunicação com servidores *proxy*

Códigos de Status

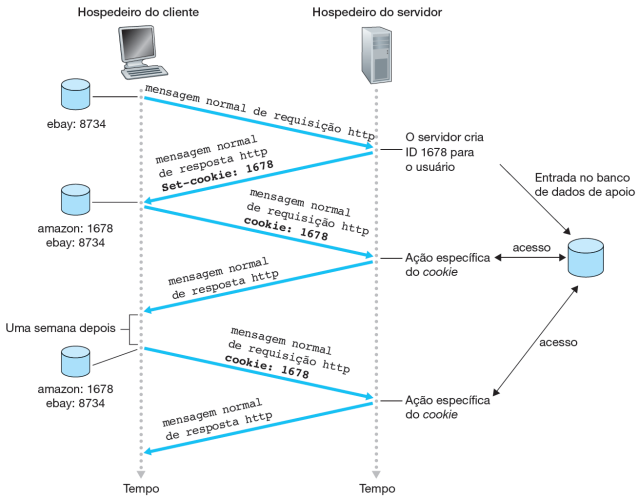
- 1xx – Informational
- 2xx – Success
  - 200 OK
- 3xx – Redirection
  - 301 Moved Permanently
  - 304 Not Modified
  - 307 Temporary Redirect
- 4xx – Client Error
  - 400 Bad Request
  - 401 Unauthorized
  - 404 Not Found
- 5xx – Server Error
  - 503 Service Unavailable
  - 505 HTTP Version Not Supported

Cookies

- Gerados e lembrados pelo servidor (RFC 6265), usados mais tarde para:
  - Autenticação de usuários
  - Monitoração de suas preferências e/ou escolhas prévias

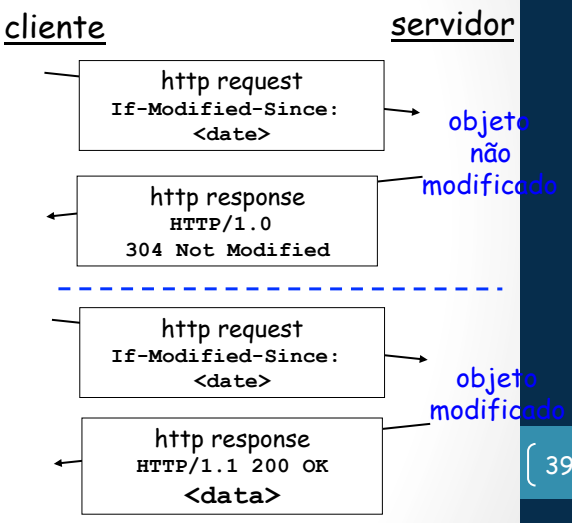
- Necessita de quatro componentes:
  - uma linha de cabeçalho de cookie na mensagem de resposta HTTP;
  - uma linha de cabeçalho de cookie na mensagem de requisição HTTP;
  - um arquivo de cookie mantido no sistema final do usuário e gerenciado pelo navegador do usuário;
  - um banco de dados de apoio no site.

Cookies



GET Condicional: caches no cliente

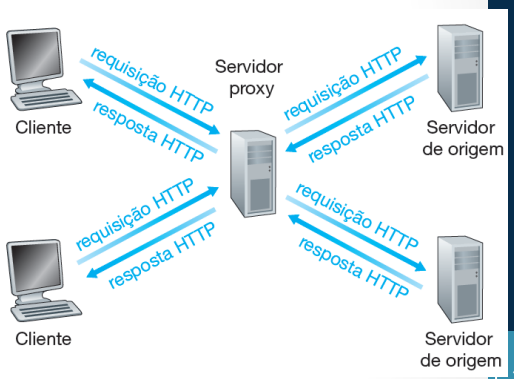
- servidor: só envia o objeto solicitado se sua versão for mais atual que a do cliente
- cliente: especifica, na requisição HTTP, a data da versão armazenada no cache local:
  `If-Modified-Since: <date>`
- servidor: resposta não contém o objeto se a cópia do cliente estiver atualizada:
  `304 Not Modified`



Caches Web (Servidor Proxy)

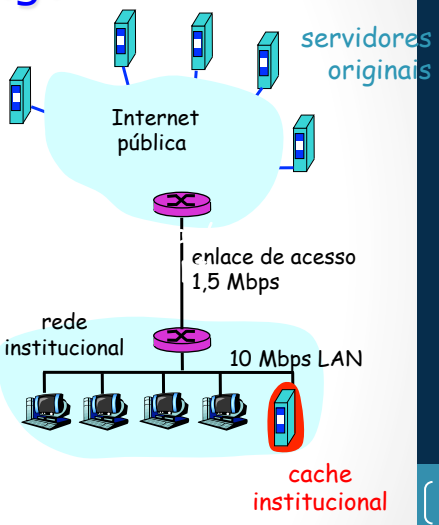
Objetivo: atender o cliente sem envolver o servidor Web, detentor da informação original

- usuário configura o browser:
  - acesso à Web é feito através de um servidor proxy
- cliente envia todos os pedidos http para o proxy:
  - se o objeto existe no cache, o proxy retorna o objeto
  - senão, o proxy solicita o objeto ao servidor original e o envia ao cliente



Por que Web Caching?

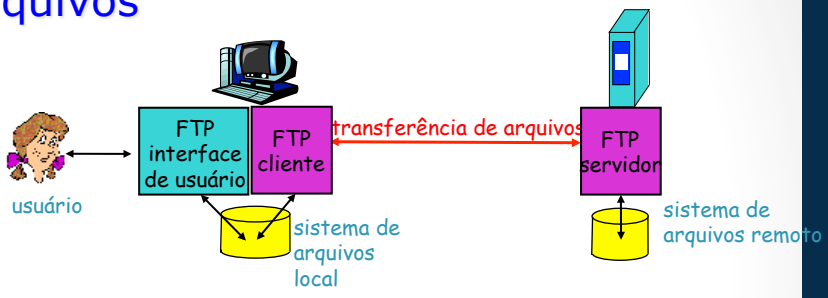
- armazenamento fica "perto" do cliente (p.ex., na mesma rede)
- menor tempo de resposta
- reduz o tráfego para servidores distantes:
  - links externos podem ser caros e facilmente congestionáveis
- caches hierárquicos e cooperativos (NLANR)
- ICP (RFC 2186)
  - Internet Caching Protocol, suportado pelo Squid



Protocolo FTP

TRANSFERÊNCIA DE ARQUIVOS

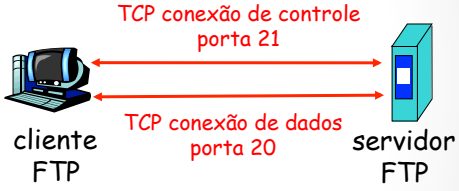
FTP – Protocolo de Transferência de Arquivos



- transferência de arquivos de/para um host remoto
- modelo cliente servidor:
  - *cliente:* lado que inicia a transferência (em qualquer sentido)
  - *servidor:* host remoto
- FTP: RFC 959 (de 1971)
- servidor ftp: porta 21

ftp: controle separado, conexões de dados

- cliente ftp contacta o servidor ftp na porta 21, especificando TCP como protocolo de transporte
- duas conexões TCP paralelas são abertas:
  - controle: troca de comandos e respostas entre cliente e servidor (controle "out of band")
  - dados: dados trocados com o servidor (porta 20); não persistente
- servidor ftp mantém o estado: diretório corrente, autenticação anterior



ftp: comandos, respostas

Exemplos de comandos:

- envio de um texto ASCII sobre canal de controle
- `USER username`
- `PASS password`
- `LIST` retorna lista de arquivos no diretório corrente
- `RETR filename` recupera (obtém) o arquivo
- `STOR filename` armazena o arquivo no host remoto

Códigos de retorno:

- código de status e explicação (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`

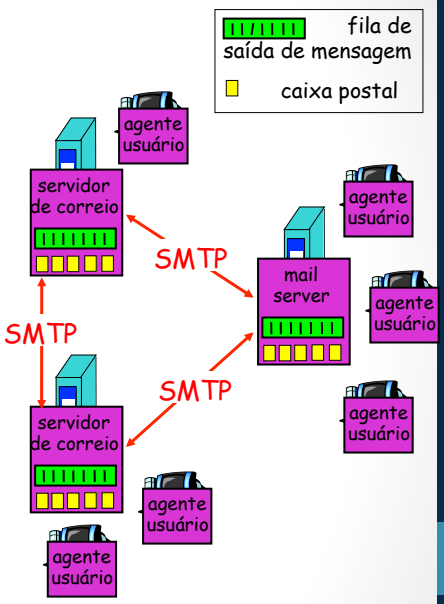
Correio Eletrônico

Três componentes principais:

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

Agente de usuário

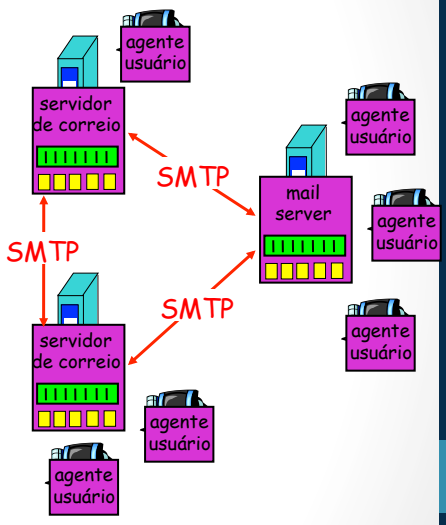
- leitor de correio
- composição, edição, leitura de mensagens
- ex: Mail, Thunderbird, Outlook, pine, ...
- mensagens de entrada e de saída são armazenadas no servidor de correio (smtp)



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 ser enviadas
- protocolo smtp: permite aos servidores de correio trocar mensagens entre eles
  - cliente: servidor de correio que envia
  - servidor: servidor de correio que recebe
- tratamento de erros

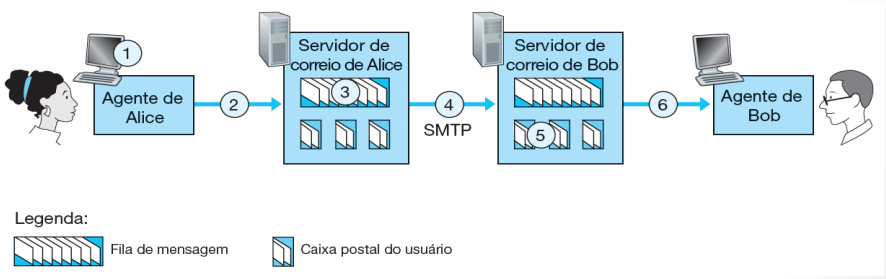


Correio Eletrônico: SMTP [RFC 5321]

- SMTP usa TCP para transferência confiável de mensagens de correio do cliente ao servidor, pela porta 25
- transferência direta: servidor de origem envia para o servidor de destino; não usa servidores intermediários
- na sua forma padrão, as mensagens SMTP devem ser formatadas em código ASCII de 7 bits (legado dos primórdios da Internet) --> codificação/decodificação
- três fases de transferência:
  - apresentação: troca de endereços de origem/destino
  - transferência de mensagens (via TCP); conexões persistentes
  - encerramento
- interação comando/resposta
  - comandos: texto ASCII
  - resposta: código de status e frase de explicação

SMTP

- O SMTP transfere mensagens de servidores de correio remetentes para servidores de correio destinatários:



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: comentários

- SMTP usa conexões persistentes
- SMTP exige que as mensagens (cabeçalho e corpo) estejam em ASCII de 7 bits
- Algumas seqüências de caracteres não são permitidas nas mensagens (ex., `CRLF.CRLF`). Dados binários têm que ser codificados em ASCII
- `CRLF.CRLF` indica o final da mensagem

Comparação com http:

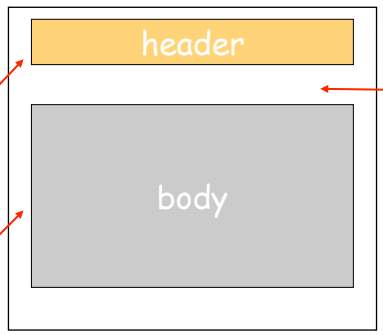
- http: pull (recuperação)
- smtp: push (envio)
- smtp exige que cada msg esteja em formato ASCII, inclusive o corpo
- Dados http mantêm formato original
- http: cada objeto encapsulado na sua própria mensagem de resposta
- smtp: múltiplos objetos são enviados numa única msg

Formato das Mensagens

SMTP: protocolo para troca de mensagens

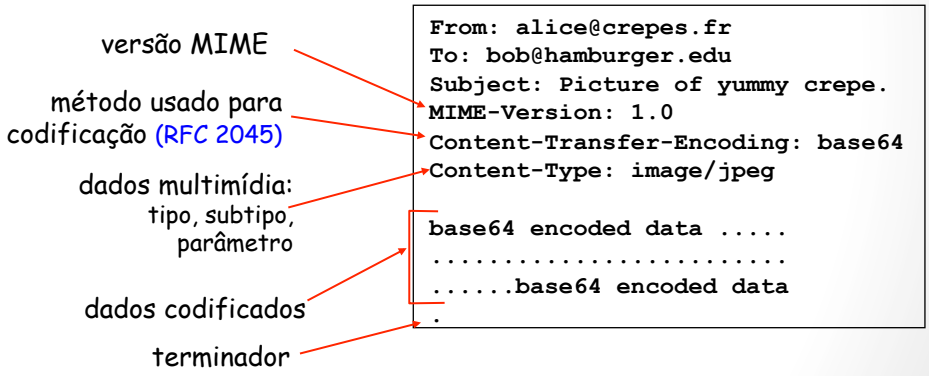
RFC 5322: define formato das linhas de cabeçalho das mensagens

- linhas de cabeçalho:
  - From: alice@deinf.ufma.br
  - To: bob@icmc.usp.br
  - Subject: Slides de Redes
- corpo:
  - a mensagem em ASCII, somente com caracteres



Formato das Mensagens: extensões multimídia

- MIME: *Multipurpose Internet Mail Extensions* – para transmitir textos que não estão no padrão ASCII. Ex: imagens, outro idioma que não o Inglês
- linhas adicionais no cabeçalho declaram o tipo de conteúdo



Tipos MIME (RFC 2046)

`Content-Type: type/subtype; parameters`

Text

- `text/plain`, `text/html`
- `text/plain; charset="ISO-8859-1"` (línguas latinas)

Image

- `image/jpeg`, `image/gif`

Audio

- `audio/basic` (8 bits), `audio/32kadpcm` (32 Kbps)

Video

- `video/mpeg`, `video/quicktime`

Application

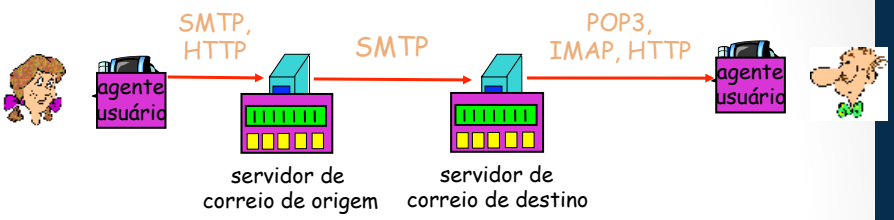
- dados que precisam ser processados por uma aplicação antes de serem apresentados
- `application/msword`, `application/octet-stream` (dados binários genéricos)

Tipo Multipart

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

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

Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

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

delimitador

Protocolos de acesso a correio



- SMTP: envia emails do servidor de origem para o de destino; e do agente de usário para o servidor de origem
- Protocolo de acesso: recupera mensagens do servidor de correio
  - POP: Post Office Protocol [RFC 1939]
    - autorização (agente <--> servidor), download, atualização
  - IMAP: Internet Mail Access Protocol [RFC 3501]
    - maiores recursos (mais complexo)
    - manipulação de caixas postais remotas (criação de pastas, leitura parcial de mensagens, busca, etc.)
  - HTTP: Hotmail, Yahoo, Gmail (http: browser <--> servidor)

Protocolo POP3

fase de autorização
- comandos do cliente:
  - `user:` nome do usuário
  - `pass:` senha
- respostas possíveis do servidor:
  - `+OK`
  - `-ERR`

fase de transação, cliente:
- `list:` lista mensagens e tamanhos
- `retr:` recupera mensagem pelo número
- `dele:` apaga
- `quit`

- POP3 não mantém estado entre sessões dos clientes

```
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
```


DNS

SERVIÇO DE NOMES

DNS: Domain Name System

Pessoas: muitos identificadores
- RG, nome, passporte

Hosts da Internet, roteadores:
- endereços IP (32 bits) - usados para endereçar datagramas
- nomes - usados por humanos

- Como relacionar nomes de hosts com endereços IP?

Domain Name System:
- *base de dados distribuída:* implementado numa hierarquia de vários *servidores de nomes*
- *protocolo da camada de aplicação:* hosts, roteadores comunicam-se com servidores de nomes para *resolver nomes* (tradução nome/endereço)
  - função interna da Internet, implementada como um protocolo da camada de aplicação
  - complexidade na "borda" da rede
  - outros serviços: aliases de host e servidor de email, load balancing
- máquinas Unix: Bind, porta 53, udp
- RFCs 1034, 1035

DNS: Arquitetura

Por que não usar um DNS centralizado?
- ponto único de falha
- volume de tráfego
- base de dados distante
- manutenção

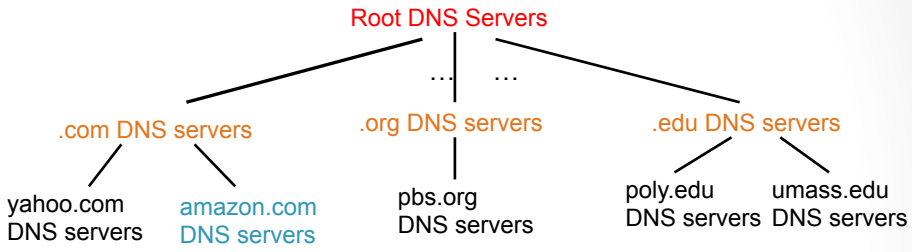
Não tem escalabilidade!

Solução distribuída, hierárquica: nenhum servidor tem todos os mapeamentos de nomes para endereços IP

servidor de nomes local:
- cada empresa/instituição tem um *servidor de nomes local (default)*
- Consultas dos computadores locais ao DNS vão primeiro para o servidor de nomes local

servidor de nomes com autoridade:
- para um computador: sempre contém o nome e o endereço IP daquele computador
- muitos servidores de nomes locais também são *authoritative*

DNS: uma base de dados distribuída e hierárquica



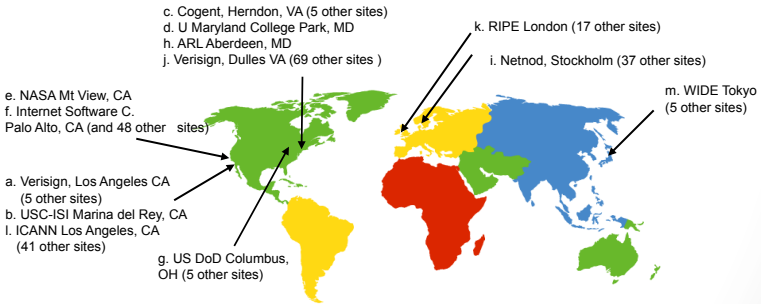
Cliente necessita do end. IP para www.amazon.com

1ª aproximação:

- cliente pesquisa no servidor raiz (root) sobre servidor DNS de '.com'
- cliente pesquisa no servidor DNS '.com' sobre servidor DNS de amazon.com
- cliente pesquisa no servidor DNS de amazon.com sobre endereço IP de www.amazon.com

Servidores de Nomes Raiz

- Contactados pelos servidores de nomes locais quando estes não conseguem resolver um nome
- Funcionalidades:
  - buscam servidores de nomes com autoridade se o mapeamento do nome não for conhecido;
  - Obtêm o mapeamento nome → IP;
  - retornam o mapeamento para o servidor de nomes local.



Servidores TLD e Authoritative

Servidores Top-level domain (TLD):

- respondem pelos domínios com, org, net, edu, aero, jobs, museums e todos os domínios top-level de países: br, uk, fr, cn, jp
- Empresa Network Solutions mantém servidores TLD para domínio .com
- Instituição Educause mantém servidores TLD para .edu
- no Brasil, a RNP mantém servidores TLD para Universidades e instituições de pesquisa: edu.br, br

Servidores DNS com Autoridade (Authoritative):

- fornecem mapeamentos 'com autoridade' dos nomes de hosts da organização para endereços IP
- administrados pelas próprias organizações ou por um provedor de serviços (ISP)

Servidores de Nomes Locais

- Oficialmente, não pertecem à hierarquia do DNS
- Cada provedor (residencial, empresarial, universidades) possui um servidor DNS local
  - também chamado de 'default name server'
- Sempre que um host faz uma pesquisa de nomes, esta é enviada primeiramente ao servidor DNS local:
  - este possui uma cache com pares de traduções [nome, endereço] recentes (pode estar desatualizada!)
  - DNS local funciona como um proxy, repassando a pesquisa para cima, na hierarquia, caso não possua a resposta

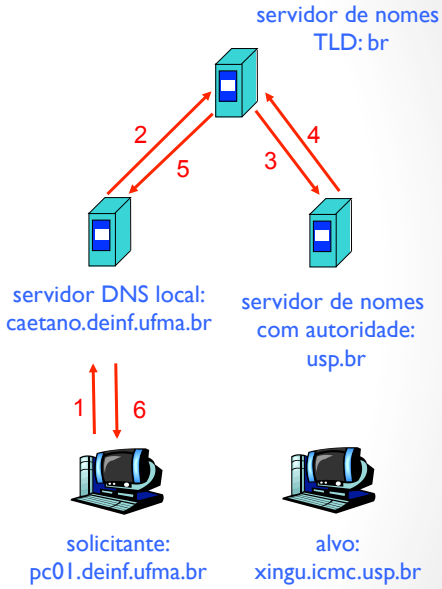
DNS: exemplo de resolução de nomes

Host **pc01.deinf.ufma.br** quer o endereço IP de **xingu.icmc.usp.br**

1. contacta seu servidor DNS local, **caetano.deinf. ufma.br**

2. caetano.deinf.ufma.br contacta o servidor top-level do **domínio br**, se necessário

3. o servidor TLD br contacta o servidor de nomes com autoridade do domínio **usp.br**, se necessário

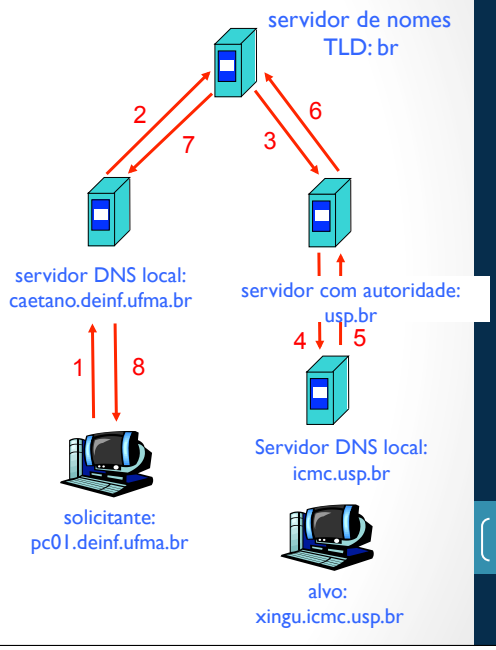


DNS: exemplo

4. o servidor de nomes usp.br contacta o servidor de nomes local do domínio **icmc.usp.br**, se necessário

O Servidor *top-level*:

- pode não conhecer o servidor de nomes local para um certo nome
- neste caso, pesquisa no *servidor de nomes com autoridade* do domínio superior



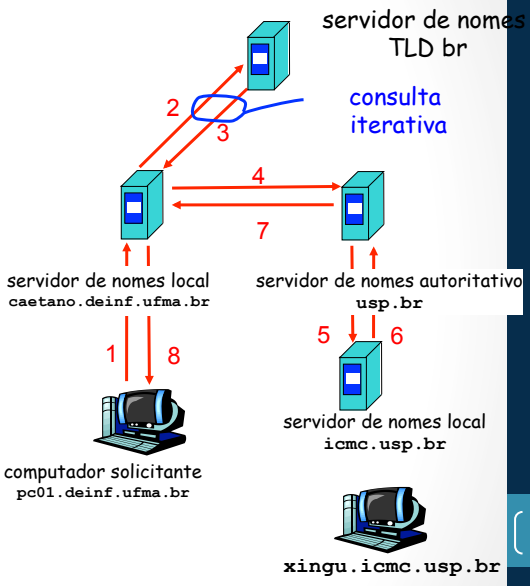
DNS: consultas iterativas e recursivas

consulta recursiva:

- transfere a tarefa de resolução do nome para o servidor de nomes consultado
- mais mensagens

consulta iterativa:

- servidor contactado responde com o nome de outro servidor de nomes para contato
- diminui a carga na rede e nos servidores



DNS: caching

- Quando um servidor de nomes aprende um mapeamento, ele o guarda em sua cache
  - entradas na cache são apagadas depois de um tempo (TTL)
  - tipicamente, servidores TLD são guardados na cache dos DNS's locais
    - evita visitar os servidores Raiz com frequência
- Entradas na cache podem ficar desatualizadas
  - tradução nome→endereço é *best effort*
  - se um host muda seu endereço, pode levar um tempo até que toda a Internet fique sabendo (TTLs expirem)
- RFC 2136 define os mecanismos de update/notificação do DNS

Registros DNS

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

RR format: `(name, value, type, ttl)`

type=A

- **`name`** é o nome do host
- **`value`** é o endereço IP

type=NS

- **`name`** é um domínio (p.ex., deinf.ufma.br)
- **`value`** é o hostname do servidor de nomes com autoridade para este domínio

type=CNAME

- **`name`** é um "apelido"
- **`value`** é o nome "canônico" (real)
- `www.deinf.ufma.br` é de fato `www.caetano.deinf.ufma.br`

type=MX

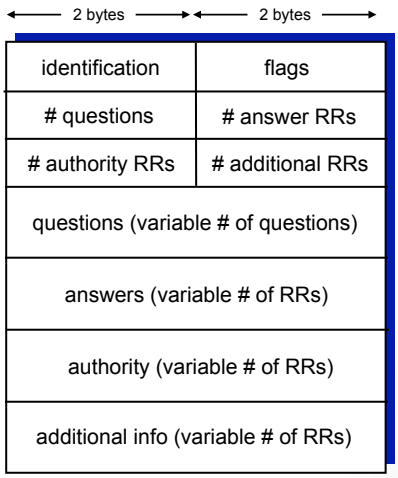
- **`value`** é o nome de um servidor de nomes associado com **`name`**

DNS: protocolo e mensagens

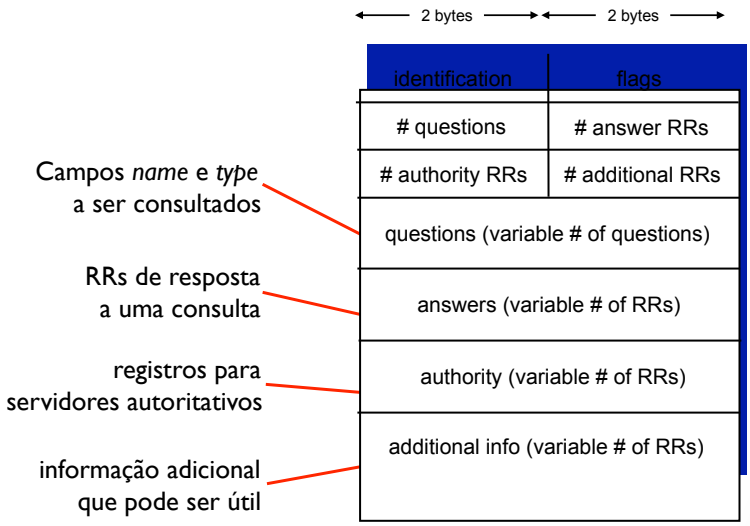
Protocolo DNS: mensagens de *consulta* e *resposta* , ambas com o mesmo *formato*

Cabeçalho:

- identificação: número de 16 bits identifica consulta; resposta usa o mesmo número
- flags:
  - consulta ou resposta
  - recursão desejada
  - recursão disponível
  - resposta é autoritativa



DNS: protocolo e mensagens



P2P

PEER-TO-PEER