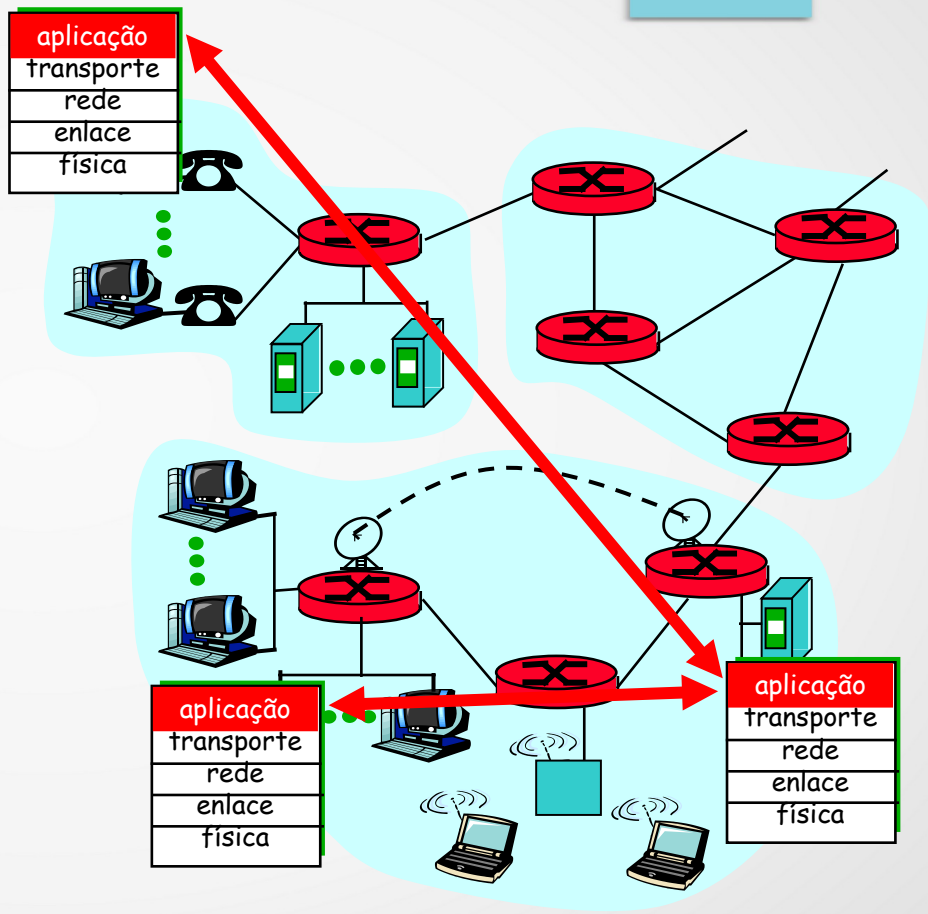
Camada de Aplicação

Principais Protocolos



Camada de Aplicação

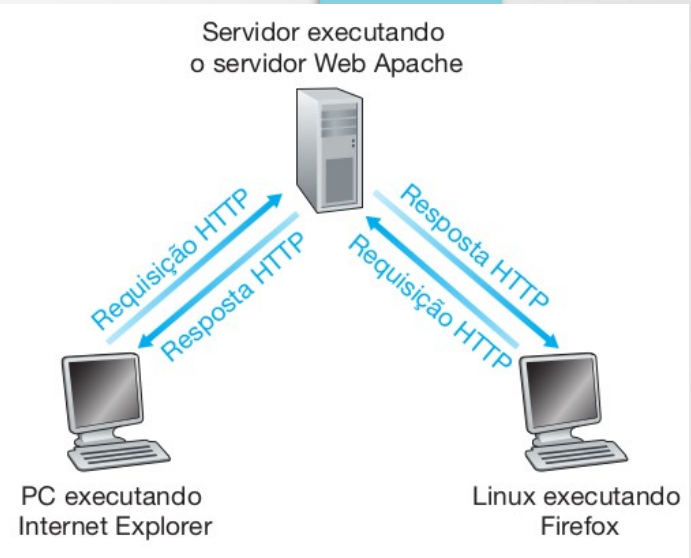
Objetivos:

- Examinar o funcionamento de algumas aplicações e protocolos de aplicação populares, tais como: http, ftp, smtp, pop, dns e bittorrent

Protocolo HTTP (RFC:1945,7230 e 7540)

HyperText Transfer Protocol

- protocolo da camada de aplicação
- usa o TCP, porta 80
- Modelo cliente/servidor:
  - *cliente:* geralmente um browser que solicita, recebe e apresenta objetos da Web
  - *server:* envia objetos em resposta a pedidos.
    - Ex: Apache HTTP, Nginx, MSIIS, cloudflare, etc.



- HTTP é stateless
  - servidor não mantém informação sobre os pedidos passados pelos clientes
  - **Protocolos que mantém informações de estado (statefull) são complexos!**
    - tem que manter e organizar dados antigos
    - se ocorrer um crash as informações podem ser perdidas ou gerar inconsistências entre o cliente e o servidor

Páginas web

HTTP é utilizado para troca de páginas web e objetos entre cliente e servidor

- Página web é constituída por objetos
  - Arquivo HTML, imagem, vídeo, javascript, CSS, etc
- Ex: uma página web que possui um arquivo base html e esse arquivo referencia 4 imagens, terá um total de 5 objetos a serem transferidos.
- A url de uma paǵina web tem 2 elementos importantes:
  - nome (ou endereço do host) e o endereço para o objeto. Ex:

https://www.sbv.ifsp.edu.br/images/marca-ifsp-sbv-small.jpg

Exemplo de Operação. Usuário entra com a URL: www.ifsp.edu.br/Departmento/home.index

1a. cliente http inicia conexão TCP ao servidor http em www.ifsp.edu.br Porta 80 é a default para o servidor http .

1b. servidor http no host www.ifsp.edu.br esperando pela conexão TCP na porta 80. "aceita" conexão, notificando o cliente

2. cliente http client envia http *request message* (contendo a URL) para o socket da conexão TCP

3. servidor http recebe mensagem de pedido, forma *response message* contendo o objeto solicitado (Departmento/home.index), envia mensagem para o socket

4. servidor http fecha conexão TCP.

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

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

tempo

Conexões persistentes e não-persistentes

Não-persistente

- http/1.0: server analisa pedido, envia resposta e fecha a conexão TCP
- 2 RTTs para obter um objeto
  - Conexão TCP
  - solicitação e transferência do objeto
- cada transferência sofre atraso por causa do mecanismo de slow-start do TCP e sobrecarrega o S.O.
- muitos browsers abrem várias conexões paralelas

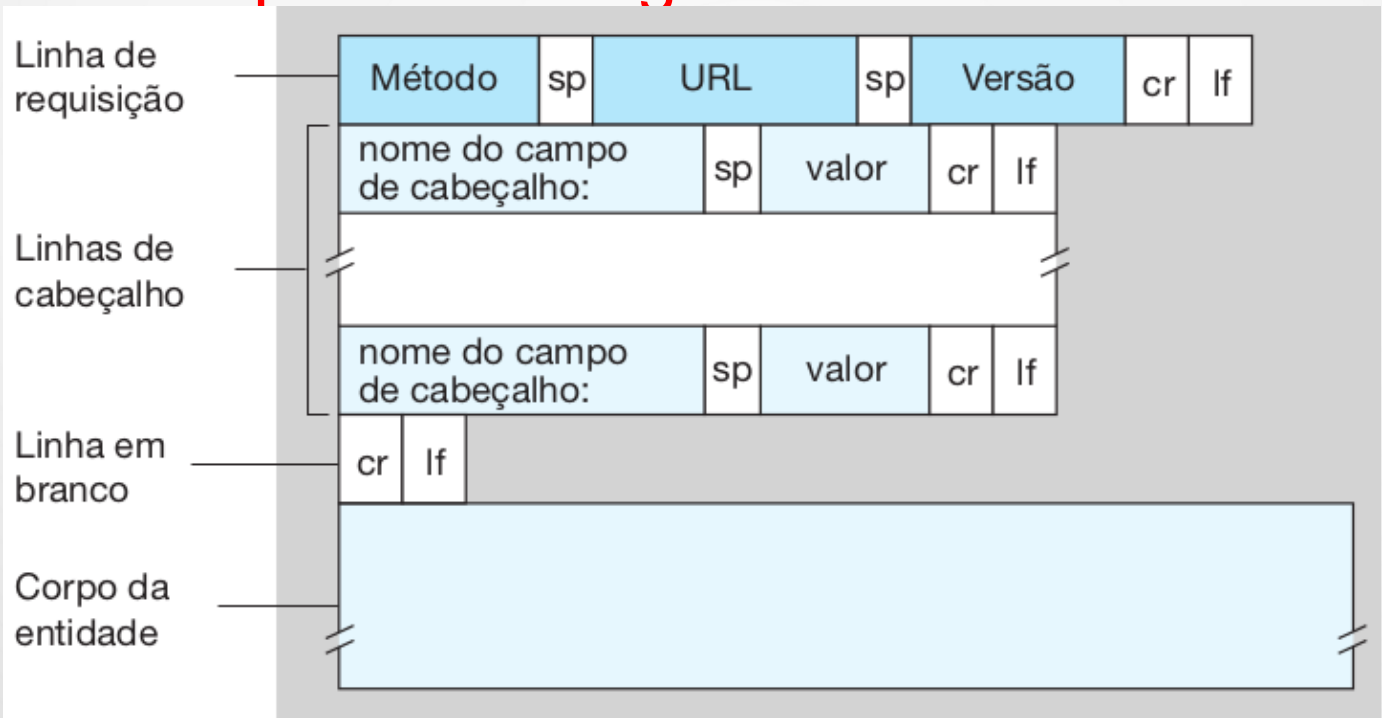
Persistente

- modo default à partir do http/1.1
- Server mantém a conexão aberta após a resposta
- Vários objetos podem ser enviados e requisitados na mesma conexão TCP
- o cliente envia requests tão logo encontre objetos referenciados.
- poucos RTTs, menos slow start.

Formato das Mensagens HTTP

- Há 2 tipos de mensagens HTTP: *request* e *response*
  - ASCII (formato legível para humanos)

- HTTP Request message:



Mensagem Request HTTP

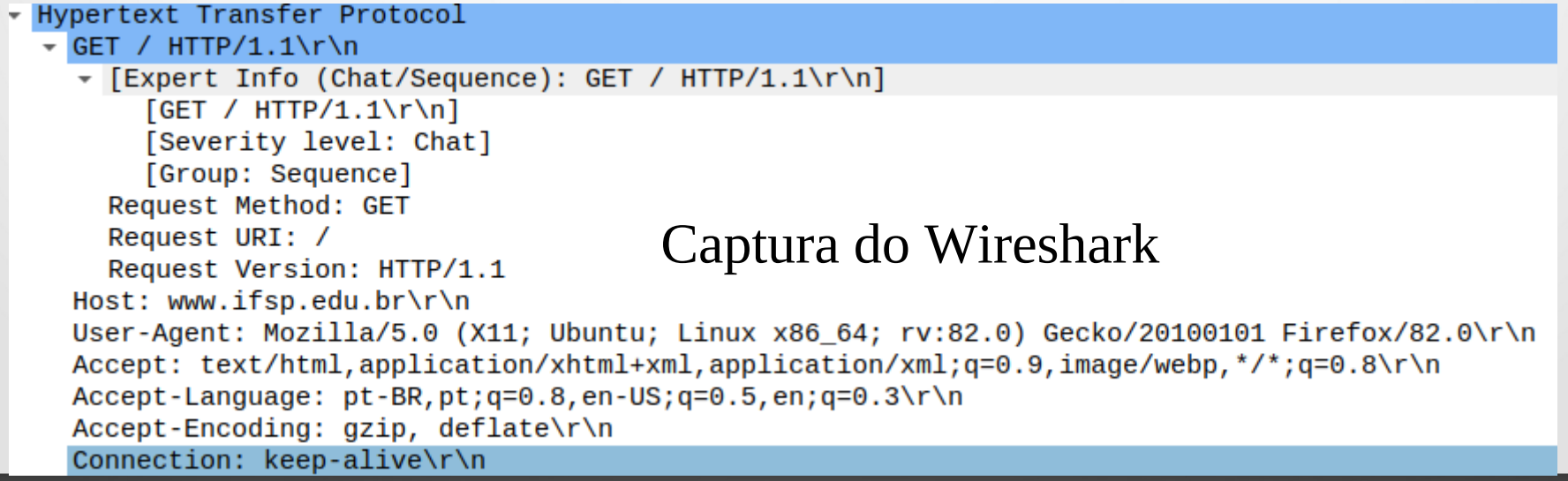
Ex: métodos: GET,POST, HEAD, PUT e DELETE

linha de requisição (método, URL e versão)

```
GET /somedir/page.html HTTP/1.1
Host: www.someshool.edu
Conection:close
User-agent: Mozilla/5.0
Accept-language:fr
(extra carriage return, line feed)
```

linhas de cabeçalho

Carriage return, line feed : indica fim da mensagem



Captura do Wireshark

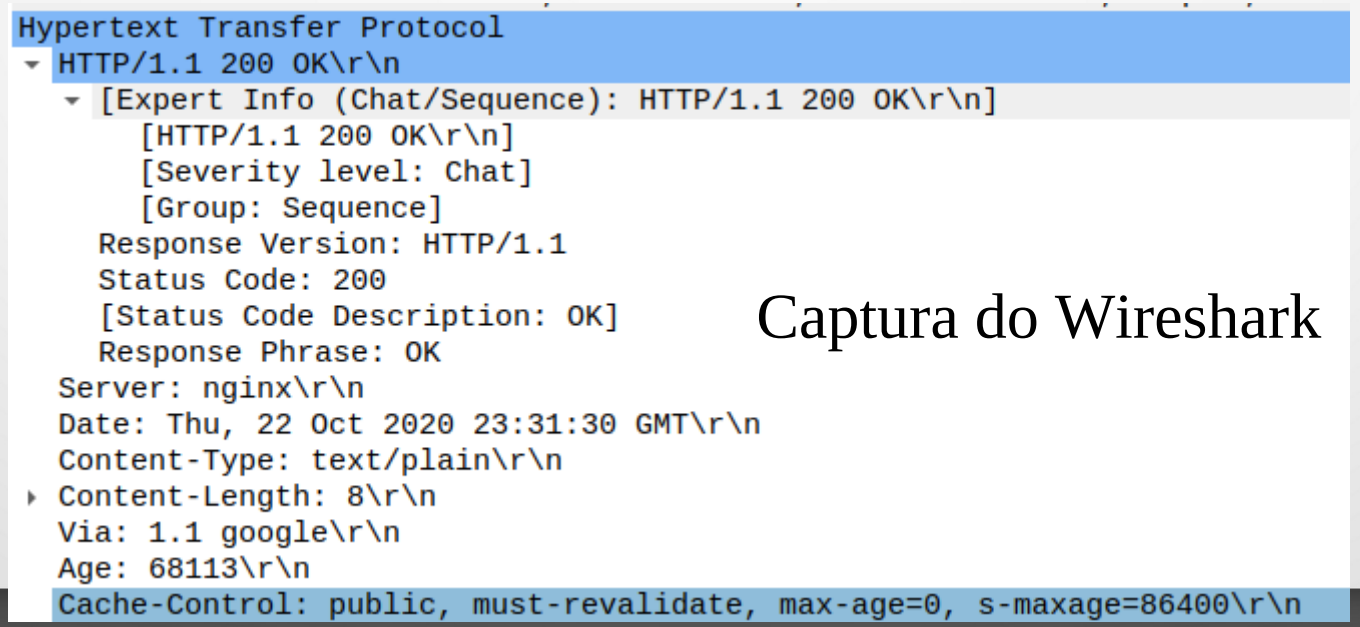
Mensagem Response HTTP

Linha de status
(versão, código, msg)

Linhas de
cabeçalho

Corpo

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



Captura do Wireshark

Códigos de status das respostas

200 OK

Requisição aceita, objeto requisitado será enviado

301 Moved Permanently

Objeto requisitado foi removido, nova localização será informada

400 Bad Request

Requisição não entendida pelo servidor

404 Not Found

Objeto requisitado não foi encontrado

505 HTTP Version Not Supported

Versão requisitada não é suportada pelo servidor

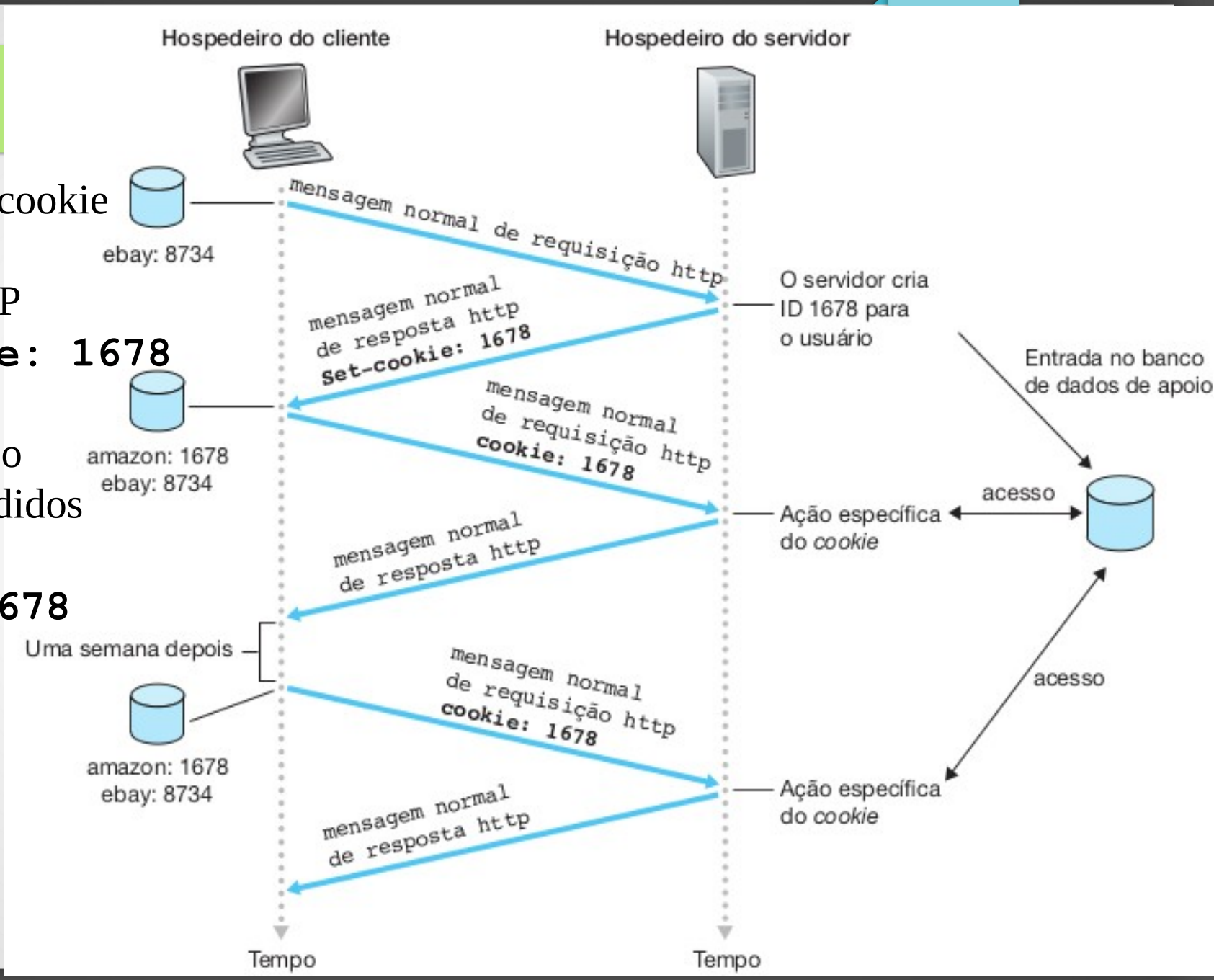
Cookies

- Cookie é um mecanismo que permite que um site monitore os usuários.
- São gerados e lembrados pelo servidor, para serem utilizados mais tarde, com os propósitos de:
  - autenticação
  - lembrar preferências dos usuários ou escolher prévias
  - sugerir recomendações
  - manter status da sessão
- cookies podem ser usados para criar uma camada de sessão de usuário sobre HTTP sem estado, tornando uma solução statefull.

Cookies

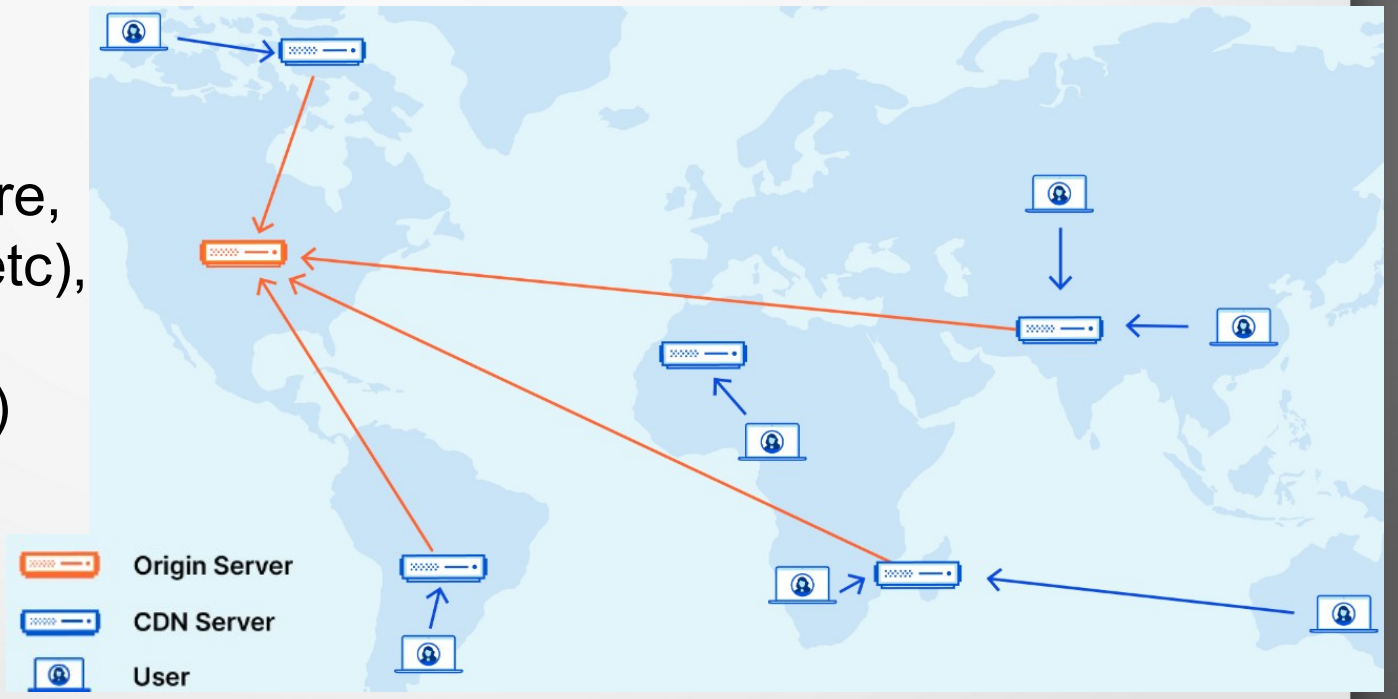
Servidor cria cookie p/ cliente na resposta HTTP **`Set-cookie: 1678`**

Cliente envia o cookie em pedidos posteriores **`cookie: 1678`**



CDN - Rede de Distribuição de Conteúdo

- Com a larga utilização de aplicações multimídia, como fotos, vídeos e streamings vídeo e áudio, redes CDN tem sido muito utilizadas.
- Uma CDN instala caches geograficamente próximos dos usuários finais, para que acessem mais rapidamente o conteúdo.
- Existem CDNs compartilhadas (GoCache, Cloudflare, Akamai, Limelight, etc), e CDNs dedicadas (Google e Microsoft)



Fonte da imagem: https://www.cloudflare.com/pt-br/learning/cdn/what-is-a-cdn/

Problemas HTTP/1.1 e as novas versões

- Com o HTTP1.1 foi introduzida a possibilidade de realizar múltiplos GETs em uma única conexão TCP. Porém:
  - O servidor responde em ordem FCFS (First-come-first-served)
  - Pequenos objetos podem ter que esperar muito se houver grandes objetos antes. Ex: um grande arquivo JavaScript pode atrasar a transmissão dos demais objetos.
  - Alguns objetos afetam mais a percepção do usuário, por exemplo, elementos do final da página, ou de segundo plano (como um arquivo CSS) impactam menos do que os elementos da parte superior da página.
  - Retransmissão de segmentos perdidos paralisam todas as transmissões, no que é conhecido como HOL (head-of-line) blocking .
- Para amenizar, a maioria das aplicações abre conexões TCP simultâneas para realizar o paralelismo.
- As novas versões do HTTP visam melhorar esses problemas.

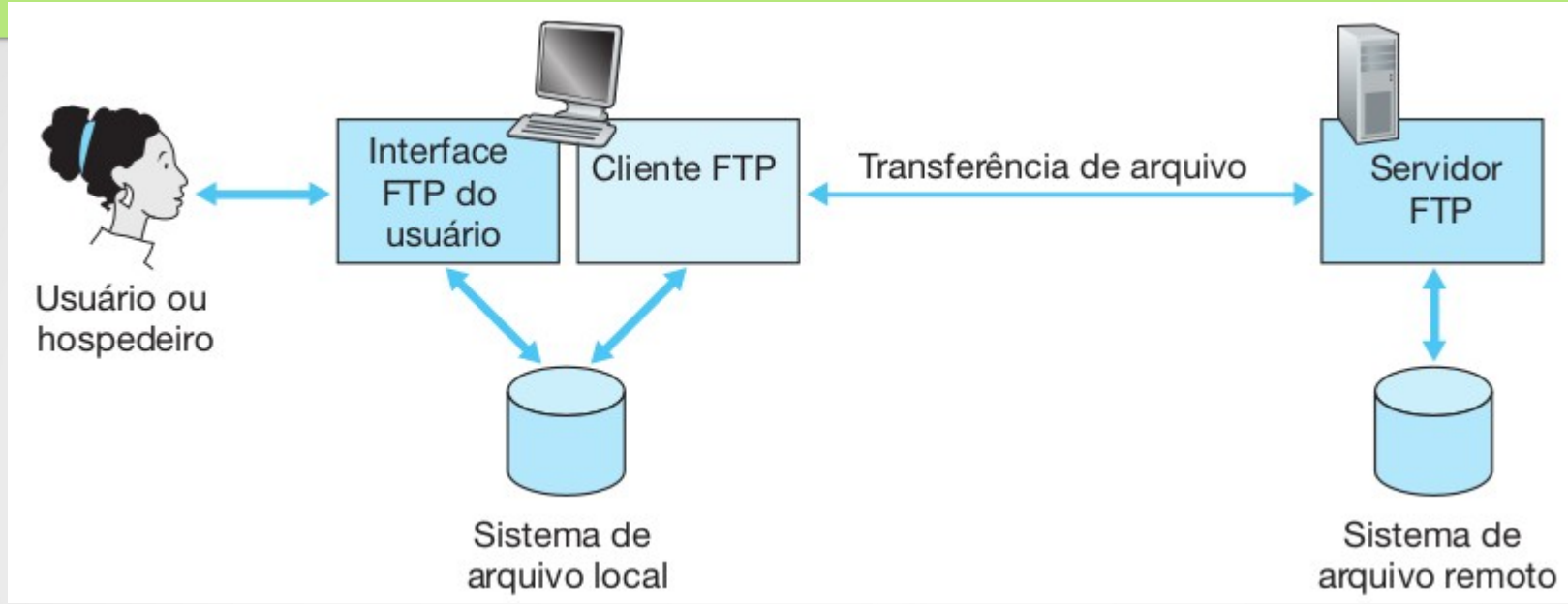
HTTP/2 (RFC 7540)

- HTTP/2 foi criado principalmente para reduzir o delay em requisições com múltiplos objetos, tornando o desempenho bem melhor. Entre as melhorias estão:
- **Prioridade ponderada:** ordem de transmissão baseada em prioridade dos objetos para o cliente (não precisa ser FCFS)
- **Multiplexação:** permite vários streamings simultâneos, enviando os objetos em paralelo e o cliente decide o que renderizar primeiro.
- **Server push:** permite ao servidor enviar objetos ao cliente, antes da requisição.
- **Compactação do cabeçalho:** Usa HPACK, mecanismo de compactação que elimina informações redundantes dos cabeçalhos. O HTTP/1.1 é textual, o HTTP/2 é binário.
- **Objetos divididos em partes:** envia as partes dos vários objetos alternadamente, de modo que um grande objeto não atrase o envio de objetos menores.
- **Elimina o HOL blocking da aplicação**: mas há o HOL blocking de transporte.
- Em 09/24, ainda é a versão predominante, sendo usado em cerca de 35% dos sites da Internet (Fonte: https://w3techs.com/technologies/details/ce-http2).

HTTP/3 (RFC 9114)

- HTTP/3, homologado em 06/06/2022, fez alterações substanciais em relação às versões anteriores.
  - **Trocou o TCP, pelo QUIC** (Quick UDP Internet Connections), que é executado sobre o UDP.
  - Devido ao QUIC possuir um handshake mais rápido, é capaz de manter uma conexão segura, substituindo o conjunto HTTP/2+TCP+TLS.
  - Possui **recursos de segurança obrigatórios**, de modo que não é possível estabelecer uma conexão sem criptografia.
  - Em algumas situações possui latência e tempo de carregamento melhores que as versões anteriores, chegando a ser 3x mais rápido que o HTTP/1.1 em alguns condições.
- Em 09/24, já era usado em cerca de 31% dos sites da Internet, segundo a w3techs (https://w3techs.com/technologies/details/ce-http3).

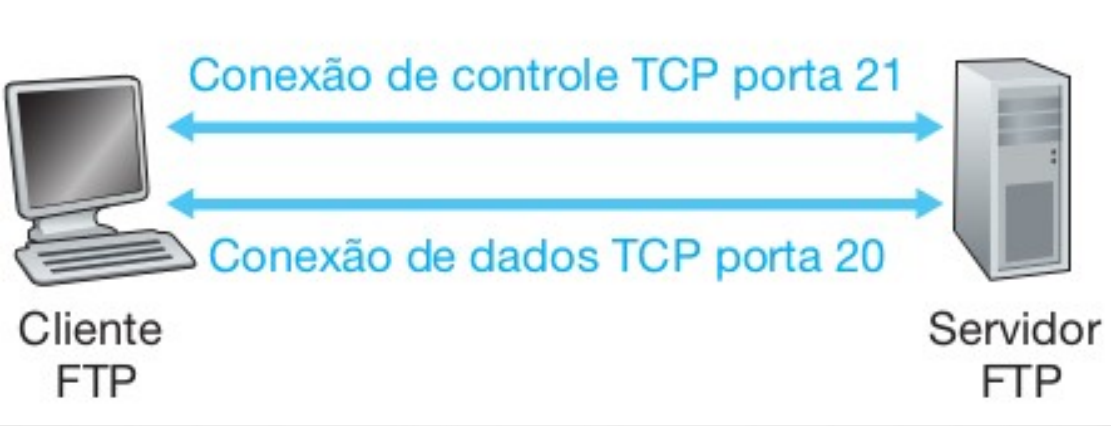
ftp: o protocolo de transferência de arquivos



- Protocolo para transferência de arquivos entre hosts remotos
- Modelo cliente servidor
  - ***Cliente:*** quem inicia a transferência
  - ***Servidor:*** host remoto
- Especificado pela RFC 959.

ftp: controle separado, conexões de dados

- cliente ftp contata 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 (ex: put, get).
  - dados: dados do arquivo trocados com o servidor



ftp: comandos, respostas

Exemplos de comandos

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

Exemplos de códigos de retorno

- código de status e frase (idem 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

Aplicação para o envio de e-mails. Possui três componentes principais:

Agente de usuário (leitor de correio, ex: MS Outlook, Thunderbird, apps)

- Faz composição, edição e leitura das mensagens.

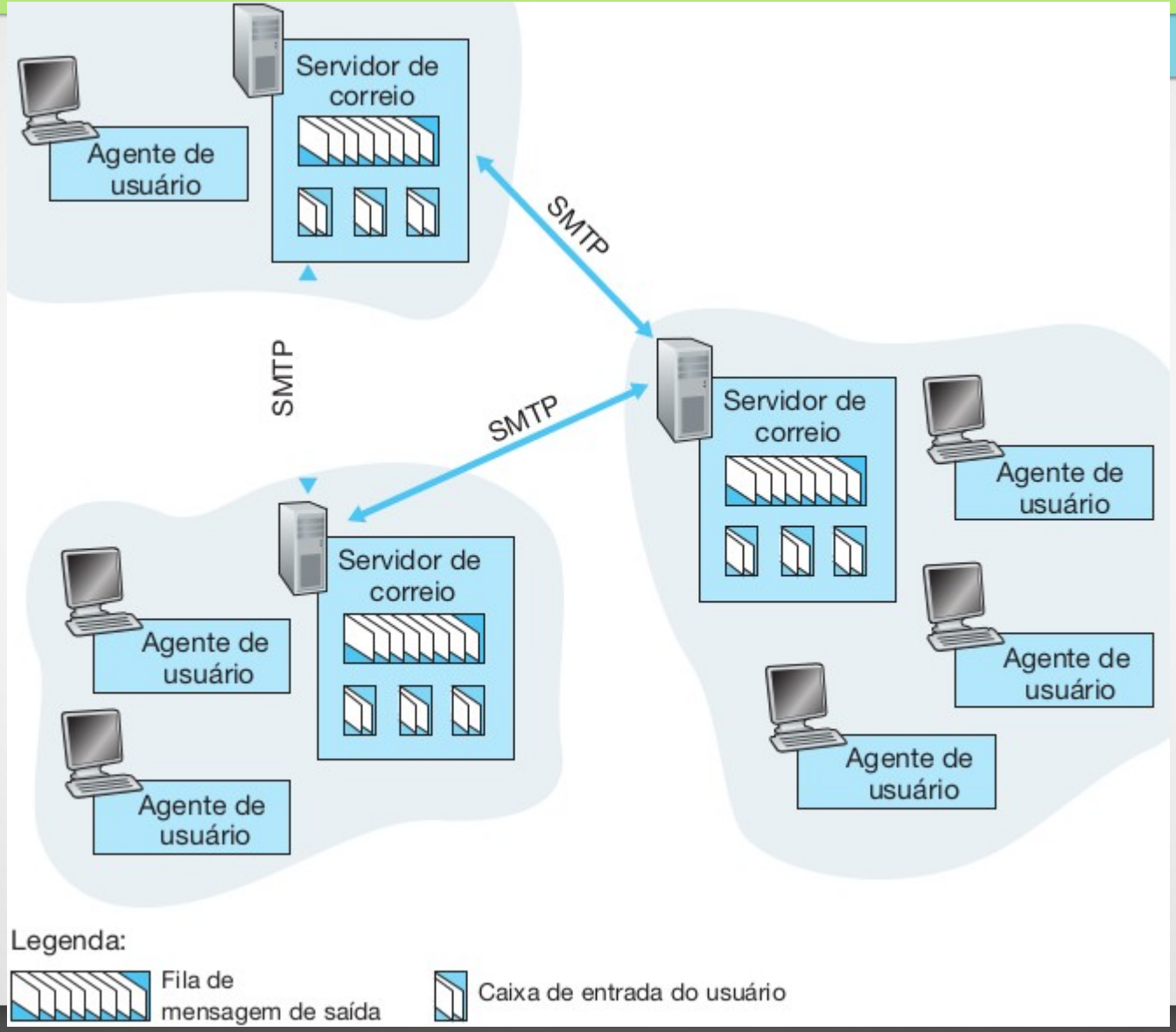
Servidores de Correio (ex: postfix, sendmail, exchange)

- caixa postal contém mensagens que chegaram (ainda não lidas)
- fila de mensagens contém as mensagens de correio a serem enviadas

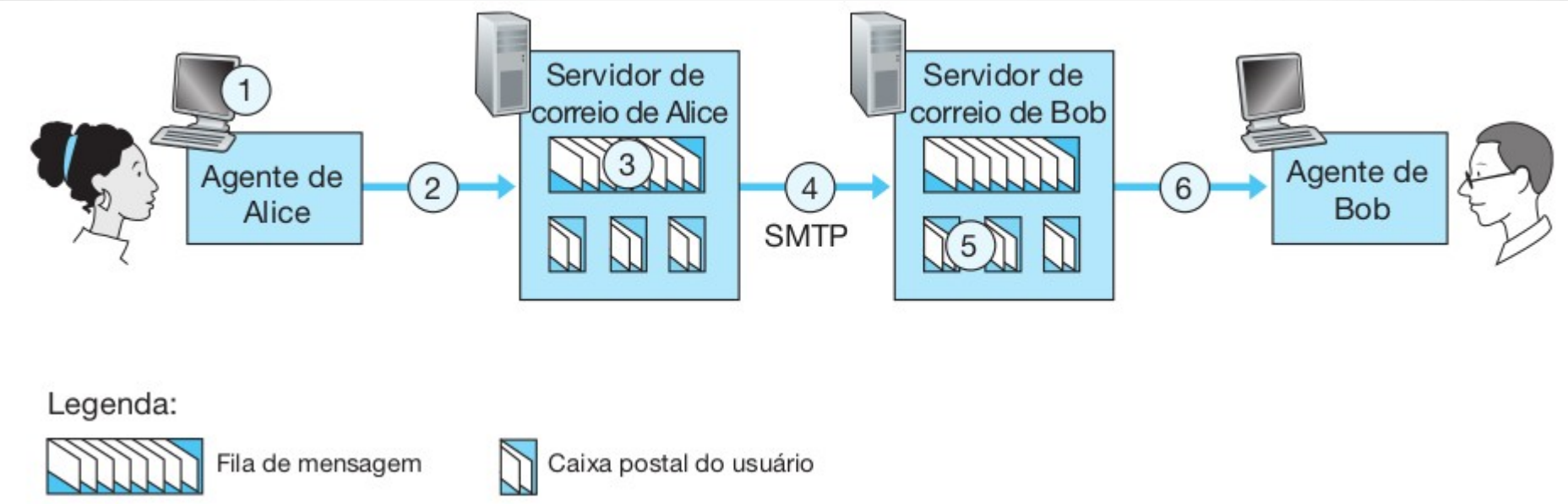
Protocolo de aplicação SMTP (Simple Mail Transfer Protocol): regras para a troca de mensagens entre os servidores

- Usa TCP, porta 25 (desativada atualmente) e porta 587 (exige autenticação)
- Realiza a interação através de comando (texto ASCII) e resposta (código de status e frase)
- mensagens são formatadas em código ASCII de 7 bits
- Algumas sequências de caracteres não são permitidas (ex: CRLF.CRLF indica o fim).
- Usa conexão persistente
- múltiplos objetos são enviados na mesma mensagem (multipart message)

Correio eletrônico



Correio eletrônico: servidores de correio



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
```

Formato das Mensagens

RFC 5321: padrão para a troca de mensagens de e-mail

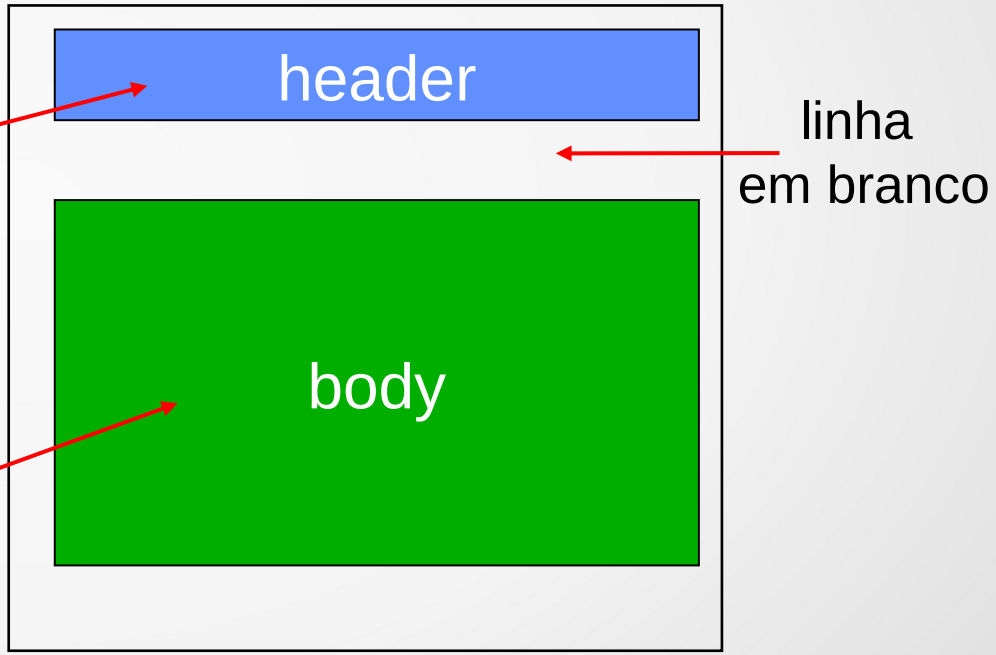
RFC 2822: sintaxe para mensagens de e-mail (análogo ao HTML)

linhas de cabeçalho, ex:,

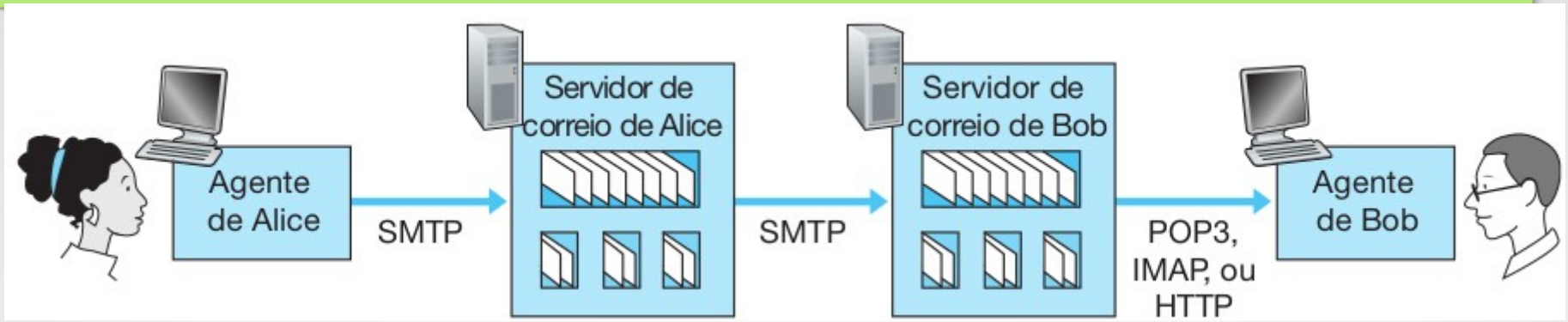
- From:
- To:
- Subject:

diferente dos comandos SMTP!

- corpo
  - a “mensagem”, somente com caracteres ASCII



Protocolos de acesso ao correio



- SMTP: entrega e armazena no servidor do destinatário
- Protocolo de acesso: recupera mensagens do servidor
  - POP3: Post Office Protocol [RFC 1939]
    - autorização (agente <->servidor) e download
  - IMAP: Internet Mail Access Protocol [RFC 3501]
    - melhores recursos que o POP3(mais complexo)
    - permite manipulação de mensagens armazenadas no servidor e organização em pastas
  - HTTP: utilizado por webmails como Gmail, Hotmail, Yahoo! Mail, etc, que utilizam uma interface web sobre o SMTP e IMAP.

POP3 e IMAP

- POP3, permite dois modos de funcionamento:
  - “download e excluir”. Nesse caso o cliente não pode reler um e-mail, caso utilize agentes de usuário em locais diferentes.
  - “download e manter”: Cópia da mensagem é mantida, permitindo acesso em agentes de usuário diferentes.
- POP3 não mantém estado entre as sessões distintas.
- Já o IMAP permite manter todas as mensagens no servidor e que o usuário organize as mensagens em pastas diferentes.
- Em geral, uma nova mensagem vai para a pasta INBOX, mas o usuário pode criar filtros e alterar esse padrão.
- IMAP mantém estado do usuário entre sessões distintas.

protocolo POP3

fase de autorização

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

fase de transação, cliente:

- **`list:`** lista mensagens e tamanhos
- **`retr:`** recupera mensagem pelo nº
- **`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
```

DNS: Domain Name System

- Pessoas memorizam com mais facilidade nomes do que números. Afinal, o que é mais fácil de memorizar: http://www.google.com ou o IP 172.217.173.78 ?
- DNS é um protocolo de camada de aplicação que realiza a resolução de nomes, ou seja traduz o nome do host em endereço IP.
- Ex: usuário digita o nome de um site. Antes do HTTP ocorre uma consulta ao servidor DNS para resolver este nome, ou seja, descobrir o endereço IP.
- O DNS é uma base de dados distribuída implementada numa hierarquia de muitos servidores de nomes.
- **Principais serviços do DNS:**
  - Tradução de hostname para endereço IP
  - Host aliasing: canonical, alias names
  - Mail server aliasing
  - Balanceamento de carga: réplicas do servidor com vários endereços IP associados ao mesmo nome.

Servidores de Nomes DNS

E por quê não centralizar o DNS?

- Um ponto único de falha
- Volume de tráfego muito alto
- Base de dados distante da maioria dos usuários
- Manutenção difícil

- *Nenhum servidor tem todos os mapeamentos de nomes para endereços IP*
- servidores de nomes locais:
  - Cada ISP ou empresa tem um *servidor de nomes local*, que irá atender as consultas dos computadores locais.
  - Quando não souber, enviará a requisição para outros servers

Estrutura hierárquica DNS

- **Servidor raiz:**

Apenas 13 no mundo (com várias instâncias de réplica). Fornecem os endereços de servidores TLD.

- **Servidor de Domínio de Alto Nível (TLD):**

Responsáveis por domínios de alto nível (ex: com, org, net, edu, gov, etc) e de países (br, pt, fr, jp, uk, etc)

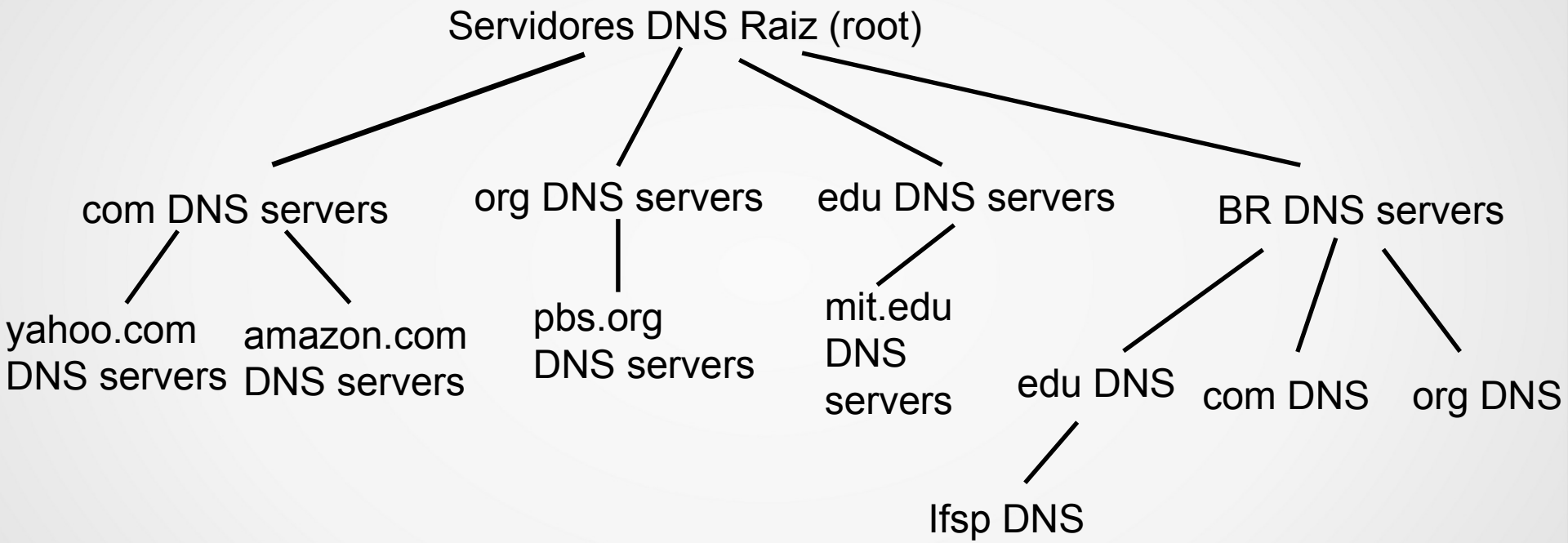
- **Servidor autoritativo:**

Mantém os registros de hosts de organizações que podem ser acessados publicamente. Podem ser mantidos dentro da própria organização ou terceirizados, como em ISP's e hospedagens.

- **Servidor local:**

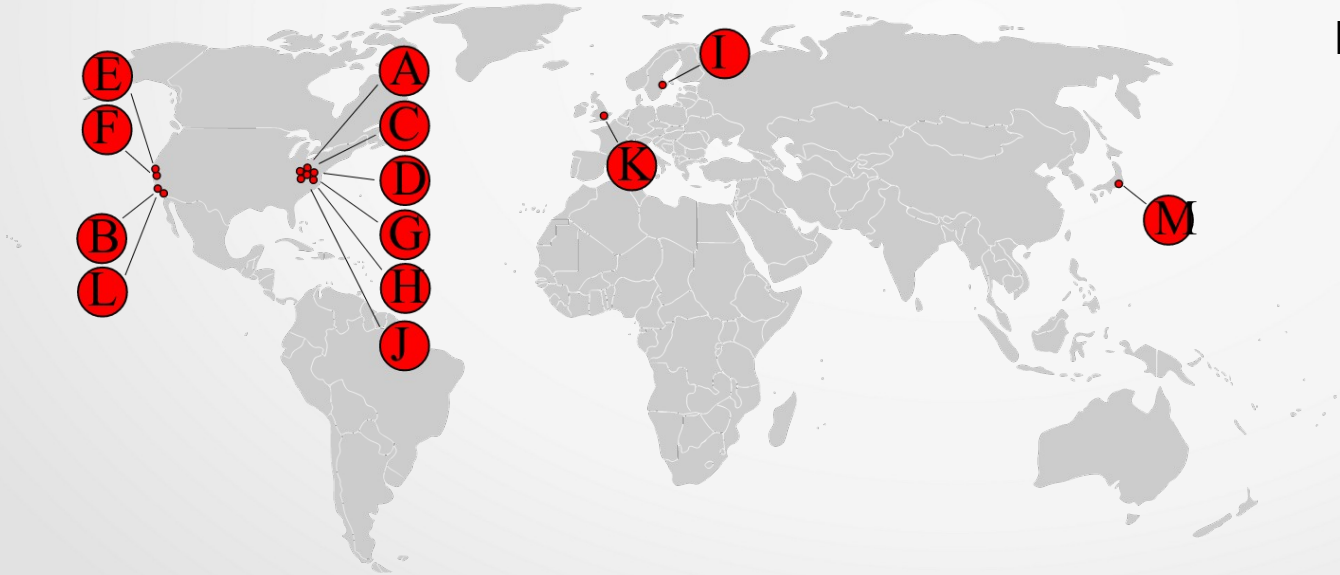
Não pertence à hierarquia diretamente, mas pode ser instalado localmente para acelerar as resoluções. ISP's de acesso mantém servidores DNS locais para seus clientes.

Estrutura hierárquica DNS



DNS: Servidores de Nomes Raiz

- Tem como objetivo resolver nomes que não são resolvidos por servidores locais. Eles basicamente:
  - Recebem a requisição;
  - Conseguem o mapeamento;
  - Retornam o mapeamento p/ o servidor de nomes local.



Existem apenas 13 servidores de nomes raiz no mundo, identificados de A a M, que são mantidos por 12 operadores independentes.

Fonte da imagem: https://commons.wikimedia.org/wiki/File:Root-historic.svg

Instâncias dos servidores DNS raiz

Em 12/09/2024, haviam 1867 instâncias dos 13 servidores ROOT DNS, que são mantidos por 12 operadores independentes. Fonte: https://root-servers.org/

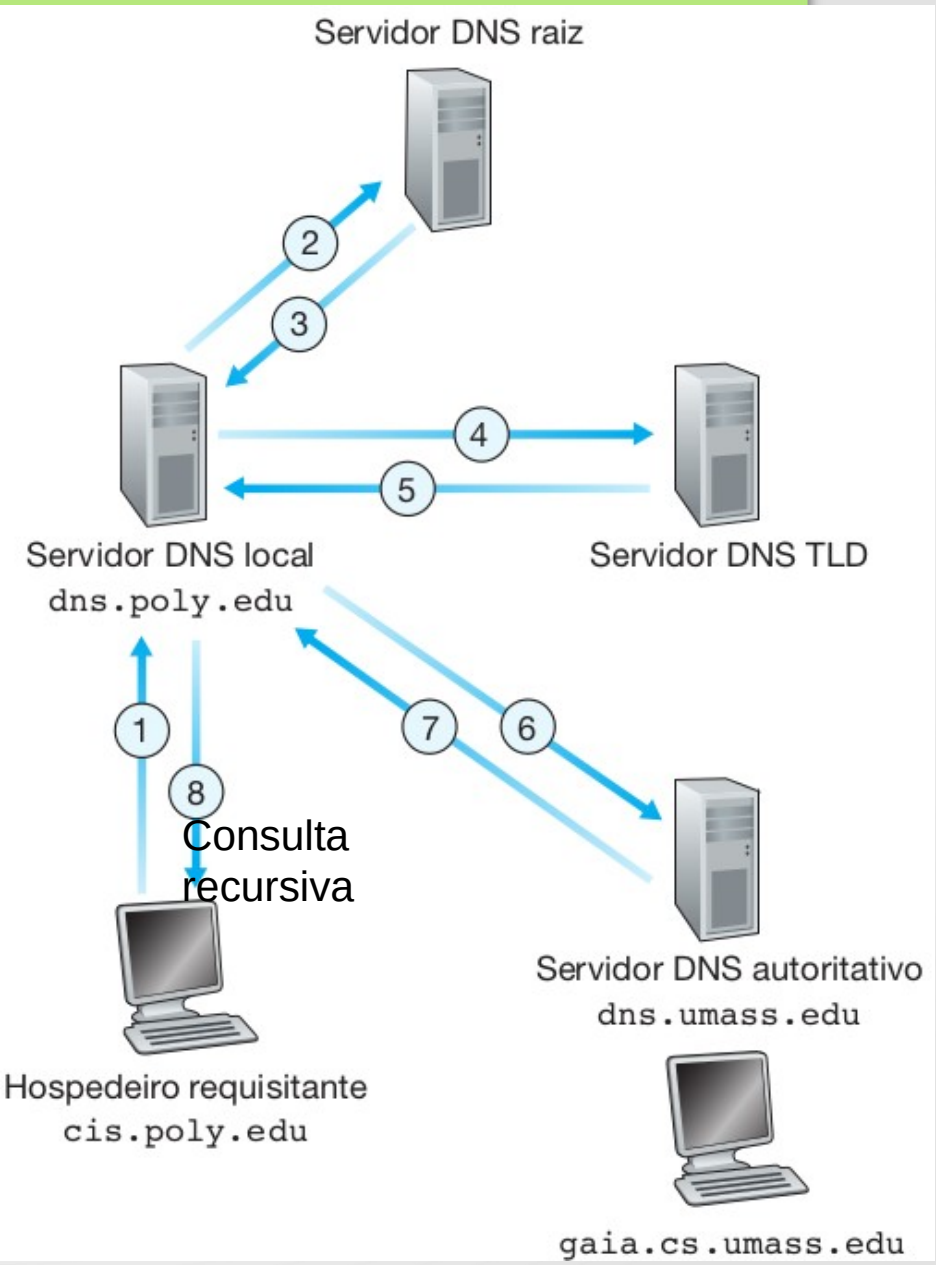
DNS: exemplo simples

Host **cis.poly.edu** quer o end IP de **gaia.cs.mit.edu**

1. host contata o DNS local: **dns.poly.edu**
2. **dns.poly.edu** contata o server raiz
3. server raiz responde com end. do server TLD p/ "edu".
4. **dns.poly.edu** reenvia a requisição ao server TLD.
5. o server TLD responde com o end. do server autoritativo de umass.edu.
6. o server **dns.poly.edu** envia a requisição para **dns.umass.edu**.
7. o server autoritativo **dns.umass.edu responde** responde com o end. de **gaia.cs.mit.edu.**
8. o server local **dns.poly.edu** envia o endereço de **gaia.cs.mit.edu** ao host **cis.poly.edu**.

Consulta 1 é recursiva (dns.poly.edu) envia as demais em nome do host

Consulta 2, 4 e 6 são iterativas, pois são retornadas diretamente ao requisitante.



DNS: armazenando e atualizando registros

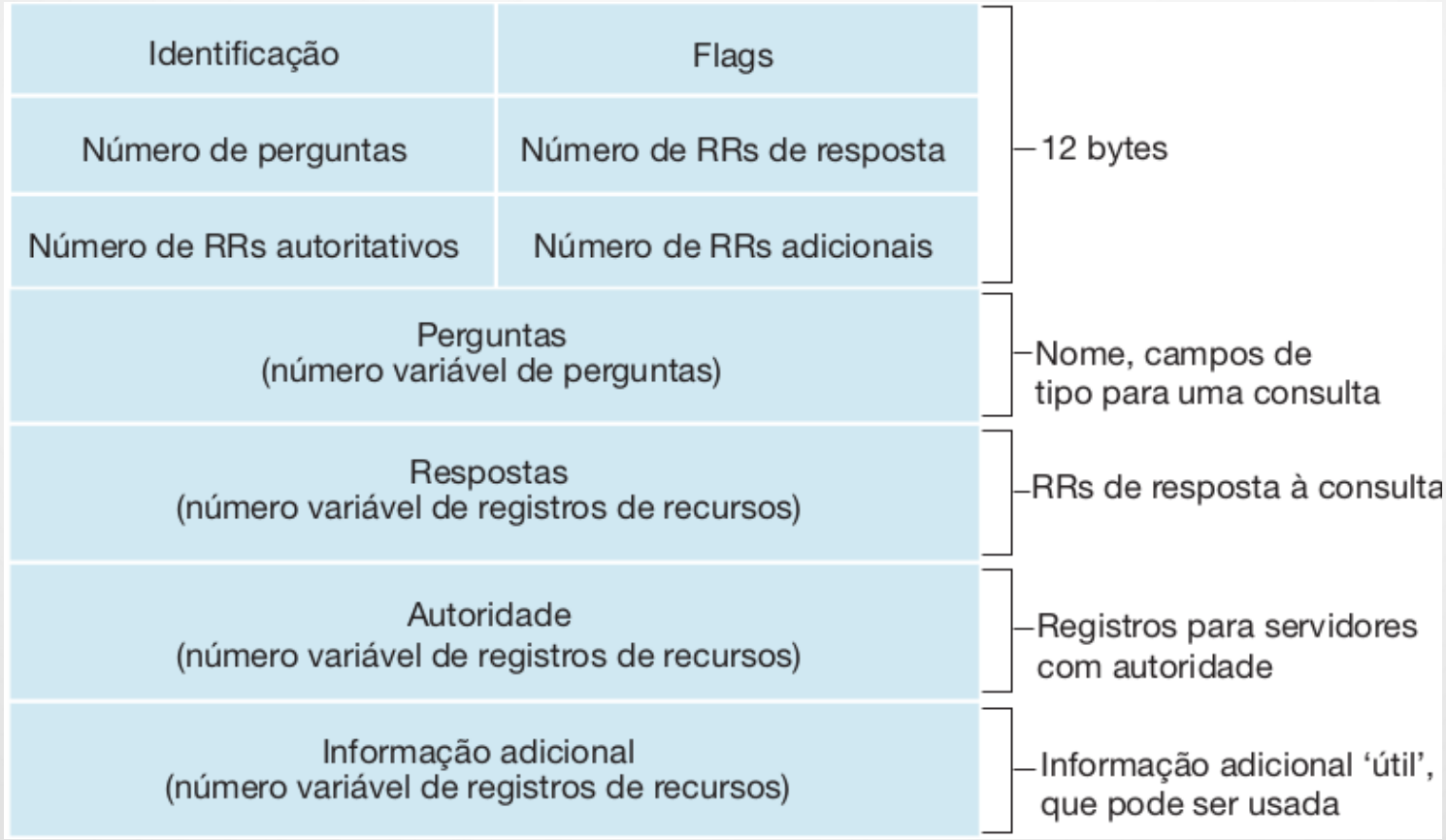
- uma vez que um servidor de nomes aprende um mapeamento, ele armazena o mapeamento num registro do tipo *cache,* assim as próximas requisições são respondidas mais rapidamente.
  - registro do cache tornam-se obsoletos (desaparecem) depois de um certo tempo, pois os endereços podem mudar.
- mecanismos de atualização e notificação foram projetados pelo IETF
  - Ex: RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

Registros do DNS

- DNS armazena registros de recursos (RR), no formato: RR (name, value, type,ttl).
- TTL é o tempo de vida útil, enquanto name e value dependem do type:
  - Type A - aponta um host (computador ou servidor), **name** é o nome do computador, **value** é o endereço IP
  - Type NS - aponta um servidor DNS autoritativo, **name** é um domínio (ex:foo.com) **value** é o endereço IP do servidor de nomes autoritativo para este domínio
  - Type CNAME - aponta um nome adicional para um nome real, **name** é um “apelido” para algum nome “canônico” (o nome real).
    Ex: foo.com, relay1.bar.foo.com, CNAME
  - Type MX ->aponta um mail server, **value** é o nome do servidor de correio associado com **name.** Ex: foo.com, mail.bar.foo.com,MX.

DNS: mensagens

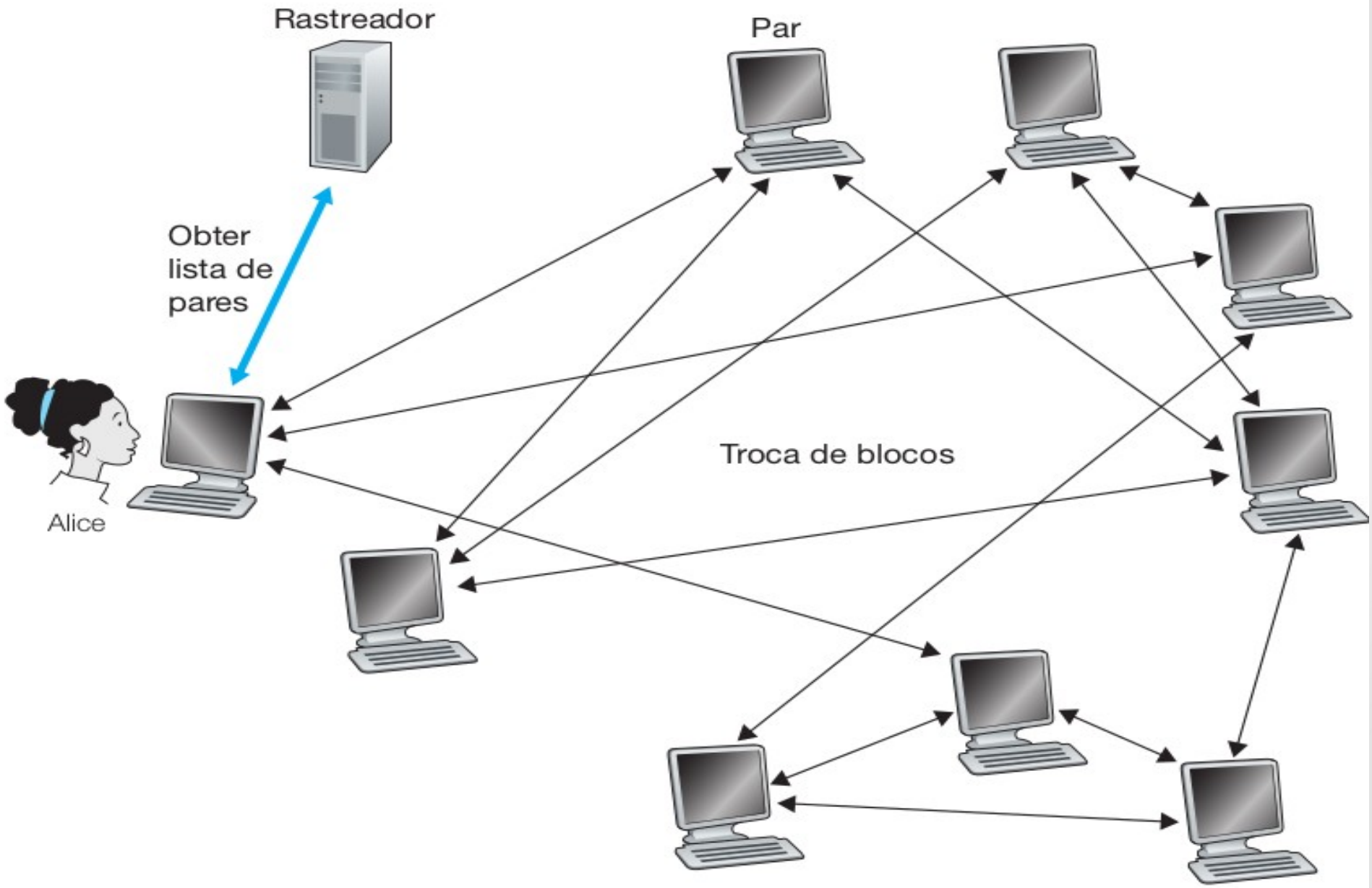
Mensagem de *consulta* e *resposta* , ambas com o mesmo *formato de mensagem*



Aplicações P2P: Bittorrent

- O Bittorrent divide um arquivo em blocos, geralmente de 256 KB. O conjunto de todos os blocos é o torrent.
- Quando um novo par entra, ele recebe de um rastreador (tracker) uma lista de IPs de outros pares. A quantidade de pares é definida nas configurações da aplicação.
- O novo par irá então tentar estabelecer conexões TCP com os pares vizinhos para que possa receber alguns pedaços. Conforme acumula pedaços, o par irá também fazer upload de blocos para outros pares.
- Quando o par tem arquivo inteiro, ele pode (de forma egoísta) sair ou (de forma altruísta) permanecer.

DISTRIBUIÇÃO DE ARQUIVOS COM O BITTORRENT



Aplicações P2P: Bittorrent

- Periodicamente, um par pede a cada um de seus pares vizinhos a lista de quais blocos eles têm.
- De posse das listas de todos os vizinhos, o par requisita os blocos que ele não tem aos vizinhos que possuem os blocos faltantes.
- Como prioridade são solicitados os blocos mais raros primeiro, ou seja, os que têm o menor número de cópias, pois são os mais propensos a ficarem indisponíveis caso os detentores se desconectem da aplicação.
- Desse modo a aplicação procura equalizar os números de cópias de cada bloco no torrent.

Aplicações P2P: Bittorrent

- Como um par recebe muitos pedidos, há um algoritmo de troca inteligente, que dá prioridade aos pares vizinhos que fornecem com a maior taxa.
- Continuamente, o par mede a taxa em que recebe bits e determina os 4 pares que fornecem na taxa mais alta. Então, ele prioritariamente envia blocos a esses 4 pares.
- A cada 30s um 5º par aleatório é adicionado, para que possa também receber blocos para trocar. Caso haja reciprocidade, ele pode substituir um dos 4 melhores.
- Assim, se juntam pares com taxas semelhantes, ou seja, melhores uploaders irão obter melhores taxas de download.

Camada de Aplicação: Sumário

Nosso estudo das aplicações está agora completo!

- exigências dos serviços de aplicação:
  - Confiabilidade (tolerância a perda), vazão (banda passante), temporização
- paradigma cliente-servidor
- paradigma peer-to-per
- modelo do serviço de transporte da Internet
  - orientado à conexão, confiável: TCP
  - não confiável, datagramas: UDP

- protocolos específicos:
  - http
  - ftp
  - smtp, imap, pop3
  - dns
  - bittorrent