PTC 3450 - Aula 07

2.2 Web e HTTP

2.3 Correio eletrônico na Internet

2.4 DNS – O serviço de diretório da Internet

(Kurose, p. 83 - 96)

(Peterson, p. 239-240 e 425 - 444)

28/03/2017

Capítulo 2: conteúdo

2.1 Princípios de aplicativos de rede

2.2 *Web* e HTTP

2.3 Correio eletrônico

- SMTP, POP3, IMAP

2.4 DNS

2.5 Aplicativos P2P

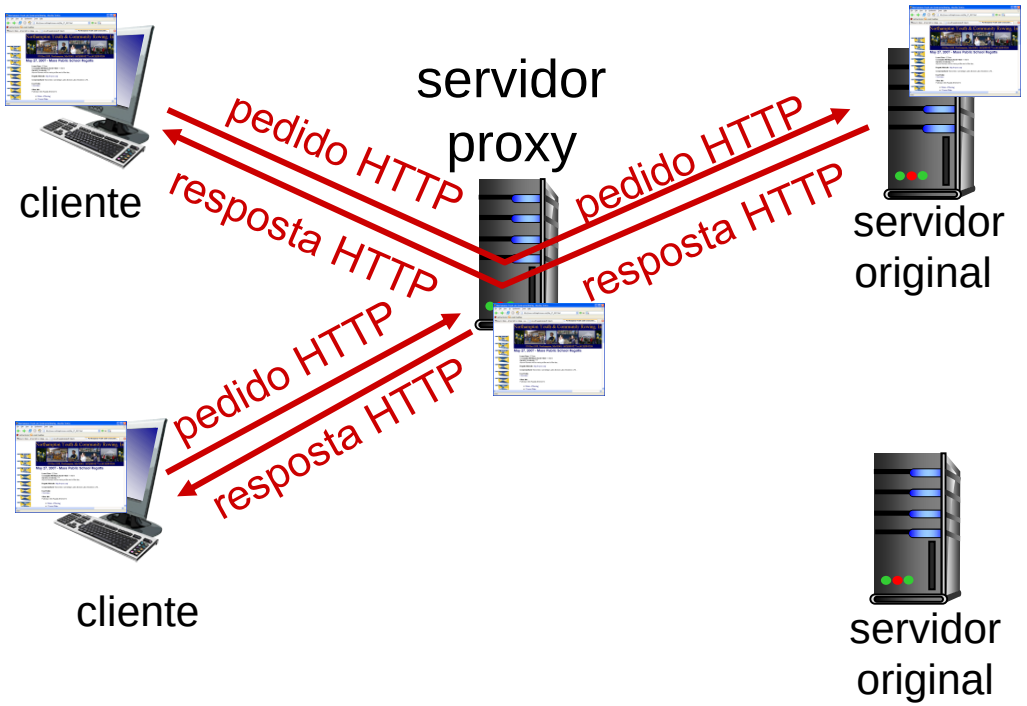
2.6 *Streaming* de vídeo e redes de distribuição de conteúdo

2.7 Programando *socket* com UDP e TCP

Caches *Web* (servidores *proxy*)

objetivo: atender pedido cliente sem envolver servidor original

- usuário configura navegador: acessos Web via *cache*
- navegador envia todos pedidos HTTP para *cache*
  - objetos no *cache*: *cache* retorna objeto
  - se não *cache* requer objetos do *servidor* original, então retorna objeto a cliente

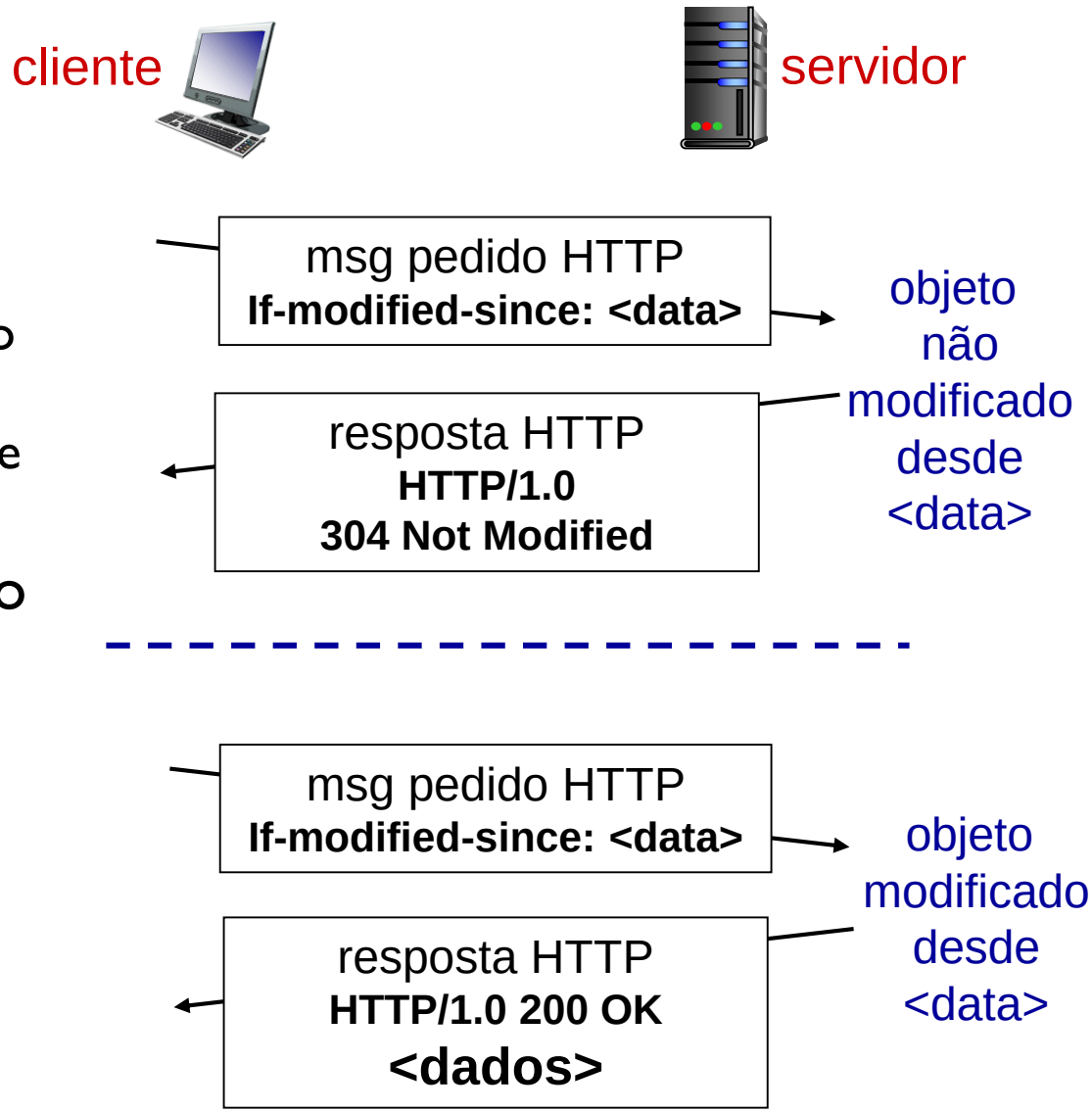


GET Conditional

- *Objetivo:* não enviar objeto se *cache* tem versão atual armazenada
  - sem atraso de transmissão de objeto
  - menor utilização do enlace
- *cache*: especifica data da cópia em *cache* no pedido HTTP

  `If-modified-since: <data>`
- *servidor*: resposta não contém objeto se cópia em cache é atual:

  `HTTP/1.0 304 Not Modified`



Capítulo 2: conteúdo

2.1 Princípios de aplicativos de rede

2.2 Web e HTTP

2.3 Correio eletrônico

- SMTP, POP3, IMAP

2.4 DNS

2.5 Aplicativos P2P

2.6 *Streaming* de vídeo e redes de distribuição de conteúdo

2.7 Programando *socket* com UDP e TCP

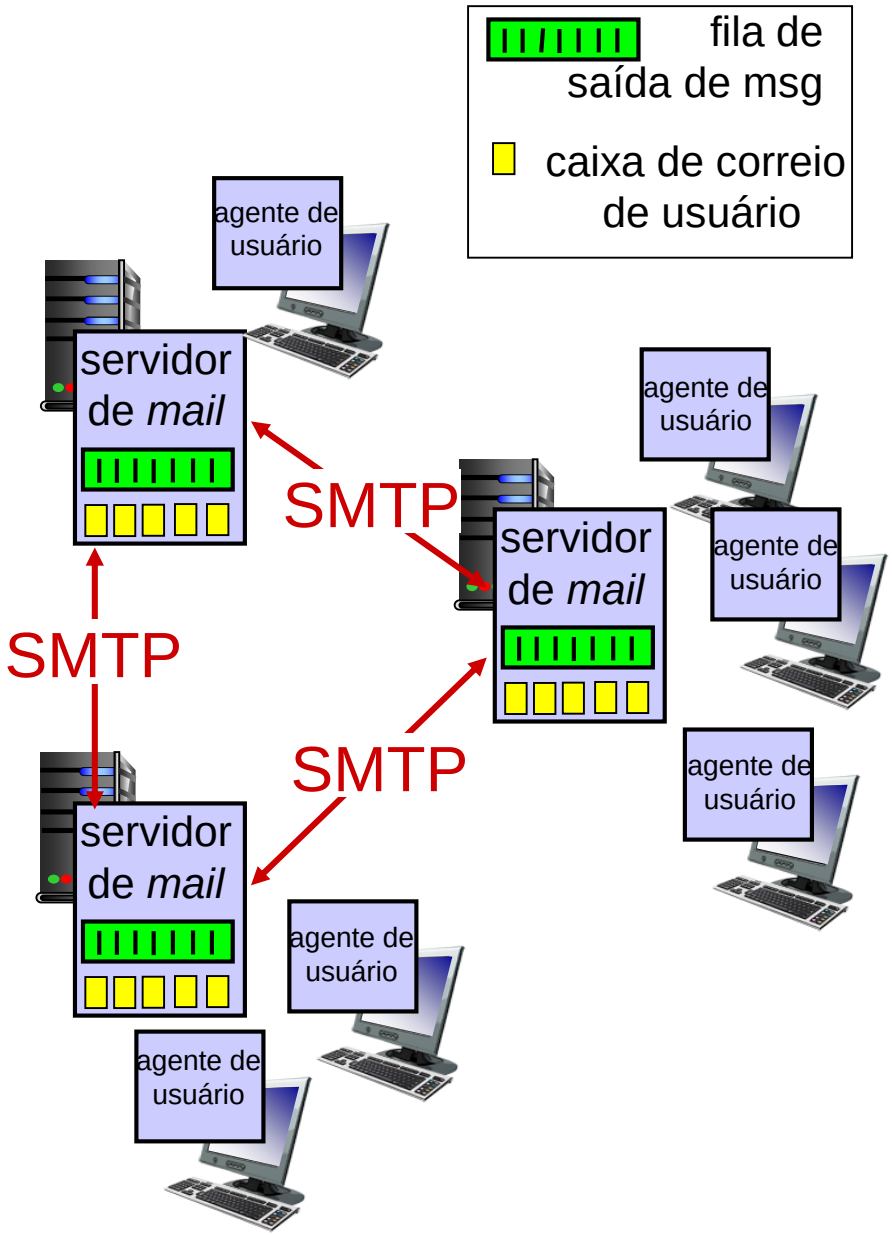
Correio Eletrônico

3 componentes principais:

- agentes de usuário
- servidores de *mail*
- *simple mail transfer protocol*: SMTP

Agente Usuário

- "leitor de *mail*"
- compor, editar, ler mensagens de *mail*
- e.g., Outlook, Thunderbird, iPhone *mail cliente, navegador web*
- envia e recebe mensagens armazenadas no servidor



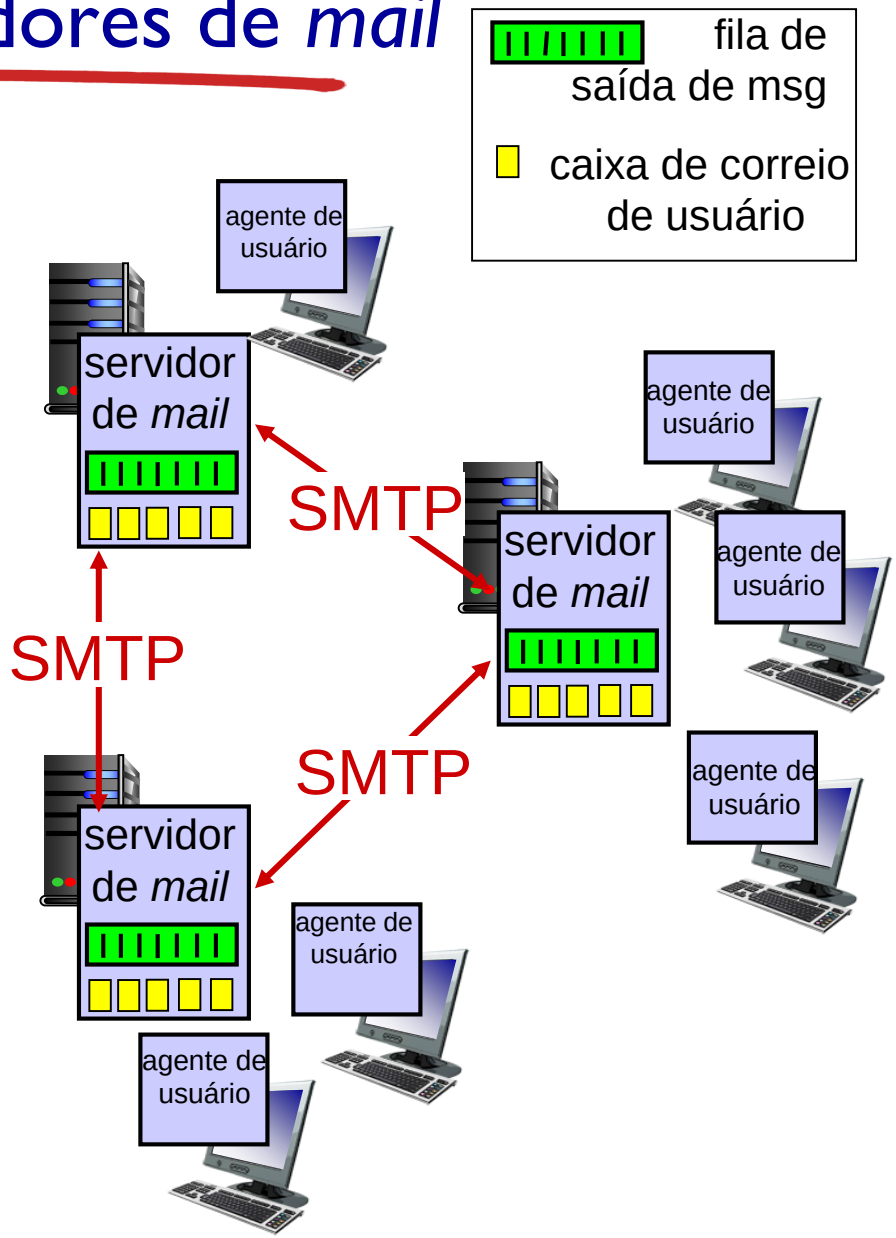
Correio eletrônico: servidores de *mail*

servidores de *mail*:

- *caixa de correio* contém mensagens que chegam para usuário
- *fila de mensagens* saindo (a serem enviadas)

Protocolo SMTP (1982)

- entre servidores de *mail* para enviar as mensagens de *email*
  - cliente: servidor de *mail* enviando
  - "servidor": servidor de *mail* recebendo

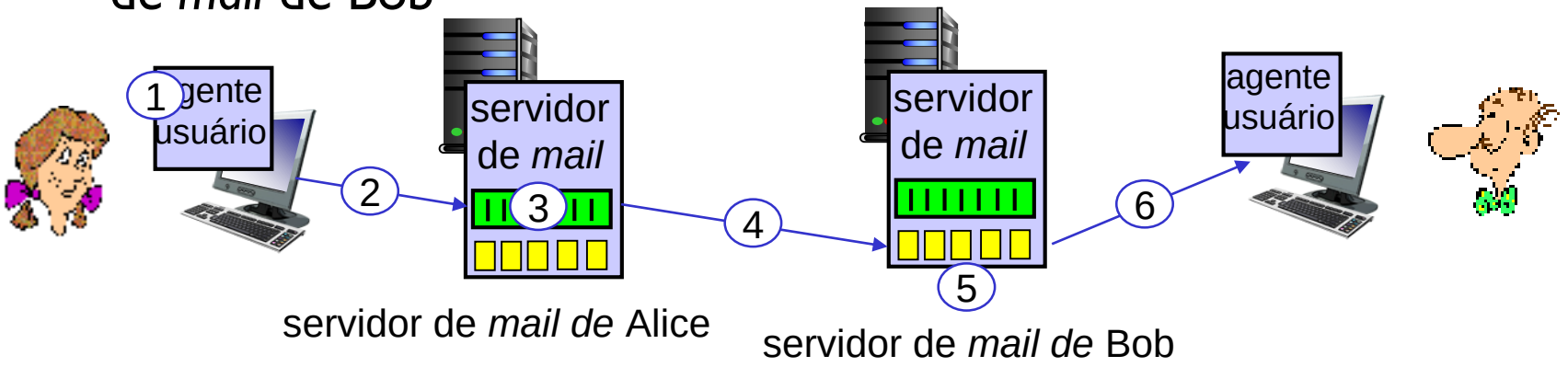


Correio Eletrônico: SMTP [RFC 5321 (2008)]

- usa TCP para transferir mensagens de *email* de forma confiável de cliente para servidor, porta 25
- transferência direta: servidor enviando para servidor recebendo – se destino estiver fora do ar espera e tenta novamente depois (30 min) (fila de mensagens)
- três fases de transferência
  - *handshaking* (saudações)
  - transferência de mensagens
  - encerramento
- interação comando/resposta (como HTTP)
  - comandos: texto ASCII
  - respostas: código e frase de estado
- mensagens (cabeçalhos e conteúdo) precisam estar em ASCII 7-bits (resquícios de sua antiguidade – 1982)

Cenário: Alice manda mensagem para Bob

1) Alice usa AU para compor mensagem "para" `bob@lcs.poli.usp.br`
2) AU de Alice manda mensagem para seu servidor de *mail*; mensagem colocada na fila de mensagens
3) lado cliente do SMTP abre conexão TCP com servidor de *mail* de Bob
4) cliente SMTP envia mensagem de Alice sobre a conexão TCP
5) servidor de *mail* de Bob coloca mensagem na caixa de entrada de Bob
6) Bob usa seu agente de usuário para ler mensagem



Exemplo de interação SMTP

```
S: 220 smtp.uol.com.br
C: HELO whitehouse.gov
S: 250  Hello whitehouse.gov, pleased to meet you
C: MAIL FROM:<obama@whitehouse.gov>
S: 250 obama@whitehouse.gov... Sender ok
C: RCPT TO:<bobinho@uol.com.br>
S: 250 bobinho@uol.com.br ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Dear Ze Bobinho
C: Here is Pres. Obama. How are you?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 uol.com.br closing connection
```

Experimentando interação SMTP

- **`telnet nomedoservidor 25`** (ou 587 – TLS)
- Exemplo `smtp.lcs.poli.usp.br`
- veja resposta 220 do servidor
- entre comandos HELO, MAIL FROM, RCPT TO, DATA, QUIT

assim você pode mandar *email* sem usar um cliente de *email* (leitor)

SMTP: palavras finais

- SMTP usa conexões persistentes
- SMTP requer mensagem (cabeçalho e corpo) em 7-bit ASCII
- servidor SMTP usa `CRLF.CRLF` para determinar fim da mensagem

comparação com HTTP:

- HTTP: puxa (*pull*)
- SMTP: empurra (*push*)
- ambos têm interação comando/resposta em ASCII
- HTTP: cada objeto encapsulado em sua própria msg resposta
- SMTP: múltiplos objetos enviados em mensagem

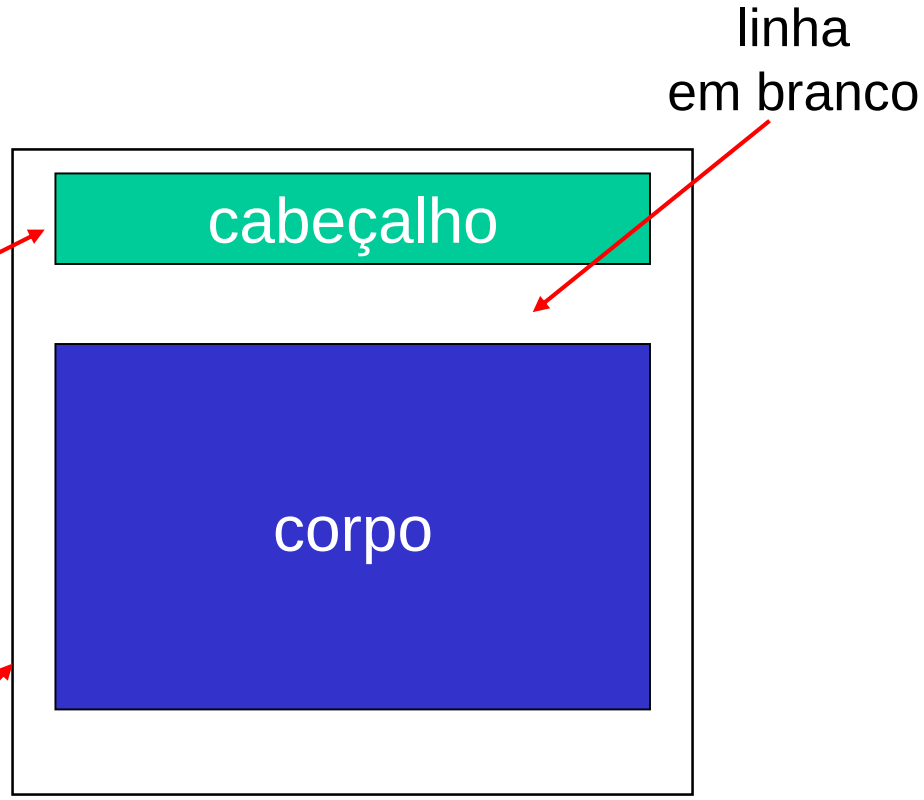
Formato da mensagem de *mail*

SMTP: protocolo para troca de mensagens *email*

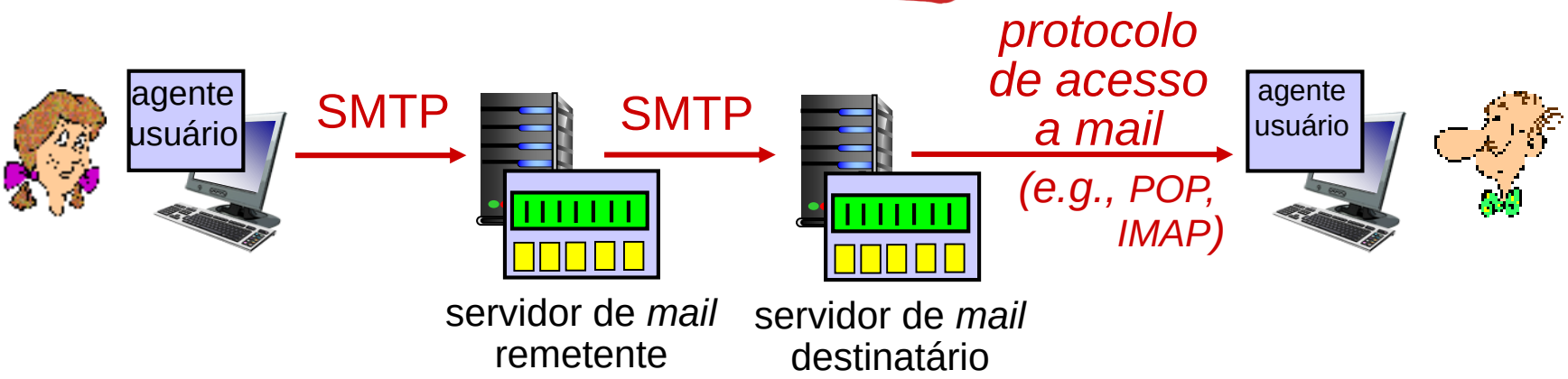
RFC 5322: padrão para formato de mensagens de texto:

- linhas cabeçalho, e.g.,
  - To:
  - From:
  - Subject:

  diferentes dos comandos SMTP MAIL FROM, RCPT TO:!
- Corpo: a "mensagem"
  - apenas caracteres ASCII



Protocolos de acesso a *mail*



- SMTP: entrega/armazenagem no servidor do destinatário
- protocolo de acesso a *mail*: recuperação do servidor
  - POP: *Post Office Protocol* [RFC 1939]: autorização, *download* (porta 110) *(1988)*
  - IMAP: *Internet Mail Access Protocol* [RFC 3501]: mais ferramentas, incluindo manipulação de mensagens armazenadas no servidor *(1994)*
  - HTTP: gmail, Hotmail (1° público e gratuito - 1996), Yahoo! Mail, etc.

Protocolo POP3

- Utilizar telnet porta 110 (e.g. `pop3.uol.com.br`)

fase de autorização

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

fase de negociação, cliente:

- **`list`**: list números das mensagens
- **`retr`**: recupera mensagem por número
- **`dele`**: apaga
- **`quit`**

```
S: +OK POP3 server ready
C: USER marcio
S: +OK
C: PASS hahaha
S: +OK usuário entrou com sucesso
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <conteúdo da mensagem 1>
S: .
C: dele 1
C: retr 2
S: <conteúdo da mensagem 2>
S: .
C: dele 2
C: quit
S: +OK servidor POP3 terminando
                    transmissão
```

POP3 (mais) e IMAP (RFC 3501)

mais sobre POP3

- exemplo anterior usa POP3 no modo "*download* e apaga"
  - Marcio não pode reler suas mensagens se mudar de cliente
- POP3 "*download* e *mantém*": cópia das mensagens em clientes diferentes
- POP3 é sem memória entre sessões

IMAP

- Mais complexo – feito para usuário manter todas as mensagens em único lugar: no servidor
- permite ao usuário organizar suas mensagens em pastas
- mantém estado do usuário entre sessões:
  - nomes de pastas e mapeamento entre IDs de mensagem e nome de pasta

Capítulo 2: conteúdo

2.1 Princípios de aplicativos de rede
- arquitetura de *apps*
- requsitos para *apps*

2.2 *Web* e HTTP

2.3 FTP

2.4 Correio eletrônico
- SMTP, POP3, IMAP

2.5 DNS

2.6 Aplicativos P2P

2.7 Programando *socket* com UDP e TCP

DNS: *Domain Name System* (RFC 1034 e 1035)

pessoas: vários identificadores:

- CPF, nome, num. do passporte
- Cada um mais relevante em cada situação

hosts e roteadores Internet:

- *Endereço IP* (32 bit – 4 bytes) – usado para endereçar datagramas
- *Hostname* - "nome", e.g., www.yahoo.com – usado por humanos

Q: como mapear entre endereço IP e nome, e vice versa?

Domain Name System (1983):

- *banco de dados distribuído* implementado de forma hierárquica em muitos *name servers* (ou servidores DNS)
- *protocolo da camada de aplicação:* *hosts* e *name servers* comunicam-se para *decifrar* nomes (tradução endereço IP/nome)
  - nota: função de núcleo da Internet, implementada como protocolo da camada de aplicação
  - complexidade na "borda" da rede