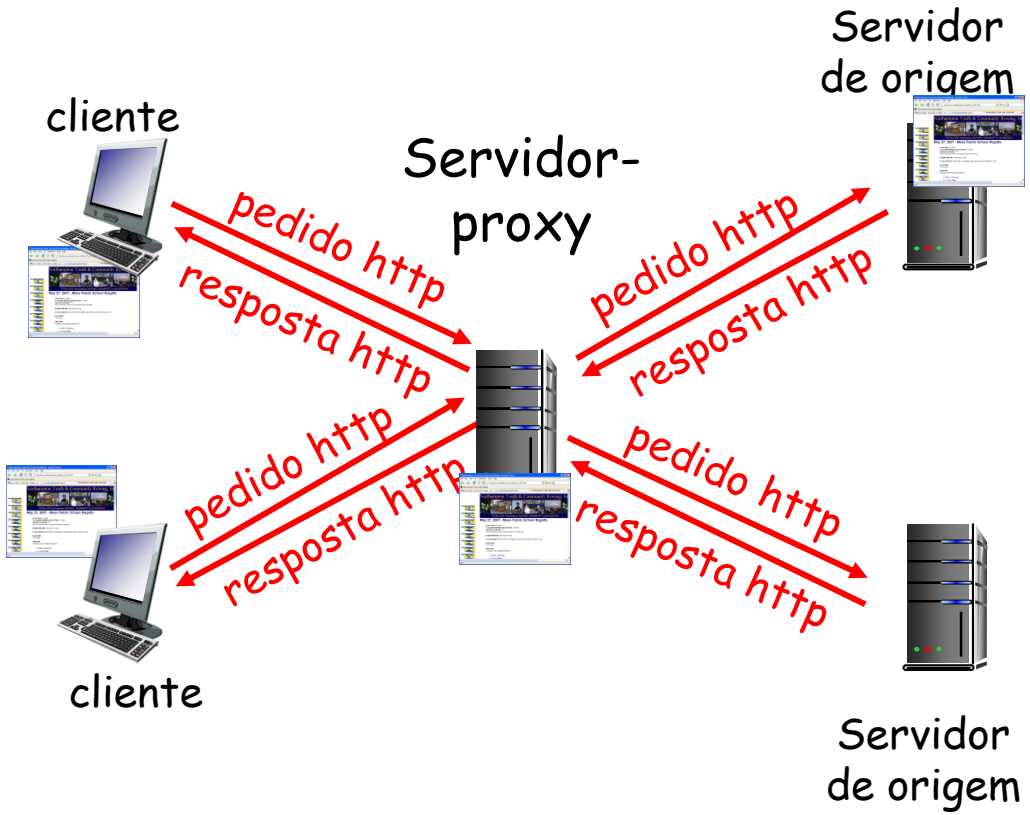
Web Cache (servidor proxy)

Meta: atender pedido do cliente sem envolver servidor de origem

- usuário configura browser: acessos Web via proxy
- cliente envia todos pedidos http ao proxy
  - se objeto no cache do proxy, este o devolve imediatamente na resposta http
  - senão, solicita objeto ao servidor de origem, depois devolve resposta http ao cliente



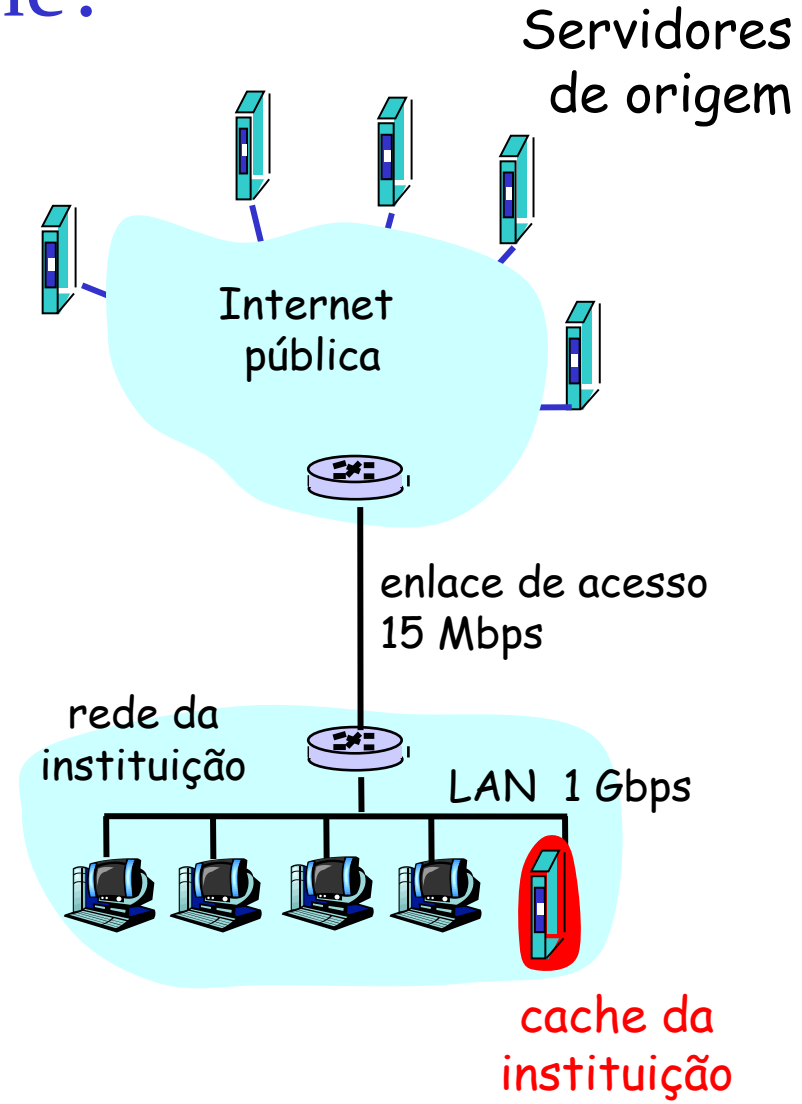
Por que usar Web cache?

Suposição: cache está "próximo" do cliente (p.ex., mesma rede local)

⇒ tempo de resposta menor

⇒ diminui tráfego aos servidores distantes

- muitas vezes o enlace que liga a rede da instituição ou do provedor à Internet é um gargalo



Web cache (mais)

- cache age tanto como cliente quanto como servidor
  - ✓ servidor para a requisição original do cliente
  - ✓ cliente quando faz pedido ao servidor de origem

- tipicamente, o Web cache é instalado pelo provedor (ISP da universidade, empresa ou residencial)

- possibilita a provedores de conteúdo que não têm muita banda distribuir seus conteúdos de forma mais eficiente

Cache: exemplo

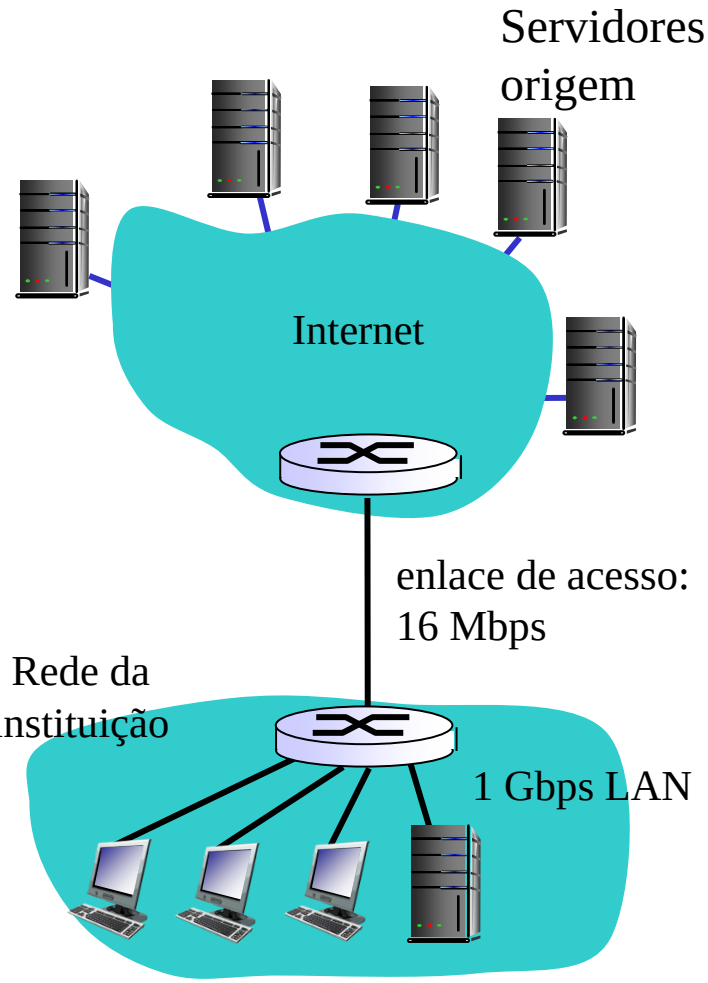
Hipóteses:

- Tamanho médio de um obj: 1 Mbit
- Taxa média de req dos browsers p/ servidor de origem: 15 req/seg
- Taxa média de dados p/ browser: 15 Mbps
- RTT do roteador da instit. p/ qq. servidor de origem: 2 seg
- Taxa do enlace de acesso: 16 Mbps

Consequências:

- ✓ Utilização da LAN: 15Mbps/1000 = 0.015
- ✓ Utilização no enlace de acesso: 15 Mbits/16 Mbps (~94%) *problema!*
- ✓ Atraso total = RTT + atraso de acesso à Internet + atraso da LAN
- ✓ = 2 seg + minutos + microssegs

Sem usar cache



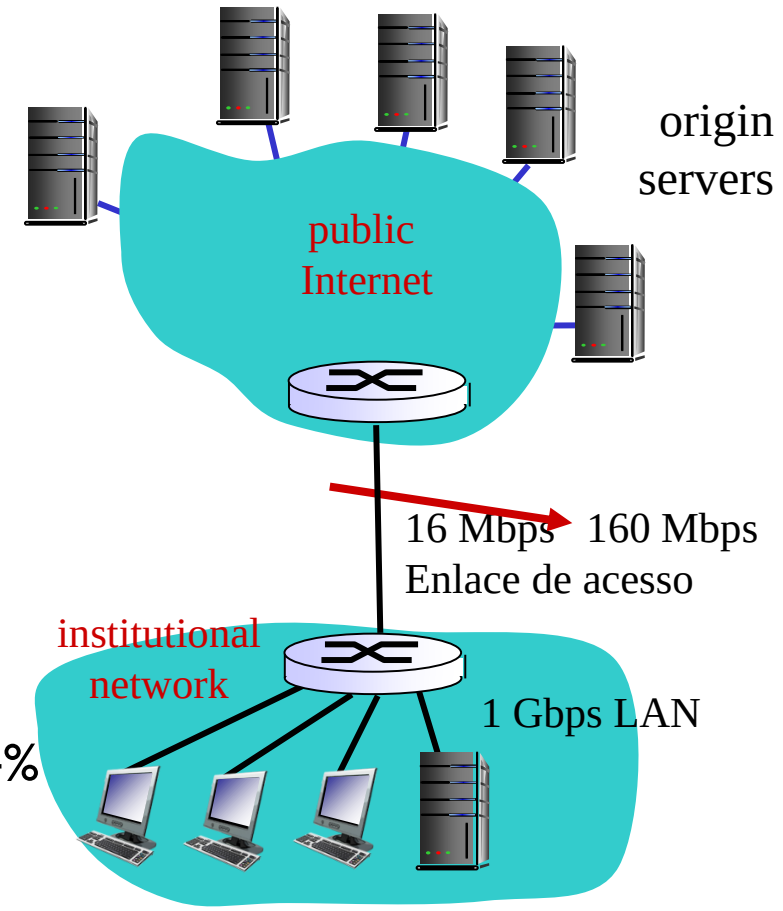
1ª solução: enlace mais rápido

hipóteses:

- Tamanho médio de um obj: 1 Mbit
- Taxa média de req dos browsers p/ servidor de origem: 15 req/seg
- Taxa média de dados p/ browser: 15 Mbps
- RTT do roteador da instit. p/ qq. servidor de origem: 2 seg
- Enlace de acesso: ~~16 Mbps~~ 160 Mbps

consequências:

- Utilização da LAN: 0.015
- Utilização do enlace de acesso = ~~94%~~ 9.4%
- Atraso total = atraso da Internet + atraso do acesso + LAN delay
  = 2 seg + ~~minutes~~ + µsegs msecs



Aumento da banda do enlace costuma ser uma solução cara!

Usando Cache

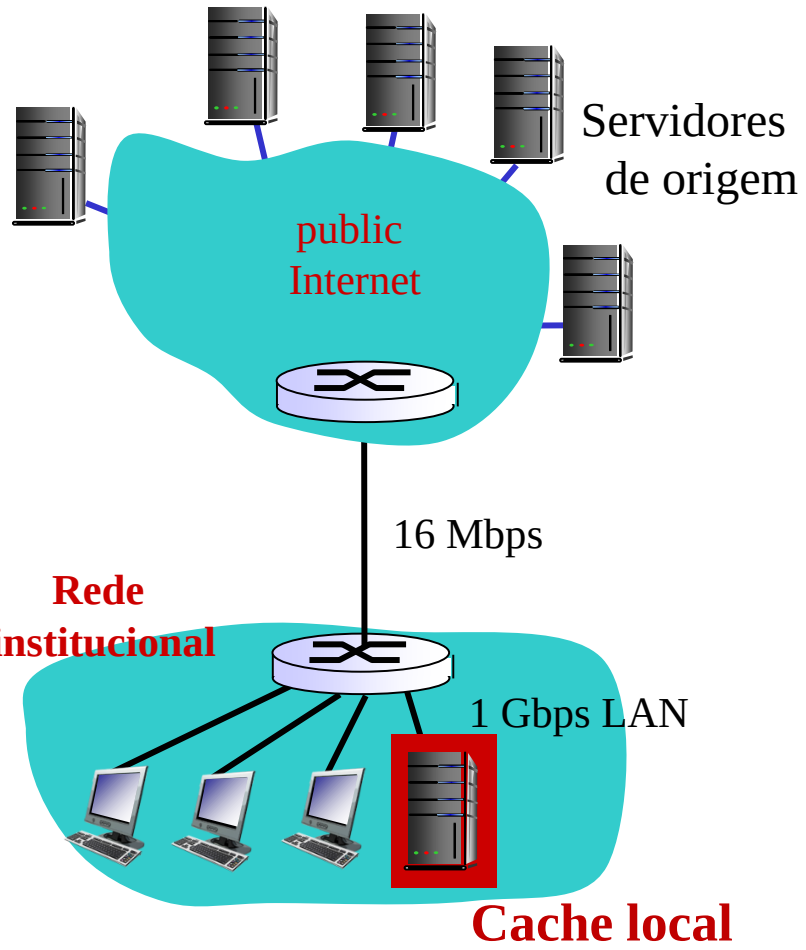
Mesmo cenário:

- Tam médio do obj: 1 Mbit
- Taxa média de reqs a partir dos browsers p/ serv orig: 15 req/seg
- Taxa media de dados p/ browser: 15 Mbps
- RTT do roteador institucional a qualquer servidor de origem: 2 seg
- Taxa do enlace de acesso : 16 Mbps

Desempenho?

Necessário calcular a utilização do enlace e o atraso

Custo: web cache (barato!)



Usando Cache

Calculando a utlização do enlace de acesso e o atraso com cache:

Suponha que o cache hit é 0.4

- 40% requisições são satisfeitas pelo cache, 60% req. satisfeitas no servidor de origem

❖ Utilização do enlace de acesso:

- 60% das req no enlace de acesso

❖ Taxa de download sobre o enlace de acesso = 0.6*15 Mbps = 9.0 Mbps

- utilização = 9.0/16 = 0,58

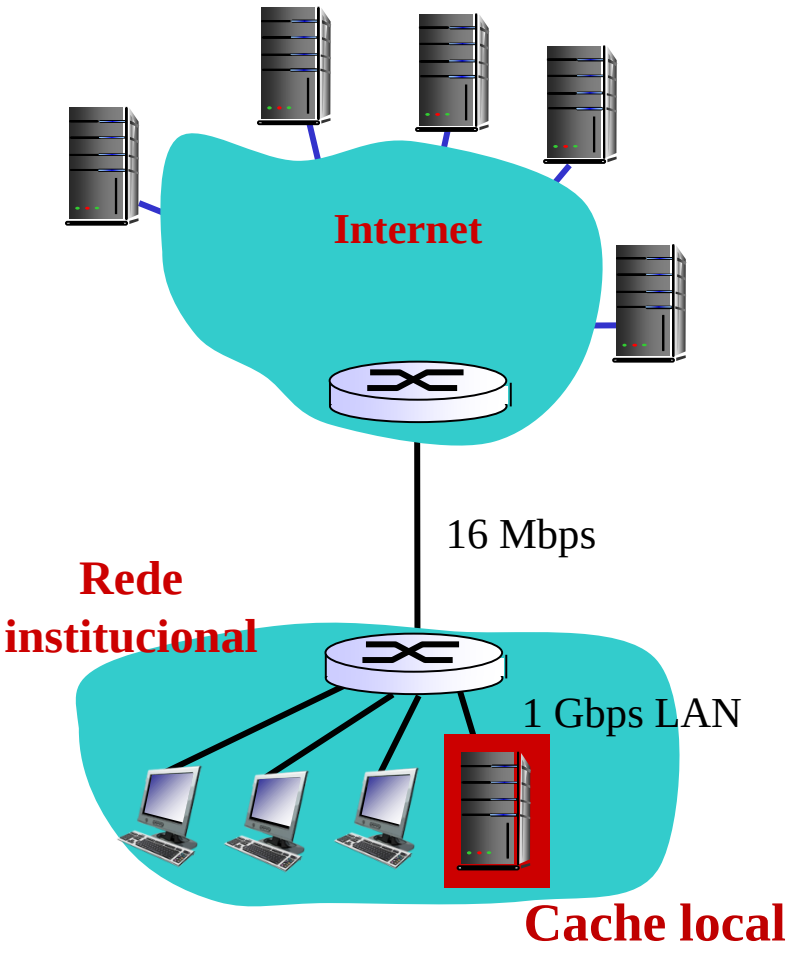
❖ *Atraso total*

= 0.6 * (atraso ao serv origem) +
0.4 * (atraso qdo cache satisfeito)

= 0.6 (2.01) + 0.4 (~msecs)

= ~ 1.2 segs (em media!)

Menor do que com enlace de acesso 160 Mbps (e mais barato tb!)

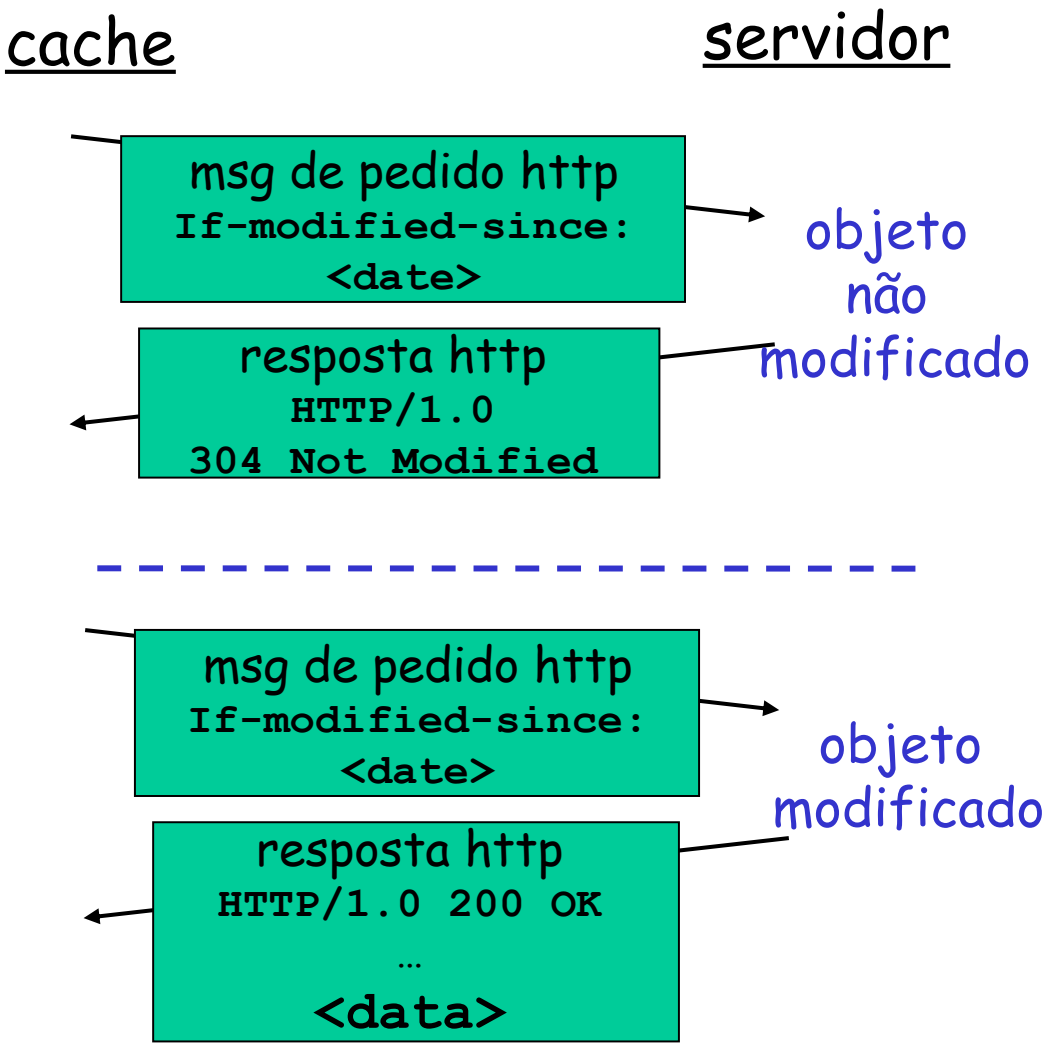


GET condicional

- Meta: não enviar objeto se cliente já tem (no cache) a versão atual
  - Sem atraso na TX do objeto
  - menor utilização do enlace
- cache: especifica data da cópia no cache no pedido http

  `If-modified-since: <date>`
- servidor: resposta não contém objeto se cópia no cache é atual:

  `HTTP/1.0 304 Not Modified`



HTTP/2 (RFC 7540)

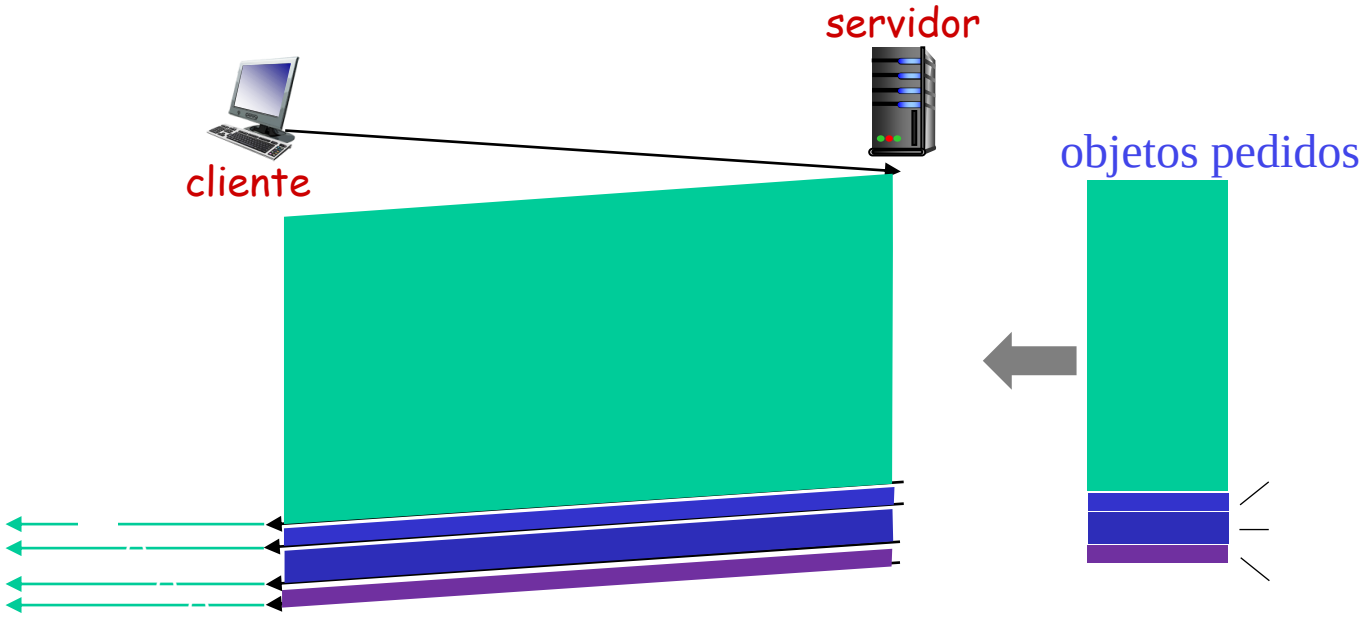
Objetivo principal:

Diminuir a latência do carregamento de páginas com objetos de tipos e tamanhos diferentes

- Consertar o problema de bloqueio do cabeça da fila do HTTP 1.1 (HOL – FCFS – objetos pequenos são prejudicados)
  - No http 1.1: abre conexões TCP paralelas
    - Problema: navegador se aproveita do controle de congestionamento do TCP para obter mais banda “trapaceando”
- Carregamento de elementos da página em paralelo através de uma única conexão TCP, reduzindo o número de conexões paralelas do HTPP 1.1
- Como resolver o HOL com apenas uma conexão TCP?
  - Multiplexa (Intercala) os quadros de resposta do servidor
  - Subcamada de enquadramento do protocolo http 2 é usada tanto para gerar os quadros no servidor quanto para remontá-los no cliente

HTTP/2: reduzindo a latência

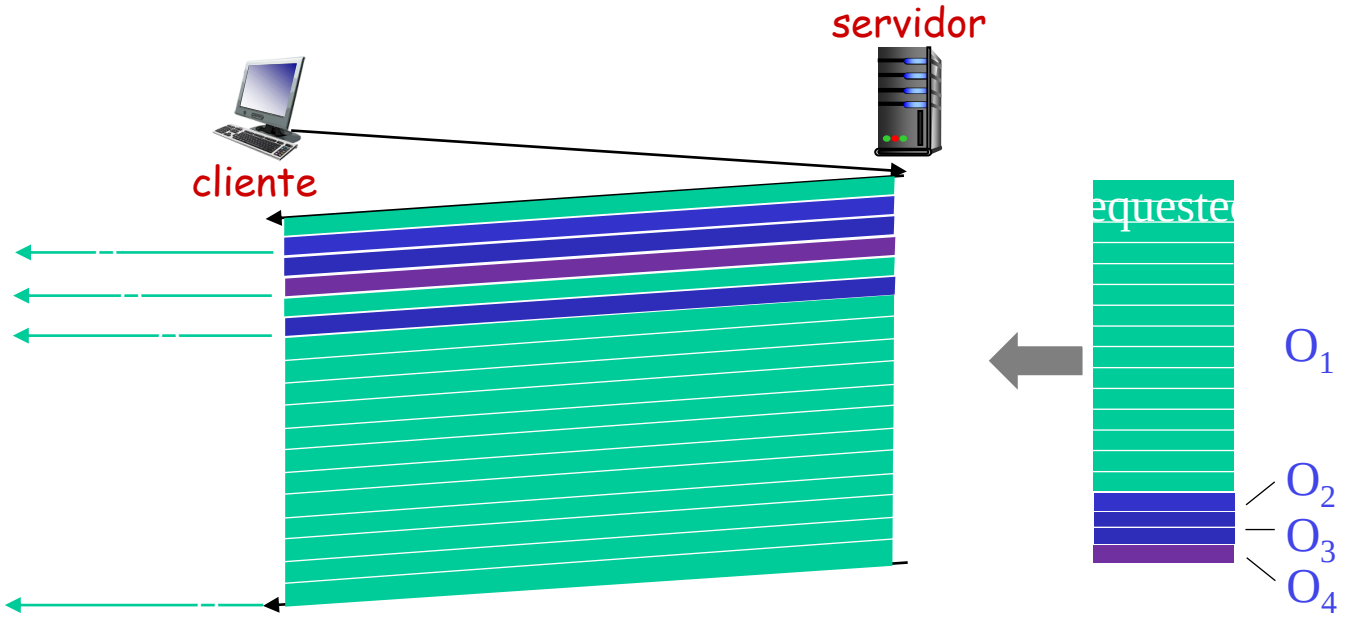
HTTP 1.1: cliente pede 1 objeto grande (por ex., arquivo de video) e 3 objetos menores)



Objetos entregues na ordem solicitada: O_2, O_3, O_4 esperam a vez atrás do objeto O_1

HTTP/2: reduzindo a latência

HTTP/2: objetos são divididos em quadros e a transmissão destes é realizada de forma intercalada



Objetos O_2, O_3, O_4 são entregues rapidamente, enquanto O_1 é ligeiramente atrasado

HTTP/2

Outros objetivos:

- permitir a priorização de solicitações

  cliente especifica um peso para cada mensagem enviada ao servidor, que utiliza essa prioridade para enviar primeiro os quadros de respostas a solicitações que tenham maior prioridade

- ***push***

  servidor pode enviar objetos adicionais, baseado na página ***html*** base, sem que o cliente tenha solicitado, eliminando uma latência adicional de espera pelas solicitações

- oferecer compressão mais eficiente dos campos de cabeçalho

HTTP/2

Outras características e diferenças para o http 1.1

- Mecanismos de negociação para permitir a clientes e servidores escolher o HTTP 1.1, HTTP/2 ou outros protocolos

- Manutenção de compatibilidade de alto nível como HTTP 1.1

- Mantém a maior parte da sintaxe de alto nível do HTTP 1.1 tais como: métodos, códigos de status, campos de cabeçalhos e URLs

HTTP/2 p/ HTTP/3

Objetivo principal

diminuir o atraso em requisições com múltiplos objetos HTTP

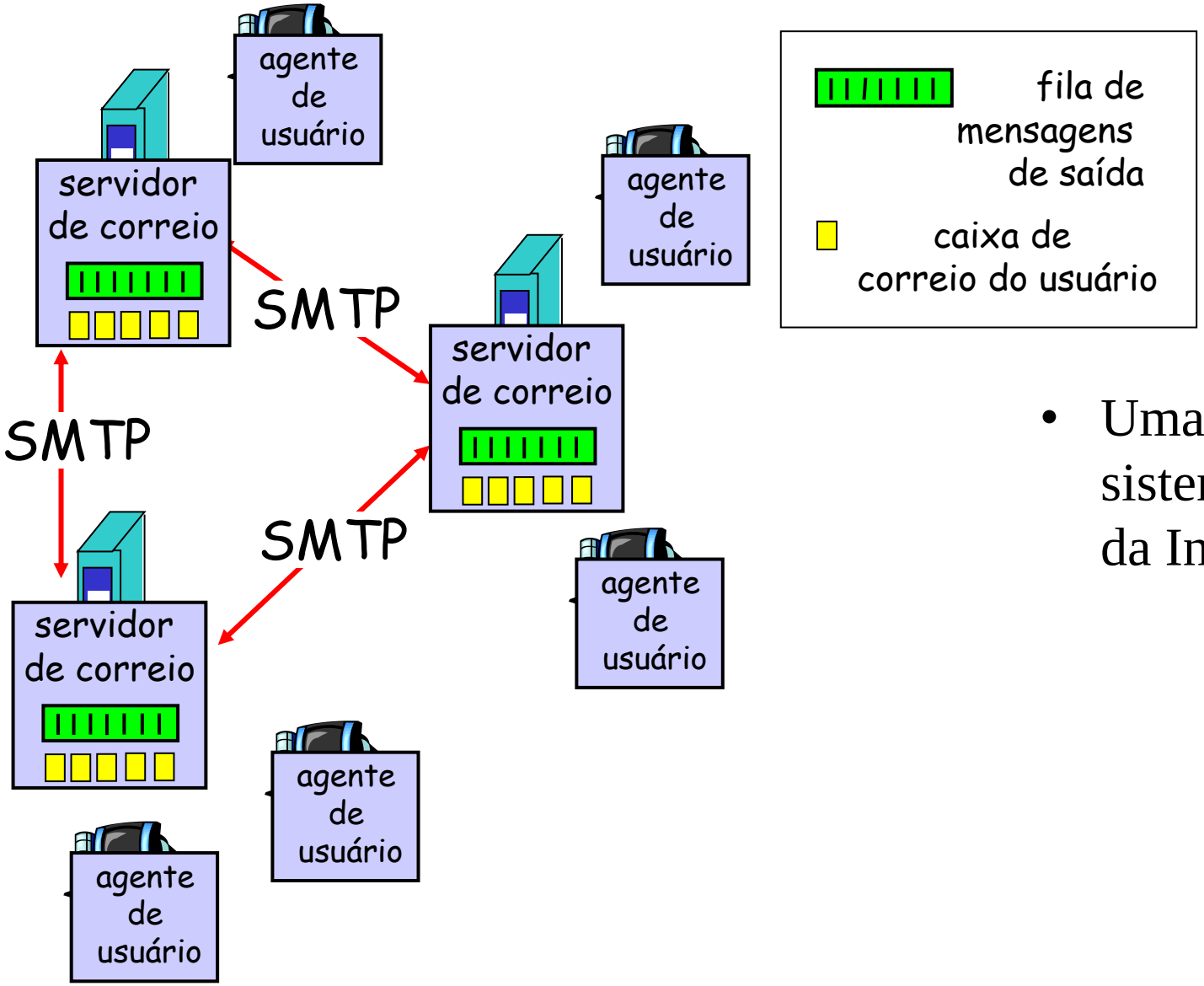
HTTP/2 sobre uma conexão TCP única significa:

- recuperação de perda de pacotes ainda paralisa todas as transmissões de objetos
  - assim como no HTTP 1.1, os navegadores são incentivados a abrir múltiplas conexões TCP para reduzir este problema e aumentar a vazão
- não fornece segurança sobre as conexões TCP

HTTP/3:

- **Adiciona segurança**
- **Controle de congestionamento sobre UDP**
- **Mais paralelismo**

Correio eletrônico na Internet: e-mail



- Uma visão do sistema de e-mail da Internet.

Correio Eletrônico

Três grandes componentes:

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

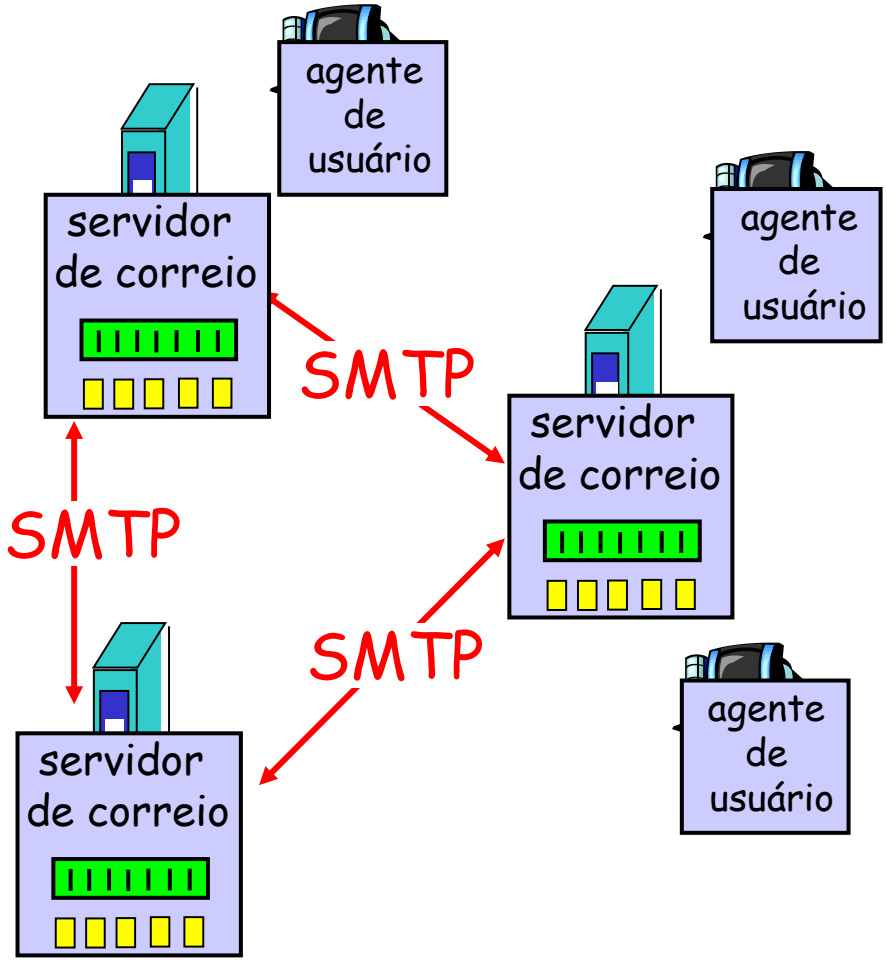
Agente de Usuário

- "leitor de correio"
- compor, editar, ler mensagens de correio
- ex: Outlook, cliente de e-mail p/ iPhone
- mensagens de saída e chegando são armazenadas no servidor

Servidores de correio

Servidores de correio:

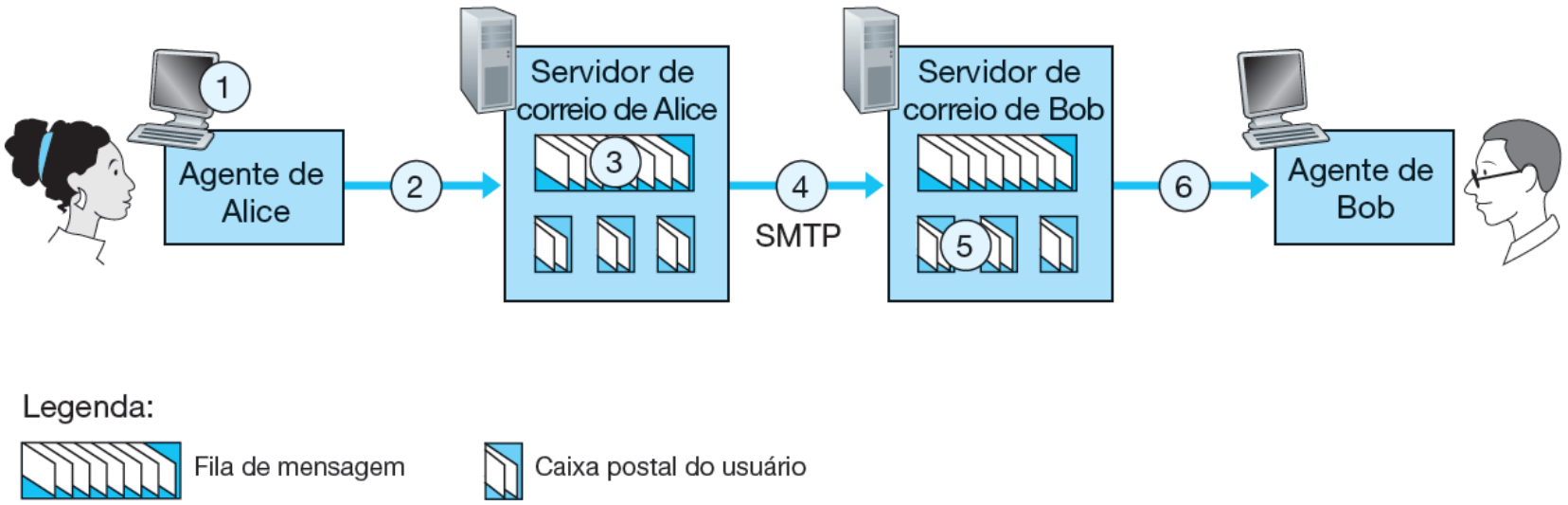
- caixa de correio contém mensagens de chegada (ainda não lidas) p/ usuário
- fila de mensagens contém mensagens de saída (a serem enviadas)
- Protocolo SMTP: transfere msgs entre servidores de correio remetentes e destinatários
  - cliente: servidor de correio que envia
  - "servidor": servidor de correio que recebe



Protocolo SMTP (RFC 5321)

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

Alice envia uma mensagem a Bob:

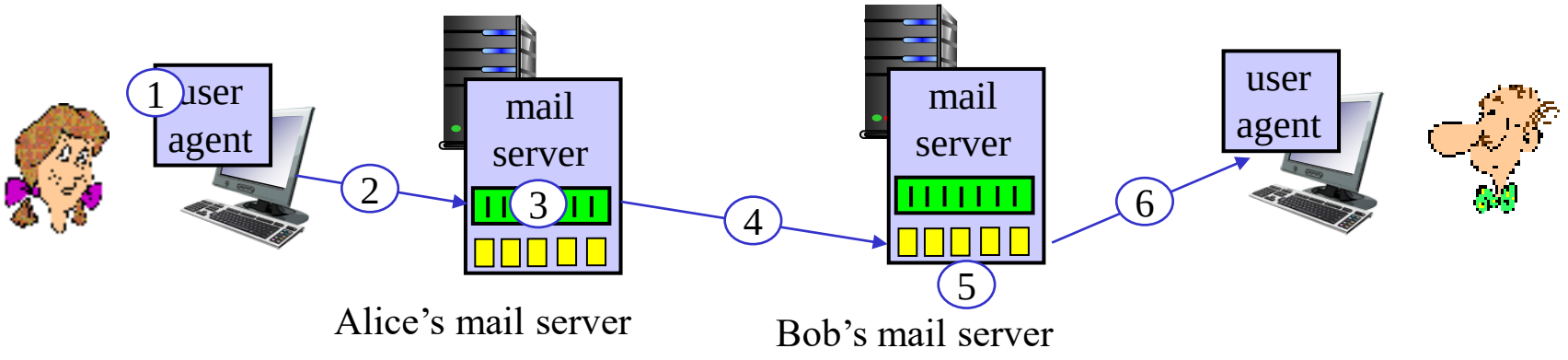


Protocolo SMTP (RFC 5321)

- usa o TCP para a transferência confiável de msgs do correio do cliente ao servidor. Porta 25.
- transferência direta: servidor remetente ao servidor receptor
- três fases da transferência
  - *handshaking* (procedimentos iniciais de saudação)
  - transferência das mensagens
  - encerramento
- interação comando/resposta (assim como o http)
  - comandos: texto ASCII
  - resposta: código de status e frase
- Mensagens em ASCII – 7 bits

Cenário: Alice envia msg para Bob

1) Alice usa um agente usuário (UA) para compor mensagem para `bob@ufrrj.br`
2) UA de Alice envia mensagem para seu servidor de correio; mensagem é colocada na fila de mensagens
3) lado cliente do SMTP abre uma conexão TCP com o servidor de correio de Bob
4) "cliente" SMTP envia a msg de Alice sobre a conexão TCP
5) Servidor de correio de Bob coloca a msg na caixa de correio de Bob
6) Bob chama seu agente usuário para ler a mensagem



Interação SMTP típica

```
  Telnet doces.br 25
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: Voce gosta de chocolate?
  C:   Que tal sorvete?
  C: .
  S: 250 Message accepted for delivery
  C: QUIT
  S: 221 doces.br closing connection
```

Experimente uma interação SMTP

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

Obs.: estes comandos permitem que você envie correio sem usar um agente usuário (leitor de correio). Somente irá funcionar se <nomedoservidor> permitir conexões para a porta 25, o que é raro, atualmente, por questões de segurança

SMTP: últimas palavras

Comparação com http

- http: pull (puxar)
- smtp: push (empurrar)
- ambos tem interação comando/resposta, códigos de status em ASCII
- http: cada objeto é encapsulado em sua própria mensagem de resposta
- smtp: múltiplos objetos enviados numa mensagem de múltiplas partes
- smtp usa conexões persistentes: várias msgs podem ser enviadas na mesma conexão TCP
- smtp requer que a mensagem (cabeçalho e corpo-texto) esteja em ASCII de 7-bits
- algumas cadeias de caracteres não são permitidas numa mensagem (p.ex., `CRLF.CRLF`).
  - servidor smtp usa `CRLF.CRLF` para reconhecer o final da mensagem

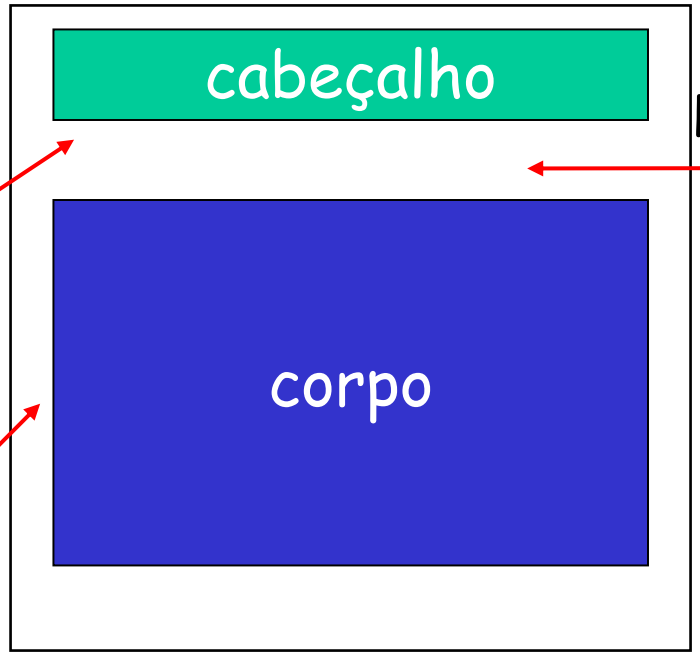
Formato de uma mensagem

SMTP: protocolo para trocar msgs de correio

RFC 822: define a sintaxe para formato de mensagem de texto (como html)

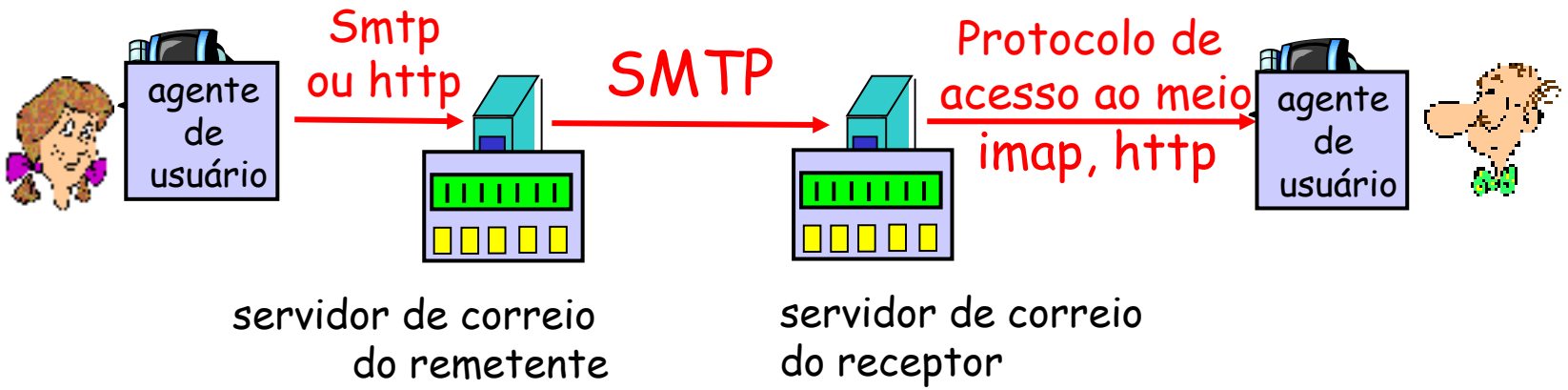
- linhas de cabeçalho, p.ex.,
  - From: alice@crepes.fr
  - To: bob@hambúrguer.edu
  - Subject: blablabla

Diferente dos comandos SMTP: FROM, RCPT TO



- corpo
  - a "mensagem", somente de caracteres ASCII

Protocolos de acesso ao correio



- **SMTP:** entrega/armazenamento no servidor do receptor
- Protocolo de accesso ao correio: recupera do servidor
  - **IMAP:** Internet Mail Access Protocol [RFC 3501]
    - Inclui manuseio de msgs armazenadas no servidor
    - Provê a recuperação, remoção (delete) e organiza as mensagens em pastas
  - **HTTP:** Hotmail , Gmail, Yahoo! Mail, etc. – provê uma interface Web sobre o SMTP. Para recuperá-las localmente usa o IMAP ou http

Correio eletrônico via web: ***http***

- Agente usuário (UA) é um navegador (IE, Firefox, Chrome, etc)
- Usuário se comunica com sua caixa de correio remota (servidor) através do protocolo ***http***, tanto para enviar quanto receber as mensagens
- SMTP é usado apenas na comunicação entre os dois servidores
- O usuário necessita apenas ter acesso a um navegador qualquer, podendo estar em qualquer lugar
- Assim como o IMAP, os usuários podem organizar suas mensagens em pastas, de forma hierárquica