Redes de Computadores I

Camada de Aplicação no TCP/IP

Prof. Ricardo Couto A. da Rocha
rcarocha@ufg.br
UFG – Regional Catalão

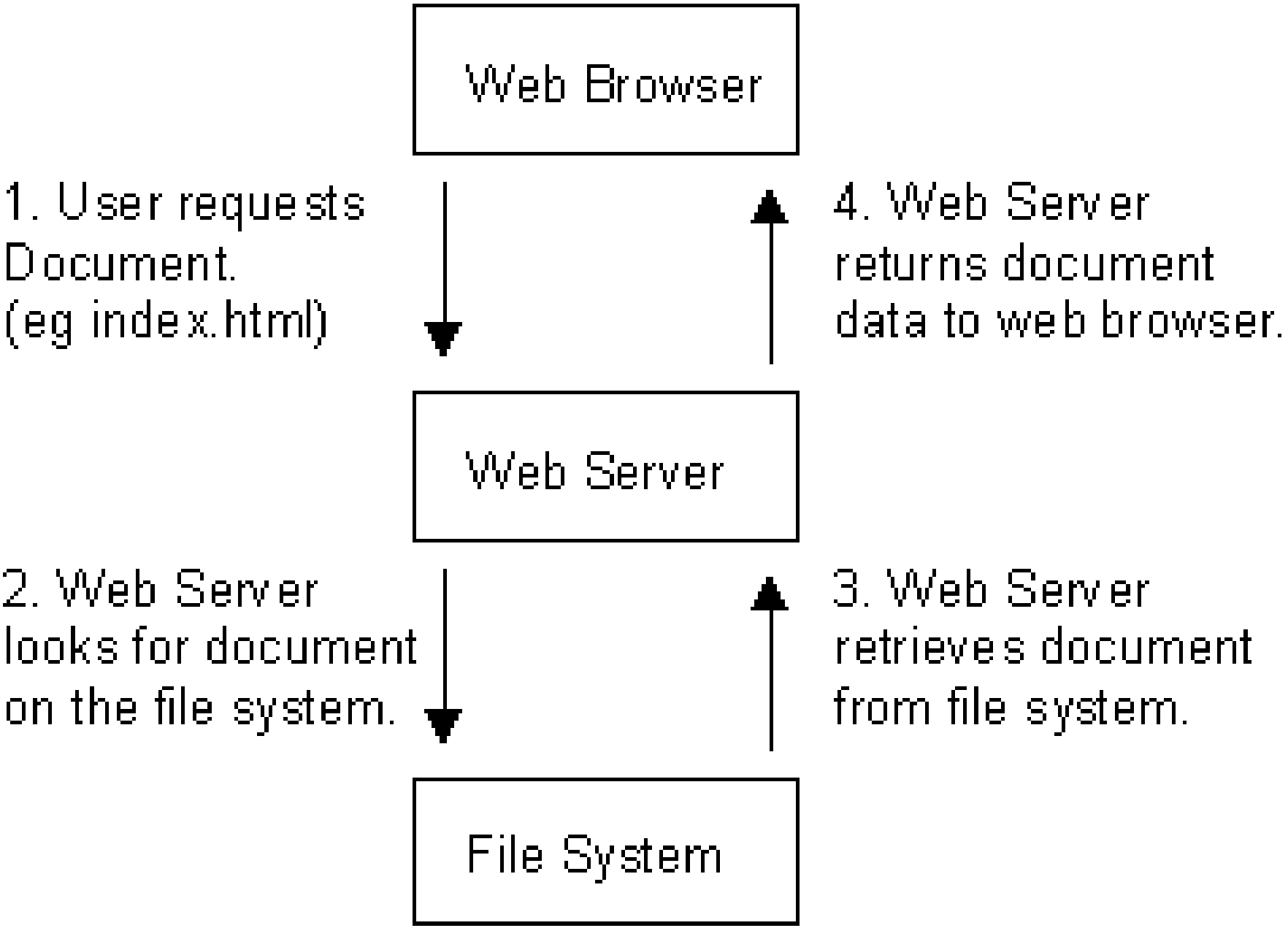
Roteiro

- Conceitos Básicos e Arquitetura
- DNS – Domain Name System
- Protocolo HTTP
- SMTP → Serviço de email
- Programação com Sockets
- Requisitos de aplicações

HTTP – Breve histórico

- Início 1989 – CERN (Centro Europeu de Pesquisas Nucleares) precisava de um meio de viabilizar o trabalho cooperativo por cientistas espalhados por diversos países, através da troca de documentos.
- 1991 – primeira versão da Web, baseada em texto.
- 1993 – NCSA - browser gráfico Mosaic
- 1994 – Netscape
- 1994 – CERN e M.I.T fundam o W3C (w3.org)

HTTP - O Servidor Web



Servidor Web

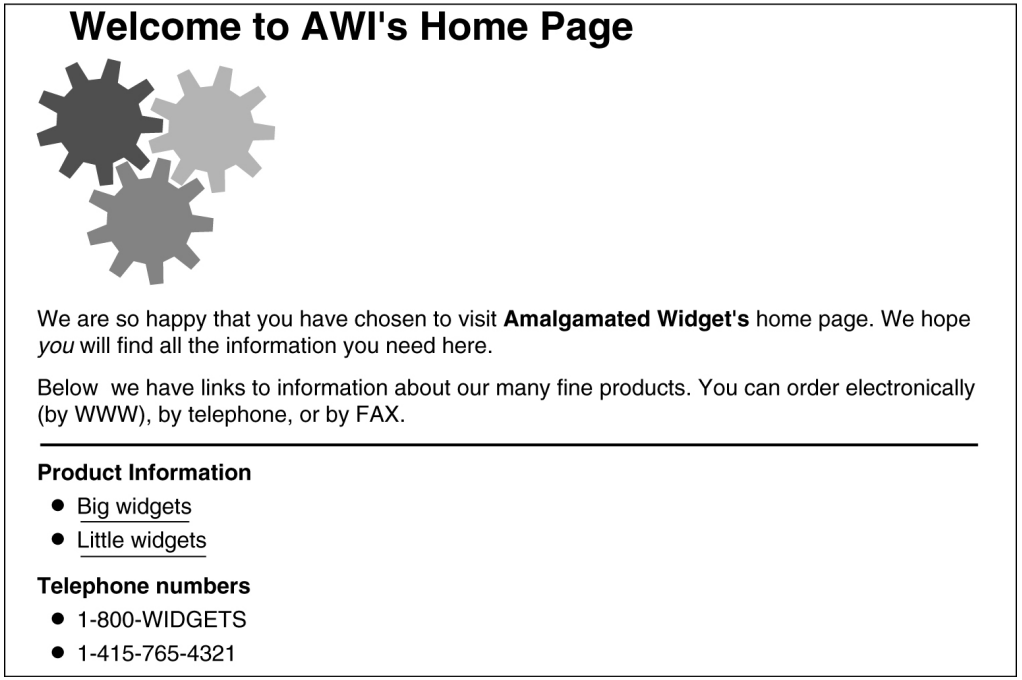
- A resposta do servidor também é na forma texto – um hipertexto no formato HTML:

```
<html>
   <head>
      <title>....</title>
   ...
   </head>
<body>
   Conteudo da página
</body>
</html>
```

Formato HTML

```
<html>
<head><title> AMALGAMATED  WIDGET,  INC. </title> </head>
<body> <h1> Welcome to AWI's Home Page</h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget's </b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by fax. </p>
<hr>
<h2> Product information  </h2>
<ul>
   <li> <a href="http://widget.com/products/big">  Big widgets </a>
   <li> <a href="http://widget.com/products/little">  Little widgets </a>
</ul>
<h2> Telephone numbers</h2>
<ul>
   <li> By telephone: 1-800-WIDGETS
   <li> By fax: 1-415-765-4321
</ul>
</body>
</html>
```

(a)

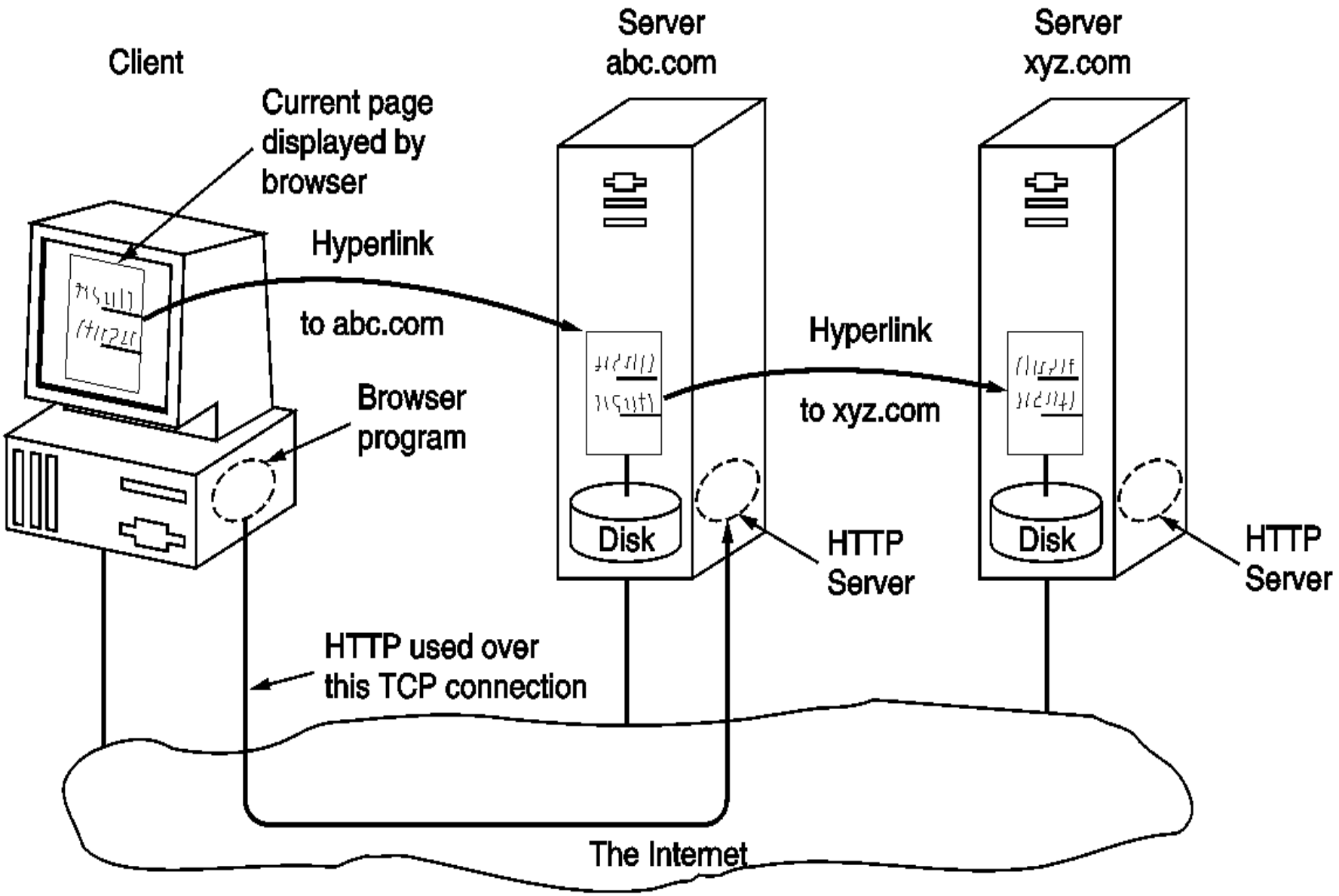


- HTML da página
- Página formatada pelo browser

Tags - Formato HTML

Tag	Description
<html> ... </html>	Declares the Web page to be written in HTML
<head> ... </head>	Delimits the page's head
<title> ... </title>	Defines the title (not displayed on the page)
<body> ... </body>	Delimits the page's body
<h *n*> ... </h*n*>	Delimits a level *n* heading
...	Set ... in boldface
...	Set ... in italics
...	Center ... on the page horizontally
...	Brackets an unordered (bulleted) list
...	Brackets a numbered list
	Starts a list item (there is no)
 	Forces a line break here
<p>	Starts a paragraph
<hr>	Inserts a Horizontal rule
<img src="...">	Displays an image here
 ... 	Defines a hyperlink

O modelo da operação da Web

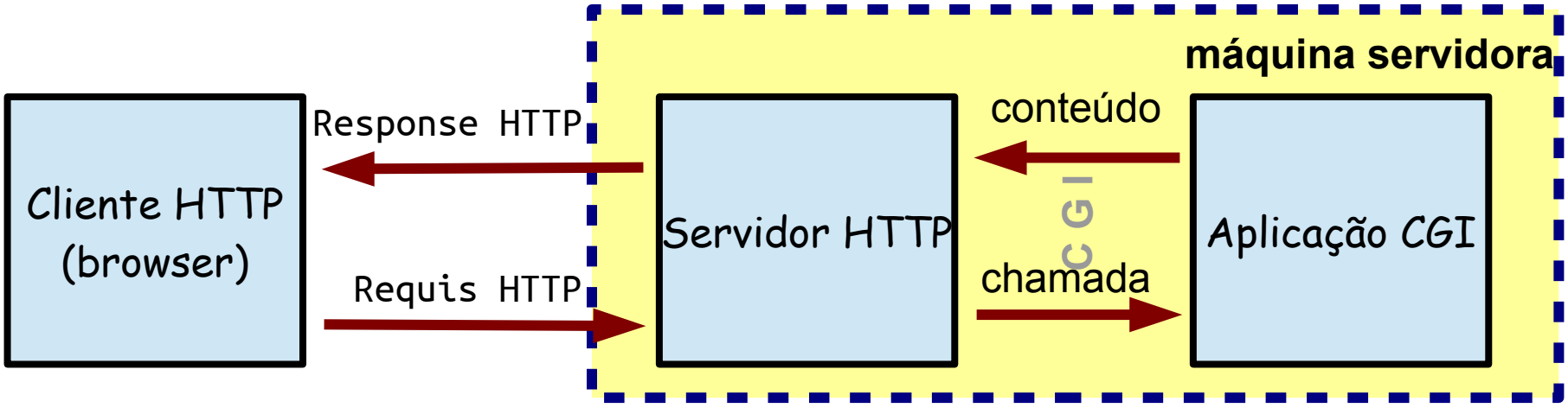


HTTP ou Serviço Web

- Idéia básica → distribuição de ***conteúdo estático*** (tipicamente páginas escritas em HTML).
- Evolução → distribuição de ***conteúdo gerado dinamicamente,*** em resposta a dados fornecidos pelo usuário (por exemplo através da consulta a um banco de dados corporativo).
- CGI – Common Gateway Interface: a forma mais antiga de prover conteúdo dinâmico.

CGI - Common Gateway Interface

- CGI → interface padrão entre servidores HTTP e programas no servidor
  - Servidores HTTP repassam para aplicações (tipicamente scritps) informações sobre requisições feitas
  - Servidores encaminham conteúdo web gerado pelas aplicações para os clientes requisitantes
- Modelos ainda aplicável, mas o uso dos servidores de aplicação (mantendo diversas aplicações) e interpretando a totalidade da requisição/resposta HTTP é mais flexível, poderoso e popular



URI – Uniform Resource Id

URI: Uniform Resource Identification

- Constituem a forma de endereçamento dos *recursos* na Web.

  Exemplo: http://www.uol.com.br/esportes/index.html

Unifica os conceitos de **URL** e RURL

É formado por quatro partes:

- o protocolo utilizado: **http**
- nome DNS da estação (e porta, se necessário): **www.uol.com.br**
- caminho de busca / Diretório: **esportes/index.html**

- URI, em relação à URL, permite o acréscimo de propriedades do conteúdo

- Sintaxe:
  - **scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]**

URL/I

- O protocolo define o tipo do servidor no qual está localizado o recurso. A partir desta informação o *browser* determina a porta e o tipo de protocolo utilizado. Os principais protocolos são:
  - `ftp:` file transfer protocol
  - `http:` hypertext transfer protocol
  - `mailto:` envio de e-mail para certo endereço
  - `telnet`:
  - `skype:` chamada Skype

URL – Uniform Resource Locator

Name	Used for	Example
http	Hypertext (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	/usr/suzanne/prog.c
news	News group	news:comp.os.minix
news	News article	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Sending email	mailto:kim@acm.org
telnet	Remote login	telnet://www.w3.org:80

HTTP

- **HTTP** - HyperText Transmission Protocol
  - Protocolo usado na comunicação entre o servidor Web (servidor HTTP) e o browser/navegador (cliente HTTP).
- HTTP estabelece como clientes realizarão consultas por mensagens pela rede e como servidores responderão às consultas.
- A idéia inicial era a de que o HTTP deveria ser um protocolo bem simples.
- Mesmo após várias modificações e melhorias, a idéia básica do HTTP continua sendo a mesma – simplicidade.

HTTP - HyperText Transmission Protocol

- Porta padrão e protocolo de transporte do processo servidor HTTP: `TCP 80`
- Uma vez estabelecida a conexão, o cliente envia um pedido e o servidor envia a resposta correspondente;
- Tendo atendido o pedido, a conexão é encerrada (a princípio).
- Modelo de interação é **requisição-resposta**.
  - Toda conversa servidor-cliente é um conjunto de requisição-resposta.
- Todas as mensagens/comandos HTTP são textuais

HTTP


Servidor executando
o servidor Web Apache
Requisição HTTP
Resposta HTTP
Resposta HTTP
Requisição HTTP
PC executando
Explorer
Mac executando
Navigator

Visão geral do HTTP

- Utiliza TCP:
  - Cliente inicia conexão TCP (cria socket) para o servidor na porta 80
  - Servidor aceita uma conexão TCP do cliente
  - mensagens HTTP (mensagens do protocolo de camada de aplicação) são trocadas entre o browser (cliente HTTP) e o servidor Web (servidor HTTP)
  - A conexão TCP é fechada
- HTTP é "stateless" (sem estado)
  - O servidor **não** mantém informação sobre os pedidos passados pelos clientes
- Protocolos que mantêm informações de "estado" são complexos!
  - Histórico do passado (estado) deve ser mantido
  - Se o servidor/cliente quebra, suas visões de "estado" podem ser inconsistentes, devendo ser reconciliadas

Passo de cliente HTTP

Um exemplo:

http://www.w3.org/hypertext/WWW/TheProject.html

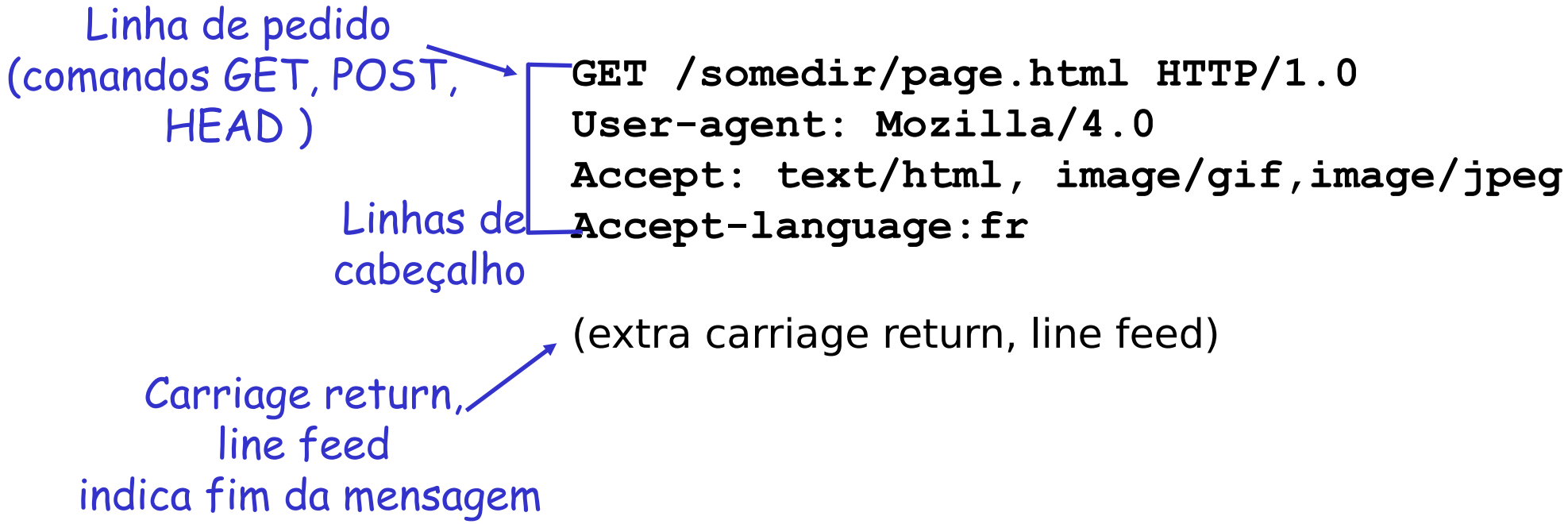
1. Browser interroga DNS pelo IP de `www.w3.org`
2. DNS responde: `18.23.0.23`
3. Browser estabelece a conexão na porta `80` de `18.23.0.23`
4. Browser envia comando: **`GET /hypertext/WWW/TheProject.html`**
5. [Exemplo] O servidor www.w3.org envia o arquivo `TheProject.html` (supondo que é conteúdo estático) armazenado no diretório `/hypertext/WWW/`

   A correspondência entre o conteúdo pedido e ***o que de fato será feito no servidor*** *é responsabilidade do servidor e não pode ser prevista pelo cliente*.

6. O cliente armazena a resposta do servidor na memória
7. A conexão TCP é encerrada
8. O browser interpreta e apresenta o conteúdo contido em `TheProject.html` (provavelmente um hipertexto HTML)

Mensagem HTTP request

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



HTTP – Comandos

Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
LINK	Connects two existing resources
UNLINK	Breaks an existing connection between two resources

Resposta de requisição HTTP

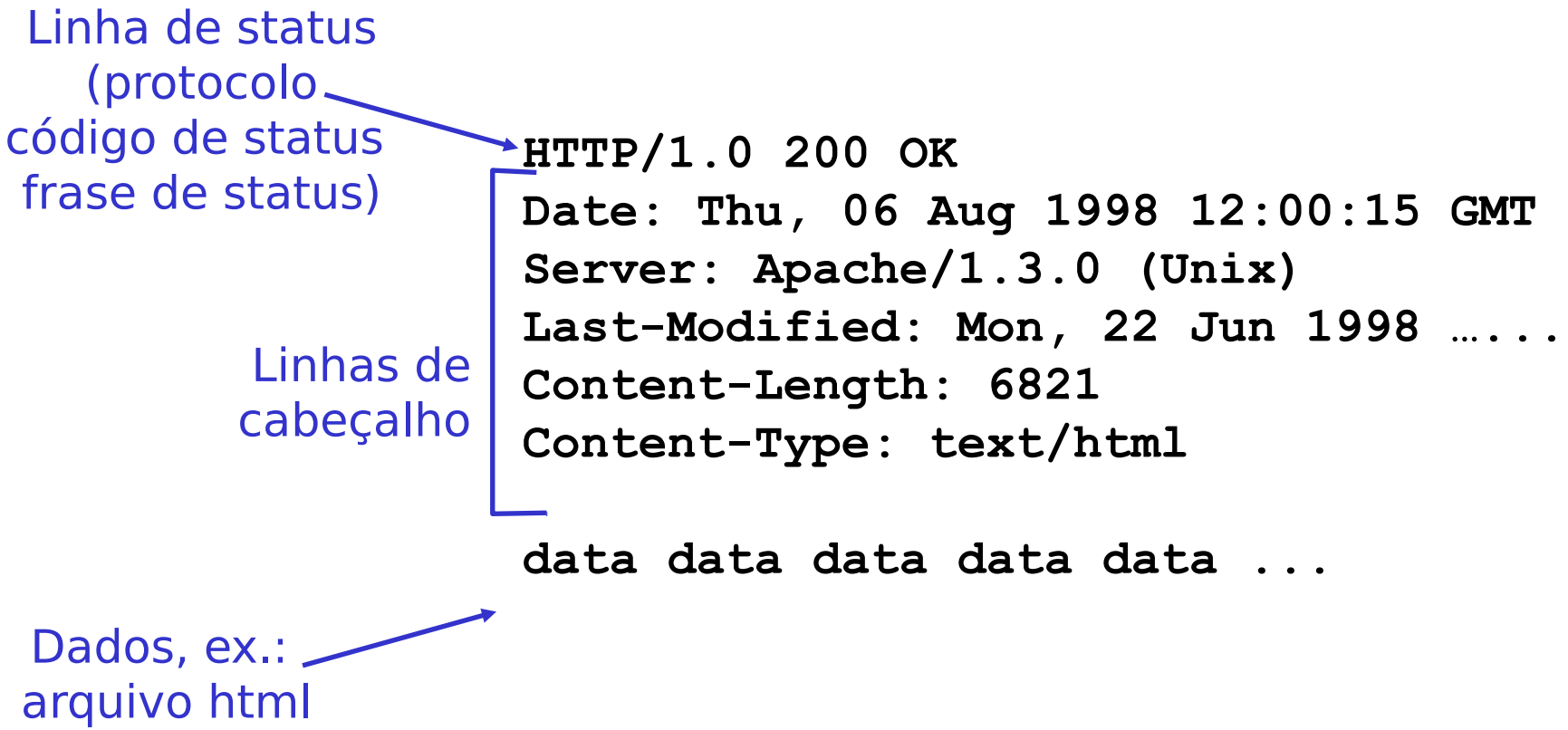
```
HTTP/1.0 200 Document follows
 MIME-Version: 1.0
 Server:CERN/3.0
 Content-Type: text/html
 Content-Length:8247
```

Cabeçalho

```
 <HEAD> <TITLE> The World Wide Web Consortion
  </TITLE>
 </HEAD>
 <BODY>
    ...
 </BODY>
```

Conteúdo

Mensagem HTTP response



Códigos de status das respostas

- Na primeira linha da mensagem de resposta servidor ➔ cliente.
- Alguns exemplos de códigos:

  - **200 OK** : Requisição bem-sucedida, objeto requisitado a seguir nesta mensagem
  - **301 Moved permanently** : Objeto requisitado foi movido, nova localização especificada a seguir nesta mensagem (Location:)
  - **400 Bad request** : Mensagem de requisição não compreendida pelo servidor
  - **404 Not Found** : Documento requisitado não encontrado neste servidor
  - **505 HTTP version not supported**

Cabeçalho HTTP

Header	Type	Contents
User-Agent	Request	Information about the browser and its platform
Accept	Request	The type of pages the client can handle
Accept-Charset	Request	The character sets that are acceptable to the client
Accept-Encoding	Request	The page encodings the client can handle
Accept-Language	Request	The natural languages the client can handle
Host	Request	The server's DNS name
Authorization	Request	A list of the client's credentials
Cookie	Request	Sends a previously set cookie back to the server
Date	Both	Date and time the message was sent
Upgrade	Both	The protocol the sender wants to switch to
Server	Response	Information about the server
Content-Encoding	Response	How the content is encoded (e.g., gzip)
Content-Language	Response	The natural language used in the page
Content-Length	Response	The page's length in bytes
Content-Type	Response	The page's MIME type
Last-Modified	Response	Time and date the page was last changed
Location	Response	A command to the client to send its request elsewhere
Accept-Ranges	Response	The server will accept byte range requests
Set-Cookie	Response	The server wants the client to save a cookie

Conexões HTTP

- HTTP não persistente
  - No máximo, um objeto é enviado sobre uma conexão TCP
  - O HTTP/1.0 utiliza HTTP não persistente
- HTTP persistente
  - Múltiplos objetos podem ser enviados sobre uma conexão TCP entre o cliente e o servidor
  - O HTTP/1.1 utiliza conexões persistentes em seu modo padrão
  - Evita-se o tempo necessário para início e término de conexões TCP e um possível overhead para gerenciamento das conexões
    - **Sem pipeline**: cliente envia nova requisição pela conexão, após o recebimento da resposta (conteúdo web)
    - **Com pipeline**: cliente pode enviar requisições subsequentes, mesmo que não tenha recebido todo o conteúdo anteriormente solicitado

Conexões HTTP

- Conexões HTTP persistentes e com pipeline aumentam o desempenho da interação cliente-servidor HTTP, mas ainda não permitem o carregamento concorrente de conteúdo
  - Na prática, os browsers estabelecem várias conexões com o servidor para requisitar objetos web concorrentemente
  - Exemplo de configuração de browsers (Firefox 2.0 / 46 – 7 anos de diferença)

Máximo de conexões	24 / 256
Máx. conexões por servidor	8 / 6
Máx. requisições por pipelining	4 / 32
Timeout de conexões	300 s / 90s

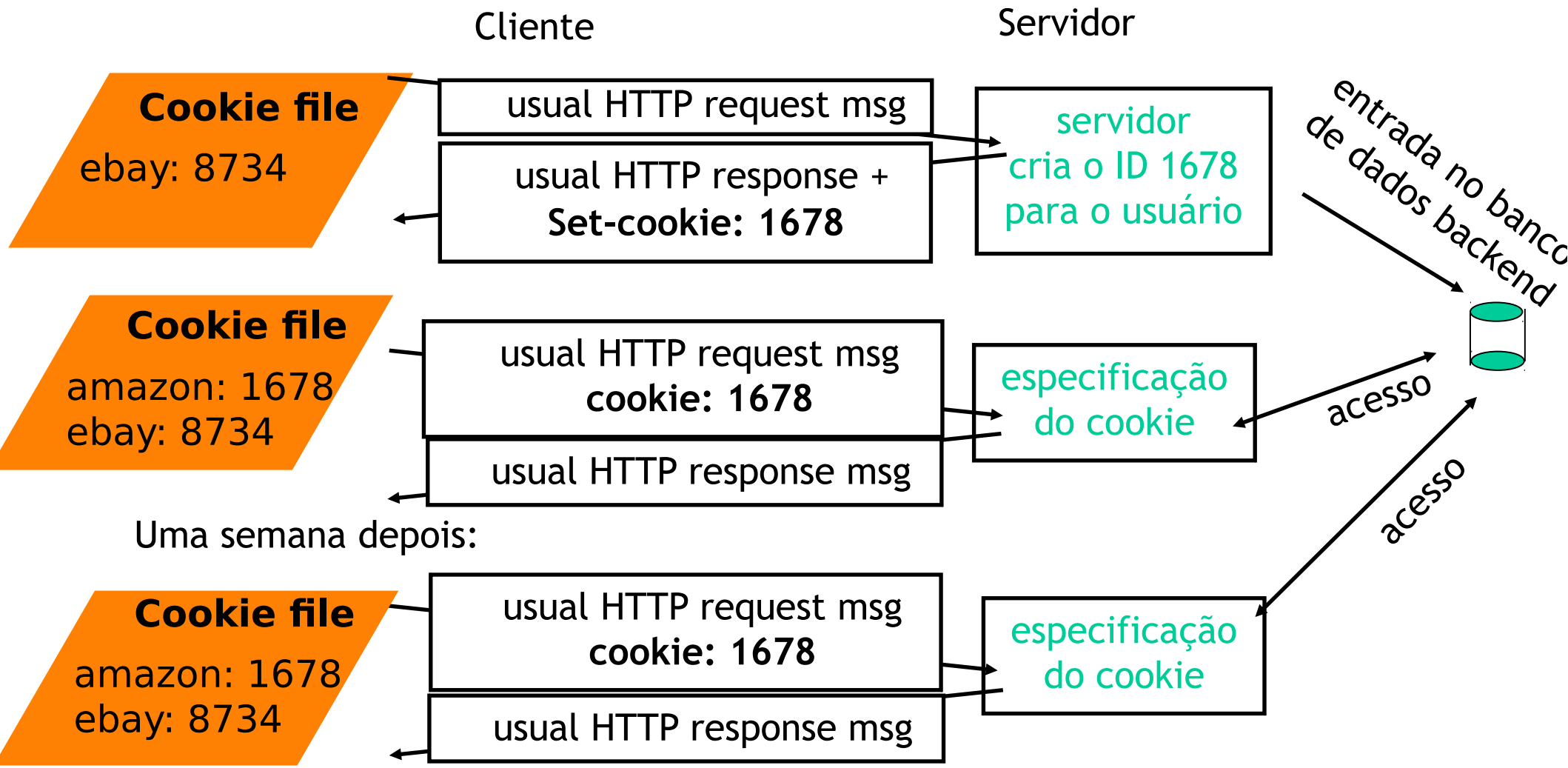
Estado na interação web

- HTTP não possui estado (stateless)
- O que acontece nas seguintes situações?
  - Usuário acessa página web protegida, mas só precisa digitar login/senha uma vez
    - Browser envia login/senha em toda requisição, mantendo em memória o último login feito pelo usuário. Falsa noção de estado!
  - Cliente interrompe download de objeto web e mais tarde continua de onde havia parado
    - Cliente armazena último byte baixado. Nas requisições subsequentes ele utiliza GET parcial, informando a região (byte origem e byte fim) do objeto requisitado

Estado na interação web

- Mecanismos para manutenção de estado em interações na web
  - Cookies
  - Sessões do lado do servidor
  - Variáveis escondidas em formulários
  - Parâmetros codificados nas URLs
- Elementos de cookies
  - Linha de cabeçalho do cookie na mensagem HTTP response
  - Linha de cabeçalho do cookie na mensagem HTTP request
  - Arquivo de cookies mantido no host do cliente e mantido pelo browser do usuário
  - Banco de dados no servidor, responsável por associações entre cookies e entradas em um banco de dados

Estado no HTTP: Cookies



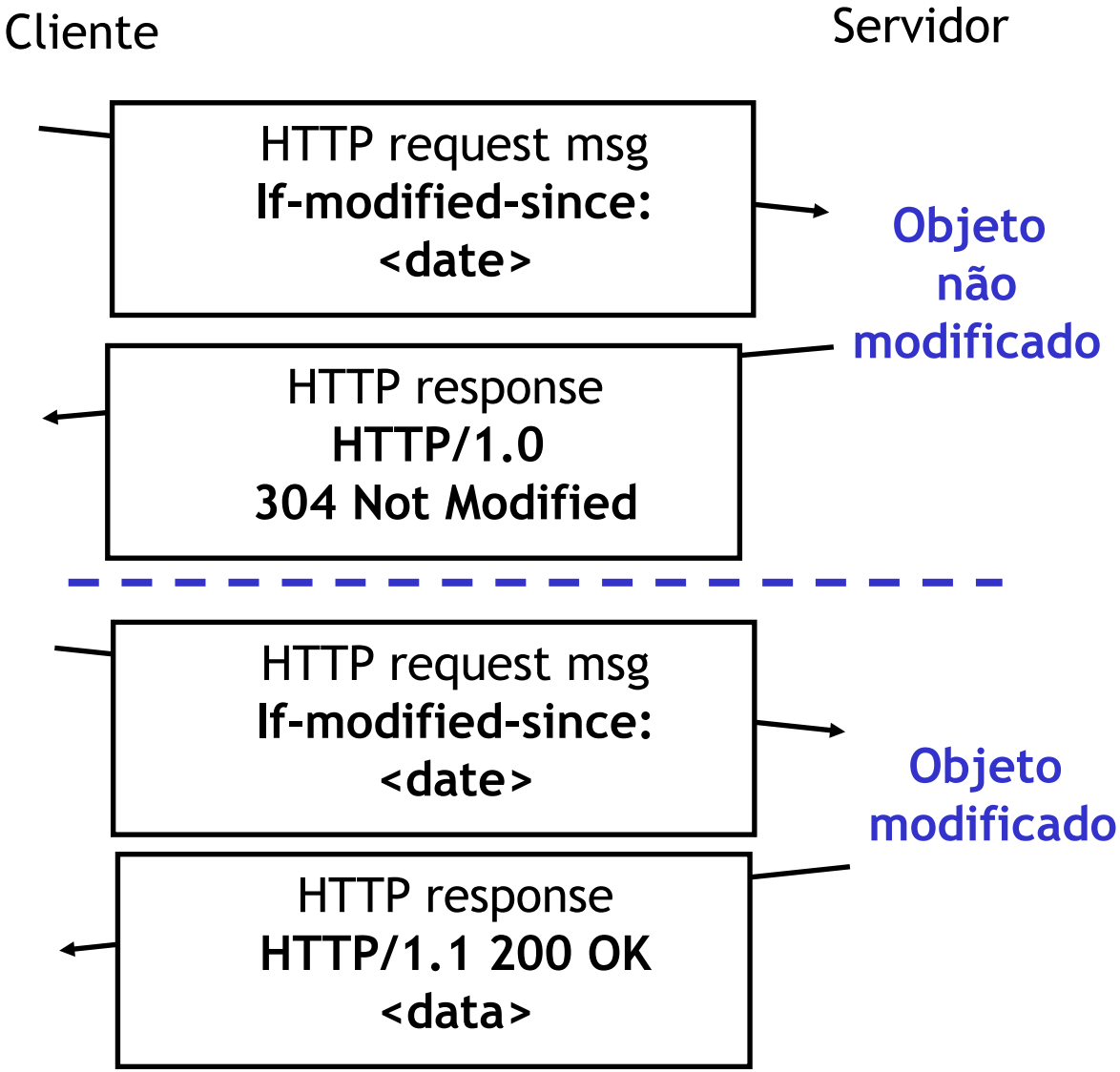
Demonstração de cookies com acesso real a página em outro video

Caching de objetos

- Podem ser implementados tanto no cliente como em servidor proxy (a ser discutido mais adiante)
- Qual objeto deve ser carregado do cache, ao invés do servidor?
  - Quando foi a última modificação no objeto? Cabeçalho **`Last-modified`**!
  - Objetos modificados recentemente são bons candidatos a serem sempre recarregados.
  - Tag HTML permite indicar quando páginas web sempre expiram ou qual é a sua data de expiração
- Implementação de caching é baseada fortemente no GET condicional
  - GET se modificado desde data X

Caching com GET condicional

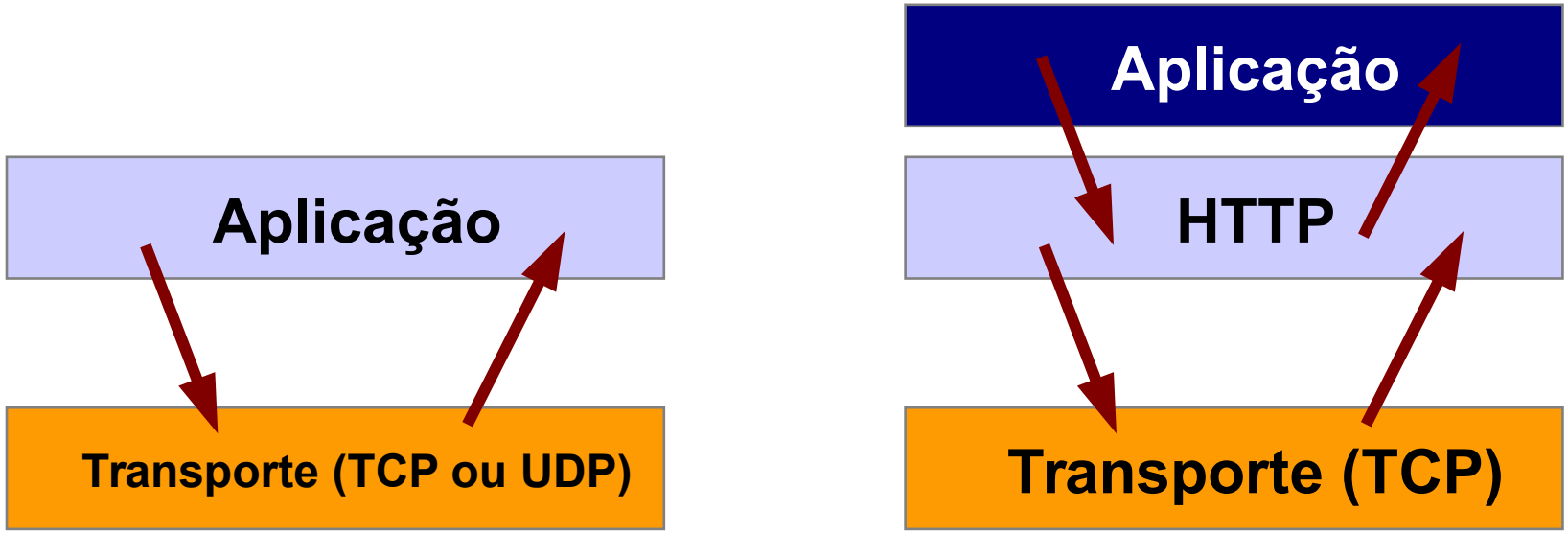
- Cliente: especifica data da versão armazenada no pedido HTTP
  - `If-modified-since: <date>`
- Servidor: resposta não contém objeto se a cópia é atualizada:
  - `HTTP/1.0 304 Not Modified`



HTTP com Protocolo de Transporte

- HTTP é um protocolo extremamente popular e difundido na Internet
  - Aplicações web (baseadas em HTTP) são sustentáculo do interesse na Internet.
- Conversas de clientes com a porta 80 dificilmente são bloqueados (exceto quando se quer bloquear o tráfico HTTP explicitamente)
- Por este motivo, protocolos que seguem o modelo request-reply de interação usualmente exploram o HTTP como protocolo de transporte.

HTTP como protocolo de transporte



Exemplos:

- **SOAP** → Requisições a WebServices
- **DASH** (Dynamic Adaptive Streaming over HTTP) → usado no Youtube e Netflix para transmitir videos!

Roteiro

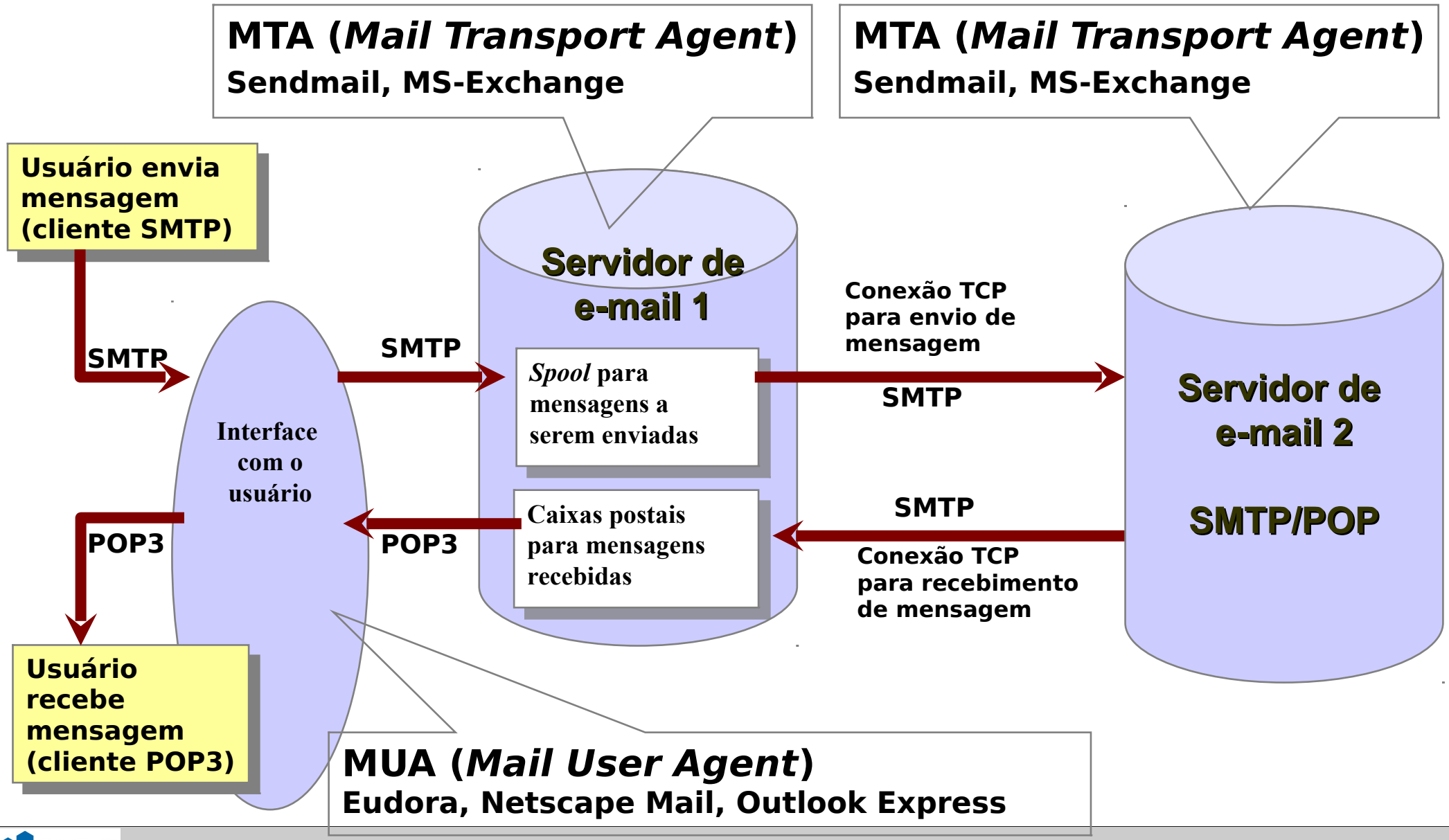
- Conceitos Básicos e Arquitetura
- DNS – Domain Name System
- Protocolo HTTP
- SMTP → Serviço de email
- Programação com Sockets
- Requisitos de aplicações

Correio Eletrônico

Existem duas categorias de aplicações no correio eletrônico:

- MUA (*Mail User Agent*)
  - Interage com o usuário para
    - Compor mensagem a ser enviadas.
    - Visualizar as mensagens recebidas.
  - Avisa periodicamente ao usuário se chegou mensagem
  - Ex.: Eudora, Outloook Express, Webmail BOL
- MTA (*Mail Transport Agent*)
  - Põe em um *spool* (fila) as mensagens a serem enviadas
  - Descobre através do DNS o IP de servidor destino
  - Estabelece uma conexão SMTP com o servidor destino para o envio da mensagem
  - Tenta enviar a mensagem de imediato, caso não consiga, periodicamente tenta transmitir as mensagens ainda não enviadas
    - Caso não possa enviar dentro de × horas/dias, retorna a mensagem para o usuário remetente e informa-o do fato.

Arquitetura do Correio Eletrônico



Mensagem de Correio

- Dividida em duas partes: cabeçalho e corpo
- Cabeçalho
  - Contém informações necessárias à transferência da mensagem
  - Informa por quais servidores de correio a mensagem passou
  - Informa a data de envio e de recebimento
  - Contém os seguintes campos, dentre outros:
    - Endereço do remetente (campo "From: ")
    - Endereço do destinatário (campo "to: ")
    - Assunto da mensagem (campo "subject: ")
    - Destinatários que receberão uma cópia da msg. (campo "cc: ")
    - Destinatários que receberão uma cópia escondida (campo "bcc: ")
- Corpo
  - Contém a mensagem propriamente dita

Endereço Eletrônico

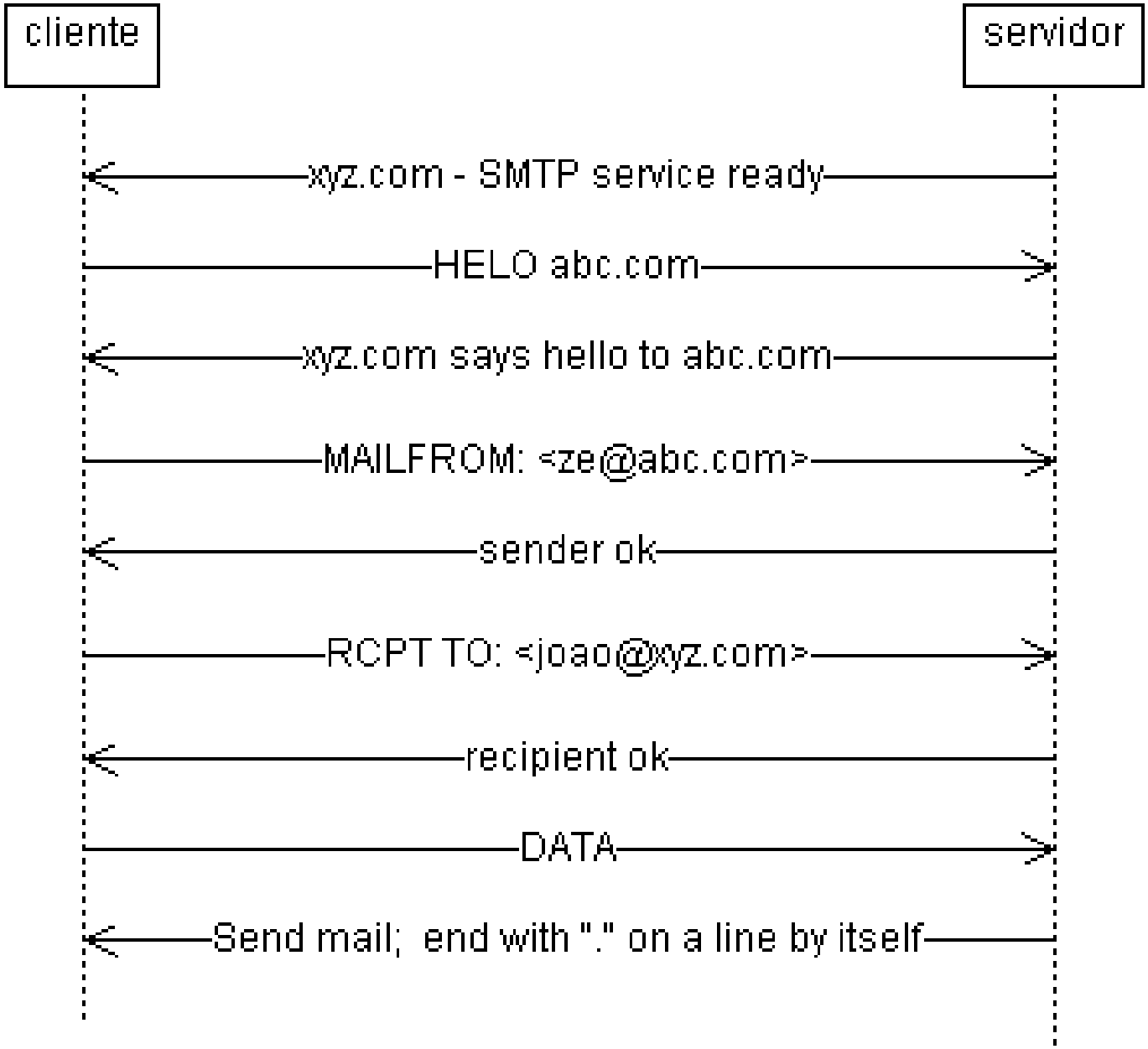
- Formato do Endereço Eletrônico
  - São escritos através de um par de identificadores separados pelo símbolo @

    nome_do_usuário@nome_do_dominio
  - Exemplo: **aluno12341@ufg.br**
  - **nome_do_usuário**
    - Identificação (*Login*) do usuário na rede
    - Não podem conter espaços e nem acentos
  - **nome_do_domínio**
    - Identifica o domínio do usuário destinatário
    - Pode ser o nome do servidor de correio do domínio
    - Será usado para a procura do registro MX do DNS

SMTP

- *Simple Mail Transport Protocol*
  - Protocolo da família TCP/IP encarregado de transmitir mensagens de correio eletrônico
  - Se assemelha ao sistema de correio comum
    - Confia integralmente nas informações passadas pelos agentes
  - Utiliza o TCP
  - Atende requisições de clientes na porta 25
  - Utiliza o registro MX do DNS para descobrir o servidor de destino
  - Paradigma comando/resposta
    - Para cada comando enviado do Emissor-SMTP para o Receptor-SMTP ocorrerá uma resposta do Receptor com um código numérico seguido de um texto
    - Comandos básicos obrigatórios
      - **HELO**, **MAIL FROM**, **RCPT TO**, **DATA**, **NOOP**, **QUIT** e **RSET**

SMTP


cliente
servidor
xyz.com - SMTP service ready
HELO abc.com
xyz.com says hello to abc.com
MAILFROM: <ze@abc.com>
sender ok
RCPT TO: <joao@xyz.com>
recipient ok
DATA
Send mail; end with "." on a line by itself

SMTP

Passos para o envio de uma mensagem no SMTP

1. Cliente se conecta no servidor SMTP ***A*** (na porta 25) e informa o destinatário da mensagem; Ex.: `ricardo@ufg.br`

2. Servidor SMTP ***A*** tenta descobrir os servidores do tipo MX configurados para o domínio `ufg.br`, enviando uma requisição para o servidor DNS da sua rede local;
O servidor MX (Mail Exchange) corresponde ao servidor de e-mail do domínio informado como destinatário na mensagem do usuário

3.Se o servidor SMTP de destino existe, o cliente transfere a mensagem para o servidor SMTP ***A***

4.O Servidor SMTP ***A*** conecta na porta 25 do servidor SMTP de destino (smtp do domínio `catalao.ufg.br`), e transfere a mensagem de e-mail do usuário/cliente;

POP3

- *Post Office Protocol*
  - Protocolo da família TCP/IP encarregado de resgatar as mensagens de correio eletrônico que estão no servidor.
  - Utiliza o TCP
  - Trabalha na porta 110
  - Paradigma comando/resposta
    - Para cada comando enviado do Emissor-POP3 para o Receptor-POP3 ocorrerá uma resposta do Receptor com um código numérico seguido de um texto
    - Comandos básicos obrigatórios
      - **USER**, **PASS**, **STAT**, **LIST**, **RETR**, **DELE** e **QUIT**

POP3

- Passos para o recebimento de mensagens de e-mail
  - Cliente se conecta no servidor POP3 (na porta 110)
  - Informa usuário e senha (Autenticação do usuário)
  - Se as informações de autenticidade estiverem corretas, então as mensagens adicionais do protocolo podem obter sucesso.

SMTP e Spam

- Afinal, por que é tão fácil disseminar Spam?
  - Meta-informações (`from:`,`to:`,...) são livremente incluídas nas mensagens
  - Autenticação dos remetentes é praticamente nula!
    - Campo ***HELO*** e ***MAIL from:*** são fáceis de forjar
    - MTAs tipicamente não autenticam domínios e usuários nos domínios
  - Servidores de e-mail mal protegidos
  - Extensão para autenticação foi desenvolvida, mas o problema agora é disseminá-la nos servidores atuais
    - O comprometimento com o recebimento de qualquer mensagem evita a implantação da autenticação
- Alternativas
  - Filtros de usuários - cliente, tipicamente
  - Confirmação de remetentes - servidor
  - Filtros automáticos (bayesianos) – cliente, tipicamente
  - Listas negras de IPs de spammers – servidor
  - Validação de domínio (domainkeys) – servidor

IMAP

- IMAP - Internet Message Access Protocol
  - Protocolo com o qual clientes de e-mail acessam servidores de e-mail (substituição ao POP3)
  - Além de baixar mensagens, permite:
    - Estabelecer uma estrutura hierárquica de diretórios no servidor
    - Manutenção de cópias nos clientes ➔ servidor mantém a infra-estrutura real que mantém as mensagens
    - Estrutura de armazenamento de e-mails nos clientes é sincronizada com a estrutura no servidor
  - No IMAP, não apenas mensagens, mas a estrutura de diretórios de e-mails são mantidas no servidor
  - Qualquer cliente de e-mail (programa ou webmail) terá acesso à mesma estrutura de diretórios e e-mails mantida no servidor.
  - Resolve o problema de sincronização entre clientes
  - Exemplo de clientes: Thunderbird, Squirrelmail, Outlook, KMail
- IMAP é um protocolo muito complexo e exige servidores mais poderosos