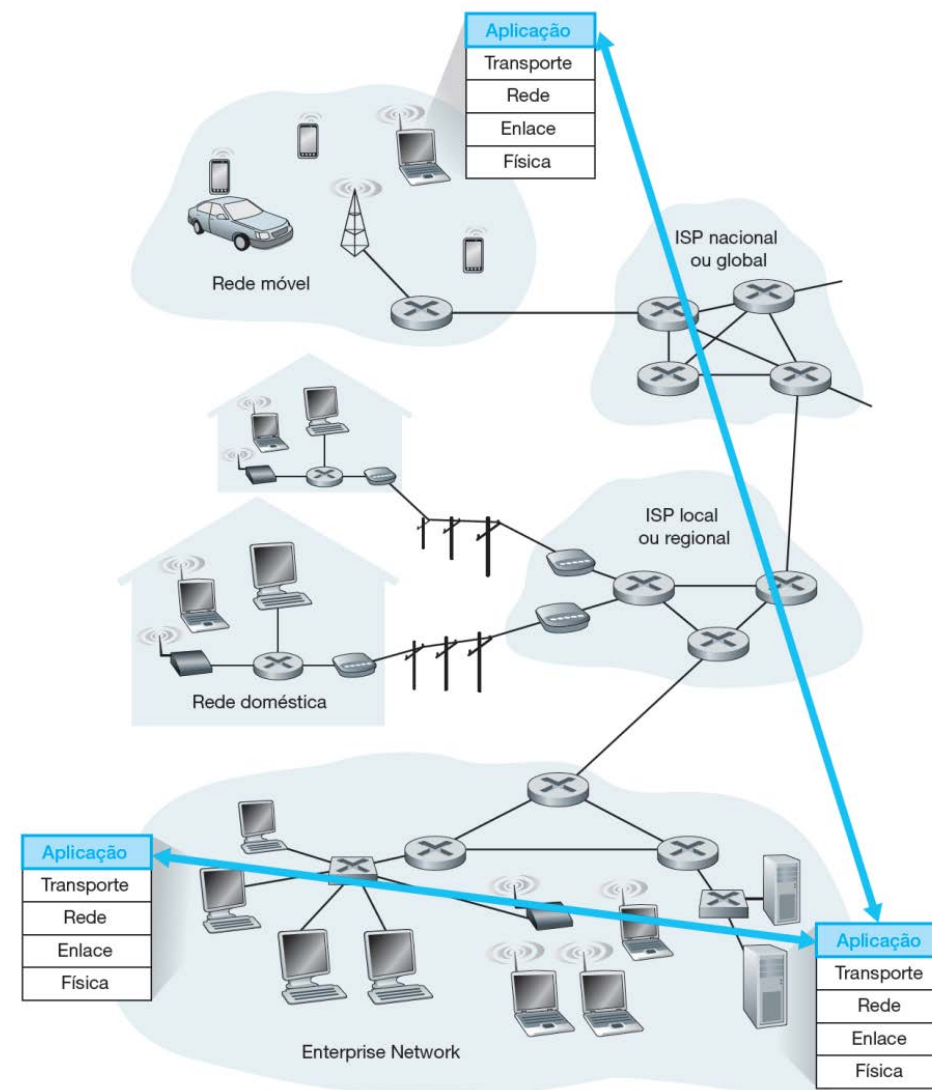


Roteiro - Unidade II

- Programação com Sockets
 - API sockets em Python
- Aplicações, protocolos e serviços TCP/IP
 - **Protocolos HTTP, DNS e DHCP**
 - Protocolos de transporte TCP e UDP
 - Protocolo da Internet (IP)
 - Protocolo Ethernet e Wi-Fi

A Camada de Aplicação

- Uma aplicação de rede consiste em um programa executado nos sistemas finais que se comunicam por meio da rede
 - Exemplos:
 - Numa aplicação web há dois programas distintos:
 - O navegador que é executado no lado do cliente
 - E o servidor web que é executado no lado do servidor, provendo conteúdo ao cliente
 - Em um compartilhamento de arquivos P2P:
 - Existem diferentes programas (mas que implementam o mesmo protocolo) em cada máquina que participa do compartilhamento



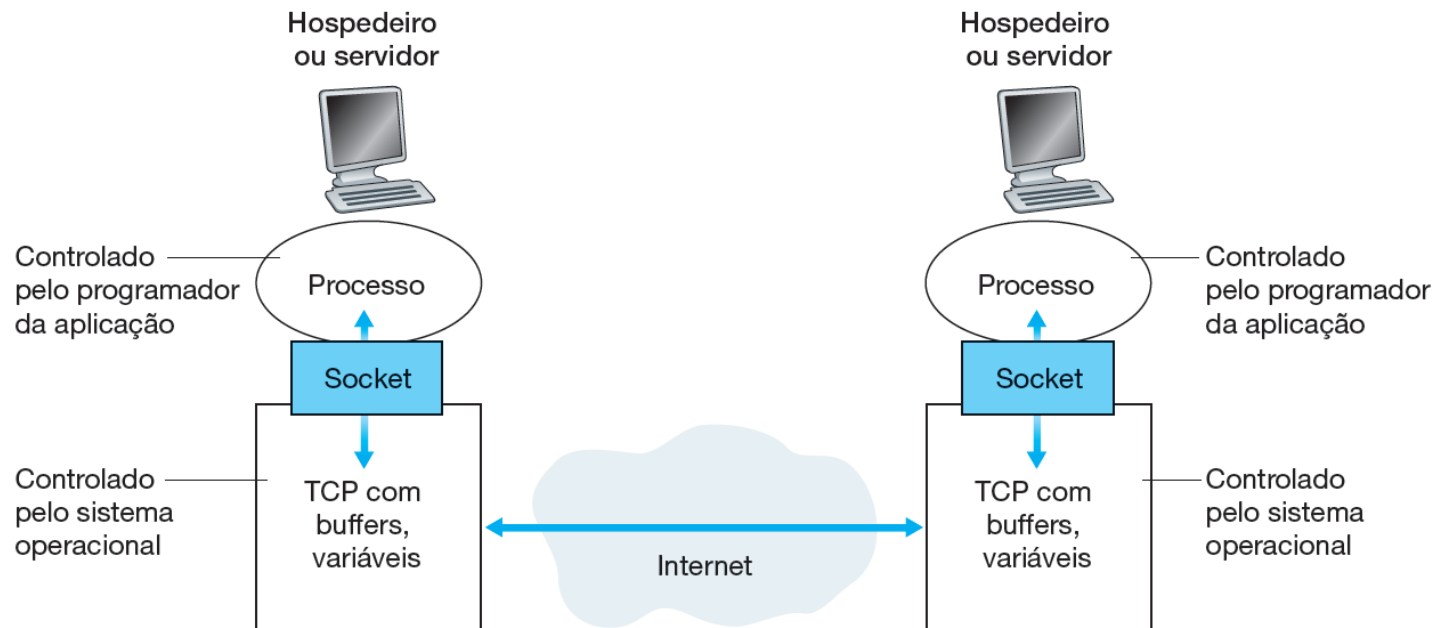
Protocolos da Camada de Aplicação

- Definem como os processos de uma aplicação trocam mensagens entre si, em particular:
 - Tipos de mensagens trocadas, por exemplo, de requisição ou resposta
 - A sintaxe dos vários tipos de mensagens, tais como os campos da mensagem
 - A semântica dos campos, isto é, o significado da sua informação
 - Regras para determinar quando e como um processo envia e responde mensagens
- Uma aplicação de rede consiste em pares de processos que enviam mensagens uns para os outros por meio de uma rede
 - Processo: programa que executa num sistema final
 - Processos no mesmo sistema final se comunicam usando comunicação interprocessos (definida pelo sistema operacional)
 - Processos em sistemas finais distintos se comunicam trocando mensagens através da rede

Protocolos da Camada de Aplicação

- Comunicação entre processos

- 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:
 - O endereço do host e
 - Um identificador que especifica o processo receptor no host de destino



Um socket é análogo a uma porta:

- Processo transmissor envia a mensagem através da porta
- O processo transmissor assume a existência da infraestrutura de transporte no outro lado da porta que faz com que a mensagem chegue ao socket do processo receptor

De que serviços uma aplicação necessita?

- Integridade dos dados (sensibilidade a perdas)
 - Algumas aplicações (ex: transferência de arquivos, transações web) requerem uma transferência 100% confiável
 - Outras (ex: áudio) podem tolerar algumas perdas
- Temporização (sensibilidade a atrasos)
 - Algumas aplicações (ex: telefonia Internet, jogos interativos) requerem baixo retardo para serem “viáveis”
- Vazão (*throughput*)
 - Algumas aplicações (ex: multimídia) requerem quantia mínima de vazão para serem “viáveis”
 - Outras aplicações conseguem usar qualquer quantia de banda disponível
- Segurança
 - Criptografia, integridade dos dados, autenticação, ...

De que serviços uma aplicação necessita?

- Requisitos de aplicações de rede

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 da Internet

Serviços providos pelos protocolos de transporte da Internet

- Serviço TCP:

- Transporte confiável entre processos remetente e receptor
- Controle de fluxo e de congestionamento: para evitar “afogar” receptor
- Não provê: garantias temporais ou de banda mínima
- Orientado a conexão: apresentação requerida entre cliente e servidor

- Serviço UDP:

- Transferência de dados não confiável entre processos remetente e receptor
 - Não provê: estabelecimento da conexão, confiabilidade, controle de fluxo, controle de congestionamento, garantias temporais ou de banda mínima
- Mas qual é o interesse em usar UDP?
 - Tempo de resposta baixo!

Aplicações da Internet

Seus protocolos 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

Protocolos da Camada de Aplicação

- Alguns protocolos da camada de aplicação estão definidos em RFCs (*Request for Comments*)
 - Documentos técnicos desenvolvidos e mantidos pelo IETF (*Internet Engineering Task Force*)
 - São de domínio público - <https://www.tools.ietf.org/html>
 - Especificam os detalhes de funcionamento de determinado protocolo
 - Exemplo:
 - HTTP/1.1 - RFC2616
 - SSH - RFC4251
 - SMTP - RFC5321
- Outros são proprietários e não estão disponíveis ao público
 - Exemplo:
 - Skype
 - Whatsapp

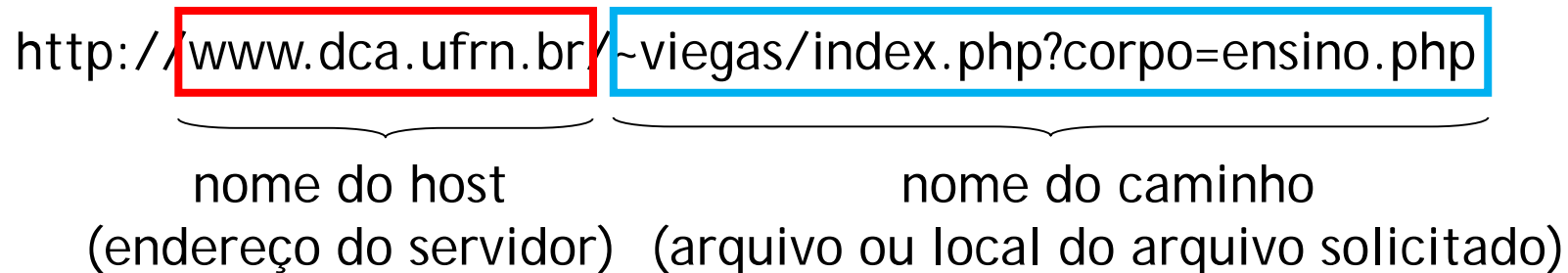
World Wide Web (WWW)

- É um sistema de documentos dispostos em milhares de computadores na Internet que permitem o acesso às informações apresentadas no formato de hipertexto
- Teve seu início no Conselho Europeu para Pesquisa Nuclear (CERN) em 1989
 - Tim Berners-Lee
- Sua enorme popularidade se deve, principalmente, a dois fatores:
 - Interface gráfica colorida e de fácil utilização
 - Uma imensa variedade de informações de todos os tipos

A Web e o HTTP

- Páginas Web consistem de objetos
 - Um objeto pode ser um arquivo HTML, uma imagem JPEG, um applet Java, um arquivo de áudio,...
- A base é uma página web é um arquivo em linguagem HTML que inclui vários objetos referenciados
 - Cada objeto é endereçável por uma URL (*Uniform Resource Locators*)
 - Exemplo de URL:

`http://www.dca.ufrn.br/~viegas/index.php?corpo=ensino.php`



nome do host
(endereço do servidor)

nome do caminho
(arquivo ou local do arquivo solicitado)

- Ao digitar o URL, o navegador/cliente solicita ao servidor uma página HTML ou um arquivo
 - Utiliza o protocolo HTTP para a comunicação

Protocolo HTTP

- HTTP: HyperText Transfer Protocol

- Modelo cliente/servidor

- Cliente: navegador pede/recebe e “visualiza” objetos Web
 - Servidor: servidor web que envia objetos em resposta a pedidos

- Usa serviço de transporte TCP:

- Cliente inicia conexão TCP (cria socket) com o servidor, comumente na porta 80
 - Servidor aceita conexão TCP do cliente
 - Mensagens HTTP são trocadas entre cliente e servidor
 - Encerra conexão TCP

- HTTP é “sem estado”

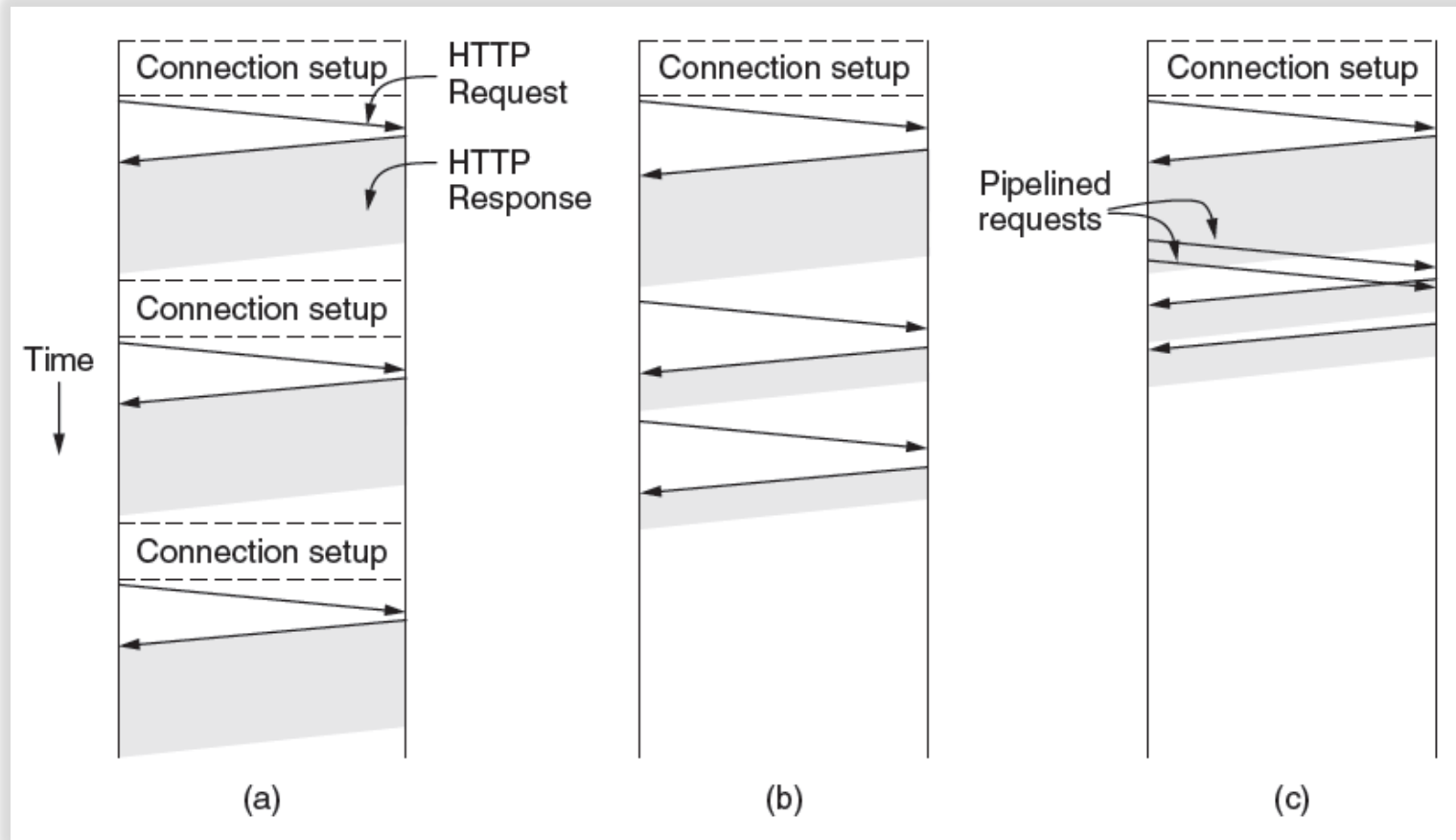
- Servidor não mantém informação sobre pedidos anteriores do cliente



Protocolo HTTP

- As conexões HTTP são estabelecidas de duas maneiras:
 - HTTP não persistente
 - A conexão é encerrada logo após uma solicitação
 - No máximo um objeto é solicitado e enviado numa conexão TCP
 - Para solicitar múltiplos objetos requer o uso de múltiplas conexões
 - HTTP persistente
 - A conexão se mantém aberta durante um período de tempo determinado
 - Múltiplos objetos podem ser solicitados e enviados sobre uma única conexão TCP entre cliente e servidor
 - Permite o envio de objetos em forma de pipeline
 - Múltiplos objetos podem ser enviados de uma vez, sem esperar a confirmação

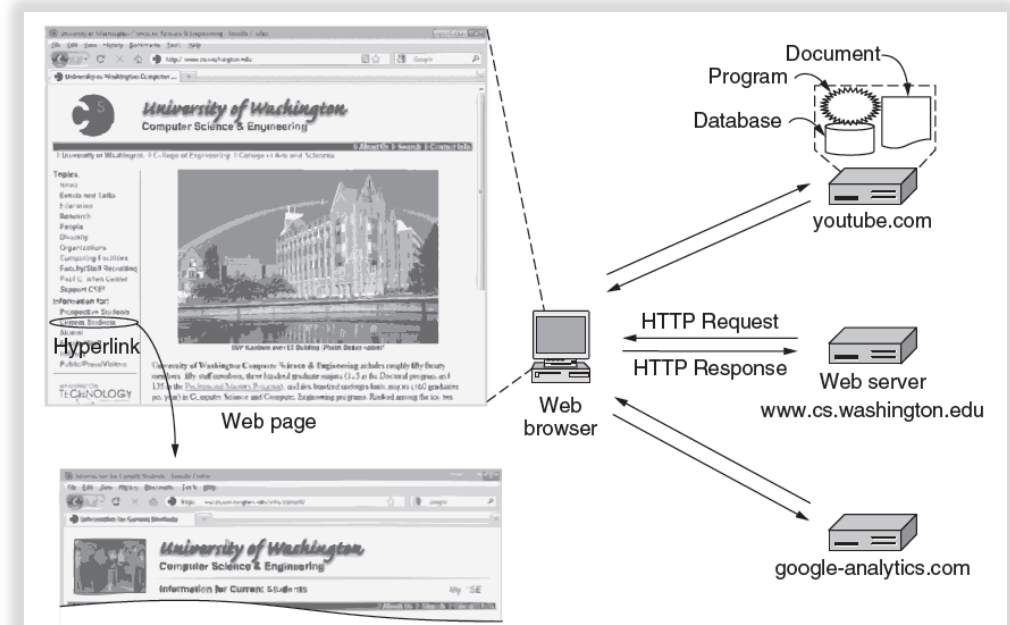
Protocolo HTTP



- (a) Múltiplas conexões e solicitações sequenciais (não persistente)
- (b) Uma conexão persistente e solicitações sequenciais
- (c) Uma conexão persistente e solicitações por *pipeline*

Protocolo HTTP

- Funcionamento:
 - O lado cliente:
 - O navegador realiza uma série de tarefas para exibir o conteúdo solicitado pela URL:
 - Obtém o IP do servidor solicitando ao servidor de nomes (DNS)
 - Estabelece uma conexão TCP com o servidor na porta 80
 - Solicita a página index.html usando um comando HTTP
 - Caso a página inclua links para outros recursos para exibição (URLs), buscará estes recursos da mesma maneira em outros servidores
 - Exibe a página
 - Encerra a conexão após um tempo



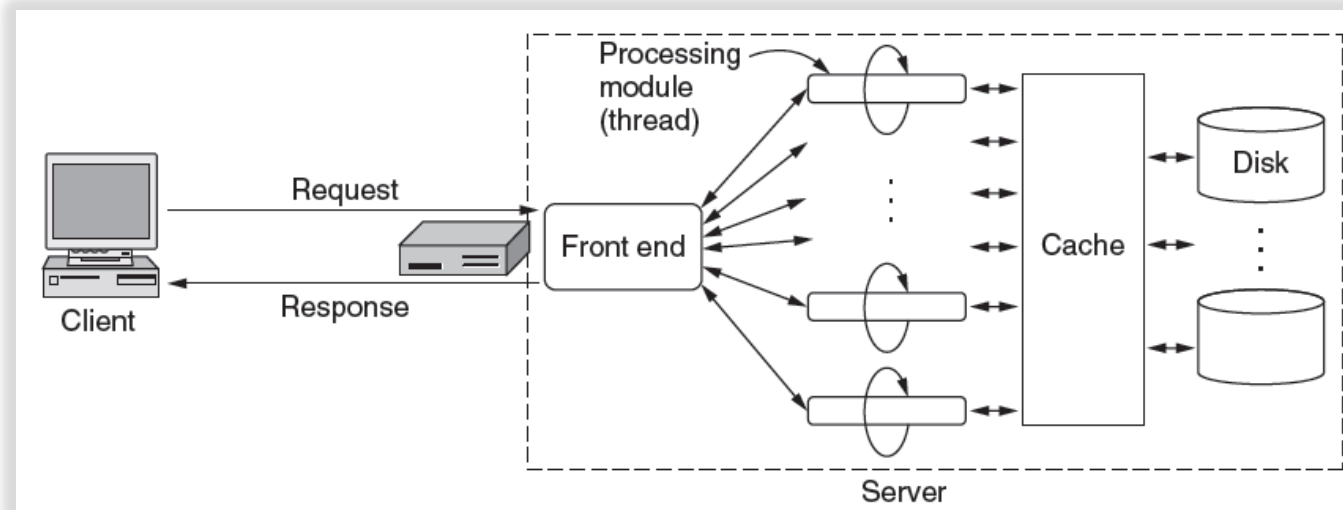
Protocolo HTTP

- Funcionamento:

- O lado servidor:

- As etapas que o servidor executa:

- Aceitar uma conexão TCP de um cliente (geralmente de um navegador)
 - Obter o caminho até a página, que é o nome do arquivo solicitado
 - Obter o arquivo (do disco ou cache) ou gerar o conteúdo de forma dinâmica
 - Enviar o conteúdo ao cliente
 - Encerrar a conexão TCP



Protocolo HTTP

- A comunicação HTTP ocorre por meio de métodos (comandos):
 - O HTTP aceita operações chamadas de métodos
 - Representam a ação desejada
 - Cada solicitação consiste de uma ou mais linhas de texto ASCII, sendo a primeira palavra da primeira linha o método solicitado:

Método	Descrição
GET	Lê uma página/recurso web
HEAD	Lê um cabeçalho de página web. Pode ser usado para indexação ou testar a validade de um URL
POST	Cria uma página web. Usado para envio de dados de formulários para o servidor
Outros: PUT, DELETE, TRACE, CONNECT, OPTIONS	

Protocolo HTTP

- Códigos de erro:

- Toda solicitação obtém uma resposta que possui uma linha de status, com um código de três dígitos informando se a solicitação foi atendida ou qual foi o erro:

Código	Significado	Exemplos
1xx	Informação	100 = servidor concorda em tratar da solicitação do cliente
2xx	Sucesso	200 = solicitação com sucesso 204 = nenhum conteúdo presente
3xx	Redirecionamento	301 = página movida 304 = página em cache ainda válida
4xx	Erro do cliente	403 = página proibida 404 = página não localizada
5xx	Erro do servidor	500 = erro interno do servidor 503 = tente novamente mais tarde

Protocolo HTTP

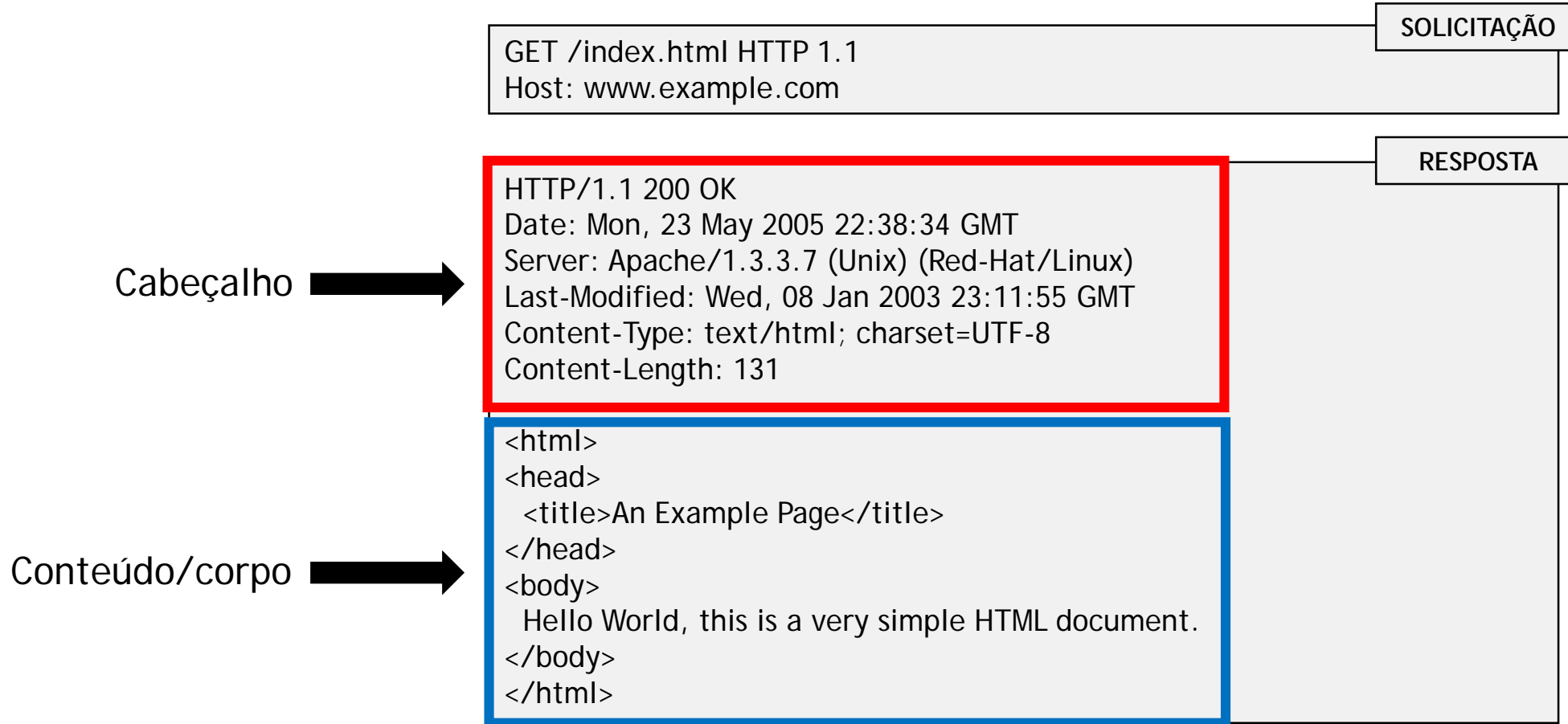
- Cabeçalhos de mensagens:

- Toda solicitação pode ser seguida de linhas adicionais contendo mais informações, chamadas de cabeçalhos de solicitação
- De forma análoga, as respostas podem ser seguidas de linhas denominadas cabeçalhos de resposta
- Alguns possíveis cabeçalhos (a lista é extensa):

Cabeçalho	Tipo	Conteúdo
User-Agent	Solicitação	Informações sobre o navegador e sua plataforma.
Host	Solicitação	Nome de domínio do servidor
Accept	Solicitação	O tipo de páginas que o cliente pode manipular.
Accept-Encoding	Solicitação	As codificações de páginas que o cliente pode manipular.
Cookie	Solicitação	Cookie previamente definido, enviado de volta ao servidor.
Set-Cookie	Resposta	Cookie para ser armazenado no cliente.
Expires	Resposta	Data e hora de quando a página deixa de ser válida.
Last-Modified	Resposta	Data e hora da última modificação da página.

Protocolo HTTP

- Exemplo de pedido HTTP



Protocolo HTTP

- Cookies

- Em algumas aplicações é necessário identificar preferências/informações do usuário para personalizar o conteúdo exibido
 - Exemplos:
 - Produtos em uma cesta de compras de um site de e-commerce
 - Exibir produtos relacionados ao gosto musical
 - Recomendação de acesso a algum serviço
- Essas preferências/informações são armazenadas na forma de cookies, que basicamente são *strings* contendo informações
 - Definido pela RFC 6265
- O cookie é armazenado no cliente para ser utilizado em novas requisições ao servidor (domínio)

Protocolo HTTP

- Cookies

- A tecnologia dos cookies tem quatro componentes:

1. Uma linha de cabeçalho de cookie na mensagem de resposta HTTP;
2. Uma linha de cabeçalho de cookie na mensagem de requisição HTTP;
3. Um arquivo de cookie mantido no sistema final do usuário e gerenciado pelo navegador do usuário;
4. Um banco de dados de apoio no site.

