URI | UNIVERSIDADE REGIONAL INTEGRADA DO ALTO URUGUAI E DAS MISSÕES

30-765

Redes de Computadores II

MSc. Fernando Schubert

CAMADA DE APLICAÇÃO - AGENDA

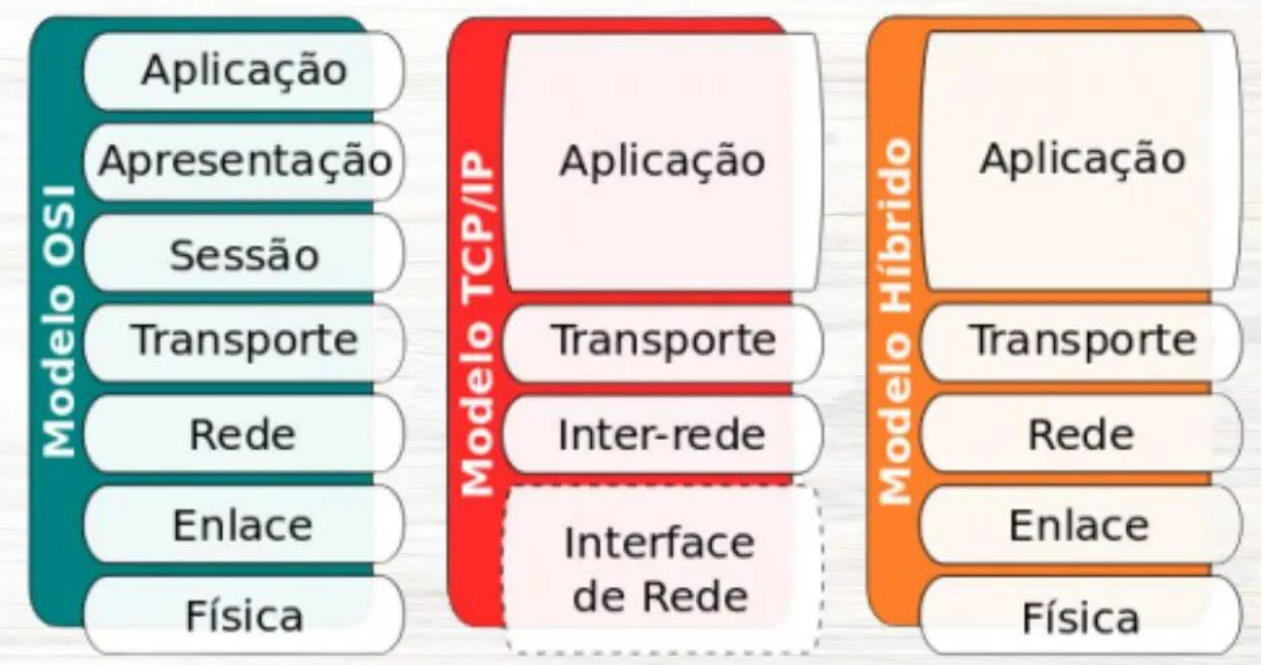
- Introdução;
- DNS (Domain Name System);
- Correio Eletrônico;
- WWW

INTRODUÇÃO

Tópicos

- A Camada de Aplicação;
- Arquitetura de aplicação de rede;
- Protocolos da camada de aplicação.

VISÃO GERAL


Modelo OSI
Aplicação
Apresentação
Sessão
Transporte
Rede
Enlace
Física
Modelo TCP/IP
Aplicação
Transporte
Inter-rede
Interface de Rede
Modelo Híbrido
Aplicação
Transporte
Rede
Enlace
Física

VISÃO GERAL

- Aplicações são a razão de ser de uma rede de computadores;
- Se não fosse possível disponibilizar aplicações úteis, não haveria necessidade de projetar protocolos de rede para suportá-las;
- Assim, a camada de aplicação oferece serviços diretamente para o usuário através de todo o arcabouço estudado até o momento.

VISÃO GERAL

- O cerne do desenvolvimento de aplicação de rede é escrever programas que rodem em sistemas finais diferentes e que se comuniquem entre si pela rede, exemplos:
- Numa aplicação Web há dois programas distintos:
  - O browser, que roda na máquina cliente;
  - E o servidor Web, que roda na máquina do servidor;

• Em um compartilhamento de arquivos P2P:

- Existem programas em cada máquina que participa do compartilhamento;
- Estes programas podem ser idênticos ou semelhantes;

• A base das aplicações é a interação cliente-servidor e a comunicação entre pares.

ARQUITETURA DE APLICAÇÃO DE REDE

• A arquitetura da aplicação determina como a aplicação é organizada nos vários sistemas finais;
• Arquitetura de aplicação é diferente da arquitetura de rede;
• Duas arquiteturas mais utilizadas em aplicações modernas:

- Cliente-servidor;
- Peer-to-Peer (P2P).

ARQUITETURA DE APLICAÇÃO DE REDE

- Arquitetura Cliente-Servidor:
  - Há um hospedeiro sempre em funcionamento: o servidor;
  - O servidor possui um endereço fixo e bem conhecido;
  - O servidor atende a requisições de muitos outros hospedeiros, os clientes;
  - Clientes não precisam estar sempre em funcionamento;
  - Clientes não se comunicam diretamente uns com os outros;
- Exemplos de aplicações:
  - Web;
  - FTP;
  - Telnet;
  - E-mail.

ARQUITETURA DE APLICAÇÃO DE REDE

- Arquitetura Peer-to-Peer (P2P):
  - A comunicação ocorre de forma direta entre pares de hospedeiros;
  - Estes pares não são de propriedade de provedores de serviços, mas são controlados por usuários finais;
  - Assim, não há garantias de que os pares estejam sempre em funcionamento;
- Exemplos de aplicações:
  - BitTorrent;
  - eMule;
  - LimeWire;
  - Skype;
  - PPLive (aplicação de IPTV)

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 e resposta;
- A sintaxe dos vários tipos de mensagens, tais como os campos da mensagem e como os campos são delimitados;
- A semântica dos campos, isto é, o significado da sua informação;
- Regras para determinar quando e como um processo envia e responde mensagens.

PROTOCOLOS DA CAMADA DE APLICAÇÃO

• Alguns protocolos da camada de aplicação estão definidos em RFCs, ou seja, são de domínio público;

- Exemplo: HTTP (protocolo da Web) – RFC 2616;

• Outros são proprietários e não estão disponíveis ao público;

- Exemplo: A maioria dos sistemas de compartilhamento de arquivos P2P;

• Um protocolo da camada de aplicação é apenas uma parte da aplicação de rede;

- Exemplo: O HTTP é o protocolo da Web, que por sua vez é composta de vários outros elementos: formato de documentos (HTML), browsers (Firefox, Chrome), servidores (Apache, Microsoft), etc.

DNS

• Na ARPANET havia apenas um arquivo que continha mapeamentos Nome / IP (hosts.txt):

- Para poucos hosts, isso pode funcionar;
- Mas para milhões de hosts conectados, não;

• Assim foi criado o sistema de nomes e domínios, o DNS (Domain Name System);

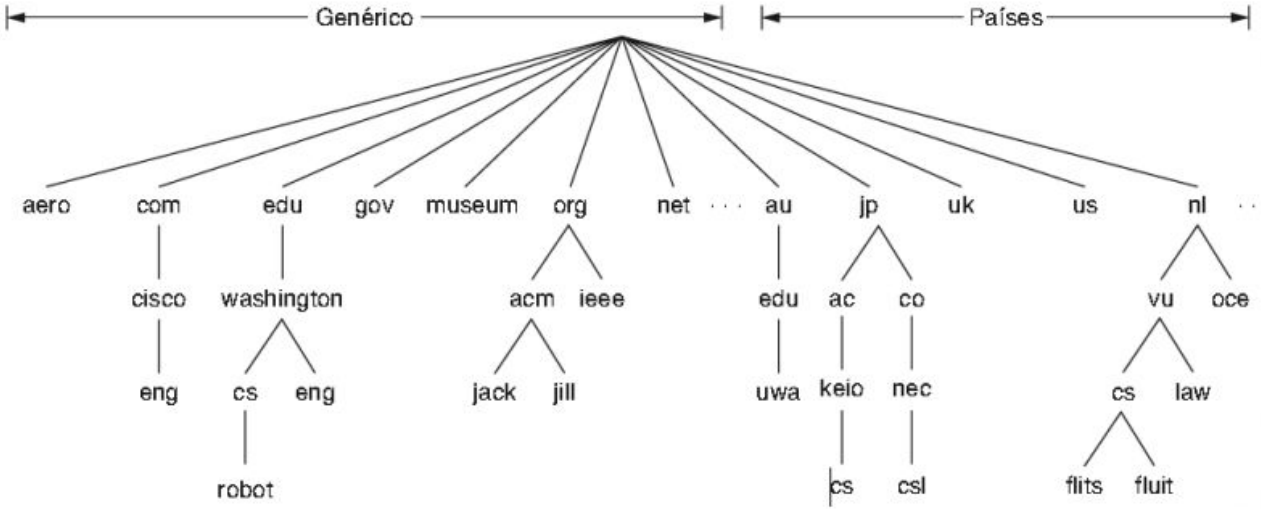
- Definido nas RFCs 1034, 1035, 2181;
- Detalhado em várias outras.

DNS

- Funcionamento básico:
  - Uma aplicação faz uma chamada a um procedimento de biblioteca denominado resolvedor, passando como parâmetro o nome a ser “resolvido”;
  - O resolvedor envia uma consulta contendo o nome para um servidor DNS local;
  - Este servidor retorna com o endereço IP ao resolvedor;
  - O resolvedor repassa o endereço retornado para a aplicação;
  - As mensagens de consulta e resposta são enviadas como mensagens UDP;
  - De posse do IP, a aplicação pode estabelecer o tipo de comunicação de sua escolha.

DNS

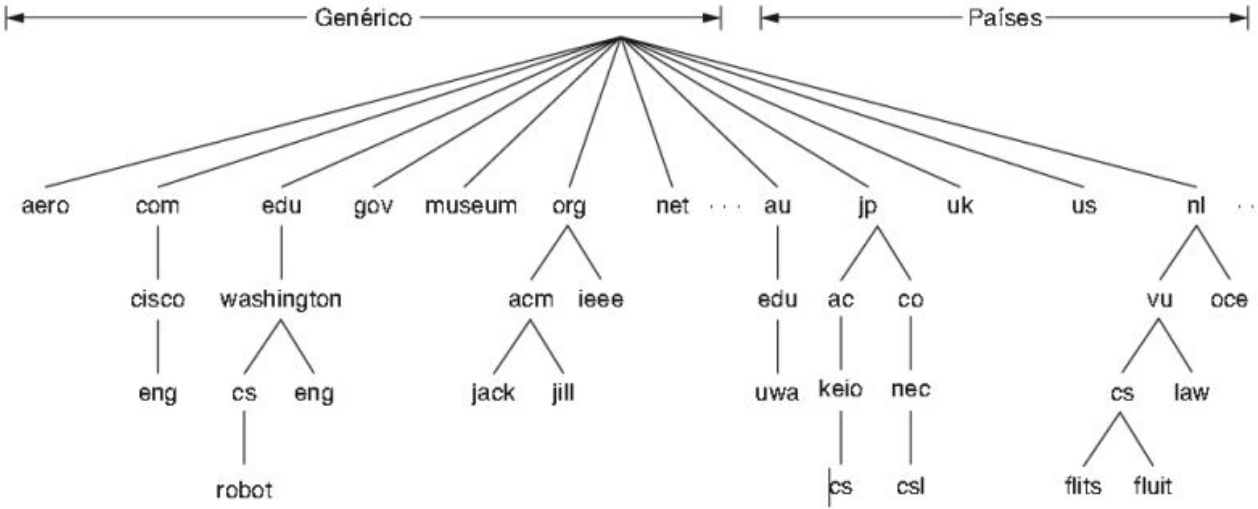
- Os **nomes** são definidos em uma estrutura hierárquica (1/3):



- Domínio de nível superior: genéricos e de países;
- Cada nível define um domínio independente e autônomo;
- Cada domínio controla seus próprios subdomínios;

DNS

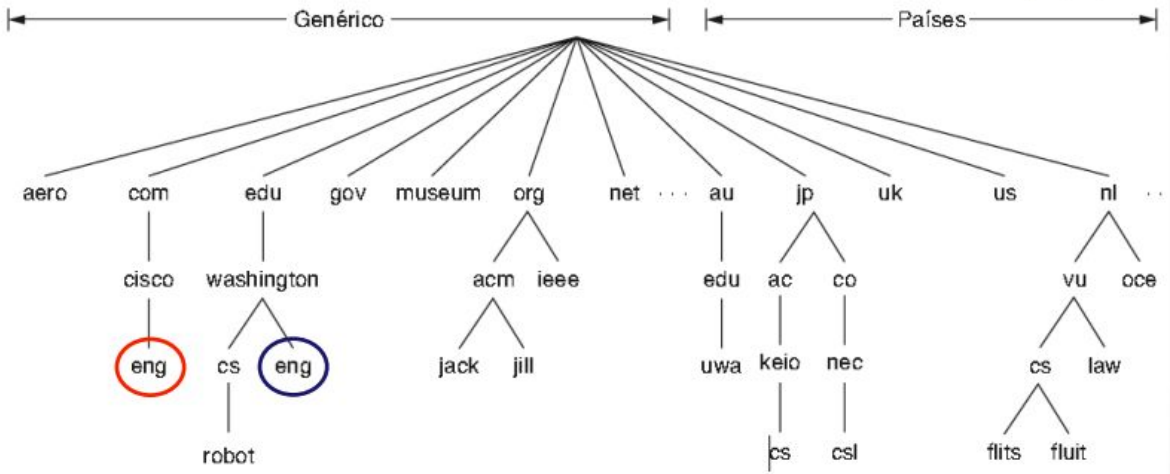
- Os nomes são definidos em uma estrutura hierárquica (2/3):



- A informação é distribuída pelos vários servidores da rede;
- Escalável (não há centralização de dados);

DNS

- Os nomes são definidos em uma estrutura hierárquica (3/3):



- O nome do domínio é ascendente e não haverá conflitos;
  - ***eng.cisco.com***, departamento de engenharia da Cisco;
  - ***eng.washington.edu***, departamento de língua inglesa da Universidade de Washington.

DNS

• Cada domínio pode ter um registro de recursos (um banco de dados DNS) associado a ele;

• Para um host comum, o registro de recursos costuma ser composto apenas pelo seu endereço IP, mas existem muitos outros tipos;

• Quando um resolvedor repassa um nome de domínio a um servidor DNS, ele recebe na verdade os registros de recursos associados a ele;

• Portanto, a principal tarefa do servidor DNS é mapear nomes de domínios em registros de recursos.

DNS

- Um registro de recursos é uma tupla de cinco campos:

1. Nome_dominio;
2. Tempo_de_vida;
3. Classe;
4. Tipo;
5. Valor.

DNS

2. **Tempo_de_vida:**
- Define um tempo para validade do registro;
- Registros mais estáveis recebem tempos maiores;

3. Classe:
- Para informações relacionadas à Internet, recebe valor **IN**;
- Existem outras classes, mas raramente são utilizadas na prática.

DNS

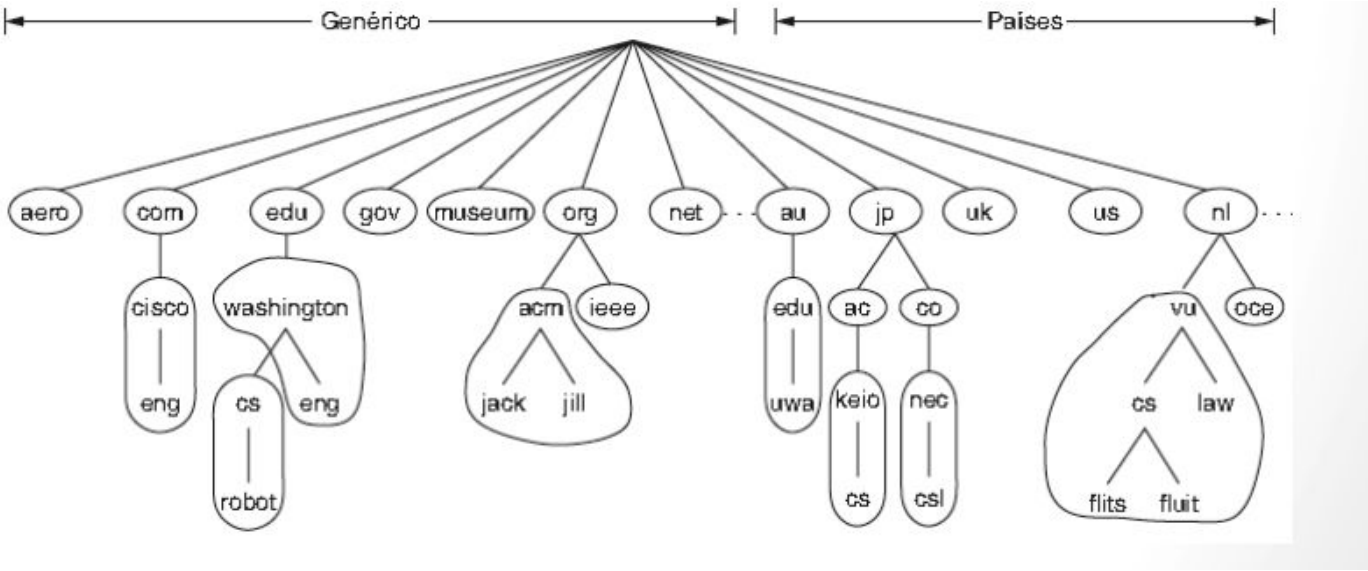
4. **Tipo:**
- Informa o tipo do registro;
5. **Valor:**
- Valor associado ao registro.

Tipo	Significado	Valor
A	Endereço IPv4.	Inteiro de 32 bits.
AAAA	Endereço IPv6.	Inteiro de 128 bits.
MX	Troca de mensagens de correio eletrônico.	Prioridade, domínio disposto a aceitar correio eletrônico.
NS	Servido de nomes.	Nome para um servidor para este domínio.
CNAME	Nome canônico	Nome de domínio.
PTR	Ponteiro.	Nome alternativo de um endereço IP.
SRV	Serviço.	*Host* que oferece o serviço.

DNS - SERVIDORES DE NOMES

• **Por que usar servidores? Apenas um não resolveria?**

- Na teoria sim, mas na prática seria impossível;
- Problemas de sobrecarga e alta dependência inviabilizam esta solução;
- Assim, o espaço de nomes do DNS é dividido em zonas não sobrepostas, exemplo:



DNS - SERVIDORES DE NOMES

• Cada zona está associada a um ou mais servidores de nomes;

• Estes servidores mantêm o banco de dados da zona;

• Normalmente, uma zona terá um servidor de nomes primário, que recebe a informação de um arquivo em seu disco, e servidores de nomes secundários, que recebem informações do servidor primário;

• Para melhoria de confiabilidade, alguns servidores de nomes podem estar localizados fora da zona.

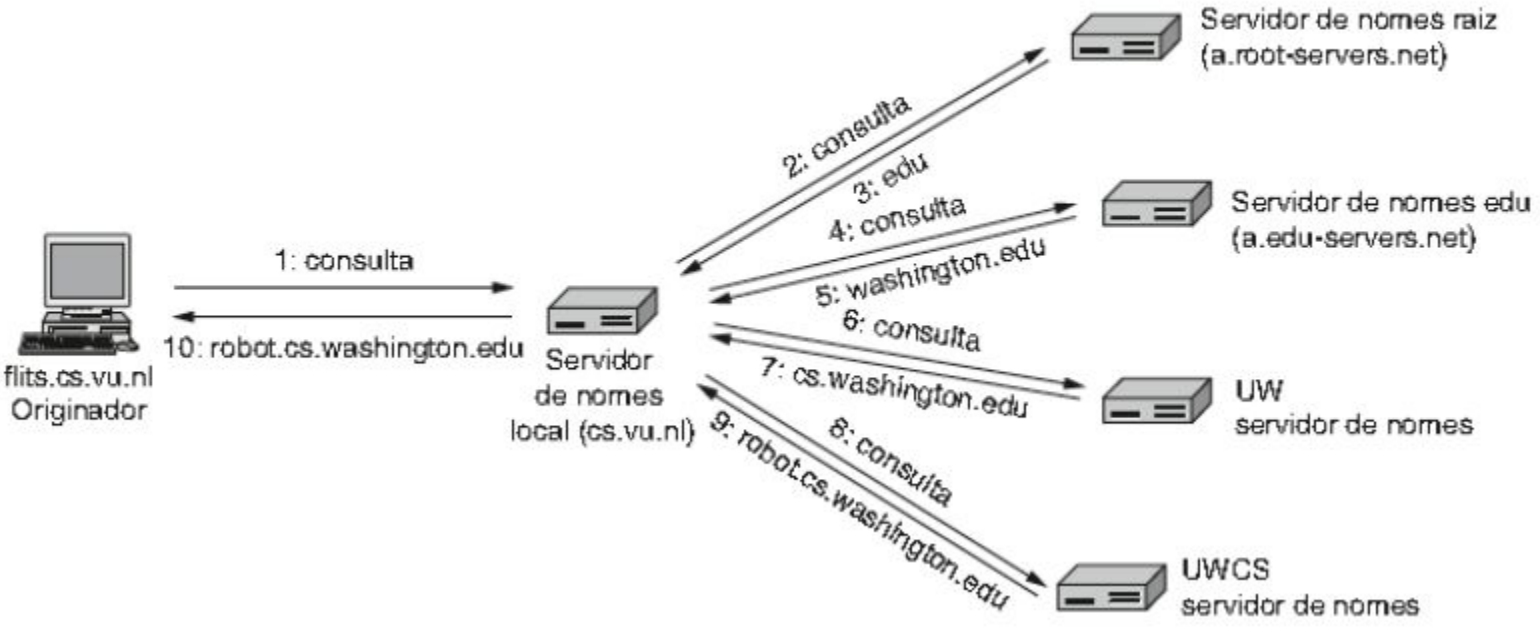
• O processo de pesquisa de um nome e localização de um endereço é chamado resolução de nomes;

• Um registro oficial é aquele que vem da autoridade oficial que controla o registro, portanto, está sempre correto;

• Um registro de cache é aquele retornado por um servidor que armazenou temporariamente a informação por questão de performance, portanto, pode estar desatualizado.

DNS - SERVIDORES DE NOMES

- Exemplo de resolução de nome:



DNS - SERVIDORES DE NOMES

• **Dois mecanismos de consulta:**

- **Consulta recursiva:** o servidor de nomes local retorna a resposta final ao originador, fazendo quantas chamadas forem necessárias a outros servidores de nome (representado na figura anterior);
- **Consulta iterativa:** o servidor de nomes apenas retorna uma resposta parcial, com a informação que lhe compete, não realiza chamadas a outros servidores para completar a resposta.

DNS - SERVIDORES DE NOMES

• **Mecanismo de caching:**

- Todas as respostas, incluindo as parciais, são armazenadas em cache para atendimento rápido a novas solicitações;
- Caso haja solicitações para um host diferente de um mesmo domínio, o caminho é encurtado fazendo uma solicitação direta ao servidor de nomes oficial, sem passar por servidores de hierarquias mais altas;
- O cache melhora a performance, mas deve haver cuidado com as possíveis alterações de informações, por isso cada registro de recurso possui um campo TTL (tempo de vida).

A WORLD WIDE WEB (WWW)

- **Tópicos**
  a. Introdução;
  b. Arquitetura;
  c. Páginas estáticas;
  d. Páginas dinâmicas;
  e. Protocolo de transferência.

INTRODUÇÃO

• A World Wide Web, ou WWW, ou Web, é uma estrutura que permite o acesso a documentos vinculados espalhados por milhões de máquinas na Internet;
• Sua enorme popularidade se deve, principalmente, a dois fatores:

- Interface gráfica colorida e de fácil utilização para iniciantes;
- Uma imensa variedade de informações sobre quase todos os assuntos imagináveis;
  • Teve seu início em 1989 no CERN (European Organization for Nuclear Research), para ajudar grandes equipes de membros espalhados por vários países a colaborar compartilhando relatórios, plantas, desenhos, fotos e outros documentos.

INTRODUÇÃO

• A proposta para uma teia de documentos interligados veio do físico Tim Berners-Lee;

• O primeiro protocolo foi apresentado em 1991, chamando a atenção de muitos pesquisadores;

• Em 1993, Marc Andressen, da Universidade de Illinois, lançou um navegador chamado Mosaic;

• Um ano depois, ele formava sua empresa, a Netscape Communications Corp., que “lutou” durante alguns anos contra a Microsoft e seu navegador, o Internet Explorer.

INTRODUÇÃO

• No decorrer das décadas de 1990 e 2000, sites e páginas Web cresceram exponencialmente, atingindo milhões de sites e bilhões de páginas;
• Algumas delas se tornaram tremendamente populares:

- Amazon, 1994, mercado de US$ 50 bilhões;
- eBay, 1995, US$ 30 bilhões;
- Google, 1998, US$ 150 bilhões;
- Facebook, 2004, US$ 15 bilhões;

  • Em 1994, o CERN e o MIT criaram o W3C (World Wide Web Consortium), www.w3.org ou www.w3c.br, uma organização responsável por organizar e padronizar o desenvolvimento Web.

ARQUITETURA

• O lado cliente (1/3):

• Para identificar uma página é utilizada a URL (Uniform Resource Locator), que é definida por três partes:

- O protocolo (também conhecido como esquema);

- O nome DNS da máquina onde está localizada;

- O caminho, que especifica exclusivamente onde está a página.

ARQUITETURA

• O lado cliente (2/3):

• Exemplo de URL: http://www.cs.washington.edu/index.html;

• O navegador realiza uma série de tarefas para exibir a URL:

1. Obtém o IP do servidor solicitando ao DNS o endereço de www.cs.washington.edu;

2. Estabelece uma conexão TCP com o servidor na porta 80;

3. Solicita a página index.html usando um comando HTTP;

4. Caso a página inclua links para outros recursos para exibição (URLs), buscará estes recursos da mesma maneira;

5. Exibe a página;

6. Encerra a conexão após um tempo.

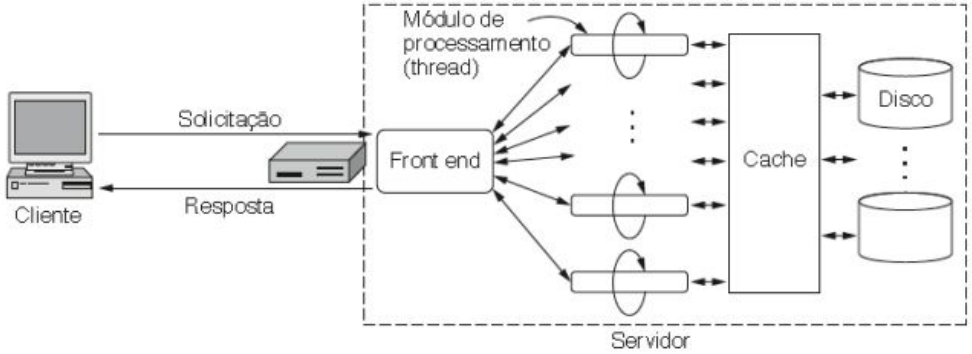
ARQUITETURA

- **O lado cliente** (3/3):
  - Diferentes protocolos que podem ser usados em uma URL:

Nome	Usado para	Exemplo
http	Hipertexto (HTML).	http://www.decom.ufop.br/reinaldo
https	Hipertexto com segurança.	https://www.bank.com/accounts/
ftp	FTP.	ftp://ftp.cs.vu.nl/pub/minix/README
file	Arquivo local.	file://usr/suzana/prog.c
mailto	Envio de e-mail.	mailto:fulano@acm.org
rtsp	*Streaming* de mídia.	rtsp://youtube.com/montypython.mpg

ARQUITETURA

- **O lado servidor**:
  - Arquitetura de um servidor Web:



  - Tarefas:
    - Aceitar uma conexão TCP de um cliente (um navegador);
    - Obter o caminho até a página (ou programa);
    - Obter o arquivo (ou gerar o conteúdo dinâmico);
    - Enviar o conteúdo ao cliente;
    - Encerrar a conexão.

ARQUITETURA

- Cookies:
  - Em algumas aplicações é necessário identificar certas informações do usuário para personalizar conteúdo;
  - Exemplo: produtos em uma cesta de compras de um site comercial;
  - Este problema é resolvido com um mecanismo chamado cookie, que trata-se apenas de uma string pequena contendo algumas informações;
  - Quando o cliente solicita uma página, o servidor pode fornecer informações adicionais, na forma de um cookie junto com a página retornada.
  - O cookie é armazenado no cliente para ser utilizado em novas requisições ao mesmo servidor;
  - Um cookie pode conter até cinco campos
  - Cookies estão envolvidos com algumas questões de privacidade e segurança, muitos usuários preferem desativar o seu uso no navegador.

PÁGINAS ESTÁTICAS

• As páginas estáticas são a forma mais simples de página web, sendo armazenadas em um servidor, que as retorna para serem diretamente exibidas no navegador quando solicitadas.

• Normalmente, as páginas estáticas são desenvolvidas em linguagem HTML (HyperText Markup Language):

• HTML é uma linguagem de marcação que utiliza tags para determinar a estrutura e formatação do conteúdo.

• Exemplos de tags:

  • `<b>... </b>` para negrito.

  • `<img>...</img>` para inserir imagens.

  • `<body>...</body>` para determinar o conteúdo da página.

  • `<a>...</a>` para definir hiperlinks.

PÁGINAS ESTÁTICAS

- A HTML já passou por várias evoluções:

Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hiperlinks	X	X	X	X	X
Imagens e listas	X	X	X	X	X
Mapas e imagens ativas		X	X	X	X
Formulários		X	X	X	X
Equações			X	X	X
Barras de ferramentas			X	X	X
Tabelas			X	X	X
Recursos de acessibilidade				X	X
Objetos inseridos				X	X
Folhas de estilo				X	X
Scripting				X	X
Vídeo e áudio					X
Gráficos e vetores em linha					X
Representação XML					X
Threads em segundo plano					X
Armazenamento pelo navegador					X
Tela de desenho					X

PÁGINAS DINÂMICAS

• O modelo de páginas estáticas foi útil nos primeiros momentos da Web, quando um grande volume de informação foi inserido.

• Atualmente, grande parte do uso da Web está voltado para aplicações e serviços:

• Comércio eletrônico;

• Pesquisa em catálogos de bibliotecas ou na própria Web;

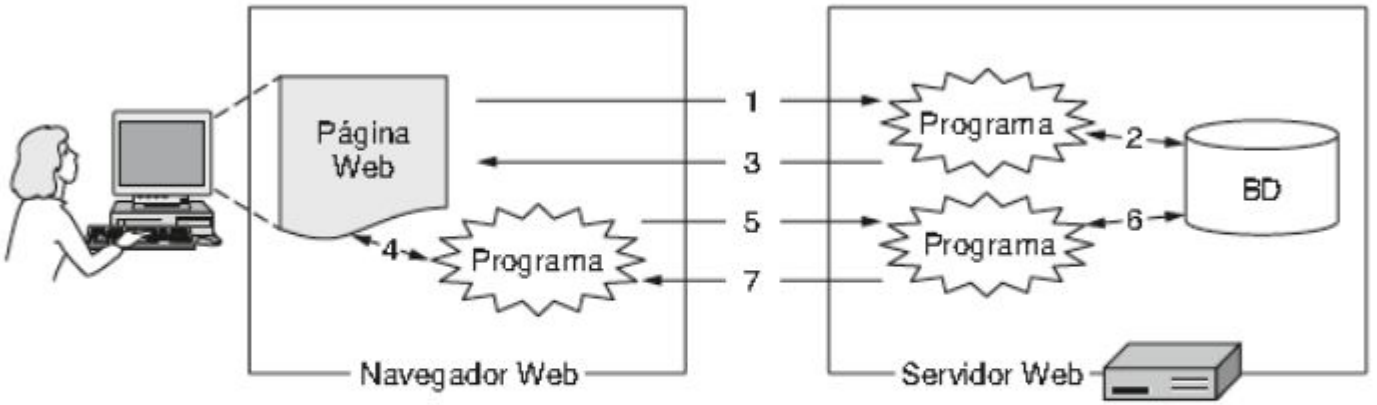
• Leitura e envio de e-mails;

• Colaboração e redes sociais;

• Neste novo modelo, as páginas são construídas dinamicamente, com base em dados fornecidos pelos usuários.

PÁGINAS DINÂMICAS

- Geração de páginas dinâmicas:



- Pode ocorrer no lado do cliente (navegador);
- Ou no lado servidor.

PROTOCOLO DE TRANSFERÊNCIA

• O protocolo utilizado para transportar toda a informação entre os servidores Web e os clientes Web é o HTTP (HyperText Transfer Protocol), especificado na RFC 2116;

• HTTP é um protocolo simples:

• Funciona no estilo solicitação-resposta, rodando sobre o TCP;

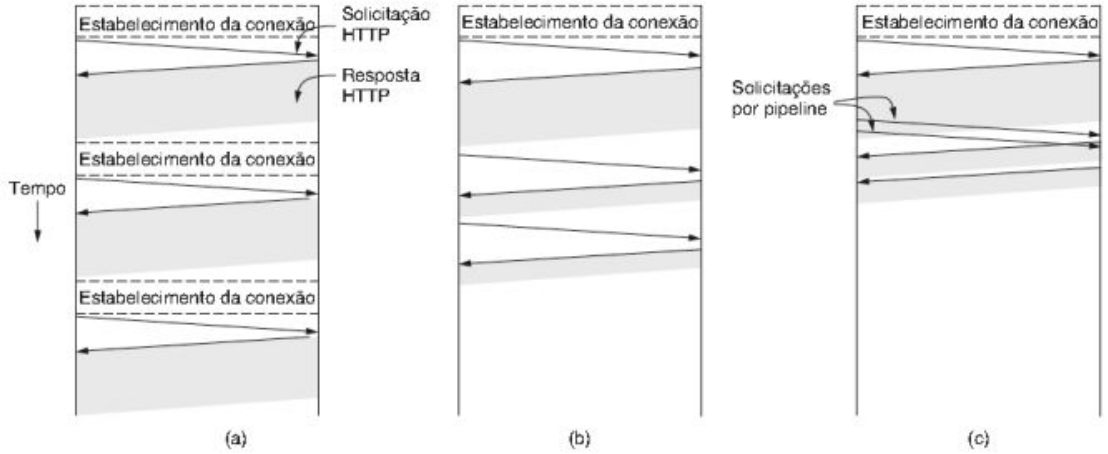
• Especifica quais mensagens os clientes podem enviar para os servidores e quais respostas recebem de volta;

• Assim como no SMTP, os cabeçalhos são dados em ASCII e o conteúdo é dado em formato do tipo MIME;

• Parte do sucesso da Web é creditado à simplicidade do HTTP, que facilitou o seu desenvolvimento e implantação.

PROTOCOLO DE TRANSFERÊNCIA

- Conexões:
  - Utiliza protocolo TCP na porta 80;
  - Conexões são persistentes;
  - Os dados podem ser requisitados em pipeline;



(a) múltiplas conexões e solicitações sequenciais.
(b) Conexão persistente e solicitações sequenciais.

PROTOCOLO DE TRANSFERÊNCIA

- **Métodos:**
  - O HTTP aceita operações chamadas ***métodos***;
  - 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 Web.
HEAD	Lê um cabeçalho de página Web. Pode ser usado para indexação ou testar a validade de um URL.
POST	Acrescenta algo a uma página Web. Usado para envio de dados de formulários para o servidor.
PUT	Armazena uma página Web. É o contrário de GET, possibilita criar uma coleção de páginas Web em um servidor remoto.
DELETE	Remove a página Web.
TRACE	Ecoa a solicitação recebida. Serve para depuração, envia o servidor a enviar de volta a solicitação para saber qual solicitação o servidor recebeu de fato.
CONNECT	Conecta através de um *proxy*.
OPTIONS	Consulta opções para uma página. Possibilita descobrir quais são os métodos e cabeçalhos que podem ser usados com uma página.

PROTOCOLO DE TRANSFERÊNCIA

- **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.

naveg
ador

PROTOCOLO DE TRANSFERÊNCIA

- **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.
Accept	Solicitação	O tipo de páginas que o cliente pode manipular.
Accept-Charset	Solicitação	Os conjuntos de caracteres aceitáveis para o cliente.
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.
Cache-Control	Ambos	Diretivas para o modo de tratar caches.

navegador

PROTOCOLO DE TRANSFERÊNCIA

- **Caching**:
  - Normalmente os usuários retornam às páginas visitadas com frequência;
  - Muitos recursos utilizados nunca mudam, ou mudam pouco, seria um desperdício capturar todos eles toda vez que uma página fosse novamente solicitada;
  - O HTTP usa duas estratégias para enfrentar este problema: