PMR3412 - Redes Industriais - 2021

Aula 06 - Aplicações TCP/IP: Domain Name System e HTTP/1.1

Prof. Dr. André Kubagawa Sato
Prof. Dr. Marcos de Sales Guerra Tsuzuki

23 de Setembro de 2021

PMR-EPUSP

Revisão - Aplicações TCP/IP

- Até agora, vimos as seguintes aplicações TCP/IP:
  - Telnet: controle remoto, desenvolvido em 1969.
  - FTP: transferência de arquivo, desenvolvido em 1971.
  - SMTP: mensagem eletrônica (email), desenvolvido em 1980.
- Porém, a Internet só se estabeleceu no final da década de 80 / início de 90.

Domain Name System (DNS)

DNS - Introdução (antes do DNS)

- As camadas inferiores do modelo TCP/IP permitem acesso a outros hosts em uma rede local ou internet utilizando o endereço IP, que possui a forma `192.168.1.200`.
- Em rede pequena, é possível memorizar ou anotar os endereços de cada máquina. Porém, esta solução não é “escalável”, i.e. logo se torna inviável com o crescimento no número de hosts.
- Sendo assim, a próxima evolução foi utilizar nomes simbólicos. Assim, o comando:

```
ping 192.168.1.110
```

- pode ser substituído por:

```
ping myHost
```

- Os mapeamentos nome-end. IP eram armazenados em um arquivo `HOSTS.TXT`. Essa solução logo se mostrou pouco prática com o crescimento explosivo do número de hosts.

DNS - Introdução

- A solução foi adotar o Domain Name System (DNS).
- O DNS é o mecanismo atual que permite traduzir um nome simbólico para um endereço IP. Assim, ao invés de escrever no seu *browser*:

`172.217.162.110`

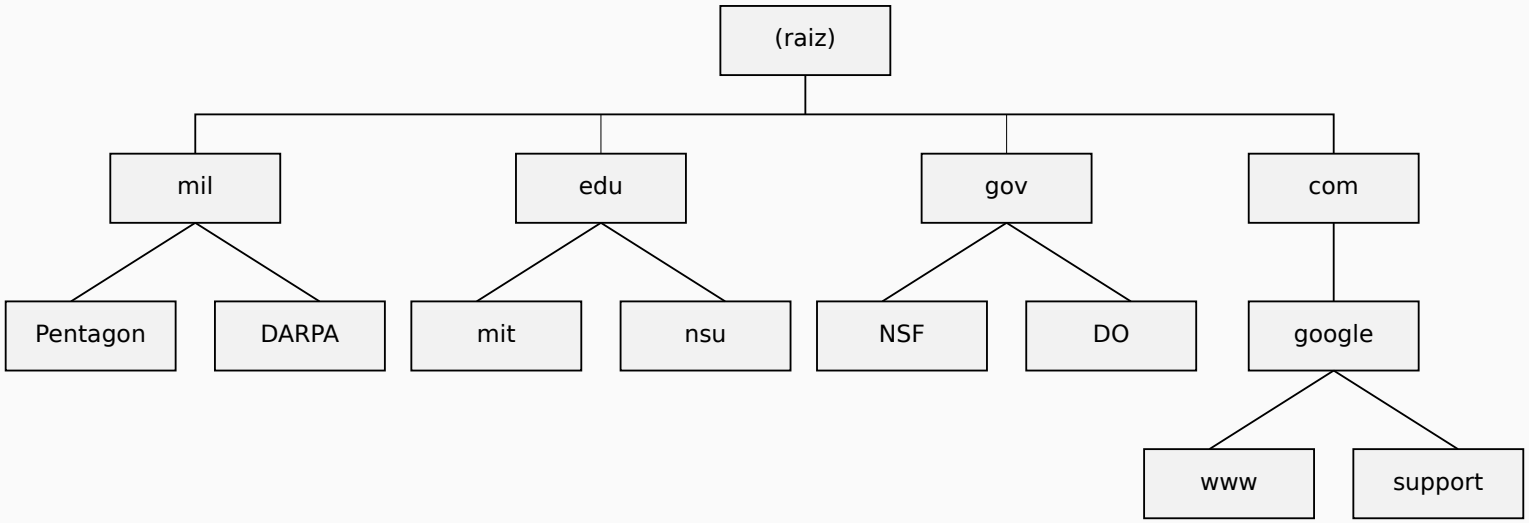
- Você geralmente escreve:

`www.google.com`

- O que permite que o DNS consiga resolver nomes para toda a internet é a possibilidade de um programa executando no host realizar o mapeamento para qualquer outro host na Internet.
- Assim, não é necessário que nenhum host possua o mapeamento completo de todos os pares de nome-end. IP da Internet.

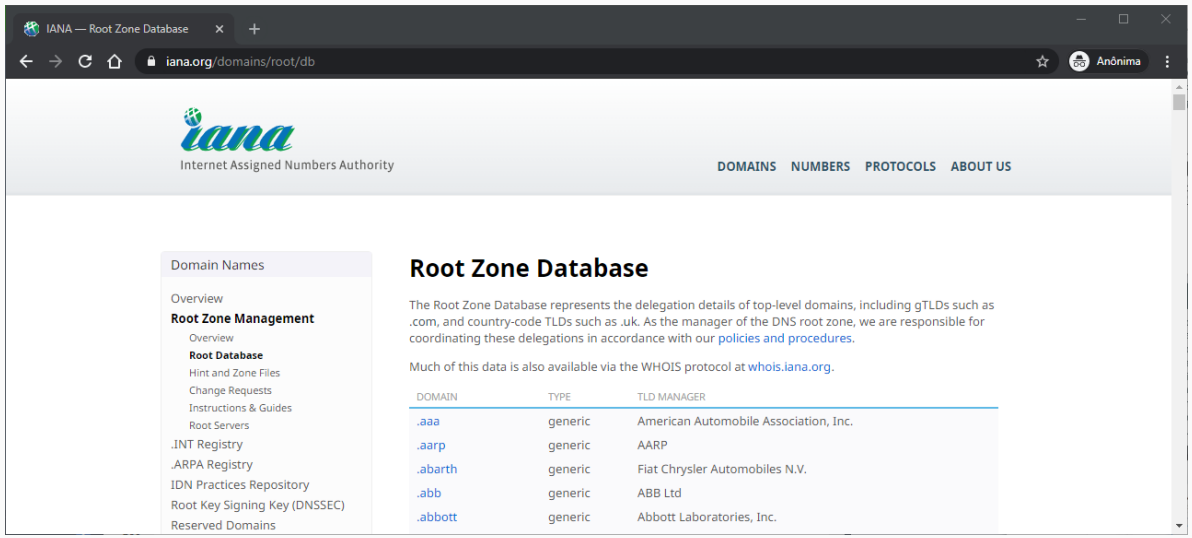
DNS - O Namespace Hierárquico

- No exemplo `support.google.com`, o host *support* existe dentro do subdomínio *google.com*. Por sua vez, *google.com* é um subdomínio de *com*.
- No DNS, é comum trabalhar apenas com parte do domínio (p. ex: *google.com*). Para demonstrar que o nome do domínio está completo, deve se adicionar um ponto ao final dele. Por exemplo: *support.google.com.*.
- Este é conhecido como um *fully qualified domain name* (FQDN).



DNS - Os Domínios de nível superior (TLD)

- Os domínios logo abaixo da raiz (*Top Level Domains* ou TLD) são controlados pela IANA e podem ser conferidos no endereço `https://www.iana.org/domains/root/db`.
- Existem domínios genéricos como `com` (comercial), `edu` (educacional), `org` (organizações não comercias), `net` (infraestrutura de rede).
- Também existem domínios de país com dois caracteres como `br` (Brasil), `uk` (Reino Unido), `de` (Alemanha), entre outros.
- Em geral, os países possuem domínios de segundo nível que replicam os domínios genéricos. Por exemplo, `.co.uk` e .ac.uk correspondem aos domínios genéricos `.com` e .edu.

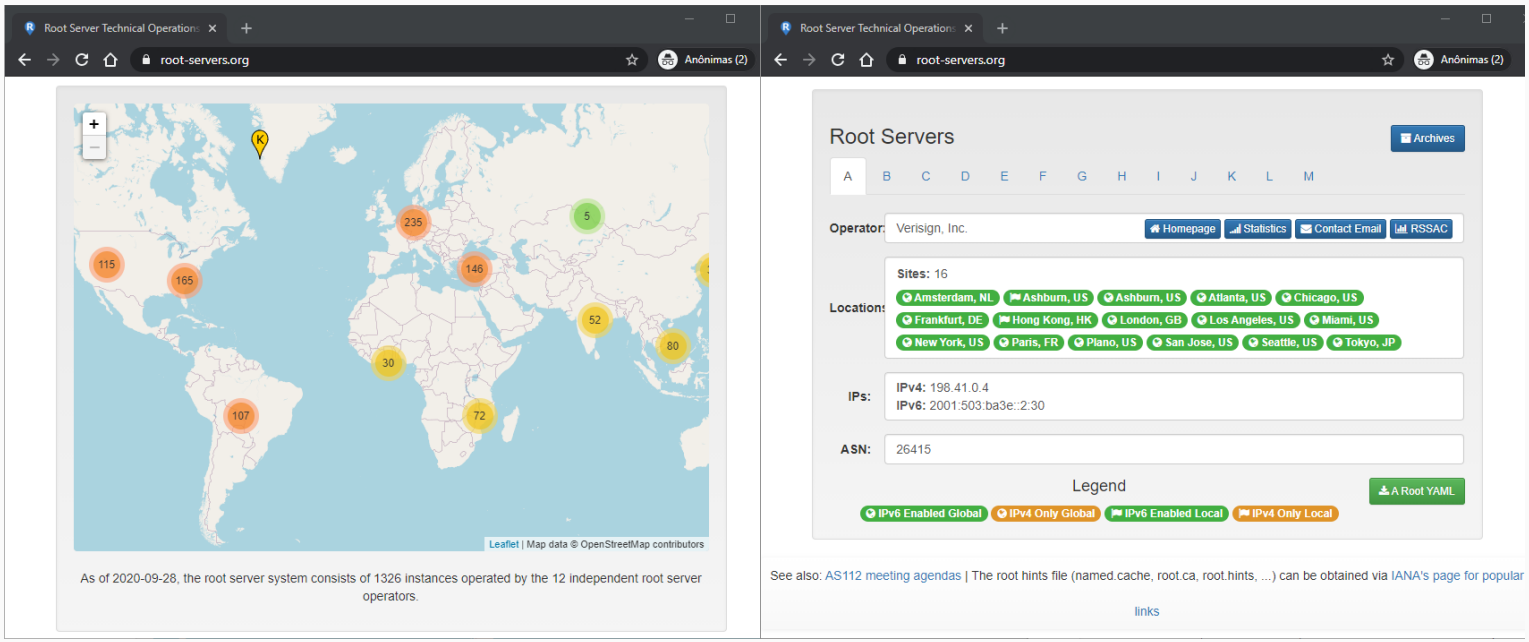


DNS - O Namespace Distribuído

- Nomes simbólicos são agrupados em zonas de autoridade (ou simplesmente zonas).
- Em cada zona, um ou mais hosts são responsáveis por manter o banco de dados e realizar a função de servidores DNS.
- Cada zona contém uma subárvore da árvore hierárquica e os nomes de uma zona são administrados independentemente de nomes em outra zona.
- Uma exceção é feita para os servidores raiz, cuja autoridade é distribuída para um conjunto de servidores.

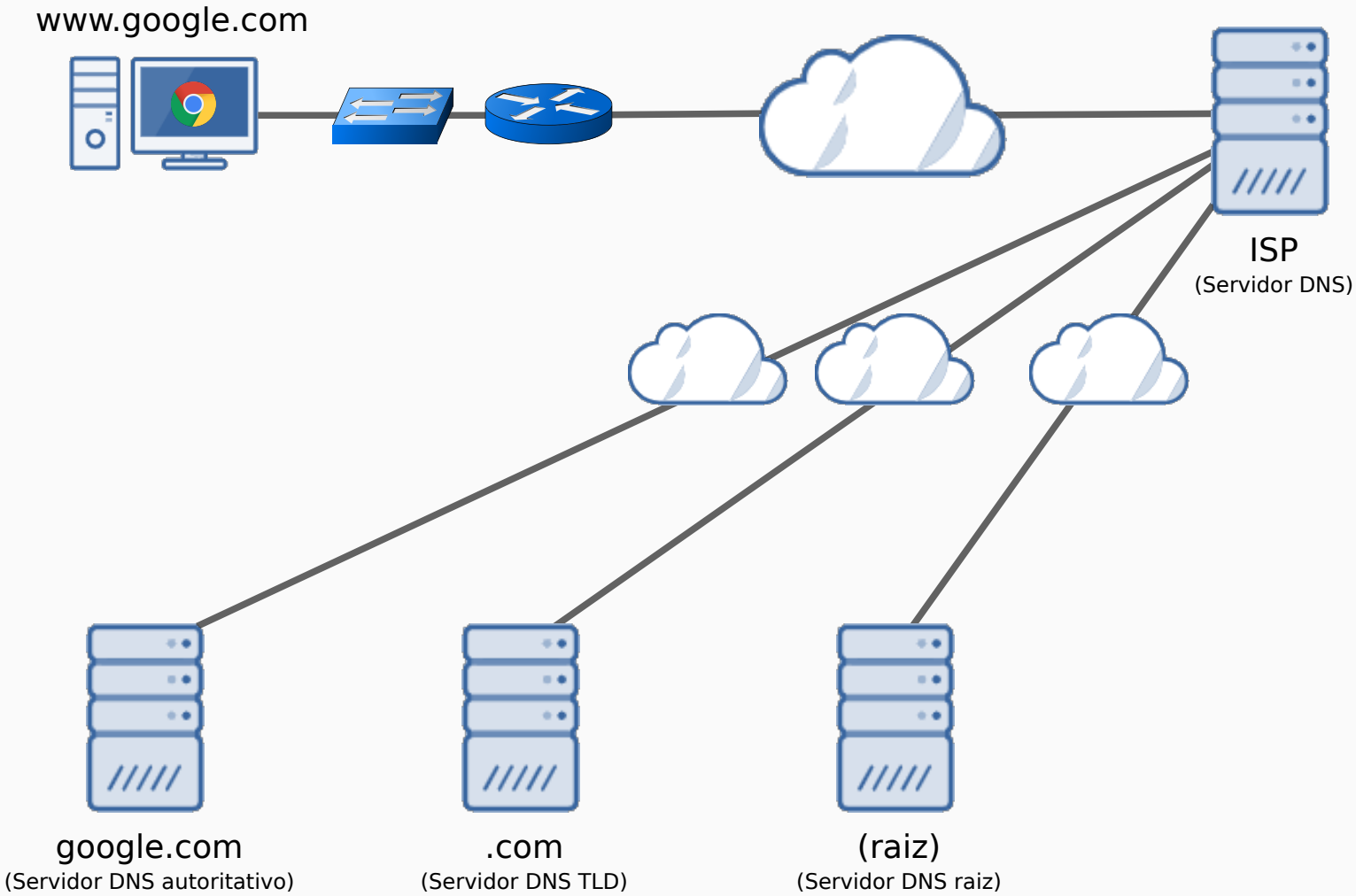
DNS - Os servidores DNS raiz

- Os servidores raiz são operados por 12 organizações, que possuem diversos servidores em cada continente.
- As organizações e os endereços IP podem ser consultados em `https://root-servers.org/`.
- Cada cliente tem programado os endereços estaticamente, uma vez que não podem ser obtidos de ouro modo.



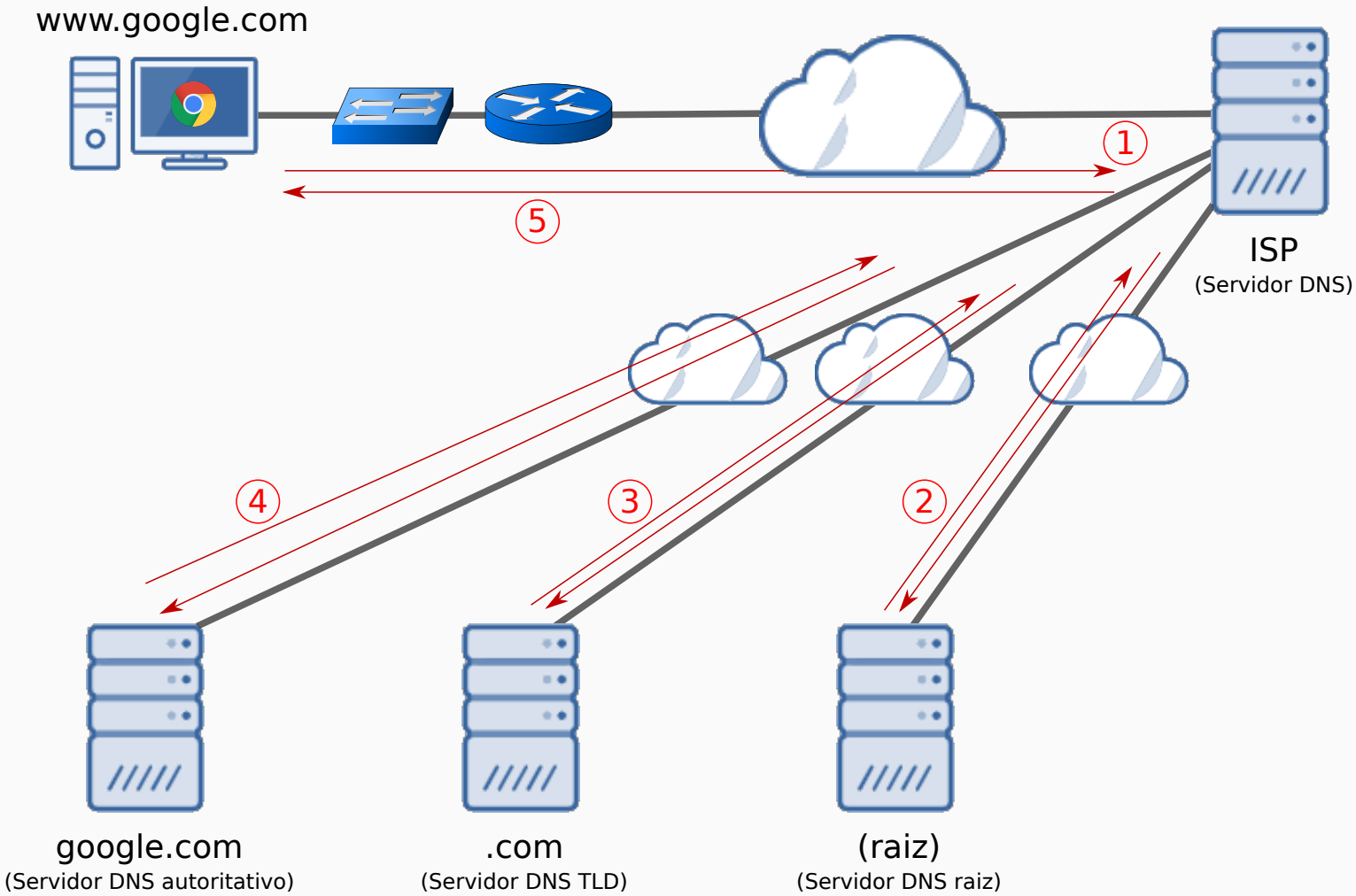
DNS - Exemplo de resolução de nome

- Exemplo: resolver o nome `www.google.com`.



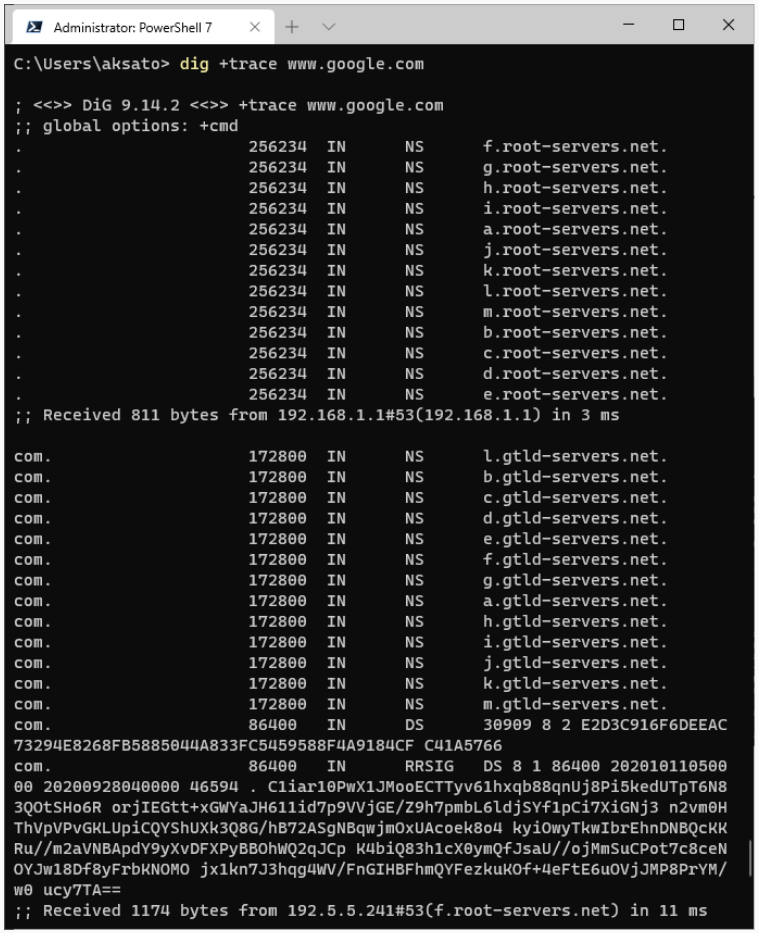
DNS - Exemplo de resolução de nome

- Exemplo: resolver o nome `www.google.com`.



DNS - Exemplo de resolução de nome

- Realizando o trace com a ferramenta dig (`https://www.isc.org/download/`)



DNS - Operação do Domain Name Resolver e Server

- O DNS segue o modelo cliente servidor: o cliente é o DNS resolver e servidor é o DNS server.

Domain Name Resolver

- As queries do resolver podem ser de dois tipos: iterativas ou recursivas.
- Quando um servidor recebe uma requisição que não consegue responder completamente:
  - query recursiva: servidor gera novas queries para determinar a resposta e encaminhar para o resolver.
  - query iterativa: servidor retorna as informações incompletas e lista de servidores adicionais para que o usuário possa concluir a query
- As respostas podem ser de dois tipos: autoritativas (quando o servidor possui autoridade) ou não autoritativas.

Domain Name Server

- Possuem autoridade sobre zero ou mais zonas
- Podem ser de três tipos: primário (possui autoridade e armazena informação em disco), secundário (possui autoridade, mas obtém informação do servidor primário) ou caching-only (não possui autoridade).

DNS - DNS resource records

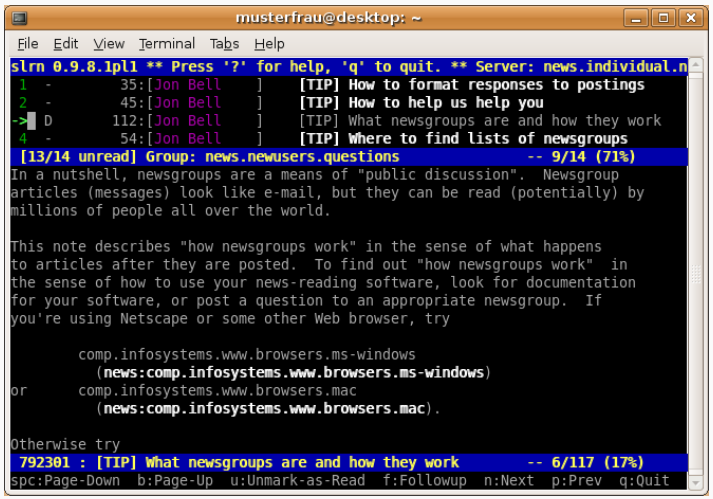
- Até agora, assumimos que o banco de dados do servidor DNS armazena apenas o mapeamento entre nomes e endereços.
- Na realidade, o banco de dados distribuído do DNS é composto de entradas de *resource records*, que mapeiam o nome do domínio para um objeto de rede.
- Uma zona é um agrupamento de *resource records*, iniciando pela entrada do *Start of Authority (SOA)*, contendo o nome do domínio.
- Abaixo são exibidos alguns tipos de *resource records*:

Tipo	Significado	RFC
A	Um endereço do host	1035
NS	Um nome de servidor autoritativo	1035
SOA	Marca o início da zona de autoridade	1035
MX	Mail-Exchange	1035
AAAA	Um endereço do host em IPv6	3596

Hypertext Transfer Protocol (HTTP)

HTTP - Antes do HTTP

- Antes do HTTP, era pré Internet, a comunicação ocorria em uma arquitetura *dial-up* (discado).
- Exemplos: Usenet e *bulletin board system* (BBS), que funcionavam como um fórum Web.
- Principal diferença entre eles: as BBSs tinham um servidor central e o Usenet era distribuído.



Usenet

BBS

HTTP - Histórico de Versões (I)

- HTTP/0.9: Primeira versão do HTTP, extremamente simples. Cada requisição consistia em uma linha, que utilizava o método GET para especificar o arquivo HTML a ser obtido:

```
GET /mypage.html
```

  cuja resposta era o próprio arquivo HTML.
- HTML/1.0: primeira versão especificada (RFC 1945), introduziu o conceito de MIME types para tratar diferentes tipos de recursos. Além disso, adicionou cabeçalhos e códigos de resposta:

```
GET /mypage.html HTTP/1.0
User-Agent: NCSA_Mosaic/2.0 (Windows 3.1)

200 OK
Date: Tue, 15 Nov 1994 08:12:31 GMT
Server: CERN/3.0 libwww/2.17
Content-Type: text/html
<HTML>
A page with an image
  <IMG SRC="/myimage.gif">
</HTML>
```

HTTP - Histórico de Versões (II)

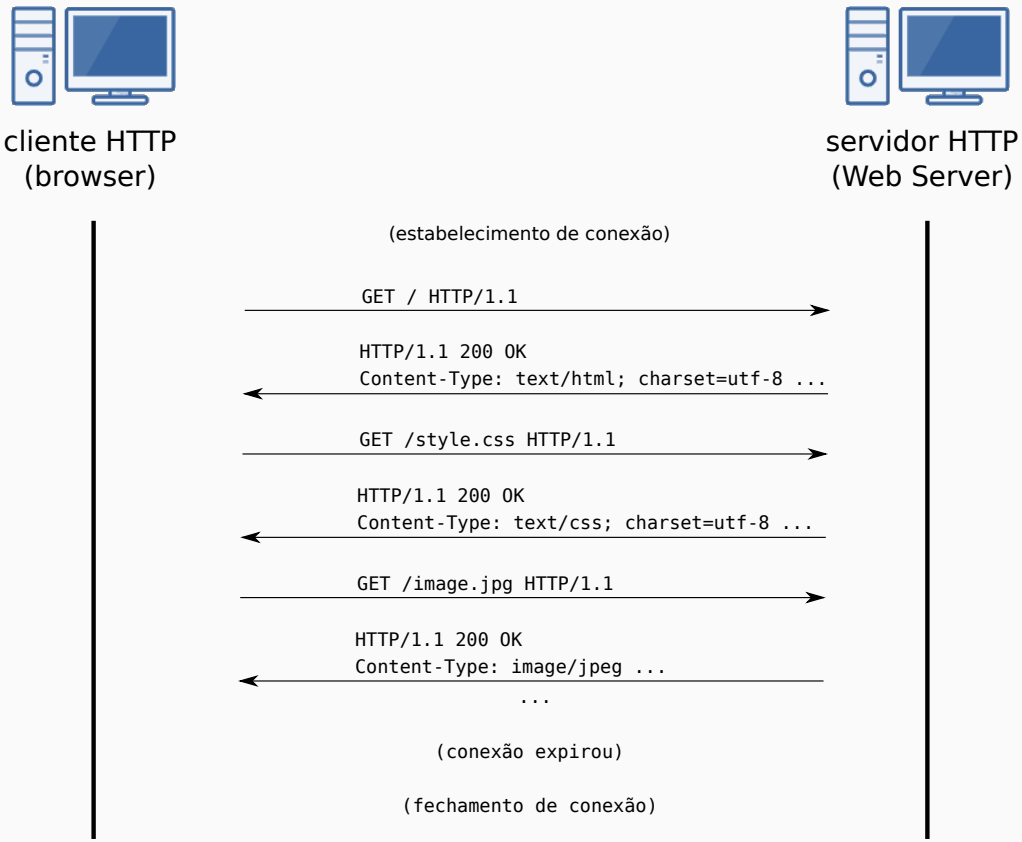
- HTTP/1.1: O protocolo HTML/1.0 demonstrava dois problemas principais: falta de uma padronização e impossibilidade de reutilizar a conexão para obter múltiplos arquivos. O protocolo HTTP/1.1, que foi a primeira versão padronizada (RFC 2616 em 1997), solucionou estes problemas, além de introduzir controle de cache e negociação de conteúdo.
- HTTP/2: oficialmente padronizado em 2015, visou melhor desempenho para o protocolo HTTP, cujo uso se tornou muito mais complexo. A principal diferença em relação à versão 1.1 é a transmissão em binário, que pode ser multiplexada, gerando requisições em paralelo, que podem ser processadas na mesma conexão.
- HTTP/3: ainda não lançada, atualmente está em versão *draft*. A principal proposta é a substituição do TCP pelo protocolo QUIC, desenvolvido pela Google, que é adaptado para multiplexação das requisições.

HTTP - Características

- O HTTP foi desenvolvido para possibilitar a transferência de arquivos HTML, que permite a criação de arquivos hipertexto com imagens, áudio e vídeo.
- No entanto, acabou se tornando o protocolo fundamental para troca de recursos na Internet.
- Assim como outros protocolos estudados, adota o modelo cliente-servidor com requisições e respostas. Utiliza o protocolo TCP, com a porta padrão 80.
- A aplicação cliente é geralmente um *Web browser*, que reconstrói o documento completo, juntando as partes, como arquivos de imagens, scripts e estilos.
- O protocolo HTTP é considerado *stateless*, uma vez que não mantém informações entre requisições/conexões. No entanto, é possível estabelecer sessões utilizando *cookies*.
- Muitas vezes a conexão entre o cliente e o seridor não é direto. Existem intermediários, chamados *proxies*, que podem ser transparentes ou não e possuem função importante de *caching*.

HTTP - Fluxo de uma Sessão HTTP

1. O cliente inicia a conexão e envia a requisição.
2. O servidor então processa a requisição e envia a resposta para o cliente.
3. O processo se repete até a conexão expirar. Deste modo, é possível obter vários arquivos em uma mesma requisição (implementada na versão HTTP/1.1).



HTTP - Métodos

- Cada requisição deve incluir um método para ser aplicado ao recurso.
- Os métodos podem ser
  - seguros, quando não geram efeitos colaterais, ou seja, quando apenas recuperam o conteúdo;
  - ou idempotentes, quando ações repetidas têm o mesmo resultado que uma única ação.
- Os métodos definidos na especificação são:

Método	Seguro	Idem.	Descrição
GET	X	X	Recupera o recurso.
POST			Aceita a entidade como subordinado ao recurso (definido pelo servidor).
PUT		X	Similar ao PUT, mas identifica o recurso na entidade.
DELETE		X	Apaga o recurso.
OPTIONS			Recupera informações sobre as opções de comunicação disponível.
HEAD	X	X	Similar ao GET, mas não recupera o corpo da resposta.
TRACE			Exibe como a mensagem foi recuperado (para testes).
CONNECT			Conecta a um proxy para abrir um tunel.

HTTP - Códigos de Status

- Principais códigos de resposta:

Informacional	Sucesso	Redirecionamento
100 Continue	200 OK	300 Multiple Choices
101 Switching Protocols	201 Created	301 Moved Permanently
	202 Accepted	302 Moved Temporarily
	203 Non-Authoritative Information	303 See Other
	204 No Content	304 Not Modified
	205 Reset Content	305 Use Proxy
	206 Partial Content	

Erro Cliente		Erro Servidor
400 Bad Request	408 Request Timeout	500 Internal Server Error
401 Unauthorized	409 Conflict	501 Not Implemented
402 Payment Required	410 Gone	502 Bad Gateway
403 Forbidden	411 Length Required	503 Service Unavailable
404 Not Found	412 Precondition Failed	504 Gateway Timeout
405 Method Not Allowed	413 Request Entity Too Large	505 HTTP Version Not Supported
406 Not Acceptable	414 Request-URI Too Long	
407 Proxy Authentication Required	415 Unsupported Media Type	

HTTP - Uniform Resource Locator (URL)

- Do ponto de vista do HTTP, um URI (Uniform Resource Identifiers) é uma *string* formatada que identifica um recurso.
- O esquema `http` é utilizado para localizar recursos de rede no protocolo HTTP, gerando URLs (Uniform Resource Locators) do tipo:

  `http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#Somewhere`
- que pode ser dividido em:
  - *Scheme*: `http`
  - *Authority*: `www.example.com`
  - *Port*: `80`
  - *Path*: `path/to/myfile.html`
  - *Query*: `key1=value1&key2=value2`
  - *Fragment*: `Somewhere`

HTTP - Formato da Mensagem

- Exemplo de formato de mensagem de requisição:

```
GET / HTTP/1.1          (start line)
Host: localhost:8000    (Message Headers)
```

- Exemplo de formato de mensagem de resposta:

```
HTTP/1.1 200 OK                                          <- start line
Date: Mon, 19 Oct 2020 17:23:33 GMT                      \
Server: SimpleHTTP/0.6 Python/3.8.3                      | Message
Last-Modified: Mon, 19 Oct 2020 17:16:40 GMT             | Headers
Content-Length: 330                                      |
Content-Type: text/html                                  /
                                                         <- linha vazia
<!DOCTYPE html>                                          \
<html lang="en">                                         |
<head>                                                   |
    <meta charset="UTF-8">                               |
    <meta name="viewport" content="width=device-width,   | Body
       initial-scale=1.0">                               |
    <title>Hello World Page</title>                      |
</head>                                                  |
<body>                                                   |
    <h1>Hello World</h1>                                 |
</body>                                                  |
</html>                                                  /
```

Referências

Referências - Aula 06

- Para o curso: livro da IBM “TCP/IP Tutorial and technical overview” (disponível em `https://www.redbooks.ibm.com/redbooks/pdfs/gg243376.pdf`.
- Para esta aula: seções 12.1, 12.2, 16.3 e RFC 2616 (HTTP/1.1).
- MDN Web Docs sobre HTTP: `https://developer.mozilla.org/en-US/docs/Web/HTTP`

The End!