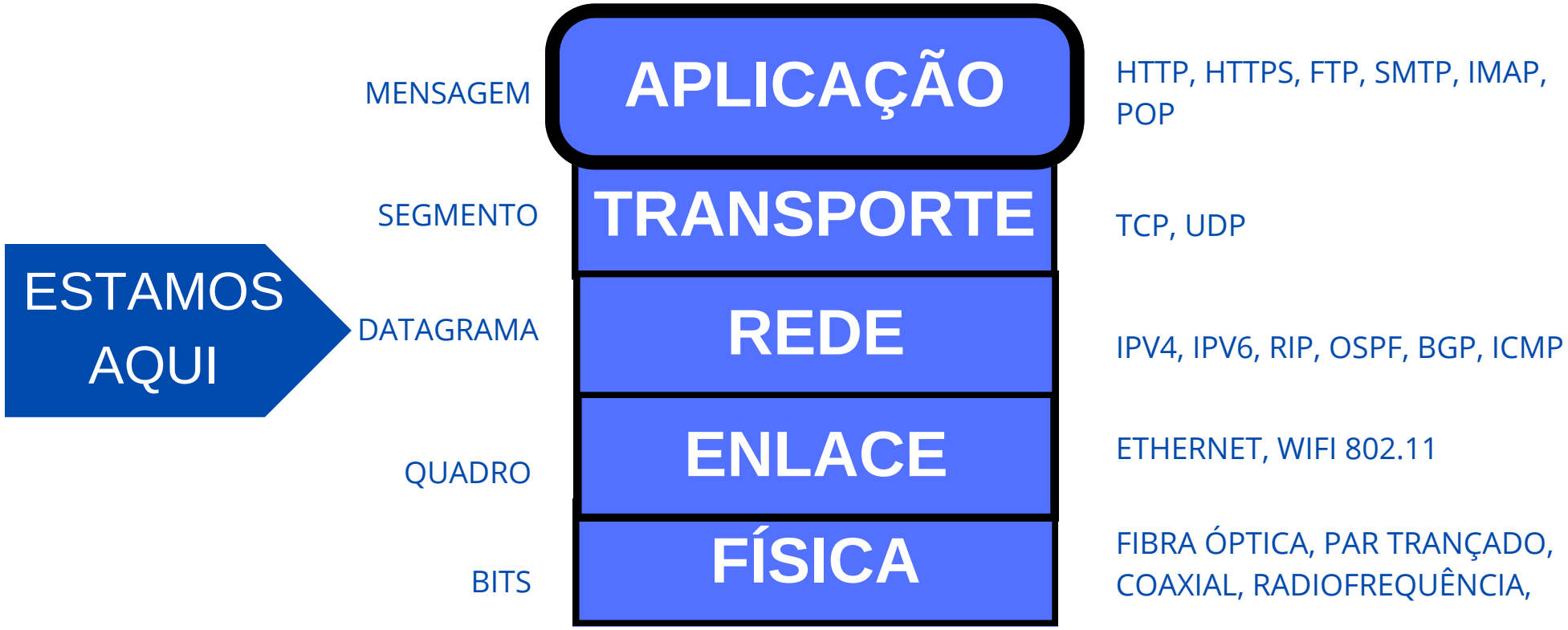
Redes de Computadores

Marisangila Alves, MSc

marisangila.alves@proton.me

Camada de Aplicação

Camada de Aplicação



Procotolo da Camada de Aplicação

Um protocolo de camada de aplicação define como processos de uma aplicação, que funcionam em sistemas finais diferentes, trocam mensagens entre si.

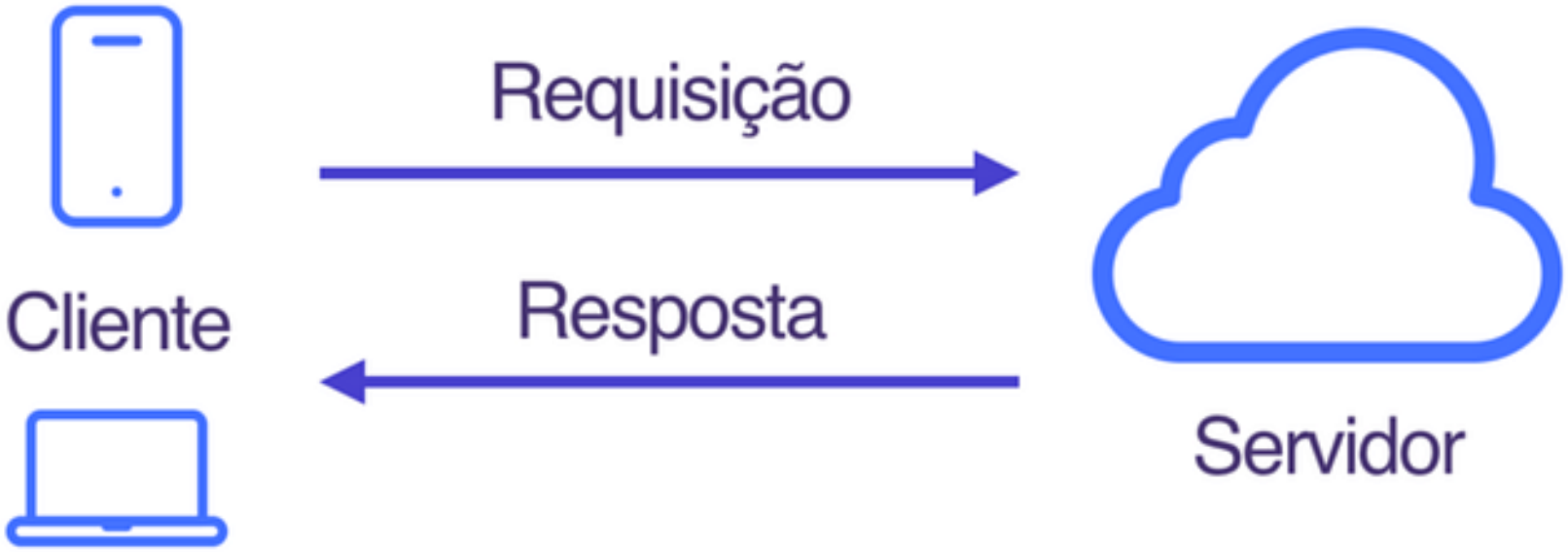
Exemplo de Aplicações e protocolos

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
	SIP [RFC 3261], RTP [RFC 3550] ou proprietária	

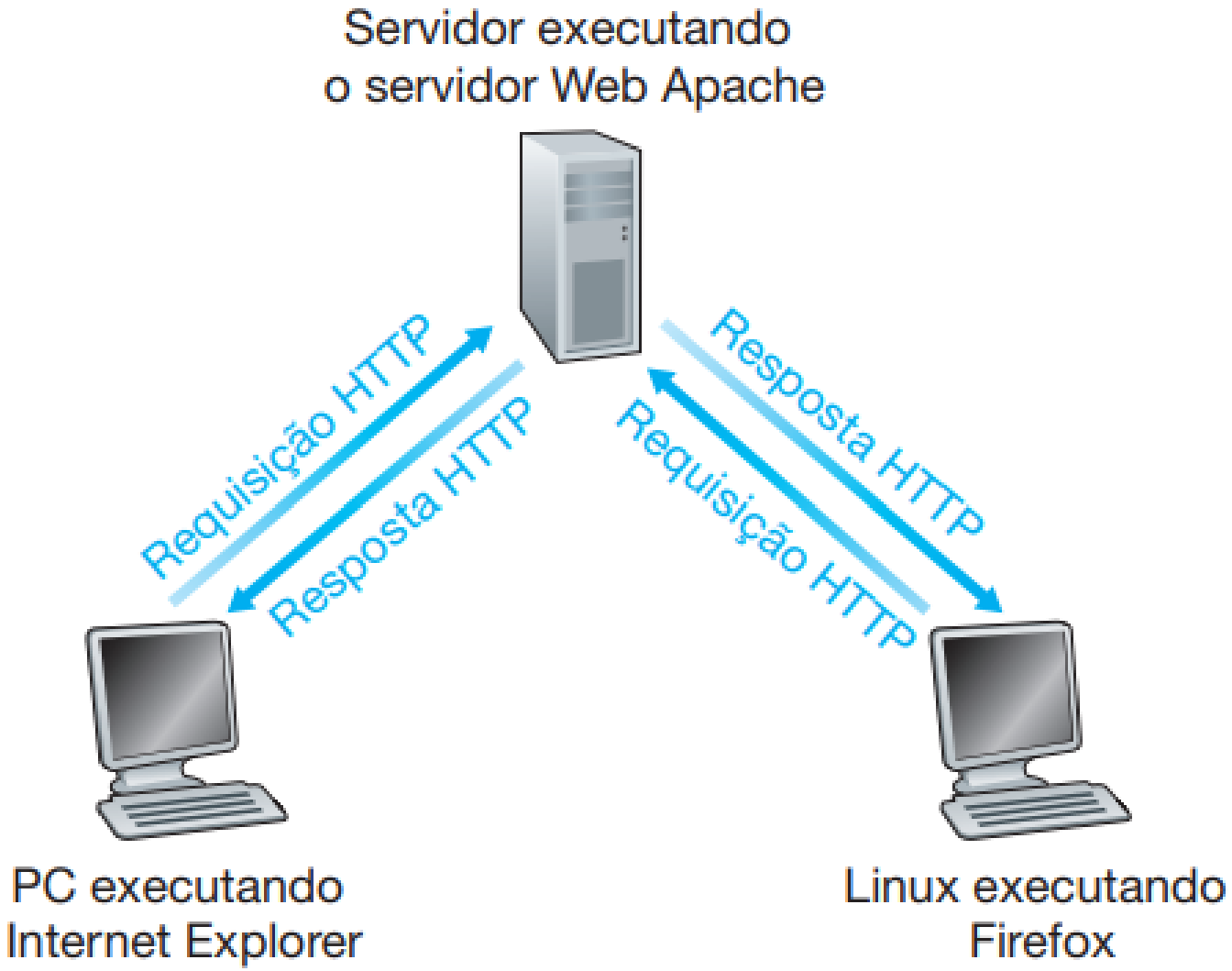
HTTP

Hypertext Transfer Protocol

HTTP - Requisição e Resposta



HTTP


Servidor executando
o servidor Web Apache
Requisição HTTP
Resposta HTTP
Resposta HTTP
Requisição HTTP
PC executando
Internet Explorer
Linux executando
Firefox

HTTP

Mensagem HTTP:

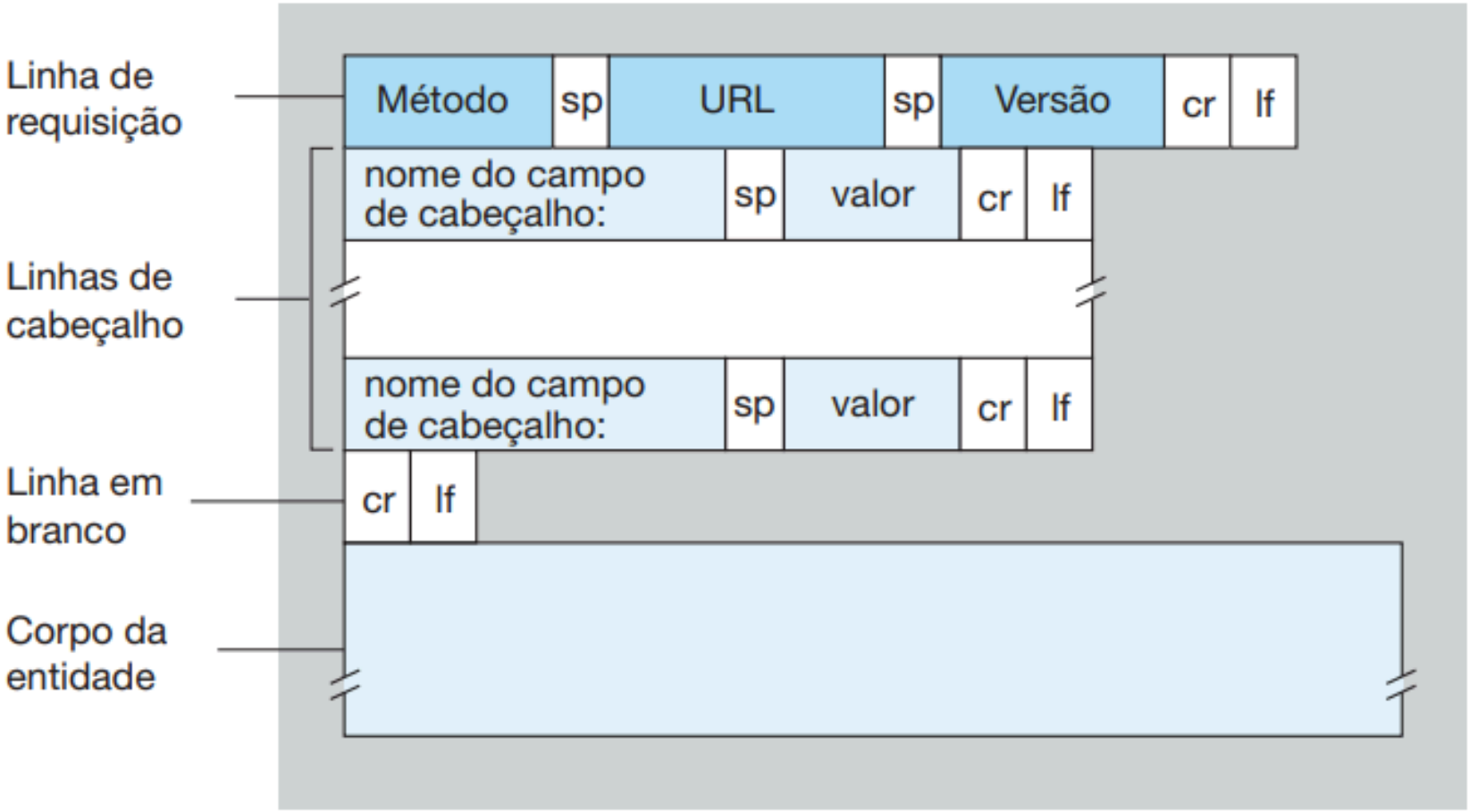
```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

(dados dados dados dados dados ...)
```

Acesse: https://reqbin.com/ e visualize.

HTTP

Cabeçalho de uma mensagem HTTP:



HTTP - *Status Code*

1xx: Mensagens de informação.

2xx: Mensagens de sucesso.

3xx: Mensagens de redirecionamento.

4xx: Mensagens de erro direcionadas ao cliente.

5xx: Mensagens de erro direcionadas ao servidor.

Acesse: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

HTTP - *Status Code*

Exemplos:

- 200: Sucess/Ok
- 301: Objeto movido.
- 401: Não autorizado.
- 404:Objeto não encontrado.
- 500: Erro interno do servidor.
- 505: Versão do HTTP não suportada.

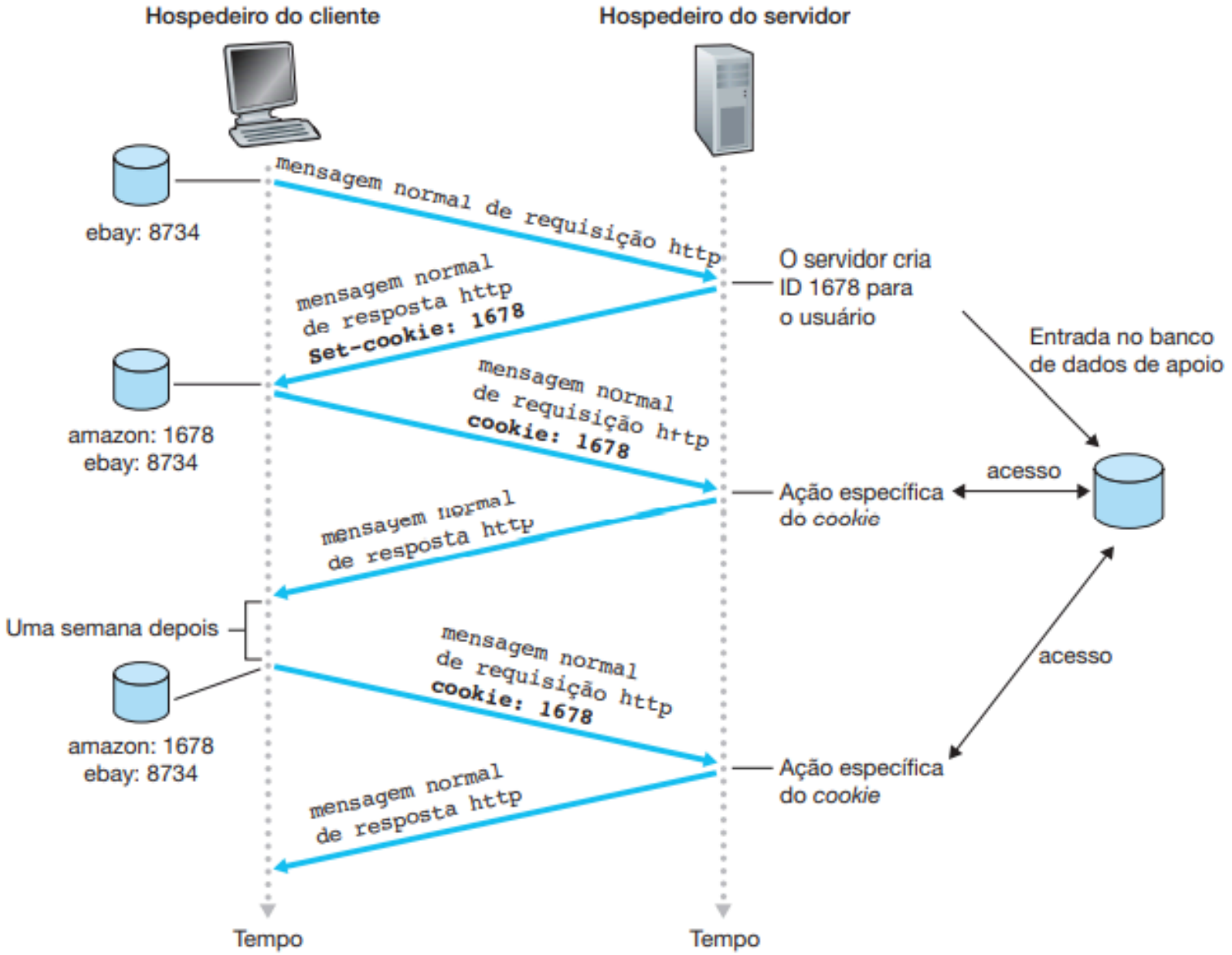
HTTP - Métodos

Principais métodos:

- **GET**: Faz requisições para solicitar dados *(exemplo: um cabeçalho e o corpo da página)*
- **HEAD**: Faz requisições como o GET, no entanto, solicita somente o cabeçalho.
- **POST**: Faz requisições para enviar dados.
- **PUT**: Faz requisições para atualizar dados.
- **DELETE**: Faz requisições para excluir dados.

Acesse: https://reqbin.com/

Cookies


Hospedeiro do cliente
Hospedeiro do servidor
mensagem normal de requisição http
ebay: 8734
mensagem normal
de resposta http
set-cookie: 1678
O servidor cria
ID 1678 para
o usuário
Entrada no banco
de dados de apoio
mensagem normal
de requisição http
cookie: 1678
amazon: 1678
ebay: 8734
acesso
Ação específica
do cookie
mensagem normal
de resposta http
Uma semana depois
mensagem normal
de requisição http
cookie: 1678
acesso
amazon: 1678
ebay: 8734
Ação específica
do cookie
mensagem normal
de resposta http
Tempo
Tempo

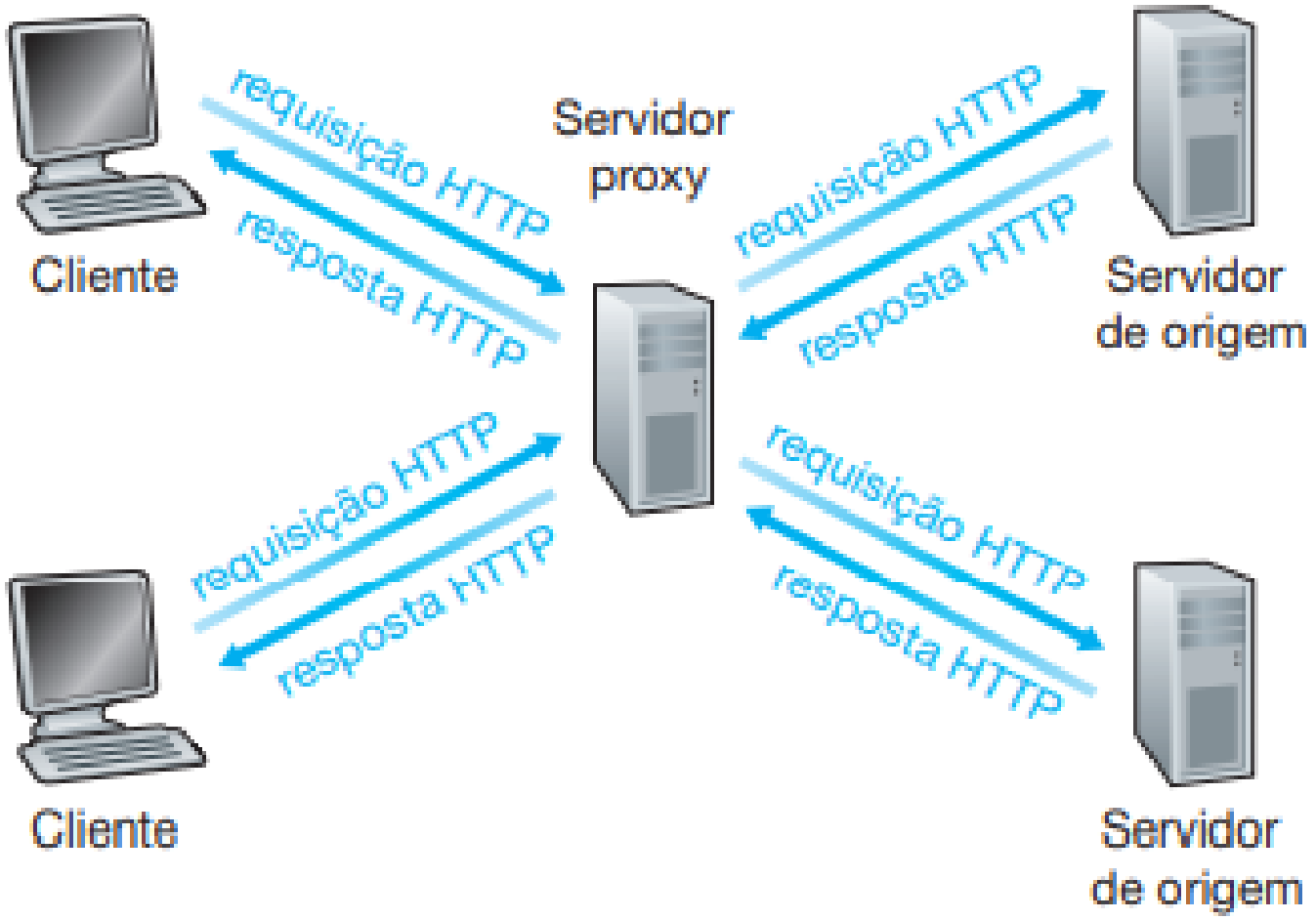
Sessão

As sessões têm um princípio similar aos cookies.
No entanto, as sessões são armazenadas pelo servidor web, e não pelo navegador.

Quando construímos uma aplicação que necessita de autenticação, no momento em que o usuário efetuar o login, podemos até permitir que algumas informações sejam armazenadas em um cookie.

Porém, para dados “sensíveis”, como usuário e e-mail, é mais adequado o armazenamento em sessões. Não é seguro que esse tipo de informação seja trafegada através da rede.

Cache *WEB*


Cliente
requisição HTTP
resposta HTTP
Servidor
proxy
requisição HTTP
resposta HTTP
Servidor
de origem
Cliente
requisição HTTP
resposta HTTP
requisição HTTP
resposta HTTP
Servidor
de origem

Cache *WEB*

Primeiramente o cliente solicita o conteúdo usando um método GET.

Na primeira que a página é acessada, está será a resposta se tudo estiver ok.

O status code é 200.

```
HTTP/1.1 200 OK
Date: Sat, 8 Oct 2011 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 7 Sep 2011 09:23:24
Content-Type: image/gif

(dados dados dados dados dados ...)
```

Cache *WEB*

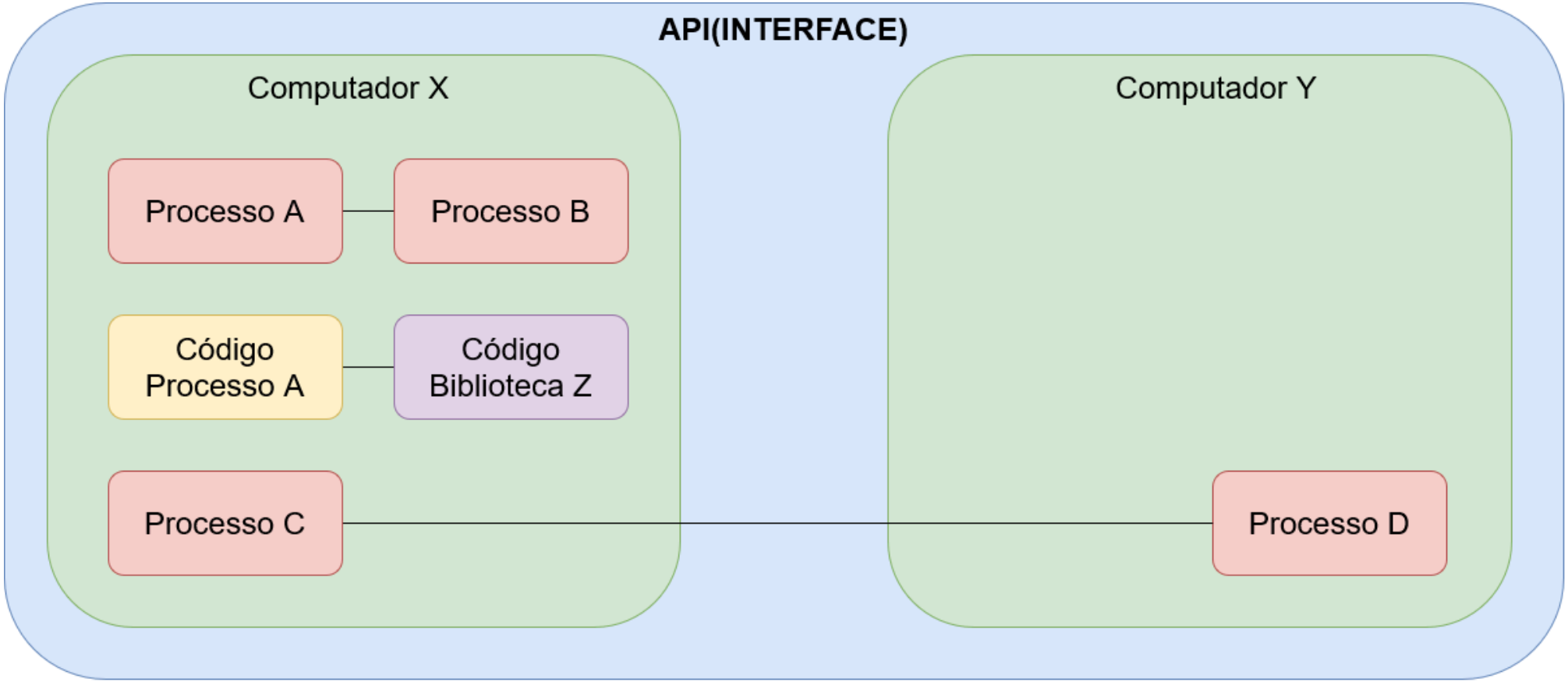
Na segunda vez, esta será a mensagem. Nesse caso o proxy verificou se havia um cache armazenado da página e perguntou a serviodr se havia alguma modifação na página. Isso significa que a página armazenada não está desatualizada.

O status code é 304.

```
HTTP/1.1 304 Not Modified
Date: Sat, 15 Oct 2011 15:39:29
Server: Apache/1.3.0 (Unix)

(corpo de mensagem vazio)
```

API *(Application Programming Interface)*



API *(Application Programming Interface)*

- Forma de comunicação entre dois processos.
    - Isto é, troca de mensagens entre duas aplicação sendo executadas em locais diferentes.

- WEB Services: Comunicação usando protocolo HTTP.

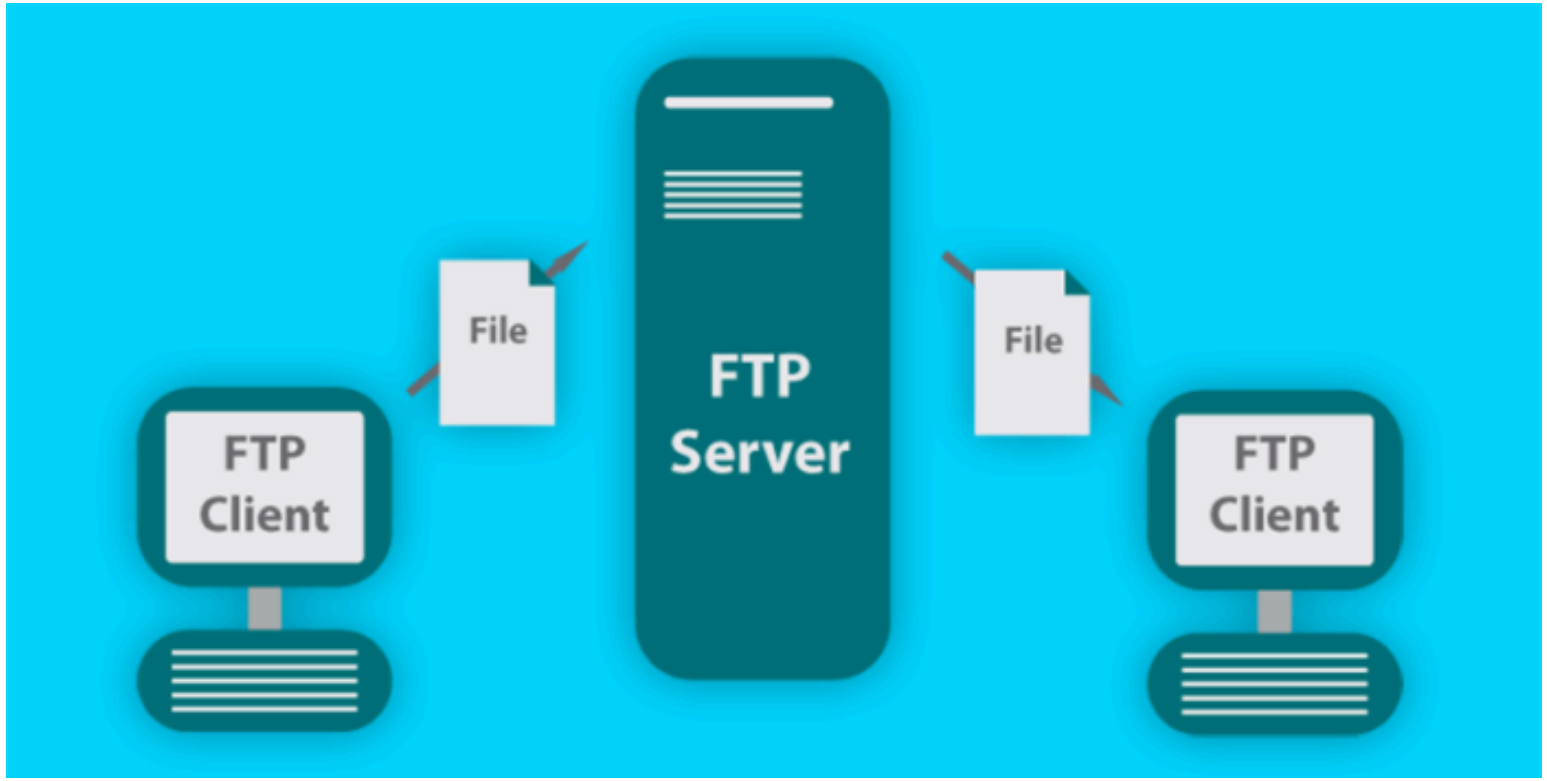
- Nem todas as APIs são Web services, porém, todos os Web Services são APIs.

WEB Service

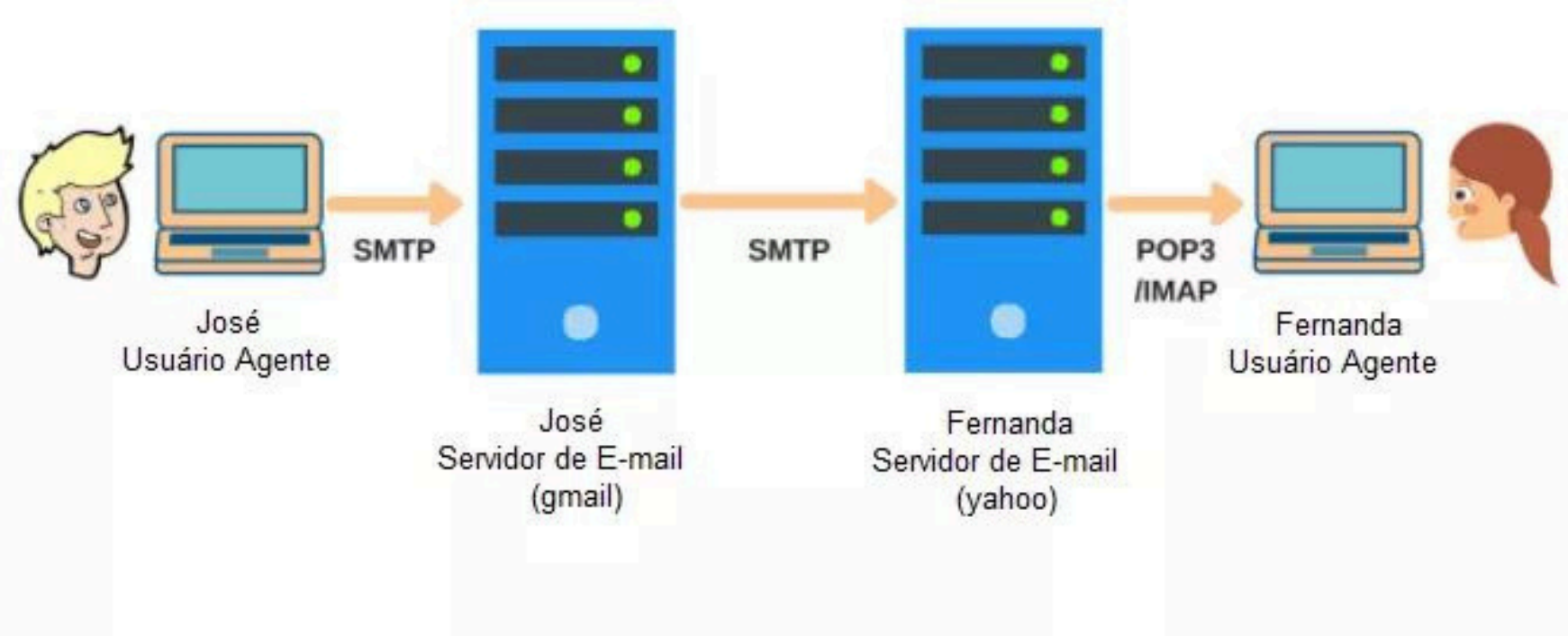
- Vamos ver como funciona um web service na prática?
- Usaremos o WEB Service disponibilizado pelo Correios.
    - Quando enviamos um CEP, o web service retorna uma reposta com logradouro, bairro, localidade e UF (estado) em um arquivo no formato JSON.
        - https://viacep.com.br/ws/cep/json/
- Vamos visualizar a resposta no site: https://reqbin.com/
- Agora vamos ver como funciona com Java Script

FTP *(File Transfer Protocol)*

Um protocolo para transferir arquivos.



Correio Eletrônico - E-mail



SMTP *(Simple Mail Transfer Protocol)*

- Usado para envio de emails.
- Não está relacionado ao recebimento de mensagens.

POP3 *(Post Office Protocol – Versão 3)*

- Faz *download* emails do servidor e os armazena localmente.
- Remove mensagens do servidor após o *download*.
- Não sincroniza informações entre dispositivos. Por exemplo, se você marcar um email como lido em um dispositivo, isso não será refletido em outros.
- Requer *backup* local no dispositivo para evitar perda de emails, já que as mensagens são removidas do servidor.

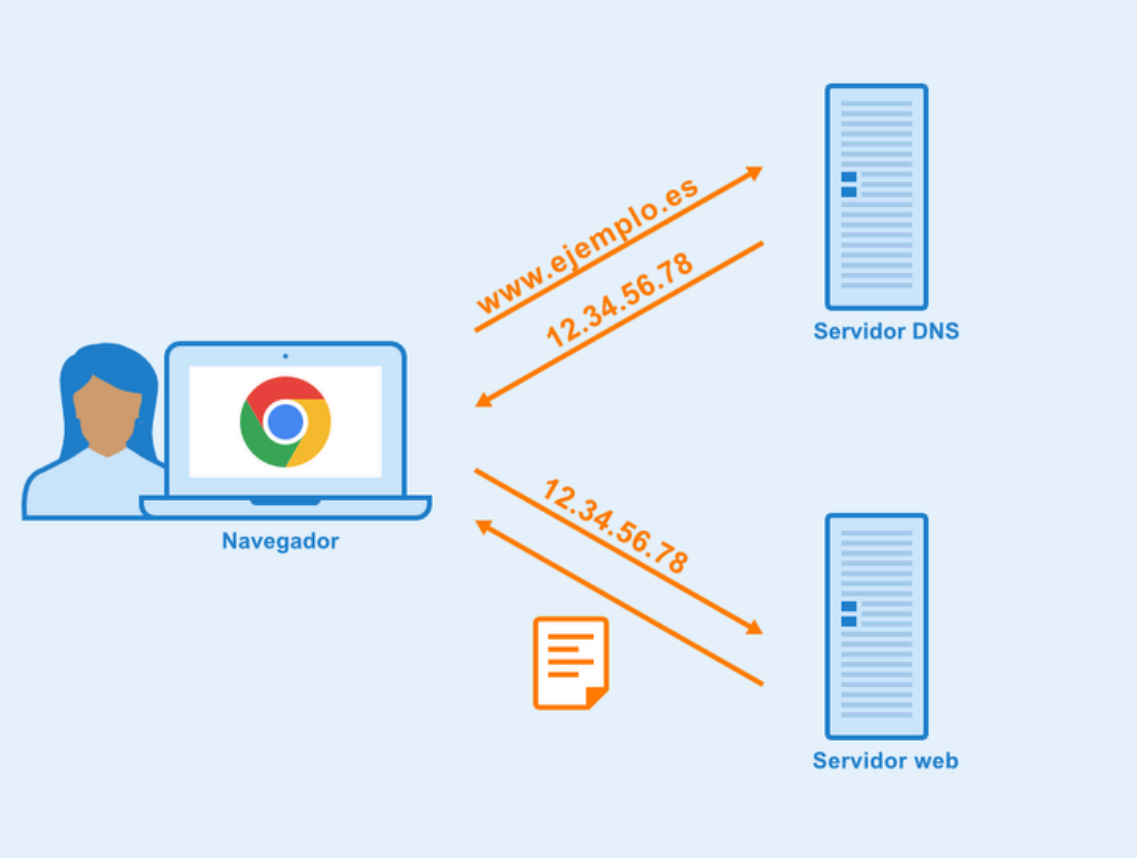
IMAP *(Internet Message Access Protocol)*

- Sincroniza emails entre todos os dispositivos que utilizam a conta.
- Ao marcar um email como lido em um dispositivo, a alteração se reflete automaticamente nos outros dispositivos.
- A principal desvantagem é que o espaço de armazenamento é utilizado na nuvem (no servidor do provedor), o que pode consumir mais espaço de armazenamento online.

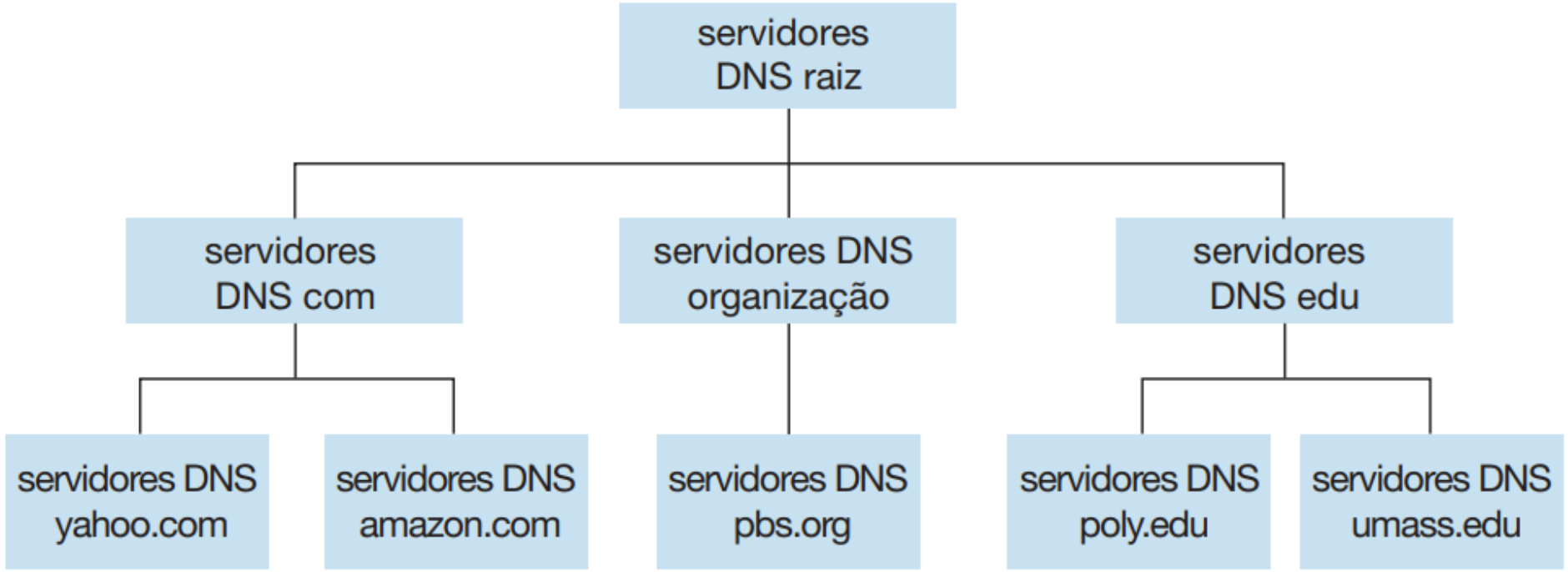
DNS *(Domain Name System)*

Um banco de dados distribuído.

Traduz uma URL para um endereço IP

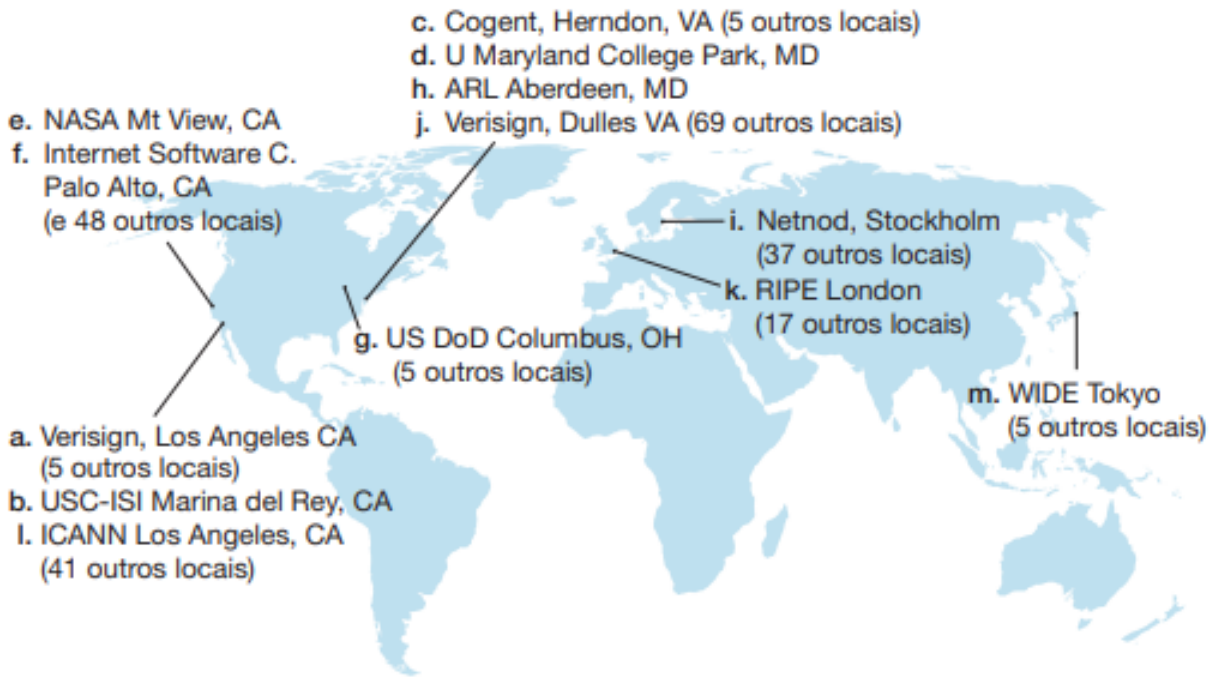


DNS *(Domain Name System)*



Use o comando: `nslookup "examplo.com"` no CMD

DNS *(Domain Name System)*

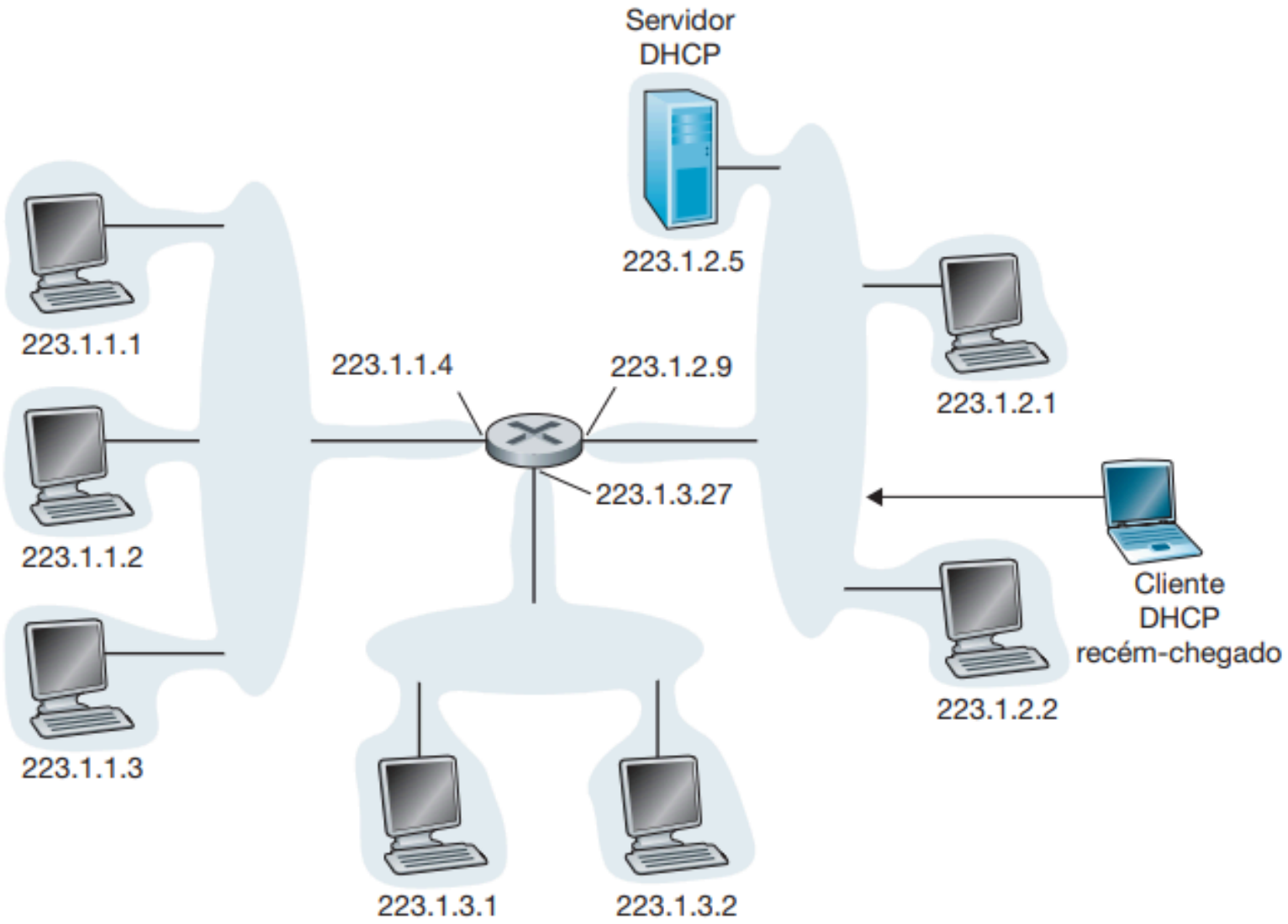


Acesse: https://www.iana.org/domains/root/db

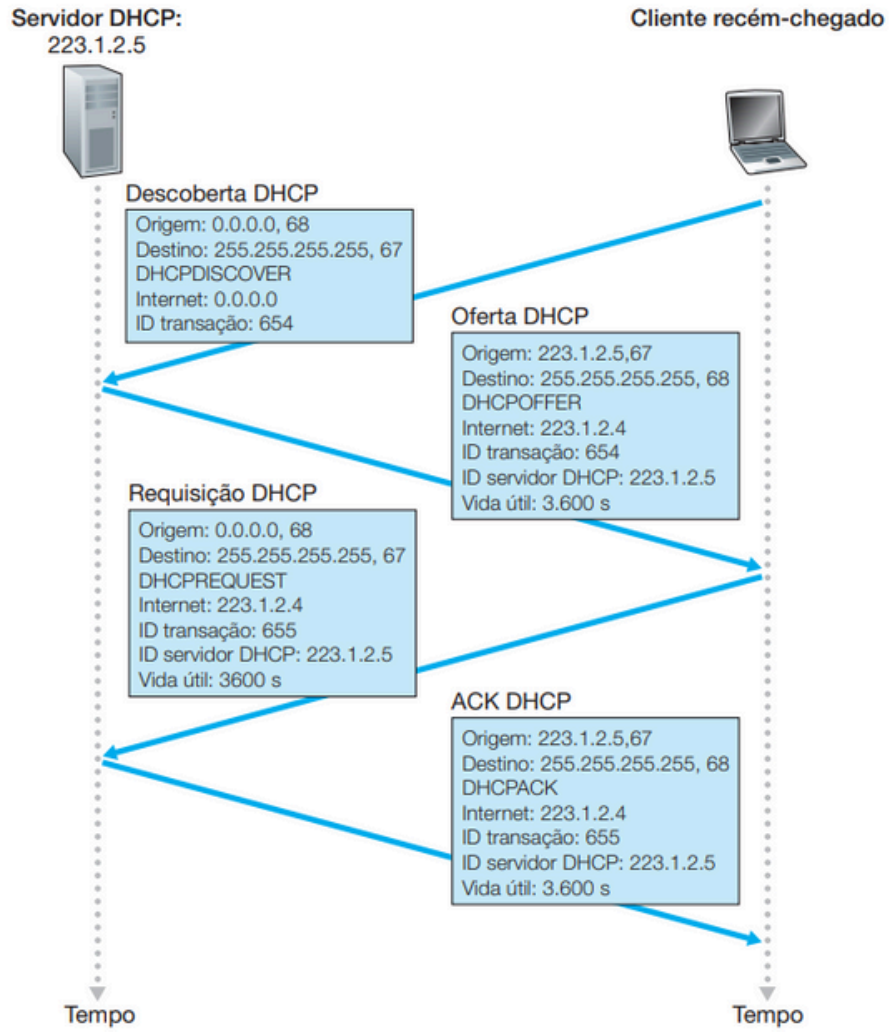
DHCP - (*Dynamic Host Configuration Protocol)*

- Quando um dispositivos se conectada a uma rede nova, ele não tem um IP.
- Então a primeira tarefa de um dispositivo recém-chegado é encontrar um servidor DHCP com quem interagir. Isso é feito utilizando uma mensagem de descoberta DHCP, a qual o dispositivo envia dentro de um pacote UDP .
- Em seguida o servidor DHCP atribui um IP para este novo dispositivo.

DHCP - (*Dynamic Host Configuration Protocol*)



DHCP - (*Dynamic Host Configuration Protocol*)



DHCP - (*Dynamic Host Configuration Protocol*)

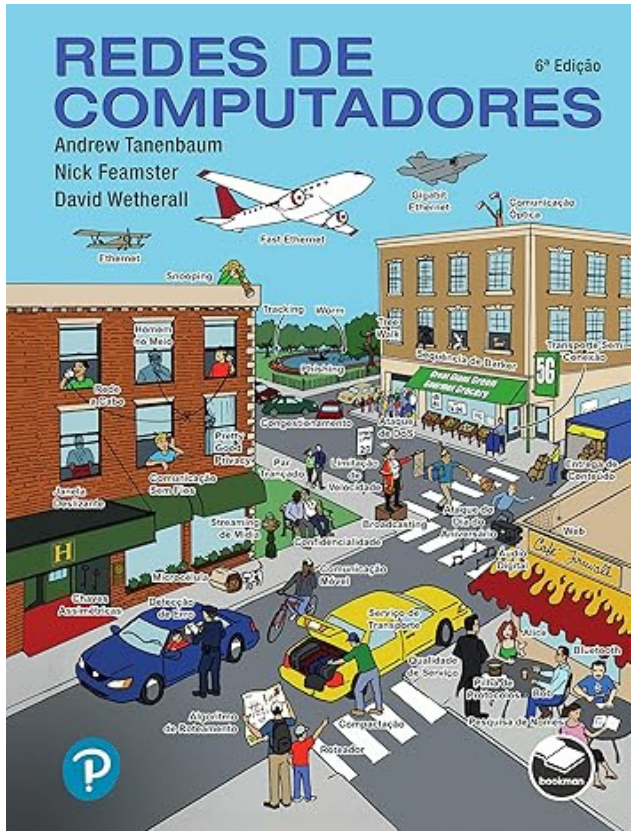
APIPA (*Automatic Private IP Addressing*):

- Método usado para atribuir automaticamente um endereço IP privado quando não é possível obter um endereço de um servidor DHCP.
- APIPA usa endereços na faixa de **169.254.0.1** a **169.254.255.254**.
- Se um dispositivo não conseguir um endereço IP via DHCP, ele automaticamente atribui a si mesmo um endereço IP APIPA, permitindo que se comunique com outros dispositivos na mesma sub-rede APIPA.
- Comunicação apenas na LAN.

Leitura Recomendada

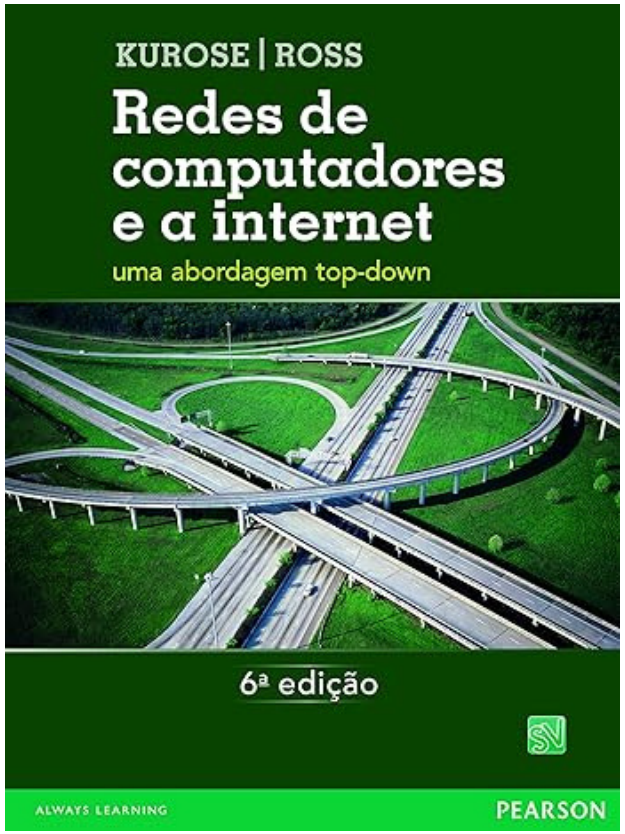
Capítulo 7 do livro:

Redes de Computadores



Capítulos 2 do livro:

Redes de Computadores e a internet



Referências

WETHERALL, J.; TANENBAUM, A. S. Redes de Computadores. 6ª edição. Rio de Janeiro: Editora Campus, 2021.

KUROSE, James F.; ROSS, Keith W. Redes de Computadores e a Internet. 5ª edição. São Paulo: Person, 2021.

FOROUZAN, Behrouz A. Comunicação de dados e redes de computadores. 4ª edição. AMGH Editora, 2010.

INTERNET ENGINEERING TASK FORCE (IETF). RFCs. Disponível em: https://www.ietf.org/process/rfcs/. Acesso em: 22 out. 2024.

Redes de Computadores

Marisangila Alves, MSc

marisangila.alves@proton.me