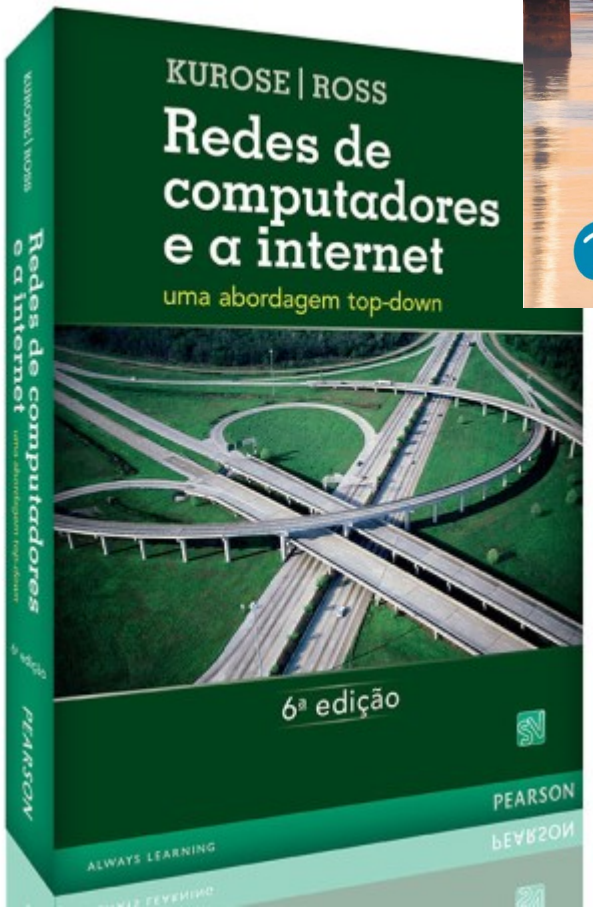
Redes de Computadores
Camada de Aplicação

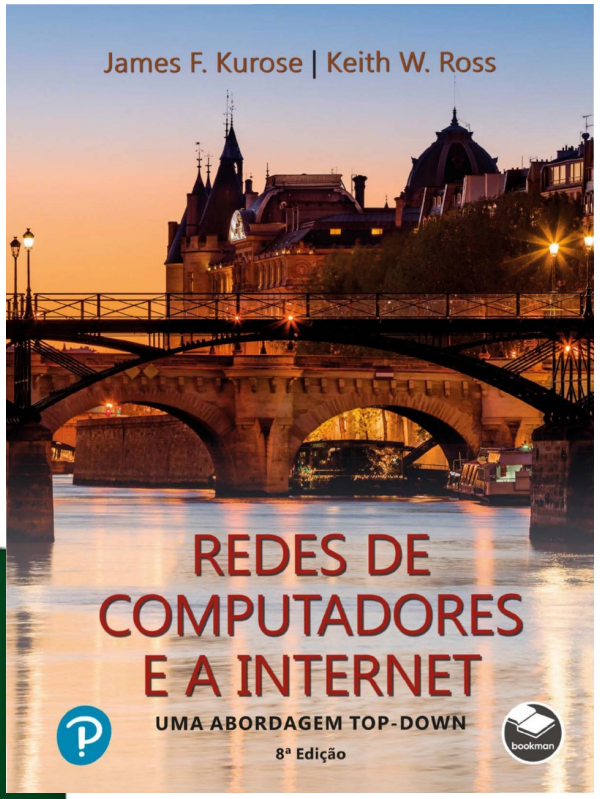
Professor: Fábio Renato de Almeida

https://github.com/fabiorenatodealmeida

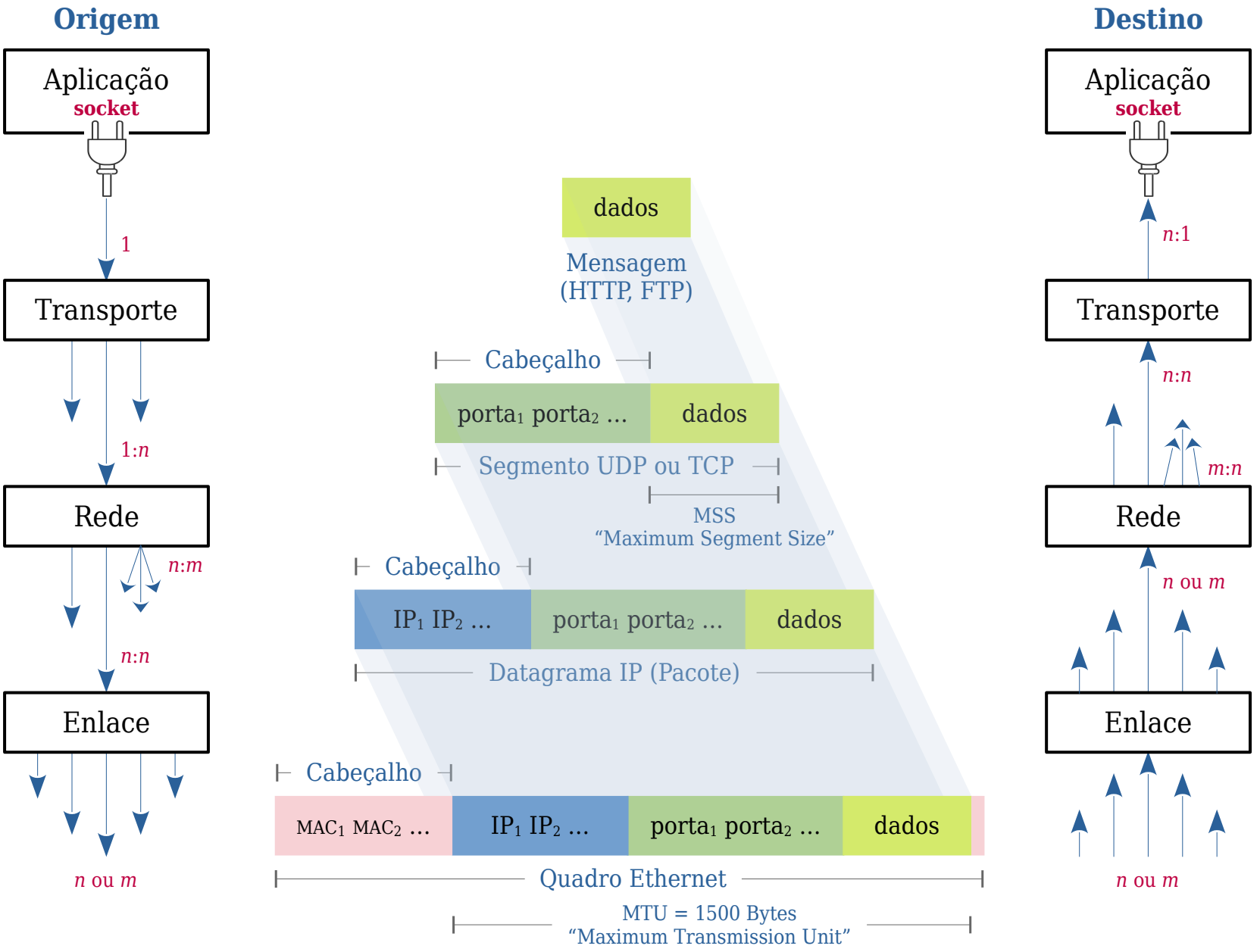
e-mail: fabiorenatodealmeida@hotmail.com

Bibliografia

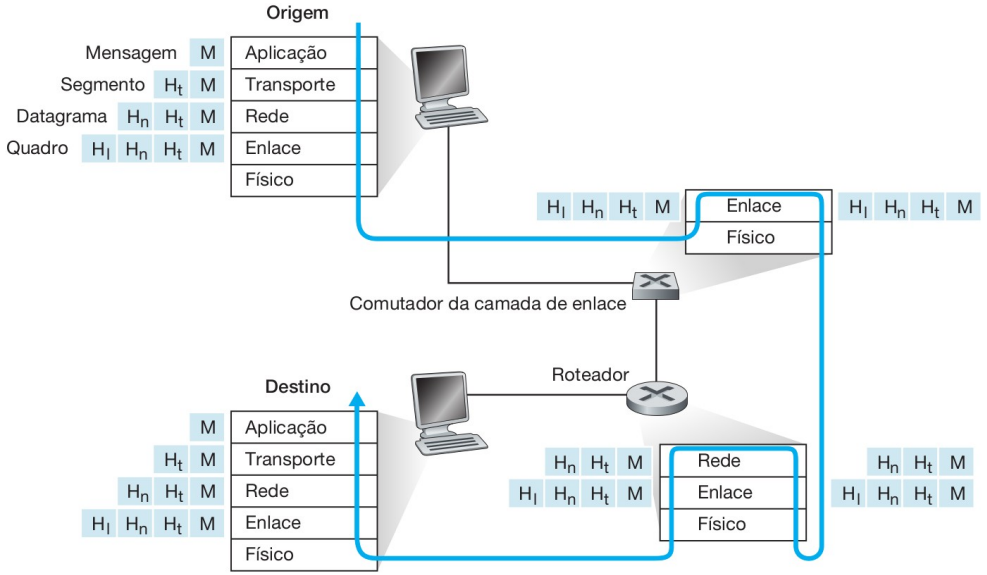


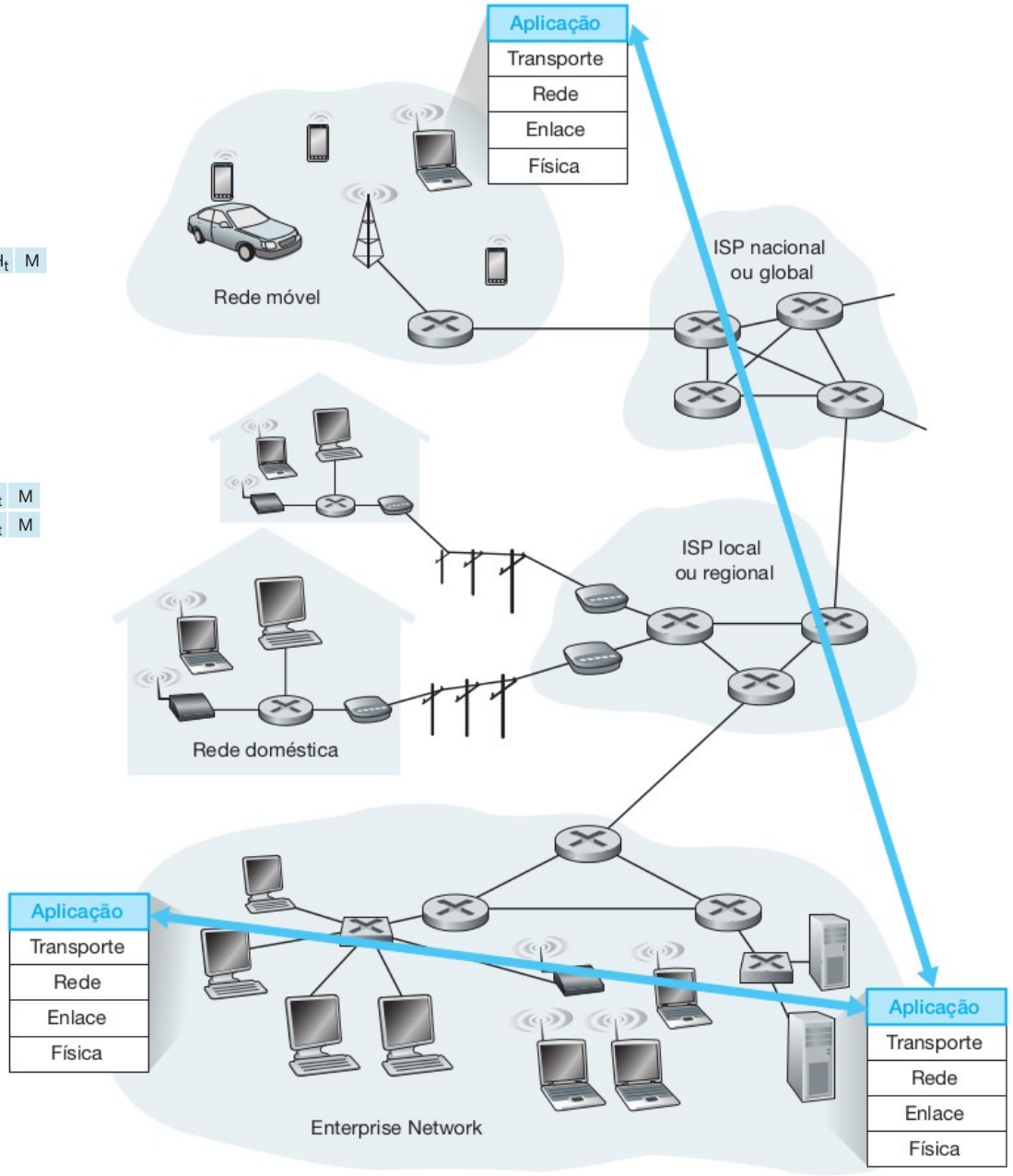


Pilha de Protocolos

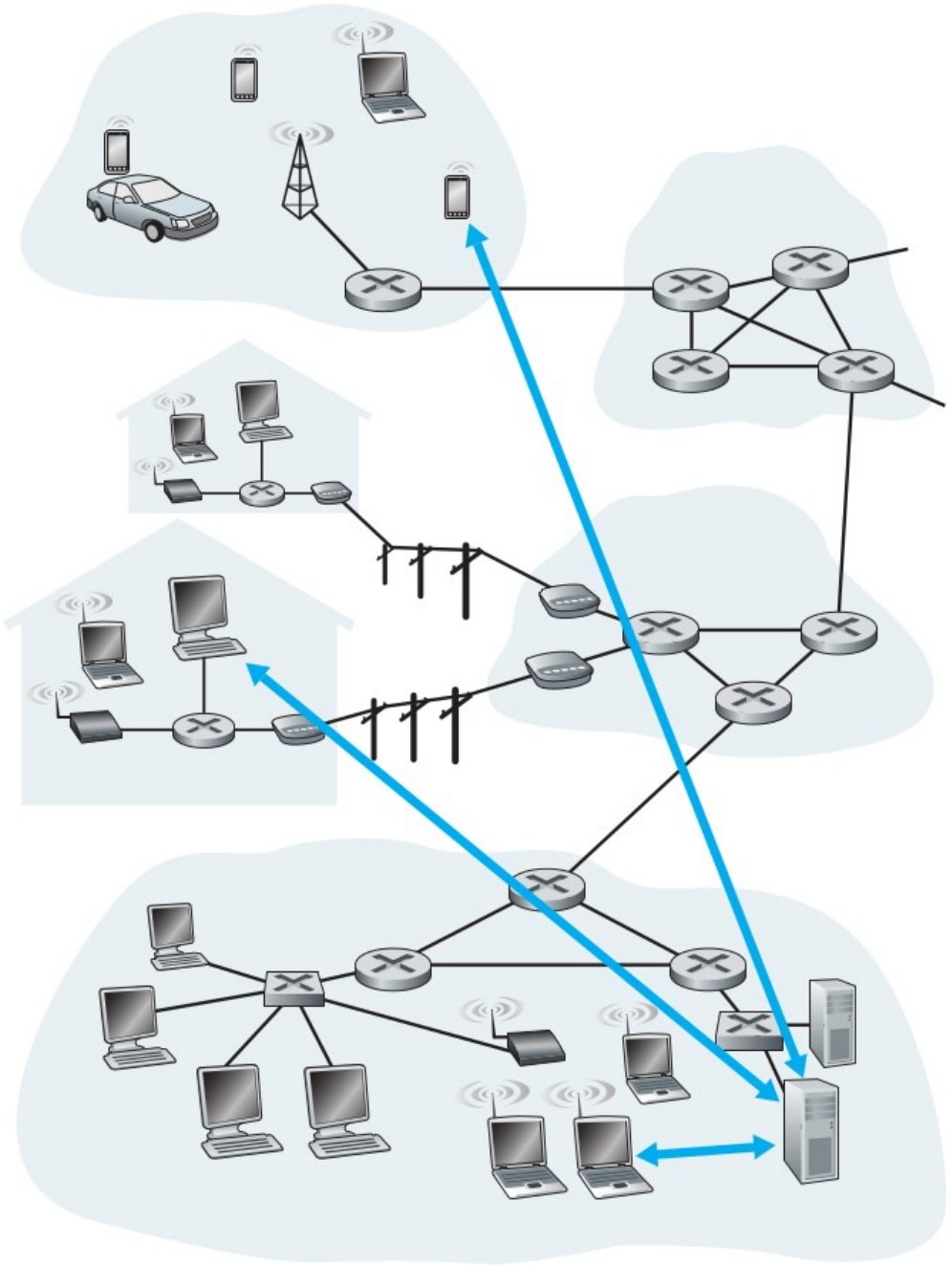


Do ponto de vista do desenvolvimento de software, a camada de Aplicação e os serviços oferecidos pela camada de Transporte são os mais importantes.





Arquitetura: Cliente/Servidor



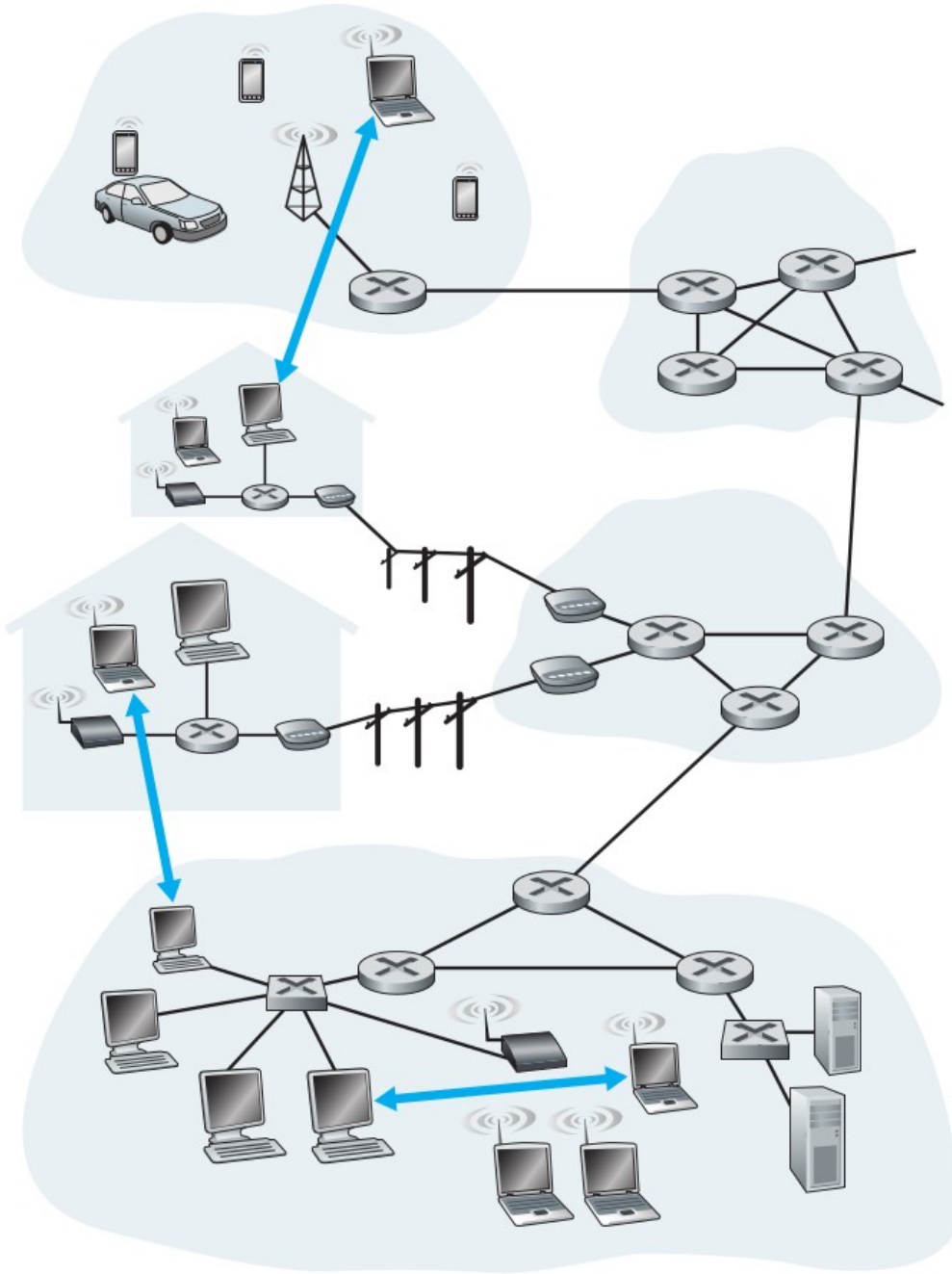
Um **host**, denominado servidor, é responsável por atender requisições de outro **host**, denominado cliente.

Exemplos:

- Web
- Transferência de arquivos (FTP)
- e-mail

Notar que na arquitetura cliente/servidor não há comunicação direta entre os clientes.

Arquitetura: Peer-to-Peer (P2P)



Os **hosts** em uma rede P2P comunicam-se diretamente. Contudo, algumas vezes ainda há a figura de um servidor central: Arquitetura Híbrida (Skype).

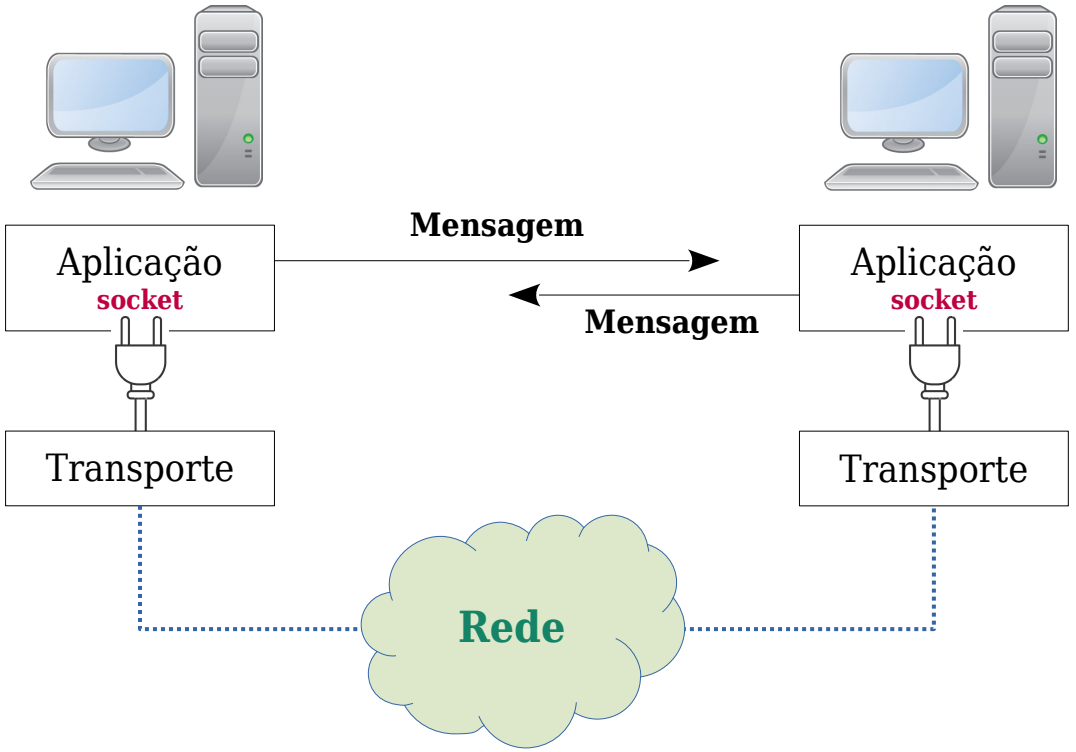
Exemplos:

- BitTorrent
- Serviços de mensagem instantânea

Vantagem: Escalabilidade

Desvantagem: O sucesso depende da participação dos usuários (nº de pares)

Comunicação entre processos



A comunicação ocorre por meio da troca (envio/recebimento) de mensagens.

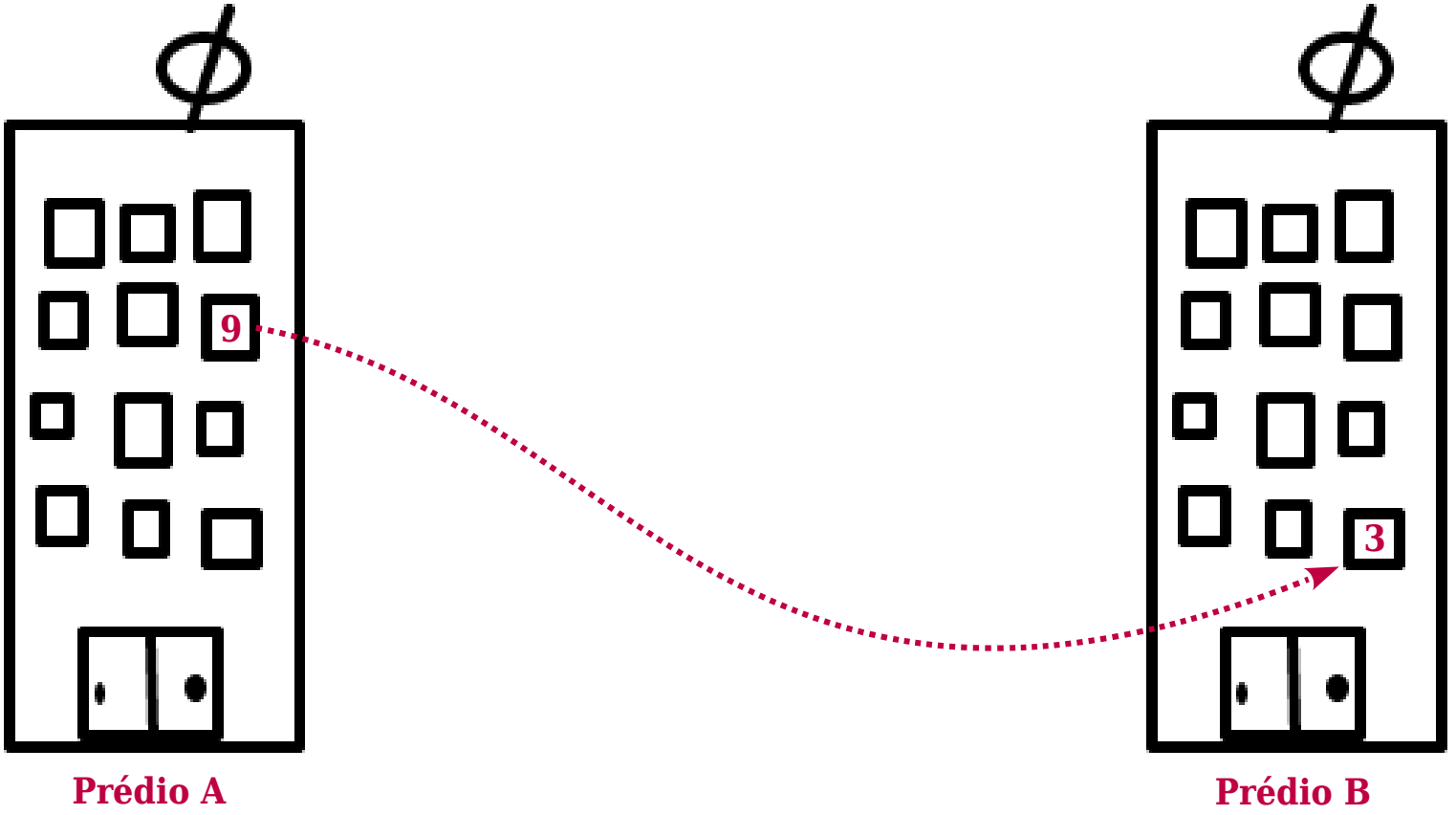
Socket: Interface de software (API) utilizada para enviar e receber mensagens pela rede.

Endereçamento:

IP: Identifica o **host** na rede.
Porta: Identifica o processo (**socket**) que receberá a mensagem.

Comunicação entre processos (Analogia)

O morador do apartamento **9** no prédio **A** quer falar com o morador do apartamento **3** no prédio **B**.



Hosts: Prédios A e B
Endereços IP: Endereços dos prédios
Números de Porta: Números dos apartamentos

TCP/UDP

Aplicação
socket

TCP/UDP
Transporte

(TCP)

→ Garantia de entrega: SIM / NÃO

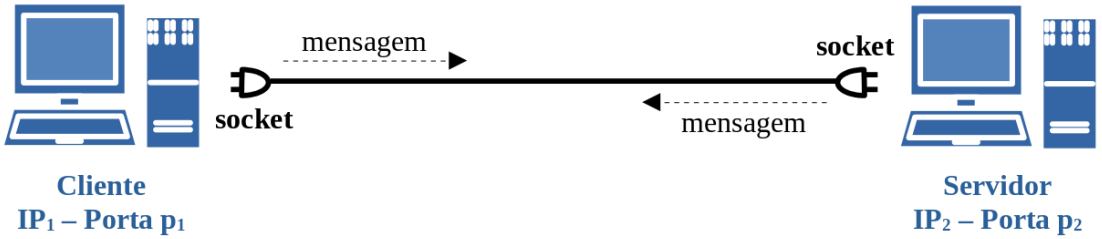
(UDP)

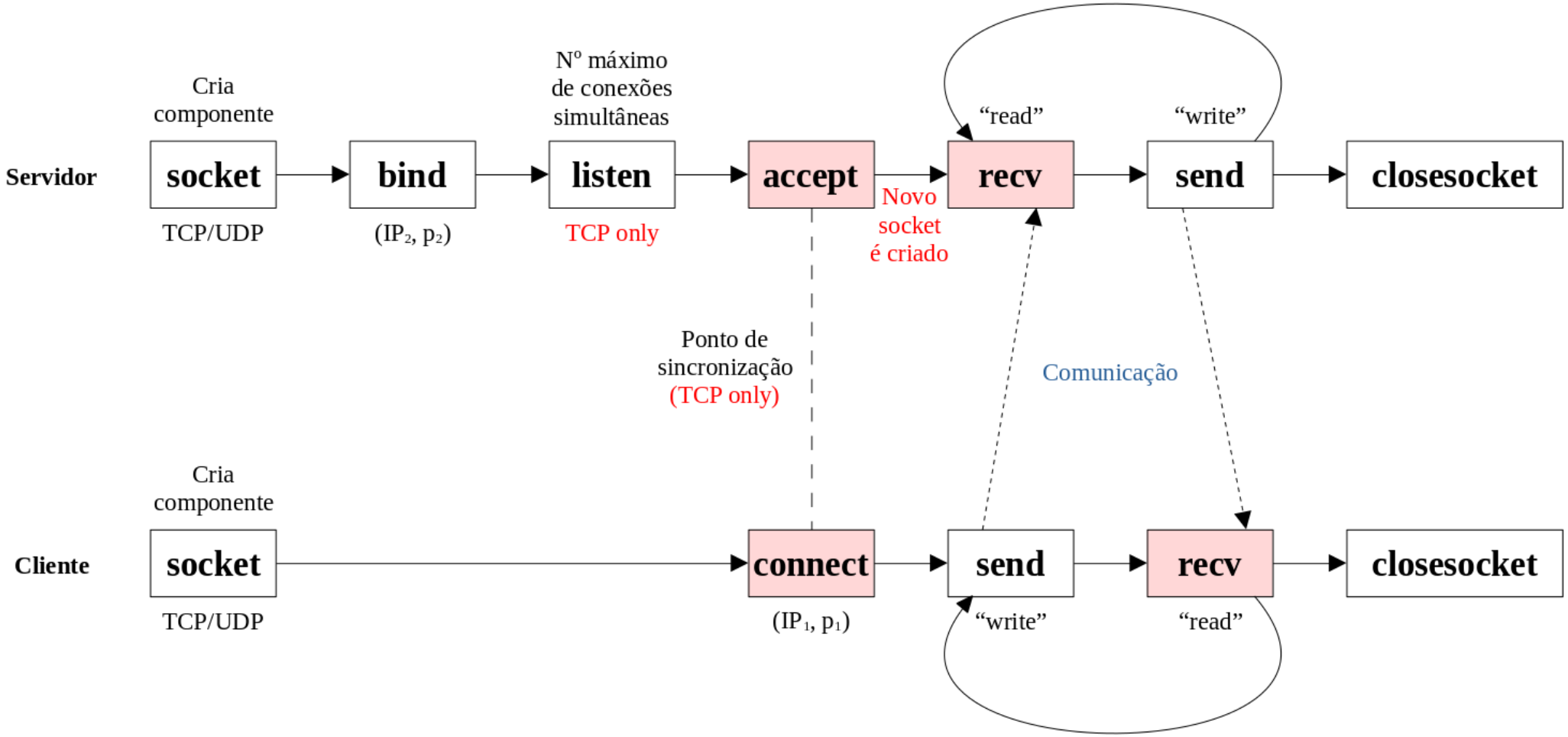
~~Vazão mínima: throughput~~

~~Temporização: Variância de atraso (jitter) máxima na entrega de pacotes.~~

Segurança: Somente com o uso dos protocolos **SSL** (Secure Socket Layer) ou **TLS** (Transport Layer Security) na camada de Aplicação.

Sockets

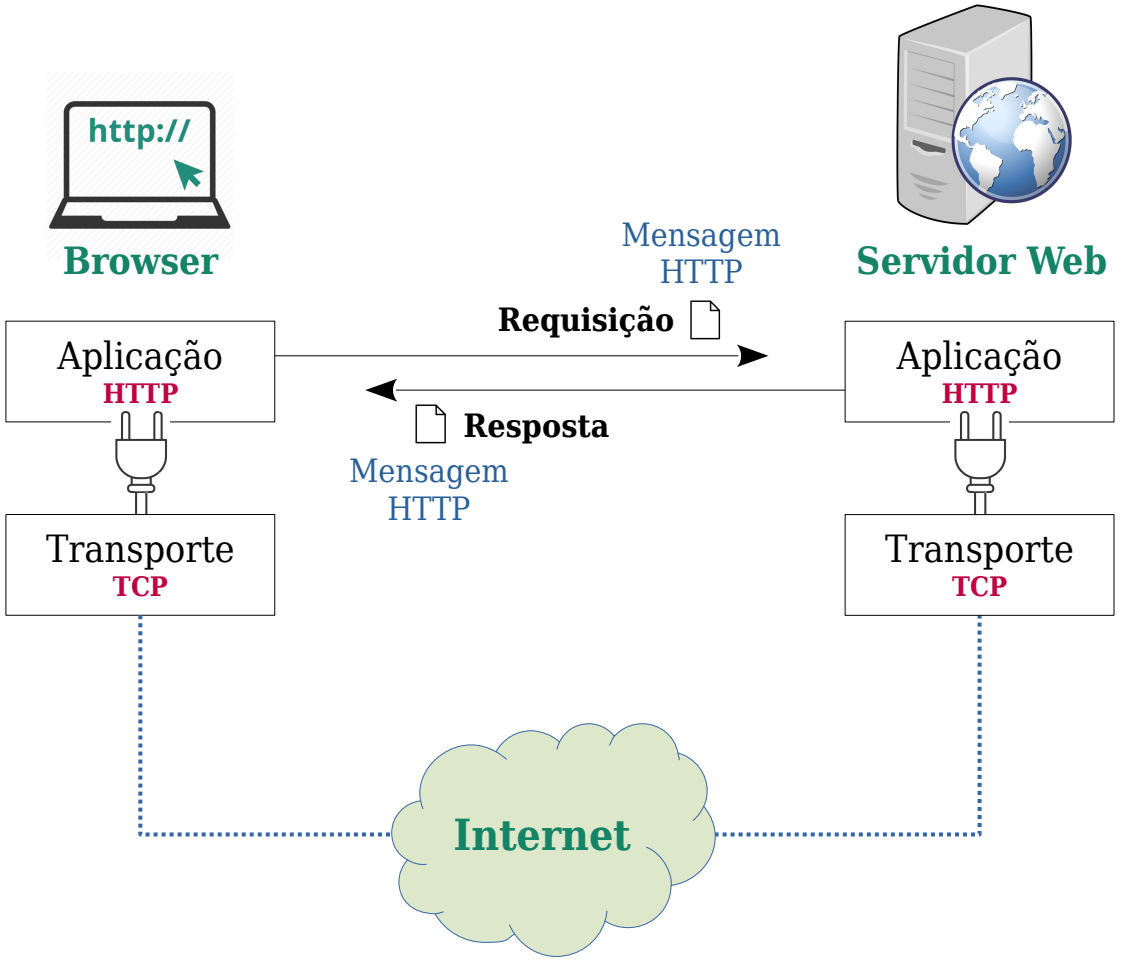




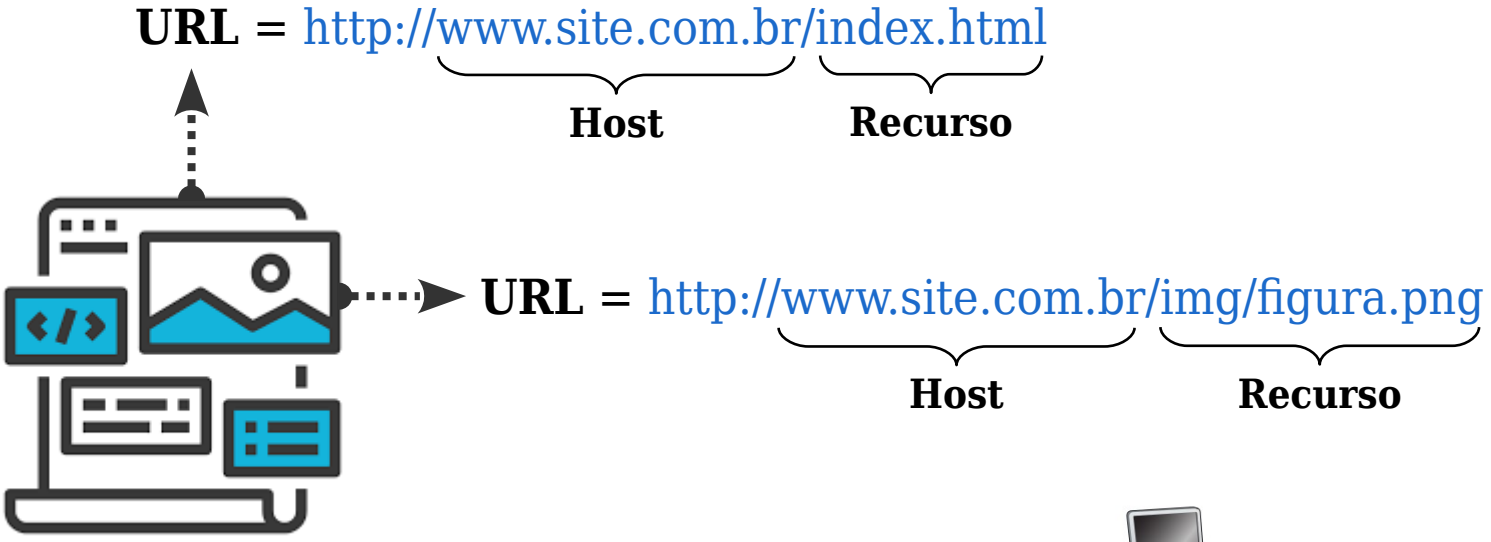
Chamada bloqueante

HTTP – HyperText Transfer Protocol

Define como os recursos (páginas html, códigos CSS e JavaScript, imagens, vídeos, etc.) são transferidos entre o cliente e o servidor Web.



Elementos de uma página Web

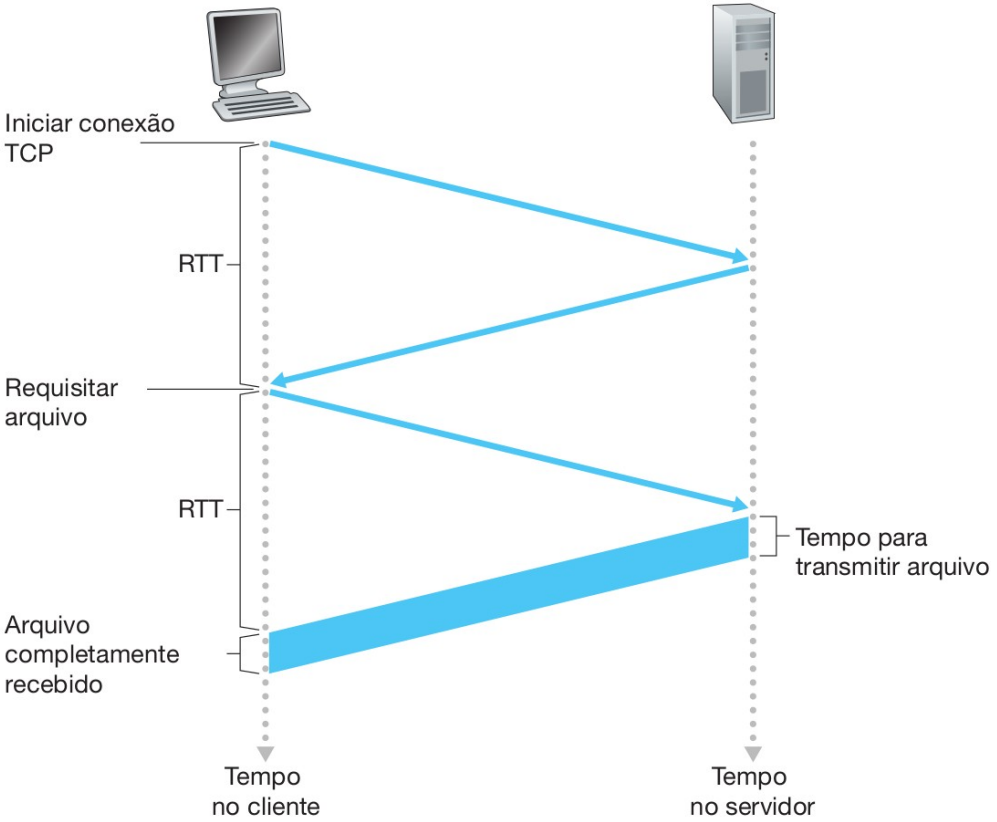


Conexão TCP...

HTTP/1.0 – Uma conexão TCP não persistente para cada requisição.

HTTP/1.1 – Uma conexão TCP persistente (modo padrão) para requisições em série e recentes.

HTTP/2 – Faz multiplexação de recursos com uma única conexão TCP.



Requisição HTTP

Exemplo

```
GET /index.html HTTP/1.1
Host: www.exemplo.com.br
Connection: keep-alive
User-Agent: Mozilla/5.0
Accept-Language: pt-BR
```

→ Utilizado por servidores proxy (cache)

→ Browser informa que prefere conexões persistentes

`close`
(conexão não persistente)

Formato

```
método URL versão
campo1: valor1
campo2: valor2
 ⋮
```

→ Linha de requisição

→ Linhas de cabeçalho

Linha em branco

```
Corpo da Entidade
```

Métodos

GET	download de recurso (parâmetros)
HEAD	`GET` sem o corpo da entidade
POST	envia dados
PUT	upload recurso
PATCH	atualiza parcialmente um recurso
DELETE	exclusão

As linhas devem terminar em **CR** + **LF**

Resposta HTTP

Exemplo

```
HTTP/1.1 200 OK
Connection: keep-alive
Keep-Alive: timeout=5
Server: Apache/2.2.3 (CentOS)
Date: Thu, 24 Feb 2022 11:53:12 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 6721
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT

DADOS...
```

Formato

```
versão código-do-estado mensagem → Linha de estado
campo1: valor1
campo2: valor2                   } Linhas de cabeçalho
 ⋮
```

Linha em branco

```
Corpo da Entidade
```

As linhas devem terminar em **CR** + **LF**

Estados

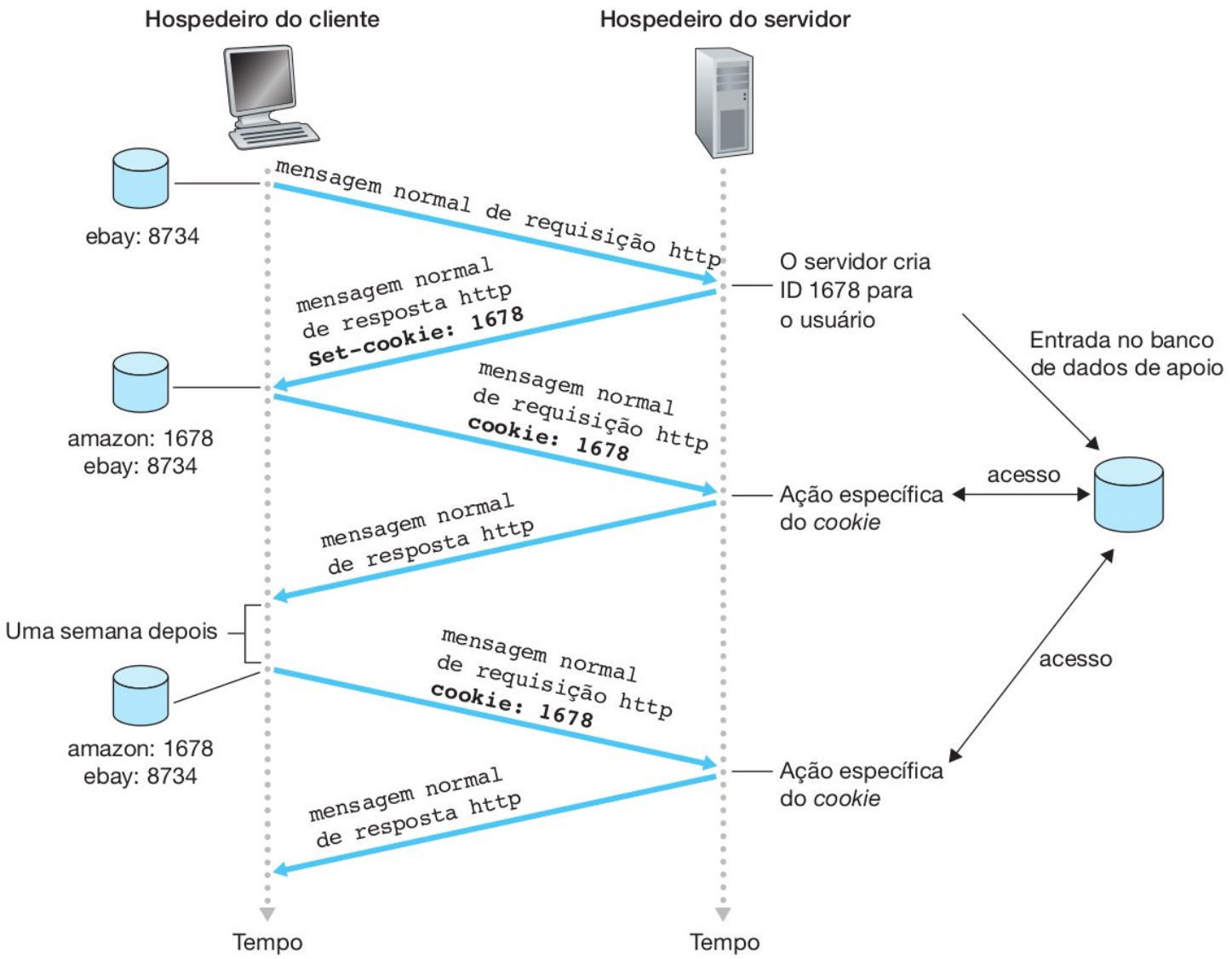
200 OK

Request → If-Modified-Since: ...
304 Not Modified

400 Bad Request
404 Not Found

Cookies

HTTP é um protocolo sem estado (stateless): o protocolo não “lembra” requisições/respostas anteriores. Contudo, o protocolo permite o envio de cookies em suas linhas de cabeçalho.



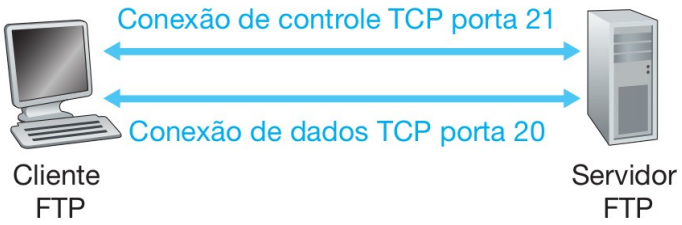
Cookie

Código de identificação exclusivo enviado durante uma requisição HTTP para que o servidor Web possa recuperar o estado necessário para atender a solicitação.

FTP – File Transfer Protocol

Assim como o HTTP, também utiliza o protocolo TCP.

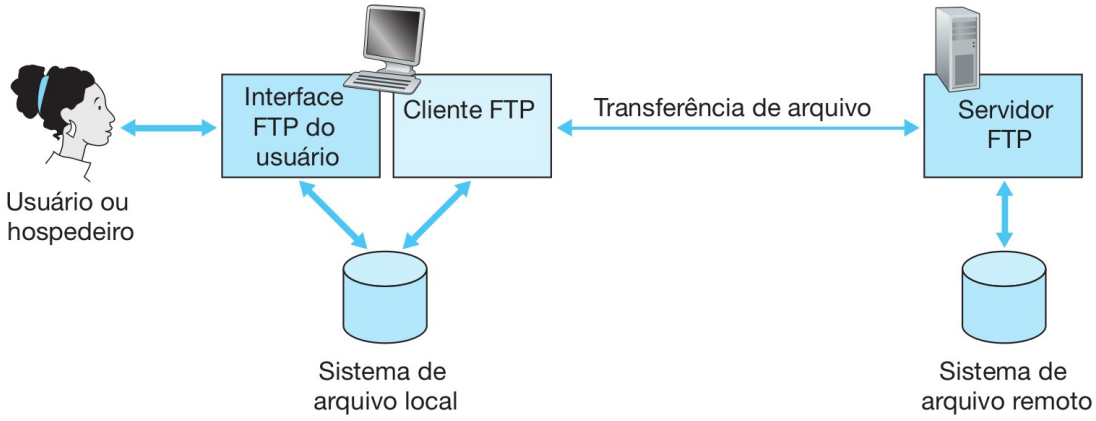
FTP utiliza conexões TCP paralelas: Uma de controle e uma de dados.



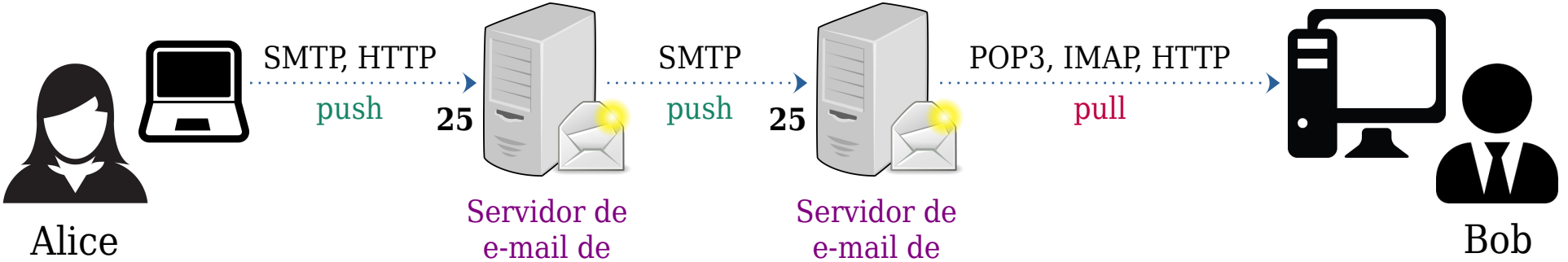
Conexão de controle: utilizada para identificação e senha do usuário, seleção de diretório, requisição de envio ou recebimento de arquivo, etc.

Conexão de dados: utilizada para transferência de arquivos.

O servidor FTP mantém informações de estado sobre o usuário, como associar a conexão de controle com a conta do usuário e lembrar do diretório corrente.



SMTP – Simple Mail Transfer Protocol (e-mail)



	Entrada	Saída
alice	12	0 → 1
john	23	

	Entrada	Saída
bob	2 → 3	
jay	3	

Protocolos de acesso:

- **POP3** (Post Office Protocol v3) – Ler e-mail e manter na caixa de entrada; Ler e excluir.
- **IMAP** (Internet Mail Access Protocol) – Permite o gerenciamento de pastas.
- **HTTP** – Envio e recebimento de e-mails pela Web.

Questão: Você enviou um e-mail. Significa que o destinatário já tem a mensagem em sua caixa de entrada?

Um e-mail de alice@gmail.com para bob@hotmail.com

gmail.com (cliente SMTP) estabelece conexão TCP com hotmail.com (servidor SMTP)

```
S → 220 hotmail.com
C → HELO gmail.com
S → 250 Hello gmail.com, pleased to meet you
C → MAIL FROM: <alice@gmail.com>
S → 250 alice@gmail.com... Sender ok
C → RCPT TO: <bob@hotmail.com>
S → 250 bob@hotmail.com... Recipient ok
C → DATA
S → 354 Enter mail, end with "." on a line by itself
C → From: alice@gmail.com                    } Linhas de cabeçalho
C → To: bob@hotmail.com                      }
C → Subject: Redes de Computadores           }
C →
C → O protocolo SMTP ...                     } Corpo da mensagem
C → .
S → 250 Message accepted for delivery
```

Caso haja outros e-mails para enviar

```
C → MAIL FROM: ...
⋮
C → .

C → QUIT
S → 221 hotmail.com closing connection
```

C → Cliente envia S → Servidor responde

DNS – Domain Name System

https://gaia.cs.umass.edu → DNS "Resolução" → 128.119.245.12

Nome do host (hostname)

Endereço IP

Interface de Rede
128.119.245.12

host: gaia.cs.umass.edu

A tarefa principal do DNS é traduzir nomes de hosts para endereços IP.

Pode-se imaginar o seguinte...

Antes de uma requisição HTTP, FTP ou SMTP

host

IP

Servidor DNS

Na verdade é um banco de dados distribuído sobre uma hierarquia de servidores, tipicamente **UNIXes**.

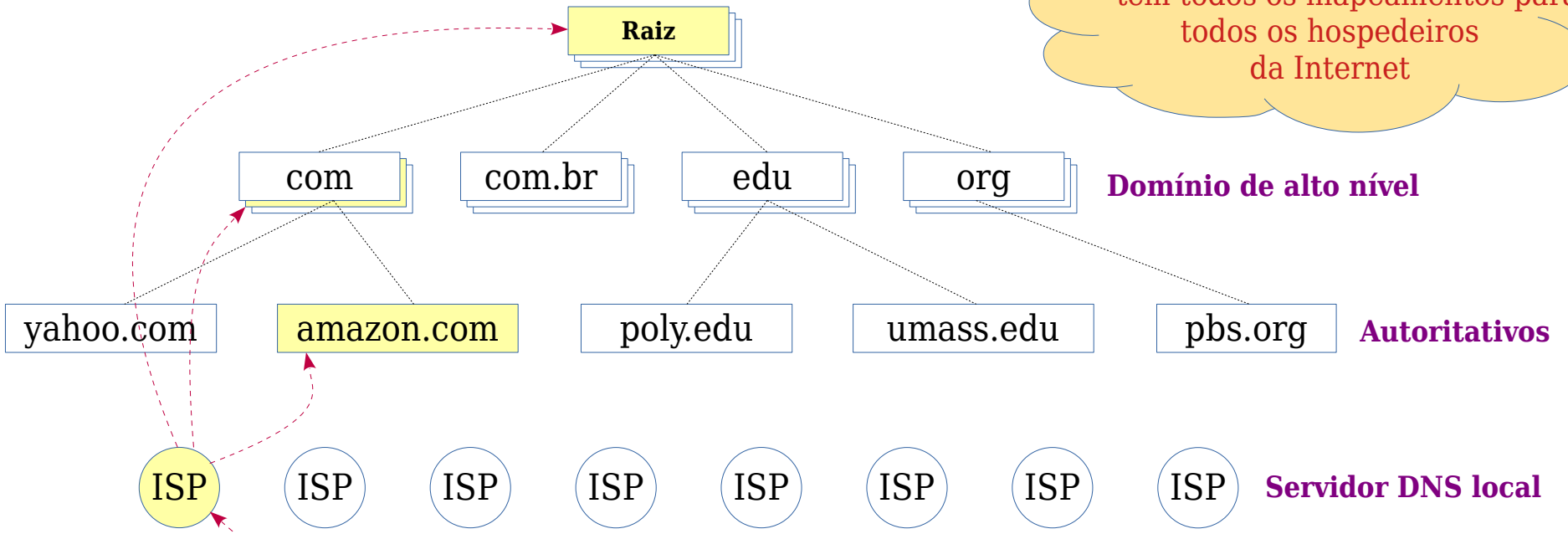
O serviço DNS utiliza o protocolo **UDP** e a porta **53**.

Serviços oferecidos pelo DNS...

- Tradução de nomes de hosts para endereços IP.
- Apelidos (aliasing): permite inclusive que os servidores Web e de e-mail tenham o mesmo apelido, mesmo sendo servidores diferentes.
- Distribuição de carga: quando há servidores replicados, o serviço DNS responde com o conjunto inteiro de endereços IP, mas faz um rodízio da ordem deles dentro de cada resposta.

Hierarquia de servidores DNS

Nenhum servidor DNS isolado tem todos os mapeamentos para todos os hospedeiros da Internet



Servidores DNS autoritativos: Servidores DNS primário e secundário da própria corporação (que possui hospedeiros públicos na Internet).

www.amazon.com

1. Requisita ao servidor DNS local o endereço de www.amazon.com.
2. Servidor DNS local contata um dos servidores raiz pedindo por uma lista de servidores de domínio de alto nível responsáveis por com.
3. Servidor DNS local contata um dos servidores de domínio de alto nível pedindo por uma lista de servidores autoritativos responsáveis por amazon.com.
4. Servidor DNS local solicita a um dos servidores autoritativos o endereço (ou endereços) IP do host www.amazon.com.

Registros DNS: (Name, Value, Type, TTL)

Consulta IP de gaia.cs.umass.edu

Servidor DNS de domínio de alto nível

Name	Value	Type
umass.edu	dns.umass.edu	NS = domain
dns.umass.edu	128.119.101.1	A = host

Servidor DNS autoritativo para umass.edu

Name	Value	Type
gaia.cs.umass.edu	128.119.245.12	A = host
umass.edu	origin-www.umass.edu	CNAME = canonical name
www.umass.edu	origin-www.umass.edu	CNAME = canonical name
umass.edu	origin-mail.umass.edu	MX = canonical name (e-mail)
origin-www.umass.edu	128.119.8.148	A = host
origin-mail.umass.edu	128.119.8.150	A = host

Cada registro de recurso DNS carrega também o tempo de vida do registro no cache: TTL (segundos)

nslookup

Consulte o seu servidor DNS local...

$ nslookup www.hotmail.com [IP-Servidor-DNS-Local]

```
Server: IP-Servidor-DNS-Local
Address: IP-Servidor-DNS-Local#53

Non-authoritative answer:
www.hotmail.com          canonical name = outlook-fd-0010.live.com.
outlook-fd-0010.live.com    canonical name = a-0010.a-msedge.net.
Name:  a-0010.a-msedge.net
Address: 204.79.197.212
Name:  a-0010.a-msedge.net
Address: 2620:1ec:c11::212
```

Consulte o servidor DNS do Google...

$ nslookup www.hotmail.com 8.8.8.8

Questões de revisão

1. Relacione algumas aplicações da Internet e os protocolos de camada de aplicação e transporte que elas usam.

2. Para uma sessão de comunicação entre um par de processos, qual processo é o cliente e qual é o servidor?

3. Em uma aplicação de compartilhamento de arquivos P2P, você concorda com a afirmação: “não existe nenhuma noção de lados cliente e servidor de uma sessão de comunicação?”

4. Que informação é usada por um processo que está rodando em um hospedeiro para identificar um processo que está rodando em outro hospedeiro?

5. Suponha que você queira fazer uma transação de um cliente remoto para um servidor da maneira mais rápida possível. Você usaria o UDP ou o TCP? Por quê?

6. Lembre-se de que o TCP pode ser aprimorado com o SSL/TLS para fornecer serviços de segurança processo a processo, incluindo a decodificação. O SSL/TLS opera na camada de transporte ou na camada de aplicação?

7. Por que HTTP, FTP, SMTP, POP3 rodam sobre TCP e não sobre UDP?

8. Por que se diz que o FTP envia informações de controle “fora da banda”?

Problema

Sugira um software de rede e descreva o protocolo da camada de aplicação necessário para atender as funcionalidades desse software. Que protocolo da camada de transporte é mais adequado para o software?

Exemplo: TDP – To Do Protocol (Afazeres)

Cliente (requisição)	Servidor (resposta)
`PUT` descrição da tarefa	`OK`
`GET` *n*	*n*. descrição da tarefa ou `NOT-FOUND`
`LIST`	1. tarefa A 2. tarefa B ⋮ ou `EMPTY`
`EXIT`	`BYE`
Qualquer requisição inválida	`ERR`