

# **AULA 02 – O PARADIGMA CLIENTE- SERVIDOR**

Por Sediane Carmem Lunardi Hernandez

1

# AGENDA

---



Introdução



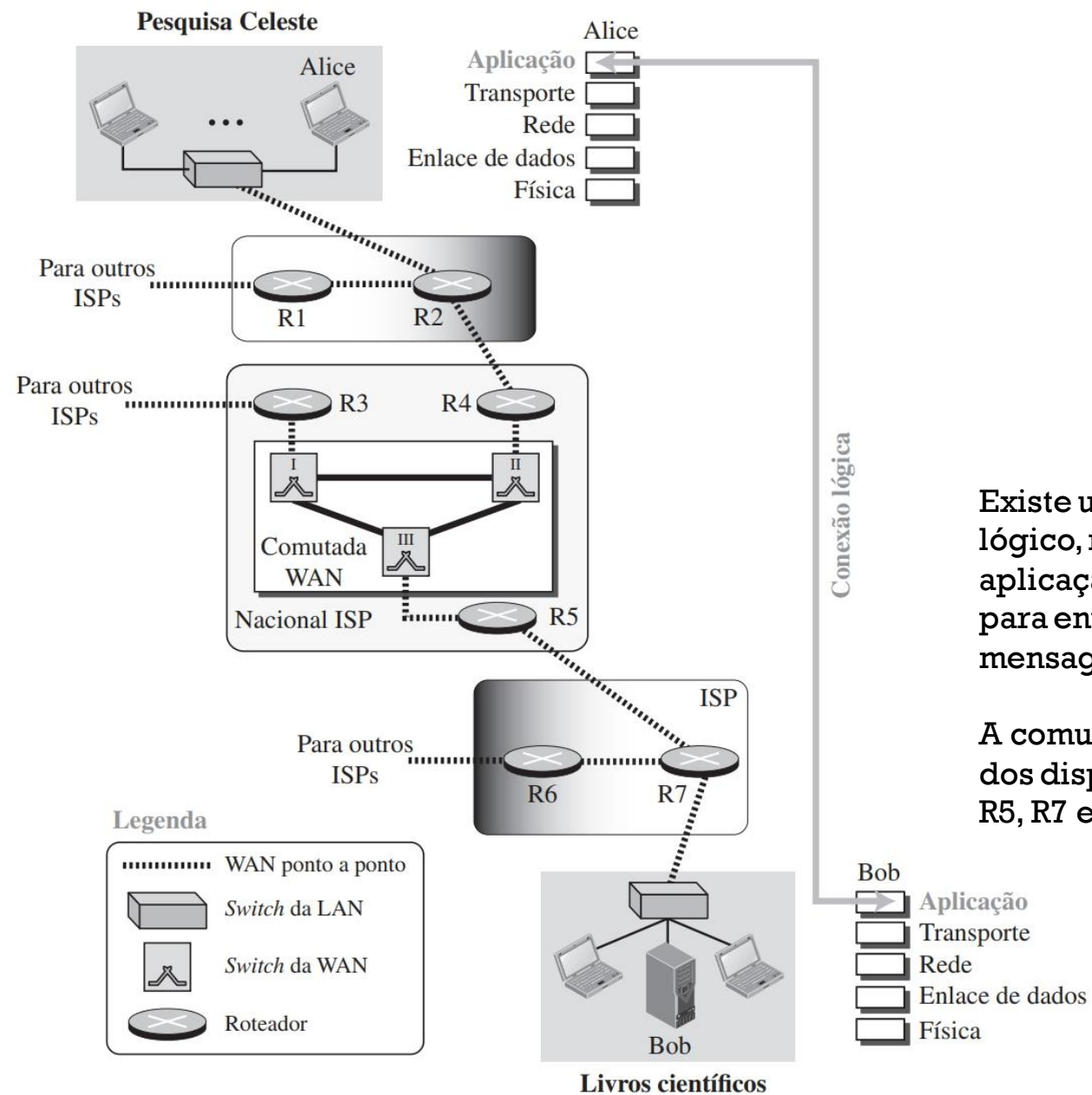
O paradigma cliente-servidor



Aplicações cliente-servidor da camada de aplicação

# 1. INTRODUÇÃO

- A camada de aplicação **é a única** que **provê serviços para o usuário**
  - As outras 4 camadas da pilha de protocolos TC/IP existem para torná-los possíveis
    - Dão suporte para essa oferta de serviços
- A comunicação na camada de aplicação é fornecida através de uma **conexão lógica entre dois processos nas duas camadas de aplicação** em que se encontram
  - é considerada a existência de uma conexão direta imaginária por meio da qual os processos podem enviar e receber mensagens



Existe um canal bidirecional lógico, na camada de aplicação, entre Alice e Bob para envio e recepção de mensagens.

A comunicação real se através dos dispositivos Alice, R2, R4, R5, R7 e Bob e canais físicos.

# 1. INTRODUÇÃO

- A camada de aplicação é flexível
  - Novos protocolos de aplicação podem ser **adicionados** à Internet
  - Outros podem ser **retirados** ou **substituídos**



Devem usar os serviços oferecidos pela camada inferior

- Protocolos da camada de aplicação podem ser padronizados ou não

# INTRODUÇÃO



- Para que **dois processos nas duas camadas de aplicação** enviem mensagens um ao outro por meio da infraestrutura da Internet, o que é necessário?
- Ambos os aplicativos devem ser capazes de solicitar e prestar serviços ou cada um deve desempenhar apenas uma dessas funções?

## 2. PARADIGMAS DA CAMADA DE APLICAÇÃO

- Dois paradigmas foram desenvolvidos desde o surgimento da Internet:
  - o paradigma peer-to-peer
  - o paradigma cliente-servidor



# 2.1 PARADIGMA PEER-TO-PEER

- Um computador conectado à Internet pode prover serviços em dado momento (processo-servidor) e receber serviços (processo-cliente) em outro; pode até mesmo prover e receber serviços ao mesmo tempo
  - Processo-servidor não precisa executar o tempo todo
- Surgiu para responder às necessidades de novas aplicações
  - Por exemplo, telefonia via Internet, compartilhamento de arquivos
- Aplicativos que utilizam esse paradigma:
  - BitTorrent
  - Skype
  - IPTV
  - Telefonia por Internet

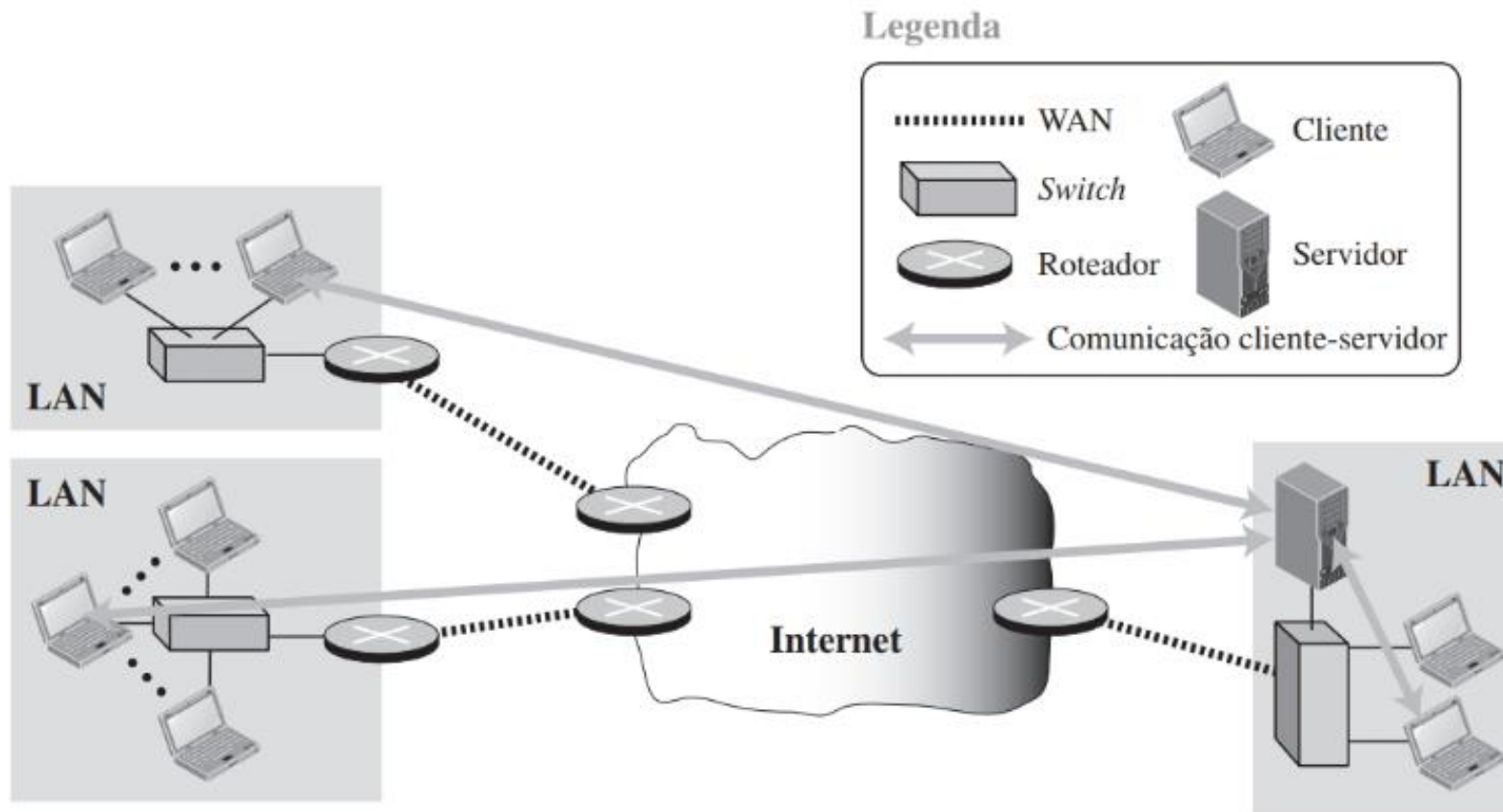




## 2.2 O PARADIGMA CLIENTE-SERVIDOR

- Provedor de serviços é um aplicativo chamado **processo-servidor**
  - **executado continuamente** (o tempo todo) à espera de outro aplicativo, chamado processo-cliente
- **Processo-cliente cria uma conexão** através da Internet e solicita o serviço oferecido pelo processo-servidor
  - o processo-cliente é iniciado quando o cliente precisa receber o serviço
- Funções bem definidas para cada um:
  - Programa-cliente **NÃO** pode executar como um programa-servidor
  - Programa-servidor **NÃO** pode executar como um programa-cliente

## 2.2 O PARADIGMA CLIENTE-SERVIDOR (CONT.)



## 2.2 O PARADIGMA CLIENTE-SERVIDOR

- Em resumo, no paradigma cliente-servidor:
  - a **comunicação** na camada de aplicação se dá entre dois aplicativos em execução denominados processos: um cliente e um servidor.
    - **cliente** é um programa em execução que inicia a comunicação enviando um pedido (ou solicitação);
    - **servidor** é outro programa que aguarda pedidos de clientes
      - **trata o pedido recebido, prepara um resultado e envia o resultado de volta ao cliente.**
  - servidor deve estar sendo executado quando o pedido de um cliente chegar
  - cliente pode ser executado somente quando necessário.

Ou seja, se tivermos dois computadores conectados um ao outro em algum lugar, podemos executar o processo-cliente em um deles e o processo-servidor no outro. No entanto, precisamos ter cuidado para que o programa-servidor seja iniciado antes do início da execução do programa-cliente.

## 2.2 O PARADIGMA CLIENTE-SERVIDOR (CONT.)

- Vários serviços tradicionais usam esse paradigma:
  - **A World Wide Web (WWW) e seu instrumento de acesso, o Protocolo de Transferência de Hipertexto (HTTP – *HyperText Transfer Protocol*)**
  - O Protocolo de Transferência de Arquivos (FTP – *File Transfer Protocol*)
  - O *Secure Shell* (SSH)
  - O servidor de e-mail; e, diversos outros.

# INTERFACE DE PROGRAMAS DE APLICATIVO



- Um processo-cliente precisa se comunicar com um processo-servidor
  - O que é necessário?
    - Um **conjunto de instruções**



# INTERFACE DE PROGRAMAS DE APLICATIVO

- **Conjunto de instruções DEVE** ser utilizada para que às quatro camadas mais baixas da pilha de protocolos TCP/IP:
  - **abram uma conexão** para comunicação
  - **enviem os dados** à outra estação
  - **recebam os dados** dela
  - **fechem a conexão**



Interface de Programação de Aplicativos  
ou Interface para Programas de  
Aplicação (API – *Application  
Programming Interface*)



# a) INTERFACE DE PROGRAMAS DE APLICATIVO

■ ...

- Uma Interface de Programação de Aplicativos (API – *Application Programming Interface*) é um **conjunto de instruções entre duas entidades**
  - uma das entidades é **o processo na camada de aplicação**; e,
  - a outra é o **sistema operacional** que encapsula as quatro primeiras camadas da pilha de protocolos TCP/IP

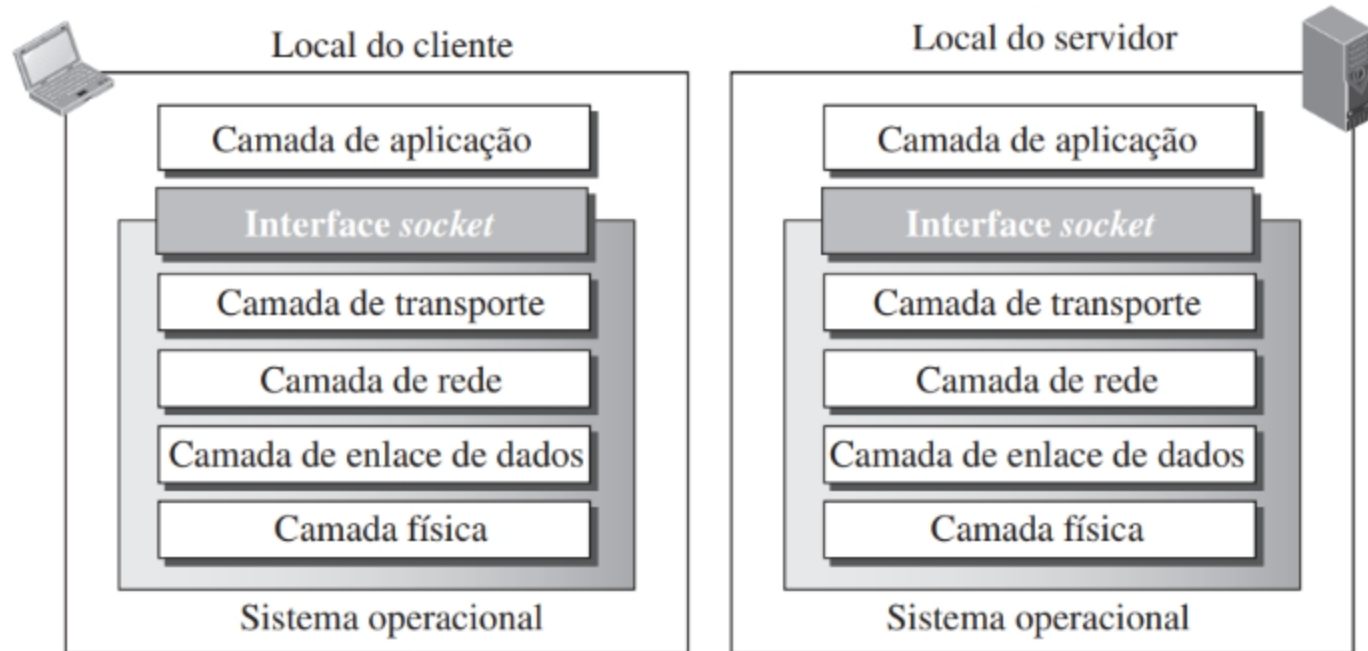


**Processos** em execução na camada de aplicação **são capazes de se comunicar com o sistema operacional** ao **enviar e receber mensagens pela Internet**.

# a) INTERFACE DE PROGRAMAS DE APLICATIVO

- Várias APIs foram projetadas para comunicação
  - Uma delas é a **interface socket**
- A **interface socket** surgiu no início de 1980 na Universidade de Berkeley como parte de um ambiente UNIX
  - É um conjunto de instruções que fornecem comunicação entre a camada de aplicação e o sistema operacional
  - É um conjunto de instruções que podem ser usadas por um processo para se comunicar com outro.

# INTERFACE SOCKET

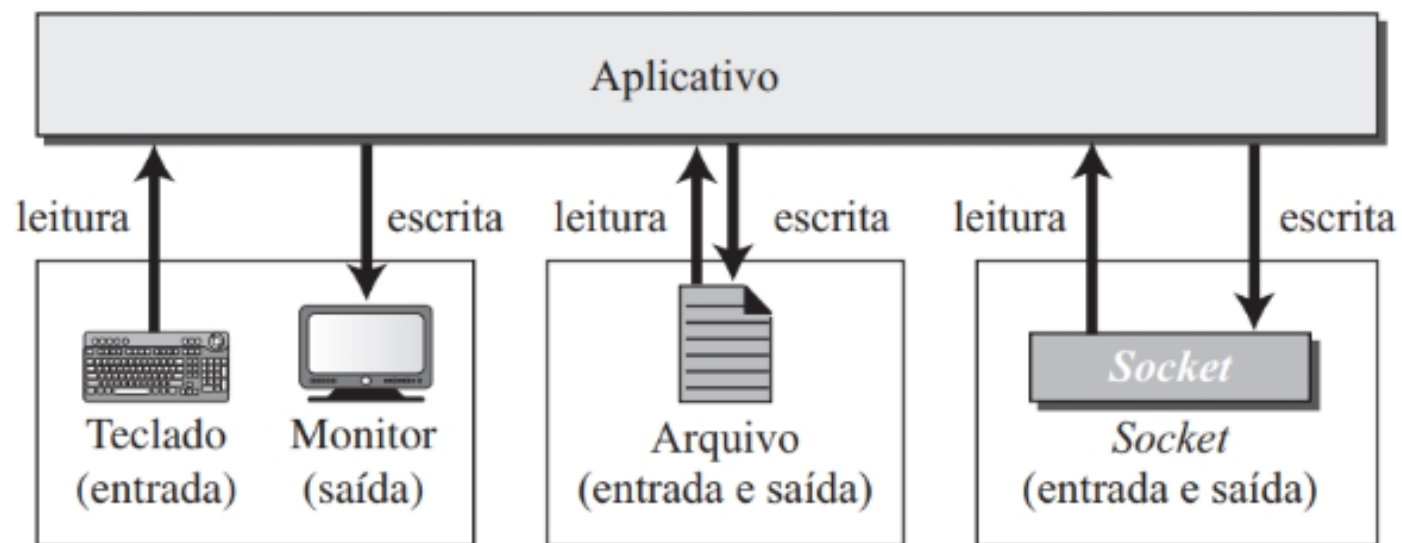


# MAS, NA PRÁTICA, O QUE É O SOCKET?

É uma **estrutura de dados criada** e **utilizada** pelo processo cliente e pelo processo servidor

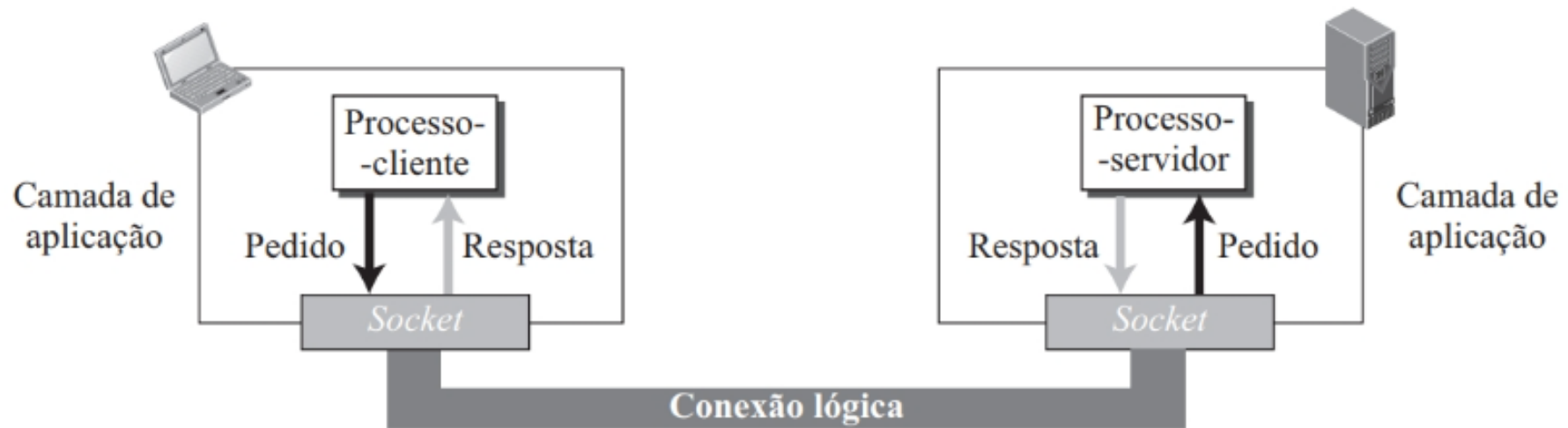


## Comparação socket



# SOCKET

- O socket não é uma entidade física, mas sim uma abstração

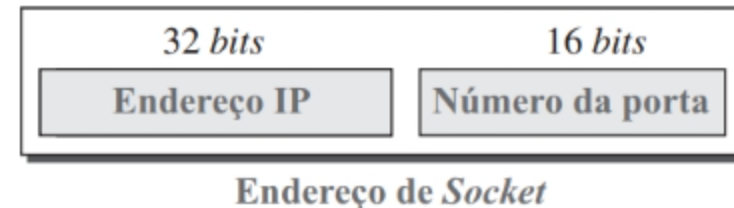


# SOCKET

- A comunicação entre um processo-cliente e um processo-servidor, na camada de aplicação, se limita à comunicação entre dois sockets, criados nos dois sistemas finais
  - **Para o cliente:** por meio do socket, o servidor recebe o pedido e envia a resposta
  - **Para o servidor:** por meio do socket, o cliente envia um pedido e precisa da resposta
- Para a comunicação cliente-servidor os **endereços** de origem e destino precisam ser definidos corretamente.

# ENDEREÇOS DE SOCKET

- **Interação** entre **cliente** e **servidor** é bidirecional
  - Precisa de um **par de endereços**
    - Endereço do socket local
    - Endereço do socket remoto
- Um endereço de socket deve:
  1. definir o **computador** (no qual um cliente ou um servidor está sendo executado)
  2. definir a **porta de comunicação** (0 e 65.535) no dispositivo que estiver





Como um cliente ou um servidor descobrem um par de endereços de socket para a comunicação?

---

O servidor precisa de um endereço de socket local (servidor) e de um endereço de socket remoto (cliente) para a comunicação

O cliente também precisa de um endereço de socket local (cliente) e de um endereço de socket remoto (servidor) para a comunicação



# DESCOBRINDO ENDEREÇOS DE SOCKET

- **Lado cliente:**

- O endereço de socket local (cliente) é fornecido pelo sistema operacional
  - o sistema operacional sabe o **endereço IP do computador no qual o processo-cliente está sendo executado**
    - **o número da porta é atribuído a um processo-cliente cada vez que o processo necessita iniciar a comunicação**
      - sistema operacional precisa garantir que o novo número de porta não seja usado por qualquer outro processo-cliente já em execução
- O endereço de socket remoto (servidor) para um cliente precisa ser descoberto
  - Executar a aplicação cliente envolve descobrir o IP do servidor
    - Sabe-se a porta, mas não o endereço IP que deve ser descoberto (DNS – *Domain Name System*)
      - DNS mapeia o **nome do servidor PARA** o **endereço IP** do computador no qual o servidor está sendo executado (aplicativo meu IP)

# DESCOBRINDO ENDEREÇOS DE SOCKET

- **Lado servidor:**

- O endereço de **socket local** (servidor) é fornecido pelo sistema operacional
  - o sistema operacional sabe o **endereço IP do computador no qual o processo-servidor está sendo executado**
    - **o número da porta de um processo-servidor precisa ser atribuído** (padronizado ou não)
      - **Protocolo de Transferência de Hipertexto** (HTTP – Protocolo de Transferência de Hipertexto) - porta 80, que não pode ser utilizado por outro processo.
- O endereço de **socket remoto** (cliente) para um servidor é o endereço de socket do cliente que inicia a conexão
  - O endereço de socket do cliente está contido no pacote de solicitação enviado para o servidor
    - torna-se o endereço de socket remoto usado para responder àquele cliente

# ATIVIDADE PRÁTICA

## 1. Descobrindo o endereço IP da sua máquina

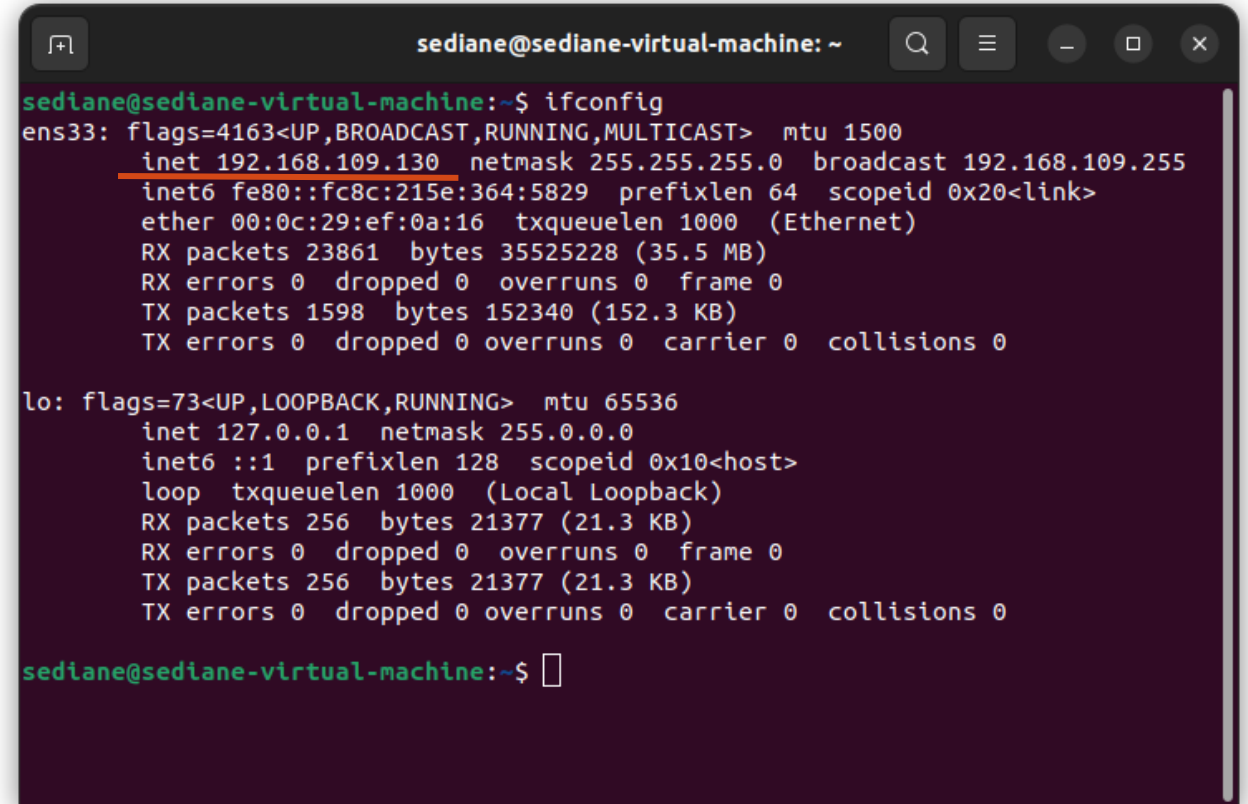
- No terminal Linux digite:

```
$ ifconfig
```

## 2. Descobrindo as portas em que servidores ficam aguardando solicitações de clientes em seu computador local

- No terminal Linux digite:

```
$ netstat -l
```



```
sediane@sediane-virtual-machine: ~  
sediane@sediane-virtual-machine:~$ ifconfig  
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  
    inet 192.168.109.130  netmask 255.255.255.0  broadcast 192.168.109.255  
    inet6 fe80::fc8c:215e:364:5829  prefixlen 64  scopeid 0x20<link>  
    ether 00:0c:29:ef:0a:16  txqueuelen 1000  (Ethernet)  
    RX packets 23861  bytes 35525228 (35.5 MB)  
    RX errors 0  dropped 0  overruns 0  frame 0  
    TX packets 1598  bytes 152340 (152.3 KB)  
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536  
    inet 127.0.0.1  netmask 255.0.0.0  
    inet6 ::1  prefixlen 128  scopeid 0x10<host>  
    loop txqueuelen 1000  (Local Loopback)  
    RX packets 256  bytes 21377 (21.3 KB)  
    RX errors 0  dropped 0  overruns 0  frame 0  
    TX packets 256  bytes 21377 (21.3 KB)  
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0  
  
sediane@sediane-virtual-machine:~$
```

# ATIVIDADE PRÁTICA

- A. Colocar um servidor de arquivos implementado em Python para executar em porta específica do computador e conectar vários clientes a ele (ver código).
- B. Vamos instalar um servidor de páginas web em nosso computador?

Mas primeiro o que eu preciso saber?

- 1. O que é a Web (conceitos introdutórios)?
  - Localizado Uniforme de Recursos - URL





# 3. APLICAÇÕES CLIENTE-SERVIDOR PADRONIZADAS

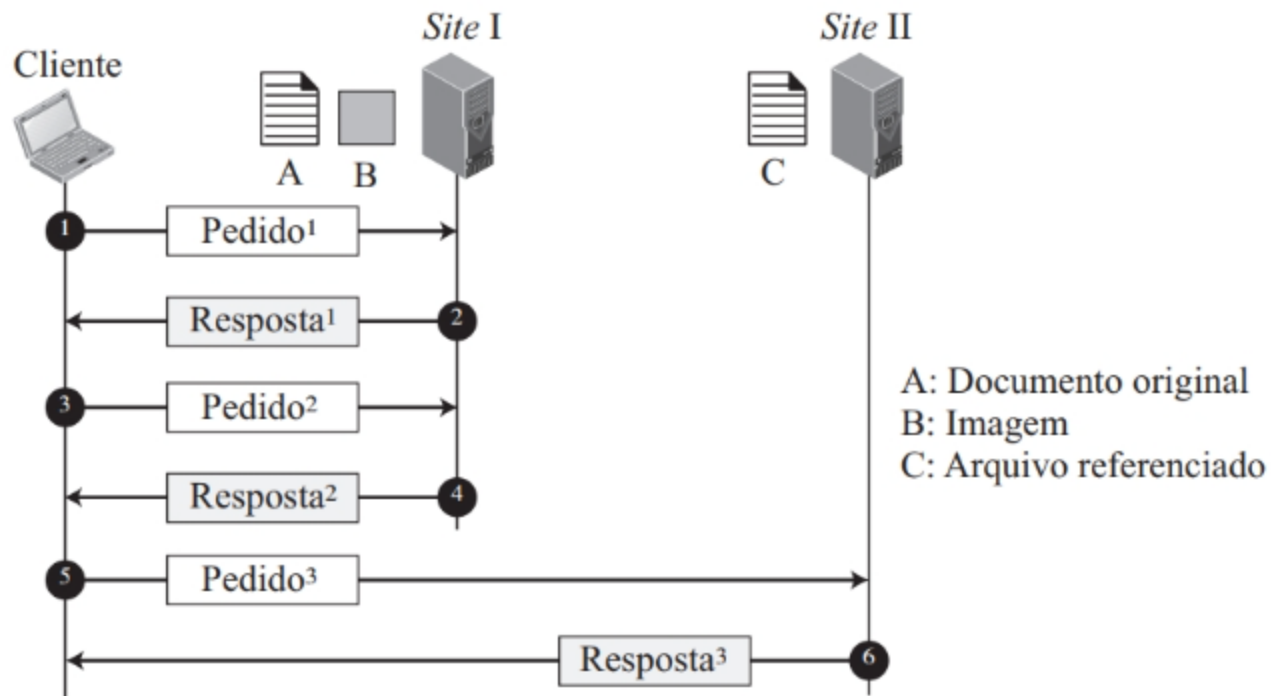
- Desde a criação da Internet, vários programas cliente-servidor foram desenvolvidos. Exemplos:
  - World Wide Web (www)
  - Protocolo de Transferência de Hipertexto (HTTP – *HyperText Transfer Protocol*)
  - Protocolo de Transferência de Arquivos (FTP – *File Transfer Protocol*)
  - Correio eletrônico
  - Telnet
  - Secure Shell (SSH)
  - Sistema de Nomes de Domínio (DNS – *Domain Name System*)

# 3.1 WORLD WIDE WEB - WWW

- A Web é um serviço cliente-servidor distribuído
  - um cliente usando um navegador (browser) pode acessar o serviço por meio de um servidor
- Serviço distribuído entre muitos locais chamados site
  - cada site possui um ou mais documentos, chamados de páginas Web
    - cada página Web pode conter algumas ligações ou links para outras páginas do mesmo ou de outros sites
      - cada uma delas é um arquivo com um nome e um endereço



# 3.1 WORLD WIDE WEB - WWW (CONT.)



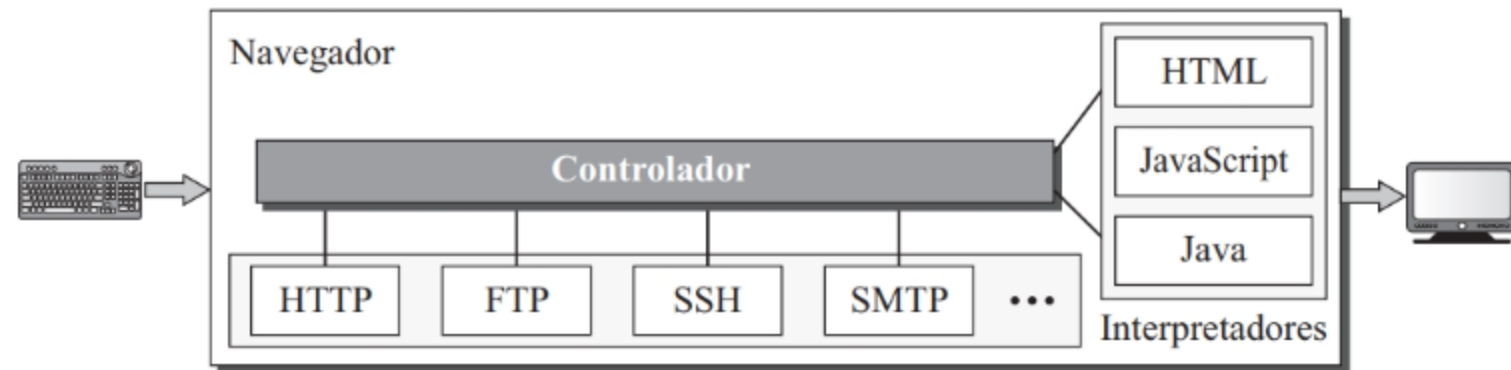
**Cliente web** (navegador – browser) - interpretam e exibem uma página web

**Servidor web** – armazena páginas web. Pedido de página web chega ao servidor e é enviado ao cliente

# 3.1 WORLD WIDE WEB - WWW (CONT.)

**Navegador** geralmente tem três partes:

1. um controlador
2. protocolos cliente
3. interpretadores



# DETALHES...

- **O CONTROLADOR:**

- **recebe** a entrada do teclado ou do mouse



- **usa** os protocolos cliente para acessar o documento



- **exibe** o documento na tela usando um dos interpretadores



# DÚVIDA

Controlador do navegador recebe os dados como/onde?

# LOCALIZADOR UNIFORME DE RECURSOS (URL)

- Uma página Web precisa ter um **identificador único**
  - Distinção entre páginas Web

- Identificador

i. .

ii. Host

iii. Porta

iv. Caminho

antes



a) Que tipo de aplicação cliente-servidor queremos usar? Ou seja, que protocolo?

Para combinar essas quatro partes, foi criado o **Localizador Uniforme de Recursos** (URL – *Uniform Resource Locator*), que usa três diferentes separadores entre as quatro partes, conforme ilustrado a seguir:

protocolo://host/caminho

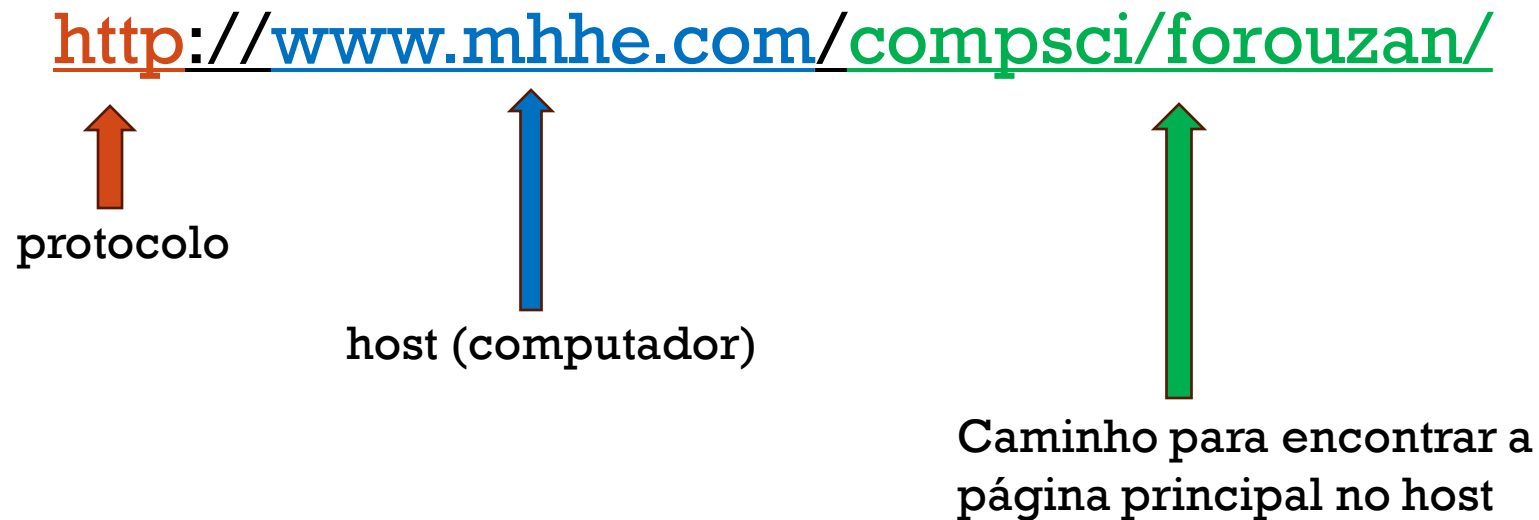
Usado na maioria dos casos

protocolo://host:porta/caminho

Usado quando o número de porta é necessário

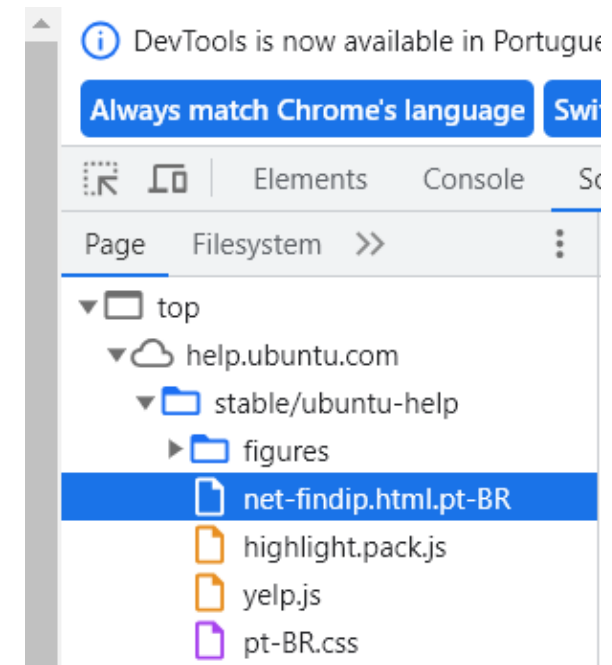
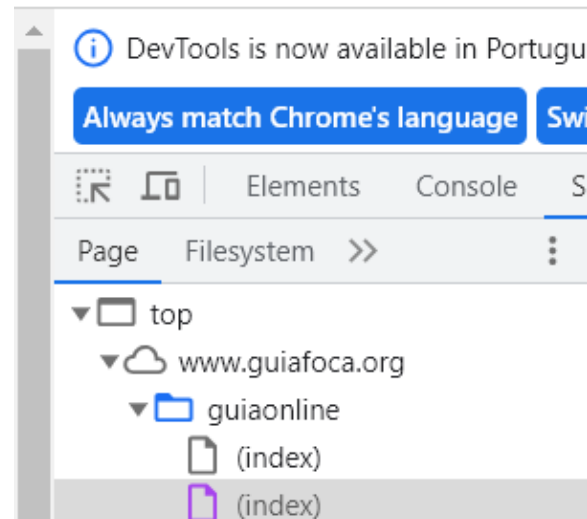
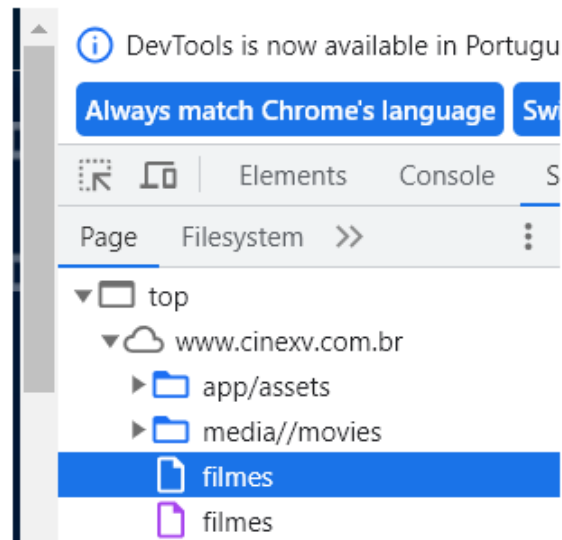
# LOCALIZADOR UNIFORME DE RECURSOS (URL)

- Defina os identificadores da página web abaixo:



# LOCALIZADOR UNIFORME DE RECURSOS (URL)

- Defina os identificadores das páginas web abaixo:
  - a) <https://www.cinexv.com.br/filmes>
  - b) <https://www.guiafoca.org/guiaonline/>
  - c) <https://help.ubuntu.com/stable/ubuntu-help/net-findip.html.pt-BR>





# DOCUMENTO WEB

- Os documentos na WWW podem ser agrupados em três grandes categorias:
  1. Estáticos
    - documentos de conteúdo fixo criados e armazenados em um servidor
    - cliente pode apenas obter uma cópia do documento
    - criados com as linguagens (dentre outras): Linguagem de Marcação de Hipertexto (HTML – *Hypertext Markup Language*), Linguagem de Marcação Extensível (XML – *Extensible Markup Language*), Linguagem de Estilo Extensível (XSL – *Extensible Style Language*) e Linguagem de Marcação de Hipertexto Extensível (XHTML – *Extensible Hy-pertext Markup Language*)
  2. Dinâmicos
    - um documento dinâmico é criado por um servidor Web sempre que um navegador solicita o documento
      - quando uma solicitação chega, o servidor Web executa um aplicativo ou um script que cria o documento dinamicamente
      - o servidor retorna o resultado do programa ou script como resposta para o navegador que solicitou o documento (conteúdo de um documento dinâmico pode variar de um pedido para outro)
      - exemplos de linguagem: JSP – *Java Server Pages* ou ASP – *Active Server Pages*
  3. Ativos
    - programa ou script é executado no lado cliente
      - Exemplo: Applet Java

# REFERÊNCIAS

- FOROUZAN, Behrouz A.; MOSHARRAF, Firouz. **Redes de computadores**. Porto Alegre: Grupo A, 2013. E-book. ISBN 9788580551693. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788580551693/>. Acesso em: 20 jul. 2023.