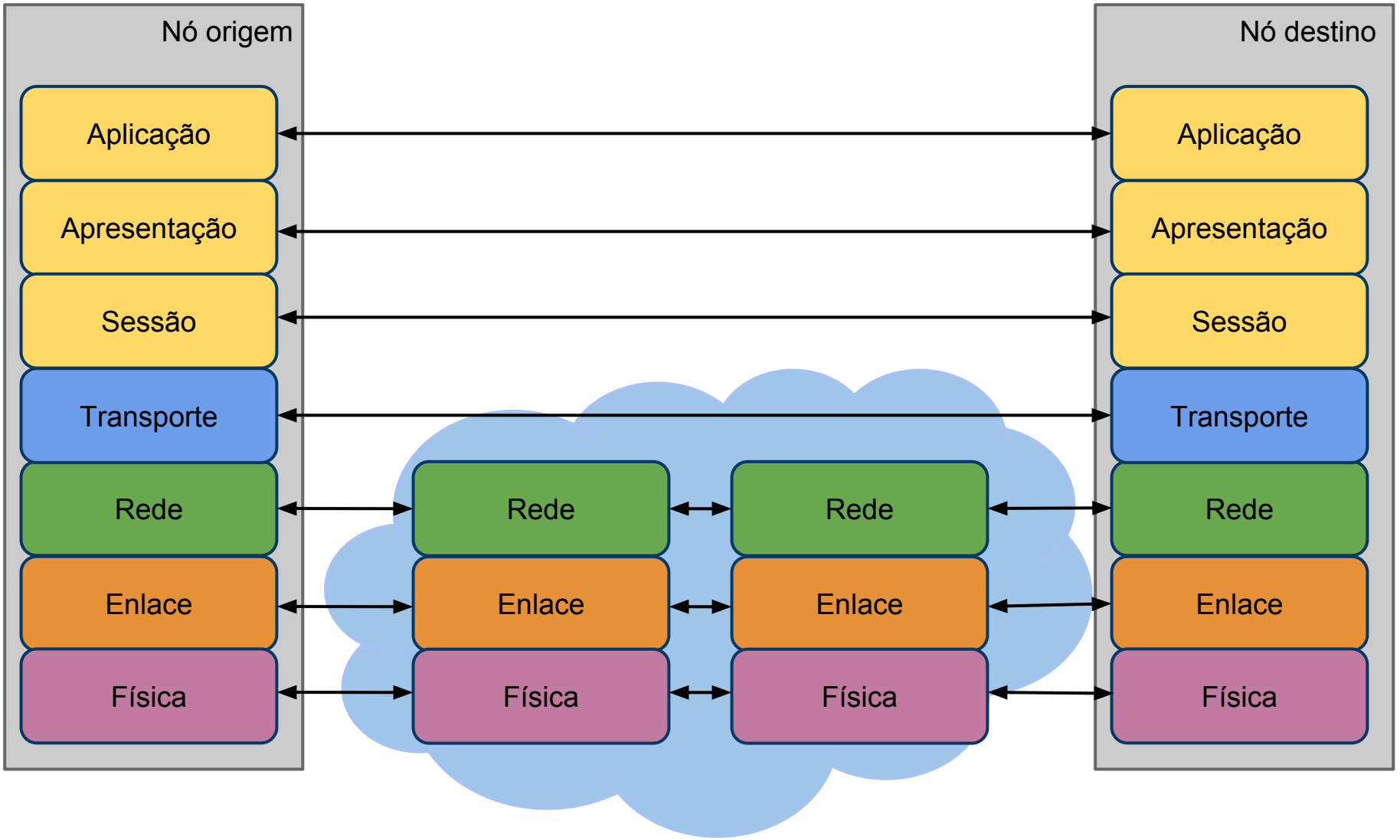
Modelo Cliente/Servidor e Introdução a Sockets

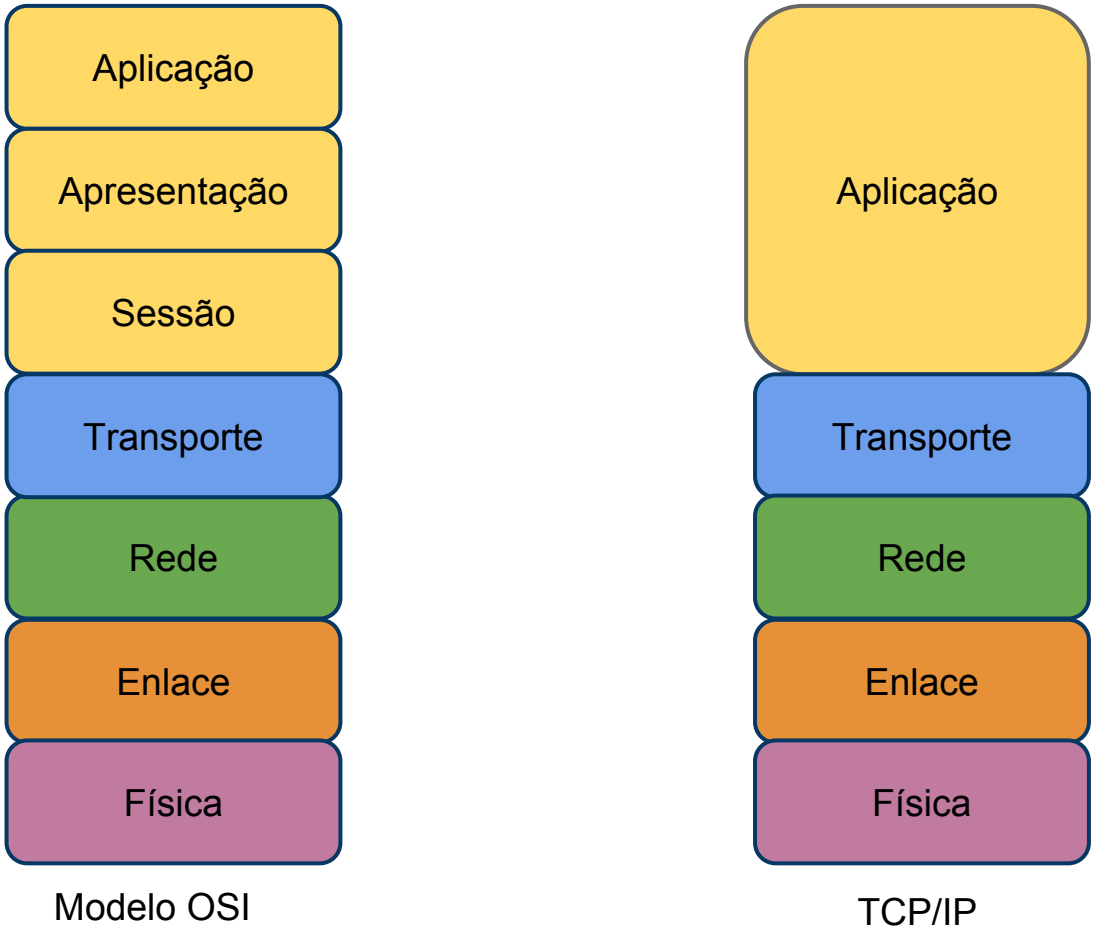
MC 833 – Programação em Redes de Computadores
Instituto de Computação – UNICAMP

Juliana Freitag Borin

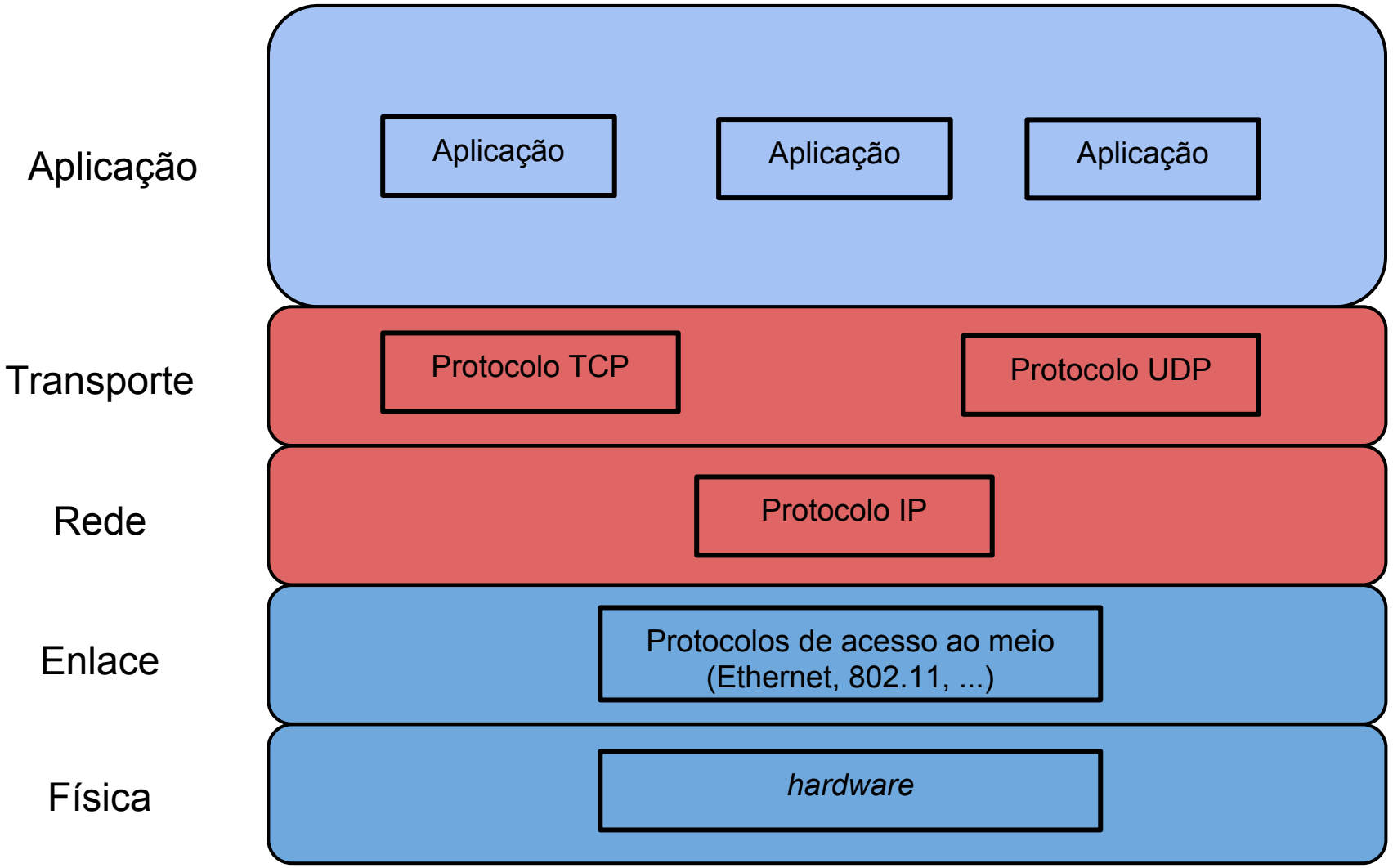
O modelo OSI de 7 camadas



OSI x TCP/IP

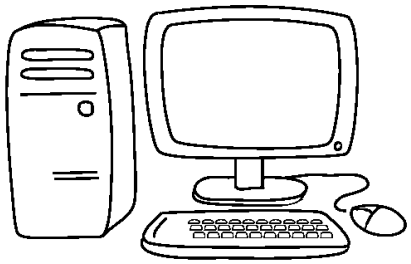

Aplicação
Apresentação
Sessão
Transporte
Rede
Enlace
Física
Modelo OSI
Aplicação
Transporte
Rede
Enlace
Física
TCP/IP

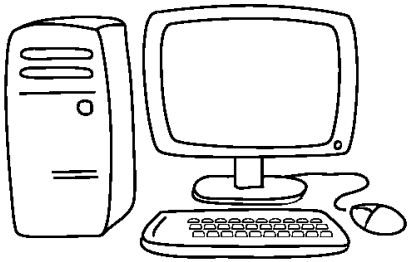
Foco nas camadas de transporte e rede



Modelo Cliente/Servidor

- Principal motivação: “Problema do encontro” (*Rendezvous problem*)





Modelo Cliente/Servidor

- Principal motivação: “Problema do encontro” (*Rendezvous problem*)

- Modelo: para qualquer par de aplicações que se comunicam, um dos lados deve iniciar a execução e esperar (indefinidamente) até ser contactado pelo outro lado.

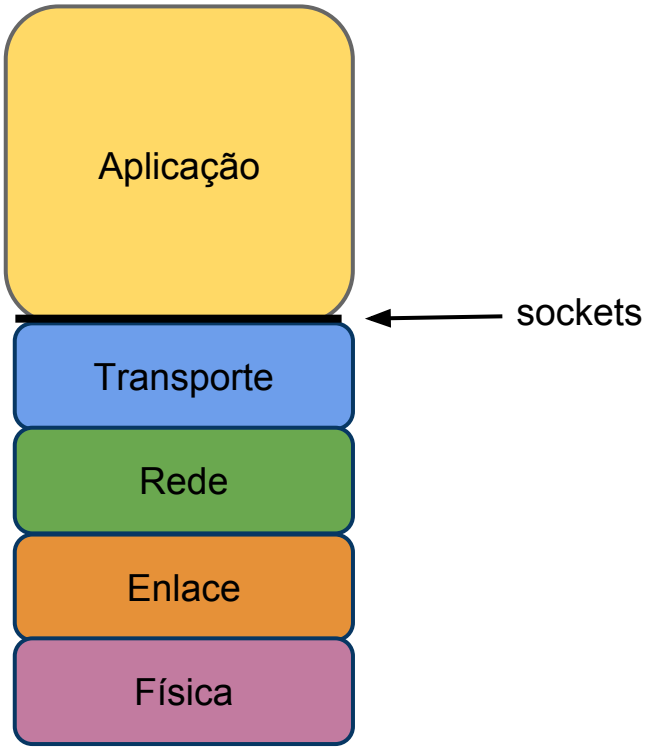
Modelo Cliente/Servidor

- De maneira geral, uma aplicação que inicia uma comunicação par-a-par é chamada **cliente**.

- Comparativamente, um **servidor** é um programa que espera por requisições de um cliente.

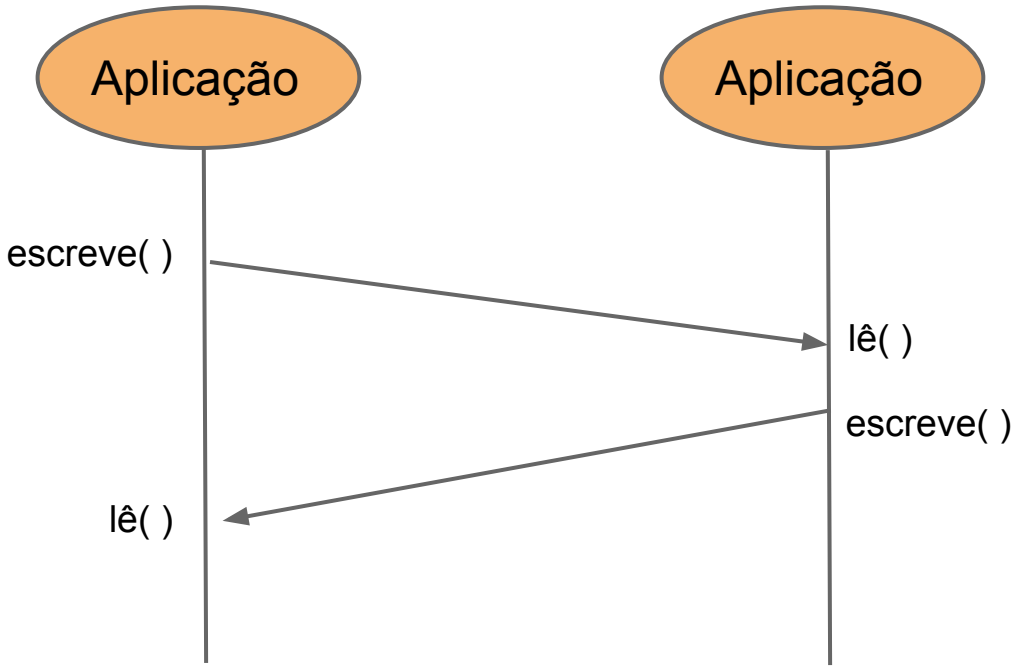
Comunicação entre cliente e servidor

- Cliente e servidor usam comunicação entre processos (*IPC - inter-process communication*) para conversar entre si.
- Há vários mecanismos de IPC - focaremos em *sockets* (Berkeley Unix).
- Sockets - API padrão para TCP/IP IPC



TCP/IP é um sistema de arquivos?

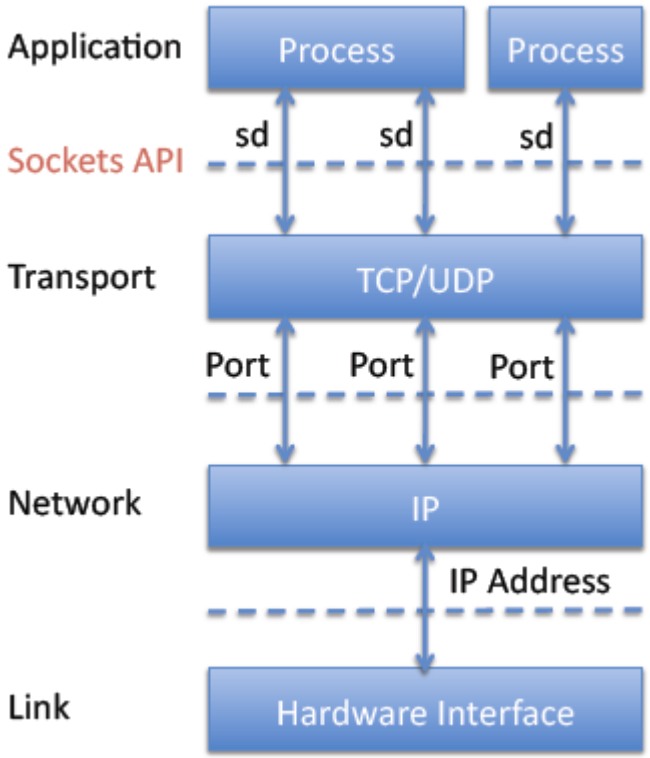
- Sockets trata o TCP/IP como se fosse um sistema de arquivos
  - rotinas para abrir e fechar uma conexão
  - rotinas para ler e escrever



Sockets e TCP/IP

- TCP/IP
  - cada ponto final é identificado por uma tupla: (porta TCP, endereço IP)
  - a conexão entre dois pontos finais é identificada pelo par [(IP, porta)$_{origem}$, (IP, porta)$_{destino}$]

- Em sistemas Unix todo fluxo de E/S é identificado por um descritor
  - socket mapeia um descritor para um ponto final
  - através da conexão entre sockets é possível conectar pontos finais e fazer operações de E/S



Sockets API para o cliente

```
int socket(int domain, int type, int protocol)
```

- Retorna um descritor associado com um novo ponto final.

```
int bind(int sd, struct sockaddr *addr, u_int addr_len)
```

- Mapeia endereço/porta para um descritor de socket (sd)

- Opcional para o cliente - pode deixar o kernel escolher uma porta disponível com o endereço IP padrão.

```
int connect (int sd, struct sockaddr *addr, u_int
addr_len)
```

- Conecta com a porta + endereço do destino

```
int send(int sd, void *buf, int len, int flags)
```

```
int recv(int sd, void *buf, int len, int flags)
```

- Comunicação

```
int shutdown(int sd, int how)
```

- Finalização parcial ou completa da conexão

Sockets API para o Servidor

```
int socket(int domain, int type, int protocol)
int bind(int sd, struct sockaddr *addr, u_int addr_len)
```

```
int listen(int sd, int backlog)
```

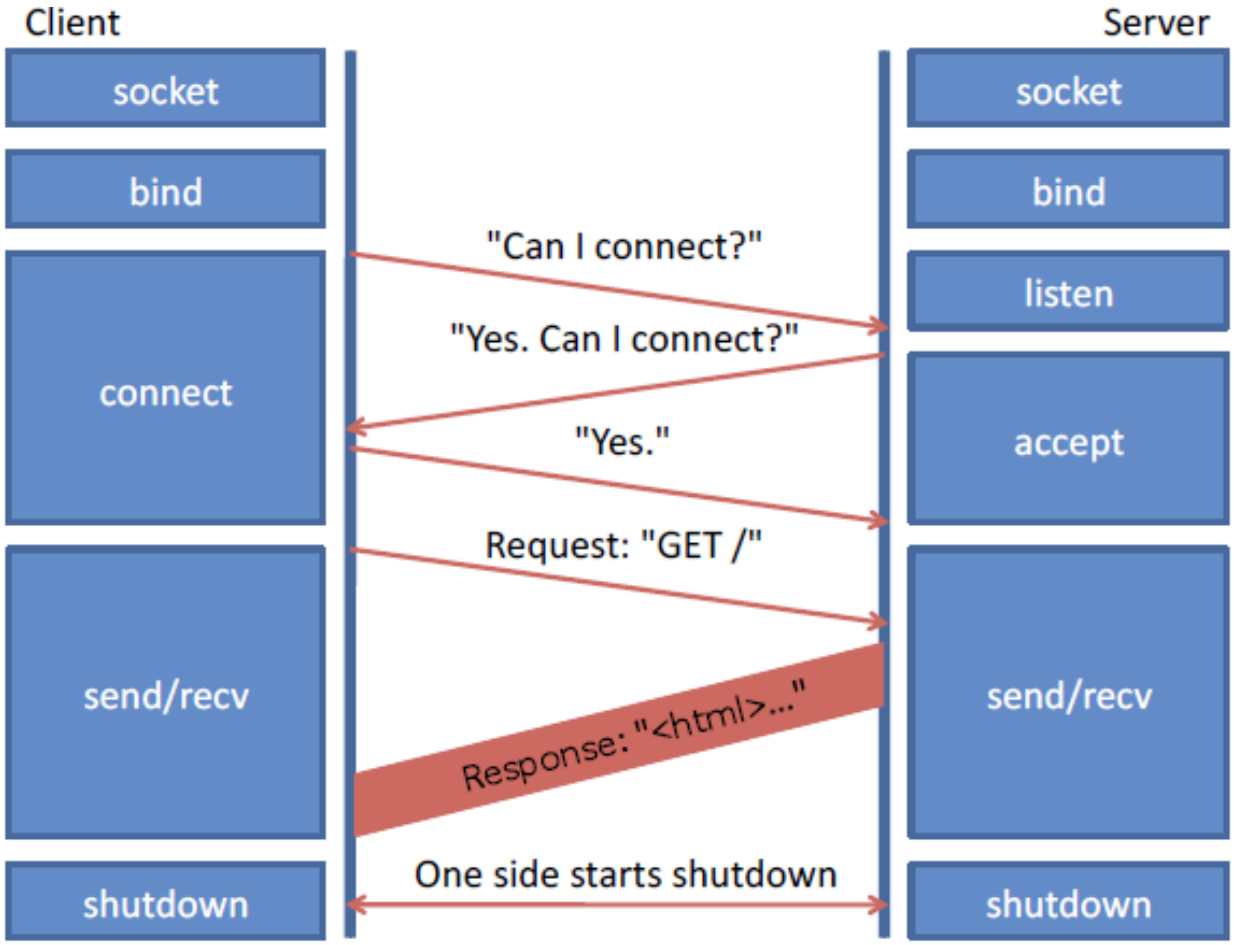
- Espera pela conexão de um cliente a essa porta

```
int accept (int sd, struct sockaddr *addr, u_int
*addr_len)
```

- - Aceita uma conexão retornando um novo descritor para o par [(IP, porta)$_{origem}$, (IP, porta)$_{destino}$]

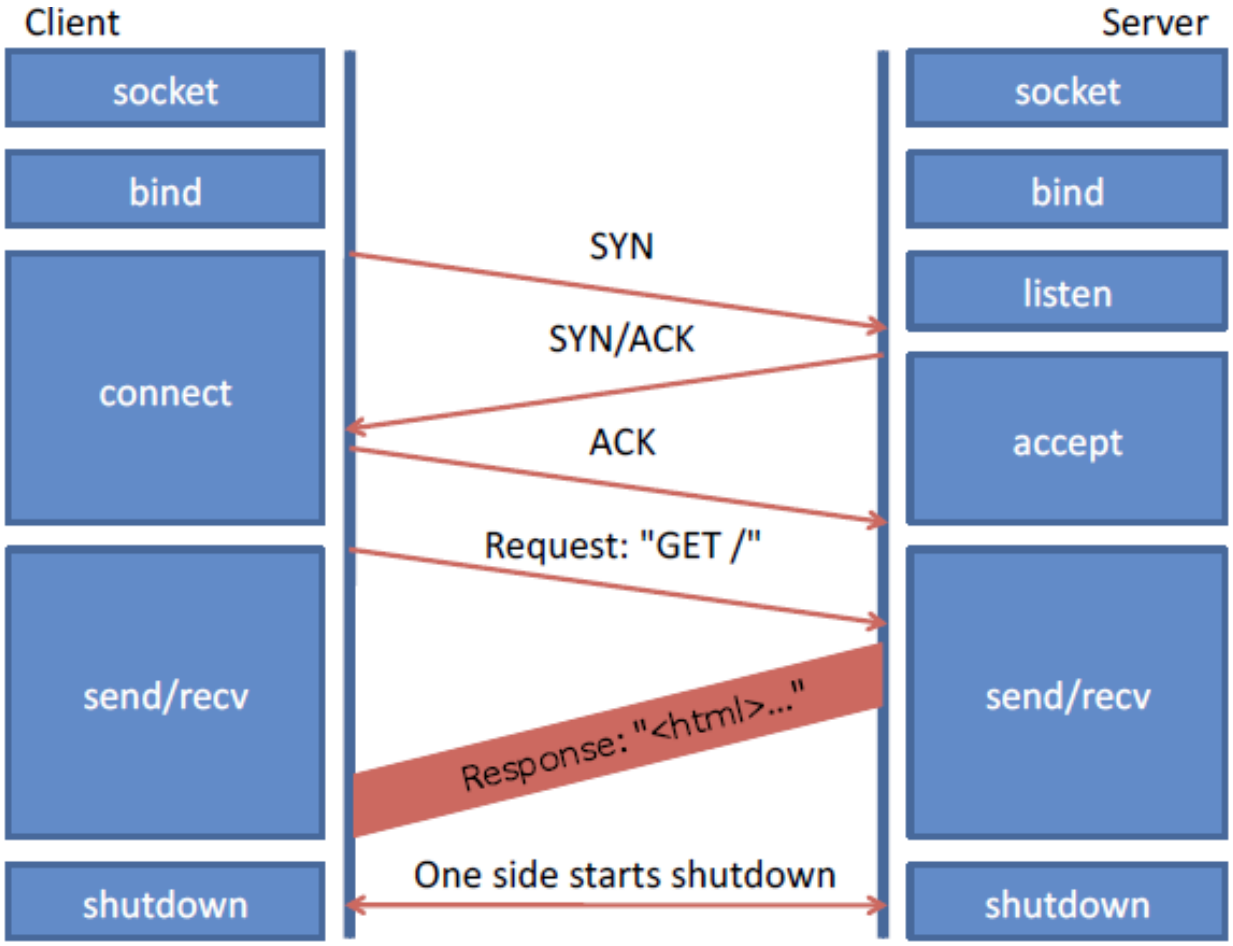
```
int send(int sd, void *buf, int len, int flags)
int recv(int sd, void *buf, int len, int flags)
int shutdown(int sd, int how)
```

Exemplo de interação entre um par cliente-servidor



Baseado nos slides de Ben Nham - Introduction to Computer Networking - Stanford University

Exemplo de interação entre um par cliente-servidor



Baseado nos slides de Ben Nham - Introduction to Computer Networking - Stanford University