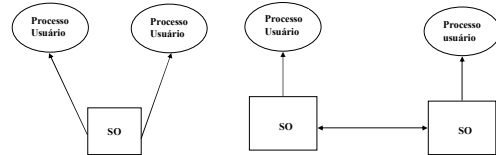


## COMUNICAÇÃO ENTRE PROCESSOS

### Sockets

## Sockets - Conceitos Básicos

- Sockets são uma forma de IPC (InterProcess Communication) que fornece comunicação entre processos residentes em sistema único ou processos residentes em sistemas remotos.



IPC em um mesmo sistema

Comunicação de Processos Remotos

## Domínios e Protocolos

- Sockets criados por diferentes programas são referenciados através de nomes
- Esses nomes devem ser traduzidos em endereços
- O espaço no qual o endereço é especificado é chamado de domínio
- Domínios básicos:
  - INTERNET (AF\_INET) - os endereços consistem do end. de rede da máquina e da identificação do no. da porta, o que permite a comunicação entre processos de sistemas diferentes
  - Unix (AF\_UNIX) - os processos se comunicam referenciando um *pathname*, dentro do espaço de nomes do sistema de arquivos

## Tipos de Sockets

- STREAM SOCKET**
  - provê seqüenciamento e fluxo bidirecional.
  - transmite dados sobre uma base confiável
  - no domínio UNIX, trabalha igual a um pipe.
  - no domínio INTERNET é implementado sobre TCP/IP.
- DATAGRAM SOCKET**
  - suporta fluxo de dados bidirecional
  - não oferece um serviço confiável.
  - Mensagens duplicadas, perdidas, e em ordem diferente (não seqüenciadas) são problemas que podem aparecer.
- RAW SOCKET**
  - permite o acesso a interface de protocolos de rede.
  - disponível para usuários avançados e que possuam autoridade de usuário root.
  - permite acesso direto a protocolos de comunicação de baixo nível.
  - permite a construção de novos protocolos sobre os protocolos de baixo nível já existentes

## Domínios e Protocolos

- Domínio Internet**
  - Implementação Unix dos protocolos TCP ou UDP
  - Consiste de:
    - end. de rede da máquina
    - identificação do no. da porta
  - Permite a comunicação entre máquinas diferentes
  - Conexões sob a forma de sockets do tipo *stream* e do tipo *datagramas*
- Portas**
  - "Endereço" para um processo comunicante
  - Inteiro de 16 bits (definido pelo usuário)
  - Portas 1 a 1023 são do sistema
  - Portas de TCP independentes das de UDP

## Domínios e Protocolos

- Protocolo TCP**
  - Transmission Control Protocol*
  - Para comunicação longa (conexão)
  - Confiável
  - Baixo desempenho em comunicações curtas (?)
  - Usos típicos:
    - login remoto
    - transferência de arquivo
- Protocolo UDP**
  - User Datagram Protocol*
  - Para comunicação curta (sem conexão)
  - Não confiável
  - Pouco prático para comunicações longas (confiabilidade precisa ser programada)
  - Usos típicos:
    - RPC
    - Broadcast

## Implementação

### Associação de sockets

- Definida por:
  - um protocolo: TCP ou UDP
  - endereço IP local
  - porta local
  - endereço IP distante
  - porta distante

## Comunicação via TCP

### Servidor

socket ( )

Cria um socket com

- Família (ou domínio): UNIX, Internet, XNS
- Tipo: stream, datagrama, puro
- Protocolo (por conseq.): TCP, UDP

```
sockfd = (int) socket (int family, int type, int protocol)
```

## Comunicação via TCP

### Servidor

socket ( )

↓

bind ( )

Atribui ao socket

- Endereço Internet (pode ser "any")
- Porta de comunicação

```
ret = (int) bind (int sockfd, struct sockaddr *myaddr, int addrlen)
```

## Comunicação via TCP

### Servidor

socket ( )

↓

bind ( )

↓

listen ( )

Declara

- Que está pronto para receber conexões
- Até quantas devem ser enfileiradas

```
ret = (int) listen (int sockfd, int backlog)
```

## Comunicação via TCP

### Servidor

socket ( )

↓

bind ( )

↓

listen ( )

↓

accept ( )

- Bloqueia até que haja pedido de conexão
- Quando houver algum, aceita

```
newsock = (int) accept (int sockfd, struct sockaddr *peer, int *addrlen)
```

## Comunicação via TCP

### Servidor

socket ( )

↓

bind ( )

↓

listen ( )

↓

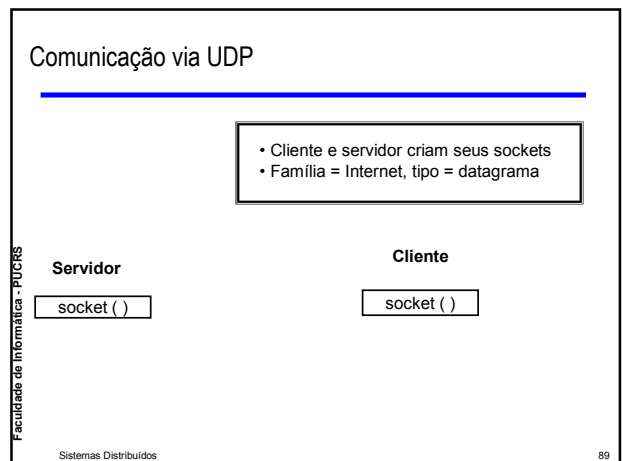
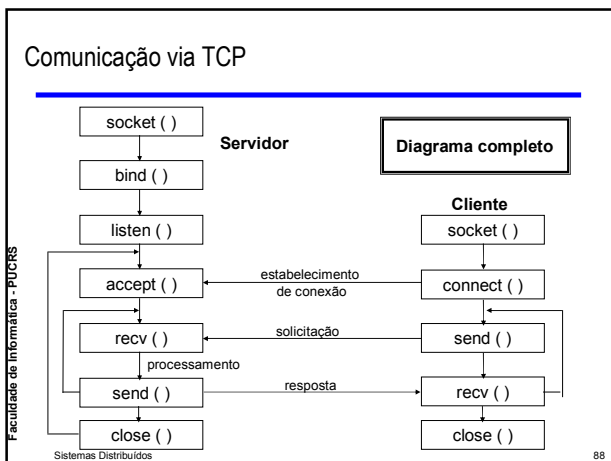
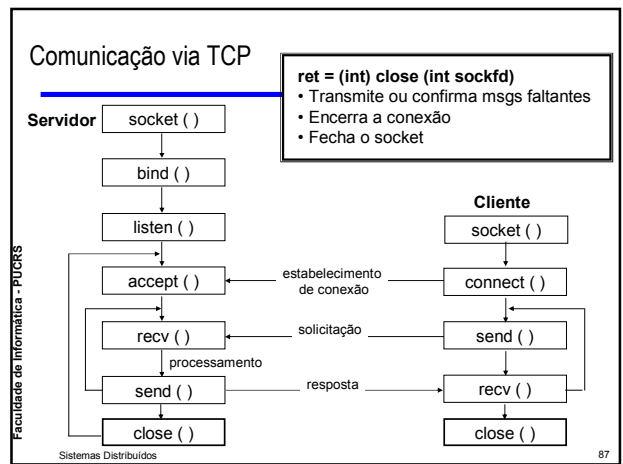
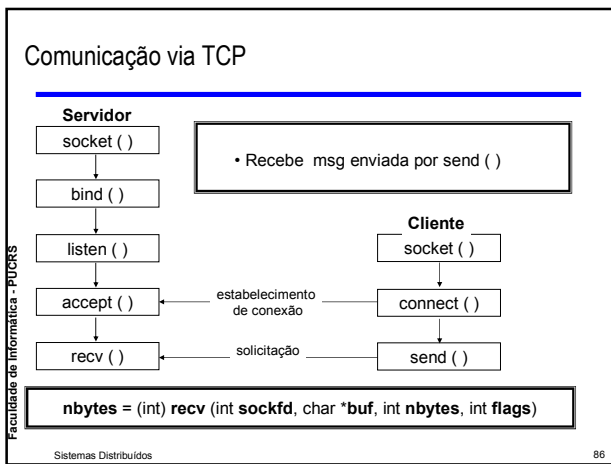
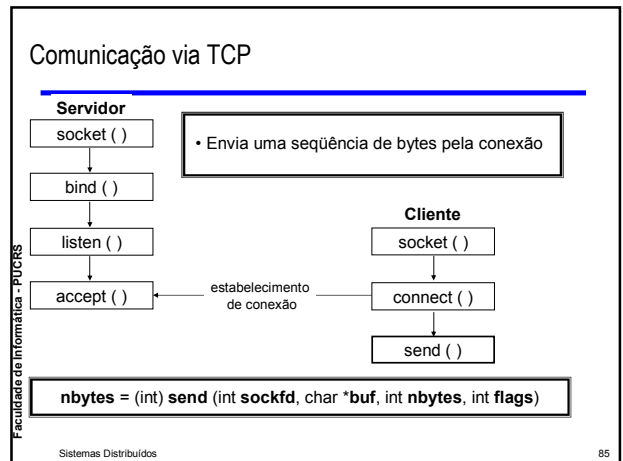
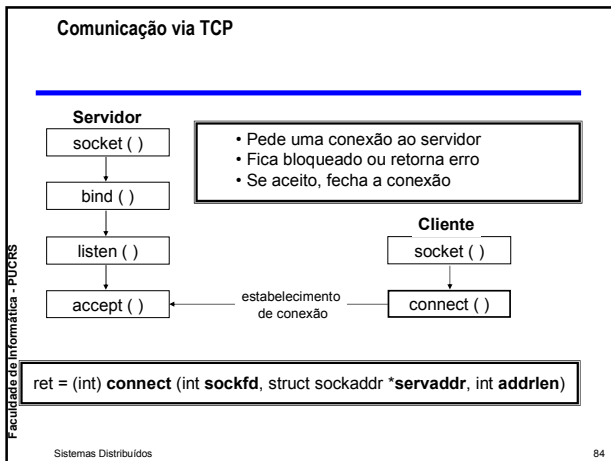
accept ( )

- Cria um socket idêntico ao do servidor (mesma família e tipo)

### Cliente

socket ( )

```
sockfd = (int) socket (int family, int type, int protocol)
```



## Comunicação via UDP

- Cliente e servidor definem endereços



## Comunicação via UDP

- Recebe pacote enviado do endereço informado
- Se não houver nada, bloqueia



```
nbytes = (int) recvfrom (int sockfd, char *buf, int nbytes, int flags, struct sockaddr *from, int *addrlen)
```

## Comunicação via UDP

- Envia pacote para o endereço informado



```
nbytes = (int) sendto (int sockfd, char *buf, int nbytes, int flags, struct sockaddr *to, int *addrlen)
```

## Comunicação via UDP

- Diagrama completo

