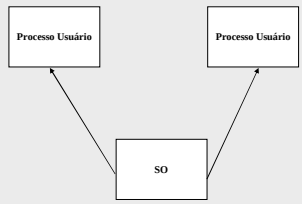
COMUNICAÇÃO ENTRE PROCESSOS
Sockets

Sockets - Conceitos Básicos

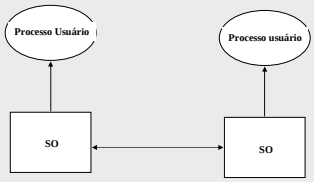
- Sockets são uma forma de IPC (InterProcess Communication) fornecida pela 4.3 BSD que fornecem comunicação entre processos residentes em sistema único ou processos residentes em sistemas remotos.

Conceitos Básicos

- Sockets criados por diferentes programas usam nomes para se referenciar
- Esses nomes geralmente devem ser traduzidos em endereços para uso
- Um endereço é especificado por um domínio



IPC em um mesmo sistema



Comunicação de Processos Remotos

Tipos de Sockets

- STREAM SOCKET - Provê sequenciamento e fluxo bidirecional. Este tipo de socket transmite dados sobre um base confiável e com capacidade de transmissão de dados expressos.
- No domínio UNIX, o SOCKET_STREAM trabalha igual a um pipe. No domínio INTERNET este tipo de socket é implementado sobre TCP/IP.

Tipos de Sockets

- SOCK_DGRAM - Suporta fluxo de dados bidirecional mas não oferece um serviço confiável como STREAM_SOCKET.
- Mensagens duplicadas, perdidas, e em ordem diferente (não sequenciadas) são problemas que podem aparecer neste tipo de socket.

Tipos de Sockets

- RAW_SOCKET - permite o acesso a interface de protocolos de rede. Disponível para usuários avançados e que possuam autoridade de usuário root
- permite que uma aplicação acesse diretamente protocolos de comunicação de baixo nível.
- permite a construção de novos protocolos sobre os protocolos de baixo nível já existentes
- normalmente orientados a datagrama

Domínios e Protocolos

- 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

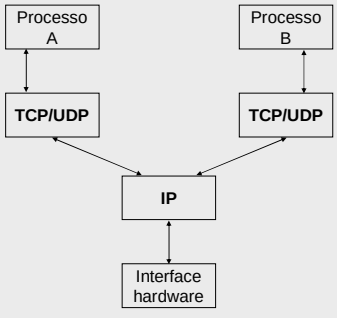
Domínios e Protocolos

- Domínio Internet
  - Implementação Unix do protocoloTCP/UDP/IP
  - 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

Tipos de Sockets

- Socket Stream
  - conexões confiáveis
- Datagramas
  - não fornecem segurança

Protocolos TCP/IP



IP

- ***Internetwork Protocol***
- **Roteamento de mensagens na rede**
- **Unidade: datagramas**
- **Fragmentação de pacotes (se > MTU)**
- **Entrega não confiável de pacotes**

TCP

- ***Transmission Control Protocol***
- **Para comunicação longa (conexão)**
- **Confiável**
- **Transmissão de fluxo de dados: Não respeita limites de mensagens**
- **Baixo desempenho em comunicações curtas (?)**
- **Usos típicos:**
  - Login remoto
  - Transferência de arquivo

UDP

- ***User Datagram Protocol***
- **Para comunicação curta (sem conexão)**
- **Não confiável**
- **Transmissão de pacotes: respeita limites de mensagens**
- **Pouco prático para comunicações longas (confiabilidade precisa ser programada)**
- **Usos típicos:**
  - RPC
  - Broadcast

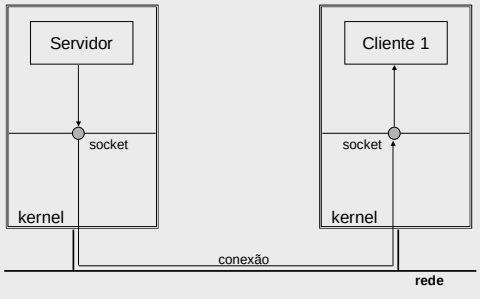
Porta

- **"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**

Associação

- **Definida por**
  - Um protocolo: TCP ou UDP
  - Endereço IP local
  - Porta local
  - Endereço IP distante
  - Porta distante

Comunicação por *sockets*



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**)

Faculdade de Informática - PUCRS

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**)

Faculdade de Informática - PUCRS

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**)

Faculdade de Informática - PUCRS

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**)

Faculdade de Informática - PUCRS

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**)

Faculdade de Informática - PUCRS

Comunicação via TCP

Servidor

socket ( )

bind ( )

listen ( )

accept ( )

- Pede uma conexão ao servidor
- Fica bloqueado ou retorna erro
- Se aceito, fecha a conexão

Cliente

socket ( )

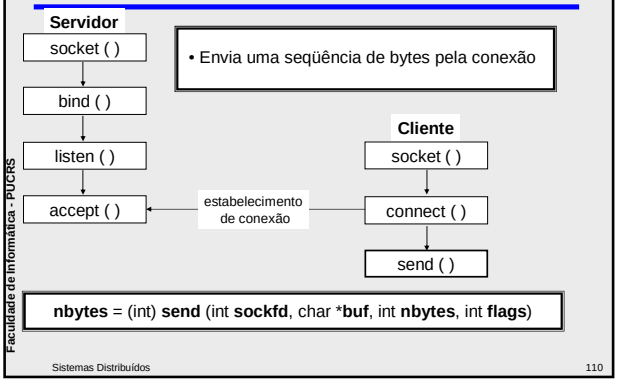
connect ( )

estabelecimento de conexão

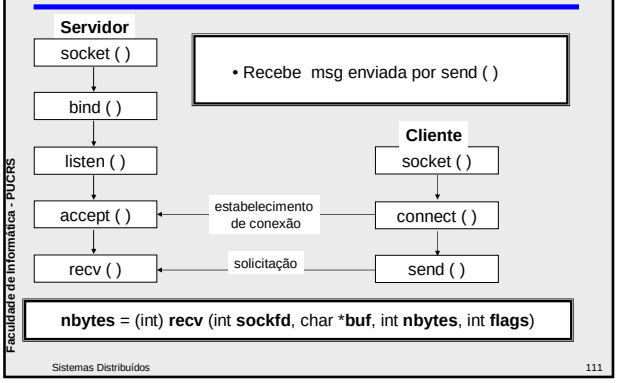
ret = (int) **connect** (int **sockfd**, struct sockaddr ***servaddr**, int **addrlen**)

Faculdade de Informática - PUCRS

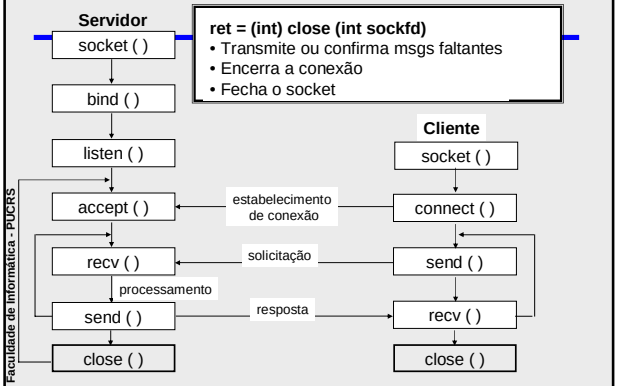
Comunicação via TCP



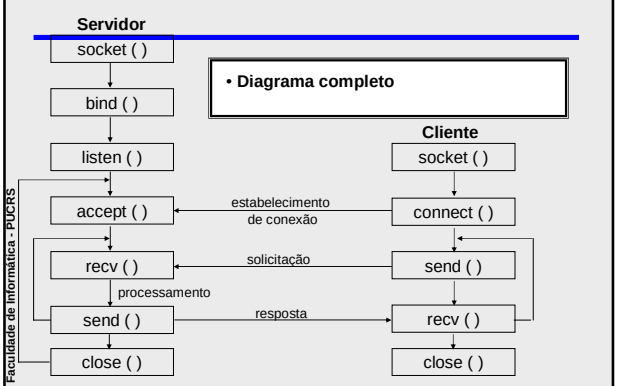
Comunicação via TCP



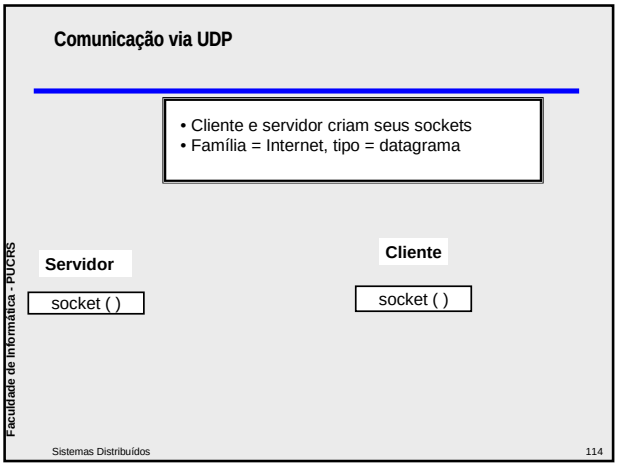
Comunicação via TCP



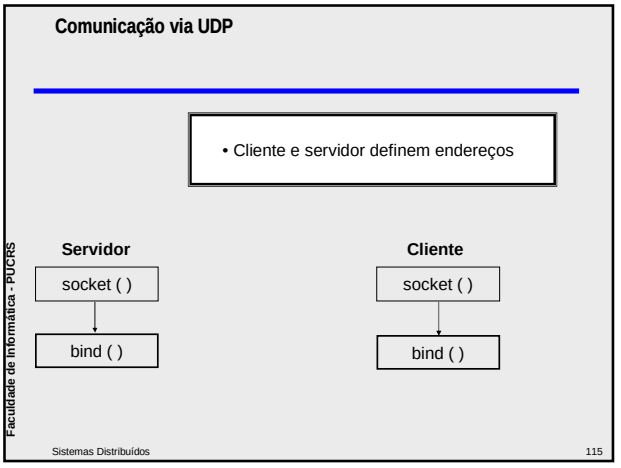
Comunicação via TCP



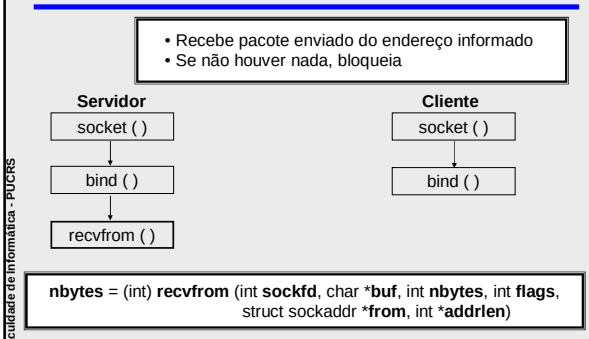
Comunicação via UDP



Comunicação via UDP

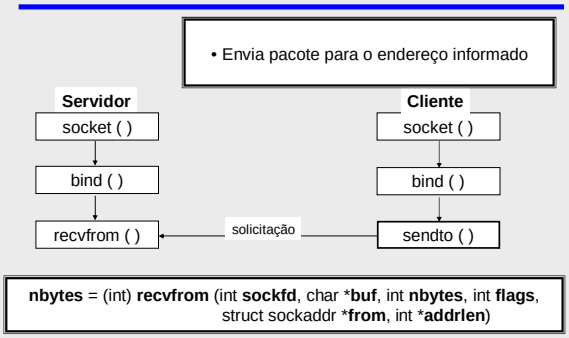


Comunicação via UDP



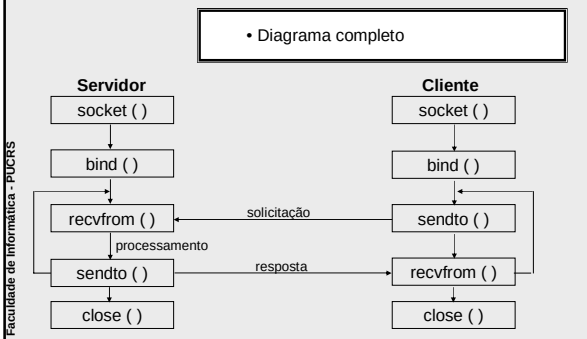
nbytes = (int) **recvfrom** (int **sockfd**, char ***buf**, int **nbytes**, int **flags**, struct sockaddr ***from**, int ***addrlen**)

Comunicação via UDP



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

Comunicação via UDP



Comunicação entre processos

- Exercícios:
  - classifique cada uma das primitivas de pipes e sockets com relação
    - a modelo implícito e explicito de endereçamento
    - a comunicação bloqueante e não bloqueante
    - e a bufferização de mensagens
    - Obs.: analise o maior número possível de opções oferecidas por estas interfaces
  - usando PIPES, teste, com programas:
    - uso bidirecional de um pipe (existe?);
    - uso concorrente de um pipe;

Comunicação entre processos

- Exercícios:
  - usando sockets em plataforma linux ou solaris, e comunicação no modo datagrama,
    - implemente em linguagem C um mecanismo de comunicação com suporte a falhas, do tipo "two-message"
      - analise bem a interface de sockets para utilizar o máximo de sua funcionalidade
  - implemente um servidor concorrente (trata vários clientes simultaneamente) de operações matemáticas utilizando pipes e depois sockets (tipo stream)
    - os clientes mandam os operandos e a operação, ficam a espera do resultado, e voltam a sortear operandos para o próximo pedido
    - o servidor calcula e manda o resultado devolta para os vários clientes
    - modifique a implementação para trabalhar com sockets no modo datagrama, adicionando o mecanismo de confirmação da questão anterior