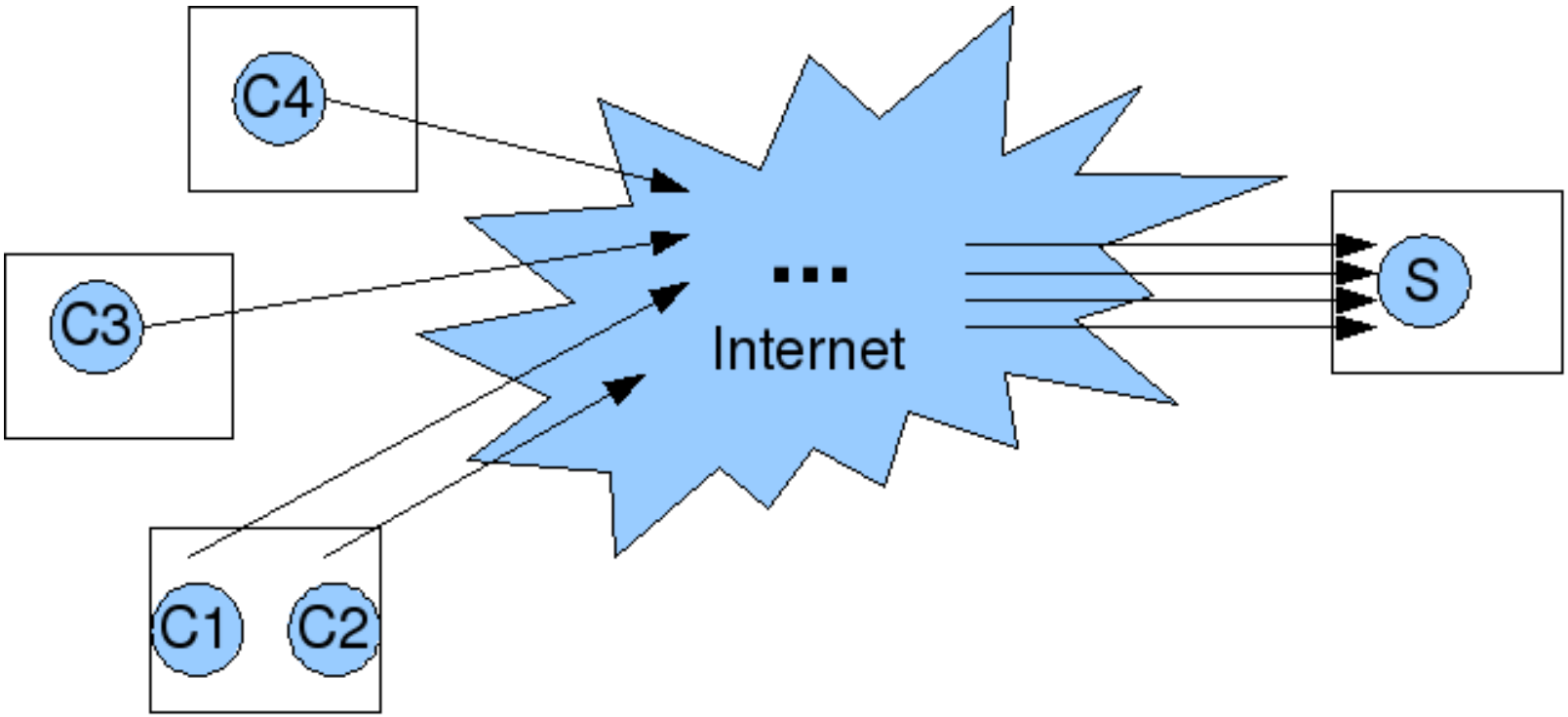
Sistemas Distribuídos (DCC/UFRJ)

Aula 2: Comunicação entre processos usando sockets

Prof. Silvana Rossetto

8 de abril de 2016

Processamento distribuído no paradigma cliente/servidor



Processamento distribuído no paradigma cliente/servidor

Cliente

- o SO permite que vários usuários executem o programa cliente concorrentemente na mesma máquina; ou
- usuários em diferentes máquinas podem executar o programa cliente simultaneamente

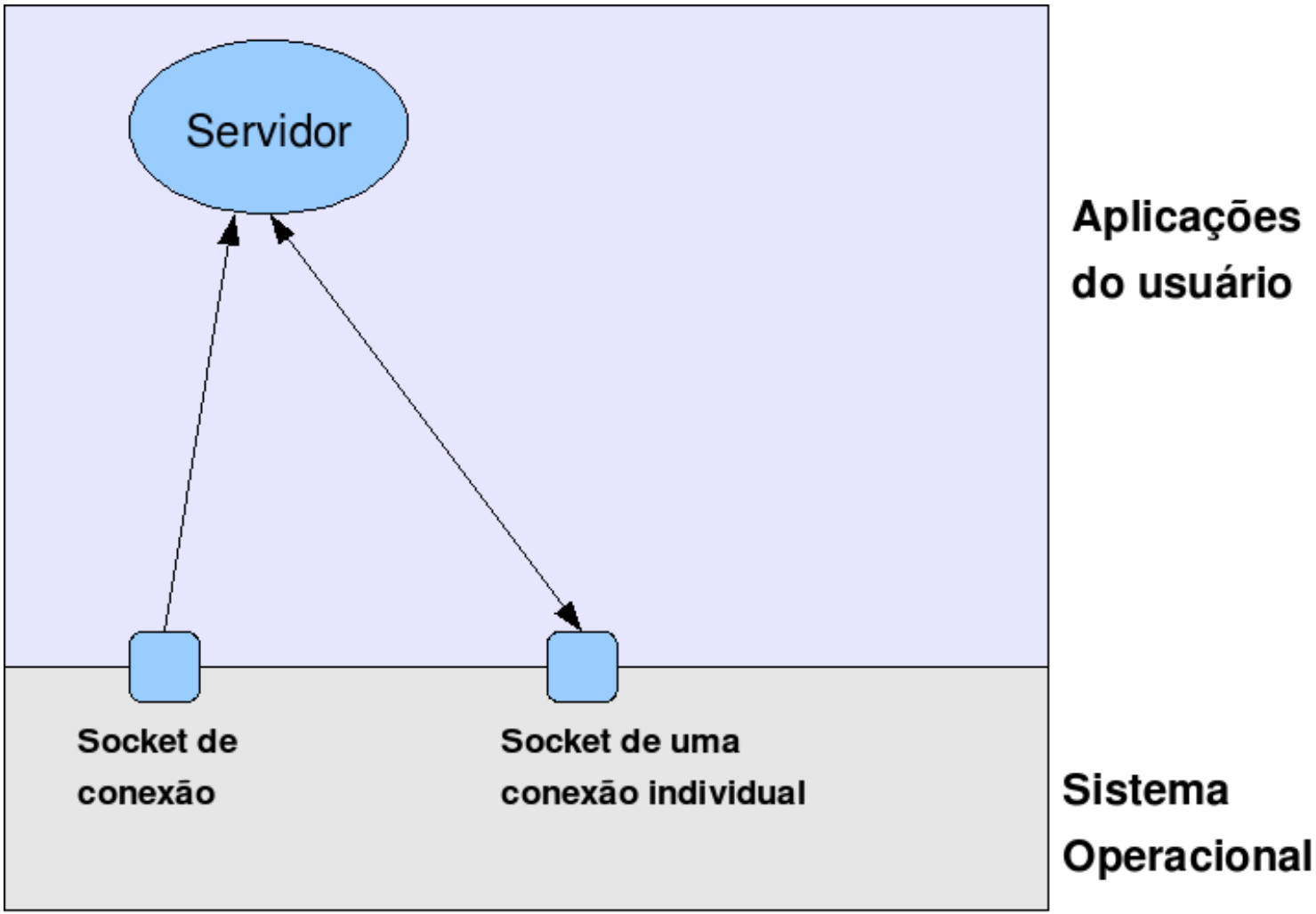
...um **programa cliente** individual opera como um programa convencional, ele não precisa gerenciar concorrência explicitamente

Processamento distribuído no paradigma cliente/servidor

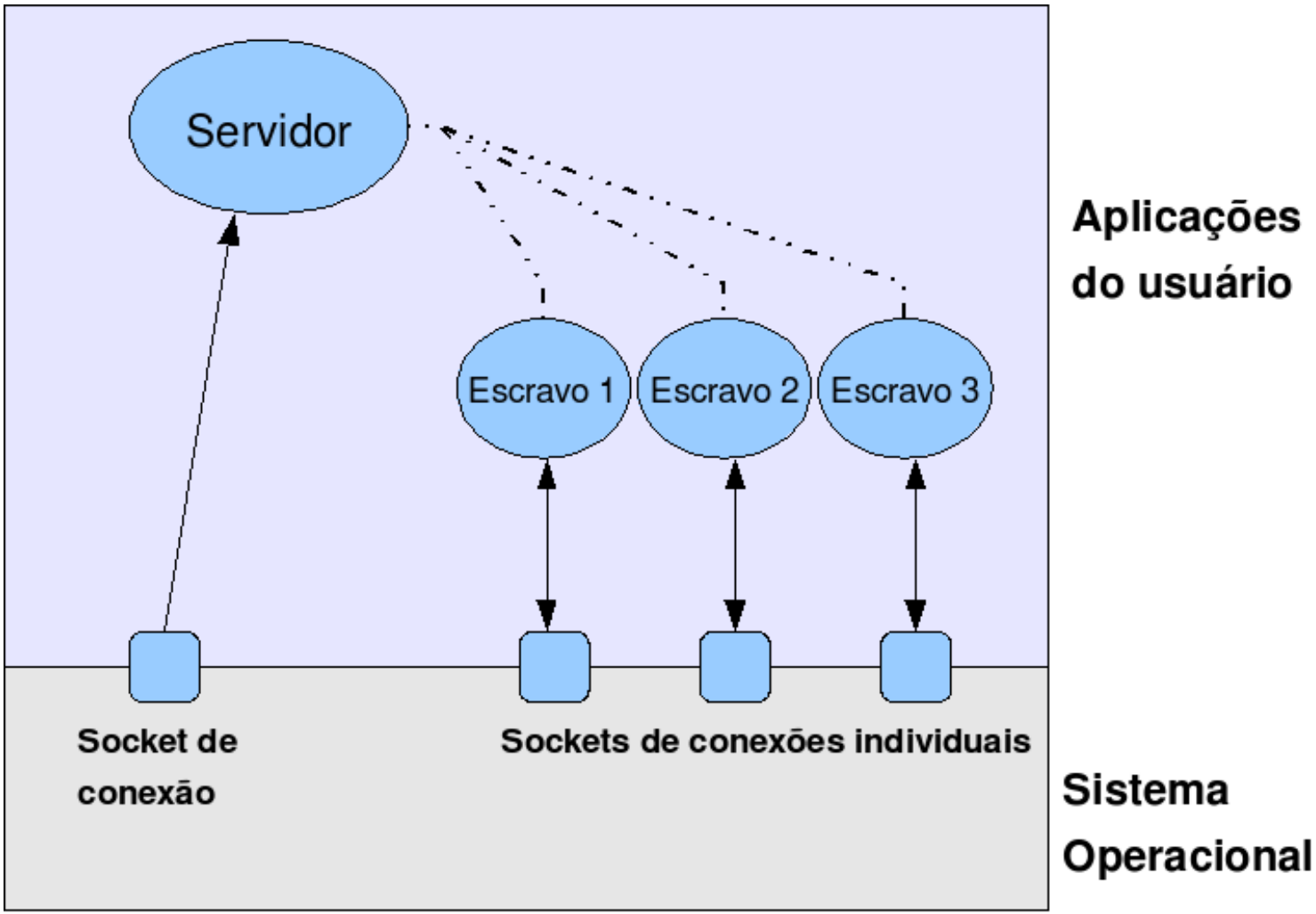
Servidor

- Do lado do **servidor** é preciso tratar as requisições recebidas concorrentemente
- O software do processo servidor deve ser explicitamente programado para tratar requisições concorrentes, uma vez que vários clientes podem contactar o servidor

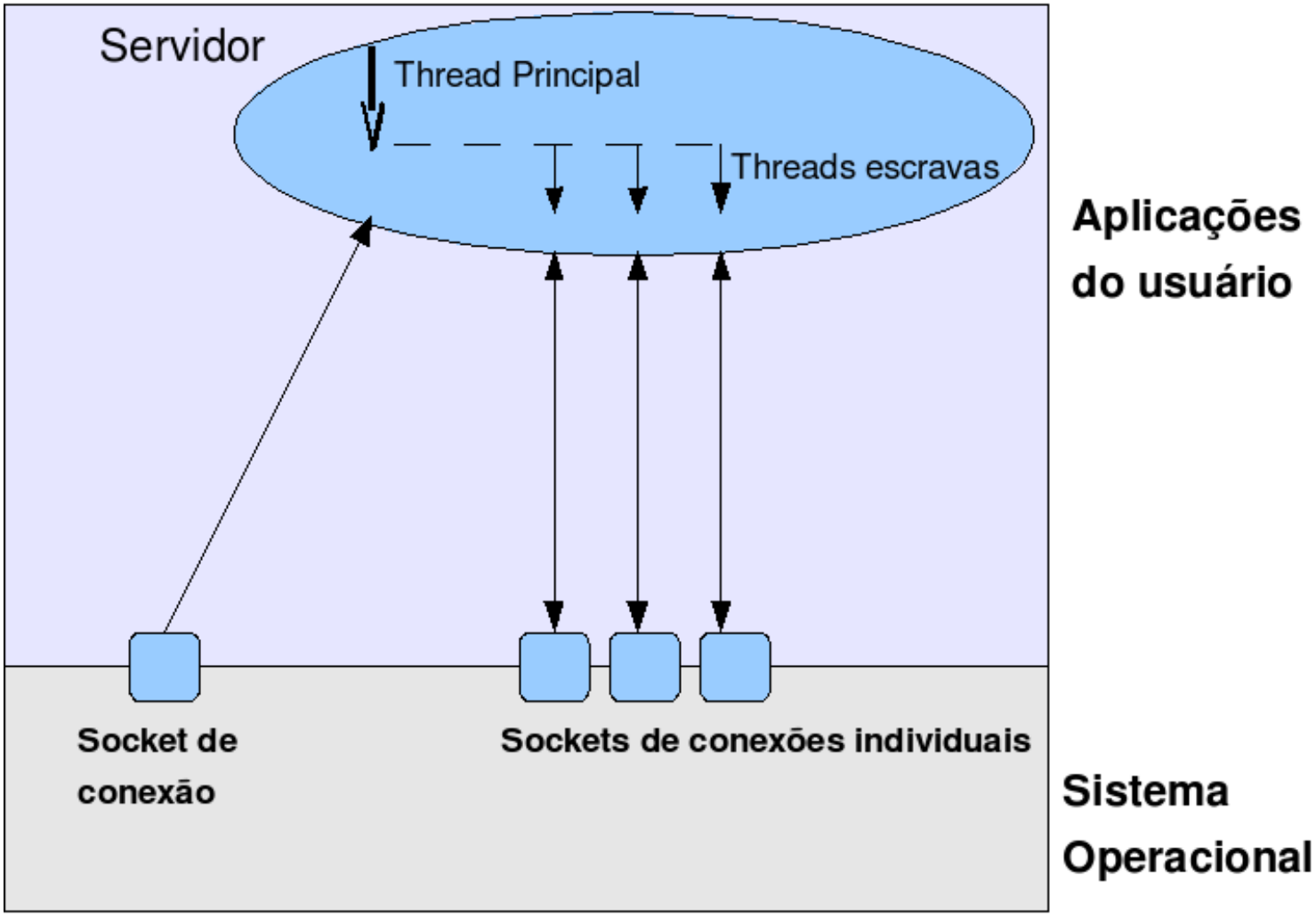
Servidor TCP iterativo



Servidor TCP concorrente multiprocesso



Servidor TCP concorrente multithreading



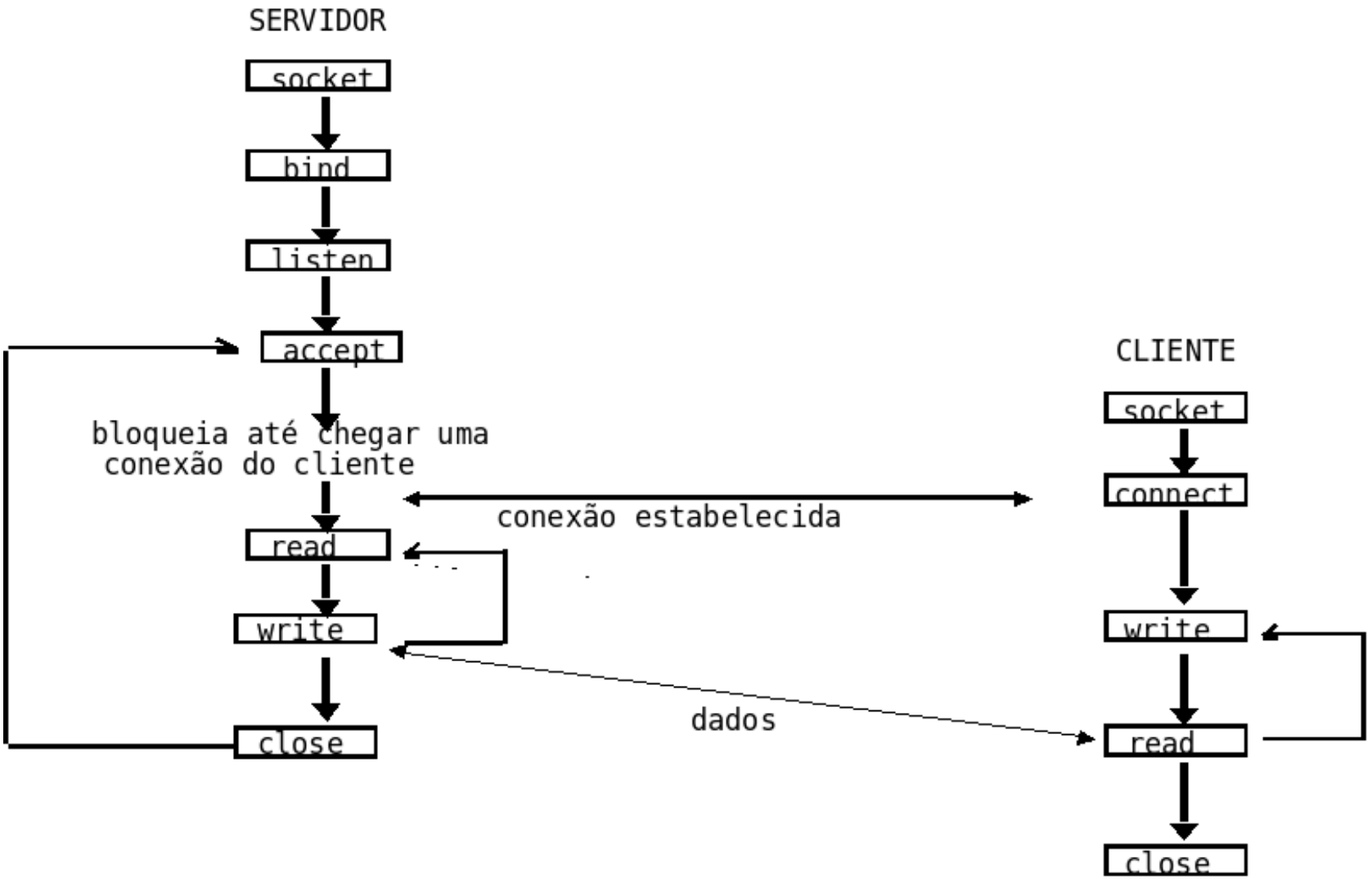
API de sockets para conexões TCP

- **socket()**
  - cria um socket usado para comunicação e retorna um descritor
- **connect()**
  - depois de criar um socket, um cliente chama `connect` para estabelecer uma conexão com um servidor, usando o descritor do socket
- **write()**
  - para enviar dados através de uma conexão TCP
- **read()**
  - para receber dados através de uma conexão TCP
- **close()**
  - para desalocar o socket

API de sockets para conexões TCP

- **bind()**
  - usado por servidores para especificar uma porta na qual ele irá esperar conexões
- **listen()**
  - servidores chamam o `listen` para colocar o socket no modo passivo e torná-lo disponível para aceitar conexões
- **accept()**
  - depois que um servidor chama `socket` para criar um socket, `bind` para especificar seu endereço e `listen` para colocá-lo no modo passivo, ele deve chamar o `accept` para pegar a primeira solicitação de conexão na fila

Interação cliente/servidor usando socket e TCP



1

[1] *Fonte: Notas de aula, profa. Noemi Rodriguez*

Nomes de domínio e funções de conversão

- O cliente TCP/IP deve especificar o endereço de um servidor usando a estrutura de dados **sockaddr_in**
- Para isso é preciso converter um **número decimal separado por pontos** ou um **nome de domínio no formato texto** em um **endereço IP de 32 bits binário**
- A interface de socket inclui funções especiais para essas conversões:
  - **inet_addr()**: recebe uma string ASCII que contém um endereço decimal separado por pontos e retorna o endereço IP em binário

Ordem dos bytes na rede

- O TCP/IP especifica uma representação padrão para inteiros binários com o **byte mais significativo primeiro**
- A interface de socket oferece funções para converter o formato de byte usado na rede para o formato usado na máquina local:
  - **htons**: *host to network short* (16 bits)
  - **ntohs**: *network to host short* (16 bits)
  - **htonl**: *host to network long* (32 bits)
  - **ntohl**: *network to host long* (32 bits)

Algoritmo de clientes TCP

- A escolha do endereço IP local e da porta local para a conexão com o servidor ocorre como "efeito colateral" da chamada `connect()`
- O IP escolhido (interface de rede) deve ser reconhecido pelo software de roteamento
- A chamada **connect()** não retorna até que a conexão TCP tenha sido estabelecida ou o timeout do TCP seja alcançado

Lendo respostas em uma conexão TCP

- O TCP é um protocolo *orientado-stream*:
  - ele garante a entrega de uma sequência de bytes que o emissor escreveu, mas não garante entregá-los no mesmo bloco que eles foram escritos
- O TCP pode decidir quebrar o bloco de dados em pedaços e transmitir cada pedaço em um segmento separado (ex., quando o buffer do receptor não tem espaço suficiente para todo o bloco)
  - **o receptor pode receber dados em pequenos rajadas, mesmo sendo enviados em uma única chamada** `write()`
- O TCP também pode escolher acumular uma certa quantidade de bytes em seu buffer de entrada antes de enviar um segmento
  - **o receptor pode receber dados em grandes rajadas, mesmo sendo enviados em várias chamadas** `write()`

Exemplo de aplicação C/S usando sockets em C

Acessar site da disciplina:

http://www.dcc.ufrj.br/ silvana/sd/cods-aula2.zip

Atividade prática

Implementar uma aplicação cliente/servidor

1. Implementar um **servidor multithreading** que ofereça as seguintes operações:
   1. Somar dois números inteiros (positivos ou negativos)
   2. Subtrair dois números inteiros (positivos ou negativos)
2. Implementar uma **aplicação cliente** que permita ao usuário interagir com o servidor fazendo chamadas às operações oferecidas
3. Experimentar os programas cliente e servidor em máquinas distintas

Referências bibliográficas

1. I. Kuz, F. Rauch, M. T. Chakravarty and G. Heiser, `www.cse.unsw.edu.au/ cs9243/lectures/comm-notes.pdf`
2. Notas de aula da profa. Noemi Rodriguez (DI/PUC-Rio)
3. J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach*, 2009
4. Comer, *Interligação de Redes com TCP/IP*. Vol. I, Campus, 2006