Programação de Sockets

Mário Meireles Teixeira. UFMA-DEINF

Programação de Sockets

Objetivo: aprender a construir aplicações cliente/servidor que se comunicam usando sockets

API de Sockets:

- introduzida no UNIX BSD 4.1, 1981
- explicitamente criados, usados e liberados pelas aplicações
- paradigma cliente/servidor
- dois tipos de serviço de transporte via sockets:
  - datagrama não confiável
  - confiável, orientado a cadeias de bytes (*byte stream*)

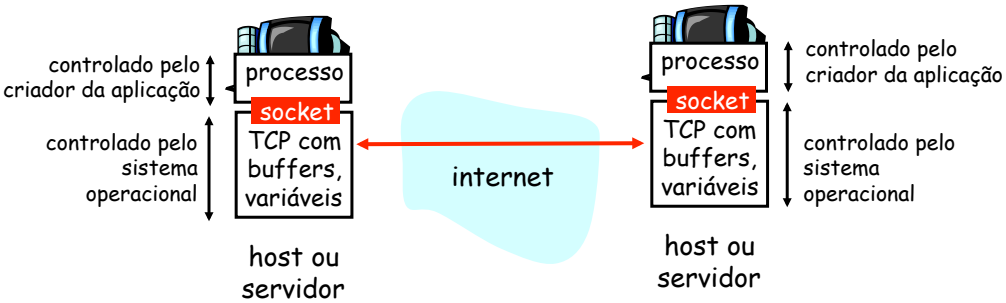
socket

Uma interface *local*, criada e mantida pelas aplicações, *controlada pelo SO,* através da qual os processos de aplicação podem tanto enviar quanto receber mensagens de outro processo de aplicação, local ou remoto.

Programação de Sockets com TCP

Socket: uma porta entre o processo de aplicação e o protocolo de transporte fim-a-fim (UDP or TCP)

serviço TCP: transferência confiável de bytes de um processo para outro



Programação de Sockets com TCP

Cliente deve contactar o servidor

- processo servidor já deve estar executando antes de ser contactado
- servidor deve ter criado socket (porta) que aceita o contato do cliente

Cliente contacta o servidor:

- criando um socket TCP local
- especificando endereço IP e número da porta do processo servidor

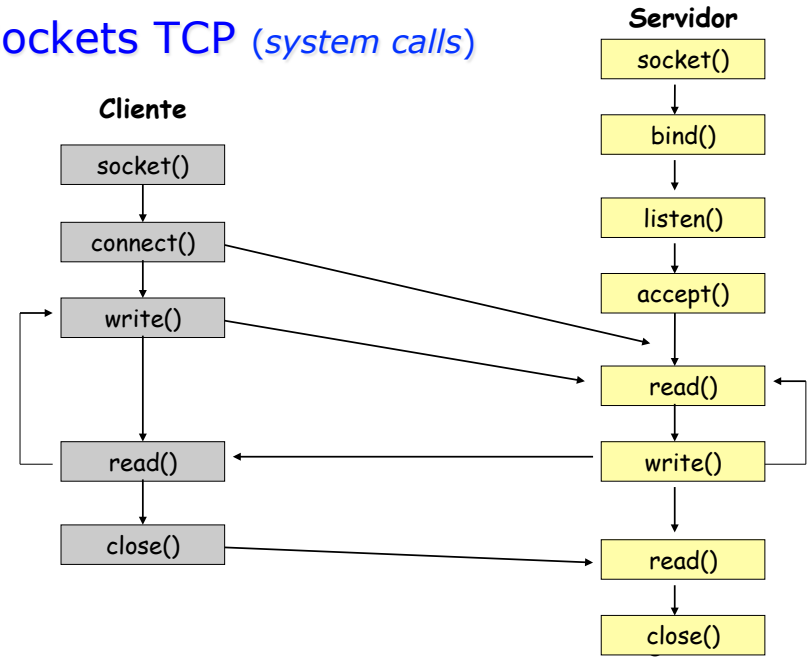
- Quando o cliente cria o socket: cliente TCP estabelece conexão com o servidor

- Quando contactado pelo cliente, o servidor cria um novo socket para comunicar-se com o cliente
  - permite que o servidor con-verse com múltiplos clientes

ponto de vista da aplicação

TCP fornece uma transferência confiável, em ordem de bytes ("pipe"), entre o cliente e o servidor

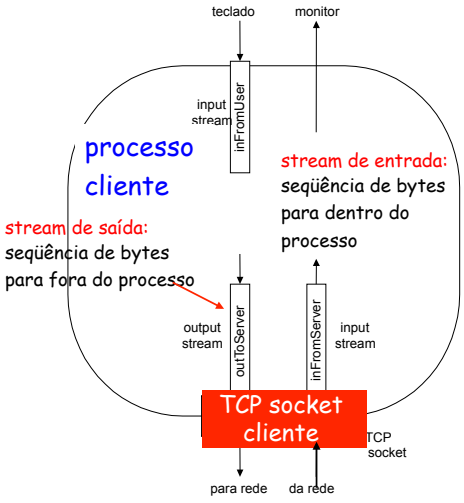
Sockets TCP (*system calls*)



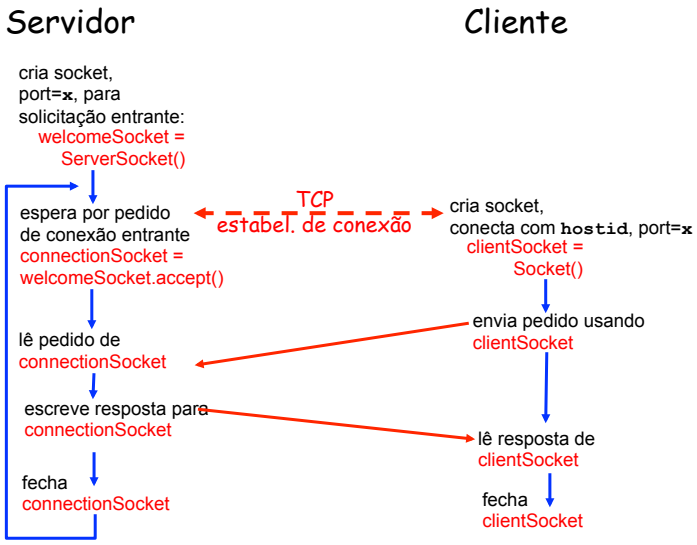
Programação de Sockets com TCP

Exemplo de aplicação cliente-servidor:

- cliente lê linha da entrada padrão do sistema (stream **`inFromUser`**) e envia para o servidor via socket (stream **`outToServer`**);
- servidor lê linha do socket;
- servidor converte linha para letras maiúsculas e envia de volta ao cliente;
- cliente lê a linha modificada a partir do stream **`inFromServer`**;



Interação Cliente/Servidor: TCP



Exemplo: cliente Java (TCP)

```
import java.io.*;
import java.net.*;
class TCPClient {

   public static void main(String argv[]) throws Exception
   {
      String sentence;
      String modifiedSentence;

      BufferedReader inFromUser =
        new BufferedReader(new InputStreamReader(System.in));

      Socket clientSocket = new Socket("hostname", 6789);

      DataOutputStream outToServer =
        new DataOutputStream(clientSocket.getOutputStream());
```

Cria stream de entrada

Cria socket cliente, conecta ao servidor

Cria stream de saída ligado ao socket

Exemplo: cliente Java (TCP)

Cria stream de entrada ligado ao socket

```
BufferedReader inFromServer =
    new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

sentence = inFromUser.readLine();
```

Envia linha para o servidor

```
outToServer.writeBytes(sentence + '\n');
```

Lê linha do servidor

```
modifiedSentence = inFromServer.readLine();
```

.....cliente aguarda resposta do servidor.....

```
System.out.println("FROM SERVER: " + modifiedSentence);

clientSocket.close();

} }
```

Exemplo: servidor Java (TCP)

```
import java.io.*;
import java.net.*;

class TCPServer {

  public static void main(String argv[]) throws Exception
   {
     String clientSentence;
     String capitalizedSentence;
```

Cria socket de aceitação na porta 6789

```
     ServerSocket welcomeSocket = new ServerSocket(6789);

     while(true) {
```

Espera, no socket de aceitação por contato do cliente

```
       Socket connectionSocket = welcomeSocket.accept();
```

Cria stream de entrada, ligado ao socket

```
       BufferedReader inFromClient =
         new BufferedReader(new
         InputStreamReader(connectionSocket.getInputStream()));
```

Exemplo: servidor Java (TCP)

Cria stream de saída, ligado ao socket

Lê linha do socket

Escreve linha para o socket

```
DataOutputStream  outToClient =
  new DataOutputStream(connectionSocket.getOutputStream());
clientSentence = inFromClient.readLine();

capitalizedSentence = clientSentence.toUpperCase() + '\n';

outToClient.writeBytes(capitalizedSentence);
            }
        }
    }
```

Fim do while loop, retorne e espere por outra conexão do cliente

Programação de Sockets com UDP

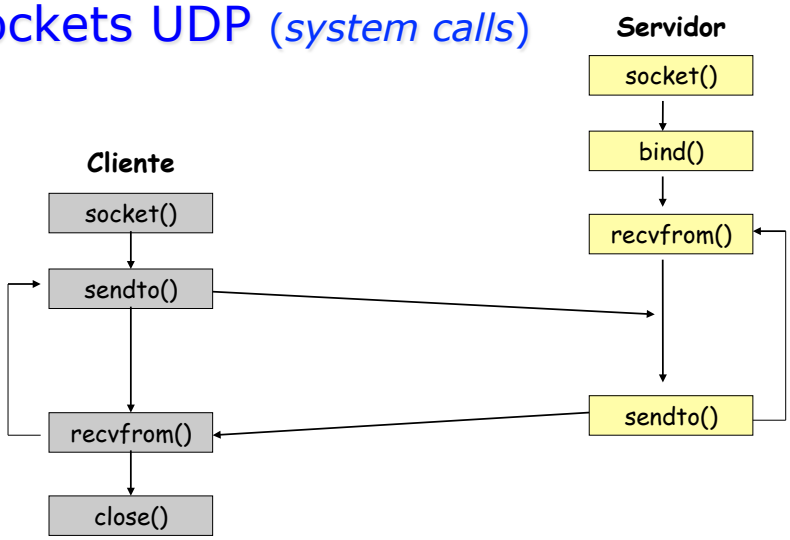
UDP: não há conexão entre o cliente e o servidor

- não existe apresentação
- transmissor envia explicita-mente endereço IP e porta de destino em cada mensagem
- servidor deve extrair o endereço IP e porta do transmissor de cada datagrama recebido
- UDP: dados transmitidos podem ser recebidos fora de ordem ou perdidos

ponto de vista da aplicação

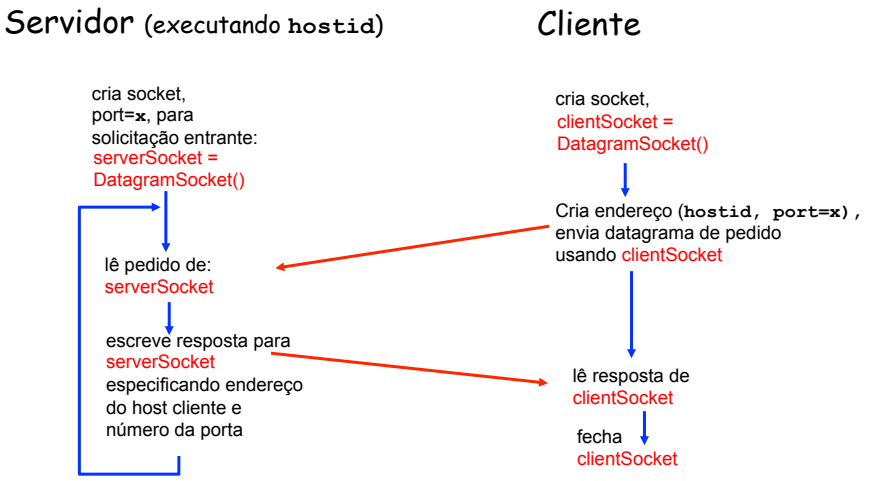
UDP fornece uma transferência não confiável de grupos de bytes ("datagramas") entre o cliente e o servidor

Sockets UDP (*system calls*)

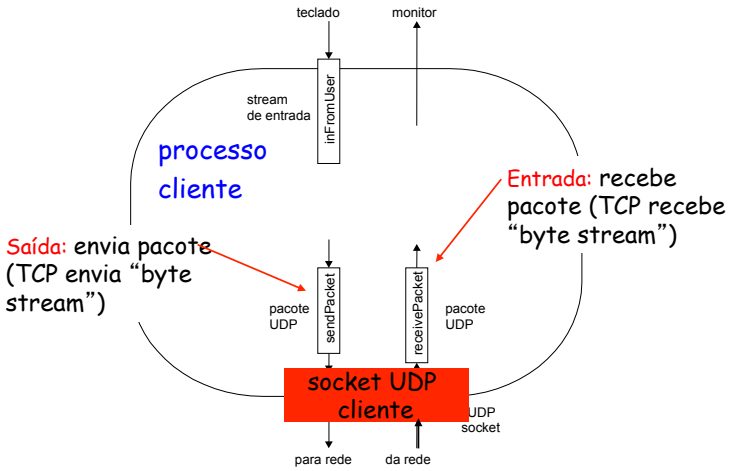


13

Interação Cliente/Servidor: UDP



Exemplo: cliente Java (UDP)



Exemplo: cliente Java (UDP)

```
import java.io.*;
import java.net.*;

class UDPClient {
   public static void main(String args[]) throws Exception
   {
```

Cria stream de entrada →
```
      BufferedReader inFromUser =
        new BufferedReader(new InputStreamReader(System.in));
```

Cria socket cliente →
```
      DatagramSocket clientSocket = new DatagramSocket();
```

Traduz hostname para endereço IP usando DNS →
```
      InetAddress IPAddress = InetAddress.getByName("hostname");

      byte[] sendData = new byte[1024];
      byte[] receiveData = new byte[1024];

      String sentence = inFromUser.readLine();
      sendData = sentence.getBytes();
```

Exemplo: cliente Java (UDP)

Cria datagrama com dados a enviar, tamanho, endereço IP e porta

Envia datagrama para servidor

Lê datagrama do servidor

```
DatagramPacket sendPacket =
   new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

clientSocket.send(sendPacket);

DatagramPacket receivePacket =
   new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(receivePacket);

… cliente aguarda resposta do servidor …

String modifiedSentence =
   new String(receivePacket.getData());

System.out.println("FROM SERVER:" + modifiedSentence);
clientSocket.close();
}

}
```

Exemplo: servidor Java (UDP)

Cria datagram socket na porta 9876

Cria espaço para datagramas recebidos

Recebe datagrama

```
import java.io.*;
import java.net.*;

class UDPServer {
  public static void main(String args[]) throws Exception
   {

DatagramSocket serverSocket = new DatagramSocket(9876);

   byte[] receiveData = new byte[1024];
   byte[] sendData  = new byte[1024];

   while(true)
    {

DatagramPacket receivePacket =
     new DatagramPacket(receiveData, receiveData.length);

serverSocket.receive(receivePacket);
```

Exemplo: servidor Java (UDP)