Redes de Computadores

Capítulo 2.7 e 2.8 - Camada de Aplicação
Programação de sockets

Prof. Jó Ueyama

Março/2014

Sockets TCP - Cliente

- Processo servidor já deve estar em execução.
- Servidor deve ter criado socket (porta) que aceita o contato do cliente.
- Cliente contata 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 TCP do servidor.

Sockets TCP - Servidor

- Quando contatado pelo cliente, o servidor cria um novo socket para o processo servidor comunicar-se com o cliente.
- Permite ao servidor conversar com múltiplos clientes
- Números da porta de origem são usados para distinguir o cliente (mais no Capítulo 3).

Terminologia: stream

- Um stream é uma seqüência de caracteres que fluem para dentro ou para fora de um processo.
- Um stream de entrada é agregado a alguma fonte de entrada para o processo, ex.: teclado ou socket.
- Um stream de saída é agregado a uma fonte de saída, ex.: monitor ou socket.

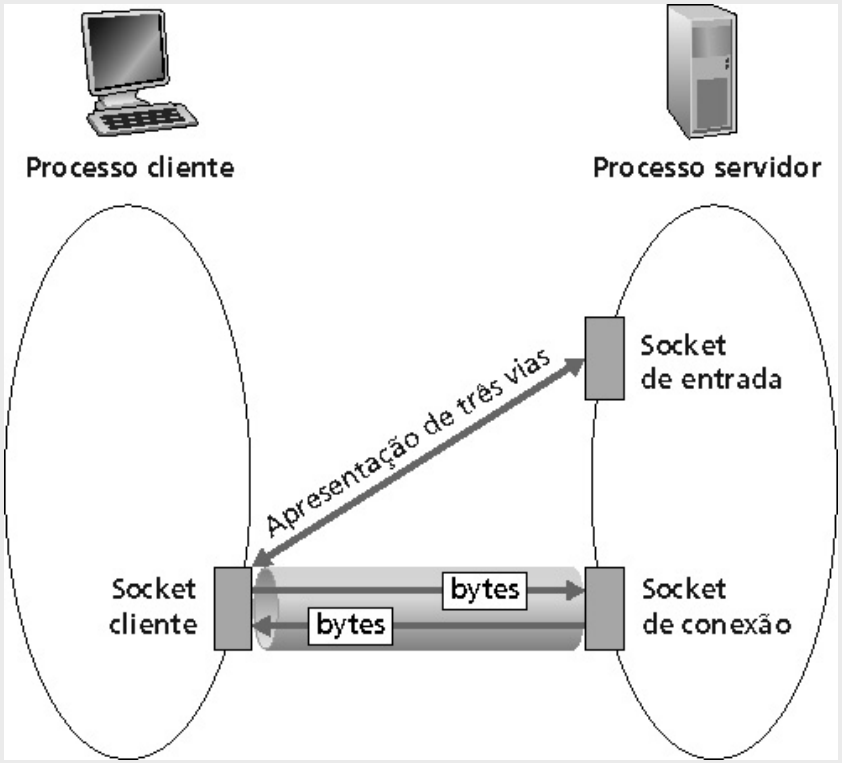
Exemplo de aplicação cliente-servidor TCP

1) Cliente lê linha da entrada-padrão do sistema (**inFromUser** stream), envia para o servidor via socket (**outToServer** stream).

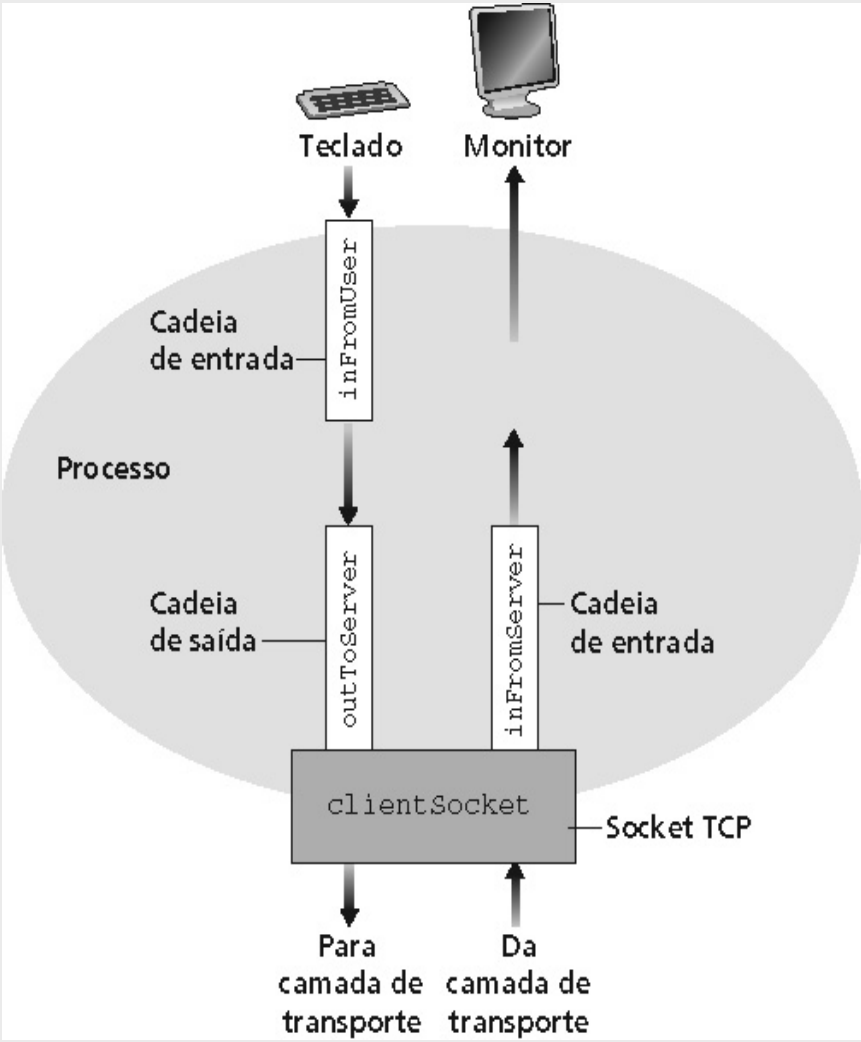
2) Servidor lê linha do socket.

3) Servidor converte linha para letras maiúsculas e envia de volta ao cliente.

4) Cliente lê a linha modificada através do (**inFromServer** stream).



Exemplo de aplicação cliente-servidor TCP



Exemplo: Cliente TCP Java

```
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());
        BufferedReader inFromServer = new BufferedReader(
                new InputStreamReader(clientSocket.getInputStream()));
        sentence = inFromUser.readLine();
        outToServer.writeBytes(sentence + '\n');
        modifiedSentence = inFromServer.readLine();
        System.out.println("FROM SERVER: " + modifiedSentence);
        clientSocket.close();
    }
}
```

Cria stream de entrada

cria stream de saída ligada ao socket

cria stream de entrada ligada ao socket

Envia linha p/ servidor

Lê linha do servidor

Exemplo: Servidor TCP Java

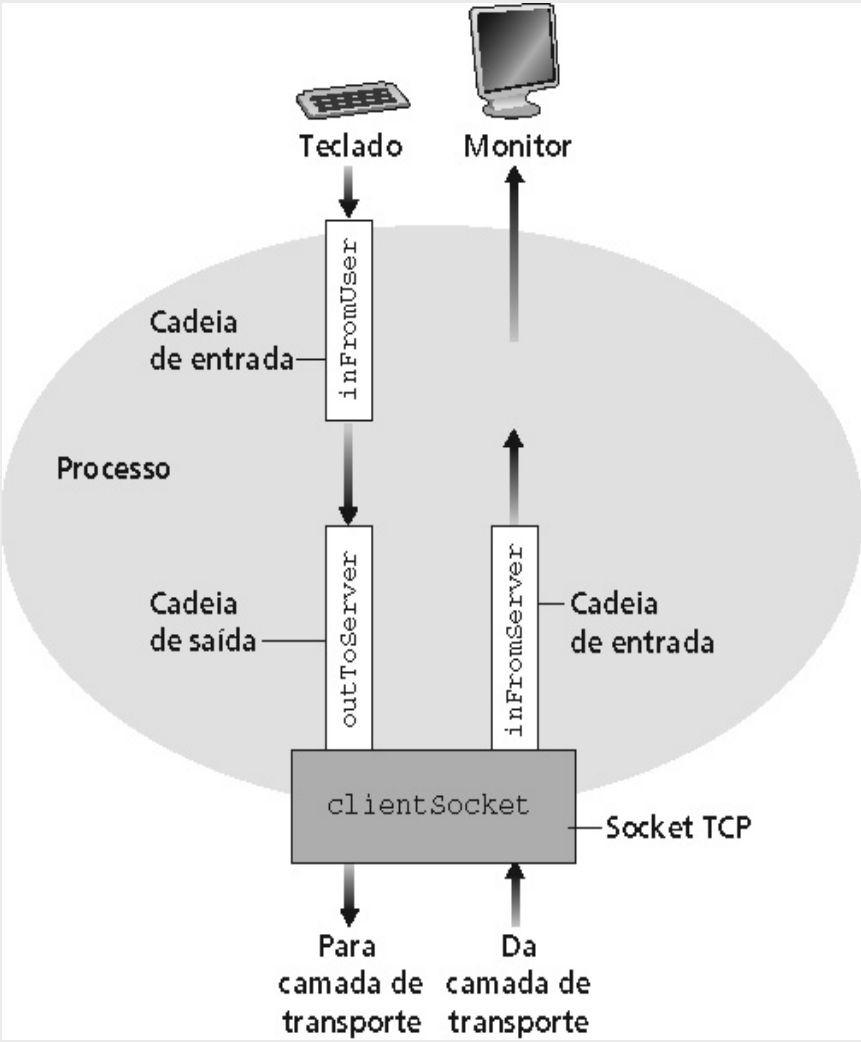
```
import java.io.*;
import java.net.*;

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

      ServerSocket welcomeSocket = new ServerSocket(6789);

      while(true) {

            Socket connectionSocket = welcomeSocket.accept();
            BufferedReader inFromClient =
                        new BufferedReader(new
                      InputStreamReader(connectionSocket.getInputStream()));

            DataOutputStream outToClient =
                    new DataOutputStream(connectionSocket.getOutputStream());
            clientSentence = inFromClient.readLine();
            capitalizedSentence = clientSentence.toUpperCase() + '\n';
           outToClient.writeBytes(capitalizedSentence);
        }
    }
}
```

Programação de sockets com UDP

- Não há conexão entre o cliente e o servidor.
- Transmissor envia explicitamente endereço IP e porta de destino em cada mensagem.
- Servidor deve extrair o endereço IP e porta do transmissor de cada datagrama recebido.
- Dados transmitidos podem ser recebidos fora de ordem ou perdidos.

Interação cliente-servidor UDP

Exemplo de aplicação cliente-servidor UDP



Exemplo: cliente UDP Java

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

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

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

      DatagramSocket clientSocket = new DatagramSocket();

      InetAddress IPAddress = InetAddress.getByName("hostname");

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

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

Cria stream de entrada

Cria socket cliente

Traduz nome do hospedeiro para endereço IP usando DNS

Exemplo: cliente UDP Java

Cria datagrama com dados a enviar, tamanho, endereço IP 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);

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

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

Exemplo: servidor UDP Java

```
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);
```

Cria socket datagrama na porta 9876

Cria espaço para datagramas recebidos

Recebe datagrama

Exemplo: servidor UDP Java

Obtém endereço IP e número da porta do transmissor

```
String sentence = new String(receivePacket.getData());

InetAddress IPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

String capitalizedSentence = sentence.toUpperCase();

sendData = capitalizedSentence.getBytes();
```

Cria datagrama para enviar ao cliente

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

Escreve o datagrama para dentro do socket

```
serverSocket.send(sendPacket);
    }
}
}
```

Termina o while loop, retorna e espera por outro datagrama

Então...

- Fim do capítulo 2