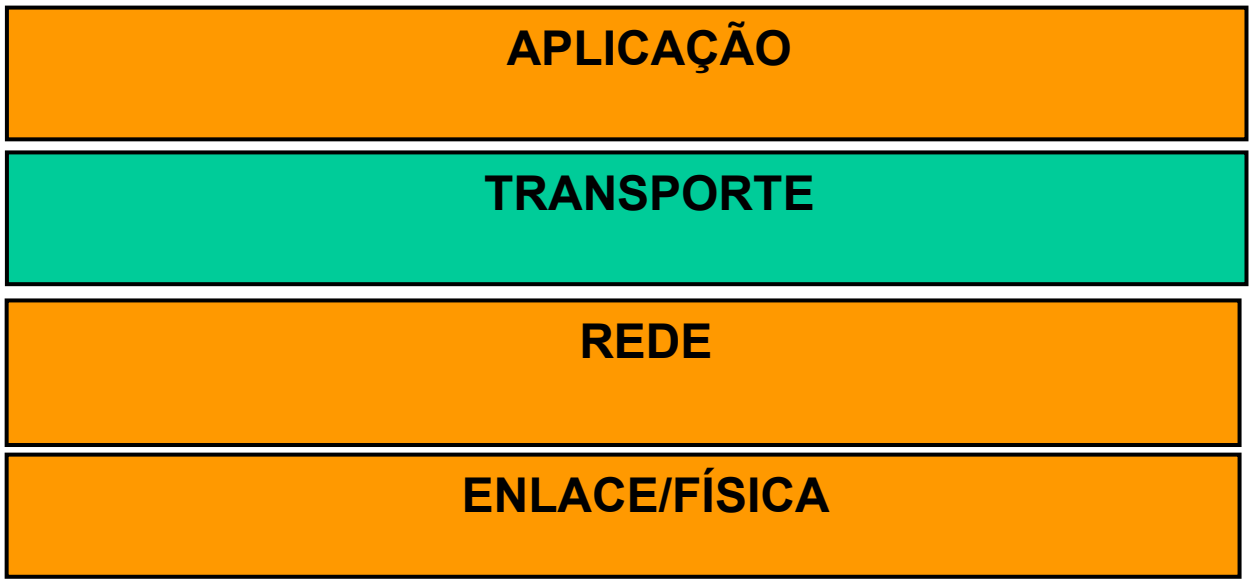
Redes de Computadores II

Programação com Sockets em Python

Prof. Ricardo Couto A. da Rocha
rcarocha@ufg.br
UFG – Regional de Catalão

Camada de Transporte

Arquitetura TCP/IP



Programação de sockets

- **Objetivo**: aprender a construir aplicações cliente-servidor que se comunicam usando sockets
- Socket API
  - Introduzida no BSD4.1 UNIX, 1981
  - Explicitamente criados, usados e liberados pelas aplicações
  - Paradigma cliente-servidor
  - Dois tipos de serviço de transporte via socket API:
  - Datagrama não confiável
  - Confiável, orientado a cadeias de bytes

Socket

Uma interface local, criada por aplicações, controlada pelo OS (uma "porta") na qual os processos de aplicação podem tanto enviar quanto receber mensagens de e para outro processo de aplicação (local ou remoto)

Chamadas de Sistema

Chamada	Tipo socket	Descrição
`socket(...)`	C ou S	Cria um socket
`bind(...)`	S ou S	associa uma porta com um socket
`connect(...)`	C	estabelece uma conexão por um socket a um destino
`listen(...)`	S	solicita a espera por conexões/pacotes em uma certa porta
`accept()`	S	aceita conexões recebidos em uma porta
`Send(...)` `recv(...)`	C ou S (TCP)	envia ou recebe bytes em uma conexão (TCP)
`sendto(...) / recvfrom(...)`	C ou S (UDP)	envia ou recebe bytes em um socket orientado a datagramas (UDP)
`close()`	C ou S	solicita o fechamento do socket (sempre unilateral)

As chamadas de sistema são as construções de mais baixo nível (do SO) para gerenciar sockets.

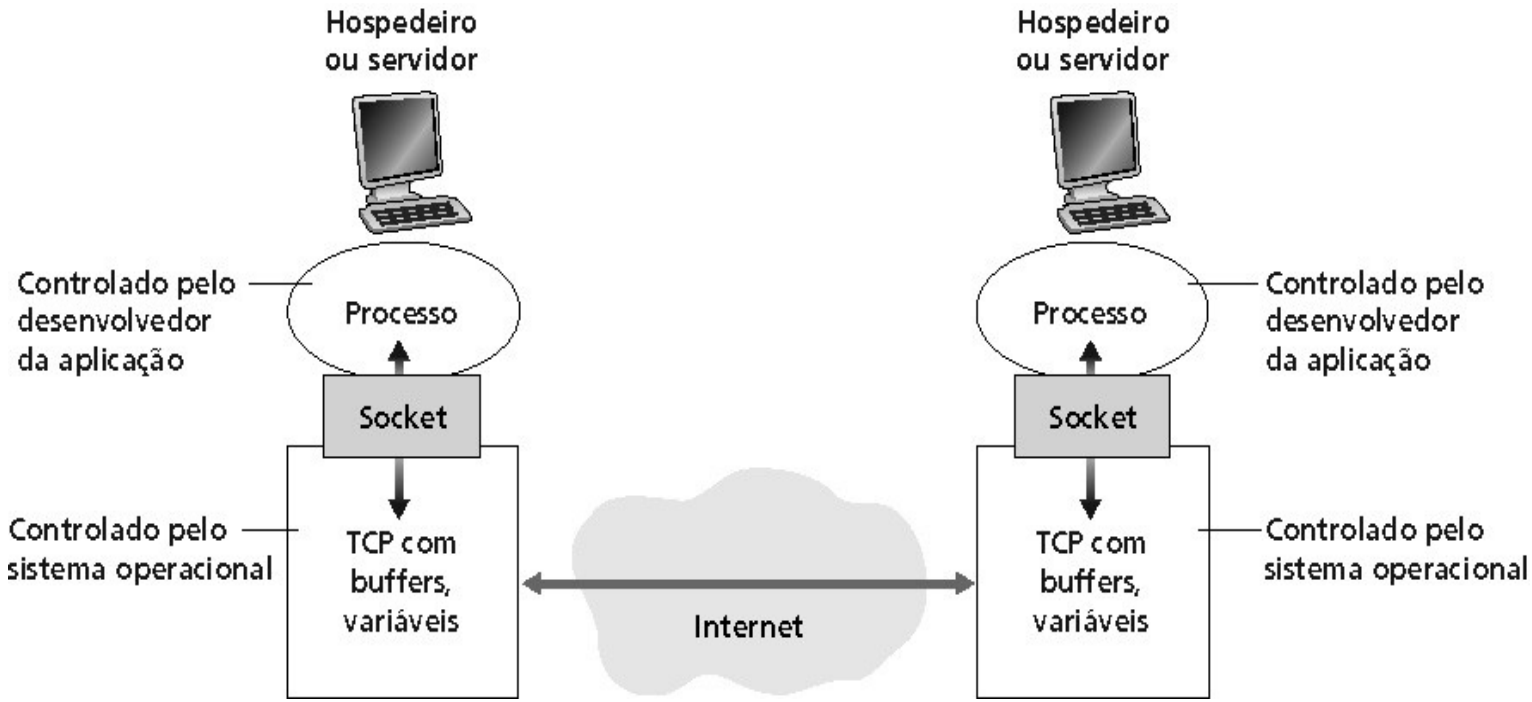
As linguagens e bibliotecas oferecem APIs que podem esconder diversos detalhes das chamadas acima.

Velocidade de Transmissão

- Velocidade do meio
  - Efetiva na camada de transporte, e não na camada de enlace
- Capacidade de transmissão (envio de **`send()`**) no socket
- Capacidade de recepção (invocação de **`recv()`**) no socket
- Tamanho dos buffers (em teoria)
  - Qual é o efeito de um buffer muito pequeno?
  - Qual é o efeito de um buffer infinito?

Programação de sockets 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 TCP

- Cliente deve contatar o servidor
  - 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
- Quando contatado pelo cliente, o TCP do 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

Ponto de vista da aplicação

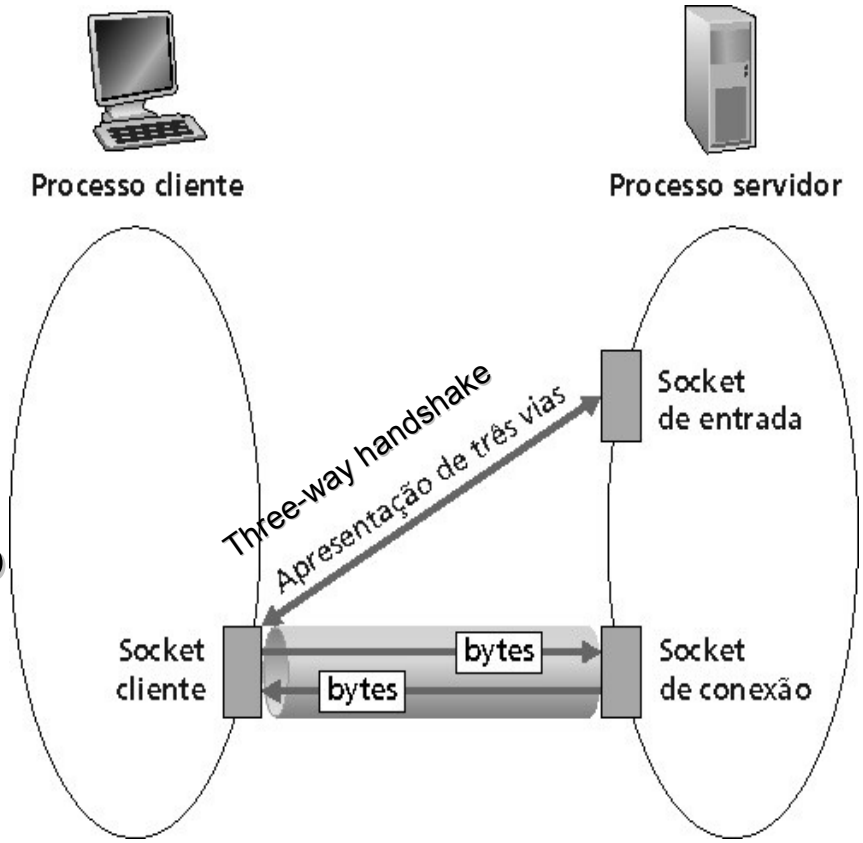
TCP fornece a transferência confiável, em ordem de bytes ("pipe") entre o cliente e o servidor

Jargão stream

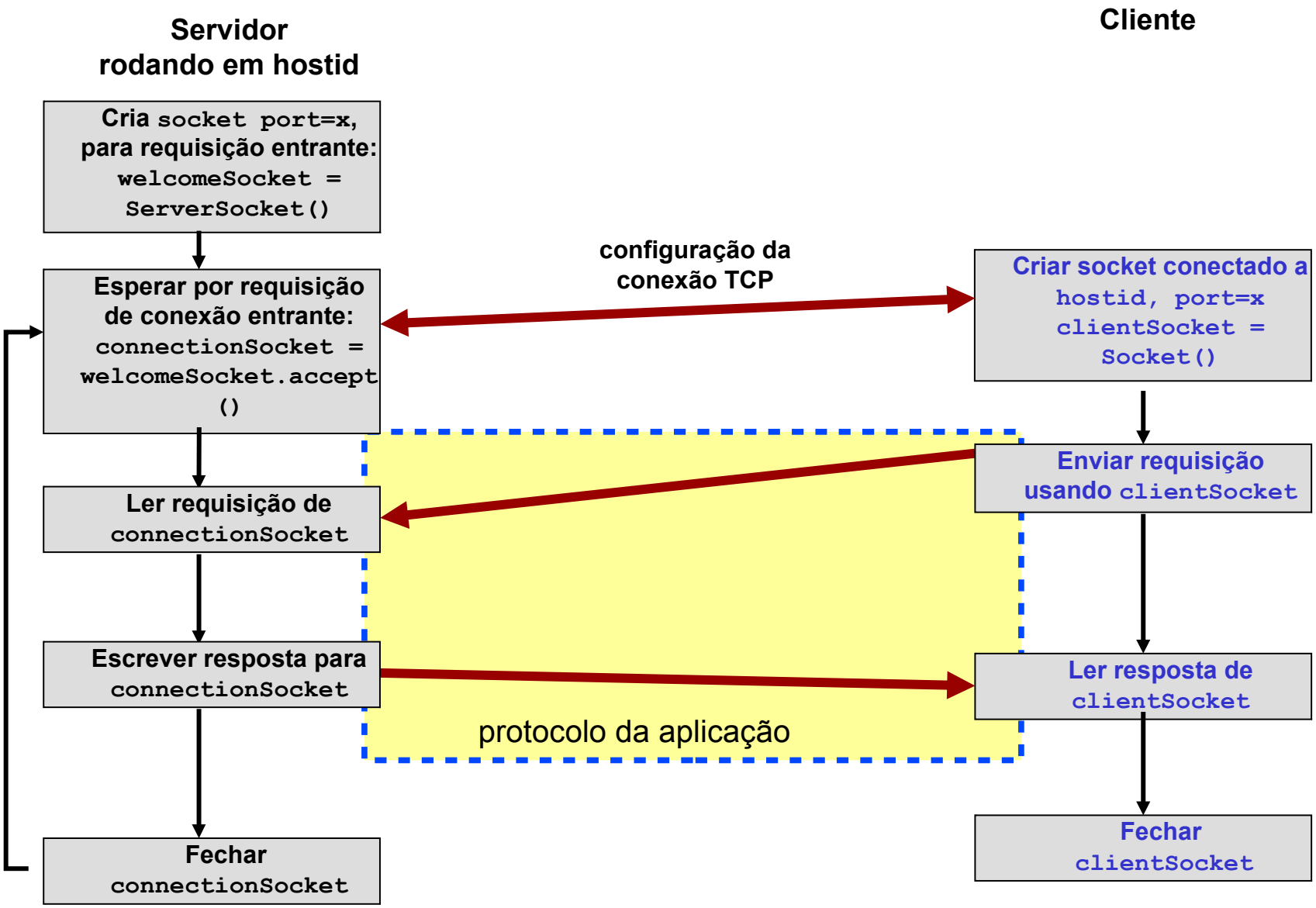
- Um **stream** é uma seqüência de caracteres que fluem para dentro ou para fora de um processo
- Um **stream de entrada** é agregado a alguma fonte de entrada para o processo, ex.: teclado ou socket
- Um **stream de saída** é agregado a uma fonte de saída, ex.: monitor ou socket
- No TCP, antes do início de uma conexão, é necessário o **handshake**, com o qual cliente TCP e servidor TCP se preparam para uma conversa.
  - Handshake é transparente para aplicações

Programação de sockets TCP

- Exemplo de aplicação cliente-servidor:
  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)



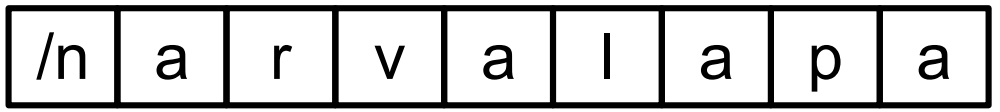
Interação cliente-servidor TCP

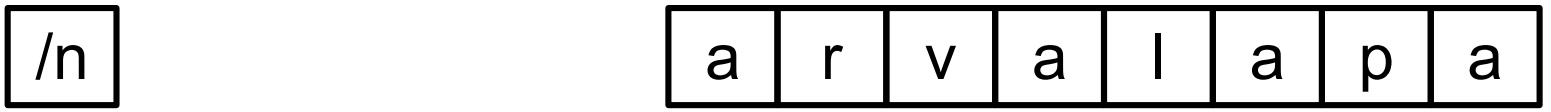


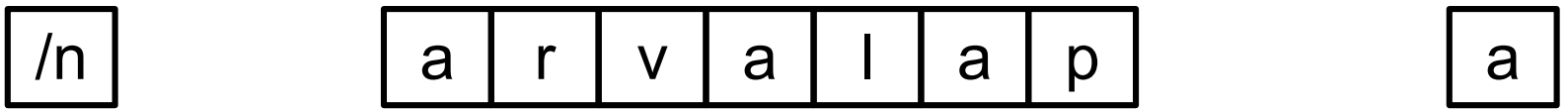
um cliente por vez

Cliente TCP
origem

Servidor TCP
destinatário







TCP oferece garantias contra erros: bytes chegarão corretos e na ordem que foram enviados. Não há nenhuma garantia de temporização.

Exemplo: Cliente Python

```
import socket

SERVIDOR = 'localhost'

PORTA = 9898

print('Iniciando cliente')

socketCliente =
socket.socket(socket.AF_INET,
socket.SOCK_STREAM)

socketCliente.connect((SERVIDOR, PORTA))

socketCliente.sendall(b'Echo\n')

data = socketCliente.recv(1024)

print('Resposta: ', data.decode("utf-8"))
```

Exemplo: Servidor Python

```
socketServidor = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
socketServidor.bind(('', 9898))
socketServidor.listen(1)
print("...esperando conexoes ...")
conexao, endereco = socketServidor.accept()
while True:
    dado = conexao.recv(1024)
    if not dado: break
    print(" >> ", dado)
    conexao.sendall(b"ok")
```

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();

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

modifiedSentence = inFromServer.readLine();

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

clientSocket.close();

    }
}
```

Envia linha para o servidor

Lê linha do servidor

Exemplo: servidor Java (TCP)

```
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()));
```

Cria socket servidor na porta 6789

Espera, no socket servidor, por contato do cliente

Cria stream de entrada ligado ao socket

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

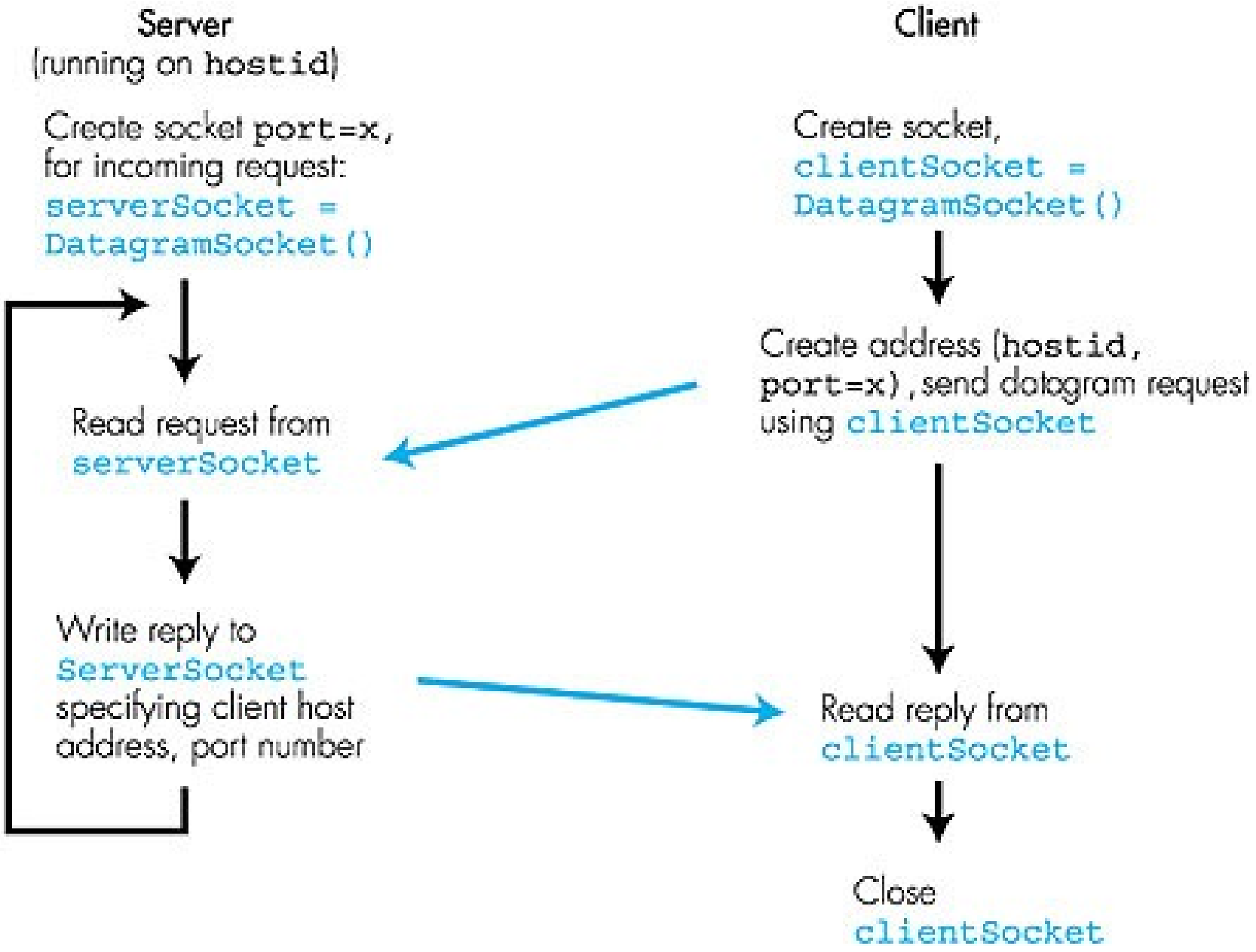
Programação de sockets UDP

- UDP: não há conexão entre o cliente e o servidor
  - Não existe apresentação
  - 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
- UDP: dados transmitidos podem ser recebidos fora de ordem ou perdidos

Ponto de vista da aplicação

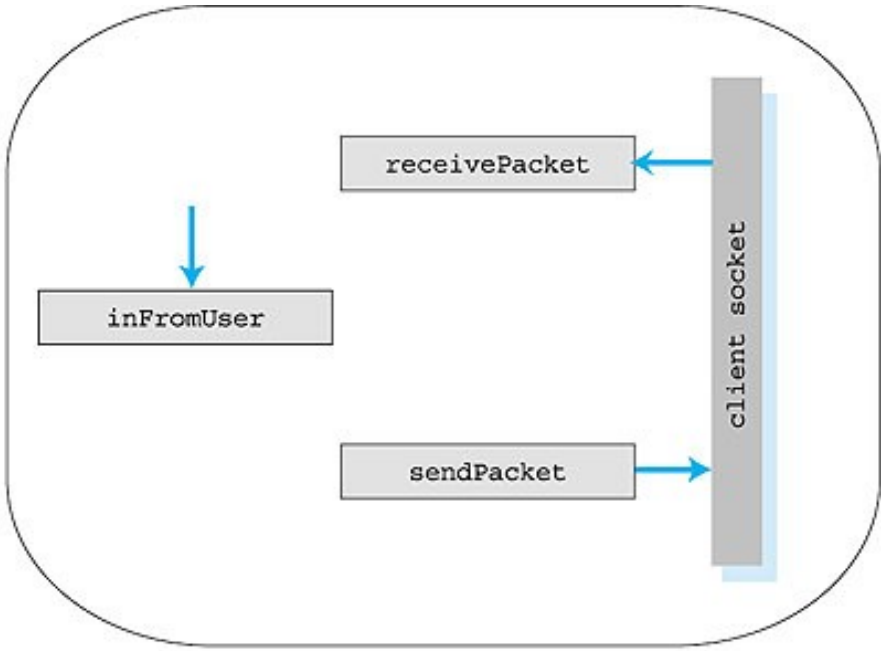
UDP fornece a transferência não confiável de grupos de bytes (datagramas) entre o cliente e o servidor

Interação cliente-servidor: UDP

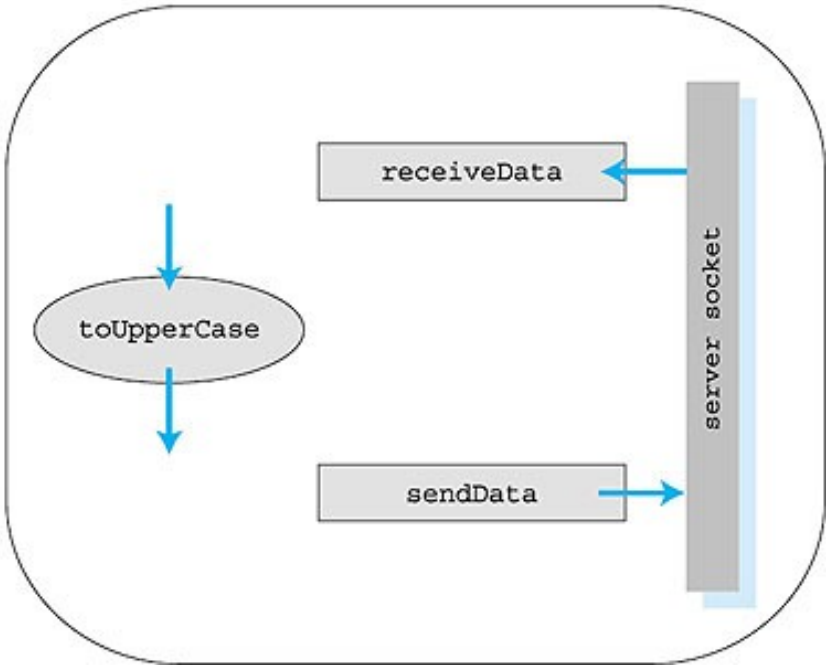


Exemplo: cliente Java (UDP)

- Lado cliente



- Lado servidor



Exemplo: cliente Java (UDP)

```
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("estacao.ufg.br");

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

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

Cria stream de entrada

Cria socket cliente

Translada nome do hospedeiro para endereço IP usando DNS

Exemplo: cliente Java (UDP)

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 Java (UDP)

```
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 Java (UDP)

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

InetAddress IPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

                String capitalizedSentence = sentence.toUpperCase();

sendData = capitalizedSentence.getBytes();

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

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

Obtém endereço IP e número da porta do transmissor

Cria datagrama para enviar ao cliente

Escreve o datagrama para dentro do socket

Termina o loop while, retorna e espera por outro datagrama