Departamento de Sistemas e Computação – FURB
Curso de Ciência da Computação
Disciplina de Sistemas Distribuídos

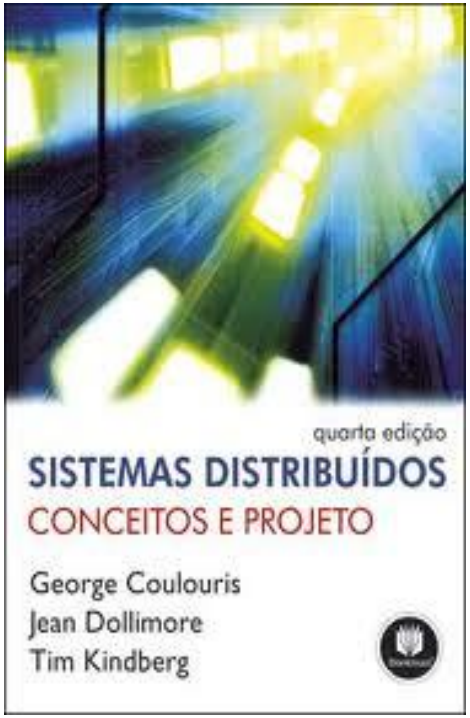
Sockets

Prof. Aurélio Hoppe
aureliof@furb.br
http://www.inf.furb.br/~aurelio/

Grupo de Processamento de Imagens,
análise de dados, robótica e
Simulação computacional

. . . aula de hoje

- Comunicação inter-processos
  - Sockets



Comunicação inter-processos

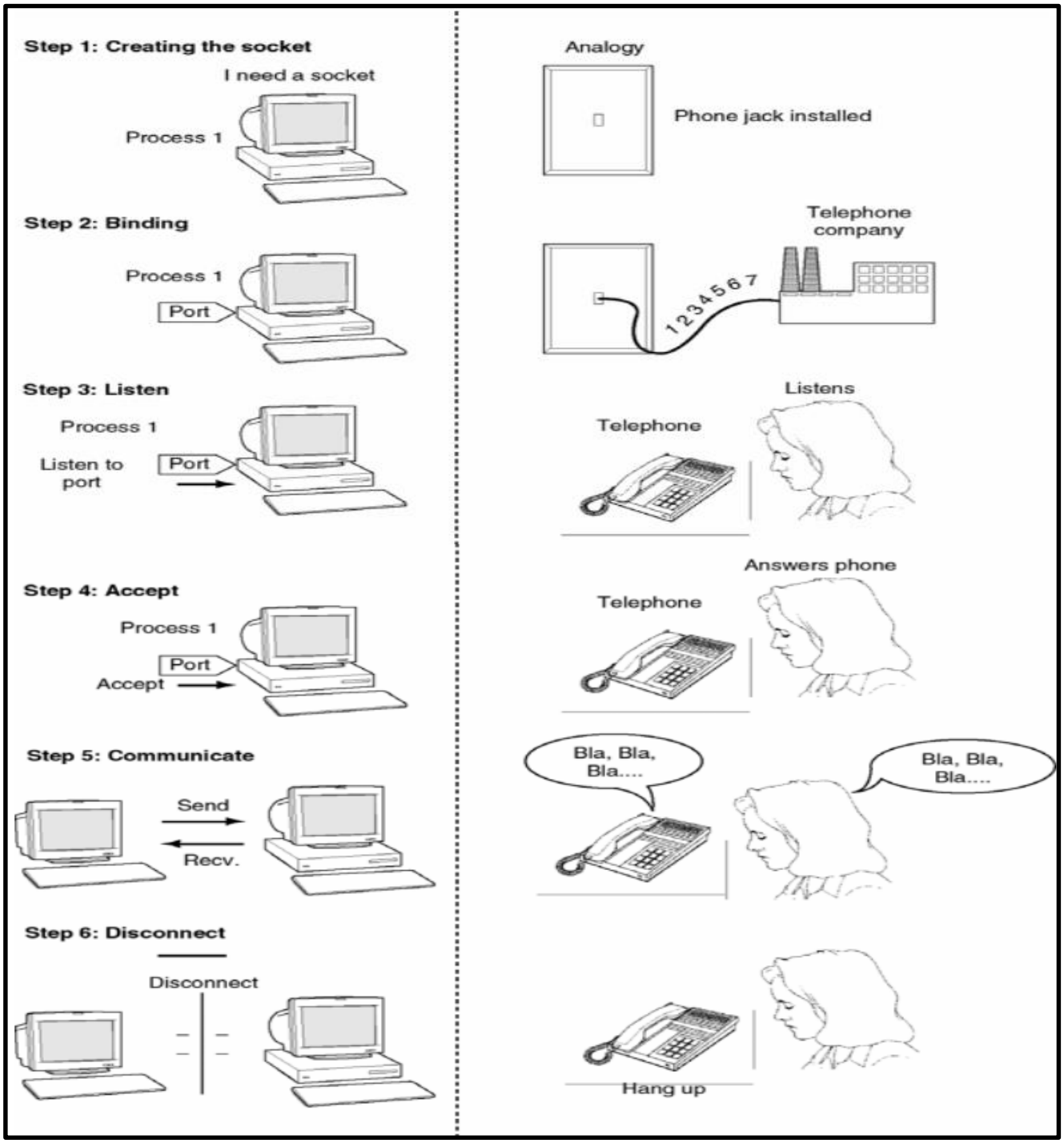
- **RMI**
  - Remote method invokation
  - Invocação remota de métodos

- **Socket**
  - Datagrama (pacote de dados na rede - UDP)
  - Stream (transferência por fluxo de bytes - TCP)

Definição

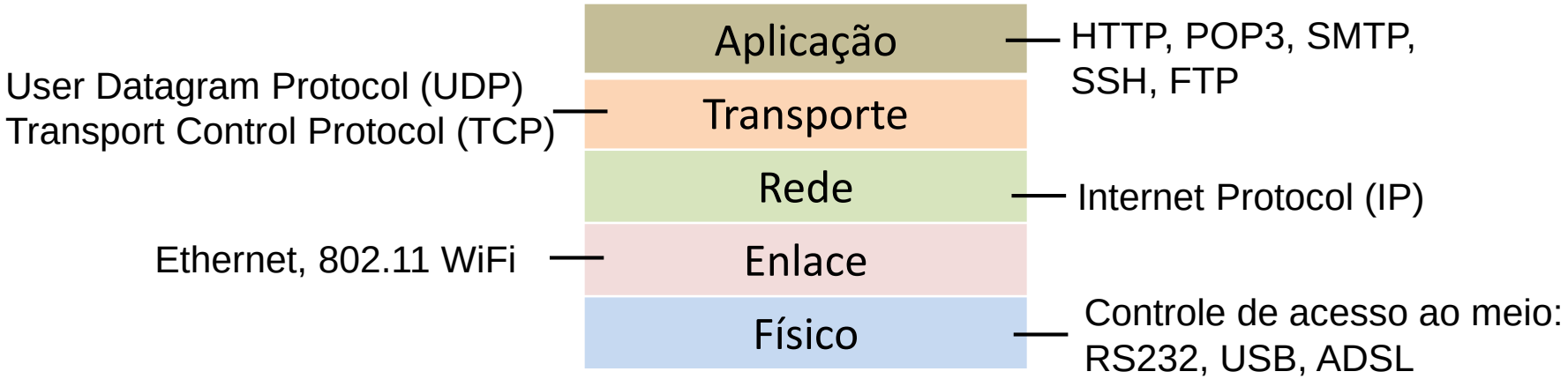
- Um Socket é um ponto final (endpoint) de um canal bidirecional de comunicação entre dois programas rodando em uma rede
- Cada Socket tem os seguintes endereços de endpoint:
    - Endereço local (número da porta) que refere-se ao endereço da porta de comunicação para camada de transporte
    - Endereço global (nome host) que refere-se ao endereço do computador (host) na rede

Sockets


Step 1: Creating the socket
I need a socket
Process 1
Step 2: Binding
Process 1
Port
Step 3: Listen
Process 1
Listen to port
Port
Step 4: Accept
Process 1
Port
Accept
Step 5: Communicate
Send
Recv.
Step 6: Disconnect
Disconnect
Analogy
Phone jack installed
Telephone company
1 2 3 4 5 6 7
Listens
Telephone
Answers phone
Telephone
Bla, Bla, Bla....
Bla, Bla, Bla....
Hang up

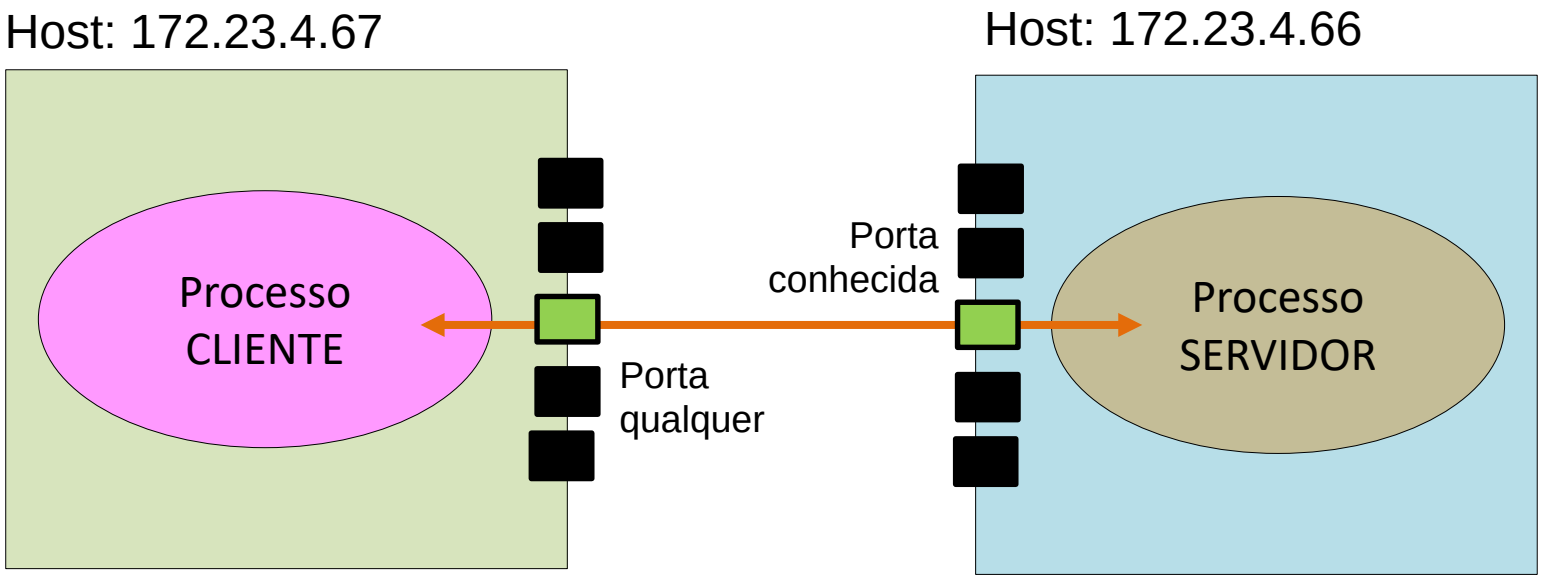
Sockets

- Os sockets UDP e TCP são a interface provida pelos respectivos protocolos na interface da camada de transporte

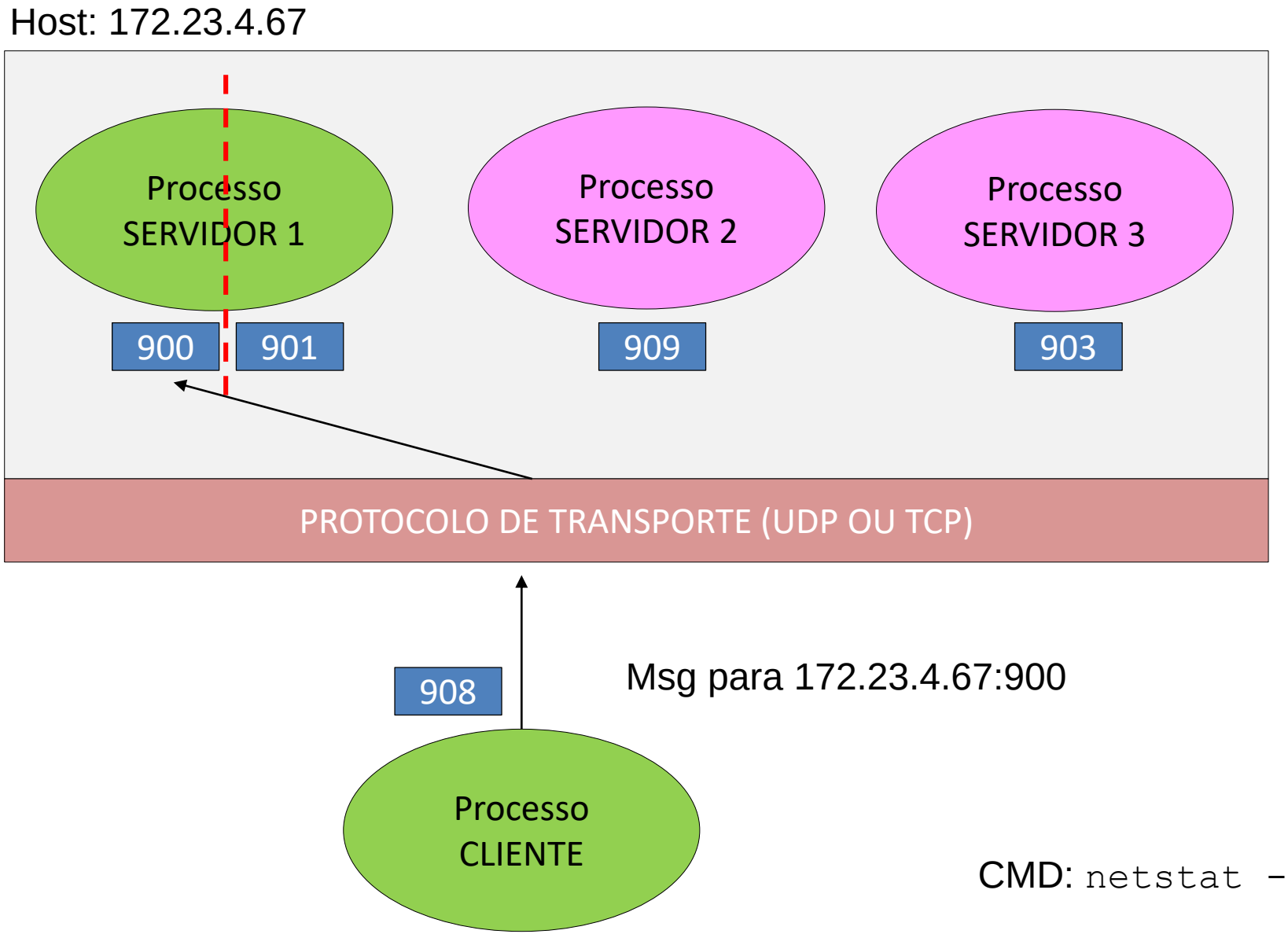


Conceito

- Utilizado para comunicação inter-processos distribuídos
- Aplicações cliente-servidor



Conceito


Host: 172.23.4.67
Processo SERVIDOR 1
Processo SERVIDOR 2
Processo SERVIDOR 3
900
901
909
903
PROTOCOLO DE TRANSPORTE (UDP OU TCP)
908
Msg para 172.23.4.67:900
Processo CLIENTE
CMD: netstat -na

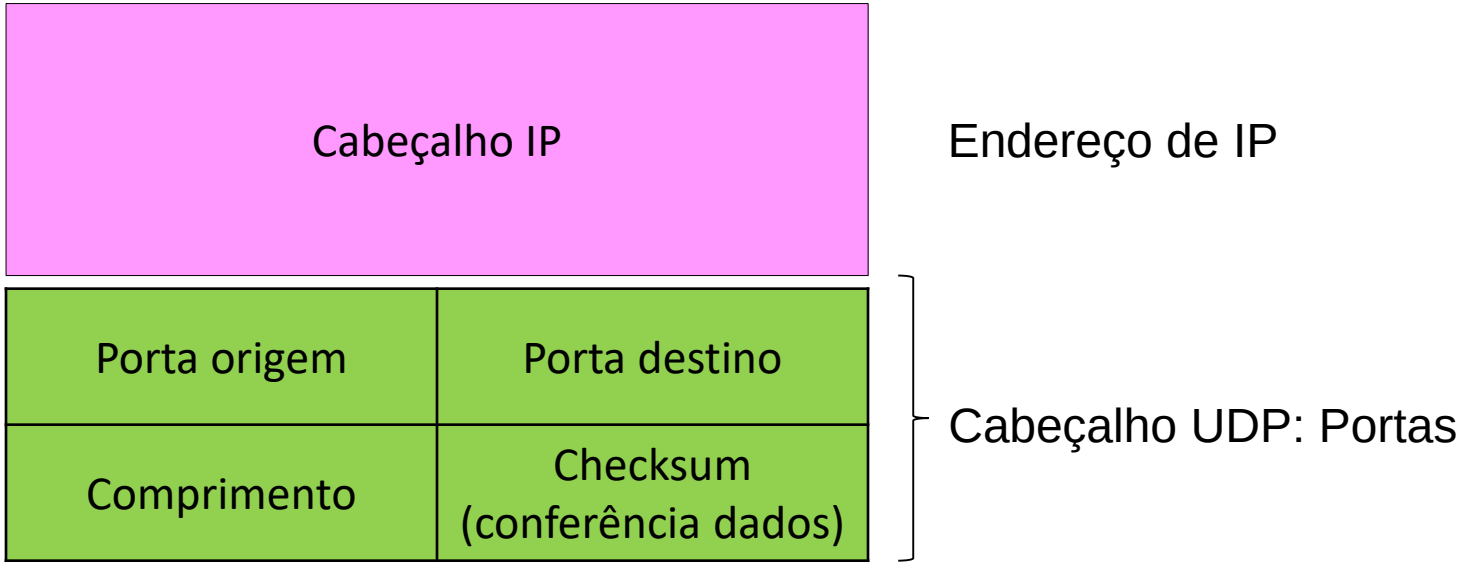
UDP -Características

- Socket UDP (User Datagram Protocol): canal não-confiável
  - Não garante entrega dos datagramas
  - Pode entregar datagramas duplicados
  - Não garante ordem de entrega dos datagramas
  - Não tem estado de conexão (escuta, estabelecida)

Mas por quê usar UDP ?

- Aplicações de tempo real; multicasting
- Velocidade crítica versus confiabilidade

Datagrama UDP



Datagrama
- Mensagem auto-contida
- Tamanho máximo: limitado pelo protocolo IPv4 (65.536 bytes)

Comandos básicos

Criar Socket

```
DatagramSocket socket = new DatagramSocket(900)
```

Receber um datagrama

```
socket.receive(req)
```

Enviar um datagrama

```
socket.send(resp)
```

Fechar um socket

```
socket.close()
```

Montar um datagrama para receber mensagem

```
new DatagramPacket(buffer, buffer.length)
```

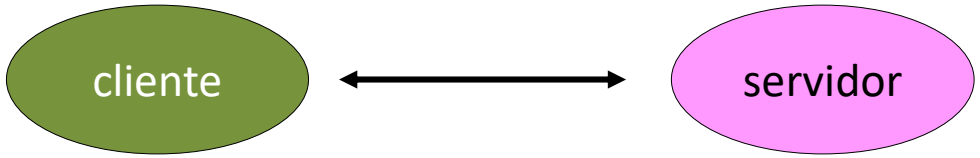
Montar um datagrama para ser enviado

```
new DatagramPacket(msg, msg.length, inet, porta)
```

Buffer e msg são byte[]

SOCKETS TCP

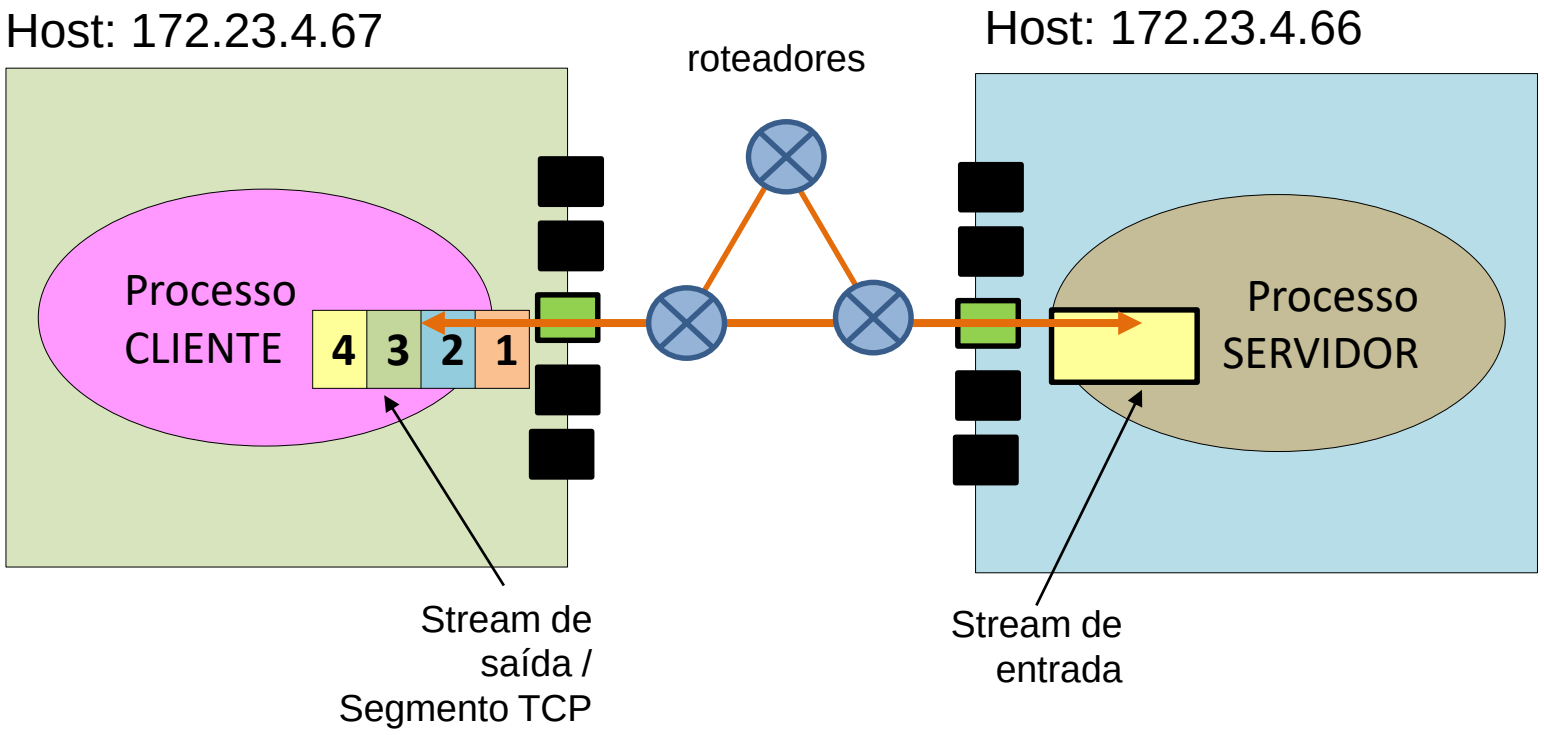
- Protocolo TCP (Transport Control Protocol) implementa um canal **confiável**
  - Do ponto de vista do desenvolvedor: fluxo contínuo (stream)
  - São fragmentados pelo TCP em segmento
  - Garante a entrega dos segmentos
  - Não há duplicação
  - Garante ordem de entrega dos segmentos
  - Possui conexão e, portanto, controla o estado de conexão (escuta, estabelecida, fechada)



SOCKETS TCP

- **Stream**
  - Um stream é uma sequência de bytes transmitida ou recebida continuamente por um processo
  - TCP preocupa-se em segmentar o stream, se necessário, e entregar os segmentos à aplicação na ordem correta.
- Para o programador de sockets TCP:
  - Basta gravar os dados num buffer de saída para que sejam enviados e
  - Ler os dados de chegada num buffer de entrada

Canal confiável



Comandos básicos

Servidor cria Socket de escuta numa porta (ex. 900)

```
ServerSocket ssocket = new ServerSocket(900)
```

Servidor aceita uma conexão e cria novo socket para atendê-la

```
Socket a = ssocket.accept()
```

Cliente cria socket de conexão

```
Socket s = new Socket("localhost", 900)
```

Cliente fecha conexão com socket

```
s.close()
```

Comandos básicos

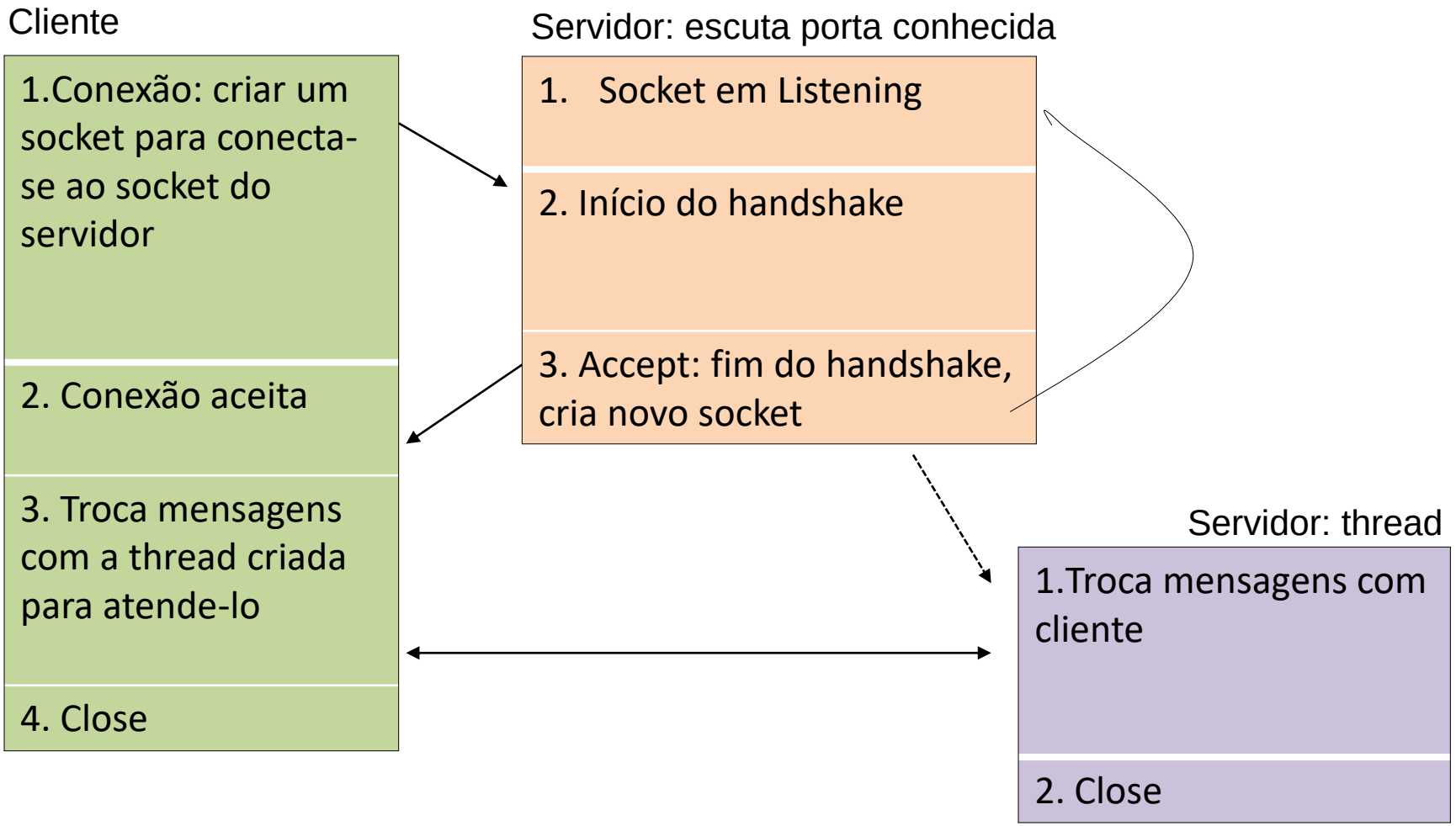
Cliente escreve no stream do socket

```
DataOutputStream sai = new DataOutputStream
                                    (sc.getOutputStream());
Sai.writeTF("msg para o servidor")
```

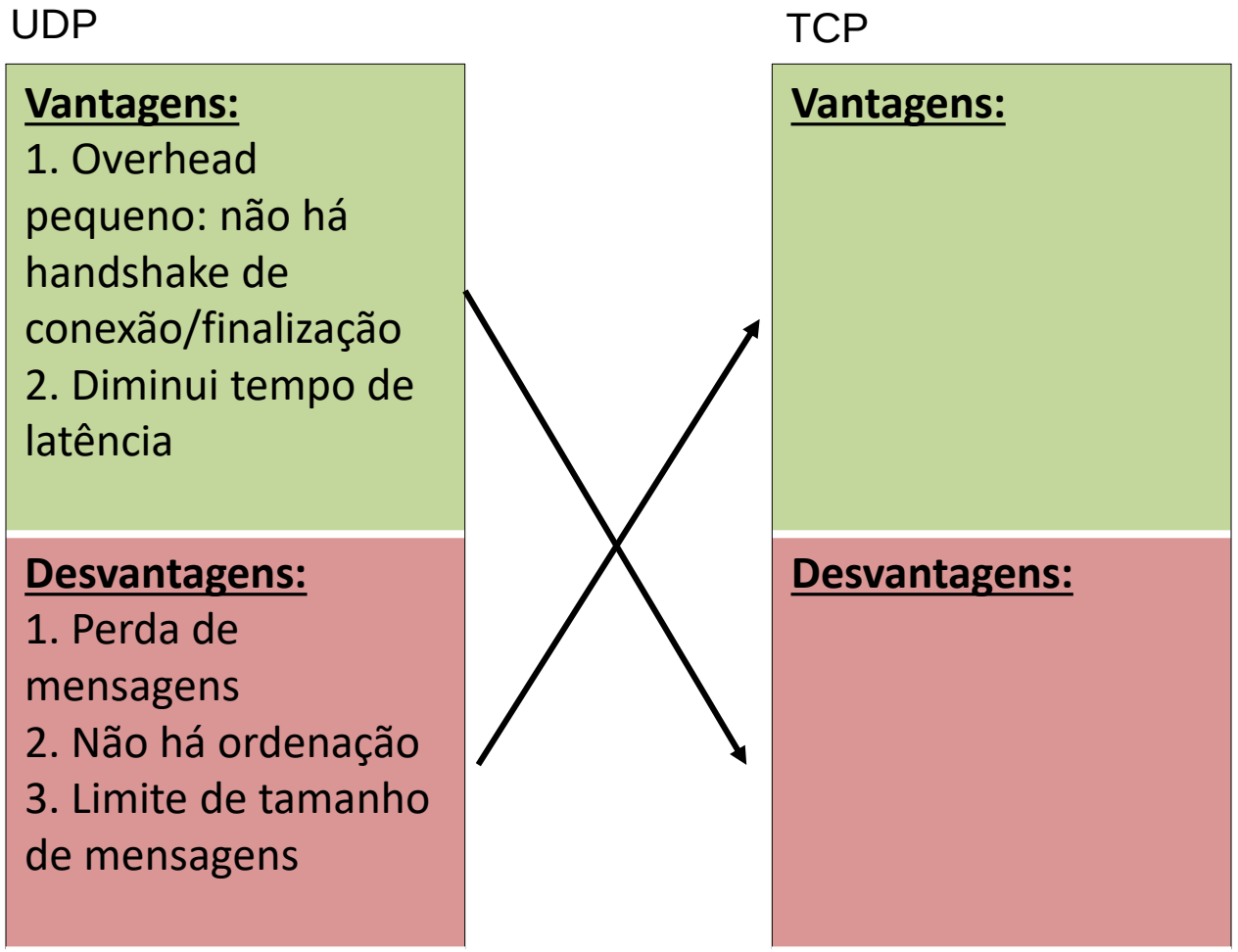
Cliente lê o stream de entrada do socket

```
DataInputStream ent = new DataInputStream
                                     (sc.getInputStream());
String recebido = ent.readUTF()
```

Esquema multi-thread



Sockets UDP x TCP



Sockets UDP x TCP