

Prof. Rogério Santos Pozza

**Universidade Tecnológica Federal do Paraná
Campus Cornélio Procopio**

Java Aplicada em Redes de Computadores

26 de abril de 2018

Na aula de hoje veremos...

- **Transmissão de dados utilizando datagramas (UDP)**
- **Endereços multicast**

UDP – User Datagram Protocol

- O protocolo UDP encapsula um datagrama em um pacote IP, porém não há garantia de entrega ao host de destino
- Ao contrário de um pacote TCP que divide os dados que ultrapassam o tamanho fixo de bytes, cada datagrama pode conter diferentes tamanhos
- O UDP também fornece os serviços de *broadcast* e *multicast* para a transmissão de pacotes
- O transmissor precisa **enviar explicitamente** o endereço IP e a porta de destino **em cada mensagem**
- Múltiplos pacotes enviados para uma máquina podem ser roteados de maneira diferente, e podem chegar em qualquer ordem

Modelo Orientado a Datagramas

- Quanto menor melhor (mais a probabilidade de entrega)
- Por segurança usa-se 512 bytes ou menos por datagrama
- O protocolo TCP libera o programador em relação ao controle

Java e UDP

A linguagem Java implementa o protocolo UDP por meio de duas classes:

- ***DatagramPacket***: armazena dados em forma de bytes
- ***DatagramSocket***: envia e recebe DatagramPackets
- Não existe algo como um **DatagramServerSocket**

A classe DatagramPacket

Usados para receber dados:

```
public DatagramPacket(byte[] buffer, int length)
```

Usados para enviar dados:

```
public DatagramPacket(byte[] buffer, int length,  
InetAddress ia, int port)
```

Métodos necessários para a comunicação:

- `getPort()` / `setPort()`
- `getData()` / `setData()`
- `getAddress()` / `setAddress()`
- `getLength()` / `setLength()`

DatagramSocket

- Um objeto da classe DatagramSocket permite tanto enviar quanto receber pacotes de datagrama (DatagramPacket)
- O método `send()` envia um pacote por vez
- O método `receive()` é bloqueante, assim como `accept()` do `ServerSocket` bloqueia a execução do programa
- O método `receive()` armazena o datagrama que veio da rede em um ***DatagramSocket*** local
- Tamanho máximo de um datagrama é 64k

DatagramSocket e DatagramPacket

- Socket utilizado por cliente ou servidor é idêntico
 - Diferem apenas pelo uso de porta anônima ou notável
- Cliente só usa anônima
- Servidor usa atribuição explícita
- Um **DatagramSocket** pode fazer ambas as operações de envio e recebimento
 - Pode receber e enviar para vários *hosts* ao mesmo tempo
 - Exceções podem ser geradas ou não

Modelo Orientado a Datagramas

- Para enviar um datagrama é preciso instanciar um objeto DatagramSocket. Esse objeto é utilizado tanto para receber quanto enviar datagramas:

```
DatagramSocket dgSocket = new DatagramSocket();
```

- Para criar um datagrama, é preciso instanciar um objeto da classe DatagramPacket, passando o dado a ser enviado, o tamanho do dado, o endereço de destino:

```
InetAddress IP = InetAddress.getByName("localhost");  
int porta = 50000;  
DatagramPacket pacoteEnviar = new DatagramPacket(dados,  
dados.length, IP, porta);
```


Modelo Orientado a Datagramas

- Para efetuar o envio, basta invocar o método **send** do objeto DatagramSocket, passando como parâmetro o pacote contendo o datagrama

```
dgSocket.send(pacoteEnviar);
```

Interação cliente/servidor em UDP

Servidor

Cria Socket para escutar determinada porta

```
DatagramSocket serverSocket =  
new DatagramSocket()
```

Lê pedido de: serverSocket
serverSocket

Escreve resposta para serverSocket
para um host e porta

Cliente

Cria Socket

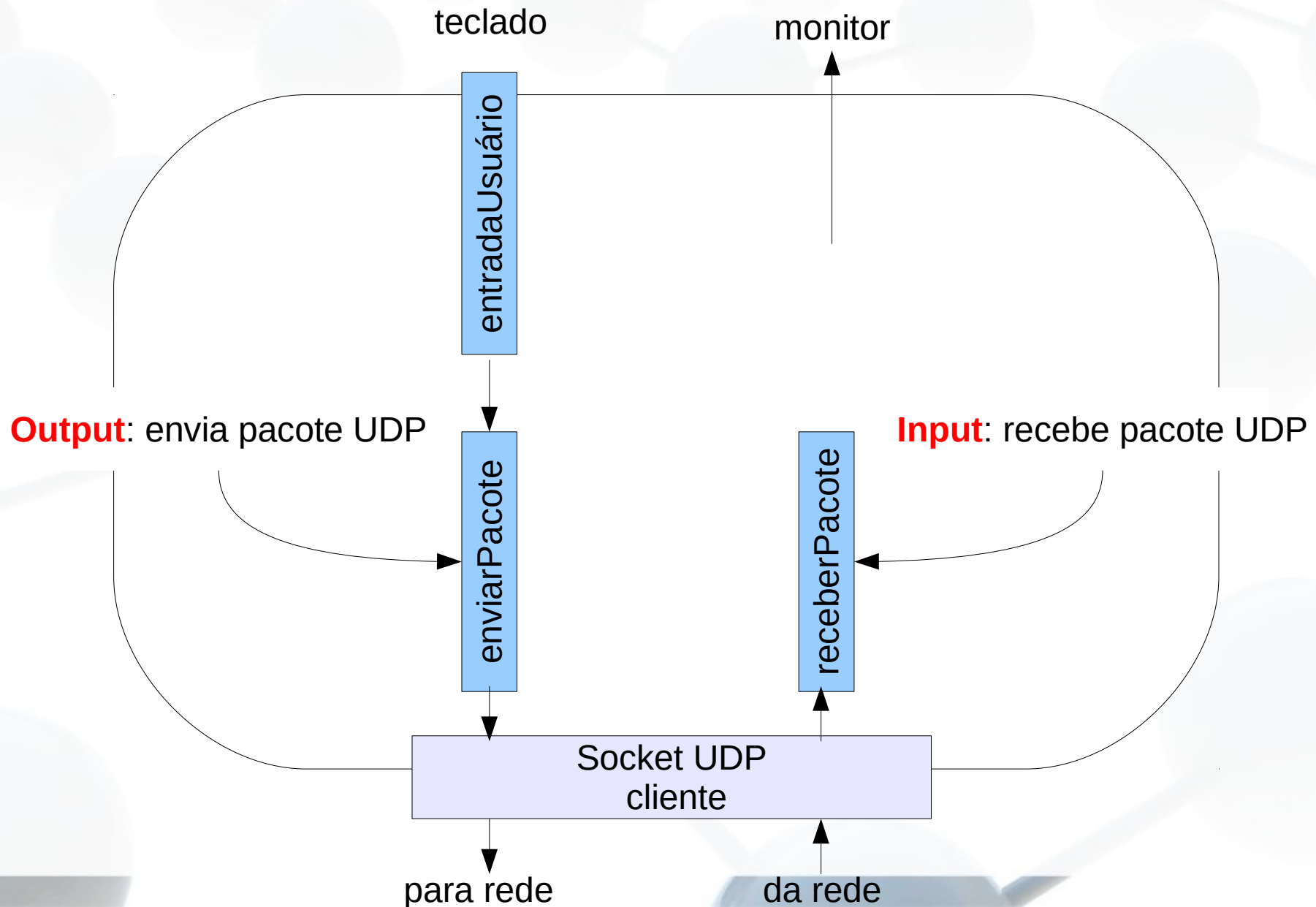
```
DatagramSocket clienteSocket =  
new DatagramSocket()
```

Envia datagrama usando clienteSocket

Lê resposta de clienteSocket

Fecha conexão clienteSocket

Modelo – Cliente Java UDP



UDPClient.java

Importar
bibliotecas

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

```
public class UDPClient {
```

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

Cria fluxo
de entrada

```
        BufferedReader entradaUsuario =  
            new BufferedReader(new InputStreamReader(System.in));
```

Cria Socket

```
        DatagramSocket clienteSocket = new DatagramSocket();
```

Traduz hostname
para IP

```
        InetAddress endereco = InetAddress.getByName("localhost");
```

```
        Byte[] dadosEnviar = new byte[1024];  
        Byte[] dadosReceber = new byte[1024];
```

```
        String mensagem = entradaUsuario.readLine();  
        dadosEnviar = mensagem.getBytes();
```

UDPClient.java

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

```
DatagramPacket pacoteEnviar =  
    new DatagramPacket(dadosEnviar, dadosEnviar.length, endereco, 60000);
```

Envia datagrama

```
clienteSocket.send(pacoteEnviar);
```

```
DatagramPacket pacoteReceber =  
    new DatagramPacket(dadosReceber,    dadosReceber.length);
```

Recebe datagrama

```
clienteSocket.receive(pacoteReceber);
```

```
// Aguardando resposta do servidor
```

```
String novaMensagem = new String(pacoteReceber.getData());
```

```
System.out.println("Mensagem do servidor: " + novaMensagem);
```

```
clienteSocket.close();
```

```
}  
}
```

UDPServer.java

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

```
public class UDPServer {
```

**Cria socket na
porta 60000**

```
    public static void main(String args[]) {
```

```
        DatagramSocket serverSocket =  
            new DatagramSocket(60000);
```

```
        Byte[] dadosReceber = new byte[1024];  
        Byte[] dadosEnviar = new byte[1024];
```

```
        while(true) {
```

```
            DatagramPacket pacoteReceber =  
                new DatagramPacket(dadosReceber, dadosReceber.length);
```

```
            serverSocket.receive(pacoteReceber);
```

```
            String mensagem = new String(pacoteReceber.getData());
```

**Reserva espaço para
Entrada de datagramas**

Recebe datagrama

UDPServer.java

Obtém endereço IP e porta do transmissor

```
InetAddress endereco = pacoteReceber.getAddress();  
int porta = pacoteReceber.getPort();
```

```
String novaMensagem = mensagem.toUpperCase();
```

```
dadosEnviar = novaMensagem.getBytes();
```

Cria datagrama
Para enviar

```
DatagramPacket enviarPacote =  
    new DatagramPacket(dadosEnviar, dadosEnviar.length, IPAddress, porta);
```

Envia o datagrama
pelo socket

```
serverSocket.send(pacoteEnviar);
```

```
}  
}
```

Classe Multicast

- A classe *Multicast* é uma especialização de um **DatagramSocket** que permite que uma aplicação receba datagramas associados a um endereço **multicast** (classe D, entre 224.0.0.1 e 239.255.255.255)
- Todos os sockets **multicast** que estejam inscritos em um endereço **multicast** recebem o **datagrama** que foi enviado para esse endereço/porta
- O método void `joinGroup()` inscreve o socket no grupo associado ao endereço **multicast** especificado como argumento
- `joinGroup(new InetSocketAddress(InetAddress endereco, int porta), NetworkInterface nif);`
 - `int porta = 60000;`
 - `InetAddress endereco = InetAddress.getByName("localhost");`
 - `NetworkInterface nif = NetworkInterface.getByName("eth0");`