

# **Relatório sobre Prática redes de computadores**

**Nome:** Jeferson Gonçalves Noronha Soriano  
**Matrícula:** 471110

## 1 INTRODUÇÃO

Primeiramente vamos entender o que é um socket, basicamente um socket é uma porta que está entre o processo de aplicação e o protocolo de transporte, para podermos fazer um processo cliente se comunicar com um processo servidor em computadores diferentes, precisamos fazer eles se comunicarem pela mesma porta, por isso utilizaremos os sockets

Vamos utilizar o Wireshark que é um programa que vigia o tráfego da rede, pegando várias informações dos dados que estão sendo trocados pela rede, como os protocolos para facilitar a análise desses protocolos e como estão rodando na rede, por isso vamos usar ele para ver como a comunicação do cliente está acontecendo com o servidor

Para o cliente se comunicar com o servidor que está rodando em uma máquina virtual que criaremos com o Virtual Box precisamos garantir que o servidor estará rodando sem parar esperando o cliente se conectar, e esse servidor tem que ter uma porta, mais preciso um socket, onde o cliente poderá se comunicar a partir de qualquer máquina

O nosso servidor pegará uma informação vinda do cliente, um string, na qual esse servidor vai transformar todos os caracteres em letra maiúscula e retornar para o cliente a mensagem dele convertida, para isso podemos usar 2 tipos de protocolos de transporte:

- **UDP: datagrama não confiável**

- onde o cliente não realiza um handshaking (uma apresentação com o servidor)

- o cliente coloca de forma explícita o seu endereço IP e porta e o servidor deve extrair

- os datagramas podem ser enviados fora de ordem ou até mesmo perdido no meio do processo

- **TCP: confiável, orientado a fluxos de bytes**

- o cliente realiza o handshaking

- o servidor espera essa comunicação acontecer e quando acontece o servidor cria um socket específico para aquele cliente

## 2 DESENVOLVIMENTO

### 2.1 Interações cliente/servidor usando o TCP

- Código: cliente TCP

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

public class TCPClient{
    public static void main(String[] args)
throws Exception {
        String sentence;
        String modifiedSentence;
        //Criar stream de entrada
        BufferedReader inFromUser = new
BufferedReader(new InputStreamReader(System.in));
        //Criar socket cliente, conectar ao
servidor
        Socket clientSocket = new
Socket("192.168.1.7",6787);
        //Criar stream de saída, ligado ao socket
        DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
        //Criar stream de entrada, ligado ao
socket
        BufferedReader inFromServer = new
BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
        sentence = inFromUser.readLine();
        //Enviar linha para o servidor
        outToServer.writeBytes(sentence + '\n');
        modifiedSentence =
inFromServer.readLine();
        //Lê linha do servidor
        System.out.println("From Server: " +
modifiedSentence);
        clientSocket.close();
    }
}
```

```
}  
  
}
```

primeiro cria um socket falando a porta que irá se conectar e o endereço ip da máquina que o server ta rodando, depois ler a string que o usuario digitar, envia para o server, e depois recebe um resposta do servidor(a string com os caracteres convertidos em maiúsculo) e imprime na tela do cliente e fecha a conexão com o server e fecha o programa

• **Código: servidor TCP**

```
import java.io.*;  
import java.net.*;  
  
public class TCPServer{  
    public static void main(String[] args)  
throws Exception {  
        String clientSentence;  
        String capitalizedSentence;  
        //Criar socket de aceitação da porta 6787  
        ServerSocket welcomeSocket = new  
ServerSocket(6787);  
        while(true){  
            //Espera do socket de aceitação por  
contato do Cliente  
            Socket connectionSocket =  
welcomeSocket.accept();  
            System.out.println("cliente conectado");  
            //Criar stream de entrada ligado ao  
socket  
            BufferedReader inFromClient = new  
BufferedReader(new  
InputStreamReader(connectionSocket.getInputStream()  
));  
            //Criar stream de saída ligado ao socket
```

```

        DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream()
);

        //Lê linha do Socket
        clientSentence =
inFromClient.readLine();

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

        //Escrever linha do Socket

outToClient.writeBytes(capitalizedSentence);

    } //Fim do Loop, retorna e espera outra
conexão do cliente

    }

}

```

o código começa criando um server socket para o servidor definindo a porta que ele vai rodar, e depois temos um loop infinito, que o que garante que o servidor vai ta sempre rodando, depois o server fica esperando um cliente se conectar a ele e quando o cliente se conecta o server cria um socket para o cliente se comunicar com ele, isso está representado no seguinte trecho de código:

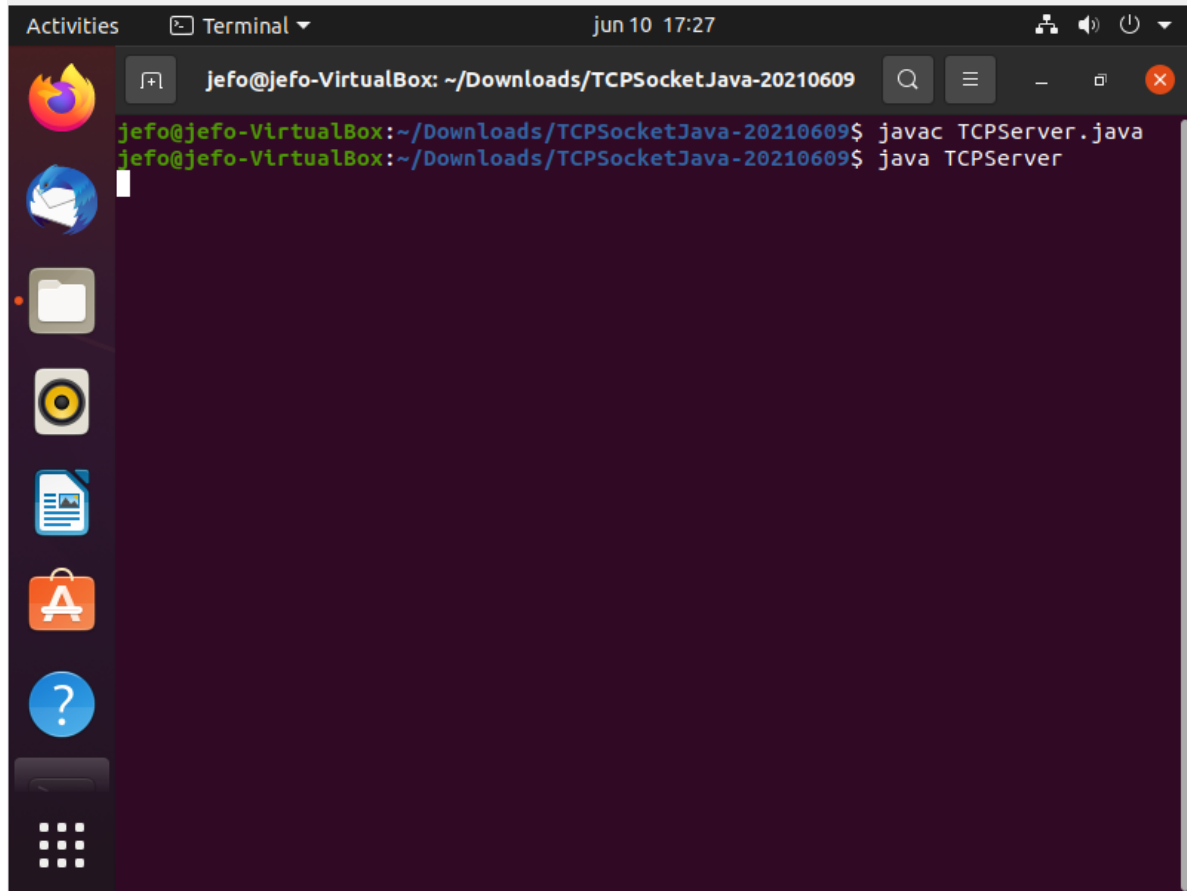
```

Socket connectionSocket =
welcomeSocket.accept();

```

depois disso o servidor pega o dado vindo do cliente, converte os caracteres em maiúsculo e envia para o cliente e volta para o início e o server vai esperar outro cliente se conectar para realizar esse processo de novo

- **Rodando o server na máquina virtual**



```
jefo@jefo-VirtualBox: ~/Downloads/TCPsocketJava-20210609
jefo@jefo-VirtualBox:~/Downloads/TCPsocketJava-20210609$ javac TCPServer.java
jefo@jefo-VirtualBox:~/Downloads/TCPsocketJava-20210609$ java TCPServer
```

#### • Rodando o cliente

agora que o server ta rodando esperando alguma máquina se conectar, vamos no nosso arquivo cliente e rodar ele e testar se esta se conectando e realizando a conexão

```
PS C:\Users\PICHAU\Desktop\5 semestre\Redes\TCP> & 'c:\Users\PICHAU\.vscode\extensions\vscjava.vscode-java-debug-0.34.0\scripts\launcher.bat' 'C:\Program Files\OpenJDK\openjdk-11.0.11_9\bin\java.exe' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\PICHAU\AppData\Roaming\Code\User\workspaceStorage\7981756c5e265d0e9506eb7251ee4b24\redhat.java\jdt_ws\TCP_bf0063c\bin' 'TCPClient'
ola mundo
From Server: resultado: OLA MUNDO
PS C:\Users\PICHAU\Desktop\5 semestre\Redes\TCP> |
```

vimos que o cliente conseguiu se conectar e conversão foi realizada

e no server foi informado que o cliente se conectou



```
jefo@jefo-VirtualBox: ~/Downloads/TCPsocketJava-20210609
jefo@jefo-VirtualBox:~/Downloads/TCPsocketJava-20210609$ javac TCPServer.java
jefo@jefo-VirtualBox:~/Downloads/TCPsocketJava-20210609$ java TCPServer
cliente conectado
```

#### • wireshark

vamos realizar tudo de novo e rastrear essa conexão com o wireshark iniciaremos o wireshark para pegar rastrear todo o tráfego

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help						
tcp.port == 6787						
	Source	Destination	Protocol	Length	Info	
00000000	192.168.1.13	192.168.1.7	TCP	64	65266 → 6787	[SYN] Seq=0 Win=64240 Len=0 MSS=
0431029	192.168.1.7	192.168.1.13	TCP	64	6787 → 65266	[SYN, ACK] Seq=0 Ack=1 Win=64240
0854043	192.168.1.13	192.168.1.7	TCP	62	65266 → 6787	[ACK] Seq=1 Ack=1 Win=64240 Len=0
9939403	192.168.1.13	192.168.1.7	TCP	62	65266 → 6787	[PSH, ACK] Seq=1 Ack=1 Win=64240
9994416	192.168.1.7	192.168.1.13	TCP	56	6787 → 65266	[ACK] Seq=1 Ack=2 Win=64239 Len=0
0958114	192.168.1.13	192.168.1.7	TCP	65	65266 → 6787	[PSH, ACK] Seq=2 Ack=1 Win=64240
0979224	192.168.1.7	192.168.1.13	TCP	56	6787 → 65266	[ACK] Seq=1 Ack=11 Win=64230 Len=0
2628718	192.168.1.7	192.168.1.13	TCP	57	6787 → 65266	[PSH, ACK] Seq=1 Ack=11 Win=64230
3898893	192.168.1.13	192.168.1.7	TCP	62	65266 → 6787	[ACK] Seq=11 Ack=2 Win=64239 Len=0
3928378	192.168.1.7	192.168.1.13	TCP	76	6787 → 65266	[PSH, ACK] Seq=2 Ack=11 Win=64239
6101496	192.168.1.13	192.168.1.7	TCP	62	65266 → 6787	[FIN, ACK] Seq=11 Ack=22 Win=64239
9286385	192.168.1.7	192.168.1.13	TCP	56	6787 → 65266	[ACK] Seq=22 Ack=12 Win=64229 Len=0

colocamos o filtro **tcp.port == 6787** que é a porta que está rodando o servidor, a ordem dos acontecimentos é, o cliente tenta abrir uma conexão com o servidor, o servidor aceita a o início da sessão e indica que esta pronto para receber os dados, o cliente envia o dados e o servidor receber esses dados faz o que tem q fazer e envia dados para o cliente, e depois o cliente informa que quer encerrar a conexão e o servidor encerra a conexão

## 2.2 Interações cliente/servidor usando o UDP

### • Código: cliente UDP

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

public 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 clienteSocket = new
DatagramSocket();
        //Traduz nome do host para o endereço
IP registrado no DNS
        InetAddress IPAddress =
InetAddress.getByName("192.168.1.7");
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
```

```

        String sentence =
inFromUser.readLine();
        sendData = sentence.getBytes();
        //Cria datagramas com: dados a enviar,
tamanho, endereço IP e porta
        DatagramPacket sendPacket = new
DatagramPacket(sendData, sendData.length,
IPAddress, 9876);
        clienteSocket.send(sendPacket);
        //Envia datagrama para o servidor
        DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);
        //Lê datagrama do servidor
        clienteSocket.receive(receivePacket);
        String modifiedSentence = new
String(receivePacket.getData());
        System.out.println("FROM SERVER: " +
modifiedSentence);
        clienteSocket.close();
    }
}

```

O cliente cria o socket do cliente, depois pega o endereço IP do servidor, depois cria uma datagrama que será enviado ao servidor, esse datagrama contém o dado(a texto que o usuário digitou) contém o endereço IP do servidor e a porta que o servidor está conectado e envia esse datagrama ao servidor, logo depois recebe um datagrama vindo do servidor que é o que usuário digitou com os caracteres em maiúsculo, mostra na tela a mensagem vinda do servidor e encerra o cliente

• **Código: servidor UDP**

```

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

//import sun.security.x509.IPAddressName;

public class UDPServer {

```



```

        public static void main(String[] args)
throws Exception{
            //Cria Socket Datagrama na porta 9876
            DatagramSocket serverSocket = new
DatagramSocket(9876);
            byte[] receiveData = new byte[1024];
            byte[] sendData = new byte[1024];
            while(true){
                //Cria espaço para o datagrama
recebidos
                DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);
                //Recebe Datagrama

serverSocket.receive(receivePacket);
                String sentence = new
String(receivePacket.getData());
                //Obtem endereço IP e numero da
porta do transmissor
                InetAddress IPAddress =
receivePacket.getAddress();
                int port =
receivePacket.getPort();
                String capitalizedSentence =
"resultado: " + sentence.toUpperCase();
                sendData =
capitalizedSentence.getBytes();
                //Cria datagrama para enviar ao
cliente
                DatagramPacket sendPacket = new
DatagramPacket(sendData, sendData.length,
IPAddress, port);
                //Escreve datagrama para dentro do
socket

```

```

serverSocket.send(sendPacket);
    } //Termina o loop, retorna e espera
por outro datagrama
    }
}

```

O server começa criando um socket na porta 9876 e depois tem um loop infinito que deixará o server rodando direto, a primeira coisa que acontece é o servidor pegar o datagrama vindo do cliente, desse datagrama, ela pega o IP e a porta que o cliente mandou, ai o servidor realiza a conversão de todos os caracteres para maiúsculas, depois cria uma datagrama onde vai ter a dado( o texto em maiúsculo) o IP e a porta e envia para o cliente

#### • Rodando o server na máquina virtual

```

jefo@jefo-VirtualBox: ~/Downloads/UDPSocketJava-20210610
jefo@jefo-VirtualBox:~/Downloads/UDPSocketJava-20210610$ javac UDPServer.java
jefo@jefo-VirtualBox:~/Downloads/UDPSocketJava-20210610$ java UDPServer

```

#### • Rodando o cliente

agora que o server ta rodando esperando alguma máquina enviar arquivo vamos realizar esse envio e ver se tudo está rodando corretamente

```

PS C:\Users\PICHAU\Desktop\5 semestre\Redes\UDP> & 'c:\Users\PICHAU\.vscode\extensions\vscjava.vscode-java-debug-0.34.0\scripts\launcher.bat' 'C:\Program Files\OpenJDK\openjdk-11.0.11_9\bin\java.exe' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\PICHAU\AppData\Roaming\Code\User\workspaceStorage\f3d51afdc0692295c56df8512ebc40f\redhat.java\jdt_ws\UDP_bf700a1c\bin' 'UDPClient'
iae amigo
FROM SERVER: resultado: IAE AMIGO

```

o conexão esta funcionando e o server esta realizando a conversão dos caracteres para maiúsculas

#### • wireshark

vamos realizar tudo de novo e rastrear essa conexão com o wireshark iniciaremos o wireshark para pegar rastrear todo o tráfego

Filter: udp.port == 9876

No.	Time	Source	Destination	Protocol	Length	Info
13	33.0146401...	192.168.1.13	192.168.1.7	UDP	62	61625 → 9876 Len=9
14	33.0152086...	192.168.1.7	192.168.1.13	UDP	1079	9876 → 61625 Len=1035

colocamos o filtro **udp.port == 9876** para mostrar so as conexões UDP na porta 9876, aqui percebemos que diferente do TCP aqui nao tem toda a parte do handshaking, no UDP o cliente envia um dado para servidor, o servidor faz toda a conversão que ele tem q fazer e retorna para o cliente e pronto