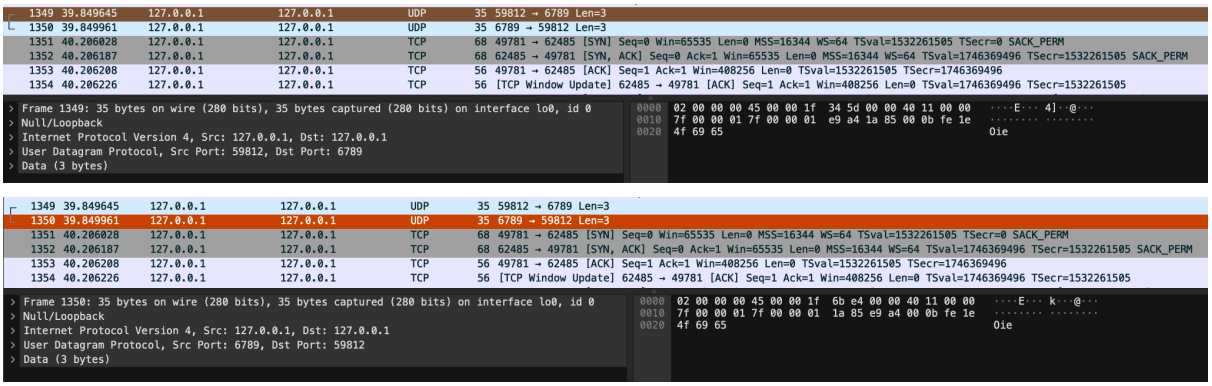
Resultados dos Testes

Cliente UDP ➛ Servidor UDP

```
~/Dropbox/socket (0.344s)
javac UDPClient.java && java UDPClient "Oie" 127.0.0.1
Reply: Oie
```

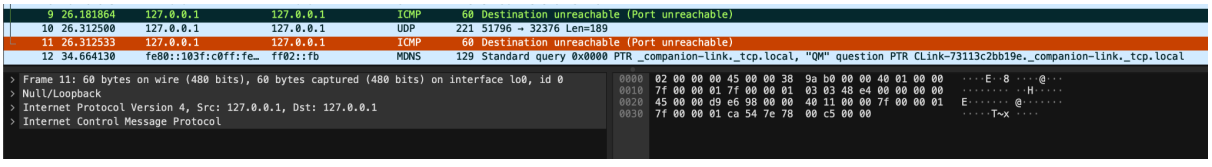
```
Last login: Sat Mar  2 22:29:22 on console
alisson@Alissons-MacBook-Air socket % javac UDPServer.java && java UDPServer
```



Cliente UDP ➛ Servidor TCP

```
~/Dropbox/socket
javac UDPClient.java && java UDPClient "Oie" 127.0.0.1
```

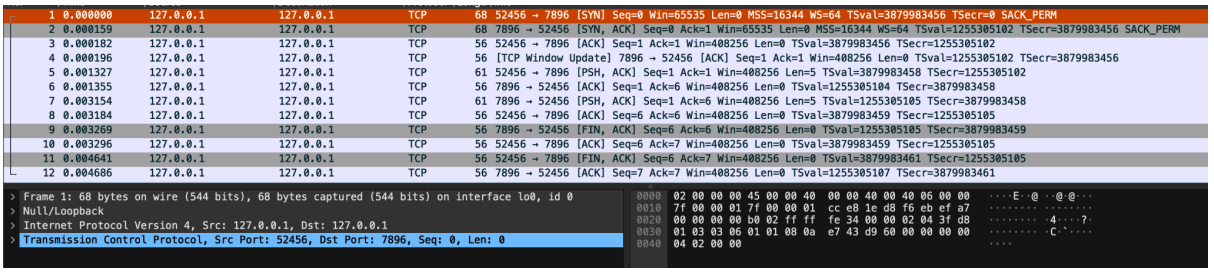
```
alisson@Alissons-MacBook-Air socket % javac TCPServer.java && java TCPServer
```



Cliente TCP ➛ Servidor TCP

```
~/Dropbox/socket (0.472s)
yes | javac TCPClient.java && java TCPClient "Oie" 127.0.0.1
Received: Oie
```

```
alisson@Alissons-MacBook-Air socket % javac TCPServer.java && java TCPServer
```

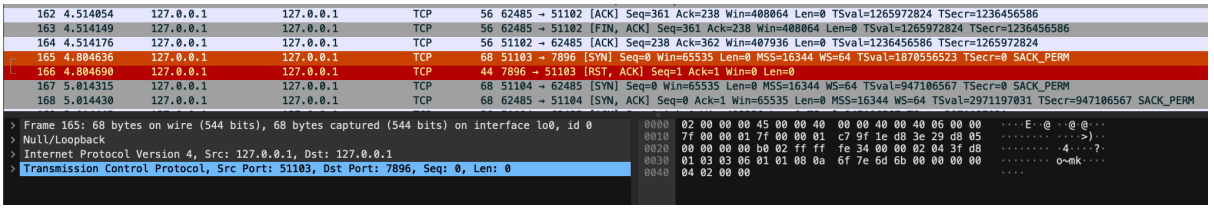


Cliente TCP ➛ Servidor UDP

```
nvim TCPClient.java

~/Dropbox/socket (0.345s)
javac TCPClient.java && java TCPClient "Oie" 127.0.0.1
IO:Connection refused
```

```
Last login: Sat Mar  2 22:29:22 on console
alisson@Alissons-MacBook-Air socket % javac UDPServer.java && java UDPServer
```



Codigo Usado

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

public class UDPClient {
    public static void main(String args[]) {
        // args fornece o conteúdo da mensagem e o nome de host do servidor
        DatagramSocket aSocket = null;
        try {
            aSocket = new DatagramSocket();
            byte[] m = args[0].getBytes();
            InetAddress aHost = InetAddress.getByName(args[1]);
            int serverPort = 6789;
            DatagramPacket request = new DatagramPacket(m, m.length, aHost,
serverPort);aSocket.send(request);
            byte[] buffer = new byte[1000];
            DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
            aSocket.receive(reply);
            System.out.println("Reply: " + new String(reply.getData()));
        } catch (SocketException e) {
            System.out.println("Socket: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO: " + e.getMessage());
        } finally {
            if (aSocket != null)
                aSocket.close();
        }
    }
}
```

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

public class TCPClient {
    public static void main(String args[]) {
        // Os argumentos fornecem a mensagem e o nome de host de destino
        Socket s = null;
        try {
            int serverPort = 7896;
            s = new Socket(args[1], serverPort); // Conexão com o servidor especificado
            DataInputStream in = new DataInputStream(s.getInputStream());
            DataOutputStream out = new DataOutputStream(s.getOutputStream());

            out.writeUTF(args[0]); // Envia a mensagem para o servidor

            String data = in.readUTF(); // Aguarda a resposta do servidor
            System.out.println("Received: " + data); // Exibe a mensagem recebida do servidor

        } catch (UnknownHostException e) {
            System.out.println("Sock:" + e.getMessage()); // Trata exceção de host
desconhe}catch (EOFException e) {
            System.out.println("EOF:" + e.getMessage()); // Trata exceção de fim de arquivo
        } catch (IOException e) {
            System.out.println("IO:" + e.getMessage()); // Trata exceção de entrada/saída
        } finally {
            // Fecha a conexão com o servidor
            if (s != null)
                try {
                    s.close();
                } catch (IOException e) {
                    /* Falha ao fechar a conexão */
                }
        }
    }
}
```

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

public class UDPServer {
    public static void main(String args[]) {
        DatagramSocket aSocket = null;
        try {
            aSocket = new DatagramSocket(6789);
            byte[] buffer = new byte[1000];
            while (true) {
                DatagramPacket request = new DatagramPacket(buffer, buffer.length);
                aSocket.receive(request);
                // Verificar se dados foram recebidos
                if (request.getLength() > 0) {
                    // Criar pacote pra responder dados recebidos
                    DatagramPacket reply = new DatagramPacket(request.getData(),
                            request.getLength(), request.getAddress(),
request.getPort()); aSocket.send(reply);
                } else {
                    // Mostrar erro
                    System.out.println("No data received from client.");
                }
            }
        } catch (SocketException e) {
            System.out.println("Socket: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO: " + e.getMessage());
        } finally {
            if (aSocket != null) aSocket.close();
        }
    }
}
```

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

public class TCPServer {
    public static void main(String args[]) {
        try {
            int serverPort = 7896;
            // Cria um socket de servidor e o associa à porta especificada
            ServerSocket listenSocket = new ServerSocket(serverPort);

            // Loop infinito para aceitar conexões de clientes
            while (true) {
                // Aguarda e aceita a conexão de um cliente
                Socket clientSocket = listenSocket.accept();

                // Inicia uma nova thread para lidar com a conexão do cliente
                Connection c = new Connection(clientSocket);
            }
        } catch (IOException e) {
            System.out.println("Listen :" + e.getMessage());
        }
    }
}

class Connection extends Thread {
    DataInputStream in;
    DataOutputStream out;
    Socket clientSocket;

    public Connection(Socket aClientSocket) {
        try {
            clientSocket = aClientSocket;
            // Cria fluxos de entrada e saída para comunicação com o cliente
            in = new DataInputStream(clientSocket.getInputStream());
            out = new DataOutputStream(clientSocket.getOutputStream());
            this.start();
        } catch (IOException e) {
            System.out.println("Connection:" + e.getMessage());
        }
    }

    public void run() {
        try {
            // Lê os dados enviados pelo cliente e os envia de volta
            String data = in.readUTF();
            out.writeUTF(data);
        } catch (EOFException e) {
            System.out.println("EOF:" + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO:" + e.getMessage());
        } finally {
            try {
                // Fecha o socket do cliente após a comunicação
                clientSocket.close();
            } catch (IOException e) {
                // TODO: tratamento de erro se o fechamento do socket falhar
            }
        }
    }
}
```