**Exp: no: 8**

Demonstration of Client server-based UDP applications using socket programming

**Aim:**

To implement and demonstrate a client-server application using **UDP (User Datagram Protocol)** for communication between two systems, showcasing the basics of socket programming and UDP’s connectionless, unreliable nature and to write the syntax, execute and place the screenshot for all the commands worked on.

1. **Write a client-server program to implement a UDP application.**

**Description:**

In this lab task, we’ll create a simple Client-Server application using UDP. UDP is a connectionless protocol that allows data to be sent without establishing a prior connection. This makes it faster but also less reliable than TCP, as it doesn’t guarantee message delivery or order.

The client will send a message to the server, and the server will respond back with a confirmation message. This application will help understand the basics of UDP socket programming, data packet transmission, and handling message exchanges over a network.

**Procedure:**

1. **Set Up the Environment:**

* Use Java for the client-server application. Ensure JDK is installed and use an editor like Eclipse or a terminal.

1. **Server Implementation:**

* The server binds to a port (5000), listens for client connections, receives messages using DataInputStream, and prints them.
* The connection remains active until the client sends "Over".

1. **Client Implementation:**

* The client connects to the server using its IP (127.0.0.1) and port (5000).
* It reads user input, sends it to the server using DataOutputStream, and terminates on "Over".

1. **Run the Programs:**

* Compile and run the Server.java file first. Then, compile and run Client.java to establish the connection and exchange messages.

**Code:**

* Server Code (Server.java):

import java.net.\*;

import java.io.\*;

 public class Server

{

    private Socket          socket   = null;

    private ServerSocket    server   = null;

    private DataInputStream in       =  null;

    public Server(int port)

    {

        try{

            server = new ServerSocket(port);

            System.out.println("Server started");

            System.out.println("Waiting for a client ...");

            socket = server.accept();

            System.out.println("Client accepted");

            in = new DataInputStream(

                new BufferedInputStream(socket.getInputStream()));

            String line = "";

            while (!line.equals("Over"))

            {

                try{

                    line = in.readUTF();

                    System.out.println(line);

                }

                catch(IOException i)

                {

                    System.out.println(i);

                }

            }

            System.out.println("Closing connection");

            socket.close();

            in.close();

        }

        catch(IOException i)

        {

            System.out.println(i);

        }

    }

    public static void main(String args[])

    {

        Server server = new Server(5000);

    }

}

* Client Code (Client.java):

import java.io.\*;

import java.net.\*;

 public class Client {

    private Socket socket = null;

    private DataInputStream input = null;

    private DataOutputStream out = null;

    public Client(String address, int port)

    {

        try {

            socket = new Socket(address, port);

            System.out.println("Connected");

            input = new DataInputStream(System.in);

            out = new DataOutputStream(

                socket.getOutputStream());

        }

        catch (UnknownHostException u) {

            System.out.println(u);

            return;

        }

        catch (IOException i) {

            System.out.println(i);

            return;

        }

        String line = "";

        while (!line.equals("Over")) {

            try {

                line = input.readLine();

                out.writeUTF(line);

            } catch (IOException i) {

                System.out.println(i);

            }

        }

        try {

            input.close();

            out.close();

            socket.close();

        } catch (IOException i) {

            System.out.println(i);

        }

    }

    public static void main(String args[])

    {

        Client client = new Client("127.0.0.1", 5000);

    }

}

**Explanation of the Code**

1. **Server Code:**
   * Creates a ServerSocket on port 5000, accepts client connections, and uses readUTF() to receive messages.
   * Prints client messages and closes the connection when "Over" is received.
2. **Client Code:**
   * Connects to the server at 127.0.0.1:5000 using a Socket.
   * Reads user input via readLine() and sends it using writeUTF().
   * Closes the connection after sending "Over".

**Output:**

* Server and Client java files stored in a separate folder

**A screenshot of a computer

Description automatically generated**

* Server.java before starting the Client.java file

A computer screen shot of a computer program

Description automatically generated

* Running Client.java file

A computer screen shot of a program

Description automatically generated

* Server.java after starting the Client.java file

A computer screen with white text

Description automatically generated

**Result:**

Thus, Client server-based UDP applications was demonstrated using socket programming and wrote the syntax, executed and placed the screenshot for all the commands worked on.