# **Name: Abdurrahman Qureshi**

# **Roll No: 242466**

Practical No: 8

Date Of Performance: 12/03/2025

Aim: To Study and Implement Socket Programming Using UDP

Lab Objectives:

The objective of this lab is to study and implement UDP (User Datagram Protocol) socket programming using Python. Students will create a client-server communication model to understand how UDP, a connectionless protocol, facilitates data transmission. The lab focuses on creating sockets, binding them to specific addresses, and sending and receiving datagrams between the client and server. This experiment will help students grasp the fundamental concepts of network communication and the differences between UDP and TCP.

Lab Outcomes:

By the end of this lab, students will be able to create and configure UDP sockets for communication between two endpoints. They will gain hands-on experience in sending and receiving data using the different java methods. Additionally, students will understand the key characteristics of UDP, such as its connectionless nature and lack of reliability guarantees, and compare it with TCP. This lab will also enhance their problem-solving skills in network programming and error handling.

Theory:

Introduction to UDP:

* UDP (User Datagram Protocol) is a transport layer protocol in the Internet Protocol (IP) suite.
* Unlike TCP (Transmission Control Protocol), UDP is connectionless, meaning it does not establish a dedicated end-to-end connection before transmitting data.
* Instead, it sends datagrams (independent packets of data) directly to the recipient without ensuring delivery, order, or error correction.
* This makes UDP faster and more efficient for applications where speed is critical, such as video streaming, online gaming, and DNS lookups.

Characteristics of UDP:

* Connectionless: No handshaking or connection setup is required before sending data.
* Unreliable: UDP does not guarantee delivery, order, or error checking. Lost or out-of-order packets are not retransmitted.
* Lightweight: UDP has minimal overhead compared to TCP, making it faster and more efficient.
* Datagram-Based: Data is sent in discrete packets called datagrams, each of which is independent of the others.
* Broadcast and Multicast Support: UDP can send data to multiple recipients simultaneously using broadcast or multicast.

Advantages of UDP:

* Low Latency: No connection setup or error recovery, making it faster than TCP.
* Efficiency: Minimal overhead due to the lack of reliability features.
* Broadcast/Multicast Support: Ideal for applications that need to send data to multiple recipients simultaneously.

Disadvantages of UDP:

* Unreliable: No guarantee of delivery, order, or error correction.
* No Congestion Control: Can overwhelm the network if not managed properly.
* Limited Data Size: Datagrams are limited in size (typically 65,507 bytes for IPv4).

Applications of UDP:

* Real-Time Applications: Video streaming, online gaming, and VoIP (e.g., Zoom, Skype).
* DNS (Domain Name System): Resolving domain names to IP addresses.
* Broadcast/Multicast Applications: Sending data to multiple recipients (e.g., live broadcasts).
* IoT (Internet of Things): Lightweight communication for devices with limited resources.

Q) Write a Java Program to Send Message “Hello Server” from client to server

CODE (Client):

import java.io.DataOutputStream;

import java.net.\*;

public class EXP8Client\_A extends Exception {

    public static void main(String[] args) {

        try {

            Socket S = new Socket("localhost", 6666);

            DataOutputStream DOS = new DataOutputStream(S.getOutputStream());

            DOS.writeUTF("Hello Server");

            DOS.flush();

            DOS.close();

            S.close();

        } catch (Exception e) {

            System.out.print(e);

        }

    }

}

CODE (Server):

import java.io.DataInputStream;

import java.net.\*;

public class EXP8Server\_A extends Exception {

    public static void main(String[] args) {

        try {

            ServerSocket SS = new ServerSocket(6666);

            Socket S = SS.accept();

            DataInputStream dis = new DataInputStream(S.getInputStream());

            String str = (String) dis.readUTF();

            System.out.println("message= " + str);

            SS.close();

        } catch (Exception e) {

            System.out.print(e);

        }

    }

}

OUPTUT:





Q) Write a Java Program to make a chat server

CODE (Client):

import java.net.\*;

import java.io.\*;

class EXP8Client\_B {

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

        Socket s = new Socket("localhost", 6666);

        DataInputStream DIN = new DataInputStream(s.getInputStream());

        DataOutputStream DOS = new DataOutputStream(s.getOutputStream());

        BufferedReader BR = new BufferedReader(new InputStreamReader(System.in));

        String str = "", str2 = "";

        while (!str.equals("stop")) {

            str = BR.readLine();

            DOS.writeUTF(str);

            DOS.flush();

            str2 = DIN.readUTF();

            System.out.println("Server: " + str2);

        }

        DOS.close();

        s.close();

    }

}

CODE (Server):

import java.net.\*;

import java.io.\*;

class EXP8Server\_B {

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

        ServerSocket SS = new ServerSocket(6666);

        Socket s = SS.accept();

        DataInputStream DIN = new DataInputStream(s.getInputStream());

        DataOutputStream DOS = new DataOutputStream(s.getOutputStream());

        BufferedReader BR = new BufferedReader(new InputStreamReader(System.in));

        String str = "", str2 = "";

        while (!str.equals("stop")) {

            str = DIN.readUTF();

            System.out.println("Client: " + str);

            str2 = BR.readLine();

            DOS.writeUTF(str2);

            DOS.flush();

        }

        DIN.close();

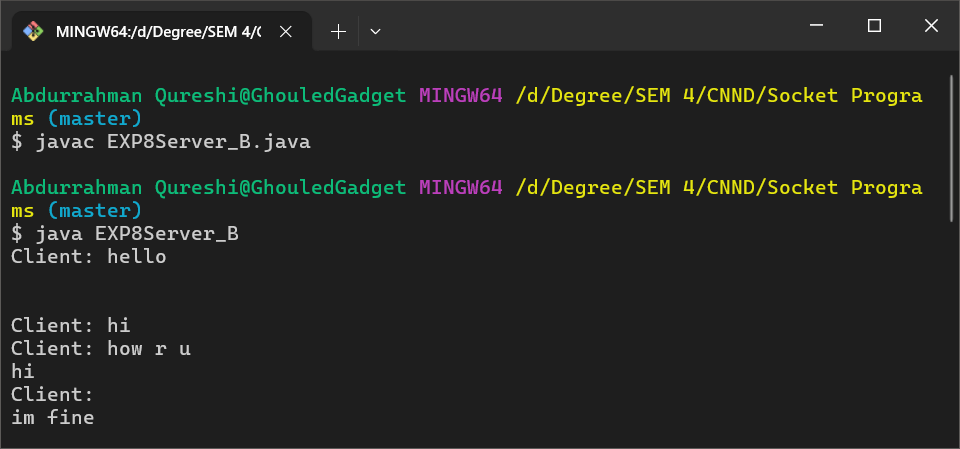
        s.close();

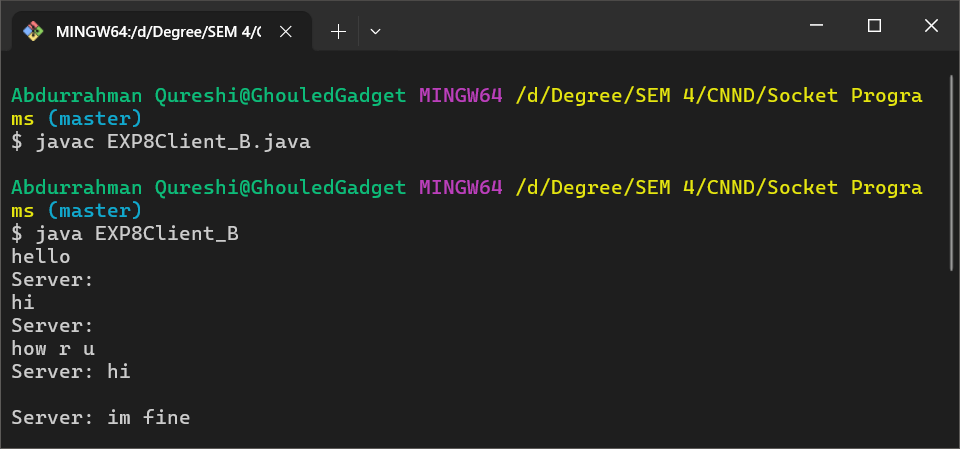
        SS.close();

    }

}

OUPTUT:





Conclusion:

This experiment offered valuable insights into the fundamentals of UDP socket programming, its advantages, and its limitations. It provided a solid foundation for understanding network communication protocols and their practical applications.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Performance  (7M) | Journal  (3M) | Lab Ethics  (2M) | Attendance  (3M) | Total  (15M) | Faculty Signature |
|  |  |  |  |  |  |