# Experiment No. 9

**Aim:** Socket programming using UDP:

**Resource required**: Java Development Kit (JDK), Text Editor / IDE, Command Prompt/ Terminal, Network

**Theory:**

Socket Programming allows two devices or applications to communicate over a network. In UDP Socket Programming, communication is based on the User Datagram Protocol (UDP), which is connectionless, faster, and does not guarantee reliable delivery of data.

UDP is suitable for applications where speed is more important than reliability, such as online gaming, video streaming, and VoIP.

UDP (User Datagram Protocol) is a connectionless transport layer protocol used to send data over a network. It allows applications to send independent packets (datagrams) without establishing a prior connection.

We'll create:

- UDPServer.java
- UDPClient.java

**Program:**

**UDP Server Code — UDPServer.java**

```
import java.net.*;


public class UDPServer {
   public static void main(String[] args) {
      try {
         DatagramSocket serverSocket = new DatagramSocket(9876);
         byte[] receiveData = new byte[1024];
         byte[] sendData;
```

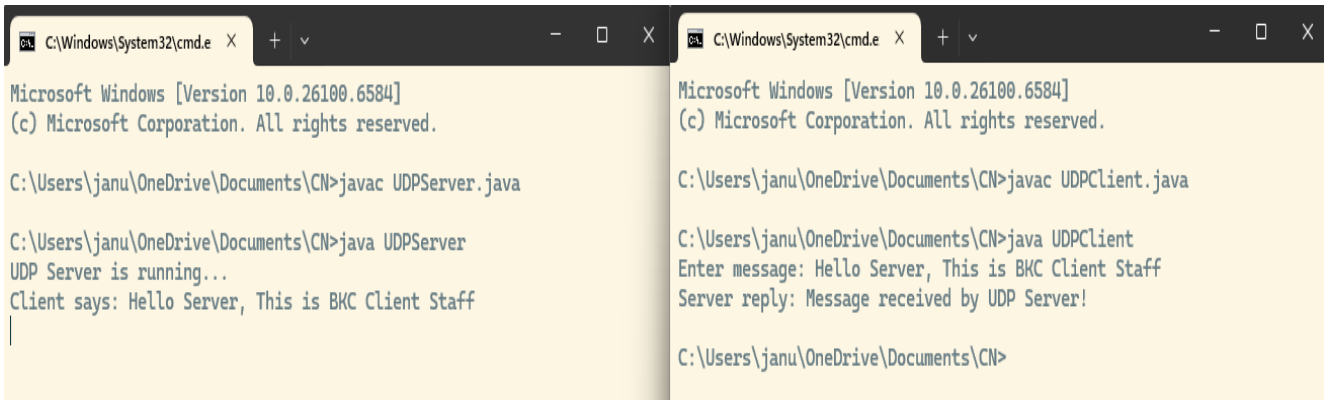
```
        System.out.println("UDP Server is running...");


        while (true) {
            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
            serverSocket.receive(receivePacket);


            String clientMessage = new String(receivePacket.getData(), 0, receivePacket.getLength());
            System.out.println("Client says: " + clientMessage);


            InetAddress clientIP = receivePacket.getAddress();
            int clientPort = receivePacket.getPort();
            String response = "Message received by UDP Server!";
            sendData = response.getBytes();


            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, clientIP, clientPort);
            serverSocket.send(sendPacket);
        }
    } catch (Exception e) {
        System.out.println(e);
    }
  }
}
```

**UDP Client Code — UDPClient.java**

```
import java.net.*;
import java.util.Scanner;


public class UDPClient {
   public static void main(String[] args) {
      try {
         DatagramSocket clientSocket = new DatagramSocket();
         InetAddress IPAddress = InetAddress.getByName("localhost");
         byte[] sendData;
         byte[] receiveData = new byte[1024];


         Scanner sc = new Scanner(System.in);
         System.out.print("Enter message: ");
         String message = sc.nextLine();


         sendData = message.getBytes();
         DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
IPAddress, 9876);
         clientSocket.send(sendPacket);


         DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
         clientSocket.receive(receivePacket);
```

```
        String serverResponse = new String(receivePacket.getData(), 0,
receivePacket.getLength());

        System.out.println("Server reply: " + serverResponse);


        clientSocket.close();

    } catch (Exception e) {

        System.out.println(e);

    }

  }

}
```

**Output:**




**Conclusion:**

In this practical, we successfully implemented Socket Programming using UDP (User Datagram Protocol) to enable communication between a client and a server.

Unlike TCP, UDP is connectionless, so the client can send data packets (datagrams) to the server without establishing a formal connection. This makes communication faster but less reliable, as packet delivery and order are not guaranteed.

From this practical, we learned:

- How to create UDP sockets using Java's DatagramSocket and DatagramPacket classes.

- How to send and receive messages between a client and server without a connection.
- The difference between connection-oriented (TCP) and connectionless (UDP) communication.

Thus, this experiment demonstrated quick, efficient, but unreliable data transmission, which is ideal for applications like streaming, online gaming, and real-time communication.