Department of Computer Engineering

Class: S.Y. B.Tech. **Semester: IV**

Course Code: DJ19CEL405 **Course Name: Computer Networks Lab**

Name: Varun Viswanath **SAP ID:60004210105**

Batch: B1 Comps

Experiment No: 9

Aim: Write a client server socket program using TCP and using UDP.

Theory:

Sockets allow communication of two processes that are running on the same or different machines. Sockets are the end of two-way communication between two programs that are running on the networks. Sockets are mostly used in client-server architecture for communication between multiple applications. Socket programming tells us how we can use socket API for creating communication between local and remote processes.

The socket is created by the combination of the IP address and port number of the software. With this combination, the process knows the system address and address of the application where data is to be sent.

UDP and TCP are two common protocols used in socket programming.

Transmission Control Protocol

TCP stands for Transmission Control Protocol. It is a transport layer protocol that facilitates the transmission of packets from source to destination. It is a connection-oriented protocol that means it establishes the connection prior to the communication that occurs between the computing devices in a network. This protocol is used with an IP protocol, so together, they are referred to as a TCP/IP.

The main functionality of the TCP is to take the data from the application layer. Then it divides the data into a several packets, provides numbering to these packets, and finally transmits these packets to the destination. The TCP, on the other side, will reassemble the packets and transmits them to the application layer.

As we know that TCP is a connection-oriented protocol, so the connection will remain established until the communication is not completed between the sender and the receiver.

Department of Computer Engineering

Class: S.Y. B.Tech. **Semester: IV**

Course Code: DJ19CEL405 **Course Name: Computer Networks Lab**

User Datagram Protocol

The UDP(User Datagram Protocol) protocol allows the computer applications to send the messages in the form of datagrams from one machine to another machine over the Internet Protocol (IP) network. The UDP is an alternative communication protocol to the TCP protocol (transmission control protocol). Like TCP, UDP provides a set of rules that governs how the data should be exchanged over the internet. The UDP works by encapsulating the data into the packet and providing its own header information to the packet. Then, this UDP packet is encapsulated to the IP packet and sent off to its destination. Both the TCP and UDP protocols send the data over the internet protocol network, so it is also known as TCP/IP and UDP/IP. There are many differences between these two protocols. UDP enables the process to process communication, whereas the TCP provides host to host communication. Since UDP sends the messages in the form of datagrams, it is considered the best-effort mode of communication. TCP sends the individual packets, so it is a reliable transport medium. Another difference is that the TCP is a connection-oriented protocol whereas, the UDP is a connectionless protocol as it does not require any virtual circuit to transfer the data.

UDP also provides a different port number to distinguish different user requests and also provides the checksum capability to verify whether the complete data has arrived or not; the IP layer does not provide these two services.

Code:

TCP

Server Side

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

public class TCPServer {
    public static void main(String[] args) {
        int port = 9999;
        try {
            ServerSocket serverSocket = new ServerSocket(port);
            System.out.println("Voting Server Started on port " + port);
            int candidate1Votes = 0;
            int candidate2Votes = 0;

            while (true) {
                Socket clientSocket = serverSocket.accept();
```

```
                System.out.println("New client connected: " +
clientSocket.getInetAddress().getHostName());

                BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(),  true);

                String clientInput = in.readLine();

                if (clientInput.equalsIgnoreCase("Candidate1")) {
                    candidate1Votes++;
                    out.println("Vote successfully registered for
Candidate1");
                } else if (clientInput.equalsIgnoreCase("Candidate2")) {
                    candidate2Votes++;
                    out.println("Vote successfully registered for
Candidate2");
                } else if (clientInput.equalsIgnoreCase("Result")) {
                    out.println("Candidate1 Votes: " + candidate1Votes +
"\nCandidate2 Votes: " + candidate2Votes);
                } else {
                    out.println("Invalid input. Please try again.");
                }

                in.close();
                out.close();
                clientSocket.close();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Client side:

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

public class TCPClient {
```

```
    public static void main(String[] args) {
        String host = "localhost";
        int port = 9999;

        try {
            Socket clientSocket = new Socket(host, port);

            BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),
true);

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

            String userInput = "";

            while (!userInput.equalsIgnoreCase("exit")) {
                System.out.println("Please enter your vote (Candidate1 or
Candidate2) or enter Result to get vote count:");
                userInput = stdIn.readLine();
                out.println(userInput);

                String serverResponse = in.readLine();
                System.out.println(serverResponse);
            }

            in.close();
            out.close();
            stdIn.close();
            clientSocket.close();

        } catch (UnknownHostException e) {
            System.err.println("Don't know about host " + host);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Department of Computer Engineering

Class: S.Y. B.Tech. **Semester: IV**

Course Code: DJ19CEL405 **Course Name: Computer Networks Lab**

Output:

```
Please enter your vote (Candidate1 or Candidate2) or enter Result to get vote count:
Candidate1
Vote successfully registered for Candidate1
Please enter your vote (Candidate1 or Candidate2) or enter Result to get vote count:
```

UDP

```
Voting Server Started on port 9999
New client connected: 127.0.0.1
```

Server side:

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

public class UDPServer {
    private static final int MAX_PACKET_SIZE = 1024;
    private static final int SERVER_PORT = 5000;
    private static HashMap<String, Integer> candidates = new HashMap<>();

    public static void main(String[] args) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(SERVER_PORT);

        System.out.println("Server is running on port " + SERVER_PORT);

        byte[] receiveData = new byte[MAX_PACKET_SIZE];
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
MAX_PACKET_SIZE);

        while (true) {
            serverSocket.receive(receivePacket);
            String candidateName = new String(receivePacket.getData()).trim();

            int currentVotes = candidates.getOrDefault(candidateName, 0);
            candidates.put(candidateName, currentVotes + 1);

            System.out.println(candidateName + " received a vote from " +
receivePacket.getAddress().getHostAddress());
        }
    }
}
```

Department of Computer Engineering

Class: S.Y. B.Tech. **Semester: IV**

Course Code: DJ19CEL405 **Course Name: Computer Networks Lab**

Client Side:

```
import java.net.*;
public class UDPClient {
    private static final int SERVER_PORT = 5000;
    private static final String SERVER_IP = "localhost";

    public static void main(String[] args) throws Exception {
        DatagramSocket clientSocket = new DatagramSocket();

        InetAddress serverAddress = InetAddress.getByName(SERVER_IP);

        System.out.println("Welcome to the voting system.");
        System.out.println("Enter the name of your candidate or 'exit' to stop
voting.");
        while (true) {
            byte[] sendData = new byte[1024];

            String candidateName = System.console().readLine();

            if (candidateName.equalsIgnoreCase("exit")) {
                break;
            }
            sendData = candidateName.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, serverAddress, SERVER_PORT);

            clientSocket.send(sendPacket);
        }
        clientSocket.close();
    }
}
```

Output:

```
Welcome to the voting system.
Enter the name of your candidate or 'exit' to stop voting.
devansh
```

```
Server is running on port 5000
devansh received a vote from 127.0.0.1
```

Conclusion: Hence a client server socket program was implemented using TCP and using UDP.