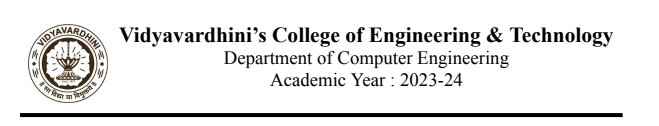
**Experiment No 6**

**Aim: Socket programming using TCP or UDP.**

Theory:

Sockets:

A socket is one endpoint of a two way communication link between two programs running on the network. The socket mechanism provides a means of inter-process communication (IPC) by establishing named contact points between which the communication takes place.

Socket are generally employed in client server applications. The server creates a socket, attaches it to a network port addresses then waits for the client to contact it. The client creates a socket and then attempts to connect to the server socket.

When the connection is established, transfer of data takes place.

Types of Sockets :

There are two primary and common types of Sockets: the datagram socket and the stream socket. 1.Datagram Socket :

This is a type of network which has connection less point for sending and receiving packets. It is similar to mailbox. The letters (data) posted into the box are collected and delivered (transmitted) to a letterbox (receiving socket).

2.Stream Socket:

In Computer operating system, a stream socket is type of interprocess communications socket or network socket which provides a connection-oriented, sequenced, and unique flow of data without record boundaries with well defined mechanisms for creating and destroying connections and for detecting errors. It is similar to phone.

A connection is established between the phones (two ends) and a conversation (transfer of data) takes place.

**Program:**

| **Client Side Program:**  import java.io.\*;  import java.net.\*;  import java.util.Scanner;  public class TCPChatClient {  private static final String SERVER\_ADDRESS = "localhost";  private static final int SERVER\_PORT = 12345;  public static void main(String[] args) {  try (Socket socket = new Socket(SERVER\_ADDRESS, SERVER\_PORT);  PrintWriter out = new PrintWriter(socket.getOutputStream(), true);  BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));  Scanner scanner = new Scanner(System.in)) {  new Thread(() -> {  String message;  try {  while ((message = in.readLine()) != null) {  System.out.println("Server: " + message);  }  } catch (IOException e) {  e.printStackTrace();  }  }).start();  String userInput;  while (true) {  userInput = scanner.nextLine();  out.println(userInput);  }  } catch (IOException e) {  e.printStackTrace();  }  }  }  **Server Side Program:**  import java.io.\*;  import java.net.\*;  import java.util.\*;  public class TCPChatServer {  private static final int PORT = 12345;  private static Set<PrintWriter> clientWriters = new HashSet<>();  public static void main(String[] args) {  System.out.println("TCP Chat Server started...");  try (ServerSocket serverSocket = new ServerSocket(PORT)) {  while (true) {  Socket clientSocket = serverSocket.accept();  new ClientHandler(clientSocket).start();  }  } catch (IOException e) {  e.printStackTrace();  }  }  private static class ClientHandler extends Thread {  private Socket socket;  private PrintWriter out;  private BufferedReader in;    public ClientHandler(Socket socket) {  this.socket = socket;  }  @Override  public void run() {  try {  in = new BufferedReader(new InputStreamReader(socket.getInputStream()));  out = new PrintWriter(socket.getOutputStream(), true);    synchronized (clientWriters) {  clientWriters.add(out);  }  String message;  while ((message = in.readLine()) != null) {  System.out.println("Received: " + message);  synchronized (clientWriters) {  for (PrintWriter writer : clientWriters) {  writer.println(message);  }  }  }  } catch (IOException e) {  e.printStackTrace();  } finally {  try {  socket.close();  } catch (IOException e) {  e.printStackTrace();  }  synchronized (clientWriters) {  clientWriters.remove(out);  }  }  }  }  } |
| --- |

**Output:**

| **Server Side:** |
| --- |
| **Client Side:** |

**Conclusion:**

TCP socket programming enabled successful client-server communication with real-time data transfer. The experiment demonstrated reliable, connection-oriented communication, making it suitable for applications requiring data integrity.