**LAB - 1**

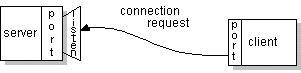
**SOCKET COMMUNICATION**

**Objective:**

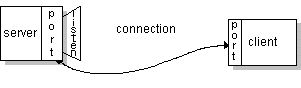
To establish communication between two Computers using “Sockets” in Computer Networks.

**Introduction:**

A **socket** is one endpoint of a two-way communication link between two programs running on the **network**. An endpoint is a combination of an IP address and a port number. Every **TCP** connection can be uniquely identified by its two endpoints. That way you can have multiple connections between your host and the server.



Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while the client reaches out to the server.

****

In order to make one computer to talk other, we need to write two programs each on one machine. Generally, one system acts as a server providing a service and other system acts a client that needed to be served by the server.

**Solution Concept:**

In order to make one computer to talk other, we need to write two programs each on one machine. Generally, one system acts as a server providing a service and other system acts a client that needed to be served by the server.

Since there needs to be communication, the most effective way is to use different threads for both reading data and writing data to sockets. The main thread focuses on reading data from the user and writing into the socket. While parallely the Listener thread reads the data on the socket and prints on the console. Hence the read/write will not depend on write/read to finish into the socket.

The data sent over the socket is called as the payload. A payload consists of two parts, namely the IP address which is sending the data and obviously the message of the data. This helps to identify which computer is sending the message over the socket in the case when multiple connections are made to the server.

You can always find the source code of this work [here](https://github.com/mohith7548/SocketProgramming/tree/master/src/PayloadServerClient).

**Networking Calls:**

1. ServerSocket(port):

This function call creates a new (server) socket on a port number and acts as Server, providing service. A server socket waits for requests to come in over the network. It performs some operation based on that request, and then possibly returns a result to the request.

1. ServerSocket.accept():

This function call accepts the client computer that is trying to connect to the server computer with the port number. The method blocks until a connection is made. It keeps on waiting for at least one client to connect.

1. Socket(hostName, port):

This function call helps the client computer to connect to the server that started service by taking its hostname (IP address) and the port number on which the service is started by the server.

**Program:**

1. **Server side:**

|  |
| --- |
| **package PayloadServerClient;  import java.io.\*; import java.net.ServerSocket; import java.net.Socket;  public class MyServer {  private static ServerSocket server;  private static Socket socket;  private static BufferedReader br;  private static ObjectOutputStream clientOutputStream;  private static ObjectInputStream clientInputStream;   private MyServer(int port) {  System.out.println("Starting server...");  try {  server = new ServerSocket(port);  System.out.println("Waiting for Client...");   socket = server.accept(); // Waits till the client connects.  System.out.println("Client connected: " + socket.getLocalAddress());   br = new BufferedReader(new InputStreamReader(System.in)); // To read from keyboard  clientOutputStream = new ObjectOutputStream(socket.getOutputStream()); // To write into Socket  clientInputStream = new ObjectInputStream(socket.getInputStream()); // To read from socket   } catch (Exception e) {  e.printStackTrace();  }   }   public static void main(String[] args) throws IOException {  new MyServer(3000);  String wBuffer;  // This listener keeps listening for the messages in the socket and prints them.  new Thread(new Listener(clientInputStream)).start();  System.out.println("Good to go now. Type your message and hit return.");  while (true) {  if(!(wBuffer = br.readLine()).equals("")) {  Payload payload = new Payload(socket.getLocalAddress().toString(), wBuffer);  clientOutputStream.writeObject(payload);  }  }  } }** |

1. **Client side:**

|  |
| --- |
| **package PayloadServerClient;  import java.io.\*; import java.net.Socket;  public class MyClient {  private static Socket socket;  private static BufferedReader br;  private static ObjectOutputStream clientOutputStream;  private static ObjectInputStream clientInputStream;   private MyClient(String ip, int port) {  try {  System.out.println("Trying to connect....");  socket = new Socket(ip, port);  System.out.println("Connected to Server.");   br = new BufferedReader(new InputStreamReader(System.in)); // To read from keyboard  clientOutputStream = new ObjectOutputStream(socket.getOutputStream()); // To write into Socket  clientInputStream = new ObjectInputStream(socket.getInputStream()); // To read from socket    } catch (Exception e) {  e.printStackTrace();  }  }   public static void main(String[] args) throws IOException, ClassNotFoundException {  new MyClient("localhost", 3000);  String wBuffer;  // This listener keeps listening for the messages in the socket and prints them.  new Thread(new Listener(clientInputStream)).start();  System.out.println("Good to go now. Type your message and hit return.");  while (true) {  if(!(wBuffer = br.readLine()).equals("")) {  Payload payload = new Payload(socket.getLocalAddress().toString(), wBuffer);  clientOutputStream.writeObject(payload);  }  }  } }** |

1. **Listener Class:**

|  |
| --- |
| **package PayloadServerClient; import java.io.ObjectInputStream;  class Listener extends Thread {  private static ObjectInputStream socketReader;   public Listener(ObjectInputStream socketReader) {  this.socketReader = socketReader;  }   public void run() {  try {  Payload payload;  while ((payload = (Payload) socketReader.readObject()) != null) {  System.out.println(payload.getIp() + ": " + payload.getMessage());  }  } catch (Exception e) {  e.printStackTrace();  }  }   public static void main(String args[]) {  Listener obj = new Listener(socketReader);  obj.start();  } }** |

1. **Payload Class:**

|  |
| --- |
| **package PayloadServerClient;  import java.io.Serializable;  public class Payload implements Serializable {  private String ip;  private String message;   Payload(String ip, String msg) {  this.ip = ip;  this.message = msg;  }   public String getIp() {  return ip;  }   public void setIp(String ip) {  this.ip = ip;  }   public String getMessage() {  return message;  }   public void setMessage(String message) {  this.message = message;  } }** |

Signature