Don Bosco Institute of Technology

Department of Information Technology

Class: SE

Semester 4

Name: Shubham Talawadekar

Roll No: 58

AIM: Understanding client-server socket and writing a small program for chat application

Theory:

In client-server applications, the server provides some service, such as processing database queries or sending out current stock prices. The client uses the service provided by the server, either displaying database query results to the user or making stock purchase recommendations to an investor.

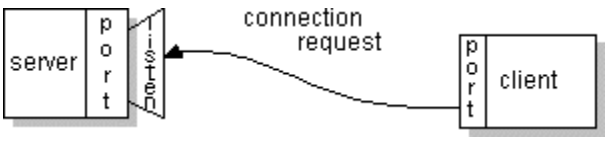
The communication that occurs between the client and the server must be reliable. That is, no data can be dropped and it must arrive on the client side in the same order in which the server sent it.

TCP provides a reliable, point-to-point communication channel that client-server applications on the Internet use to communicate with each other. To communicate over TCP, a client program and a server program establish a connection to one another. Each program binds a socket to its end of the connection. To communicate, the client and the server each reads from and writes to the socket bound to the connection.

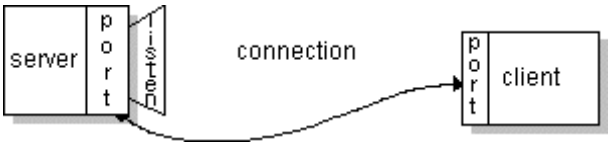
What Is a Socket?

Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

On the client-side: The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to connect with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.

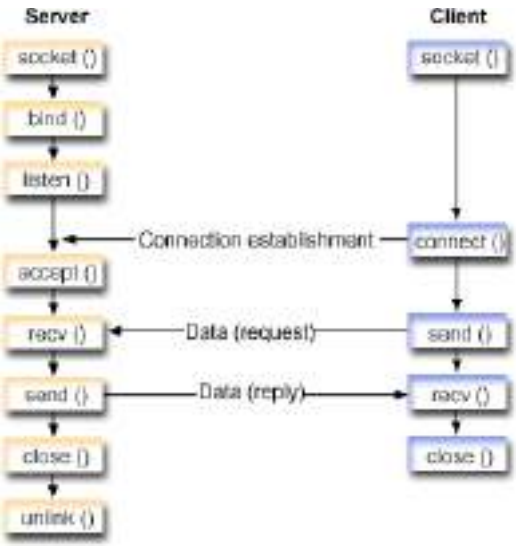


If everything goes well, the server accepts the connection. Once accepted, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connectionrequests while tending to the needs of the connected client.



On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server.

The client and server can now communicate by writing to or reading from their sockets.



Definition:

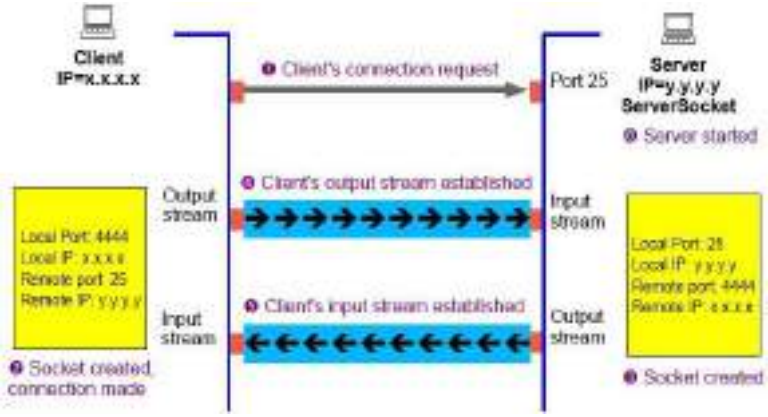
A *socket* is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent.

Endpoint = IP address + port number

Every TCP connection can be uniquely identified by its two endpoints.

The java.net package in the Java platform provides a class, java.net.Socket, that implements one side of a two-way connection between your Java program and another program on the network.

java.netincludes the ServerSocketclass, which implements a socket that servers can use to listen for and accept connections to clients.



CODE:

CLIENT SIDE:

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

class TCPClient
{
public static void main(String argv[]) throws Exception
{

  String FromServer;
  String ToServer;

   Socket clientSocket = new Socket("localhost", 5000);
          /* socket(sSocketString host, int port)  */
  BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
          /*An InputStreamReader is a bridge from byte streams
to character streams:                 It reads bytes and
decodes them into characters using a specified charset.*/
  PrintWriter outToServer = new
PrintWriter(clientSocket.getOutputStream(),true);

          /* public PrintWriter(OutputStream out, boolean
autoFlush)
              Parameters:
                out - An output stream
                autoFlush - A boolean; if true, the println()
```

SERVER SIDE:

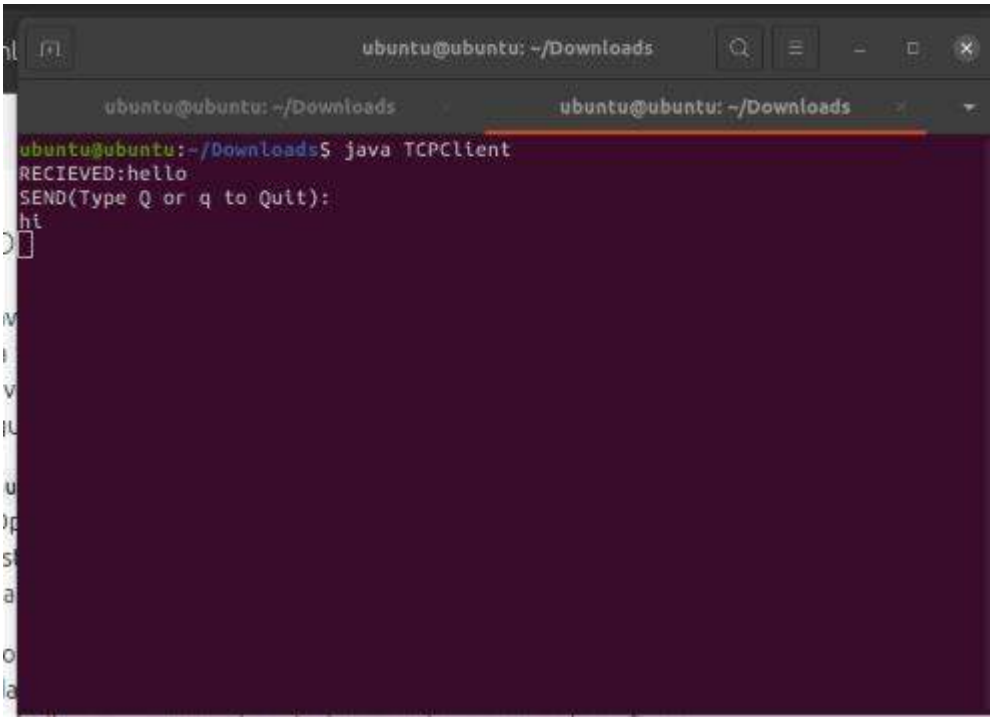
TCPServer.JAVA - WordPad

View

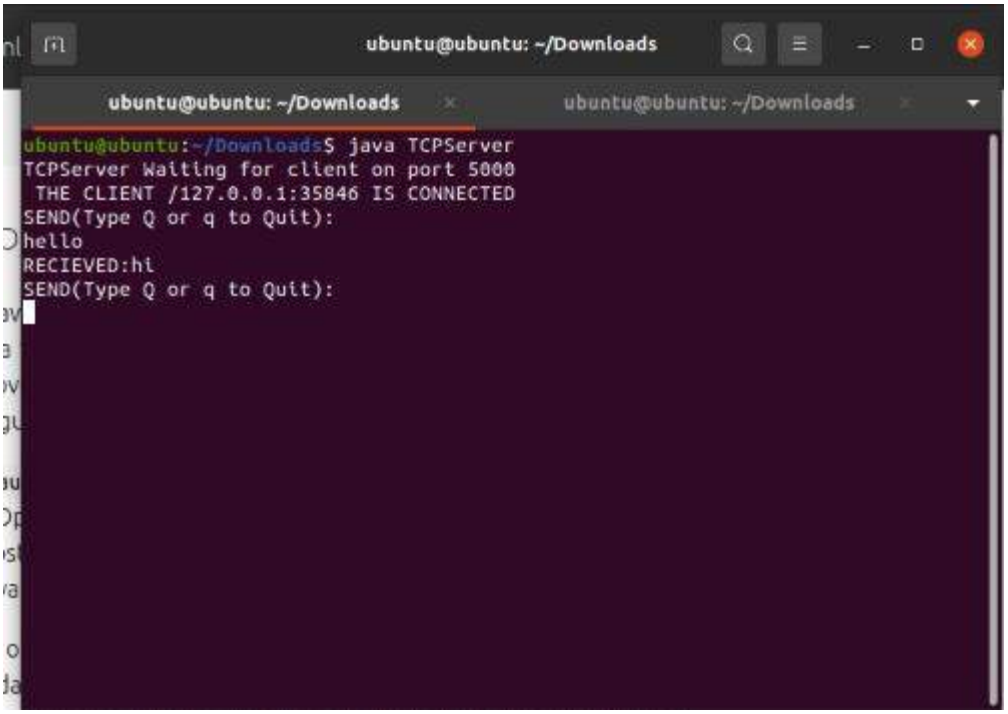
```
import java.io.*;
import java.net.*;

class TCPServer
{
  public static void main(String argv[]) throws Exception
      {
        String fromclient;
        String toclient;

        ServerSocket Server = new ServerSocket (5000);
        /* ServerSocket (Port Number) */
        System.out.println ("TCPServer Waiting for client on
port 5000");

        while(true)
        {
        Socket connected = Server.accept();
           //Server recieves request and get connected to
client socket.

           System.out.println( " THE CLIENT"+" "+
connected.getInetAddress() +":"+connected.getPort()+" IS
CONNECTED ");

           /* getInetAddress() : The remote IP address to which
this socket is connected, or null if the socket is not
connected.
           getPort() : get the port no to which the socket is
connected... */
```

Conclusion:

CLIENT SIDE:



SERVER SIDE:


ubuntu@ubuntu: ~/Downloads
ubuntu@ubuntu: ~/Downloads
ubuntu@ubuntu: ~/Downloads
ubuntu@ubuntu:~/Downloads$ java TCPServer
TCPServer Waiting for client on port 5000
THE CLIENT /127.0.0.1:35846 IS CONNECTED
SEND(Type Q or q to Quit):
hello
RECIEVED:hi
SEND(Type Q or q to Quit):