

Assignment 9

Problem Definition:

Problem Definition: Write a program using TCP socket for wired network for following a. Say Hello to Each other b. File transfer

1. Prerequisite:

- a) Socket Header
- b) Network Programming
- c) Ports

Learning Objectives:

1. To understand Work of Socket
2. Different methods associated with Client & Server Socket

New Concepts:

1. Client Server Communication
2. Port Address
3. Theory:

Introduction

TCP:

The Transmission Control Protocol provides a communication service at an intermediate level between an application program and the Internet Protocol. It provides host-to-host connectivity at the Transport Layer of the Internet model.

The client server model

Most inter-process communication uses the client server model. These terms refer to the two processes which will be communicating with each other. One of the two processes, the client, connects to the other process, the server, typically to make a request for information. A socket is one end of an inter- process communication channel. The two processes each establish their own socket.

The steps involved in establishing a socket on the client side are as follows:

1

. Create a socket with the socket() system call

ADYPSOE, Lohegaon Pune Computer Network 2019 pattern

2

. Connect the socket to the address of the server using the connect() system call

3

. Send and receive data. There are a number of ways to do this, but the simplest is to use the read()

and write() system calls.

The steps involved in establishing a socket on the server side are as follows:

1. Create a socket with the socket() system call

2. Bind the socket to an address using the bind() system call. For a server socket on the Internet, an address consists of a port number on the host machine.

3

4. Listen for connections with the listen() system call.

. Accept a connection with the accept() system call. This call typically blocks until a client connects

with the server.

5. Send and receive data

Algorithm: Server Program

1. Open the Server Socket: ServerSocket server = new ServerSocket(PORT);
2. Wait for the Client Request: Socket client = server.accept();
- 3
 - . Create I/O streams for communicating to the client
DataInputStream is = new DataInputStream(client.getInputStream()); DataOutputStream os = new DataOutputStream(client.getOutputStream());
- 4
 - . Perform communication with client Receive from client: String line = is.readLine();
- 5
 - . Send to client: os.writeBytes("Hello\n")
6. Close socket: client.close();

Algorithm: Client Program

1. Create a Socket Object: Socket client = new Socket(server, port_id);
2. Create I/O streams for communicating with the server. is = new DataInputStream(client.getInputStream()); os = new DataOutputStream(client.getOutputStream());
3. Perform I/O or communication with the server: Receive data from the server: String line = is.readLine(); Send data to the server: os.writeBytes("Hello\n");
4. Close the socket when done client.close();

TYPES OF SOCKETS

Socket Types:

There are four types of sockets available to the users. The first two are most commonly used and the last two are rarely used.

Processes are presumed to communicate only between sockets of the same type but there is no restriction that prevents communication between sockets of different types.

- Stream Sockets – Delivery in a networked environment is guaranteed. If you send through the stream socket three items "A, B, C", they will arrive in the same order – "A, B, C". These sockets use TCP (Transmission Control Protocol) for data transmission. If delivery is impossible, the sender receives an error indicator. Data records do not have any boundaries.
- Datagram Sockets – Delivery in a networked environment is not guaranteed. They're connectionless because you don't need to have an open connection as in Stream Sockets
 - you build a packet with the destination information and send it out. They use UDP (User Datagram Protocol).
- Raw Sockets – These provide users access to the underlying communication protocols, which support socket abstractions. These sockets are normally datagram oriented, though their exact characteristics are dependent on the interface provided by the protocol. Raw sockets are not intended for the general user; they have been provided mainly for those interested in developing new communication protocols, or for gaining access to some of the more cryptic facilities of an existing protocol.

Here is the description of the parameters –

- socket_family – This is either AF_UNIX or AF_INET, as explained earlier.
- socket_type – This is either SOCK_STREAM or SOCK_DGRAM.
- protocol – This is usually left out, defaulting to 0.

Once you have socket object, then you can use required functions to create your client or server program. Following is the list of functions required –

SERVER SOCKET METHODS

Sr.No.	Method & Description
1	s.bind() This method binds address (hostname, port number pair) to socket.
2	s.listen() This method sets up and start TCP listener.
3	s.accept() This passively accept TCP client connection, waiting until connection arrives (blocking).

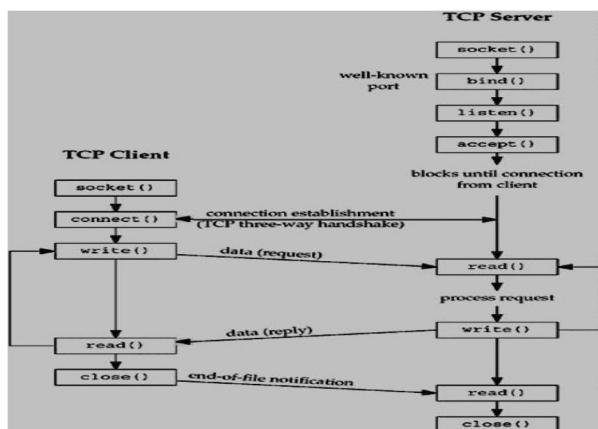
CLIENT SOCKET METHODS

Sr.No.	Method & Description
1	s.connect() This method actively initiates TCP server connection.

GENERAL SOCKET METHODS

Sr.No.	Metho d & Descri ption
1	s.recv() This method receives TCP message
2	s.send() This method transmits TCP message
3	s.recvfrom() This method receives UDP message
4	s.sendto() This method transmits UDP message
5	s.close() This method closes socket
6	socket.gethostname() Returns the hostname.

Methods Associated with Socket:The following diagram shows the complete Client and Server interaction



CONCLUSION: Thus we have successfully implemented the socket programming for TCP

The screenshot shows an IDE interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Standard icons for file operations.
- Search Bar:** CNS Practical.
- Explorer:** Shows a project structure under "CNS PRACTICAL" with files like client.java, Calc_Client.class, Calc_Server.class, ClientUDP.class, ClientUDP.java, DNS.java, DNS.png, Dvr.cpp, Exp7.cpp, Exp7.png, LRU.java, Optimal.java, server.class, server.java, ServerUDP.class, ServerUDP.java, Subnet.java, SubnetDemo.class, SubnetDemo.java, and SubnettingDemo.class.
- Code Editor:** Displays the content of client.java. The code implements a client for a UDP-based DNS server. It imports java.net.* and java.io.*. The main method creates a socket to localhost port 3333, reads input from the server, and writes responses back. A sample interaction is shown in the terminal.
- Terminal:** Shows command-line interactions:
 - PS C:\Users\harsh\Downloads\CNS Practical> javac server.java
 - PS C:\Users\harsh\Downloads\CNS Practical> java server
 - client says: hi
- Bottom Status Bar:** Indicating indexing completed, Java Ready, Line 23, Col 7 (5 selected), Tab Size: 4, UTF-8, CRLF, Java, Go Live, and other status indicators.

