Lab Assignment on Unit V

Aim: Write a program using TCP socket for wired network for following
a. Say Hello to Each other (For all students)
b. File transfer (For all students)
c. Calculator (Arithmetic) (50% students)
d. Calculator (Trigonometry) (50% students)

Requirements: Fedora 20 with Pentium IV and above, 1 GB RAM, 120 G.B HDD, Monitor, Keyboard, Mouse , Modelio, Eclipse, CDT, Python interpreter, Pydev, J2SE, Wireshark Packet Analyzer Tool.

Theory:

a. Say Hello to Each Other

1**. TCP Socket Programming for wired network**

The two key classes from the java.net package used in creation of server and client programs are: ∑ ServerSocket ∑ Socket A server program creates a specifi c type of socket that is used to listen for client requests (server socket), In the case of a connection request, the program creates a new socket through which it will exchange data with the client using input and output streams. The socket abstraction is very similar to the fi le concept: developers have to open a socket, perform I/O, and close it.

A simple Server Program in Java

The steps for creating a simple server program are:
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(); Send to client: os.writeBytes(“Hello\n”);
5. Close socket: client.close();

A simple Client Program in Java

The steps for creating a simple client program are:
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();

2. Running Socket Programs

Compile both server and client programs and then deploy server program code on a machine

which is going to act as a server and client program, which is going to act as a client. If required, both client and server programs can run on the same machine. To illustrate execution of server and client programs, let us assume that a machine called mundroo.csse.unimelb.edu.au on which we want to run a server program as indicated below:

[raj@mundroo] java SimpleServer

The client program can run on any computer in the network (LAN, WAN, or Internet) as long as there is no fi rewall between them that blocks communication. Let us say we want to run our client program on a machine called gridbus.csse.unimelb.edu.au as follows:

[raj@gridbus] java SimpleClient

The client program is just establishing a connection with the server and then waits for a message. On receiving a response message, it prints the same to the console. The output in this case is: Hi there which is sent by the server program in response to a client connection request. It should be noted that once the server program execution is started, it is not possible for any other server program to run on the same port until the fi rst program which is successful using it is terminated. Port numbers are a mutually exclusive resource.

b. File Transfer

A TCP Client initiates the communication with a server which is waiting for the connection. TCP is connection oriented and UDP is connectionless, which means that UDP sockets do not need to be connected before being used. Another difference between TCP and UDP is that there is no guarantee that a message sent via a UDP socket will arrive at its destination, and messages can be delivered in a different order than they were sent.

A TCP listener is created and starts listening to the specified port. Again the buffer size is set to 1024 bytes. A TCP listener can pre check to see if there are any connections pending before calling the AcceptTcpClient method. It returns true if there are any pending connections.