Lab 1 April 2020
TCP Server and TCP Client using Socket Class

Objectives:

- To understand the importance of IP Address and Port# in network communication.
- To understand how TCP Server is different from other servers.
- To learn how to develop a TCP Server and TCP Client using Socket class.

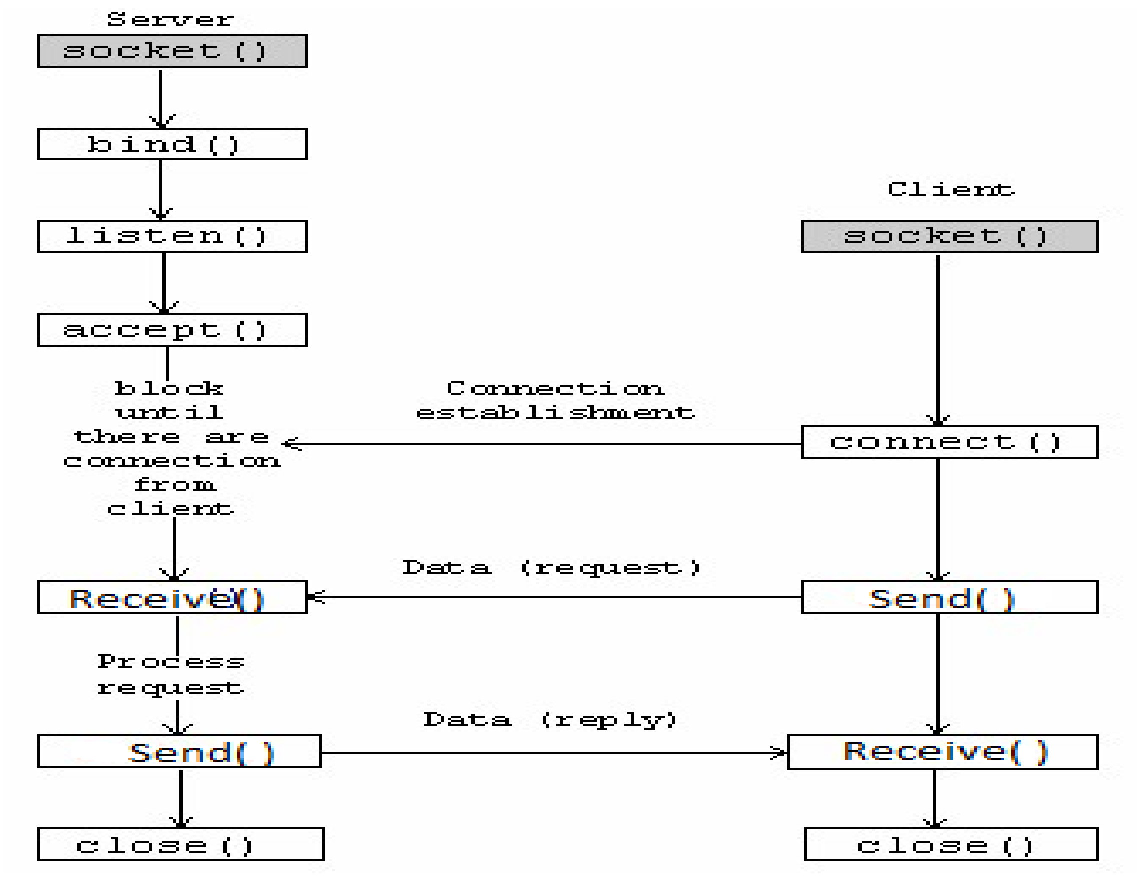
What to do first:

- Create Folder Lab11 inside Zfolder/SWE344.
- Create Folder TCPServer inside Zfolder/SWE344/Lab11
- Create Folder TCPClient inside Zfolder/SWE344/Lab11

Echo Server:

Echo server is a type of server that when receives a data packet; it returns the data packet back to the client.

TCP Client/Server Model



Steps for Iterative TCP Server:

Since **TCP** is connectionless, hence the Server must implement the following steps in sequence only.

- Server will create a socket.
- Server will bind the socket on specified local ***port#*** and local ***IP address*** or local ***IPEndPoint***.

- Once the socket has been created and bound to the local ***IPEndPoint***, the server will convert the same socket into listening socket.
- On the listening socket the server will wait for the connection requests from the client.
- The server will get the connected socket, on accepting the connection request from the client.
- Using the connected socket the server can exchange stream of bytes with the client.
- Since **TCP** is connection-oriented, hence to send data, the server will ***NOT*** specify the ***IPAddress*** and ***Port#*** of client.
- On completion of communication with the Client, the server should close the connected socket.
- At the end the server should also close the listening socket.

Steps for TCP Client:

The **TCP** Client must implement the following steps in sequence

- Client will create socket.
- Once the socket has been created, using the same socket the client will send connection request to server.
- While sending the connection request, the client will **MUST** specify the ***IPAddress*** & ***Port#*** of the server (on which the server is listening for connection request from client).
- Since ***TCP*** is connection-oriented, hence to send data, the client will not need to specify the **IP Address** and **Port#** of the server.
- On Completion of communication with the Server, the client must close Socket.

To Develop TCP Echo Server and TCP Client Using Socket Class:

Instructions:

- Make **TCP** Client & **TCP** Server applications with the following specifications. Make sure that your applications will follow the APIs as required.
- Your applications should be able to run with any other application having same specifications.
- Make your applications to support the asked specifications only.

TCP Server Specifications:

1. The server will only be able to handle single client.
2. The server will be **Echo** Server.
3. Before start up of server application, student is required to check the address of current machine.
4. The server will must bind on IP address got in step3 & port number 4400.
5. After accepting the connection request from the client, the server will send welcome message to client along with the ***IPAddress*** and ***Port#*** of the client i.e (**Well-Come : 192.168.0.4, 4200**), where 192.168.0.4 is the IP Address and 4200 is the port# of client.
6. After sending the well come message to client, the server will wait for the welcome message from the client.
7. Now the server will start running in a loop.
8. Inside loop the server will get block in a call of **Receive( ).** Whatever message it will receive from the client, it will display on the screen and the same message it will send back to the client using **Send( )**.
9. After sending message to the client, the server will start waiting to receive message from the client again by going back to step 8 or should close the connected socket if the client will stop sending data by closing connection.
10. After closing the connected socket, the server should close listening socket also.

<u>TCP Client Specifications:</u>

1. After getting connection with the server, the client will receive well come message from the server and will display the message on the screen.
2. In response of the welcome message from server, the client will send welcome message to the server along with the ***IPAddress*** and ***Port#*** of Server.
3. Now the client will start running in a loop.
4. Inside loop every time it will take input from the user. If the user input will be "stop" the client will exit.
5. If the user will type any message the client application will send it to the server using **Send**( ), and will block in **Receive( )** .
6. After receiving message from the server, the client will display the received message on the screen and will go back to step 4 or should close the socket if the server closes the connection.