PSEUDO-CODES

A) TCP/IP Client Side

1) Create a stream socket using socket() call.
2) ~~Bind~~ Connect socket to a foreign host or a server using the connect() system call.
3) Write a message to a server.
4) Upon sending a message to the server, receive an acknowledgement message from the server.
5) Read the other messages sent by the server.
6) Go to step 3 until all the information has been exchanged.
7) Close the socket and end the TCP/IP session.

B) TCP/IP Server side

1) Identify the mode of operation ~~usin~~ by accessing argv[2].
   1) One to One Mode
   2) Broadcast Mode
2) Bind the socket to a local address using bind().
3) Wait for incoming connections using listen().
4) Accept the connection using accept() and store the socket descriptor in an array & create a thread to handle that client.

→

5) When the client sends a message to the server, it'll be handled by the thread corresponding to that client.

a) For one to one mode, first an acknowledgement message to be sent to the client which ~~has~~ sent the message and then send the message ~~to~~ received by client i to client $(i\%k+1)$, k is total no. of clients. For client i, look for the $((i\%k+1)-1)$ index in array and send it the message which was received by client i.

b) For broadcast mode, first an acknowledgement message to be sent to the client i which sent a message and then send the message received by client i to all the other active k-1 clients.

c) While writing a message to the client, use mutex to maintain data integrity and also use pthread_join for the ~~concurr~~ concurrency of threads.

~~6) Go to step A until all info has been exchanged.~~

6) Go to step A and accept new connections.

7) If a particular client is to be ended, terminate the corresponding thread and free the socket associated.

A) UDP Client Side

1) Create an UDP Socket.
2) Send a test message to validate a connection with a foreign host.
3) Create a thread which will continously be receiving the messages sent to it by the server and printing them.
4) The main thread will be responsible to send messages from Client to server using SendTo(), and to receive the acknowledgement message from the server and print it in the console.
5) Continue this untill all the information has been exchanged, then terminate thread, close socket descriptor and exit.

B) UDP Server Side

1) Identify the mode of operation by accessing argv[2].
   1) One to One mode
   2) Broadcast mode
2) Create an UDP Socket.
3) Bind the socket to local address using bind().
4) Wait for the test validate message to arrive. On receiving it, store the Sin-port value of the client in an array. Every client has a unique Sin-port value.

5) If the received message is not a test message, then

~~i) For one to one mode~~

Send an acknowledgement message to the client irrespective of the mode i.e. One to one or broadcast.

i) For one to one mode, obtain the sin_port value of the client using int no. = ntohs(cliaddr.sin_port) and compare this value (i.e. no.) with the ~~value~~ sin_port values in array. If matched then send the message which was received by this client to the client with sin_port value at index (i%k+1)-1 where client i ~~sent~~ sent the message to the server.

ii) For broadcast mode, send the received message to all other clients by accessing their sin_port values in the array.

6) Go back to step 4 until all the information has been exchanged.

7) Free the socket descriptor and exit.