DX1222 Multiplayer Games Programming

Assignment 1: Chat Programme (Server & Client)

In this assignment you are to create a simple chat programme with server & client

Requirement:

1. Create chat programme using what you have learnt in Winsock (TCP based).
2. Use the I/O Multiplexing Server and I/O Multiplexing Client for your framework
    a. You can also use Multithreaded Server and Multithreaded Client

The following must be submitted

Your programme should include but not limited to the following

1. Accept and manage at least 3 multi-connection TCP/IP sockets
2. Implement the packet management codes
3. Welcome message upon joining the server
4. Broadcast chat message function
5. Whispering (Direct messaging to a single destination)
6. User List (List of connected socket descriptors)
7. Quit and close connection
8. Print out help menu
9. Documentation. Describe what you have, how it is implemented and how to use it (Commands). Split the writeup to the following 2 sections
    a. Feature List: What features are implemented and how are they implemented
    b. list of all the packet design
    c. User Guide: How to use these features
    d. Where are the codes for your features (file names, function names and, class names with line numbers)
    e. (Recommended) take the screenshots of how it works.

Refer to Annex A for more detailed writeup for each feature

Following features are considered advanced features for students who wish to achieve better grades. All the features in this "Advanced" category need to be networked features. And this means each of the features needs to work through network between the server and client. Also, each of the features needs an interactive (handshaking) between the server and client.

If the advanced feature is not a networked feature, it will not be counted in.

1. Mark the first entered session as a room master and give the privilege to add the password to enter the room. You can add more functions and features to this chatting room and room master.
2. Multiple rooms in chatting channel.
3. Friends list and other features for friends.
4. And more ideas!

Grading Criteria

1. Input
    - The Server and Client should not crash with any user input
2. Refer to Annex A for details on the basic compulsory features
3. Advanced Features

Due Date

1. Week 08 Sunday 2359: Submit
2. Week 09 (17 Dec): Present your work to the lecturer and tutor

Naming Conventions

1. Final Submission
    a. Admin_name_Assn1.zip

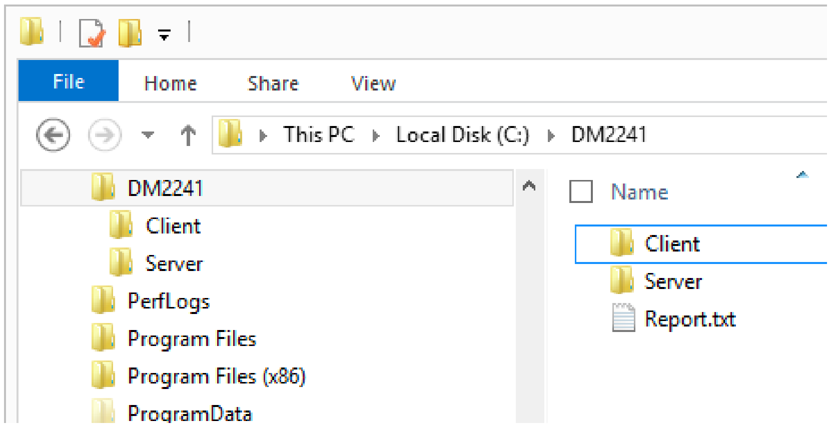
Submission Guidelines

1. Penalty will be given for late submission
2. No submission or failure to submit without a valid reason will constitute a zero to this assignment. This will affect your overall module grade
3. Avoid last-minute submission or uploading as there might be a surge in uploading traffic to BlackBoard (BB) e.g. do submit/upload to BB at least 60 mins before the deadline
4. Save multiple versions of your work in case of file corruption to the latest file. File corruption will not be a valid reason for failure to submit.
5. Always download your submission/upload from BB to double-check if your file/work has been uploaded correctly
6. Please note we take plagiarism seriously. Please refer to the below link on NYP's academic integrity.
   https://www.nyp.edu.sg/current-students/academic-matters/nyp-academic-integrity.html

Submission of Source Code

1. Submit in a zip file.
2. Use the standard naming convention as below;
    - For the server: make the folder name as "**Server**"

- For the client: make the folder name as "**Client**"
- For document: .txt or .pdf format only. Add to your **Top Folder** which has "**Server**" and "**Client**" as subfolders.



3. Zip the **Top Folder** which has "**Server**" and "**Client**" as subfolders and "**Document**" file. And put the name of the zip file as;
   "**Assignment02_<AdminNo>_<Name>.zip**"
   - Assignment02_123456Z_ TylerBlevins.zip
     - Uppercase for the Admin number checksum. e.g. 123456Z
     - Uppercase for the first letter of your name. e.g. TylerBlevins
   - No spaces in the filename.
   - *.h,*.cpp,*.sln,*.vcxproj,*.vcxproj.filters files are **ONLY** allowed.
   - No *.suo,*.sdf,*.exe,*.pdb,*.idb,*.ilk, etc.
   - No Debug, Release, ipch .vs folders.
4. Your submission zip file should be less than 10 Kbytes. If it is larger than that size, it means you have unnecessary files!
5. The zip file should contain the solution file at the root level.
6. Remember to test your submission to see if it can be opened by Visual Studio and build properly.

Annex A: Details of Basic Required Features

1. Multi-Connection

At least 3 concurrent multi-connection TCP/IP sockets

2. Implement the packet management codes.

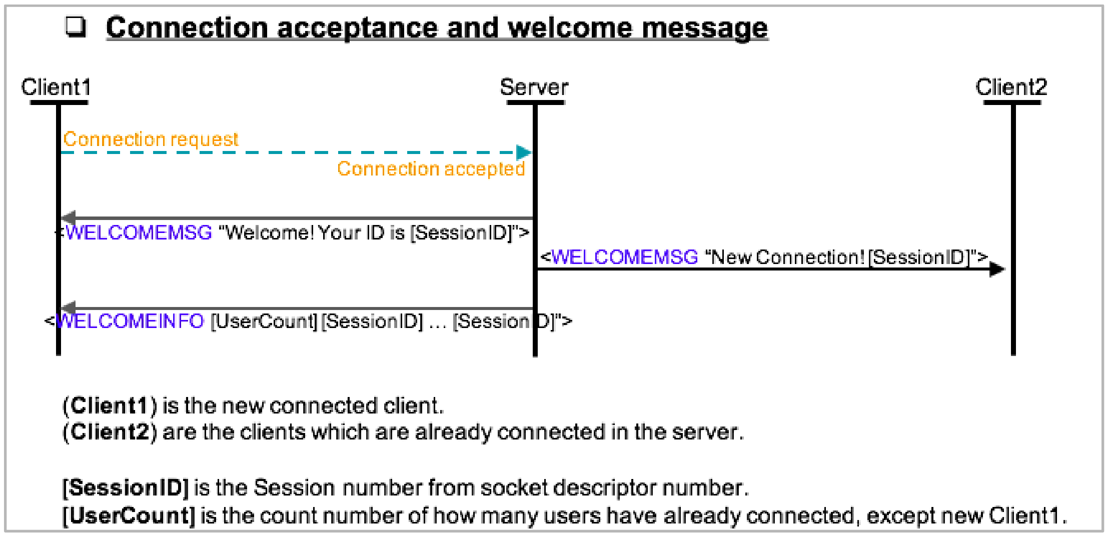
Implement the packet management codes both into Server and Client to make sure your message packet reaches, assembles, parses, and process correctly.

About packet design, you must use **Text Based Packet** and **'<'** for packet header and **'>'** for packet trailer.

3. Welcome Message

- Send a *Welcome Message* to the client immediately after the server accepted the new connection request from the client. And print out "Welcome! Your ID is ###" on your client console.
- If there are any of clients which already connected to the server, the server sends a *New Connection Message* (e.g. "New Connection! ID is ###") to the already connected clients. And send *Welcome Info Message* (e.g. "### is already connected") to the new-connected client.

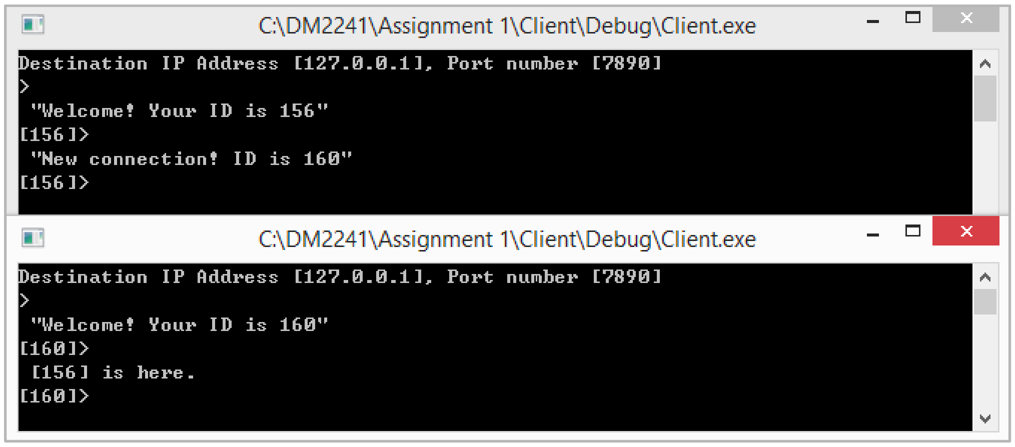
- Message flow example


❑ Connection acceptance and welcome message
Client1
Server
Client2
Connection request
Connection accepted
<WELCOMEMSG "Welcome! Your ID is [SessionID]">
<WELCOMEMSG "New Connection! [SessionID]">
<WELCOMEINFO [UserCount] [SessionID] … [SessionID]">

(**Client1**) is the new connected client.
(**Client2**) are the clients which are already connected in the server.

[SessionID] is the Session number from socket descriptor number.
[UserCount] is the count number of how many users have already connected, except new Client1.

- Example screenshot



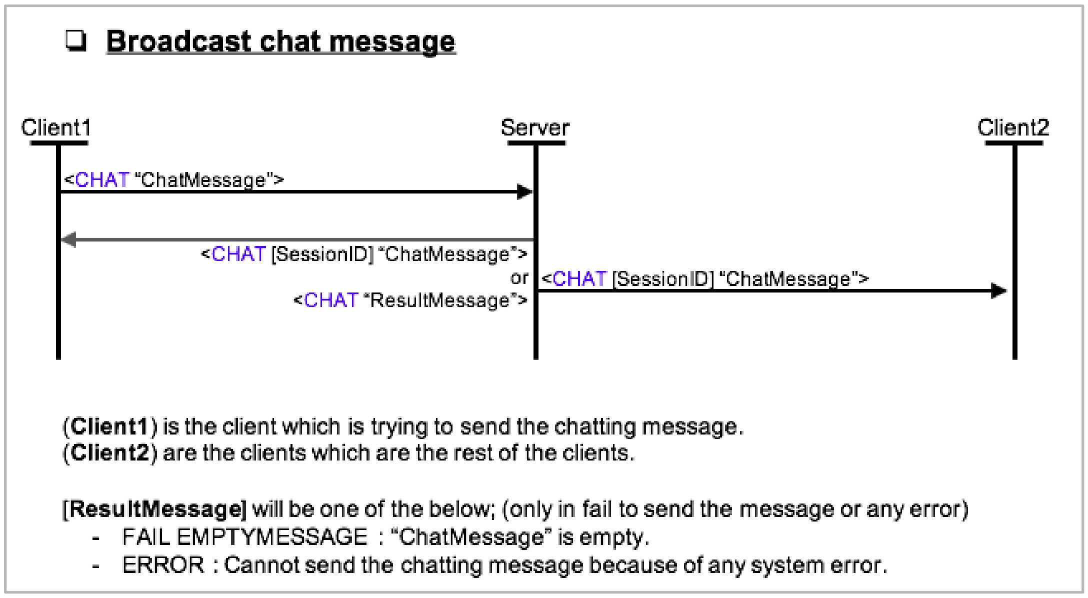
```
C:\DM2241\Assignment 1\Client\Debug\Client.exe
Destination IP Address [127.0.0.1], Port number [7890]
>
 "Welcome! Your ID is 156"
[156]>
 "New connection! ID is 160"
[156]>
```

```
C:\DM2241\Assignment 1\Client\Debug\Client.exe
Destination IP Address [127.0.0.1], Port number [7890]
>
 "Welcome! Your ID is 160"
[160]>
 [156] is here.
[160]>
```

4. Broadcast chat message

- If you type any text message and send to the server, the server will broadcast this chatting message to the all connected clients.

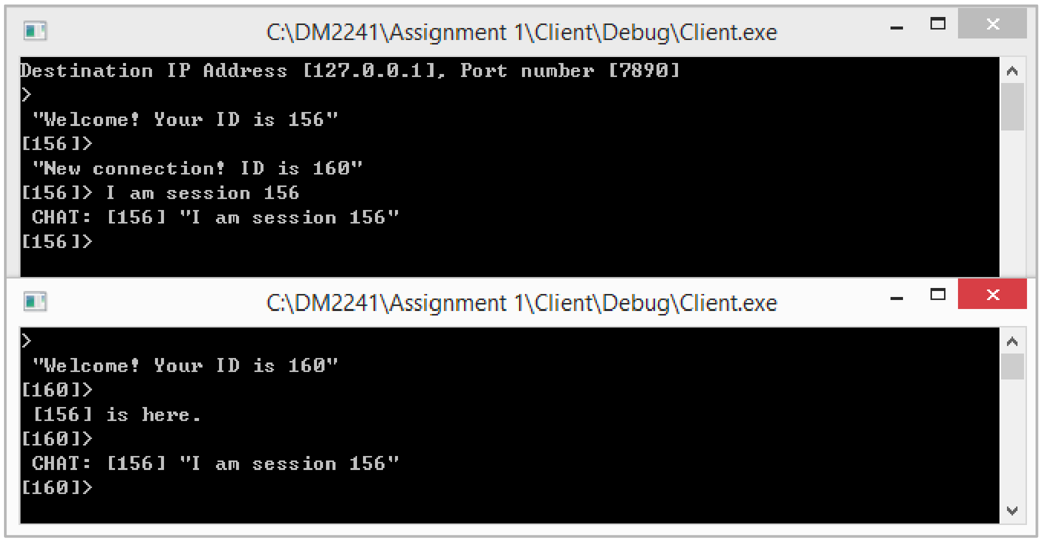
- Message flow example



(**Client1**) is the client which is trying to send the chatting message.
(**Client2**) are the clients which are the rest of the clients.

[**ResultMessage]** will be one of the below; (only in fail to send the message or any error)
- FAIL EMPTYMESSAGE : "ChatMessage" is empty.
- ERROR : Cannot send the chatting message because of any system error.

- Example screenshot



C:\DM2241\Assignment 1\Client\Debug\Client.exe

```
Destination IP Address [127.0.0.1], Port number [7890]
>
 "Welcome! Your ID is 156"
[156]>
 "New connection! ID is 160"
[156]> I am session 156
 CHAT: [156] "I am session 156"
[156]>
```

C:\DM2241\Assignment 1\Client\Debug\Client.exe

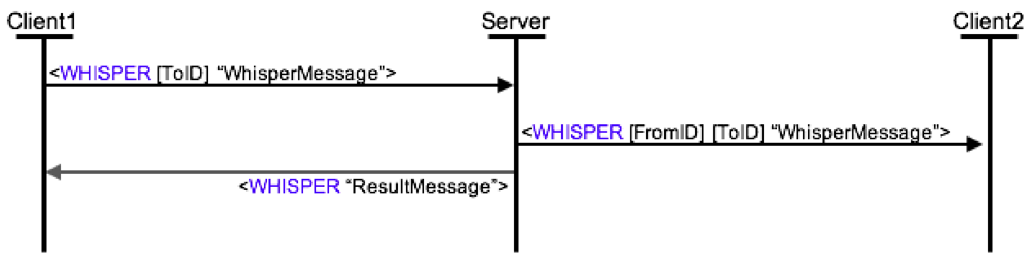
```
>
 "Welcome! Your ID is 160"
[160]>
 [156] is here.
[160]>
 CHAT: [156] "I am session 156"
[160]>
```

5. Whispering (Direct message to the single destination)

- Send the text message to the single destination.
- You can make a command such as; '/whisper SessionID TextMessage' or '/w SessionID TextMessage'.

- Message flow example

❑ Whisper message to one specific client

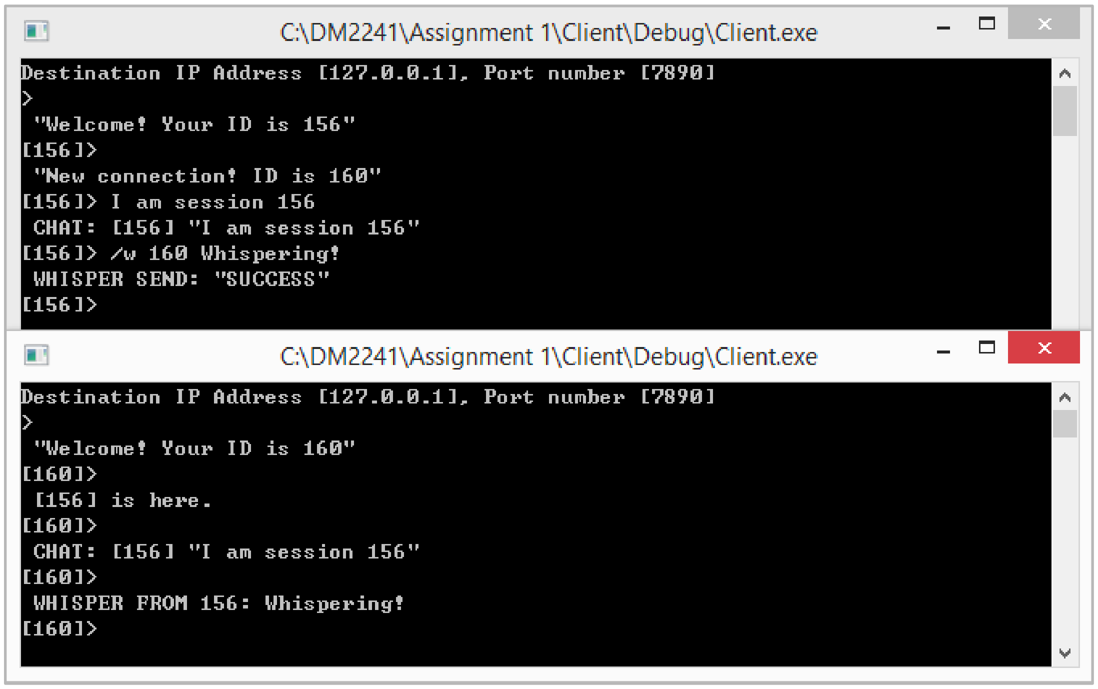

Client1
Server
Client2
<WHISPER [ToID] "WhisperMessage">
<WHISPER [FromID] [ToID] "WhisperMessage">
<WHISPER "ResultMessage">

(**Client1**) is the client which is trying to send the whisper message.
(**Client2**) is the destination client which is going to receive the whisper message.

[ToID] is the destination SessionID (client2) number which is going to receive the whisper message.
[FromID] is the source SessionID (client1) number which is sending the whisper message.

[ResultMessage] will be one of the below;
- SUCCESS : Successfully send the whisper message to the [ToID].
- FAIL INVALIDID : [ToID] is invalid number (0 or negative number) or not connected.
- ERROR : Cannot send the whisper message because of any system error.

- Example screenshot



C:\DM2241\Assignment 1\Client\Debug\Client.exe

```
Destination IP Address [127.0.0.1], Port number [7890]
>
 "Welcome! Your ID is 156"
[156]>
 "New connection! ID is 160"
[156]> I am session 156
 CHAT: [156] "I am session 156"
[156]> /w 160 Whispering!
 WHISPER SEND: "SUCCESS"
[156]>
```

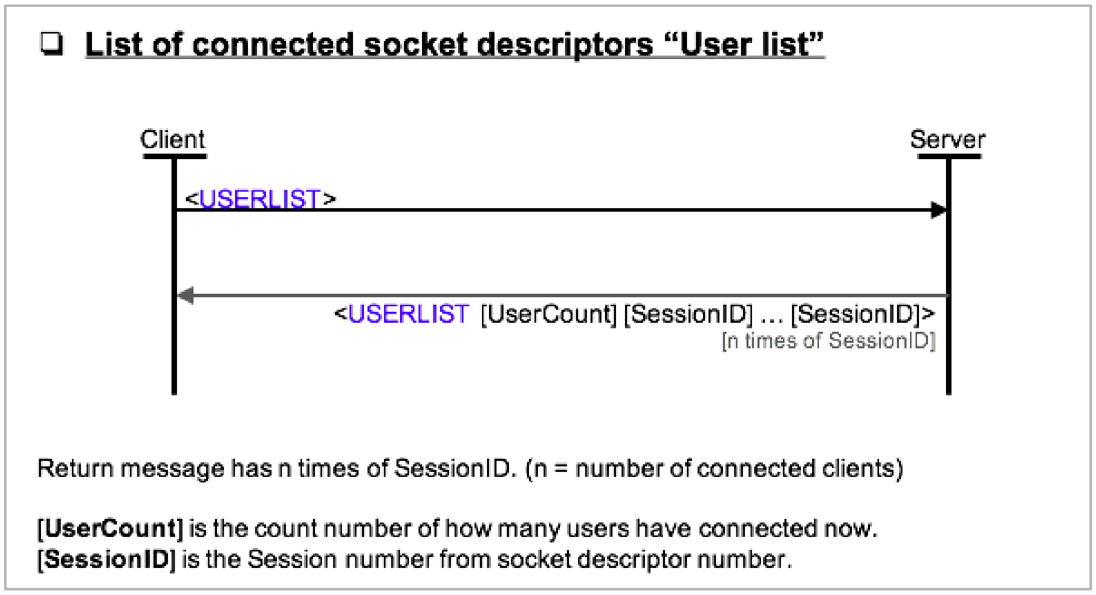
C:\DM2241\Assignment 1\Client\Debug\Client.exe

```
Destination IP Address [127.0.0.1], Port number [7890]
>
 "Welcome! Your ID is 160"
[160]>
 [156] is here.
[160]>
 CHAT: [156] "I am session 156"
[160]>
 WHISPER FROM 156: Whispering!
[160]>
```

6. List of connected socket descriptors (User list)

- Send the list of all connected SessionID.
- You can make a command such as; '/list' or '/l'.

- Message flow example

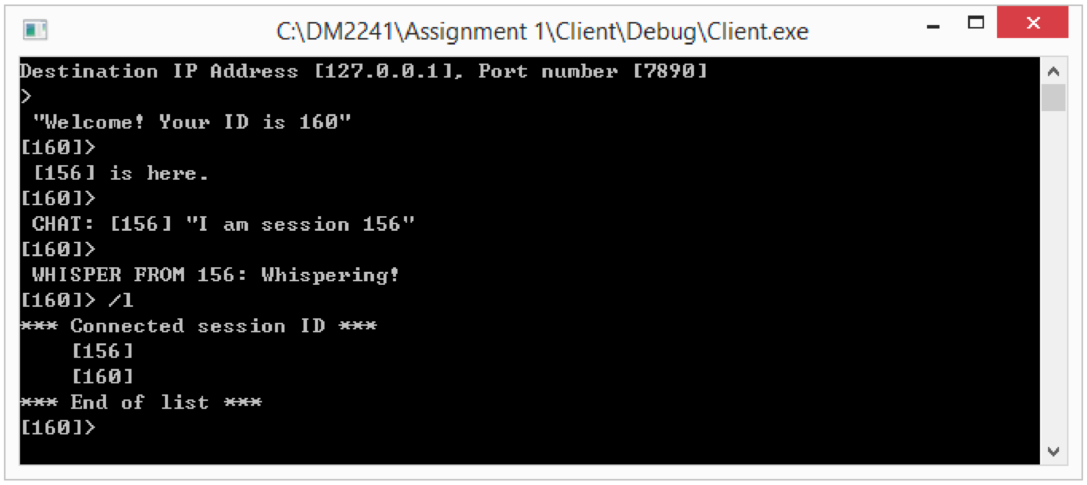


❑ **List of connected socket descriptors "User list"**

Return message has n times of SessionID. (n = number of connected clients)

[UserCount] is the count number of how many users have connected now.
[SessionID] is the Session number from socket descriptor number.

- Example screenshot



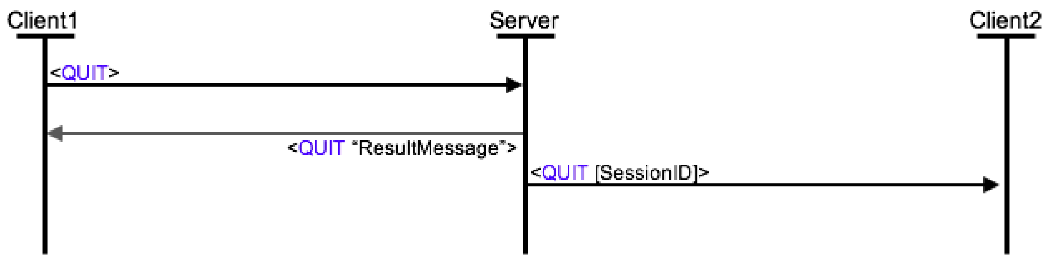
```
Destination IP Address [127.0.0.1], Port number [7890]
>
 "Welcome! Your ID is 160"
[160]>
 [156] is here.
[160]>
 CHAT: [156] "I am session 156"
[160]>
 WHISPER FROM 156: Whispering!
[160]> /l
*** Connected session ID ***
    [156]
    [160]
*** End of list ***
[160]>
```

7. Quit and close connection

- If the server receives Quit message, return back Quit message to the client and send the Quit message with client SessionID to the other clients.
- You can make a command such as; '/quit' or '/q'.

- Message flow example

❑ **Quit and close connection**



(**Client1**) is the client which is trying to quit from the server.
(**Client2**) are the other clients which are remaining in the same channel or lobby.

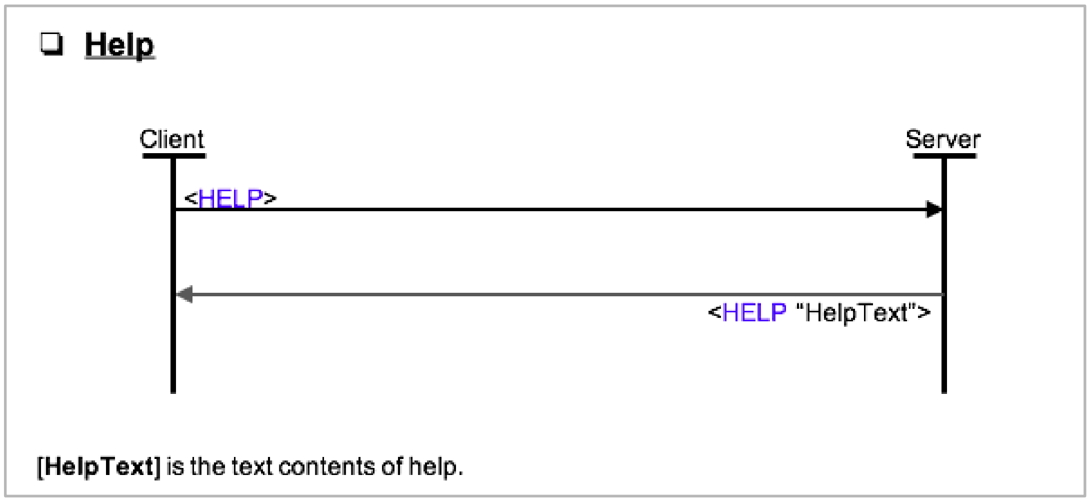
[SessionID] is the ID (client2) of the client which is going to quit the server.

[ResultMessage] will be one of the below;
- SUCCESS : Successfully quit the server.
- ERROR : Cannot quit and close the connection from the server because of any system error.

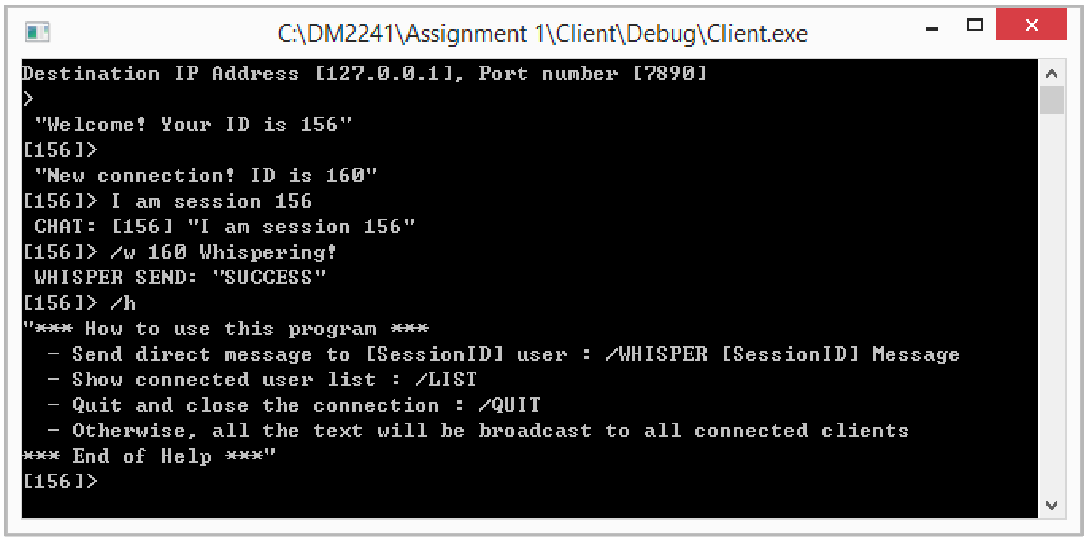
8. Print out help menu

- Print the list of commands what and how you can use with your program.
- You should receive the contents of help from the server. Do not print help from your client itself.
- You can make a command such as; '/help' or '/h'.
- You should not print out the help message direct from the client without receiving the contents from the server.

- Message flow example



- Example screenshot



```
Destination IP Address [127.0.0.1], Port number [7890]
>
 "Welcome! Your ID is 156"
[156]>
 "New connection! ID is 160"
[156]> I am session 156
 CHAT: [156] "I am session 156"
[156]> /w 160 Whispering!
 WHISPER SEND: "SUCCESS"
[156]> /h
"*** How to use this program ***
  - Send direct message to [SessionID] user : /WHISPER [SessionID] Message
  - Show connected user list : /LIST
  - Quit and close the connection : /QUIT
  - Otherwise, all the text will be broadcast to all connected clients
*** End of Help ***"
[156]>
```