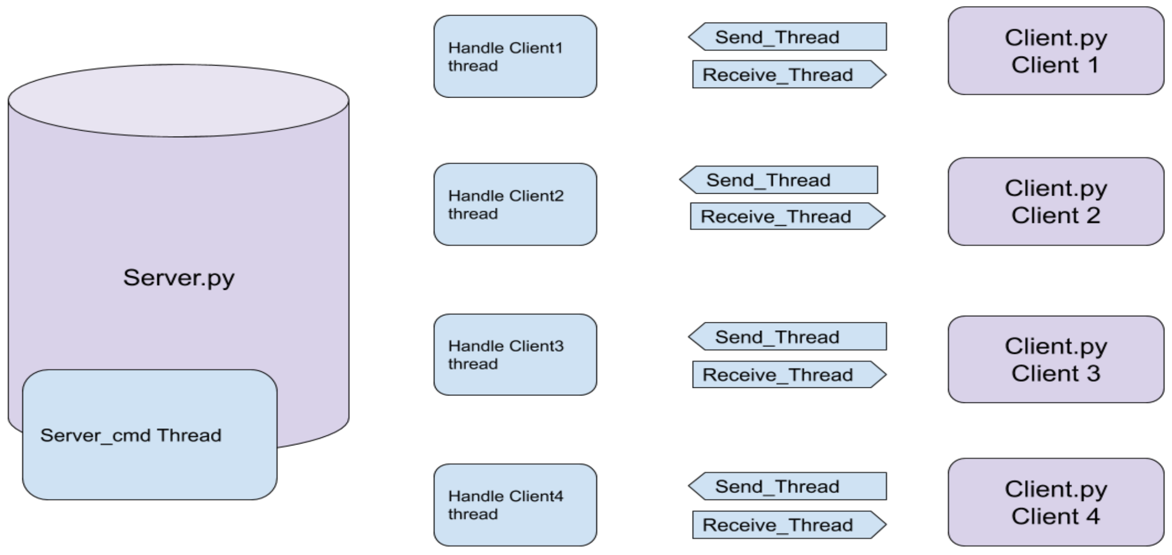
Mooyeong Lee
CSCI 3390: Intro to Network Science
April 22nd, 2020
Midterm III: Chat Room Assignment

Github Link: https://github.com/mylee1995/pythonChatRoom

The Chat Room Machine using Python consists of two python files; server.py and client.py. In these two files, socket, a low level networking interface, has been used to establish connections between the server and multiple clients. Thread is also used to run multiple operations simultaneously within the process.

After server.py opens up the socket in the port to wait clients, one can run the client.py to establish the connection between the server and the client. In order to maintain each communication to clients, the server opens up new threads after each connection. Each client handles two threads, send_thread and receive_thread to send messages and receive messages without hindering one another. “Figure 1” below depicts the logic diagram of the communication between the server and client



<Figure 1: The logic design of Chat Room>

Server.py

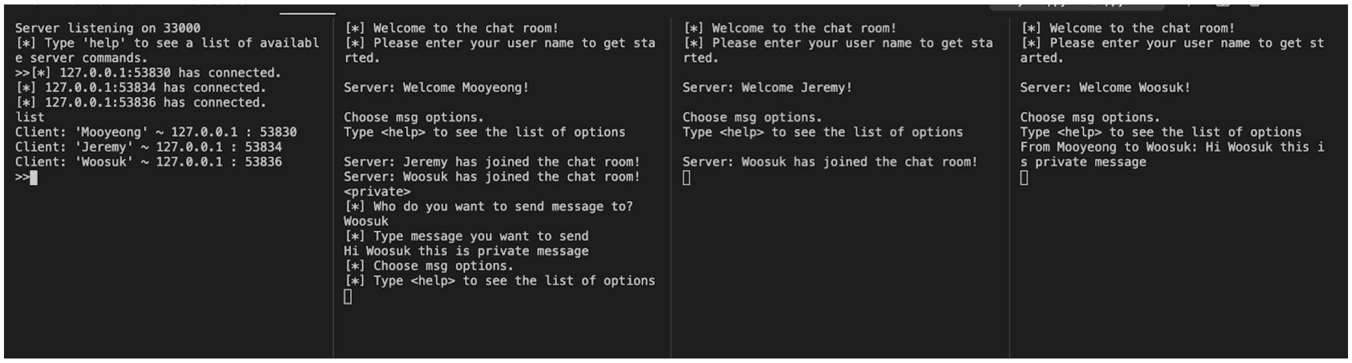
1. After server starts running, server maintainer can choose on of the following options for the chat room server
    - help: To see the list of server cmd options
    - list: To query a list of active clients
    - announce: Makes announcement from the server to all clients in the chatroom
    - quit: To close the server and disconnect all clients
    - clear: to clear the previous message in the terminal

Client.py

1. After connecting to the server, server sends the welcome message, enter to proceed
2. Type the desired user name
3. Type one of the following commands to choose options
    - <help>: Prints out the list of command options
    - <list>: Prints out the list of all active users in the chatroom
    - <private>: Allows to send private message to a specific user in the chat room
    - <announce>: Allows to send public message to all users in the chat room
    - <quit>: Quits the chatroom, disconnects from the server and exit
    - <clear>: to clear the previous message in the terminal

One-to-One communication

- A client can perform one-to-one communication via “<private>” option in client.py. After choosing “<private>” cmd option, the program asks the recipient username to send out the message to and its content After inputting the recipient and its content, the client file sends the dictionary object encoded to the server to indicate the private message. As a result, the server acknowledges
- Tutorial on One-to-One communication
    - Client.py
        - Run server.py from the terminal first: “python server.py”
        - Run client.py from another terminal: “python client.py”
        - Choose the username you desire
        - Type “<private>” to select private message option
        - Type the recipients user name
        - Type the content of the private message
        - If unsuccessful (recipient does not exist), the client will receive the error message
- Demonstration of one-to-one with 3 users (Mooyeong, Jeremy, Woosuk) and Mooyeong sends private message to Woosuk is on Figure 2 below



```
Server listening on 33000
[*] Type 'help' to see a list of availabl
e server commands.
>>[*] 127.0.0.1:53830 has connected.
[*] 127.0.0.1:53834 has connected.
[*] 127.0.0.1:53836 has connected.
list
Client: 'Mooyeong' ~ 127.0.0.1 : 53830
Client: 'Jeremy' ~ 127.0.0.1 : 53834
Client: 'Woosuk' ~ 127.0.0.1 : 53836
>>
```

```
[*] Welcome to the chat room!
[*] Please enter your user name to get sta
rted.

Server: Welcome Mooyeong!

Choose msg options.
Type <help> to see the list of options

Server: Jeremy has joined the chat room!
Server: Woosuk has joined the chat room!
<private>
[*] Who do you want to send message to?
Woosuk
[*] Type message you want to send
Hi Woosuk this is private message
[*] Choose msg options.
[*] Type <help> to see the list of options
```

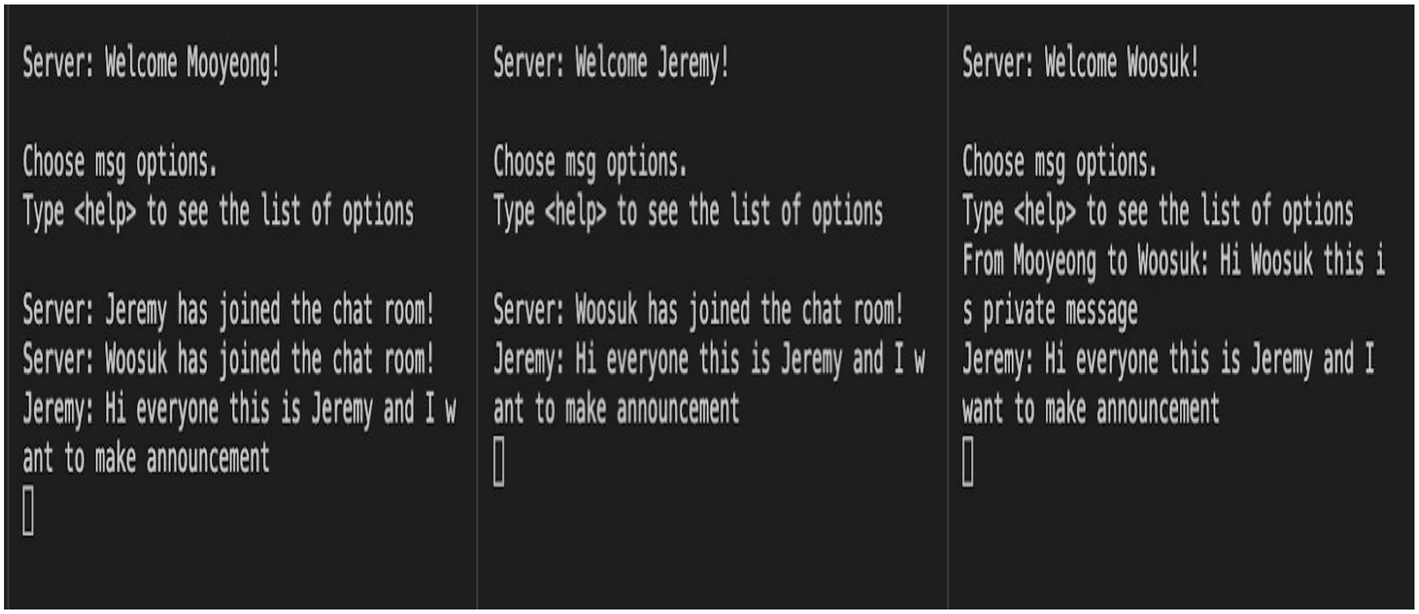
```
[*] Welcome to the chat room!
[*] Please enter your user name to get sta
rted.

Server: Welcome Jeremy!

Choose msg options.
Type <help> to see the list of options

Server: Woosuk has joined the chat room!
```

```
[*] Welcome to the chat room!
[*] Please enter your user name to get st
arted.

Server: Welcome Woosuk!

Choose msg options.
Type <help> to see the list of options
From Mooyeong to Woosuk: Hi Woosuk this i
s private message
```

<Figure 2: A private message demonstration from Mooyeong to Woosuk>

One-to-All communication

- A client can perform one-to-all communication via “<announce>” option in client.py. After choosing the “<announce>” cmd option, the program asks the content of the announcement and sends it to the server. The server decodes the message and after it acknowledges message type to be announcement, it loops through all connections in USERS variable to send out the announcement to all users
- Similarly, a server can also perform one-to-all communication from server.py. By choosing announce option in server command, the server can send announce ment to all users by iterating through USERS variable which stores all the established connections in the chatroom
- Tutorial on One-to-All communication
    - Server.py
        - Start the server by typing “python server.py” in the terminal
        - Open multiple client.py to create active users
        - Choose “announce” in server.py terminal
        - Type content of the announcement you want to make
        - See Figure 3 to see the screenshot of the tutorial
    - Client.py
        - Start the server by typing “python server.py” in the terminal
        - Open multiple client.py to create active users
        - Choose “<announce>” for message option in client.py
        - Type content of the announcement you want to make
        - See Figure 4 to see the screenshot of the tutorial

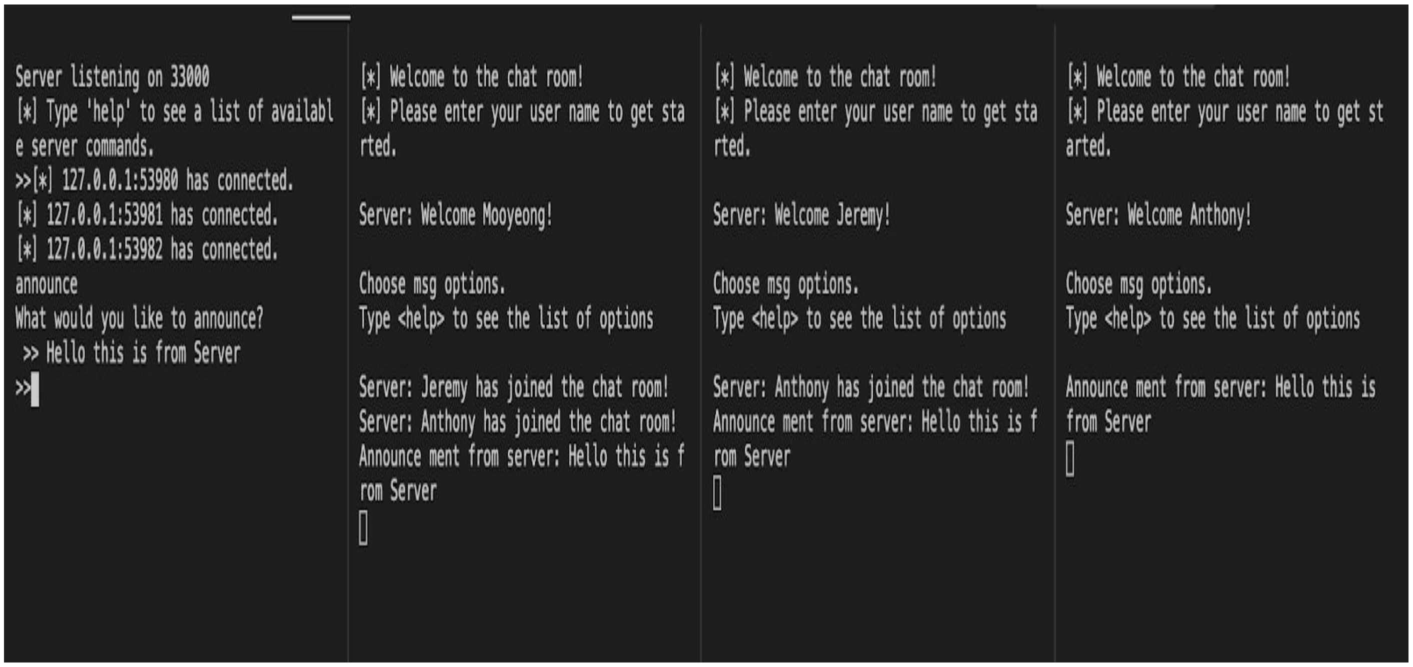


```
Server: Welcome Mooyeong!

Choose msg options.
Type <help> to see the list of options

Server: Jeremy has joined the chat room!
Server: Woosuk has joined the chat room!
Jeremy: Hi everyone this is Jeremy and I w
ant to make announcement
```

```
Server: Welcome Jeremy!

Choose msg options.
Type <help> to see the list of options

Server: Woosuk has joined the chat room!
Jeremy: Hi everyone this is Jeremy and I w
ant to make announcement
```

```
Server: Welcome Woosuk!

Choose msg options.
Type <help> to see the list of options
From Mooyeong to Woosuk: Hi Woosuk this i
s private message
Jeremy: Hi everyone this is Jeremy and I
want to make announcement
```

<Figure 3: A user "Jeremy" makes public announcements to all active users>



```
Server listening on 33000
[*] Type 'help' to see a list of availabl
e server commands.
>>[*] 127.0.0.1:53980 has connected.
[*] 127.0.0.1:53981 has connected.
[*] 127.0.0.1:53982 has connected.
announce
What would you like to announce?
 >> Hello this is from Server
>>
```

```
[*] Welcome to the chat room!
[*] Please enter your user name to get sta
rted.

Server: Welcome Mooyeong!

Choose msg options.
Type <help> to see the list of options

Server: Jeremy has joined the chat room!
Server: Anthony has joined the chat room!
Announce ment from server: Hello this is f
rom Server
```

```
[*] Welcome to the chat room!
[*] Please enter your user name to get sta
rted.

Server: Welcome Jeremy!

Choose msg options.
Type <help> to see the list of options

Server: Anthony has joined the chat room!
Announce ment from server: Hello this is f
rom Server
```

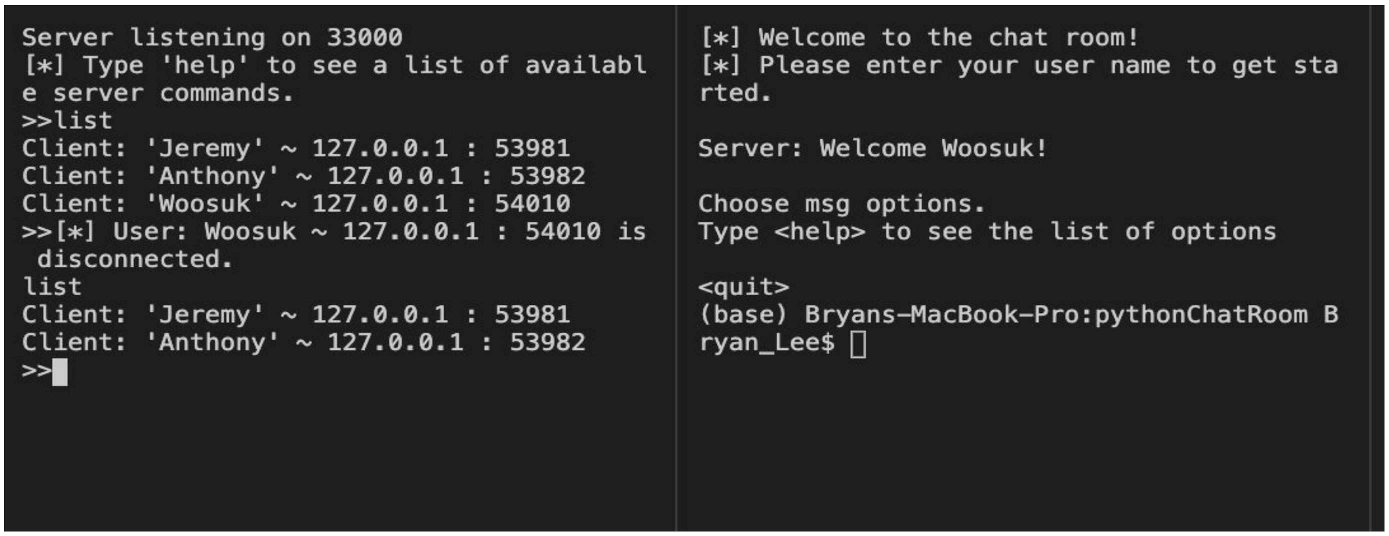
```
[*] Welcome to the chat room!
[*] Please enter your user name to get st
arted.

Server: Welcome Anthony!

Choose msg options.
Type <help> to see the list of options

Announce ment from server: Hello this is
from Server
```

<Figure 4: Server makes public announcements to all active users>

Querying a list of active clients

- The USERS variable in server.py stores all the active connections as it registers the socket in a dictionary in {socket: username} format. Since the close_connnection function deletes the disconnecting user from the USERS variable, USERS dictionary always stores only the active clients in the chatroom.
- In server.py, the server manager can query a list of active clients through “list” cmd in the server command options.
- In client.py, the user can query a list of active clients by choosing “<list>” options in the client command option. I gave access to a list of active clients to users to allow user to verify the recipient name by choosing a private message option (One-to-one communication). However, this can be improved in real cases as giving access to all users information may be dangerous

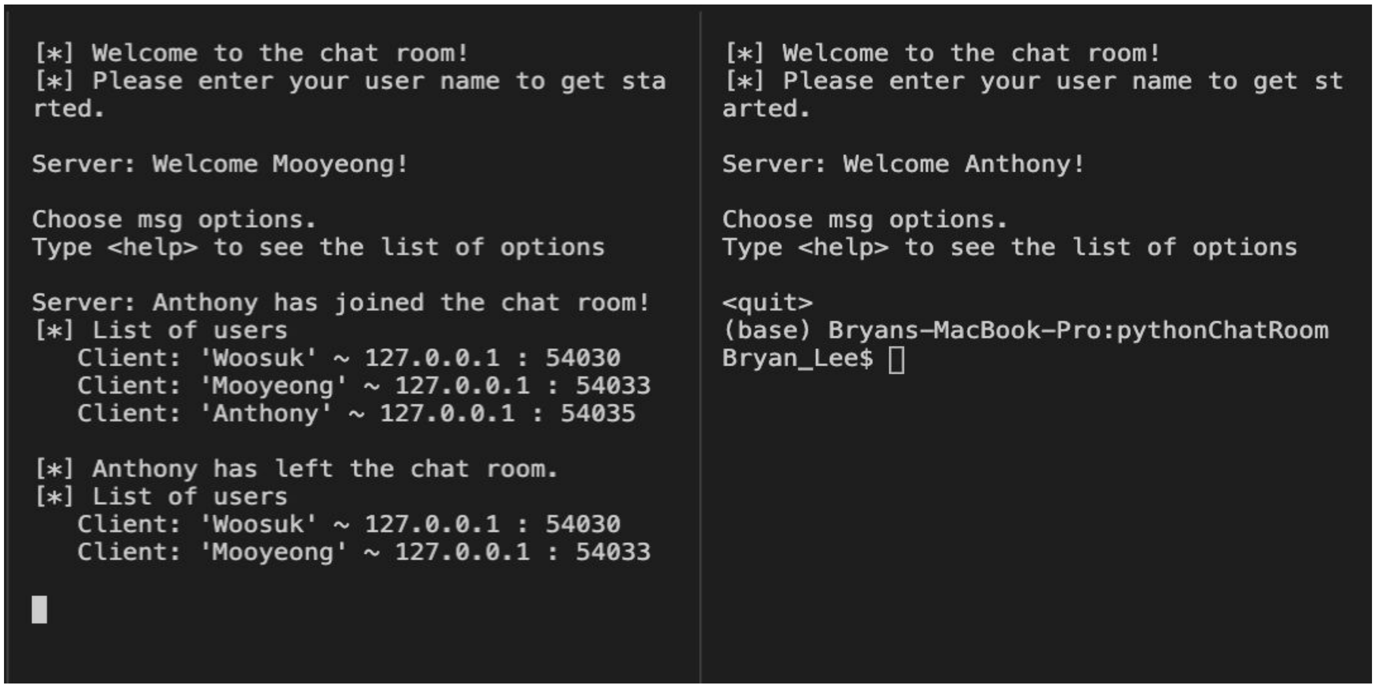
- Tutorial on querying a list of active clients
    - Server.py
        - Run server.py from the terminal: “python server.py”
        - Type “list” to query the list of active clients
        - See Figure 5 to see the screenshot of the tutorial
    - Client.py
        - Run server.py from the terminal first: “python server.py”
        - Run client.py from the terminal: “python client.py”
        - Choose the username you desire
        - Type “<list>” to query the list of active clients
        - See Figure 6 to see the screenshot of the tutorial



```
Server listening on 33000
[*] Type 'help' to see a list of availabl
e server commands.
>>list
Client: 'Jeremy' ~ 127.0.0.1 : 53981
Client: 'Anthony' ~ 127.0.0.1 : 53982
Client: 'Woosuk' ~ 127.0.0.1 : 54010
>>[*] User: Woosuk ~ 127.0.0.1 : 54010 is
 disconnected.
list
Client: 'Jeremy' ~ 127.0.0.1 : 53981
Client: 'Anthony' ~ 127.0.0.1 : 53982
>>
```

```
[*] Welcome to the chat room!
[*] Please enter your user name to get sta
rted.

Server: Welcome Woosuk!

Choose msg options.
Type <help> to see the list of options

<quit>
(base) Bryans-MacBook-Pro:pythonChatRoom B
ryan_Lee$
```

<FIgure 5: querying a list of active users in server.py. Updates list upon user’s exit>



```
[*] Welcome to the chat room!
[*] Please enter your user name to get sta
rted.

Server: Welcome Mooyeong!

Choose msg options.
Type <help> to see the list of options

Server: Anthony has joined the chat room!
[*] List of users
   Client: 'Woosuk' ~ 127.0.0.1 : 54030
   Client: 'Mooyeong' ~ 127.0.0.1 : 54033
   Client: 'Anthony' ~ 127.0.0.1 : 54035

[*] Anthony has left the chat room.
[*] List of users
   Client: 'Woosuk' ~ 127.0.0.1 : 54030
   Client: 'Mooyeong' ~ 127.0.0.1 : 54033
```

```
[*] Welcome to the chat room!
[*] Please enter your user name to get st
arted.

Server: Welcome Anthony!

Choose msg options.
Type <help> to see the list of options

<quit>
(base) Bryans-MacBook-Pro:pythonChatRoom
Bryan_Lee$
```

<Figure 6: Querying a list of active users in server.py, Updates when other user left the chat>

Conclusion:

There are multiple features that could be added on or improved on this chatroom program. For example, currently the user name serves as the id of each user, preventing duplicate ids, this can be improved by configuring indexing algorithms to create different id variables to allow duplicate username.

The assignment helped me to have better understanding over the network protocol, socket and multi threaded program. Even though the complexity of enterprise applications using networks is much more complex, the assignment provided a general design pattern in network communication.