COMP9331 Report
Name: Xinhong Yang
ZID: z5098300

Brief explanation

From the server side, the server waits inside the main while loop waiting for the new client connection. Once there is a new client establishes a connection, the server will create a new server thread to interact with that client. From the client side, each client program will start a separate thread for receiving the other client UDP connection at the beginning of the program before establishing the connection with server. The other client may then be able to communication to it.

Program design

The additional note in the assignment gave me a lot of help for getting this assignment done. I basically followed the exactly recommended procedure to design this assignment.

1: single user to login with the server

It is really similar to the previous TCP server and client lab. Just need to follow the same thing that is given in the TCP server and client code. The server is sitting inside the while(true) loop and waiting for the socket connection. The client established the 3-way handshake by connecting to that socket using the given IP address and port number. One thing that need to bear in mind is TCP connection using different port as argument.

Useful or helpful code resource:
TCPServer.java TCPClient.java in the 9331 labs

2. block functionality

This took me quick a while a finally got a solution. I used the HashMap data structure in java to store the invalid user that had attempted to login in and failed for more than the expected time. Whenever the user failed the authentication (I used authentication as a shoot cut for failing more than the expected time), I will use the username as a key and current timestamp as a key value and store them as a HashMap pair. When there is another attempt from the user, it will check the HashMap and find the timestamp record there. If the time difference from another attempt to its previous record in HashMap is more than the block time, we will delete the record and let the user typed their information. Otherwise, it will block the user.

3: multiple clients

I spent around 2 days to get the idea about how to do this part. I realised that I need to implement the multiple thread program after carefully reading the server design specification given in the assignment. Because I used java, so I used the thread object to help me implement the multiple thread server. It was very easy to understand once I got the idea about how the multiple thread works. Basically, just create another server handler class inside the server that extend ‘Thread’ and override the method ‘run’ inside the class and put the repones program inside the run. Inside the main server class and inside the main while loop that server is waiting for the socket connection. whenever there is a new connection from the client, create the multi-client handler class and start the new thread. And that is exactly how server handle the multiple client connection by creating multiple thread whenever there is a new client connects to the server.

Helpful resource:
https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html
https://www.youtube.com/watch?v=ZIzoesrHHQo&t=425s
https://www.youtube.com/watch?v=RQ2v0CSV4tY
https://www.youtube.com/watch?v=8lXI4YIIR9k&t=500s
https://www.youtube.com/watch?v=TCd8QIS-2KI&t=1953s
https://www.youtube.com/watch?v=YeXnJnQcyRY

4: posting, editing, deleting, downloading and logging off

This part is fairly straightforward once I got the idea about how to do the multiple clients. Essentially, I just put everything inside the run function and create different function for handling the different require from the client input. One of the tricky things in this part is to deal with the timestamp. Java has its own function to deal with the timestamp format that I did not know, and I did a lot of research on it. It used the library 'java.time.format.dataTimeformatter' to deal with the timestamp format. I watched some YouTube video and looked through the stack overflow to get the idea. For this part, also need to add some data structure like activeUser inside the multi-client handler class

Helpful resource:
https://www.youtube.com/watch?v=6cp4P4XZ9hE&t=320s
https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html
https://stackoverflow.com/questions/5902310/how-do-i-validate-a-timestamp

5: P2P communication

This part took me 1 day to get the idea and 1 day to finish coding. The idea is to generate a client receive thread before connecting to the server for every client. It was done by creating a client handler inside the client class. Inside the client handler class, under the run function, there is a while loop listening for the new UDP connection from another client. Once it established the connection between another client, it will start to receive the file byte by byte. Important: need to make a timeout value since the UDP transmission will lose packet. Sending the file byte by byte can be tricky especially using the datagram packet.

Helpful resource:
https://www.youtube.com/watch?v=0YvPY9gXUXg
https://stackoverflow.com/questions/13669541/string-sent-by-datagrampacket-and-datagramsocket-is-not-equals-to-literal-repres

Application layer message format

Below is my interaction message output format. (can used it as reference during marking)

Yoda terminal	Hans terminal	Server terminal
MSG Hello Message #1 posted at 18 Apr 2021 01:28:50.		yoda posted MSG #1 "Hello" at 18 Apr 2021 01:28:50.
What Error. Invalid command!		
	RDM 18 Apr 2021 01:28:49 #1, yoda: Hello, posted at 18 Apr 2021 01:28:50.	#1 yoda: Hello, posted at 18 Apr 2021 01:28:50.
OUT Bye, yoda!		yoda logout
	MSG Hello Message #2 posted at 18 Apr 2021 01:35:26.	Hans posted MSG #2 "Hello" at 18 Apr 2021 01:35:26.
	MSG Compute Network Rocks Message #3 posted at 18 Apr 2021 01:35:56.	Hans posted MSG #3 "Compute Network Rocks" at 18 Apr 2021 01:35:56.
	EDT #3 18 Apr 2021 01:35:56 Computer Network Rocks Message 3 edited at 18 Apr 2021 01:36:48.	Hans edited MSG #3 "Computer Network Rocks" at 18 Apr 2021 01:36:48.
	MSG Database Rocks Message #4 posted at 18 Apr 2021 01:37:21.	Hans posted MSG #4 "Database Rocks" at 18 Apr 2021 01:37:21.
	DLT #4 18 Apr 2021 01:37:21 Message #4 delete at 18 Apr 2021 01:37:21.	Hans deleted MSG #4 "Database Rocks" at 18 Apr 2021 01:37:21.
	MSG IoT Rocks Message #4 posted at 18 Apr 2021 01:38:35.	Hans posted MSG #4 "IoT Rocks" at 18 Apr 2021 01:38:35.
ATU Hans, 127.0.0.1, 8001, active since 18 Apr 2021 01:33:46.		Return active user list: Hans, 127.0.0.1, 8001, active since 18 Apr 2021 01:33:46.
RDM 18 Apr 2021 01:36:47 #3, Hans: Computer Network Rocks, edited at 18 Apr 2021 01:36:48. #4, Hans: IoT Rocks, posted at 18 Apr 2021 01:38:35.		Return messages: #3 Hans, Computer Network Rocks, edited at 18 Apr 2021 01:36:48. #4 Hans, IoT Rocks, posted at 18 Apr 2021 01:38:35.
UPD Hans example1.mp4 example1.mp4 has been uploaded	Received example1.mp4 from yoda	

Assumption I made for this assignment:

The userlog.txt file and the messagelog.txt file will be empty at the beginning of each test.
These files should be there in the same dir with my server and client file.
The video file should be there in the same dir with my server and client file.

Improvement

File transmission part for dealing the null value inside the byte [] array can be deal better rather than using for loop to manually count.