COMP9331
Computer Networks and Applications

Assignment

Yuqin Wang

Z5036694

Overall Design

The programs are implement on Python3.7

There are two main programs, client.py and server.py and two modules, communicateSys.py and userlog.py. One server will be created and several clients will try to login and communicate with each other. For every operation, message will be printed on server side and response message will be printed on client side.

Client program will create a TCP socket connecting with server, then prompt user to input username and password. Once username and password are correct, client login. The server will create two different UDP socket, one for sending file and the other for receiving file. The TCP and sending UDP sockets are in the main thread and the receiving UDP socket is in other thread which keep waiting next file.

Server program also create a TCP socket in main thread to listen for connection, a connection socket will be created for each client in other thread. These threads will process the client requests. Every time server start running, it will read the messagelog.txt to get the maximum sequence number. The first-time server start running, it assumes that there will be credentials.txt, messagelog.txt and userlog.txt in the same directory and the last two files are empty. After that, it can insert new record into new line. Any write operation to these files will recover the line by modified line and truncate the redundance line. When the server start running, it will print out a message about its IP address, the client must access the server through this IP address.

Login Phase Design

When the client start running the program, it will prompt client to input the username and password. When any of these values is empty, the program will ignore the input and prompt to input username and password again. Once the client input the username and password, it will send a login request to the server together with username and password. The flow chart of login phase and the request and response are shown in the table below.

Table 1. Protocol for login phase

Client's request	Server's response	Meaning
LOGIN\n username\n password\n\n	SUCCESS	Server response that client has pass the authentication. He/she can have other continue operations.
	FAIL num	Client input wrong password. 'num' represent the number of remaining login attempts that will not be locked.
	LOCK state [time]	Client username is locked by server. 'state' represents different state the client is in, the value can 001 or 0002, where 001 represent that the client has just been locked and 002 represent that the client has already been locked. When the state is 002, server will send the remain time that the client to be unlocked.
	NOEX	Client input nonexistent username.

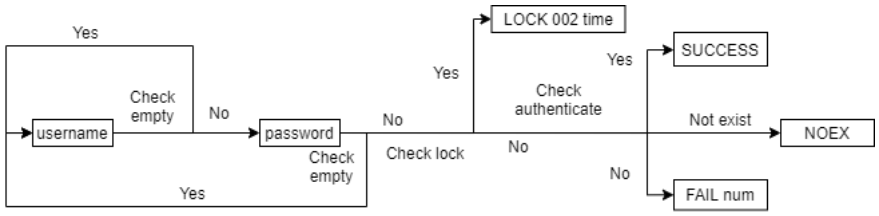

Figure 1. Flow chart for login phase

Every time client input incorrect passwords, the server will record the number of attempt and attempt timestamp based on username. Once the number of attempts reaches max login attempt, server will lock the username. The timestamp can determine how long the client has been lock. Before client is locked, server will return how many times left to login.

If client try to login with the same username (this is the drawback for this implementation, which will be discussed later) during this time, the server will response and tell the time left.

In this implementation, if client fail several times before being locked, stop attempting login for a long time (1 minute for this implementation) and attempt new login, the server recounts the number of attempts. Once the client input correct username and password, he/she can use the program to communicate to others. Also, server side will print a message that the client is login.

Communication Phase Design

After client login successfully, program will prompt client to input different command. All the command follows the specifications. When program receive input, it will first check whether the syntax is correct (correct number of arguments, #num is valid number (not letters or word) and include '#'), then the command will be translated into protocol and send to server. It should be noticed that when client give UPD command, he/she should give ATU command first to get active user list, or a message of 'no active user' will be printed. The following table describe the protocol receive from server.

Table 2. Protocol for communication phase

Command	Request Protocol	Response Protocol	Response
MSG	MSG/n massage/n username/n/n	POS\n sequence number\n timestamp\n\n	Server write message into messagelog.txt. Message post success. Server return the sequence number and timestamp.
DLT	DLT/n sequence number\n timestamp\n username\n\n	DTF 001	Delete message fail due to nonexistence sequence number.
		DTF 002	Delete message fail due to incorrect timestamp.
		DTF 003	Delete message fail due to attempt deleting another client's message.
		DTS\n delete time\n\n	Server will cover the deleted message with later messages and sequence number will be decrease by one. When recover finishes, EOF will be put on the position of file pointer and send delete success to clients.
EDT	EDT/n sequence number\n timestamp\n message\n username\n\n	ETF 001	Edit message fail due to nonexistence sequence number.
		ETF 002	Edit message fail due to incorrect timestamp.
		ETF 003	Edit message fail due to attempt deleting another client's message.
		ETS\n edit time\n\n	Server will cover the message which should be edited with new message and the position in the file of following messages will be adjusted. When recover finishes, EOF will be put on the position of file pointer and send edit success to clients.

RDM	RDM\n timestamp\n username\n\n	PRDM\n message\n\n	Server will read messagelog.txt and divide all the messages into smaller packets and send to clients. The second protocol will be used to send last packet.
		ERDM\n message\n\n	
		NMSG	There is no message.
ATU	ATU\n username\n\n	ATUL\n active user \n\n	Server will read userlog.txt and divide all the active user information into smaller packets and send to clients. The second protocol will be used to send last packet. Client will use a dictionary to store all the information of active user.
		EATU\n active user\n\n	
		NATU	There is no active user.
OUT	OUT\n username\n\n	LOS	Logout success.
UPD	UPD\n Sender username\n filename\n\n	None	This command will not be sent to server, therefore, no response from server. The file will be separated into different small packets with size 8k bytes. After a packet is send, client will wait 0.05s to make sure the second packet will not concatenate with first packet.

Consideration

Since the size of files, active user list and messages cannot be scale. When sending these objects, program will dived these into smaller packets and send. Last packet will have identification to indicate this is the last packet.
When client send file to another client, every time the packet is send, program will wait for 0.05s to make sure the audience finish writing of the previous packet.

Improvement

Login phase
Since the server lock the client base on username, meaning that the client can try different username many times which is not safe for the account. One way to improve is to lock the IP address also, it can avoid client to hack different account. Moreover, every time client is locked, the time for locking should be increased until it reaches the max lock time.

Communication phase
For RDM command, it should have two different syntax. The first one without timestamp and the second with timestamp. Server should give all the message that is not send by the client for the first syntax and send the message after timestamp for second syntax.
For UPD command, client should send to other client directly without ATU command explicitly. When the command is given, it should request server to get the specific user address and send the file base on the received address. Since when there is abundance of active user, it is hard to find the correct user to send the file.