COMP9331

Design Report

For assignment_1

Yansheng Liu
4-26-2017

First I have to say I haven't known the 'select' method in python until I complete this assignment. So both of the client and server python programs are based on multithreading (import threading).

whoelse

Find other online users.

whoelse<time>

Find history online users time seconds ago.

broadcast<message>

Send information to all online users except who blocks you.

block<username>/unblock<username>

Block/unblock a certain user.

message<user><message>

Send message to certain user.

logout

Use to log out of the system.

Client.py

login

The client part is divided into three functional parts: logging in, receiving and sending. Whenever user open the client.py, they should type in the user name and password as instructions. Then the username and password will have combined into a package and client starts to connect server. After connecting succeed and server receive the user name and password, and reply the certain 'welcome' string, client program can break the logging in loop and jump to sending and receiving part. If server sends the message other than 'welcome' one, client will show the string to user and stay in the logging in loop.

Receiving

Receiving is a loop in the main program body. Its function is like the logging in part. But in receiving part, it can only receive message from server and do different behaviors which depends on different server's replies. For example, if client receives a string which is 'The world dont need hero. This world need professionals. YOU DIED.', then client starts to close itself after sending the FIN message to server.

Sending

The sending part is a loop which use the multithreading method. So that sending and receiving can active 'at the same time'. Sending will record the user typing in and send the information after packaging. Sending doesn't care the meaning of the information. The only aim of sending part is packaging and sending message. When main program (receiving part) wants to stop itself, the sending part will stop itself automatically.

Server.py

Before log in:

After someone run the server.py, and type in necessary information, it will start opening the 'credentials.txt' and store all user name and password into a dictionary called box. And create a lot of lists, dictionaries and a timer to store information. Then welcome socket will open and wait for all clients. When server and a client connects successful, client will send a package which contains username and password. Server will check black_user_box and black_ip_box first, if the user name is in boxes, tell client 'you are blocked, plz wait.' Then use the user name and password pair to match the pairs in box.
If box[user name]==password, server will create a threading called subserver to this client. If box[user name]!=password, reply some sentences depends on the client records and close the connection.

After log in:

There are 4 classes in this part. Timerdown(timer), checkmailbox(for offline message), shout(login and log out notification) and subserver(send and receive message). All of them are based on threading.

Timerdown:

Timerdown is a timer in this program. It used to close inactive online client in a certain period. When a client online and active, it will renew its active_label. Timer will check all online user's active_label every certain time period(timeout/10.0), if a client do nothing in the whole timeout period, timer will send certain message to client. After client receives this message, it will send a confirm message to server then server will close the connection.
Timerdown will keep running until server program is closed.

Checkmailbox:

This class will run once after user login each time, if this user have offline message, checkmailbox will push all the message to this user. After finished process, this threading will end automatically. All offline message is stored in a dictionary called offline_mailbox.

Shout:

This class is used to send broadcast message to tell all online users who login or logout of the system. It will be called when user either start or end their own subserver. After sending the certain broadcast message, this threading will stop automatically.

Subserver:

Subserver is the fundamental part of the online part. Each online users corresponding to their own subserver threading. After a user login successfully, her/his subserver threading will be created, and from then on, subserver will keep running and try to catch message from the client until client

logouts(while 1). subserver uses the basic ‘IF, ELIF, ELSE’ command to estimate the information it received.

‘logout’:

When subserver detectes this string (yes, extracting the package and encoding), it will send a string 'Already logout Long may the sunshine!' to client. Then delete some necessary information (e.g. move the user out of the online list) and close the socket immediately.
‘whoelse’ and ‘whoelse<time label>’
When subserver detectes these two, it will reply users’ name who were online before the time label.
If the information doesn’t contain any time label, it will reply all online users at this time.
‘block<user name>’ and ‘unblock<user name>’
when subserver receive these two kinds of message, it will check if the user name is as same as the one who send this message. If not, it will check the block dictionary which stores all block information. Then subserver will append(user name) or pop(user name) if error occurred, send error message to client, or send succeed message to client.
‘broadcast<information>’
This will give all online user the same information except who blocks the message sender.
‘message<user name><information>’
This is used to send message to certain user. If the message receiver doesn’t online at that time, the message will be kept in the server and send it to user when she/he logs in next time.

Improvement

As I said at the start of this report, before I completed server and client programs, I haven’t known anything about ‘select’ function in python3. I have to use threading to realize the server and client. The advantage part is that, if they run in a computer which has large memory and good single core cpu, they will perform good. The disadvantage part is if computer’s memory and cpu is not good enough, server may crashes because of the overload of the computer.