REPORT

NAN ZHOU z5111593

TIP

This project is constructed based on python3. Please test it under corresponding environment.

PROGRAM DESIGN

In my program, server has been used as a central for all the clients. Clients who want to send message to other users must access to server first. Server will allocate a thread for each client. The transmission of message can be complete through communication through threads. Once client wants to send message to another user. The client will send it to server first. The server should add the message into a FIFO queue with necessary tags(e.g. name, address). Then corresponding thread which is occupied by receiver can collect the message by tags. To indicate activities of users on server. It maintains an online User table. Each row of online User table have these components: user name, status(online or not), user address, last active time. This table will help server react to most user commands quickly. And it will be updated in time.

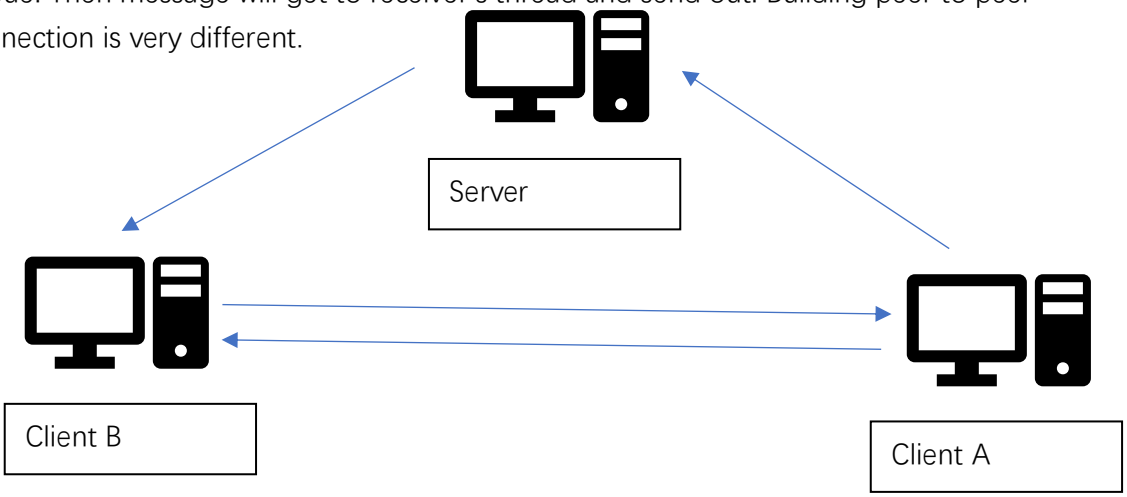
The design of client side is based on server. Since client cannot process data directly. It has been used as an input interface at most time. Each client owns 5 threads: receive thread, send thread, receive thread for peer to peer, send thread for peer to peer and one thread to activate socket on server. Communication between threads can be done through FIFO queue. Send thread generate command from interface and send them to server while receive thread display feedback from servers. Peer to peer threads are waiting at most time and trigged only if a p2p connection is fortifying.

Message format is various in my program, however the message head will always be commands. For example, of A want to message B. The request he send to server will be “message A B content”

WORKING PROCESS

Connection between users can be classified into two types: peer to peer or client-server-client. If a client wants to send message through server. He must log in first then build a TCP connection to server. Then client program can generate command by terminal then pass them to server. These commands have been edited on client side. Once server receive them, this message will get into queue and waiting for pick up of thread. Every thread will check items in

queue. Then message will get to receiver's thread and send out. Building peer to peer connection is very different.



If client A want to build p2p connection with client B, it will send its address and target name(client B) to server first. The server pass them to client B. Once client B receive data, he will initialize a TCP connection then send a hello message to client A. Hello message contains address of client B. Then client A can set up connection to B. After these, a p2p connection has finished.

In login process, client will send request which contain username and password to server. Then server should verify them and send back corresponding feedback. If user is valid, a "success' send back to client. Then client will send its username again and create socket and threads.

DESIGGN TRADEOFF

The apply of thread is too much in program. Client thread for peer to peer is a tradeoff since it was hard to switch socket while program been running. Both thread are waiting until a p2p connect get in. But in most time, they just wait.

The design of activate thread can activate socket on server side. Because of the use of queue. Every thread has to take data out of queue by themselves. For example, if user send message to another user. Receiver need to send something to sever and let its thread activate first. Then server can send message back. Activate thread in client threads is used to activate server side. It sends simple string "ack" to server every 1 second. Server will give a specific return message. Once client receive return message, it will be dropped. But it can make sure message come to client in time.

MPROVEMENT & BUG

Protocol is very fixed in my program. Since I used string to communicate client and server. Program must pay a lot of effort on deal with strings. JSON is a great choice for protocol. I used too much thread in my code. That result in waste. Because of string, all the commands have to be input exactly as format or it will pops out an error. The space can result in error as well because I use space to analysis commands.

Once you finish logout, you have to press crtl+c to close the program. Another thing is, if you build a p2p connection and close it. You have to login again if you want to build another p2p connection because it close threads which are in charge of p2p.

If you force quit a client, server will think that client is still online and log it out when timeout.

Since a build too much threads(6 thread for 1 client). It may cause some errors when too many users access to server.

Please wait a few second while client is receiving message from server.

Please tell contact me when horrible bugs happen.