Distributed System exercise 1 report

- Describe your proposed protocol using text, explaining your rationale behind each design decision and command. (3 marks)

When you run the program, you should use " python3 imclient_(user_id).py.

Run the two terminal to run same program at the same time.

Setting server as im.IMServerProxy('http://web.cs.manchester.ac.uk/r13355ty/comp28112_ex1/IMserver.php') in the cod base to make connection.

First stage : Getting two users

At the very first stage, check if there's another registered keys. If there is, then clear the server so new two users can use the program. After clearing all the pairs of keys and values, the program ask you about your role such as doctor or nurse. so even if two user choose same role, then it will make difference using index I. For instance, if first user type doctor and second user also type doctor, then first user will be doctor0 and second user will be doctor1 so system can recognise the who's the first user and second user.

At the stage we getting users, if there's only first user, then waiting for another user to come in.

After new user come in, we setting up a new key and value for checking who sent the message so at the initiation, I set up for the second user so that first user can type first.

Second stage: loop for messaging

There are two fucntions,which are called waiting for input and waiting for output function.

For waiting for input function, Get two parameter which are user role and other user.

There's infinite while loop because make second user wait until first user type so if sent_message is set as other user then, now user can type. For the type the message, call the function which are called typmessage function with parameter userrole.

In typmessage function, first we get message. If user type exit or Exit then it will quit the program.

Otherwise, message will set up as value of userrole and sent_message value will be changed to user role because now user sent the message.

Next function is waiting for output which has few functions. First one is checking time for other user's if other user don't type for 30 seconds, then will clear the server and quit the program.

Likewse, in typmessage function, if server is cleared because of time out, it will also quit the program.

Moreover, waiting for output function will check if other user accidently leave program, the this program also will crash as well. Last function is the showing output which is message sent by other user if the value of sent_message set as other user.

For my system architecture, I implement some codes to avoid deadlock. For example, if one of the users exit or terminating program because of not typing for 30 seconds, then it will make the quitting the program as well so it won't be stuck in deadlock.

– Give one example of a limitation of this system architecture, and one example of one way by which this limitation could be overcome, towards the realisation of a more realistic messaging system. (3 marks)

For my system architecture, there's a few limitations about dealing with dead lock. For example, if one of the users will terminate the program, then the other user wouldn't notice it so the other user will be stuck in that point forever. So this kind of deadlock is the one of limitation of this system architecture. However, I'm thinking if we use multithreading or doing server side code, I think we never stuck in deadlock because we can always check the state of bother users so even if one of users just accidently leave the program, server will notice if there's a user existing or not.

And trying to make a more realistic messaging program, I think the program should be simultaneously message each other, not just keep waiting for other's answer. This function can be also implemented by using multi-threading I assume.