Lab5 Design and Implementation Group 16

Changes on data structures:

1. To implement the new features in this lab, we added a new field in the message structure, so our struct message now is:

```
struct message {
        unsigned int type;
        unsigned int size;
        unsigned char source[];
        unsigned char data[];
        unsigned char session[];
};
```

2. In this lab, the structure of the client has a list of joined sessions, instead of a single session id. The server allows a client to create and join multiple sessions.

Changes on protocols:

However, we would not use this new field for features included in lab4. So those old features keep their original protocols and input formats. To implement the new features, we modified the protocol to adapt to the newly added field:

Old protocol:
 <type>:<data size>:<source>:<data>
New protocol:
<type>:<data + session name size>:<source>:<session>:<data>

Changes on commands:

1. Allow a client to join multiple session
   In lab4, once the client has already joined a session, joining or creating another session will return a JN_NAK. However, in this lab, we allow this operation. The server allows a client to create and join multiple sessions, and will add the newly joined session to the session queue and to the session list of the client structure.

2. Specify session ID when leaving a session

   New input format: “/leavesession <session ID>”

Correspondingly, in lab4, to leave a session, we just simply type "/ leavesession" without any session ID after.

In client.c, we also defined a linked list to store the session IDs that the user currently enrolled on the client side. Therefore, after the user typed in the command such as "/leavesession" or sending a message, we can check whether their aimed session is available for them (whether in the linked list), or help us print the sessions for the users.

Therefore, in this lab, we can ask the user to input the session ID they want to leave, which we put in the data field of the protocol, and send to the server.

3. Send message

   New input format:

   If single session enrolled:
   <text>

   If multiple sessions enrolled:
   <text>
   "Please choose a session to send, available sessions are:….."
   <session ID>

   To send a message, if the user only enrolled in one session, the server forwards the message to all active clients in that session. However, if the user enrolled in multiple sessions, in client.c we ask the users to choose which session they want to send to. To help them type the available session ID, in client.c we would print all session IDs where the user enrolled and the user should choose one among them. Then the server forwards the message to all active clients in the selected session.

   After a client receives a message, we can get the session ID from the new protocol, so that the program can show the session ID in front of every message the user receives.

4. Invite another user into a session

   We defined a new user input type for this feature:

   "/invite <user ID> <session ID>"

   , which is to invite the user having <user ID> to join the session named <session iD>. In our newly modified protocol, we put the <user ID> to the data filed, and <session ID> to the session field, also with newly defined message type “INVITE”.

After the server receives this invitation, it will check whether the user ID and the session ID is valid. If so, the server forwards this message directly to the targeted user, indicating that the targeted user is invited into a session. The user can simply type “Yes” or “No” to accept or refuse this invitation. If the user types “Yes”, then the client program will send a “/joinsession” request to the server. Otherwise, the user will not send anything back to the server, because according to Piazza, we do not need to notify the sender user whether the invited user accepts or not.

5. Other commands

The formats of “/list” and “/logout” commands do not change.

“/list” :
However, for each online client, the server will list all the sessions this client joined.
e.g. Online users:
    test_id1, in session: [session ID1][session ID2]

“/logout”:
If the client has joined any sessions, the server clears the client’s list of joined sessions and removes the client ID in all sessions in the session queue.

Timer

After login, if a client does not send any message or command to other clients or the server in a time interval, the server decides that the client times out. Since every client is assigned to a thread in the server program, and every data segment the client sends must be received by the server first, we added a timer to the recv() function in the server program. If the server does not receive any message from the client in our designed time interval, the client’s timer times out. Then the recv will return -1, and we will just force the client to logout.