IS-F462 Network Programming

Assignment 1 - Question 3

Introduction

This is the documentation for the group messaging app inside Linux using message queue where the design choices and particular implementations are explained.

Here Each feature would be explained with the design choices on both the server side and the client side.

General Structure

On client side

On the client side we simply use a switch case type functionality in a forever loop to give all the required functionality like

1) Asking the server to let us create or join a group or list all groups.
2) Asking the server to send messages to a user or a whole group.
3) We have also implemented a login/logout utility to account for online and offline status of users.
4) Read the messages which have been received.(Done using a child process)

Lastly you could obviously exit the whole program or keep it in background jobs in the logged-out state while doing other work on the foreground. This latter functionality represents the client in the offline state.

On server side

On the server side, we similarly use a switch case type functionality in a forever loop to handles the requests which are received from different users like -

1) Request by user to create or join a group or list all groups.
2) Request by user to send messages to another user or a group.
3) Request to login/logout of the system.
4) Request by user to read all unread messages.(Does not require any work from server at that instant.)

Lastly the server could continue to function till a SIGINT signal (Ctrl +C) is received after which it would clean-up and close the application on the server side.

General Structures and Features –

The chat groups are implemented as a combination of data structures. We have a single structure which contains the group-id and a linked list of messages and their metadata stored as the history. This allowed us to have a record of history which can be linearly accessed.

The users are implemented as another structure consisting of username , variables to know if the user is logged in or logged out and when was he last logged in. Lastly, we also have a linked list of pending messages in this data structure which are to be sent to this user whenever he/she comes online.

The membership of a user in the group is checked via a 2D matrix consisting of users and groups on the two different dimensions. This allowed us to check for membership in O(1) time.

The messages themselves are also a data structure which consists of the message type and message string itself. We also store data like the sender and recipient of message(group id also if the recipient was a group), the data about auto-delete option's validity and another data structure of timestamp for the auto delete option.

The time stamp data structure is a simple structure consisting of the time when the message was created and the duration for which the message is supposed to stay alive. This allowed us to implement the auto delete option.

Each client needs to log in to access all features of the group messaging application, this action makes their status ONLINE. Client can also go OFFLINE by simply using the log out feature.

Features of the Implementation

General Features –

Login to/Logout from the Server

On the client side, the client must choose the functionality in the options menu and a suitable message is sent to the server which does the rest of the work.

On the server side, whenever such a request of log out is received, we simply flip the status variable(to 0) of the user struct which implies that the user cannot send/receive messages .When the user logs in again, the status variable is made 1 again. The former state is offline state, and the latter is the online state of the client. For a new user, a new user data struct is formed and it is populated from the message which is received.

Send private messages

The client must choose the functionality in the options menu and then type in the user id and the message to be sent. This request would be sent to the server via a message queue.

The server would now decide if the recipient user were offline or online, if he is online then the message is sent to him via the appropriate message queue. If he is offline then the whole message structure formed is stored inside of the pending messages list of the concerned recipient user in the server, to be delivered when the user comes online.

Receiving messages –

The client when online would continuously call for message receive calls in a forever loop (through a child process of the main process),from a message queue which was populated by the server. The messages sent when the client was offline are retrieved from the pending message linked list on the server and shown, as soon as the client logs in.

The server would send all the pending messages for the user stored in the user data structure's pending messages linked list as soon as the user logs in. If the recipient user is already logged in when the message was sent by the sender, then this message would be sent by the server to the recipient instantly.

Auto-Delete Option

On the client side, we ask on every sent message if the user wants to set the auto-delete option, if yes then what should be the duration for which the message should stay alive. This information is stored in the timestamp data structure inside the message structure which is used by the server.

On the server side, whenever a message is supposed to be sent, we check if the auto delete option was switched on for this message and if yes then if the life of the message expired. If both answers were yes, then this message is not sent at all to the recipient user/group.

Group-related

For Creating a Group -

The client must choose the create group functionality from the options menu. This request would be sent to the server via a message queue and rest of work would be done by the server. This would return a group id which we can use for further group chats.

The server would first check if the maximum number of groups are already formed. If not, then it creates a group structure and initializes it with this user being a member of the group using the membership matrix value of membership[group_id][user_id]. Server then also sends the group ID to the client which requested the creation of a group.

For Joining a Group -

The client must choose the Join group functionality from the options menu and enter the group ID. This request would be sent to the server via a message queue and rest of work would be done by the server.

The server would check for the existence and capacity of this group and choose to make the user a member of the group accordingly using the membership matrix, by changing the value of membership[group_id][user_id].

For Listing All Groups

The client must choose the List group functionality from the options. This request would be sent to the server via a message queue and rest of work would be done by the server.

The server then just sends messages each consisting of a user and its status(1-online or 0-offline) OR the group id(Info stored in an array) and the participants in each group(Info stored in the membership matrix).

We have also given the data about users here so that client knows which users in a group are ONLINE/OFFLINE.

Sending messages to groups

The client must choose the Send to Group functionality from the options menu and enter the group ID along with the message to be sent. This request would be sent to the server via a message queue and rest of work would be done by the server.

The server then checks if the gid group exists or not, and if the requesting user is a member of the group from the membership matrix. In case of affirmative response to both we send the messages to all the users who are online via the message queue and for all the users who are offline, the message is stored at the tail of the pending messages linked list of the user stored at the server. The messages in this linked list would be the first to be delivered once the user comes online.