Network Programming Assignment, Design Document: Q3

Student Name	**Student ID**
Aaditya Kumar	2017A3PS0332P
Anirudh Buvanesh	2016B4A70614P
Ashish Kumar	2016B4A70636P

The problem makes use of message queues to build a client server based chat system. Details of the different features w.r.t client and server are highlighted below.

1. **Registration**
    a. **Client:** When the client enters he is prompted to enter his client id if he is coming again or enter -1 if he wants to register himself with the server. A *SYN* request is sent to the server to fetch metadata of the client such as client id and group information he is part of (see *client_sync* structure). Post this the client can receive messages from other members using the message queue. The message type is specified as *SYN_REQ.*
    b. **Server:** When a server receives a *SYN* request, it either creates a new client or fetches an existing client metadata stored in a table (*client_data_ds*) and writes the reply to the client message queue. The server maintains it's message queue which is commonly known to the client for handling all requests specified in the menu. Server sends a *SYN_ACK* for the *SYN_REQ.*
2. **Sending message**
    a. **Client:** The client has the option of sending a message to a person or group, for which he needs to specify the destination id, message he wants to send and if he wants to enable timeout which is an optional feature of group messages. The message is encapsulated in the structure specified by *message* and sent across with a message type of *SEND_MSG.*
    b. **Server:** The server appends this message to the message queue specified by destination id, in case of the group the message is written to the message queue of all the members of the group. The message is also buffered along with metadata in a table called *queued_group_messages* so that people who join after sometime get the message depending on message timeout. Server sends an *ACK* for the message.
3. **Receiving Messages**
    a. **Client:** There is a child process running on the client side which executes a blocking read call on it's message queue waiting for messages of type *SEND_MSG* to see if there are any pending messages from groups or other senders. Messages with source and timestamp are printed.

    b. **Server:** Explained as part of **2.b**

4. **Creating groups**
    a. **Client:** The client needs to specify the name of the group he wants to create, if all goes well then client receives a group id for the group he created else a message is logged indicating the reason for failure.
    b. **Server:** Server checks for if the group already exists, if not the client is made a member of the new group and the table containing group information and membership information is updated.
5. **Join group**
    a. **Client:** Client needs to specify the name of the group he wants to join, if all goes well then client receives a group id for the group he created else a message is logged indicating the reason for failure.
    b. **Server:** Server checks if the group exists and the client is not a member of the group, if this is true the group information table and membership table are updated.
6. **EXIT**
    Used to stop the process on the client side.

Additionally, semaphores are used to control printing blocks of logs atomically for maintaining cleanliness on the client side.