Assignment 1

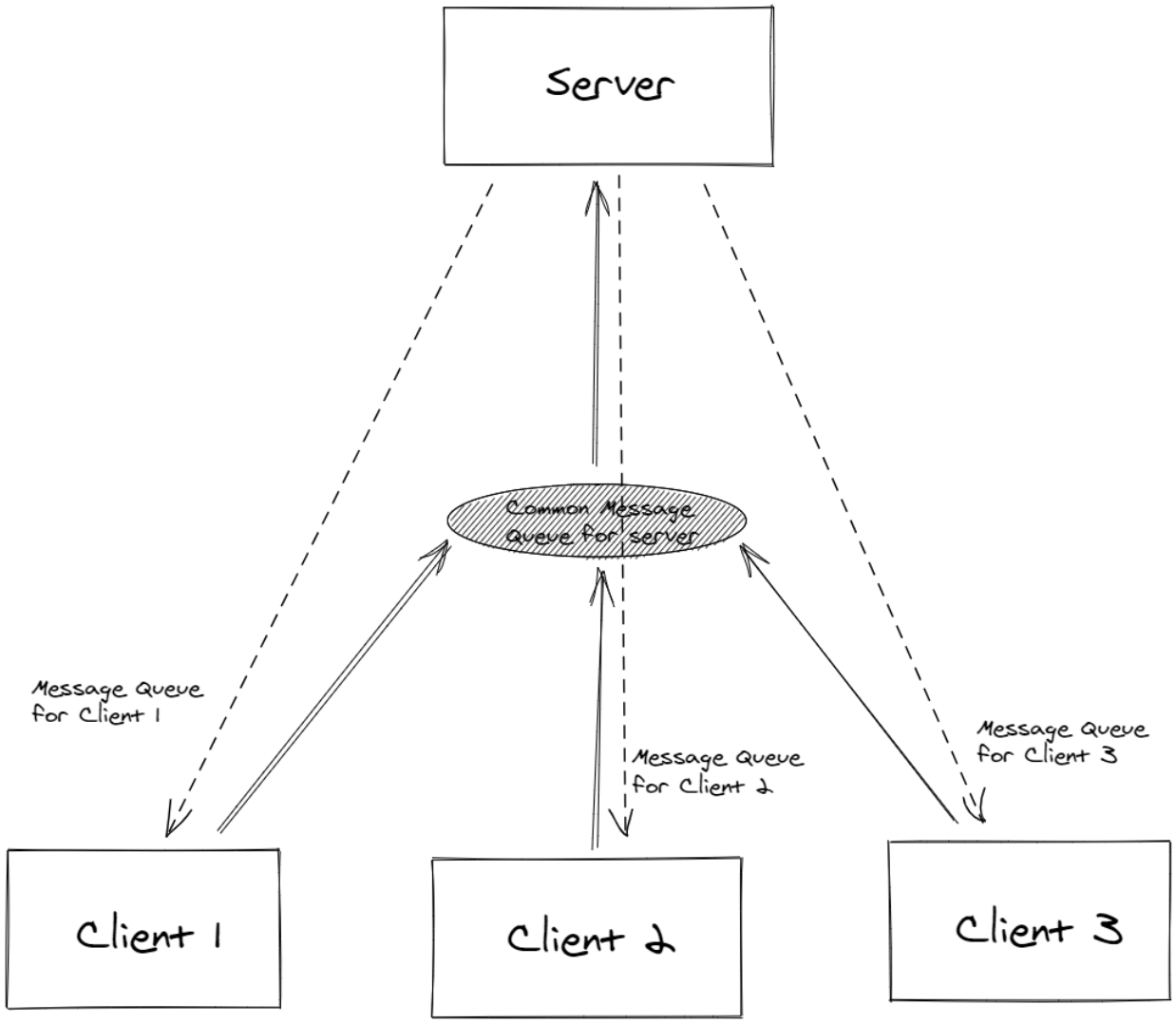
P3 - Group Chat Management System

Submitted To - Dr. Hari Babu K.
ID NOs - 2018A7PS0219P, 2018A7PS0166P

Design

This exercise develops a group chat system which allows users to create groups, list all groups on the server, join groups, send private and group messages and receive messages in online as well as offline mode. The chat system also implements an option which can be set for a group where users who join a group after < t > seconds from the time of message creation also receive it.

The design for the server/client mechanism :-



Server:

The server starts before all other clients and has its own message queue. The server coordinates between all the clients. The server receives the message from the queue and based on the type of message, it updates its internal state and sends responses to appropriate clients.

Client:

Every client is uniquely identified by their username which is entered as soon as the client program is launched. The client name cannot be “server”. Every client receives messages through its own message queue and it only sends messages to the server. The client has two threads running. One is a prompting thread which prompts and asks for user input. On receiving the command, it parses and sends the appropriate message to the server message queue. Another is a receiving message thread which reads messages from the client message queue sent by the server and then prints the messages on the terminal.

Identification of Message Queues:

The message queues take an integer as an input for identification. Typically, this would be the output of the `ftok` function. However, it can take arbitrary user-defined integers too. This is taken advantage of by us to create a separate message queue for each client. The username of each client is hashed by a simple hash function ‘djb2’ which takes a string as input and outputs a fixed length integer. Whenever a client wants to send a message to the server, it will send the message to a message queue with the ID obtained from hashing the string “server”. The message contains the fields as mentioned in ‘Message structure’ so the server can appropriately send responses.

```
typedef enum _MSG_TYPE {
    PRIVATE_MSG,
    GROUP_MSG,
    CREATE_GROUP_MSG,
    LIST_GROUP_MSG,
```

```
    JOIN_GROUP_MSG,
    AUTO_DELETE_MSG
} MSG_TYPE;


typedef struct _MSG {
    MSG_TYPE type;
    char sender[MAX_NAME_LEN];
    char receiver[MAX_NAME_LEN];
    char body[MAX_MSG_SIZE];
    char group[MAX_NAME_LEN];
    time_t time;
    time_t delete_time;
} MSG;
```

Usage

Creating and Joining group

`create` command is used to create a group and join command is used to join a group. The user who creates the group is already a member of the group.

```
$ create <group_name>
$ join <group_name>
```

List Groups

This command prints all the created groups, the client waits for the server to return the response.

```
$ list
```

Sending messages

`send` command can be used to send private and group messages with the options `-p` and `-g` respectively.

```
$ send -p <user_name> <msg>
$ send -g <group_name> <msg>
```

Auto Delete

`auto delete` is a command which allows users of a group to set delete time for the group where users who join a group after `<t>` seconds from the time of message creation also receive it.

```
$ auto delete <group_name> <t>
```

How to Run:

The following command is run on the server machine. It compiles and runs the server executable.

```
$ make run_server
```

The following command is run on the server machine. It compiles and runs the server executable.

```
$ make run_client
```

To exit the process you can press Ctrl + C, regardless of client or server.

Assumptions:

- When a user creates a group, he automatically becomes a part of the group.
- Fixed number of max users and group size.

- Clients cannot keep “server” as their username.