Lab 5: Implementing a Peer-to-Peer Chat Application

Surya Addanki 15CS10002

Satish Kumar Karri 15CS10018

1 Compilation and running procedure

- Compile the p2pchat.c using

  ```
  $ gcc -std=c99 p2pchat.c
  ```

- Or just use the Makefile to compile,

  ```
  $ make
  ```

- The file *users_list.txt* has the list of friends and their addresses and is same for all the users
- Run the chat application using your name with the following command

  ```
  $ ./a.out <PORT> <YOUR_NAME>
  ```

- To send a message to a friend, the format is

  ```
  $ <Name of friend>/<message>
  ```

- The message should not be longer than 280 characters.

2 Protocol and Working

The application uses TCP protocol underneath.

1. User can send a message to any of his peer in the format specified before.

2. Every user maintains a list of his active connections. Everytime before sending a message the user checks his connections for an active connection between the two.

3. If present, the user sends the message over the already present connection. Else the user opens a new TCP connection to that friend and adds it to the connection list.

4. The other end user accepts connections and adds the friend to his active connections list.

5. Since, there's no way for the connection accepting user to know the connection initiator's name, the initiator sends his name in the first message, which the other end has to decode.

6. When one of the users closes the connection or TCP connection timeout occurs, the connection is removed from either user connection lists.

7. On receiving a message, the message has to be printed to STDOUT.

3 Design

3.1 Data Structures

- The connections at each user are maintained in the form of a linked list. The node struct is given below. The name field is the name of the friend and fd is the file descriptor corresponding to the connection.

```
typedef struct conn_node{
        char name[20];
        int fd;
        struct conn_node *next;
}conn_node;
```

- A static friends list is maintained which is populated from the *users_list.txt* file. Every friend address has name, his IP and port his application runs on.

3.2 Implementation

- **Select()** system call has been used for implementing the multi-user application.
- Select() checks all the fds in the readset for any events and reports. We'll have to loop over the fds and handle the events.
- The overall implementation is summarized by this flowchart.