INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR

CS39006: Networks Laboratory

Assignment-8

REPORT ON Peer to Peer chat

Prepared by

G.MANEESH KUMAR(18CS10020),

IIT KHARAGPUR.

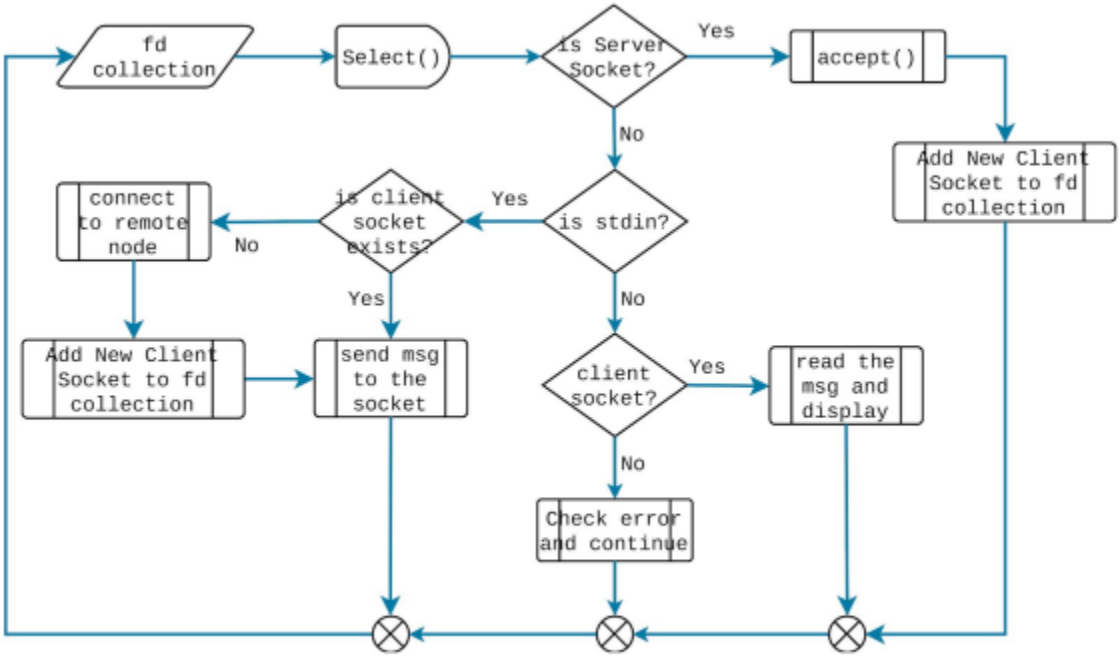
Working of code

- User first need to enter the port with the corresponding executable file as arguments .
- Reading the command line arguments.
- Each User will Contain the user info in user_in.txt.
- Reading the user info from the text file and then stored it in the struct type as shown below.

```
typedef struct user_detail
{
    struct sockaddr_in sa;
    int socket_fd;
    long last_interaction_time;
}user_detail;
```

- **sa** for storing the info of the other users.
- **socket_fd** (initializing with -1)is the communication socket of the user in sa.
- **last_interaction_time** is the time of last interaction with the corresponding user in sa.
- We are using unordered_map to store user info because each user can be identified by username.(since username is unique for the node in the peer-to-peer network).
- After that creating a socket and binding and then listening through the sockfd.

- We are using select() to reduce the blocking behaviour of blocking syscalls.
- select() will check the fd_set and destroy the non-active fd's in it.
- We are also giving the maximum fd(maxfdp1) value as the attribute to select().(NOTE: maxfdp1 is not the size of the fd_set that we are using in select).
- And the implementation of code mainly follows the state diagram given in the assignment.



- If it's a server socket ,we also read the port after the accept() ,since if the users are on the same ip,then we need a port to distinguish,but after the accept() syscall OS randomly assigns a new port. So, we also read the

port.(And also checking whether the user is present in the user info (or) not (check_user)).

- And also make the last_interaction_time to present time and socket_fd to the accept() returned fd for the accepted user.
- If it's not server socket and stdin then,read data from stdin and send the data to corresponding user, here we also checked socket_fd for the corresponding user that we want to send the message is “-1” (or) not.(.i.e already connected or not)
- If the connection with the user is not their then connect and also send the **port**.(we previously after accept() read the port number).
- And also update last_interaction_time to present time.
- And finally send message.
- If it's not server socket and not stdin then, it must be some connection socket_fd, so check them by iterating in for loop and correspondingly display message.
- And also update last_interaction_time to present time.
- As,previously mentioned select() is destructive ,we again iterate through user_info and check the active clients and push them to fd_set.
- Finally for the timeout checking we used threads which is more reasonable ,as it does parallel checking.
- Thread with function(check_timeout) is joined to main thread , as if only all the connections are timed out then display them and then our user main process will exit();

Compilation and Running procedure

- Firstly,If any of the new user want to enter peer to peer network,then accordingly change the user_in.txt file.
- Then ,next enter the command “make” in the terminal.
- After that, we can see the executable file ,then make sure you are running the executable along with the port number.
  (format:- ./(executable filename in our case peertopeer) <port>)
  Ex:- ./peertopeer 2000
- Then we can start chatting with others ,if they are also running.
- We can also run command make clean for removing the executable file.

Sample input and Sample output:-

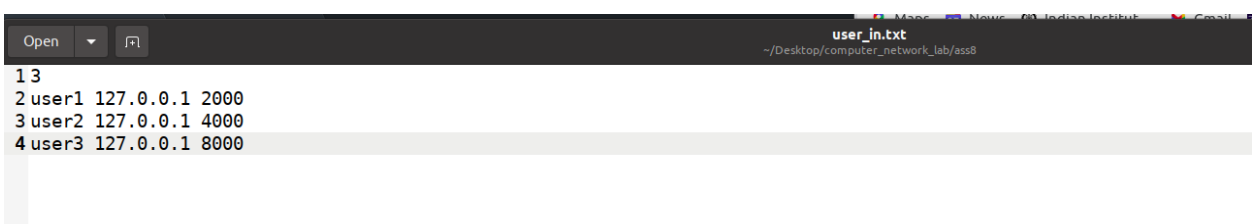
- Firstly ,add the users into the user_in.txt file as below format.

  number of users in peer to peer network
  username ip_address port
  username ip_address port
  username ip_address port
  .
  .
  .
  .

- Our sample input is

  3
  user1 127.0.0.1 2000
  user2 127.0.0.1 4000
  user3 127.0.0.1 8000

- Then run command “make”,then run executable as:
  - ./peertopeer 2000
  - ./peertopeer 4000
  - ./peertopeer 8000
- And the sample input,output is shown below.



```
manish@manish-Inspiron-7573: ~/Desktop/computer_network_lab/ass8
manish@manish-Inspiron-7573:~/Desktop/computer_network_lab/ass8$ ./peertopeer 2000

 -----------------
| User Info table |
 ------------------
user1   127.0.0.1       2000
user2   127.0.0.1       4000
user3   127.0.0.1       8000



  Socket successfully created..
  Socket successfully binded..
  Server listening..

-------- P2P chat is succesfully Started!! enjoy with chat ------

Welcome user1 .You can start chatting

Enter 'quit' to exit.

Message format: username/<msg>

user2/hello
Messege send to user2 with ip:127.0.0.1

user3/hey
Messege send to user3 with ip:127.0.0.1


user2  : hello


user3  : hello
```

```
manish@manish-Inspiron-7573: ~/Desktop/computer_network_lab/ass8
manish@manish-Inspiron-7573:~/Desktop/computer_network_lab/ass8$ ./peertopeer 4000

 -----------------
| User Info table |
 ------------------
user1   127.0.0.1       2000
user2   127.0.0.1       4000
user3   127.0.0.1       8000



  Socket successfully created..
  Socket successfully binded..
  Server listening..

-------- P2P chat is succesfully Started!! enjoy with chat ------

Welcome user2 .You can start chatting

Enter 'quit' to exit.

Message format: username/<msg>


user1  : hello

user1/hello
Messege send to user1 with ip:127.0.0.1

user3/hey
Messege send to user3 with ip:127.0.0.1


user3  : hey
```

```
manish@manish-Inspiron-7573: ~/Desktop/computer_network_lab/ass8
manish@manish-Inspiron-7573:~/Desktop/computer_network_lab/ass8$ ./peertopeer 8000

 -----------------
| User Info table |
 ------------------
user1   127.0.0.1       2000
user2   127.0.0.1       4000
user3   127.0.0.1       8000



  Socket successfully created..
  Socket successfully binded..
  Server listening..

-------- P2P chat is succesfully Started!! enjoy with chat ------

Welcome user3 .You can start chatting

Enter 'quit' to exit.

Message format: username/<msg>


user1  : hey


user2  : hey

user1/hello
Messege send to user1 with ip:127.0.0.1

user2/hey
Messege send to user2 with ip:127.0.0.1
```

```
manish@manish-Inspiron-7573: ~/Desktop/computer_network_lab/ass8
manish@manish-Inspiron-7573:~/Desktop/computer_network_lab/ass8$ ./peertopeer 2000

 -----------------
| User Info table |
 ------------------
user1   127.0.0.1       2000
user2   127.0.0.1       4000
user3   127.0.0.1       8000



  Socket successfully created..
  Socket successfully binded..
  Server listening..

-------- P2P chat is succesfully Started!! enjoy with chat ------

Welcome user1 .You can start chatting

Enter 'quit' to exit.

Message format: username/<msg>

user2/hello
Messege send to user2 with ip:127.0.0.1

user3/hey
Messege send to user3 with ip:127.0.0.1


user2  : hello


user3  : hello
```

```
manish@manish-Inspiron-7573: ~/Desktop/computer_network_lab/ass8

  Socket successfully created..
  Socket successfully binded..
  Server listening..

-------- P2P chat is succesfully Started!! enjoy with chat ------

Welcome user2 .You can start chatting

Enter 'quit' to exit.

Message format: username/<msg>


user1  : hello

user1/hello
Messege send to user1 with ip:127.0.0.1

user3/hey
Messege send to user3 with ip:127.0.0.1


user3  : hey
```

```
manish@manish-Inspiron-7573: ~/Desktop/computer_network_lab/ass8

  Socket successfully created..
  Socket successfully binded..
  Server listening..

-------- P2P chat is succesfully Started!! enjoy with chat ------

Welcome user3 .You can start chatting

Enter 'quit' to exit.

Message format: username/<msg>


user1  : hey


user2  : hey

user1/hello
Messege send to user1 with ip:127.0.0.1

user2/hey
Messege send to user2 with ip:127.0.0.1
```

Output of user1,user2,user3 and all the users