

# PROTOCOLS FOR PROJECT 3

**Shubhankar Chaudhuri**

## **Number Of Components**

---

Project 3 will have four components

- \* **Command Input**
- \* **User Module**
- \* **Localized Chat Server**
- \* **Server Configuration Administrator**

## **Command Input**

---

The command input is same as project 1 and project 2 , and it will take input from console , and preprocess the first word of user input and send it to user module or report an error message.It will shut down on exit command.  
It has the following threads

- \* a connection thread to user agent on a dynamic creation basis

It has the following data : User agent IP,user agent command receiving port

## **User Module**

---

The user module does the following work

\*has the following threads

- \***command receiver thread** from user input
- \***sender thread** to send request or message
- \***multicast receiver thread** for receiving room messages

\***receiver thread** for peer to peer messages

\***server selecting thread** which checks the command database and whenever there is a long wait and timeout it shifts server by setting a flag which gets reset as soon as the server is switched

\*it has the following data as well

**chatroom data**-----chatroom name,chatroom server multicast IP,chatroom server multicast port

**server data** -----server IP,server receiving port,server name

**neighbour data**-----neighbour name,neighbour IP,neighbour port,neighbour hosting(1 if neighbour is hosting user for temporary fault,0 if it is connected to its original server)

**user data**-----user id,user server name,user server IP,user server port,user alias,User IP,user port

**search data**-----searchid,searched alias name,number of results

**unresponded command data**-----command id,command time instance

\*gets the command from user and does the actual processing

\*the command is of the form **<commandid,command>**

it processes the command and sends message in following format

**<MessageType,servername,server IP,server port,self alias,self IP,self port,arguements>**

**Message Type = 10** send to user module in peer to peer connection with arguements being the message

**Message Type = 20** multicast send to room with arguements being the message

**Message Type = 25** search clear data with arguement search id,the server wont return any result then and send all neighbours in sequence clear message

**Message Type = 30** search request with arguements searchid and search alias

**Message Type <40** createroom ,leaveroom,joinroom

**Message Type = 27** send to a neighbour server telling to host with no arguements

**Message Type = 28** send to the neighbour telling to unhost since original server is restored with no arguements

Since there is a server name,server IP and server port in the search message,the Chat server can identify whether it is temporary hosting the user module or it manages the user module itself only!!

## Chat Server

The Chat server is the host for various chat rooms and which gets the neighbour as well as server configuration administrator and user modules message and sends them. It has the following threads

\* a **sender thread** on dynamic creation basis which sends messages to the server configuration administrator or other chat servers

\* a **reciever thread** from other chat servers as well as the server configuration administrator

\* a **command reciever** thread from user modules of the clients

It has a collection of data

**neighbour list** ----- neighbour id,neighbour server name,neighbour server IP,neighbour server receiver port,neighbour added(1 if the neighbour acknowledged itself as neighbour as well,0 if not acknowledged)

**chatroom list** ----- room id,room name,room people,room seed,room multicast IP,room multicast port

**self user list**----- user id,user alias,user room id,user IP,user receiver port

**available multicast IP list**----- multicast IP,chatroom ID (-1 if unoccupied)

**unresolved search list**----- search ID,localuser(1 if searcher is local,0 if searcher is external to server),userid,searchalias  
neighbour Id(to return search ,dont utilise if localuser is 1)

**external user list**-----userid,userhosted(1 if it is under the server temporarily due to fault tolerance,0 if it is to original server),neighbour server id,user alias,user IP,user receiver port

It receives messages from chat server,user agent as well as server configuration administrator

## Message From chat server

<Message Type,Server Name,Server IP,Server Port,arguments>

**Message Type = 0** Add Neighbour with arguments being the self user list and roomlist,both containing the number of individual lists as well,and  
neighbour list with <neighbour name,neighbour IP,neighbour port> for the number in neighbour list

**Message Type = 1** Added Neighbour with arguments 1 for success and 0 for failure

**Message Type = 2** Search request with arguments provided in search list

**Message Type = 3** Search response with arguments server name,number of users,<user alias,server IP,server port,user IP,user port> for the

number of users

**Message Type = 4** Search clear with arguments search id

### **Message To User Module**

-----

**<commandID,MessageType,arguments>**

**Message Type >99** Error with argument being the error message

**Message Type > 50** responses for createroom,joinroom,leaveroom

**Message Type = 50** response for search with arguments search id,user server name,user lp,user port

**Message Type = 40** for change in name arguments being new name(**commandid is -1**)

**Message Type = 260** for acknowledging temporary hosting

**Message Type = 310** for telling it is restored(**commandid is -1**)

### **server configuration administrator**

-----

this is the system for naming servers and providing them list of multicast IPs.It sends and receives messages only with servers and so has the following threads

\* a **receiver thread**

\* a **sender thread** on dynamic creation basis

It has got the following data

**server list** -----server id,server name,server IP,server receiver port,server multicast range ID,server neighbour list

**multicast range list** ----- multicast range id,multicast lowest,multicast highest,server id(-1 if not used)

### **Message From Chat Server**

-----

**<Messagetype,arguments>**

**Message Type = 600** join the server list with arguments server name,server IP,server port

**Message Type = 601** change the name

### **Message To Chat Server**

-----

**<Messagetype,arguements>**

**Message Type = 1001** joined the list with arguements multicast IP lower,multicast IP higher,number of neighbours,<neighbour server name, neighbour server IP,neighbour server port> for the number of neighbours

**Message Type = 1000** error with arguements being error message

**Message Type = 1002** name updated

The search id is a string made by concatenation of sender IP,sender port,searched alias timestamp.

The term data and list generally refer to the databases.