Final Report

Anand Sriramulu, Arpit Panwar, Jitesh Gupta

Classes :

i. Chat Node Class

Members	**Functionality**
bool bIsLeader	Mentions if is the leader or not
int statusServer;	Maintains the status of the server if it working in normal operation or holding election
int electionstatus;	Maintains if the current server started the election or not
long lastSeqNum;	Maintains the last seens seqNum
ipAddress[IP_BUFSIZE]; sUserName[USERNAME_BUFSIZE]; rxBytes[RXBYTE_BUFSIZE]; portNum;	All these values store the details of itself .
LEADER lead;	Stores details about the current leader in the chat system.
list<USERINFO> listofUsers;	Maintains a list of current users in the chat
map<string,string> mClientmap; map<string,double> mStatusmap;	These map the user name to ipPort and the last time those users were seen.
list<sockaddr_in> listofSockets;	Maintains a list of sockets the current user has to send out a message to .
PriorityBlockingQueue<MESSAGE> holdbackQueue; Queue<MESSAGE> consoleQueue; Queue<MESSAGE> sendQueue; Queue<string> printQueue;	Different Queues being handled by different threads to maintain ordering.

ii. udpServer

int f_socket; int f_port; std::string f_addr; struct sockaddr_in f_addrserver;	Holds details about the socket that has been created.

iii. MESSAGE Structure

char sContent[MESSAGE_SIZE];	Contains the message to be sent of size 2048 bytes
int sType;	Mentions the type of message being sent
long lSequenceNums;	contains the sequence number of the message.

iv. LISTMSG Structure

int numUsers;	Mentions the number of users in the group
char leaderip[IP_BUFSIZE];	Has the ip address of the current leader in the chat
int leaderPort;	Mentions the portNumber of the current leader.
char listUsers[MESSAGE_SIZE];	Contains the list of users
char leaderName[USERNAME_BUFSIZE];	Contains the leaders userName.

v. Leader Structure

char sIpAddress[IP_BUFSIZE]; int sPort; char sName[USERNAME_BUFSIZE];	Stores the details of the current leader .

vi. USERINFO

ipaddress[IP_BUFSIZE]; portnum[PORT_BUFSIZE]; username[USERNAME_BUFSIZE]; char rxBytes[RXBYTE_BUFSIZE];	Contains details about each user present in the chat room

vii. Queues

Blocking Queue	Modified STL Queue to make the push and pop operations thread safe.
Blocking Priority Queue	Overwritten the STL priority Queue to make the it thread safe and return the values in ascending order of the priority queue.

Threads

(Note: All the threads are waiting on a blocking event so none of them are polling thus reducing the cpu usage)

sendMsg	Used to send messages to leader/users by popping from the sendQ.
RecvMsg	This thread receives messages and ACK all the messages being received .It also does partial parsing to determine if the message has to go to Holdback Q or console Q.
processThread	This thread pops from the console Q processes the message according to the type and puts it to the send Q.
Holdback Thread	This thread pops from the holdback makes sure total ordering is maintained and pops out the message into the print

	Q.
printConsole	This thread only keeps printing onto console.
heartbeatThread	This thread sends out heartbeat signals and it spawns a thread which conducts election on receiving a message of type election.
heartbeatSend	This thread keeps sending heartbeat signals and starts election on detection on the leader being dead.
Election thread	this thread is created when heartBeatthread receives a election message so that it can conduct election and can stop once election is done.

Message Types

MESSAGE_TYPE_CHAT	Its a chat message needed to be printed onto the console
MESSAGE_TYPE_ELECTION	Needs to hold election if it is alive
MESSAGE_TYPE_USERLIST	Message contains a list of users
MESSAGE_TYPE_STATUS_JOIN	Message to indicate a join request
MESSAGE_TYPE_STATUS_LEAVE	Message to indicate someone left or crashed .
MESSAGE_TYPE_CHAT_NOSEQ	Message to indicate its a chat message from a non-leader member in the chat group
MESSAGE_TYPE_UPDATE	Message to indicate we need to perform an update of the userlist
MESSAGE_TYPE_LEADER	Message to indicate we need to update the leader in the system.
MESSAGE_TYPE_REMOVE_USER	Message to indicate we need to remove a user from the system.

Message Sizes

```
#define ACK_MSGSIZE 16
#define MESSAGE_SIZE 2048
#define MAX_USERS 25
#define NUM_THREADS 10
#define IP_BUFSIZE 16
#define PORT_BUFSIZE 5
#define USERNAME_BUFSIZE 32
#define RXBYTE_BUFSIZE 32
```

Ports Being used

Chat Port	The port Allocaion starts at portnum 8980 and is allocated in steps of 10 if that port is not available
ACK Port	Chat Port +1 will be the port on which ACK messages are exchanged.
HeartBeat Port	Chat Port +2 will be the port on which HeartBeat signals are exchanged
HeartBearACK port	Chat port +3 will be the port on which heartbeat ack signals and election messages are being sent.

Election Algorithm

(Note: Rxbytes is the number of messages received by the machine before joining the group)

We use a modified version of bully algorithm to conduct elections in the chat system.

i. Initially the user who starts the chat is the leader.

ii. When the current leader dies, we determine the death of the leader on heartBeat signaling thread. Once this is detected the server who detects this changes his own state to conducting elections
iii. He sends out a message to everyone who has a higher rxBytes values in his list if some two members have the same amount of RxBytes then he compares on the portNumber.
iv. Once he receives a signal from any of the higher value member he makes sure that he is not the leader anymore but he continues sending election message so that some member who has not detected the leader is dead yet can start the election on getting this message this helps in speeding up the detection of the leader being dead.
v. Once the leader is elected he sends out a message to everyone in the group with type MESSAGE_TYPE_LEADER so that can update all their lists where they are keeping track of the users in the chat group.
vi. the leader flushes out the holdback Q that he currently holds once he is elected the leader and all the other users in the chat group flush their holdback Q as well once they receive a message of type leader.
vii. There is a mutex lock across the conduct election which makes sure two threads do not start election when one person is already conducting election.

HeartBeat Thread
i. HeartBeatsend thread sends signals are sent out by both the sequencer and the users present in the chat group.
ii. These signals are sent out every 5 seconds.
iii. If the heartbeat thread receives the signal and checks if the last signal received from that paricular user is less than 10 seconds :

a. if I am a user I declare the sequencer is dead and conduct elections
b. If I am a sequencer I declare the user is dead and declare that he has left the group to everyone in the chat group.

Handling Message Losses and Message Duplication

i. When a new message is created we create a unique ID using the boost UUID library and we add it to a map which maps the uid to a flag which is true if it got its own message back or is false if it did not get it.
ii. When an election happens and the message is lost I send all the messages that are currently in my send Q and then re populate my Send Q with messages which I did not received yet.
iii. If i receive the same message twice I discard it by setting because the flag is set to true when i receive the message the first time.

Encryption and Decryption of Message.

we Encrypt and decrypt all the Chat messages which are being sent using Caesar Cipher encryption.It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. We support Encryption and Decryption of only ASCII values since we are doing (message+5)%255.