Milestone 2

Group Members : Anand Sriramulu Arpit Panwar Jitesh Gupta

MESSAGE details :

#define	ID	Details
MESSAGE_TYPE_CHAT	1	Indicates its a chat message
MESSAGE_TYPE_ELECTION	2	Indicates its a election message
MESSAGE_TYPE_USERLIST	3	Indicates the message contains the list of users in the chat group
MESSAGE_TYPE_STATUS	4	This indicates its a status message that is being passed around in the group
MESSAGE_TYPE_STATUS_JOIN	5	Indicates its a message which contains someone has joined the group
MESSAGE_TYPE_STATUS_LEAVE	6	Indicates its a message that someone has left the group
MESSAGE_TYPE_STATUS_PING	7	This is to check if the sequencer is alive
MESSAGE_TYPE_STATUS_PONG	8	This is to indicate the chat member is alive.

Data structures :	All the data structures of type Queue which has been implemented in such a way that they are blocking and are thread safe.
list<sockaddr_in> listOfUsers;	This is a listofUsers in the chatgroup along with their ipaddress , This is used by the sequencer to send messages to all members of the group by iterating over the list.
Queue<message> holdbackQueue;	This queue is used to put messages that are received over the network so that they can be printed onto the console according to the sequence number and we can handle message reordering here.
Queue<message> chatQueue;	This is to make sure all the messages which are normal chat messages go into this queue.This will be useful for giving priority to status messages over chat messages in the future.
Queue<message> statusQueue;	This is to put all the chat messages in the queue, this is to make sure we seperate them from the normal chat messages that are happening.
Queue<message> consoleQueue;	This is to queue all the input from the user which is entered via the console and also to put the messages which are coming as a part of the chat group to the sequencer. so that they can be processed and added a sequence number later.
Queue<message> sendQueue;	This is to send the message to all the users of the chat in the chat group or only to the sequencer if it is not the sequencer.
Queue<message> ackQueue;	Queue to handle all the ACK messages that are being passed around in the chatgroup.
Queue<string> printQueue;	This is to just print the final output into the console.

Structures	Members

MESSAGE	char sContent[2048];
	int sType;
	int lSequenceNums;

LEADER	string sIpAddress;
	int sPort;
	string sName;

Assumptions :

1. UserName of size 32
2. Messages being entered in the chat 1024 bytes

Client Creating A chat Room :
i. Creates the socket connection and listens on the port for any incoming message
ii. Adds himself to the Map userMap
iii. sets himself as the leader of the chat room by default.

Client Joining the chat room :
i. Requests to join the chat room by mentioning the ip address:portnumber of the Chat leader :

a. if the ipaddress:portnumber is the same as the leader (by checking isleader variable) then he is added to the user list by the leader.
b. The leader sends a STATUS message to every member present in the group by mentioning that a new user has joined the group.
c. The leader sends a copy of all the list of candidates in the group via a LIST message.
d. Each user on receiving the STATUS message will add them to their own USER List.
e. Sends a ACK to the leader stating that the user was added successfully.

ii. Requests to join the chat room by mentioning the ip address:port number of the Chat member:

a. The join request is forwarded to the leader as a STATUS message by the user whose ipaddress:portnumber was mentioned by the new user.
b. He adds the new user to his own Userlist map. The new user is always added to the list by the leader (the new user does not add himself to the list only other than the case when the new user is also starting a chat) .
c. The leader on receiving the status message contacts the new member by sending him the list of candidates via a LIST message.

    d. he sends a message to everyone stating that a new user has been added to the group and in turn all the users add the new user into their own group.

iii. Chat messages enter at the user end in the chat which has to be sent to the sequencer.

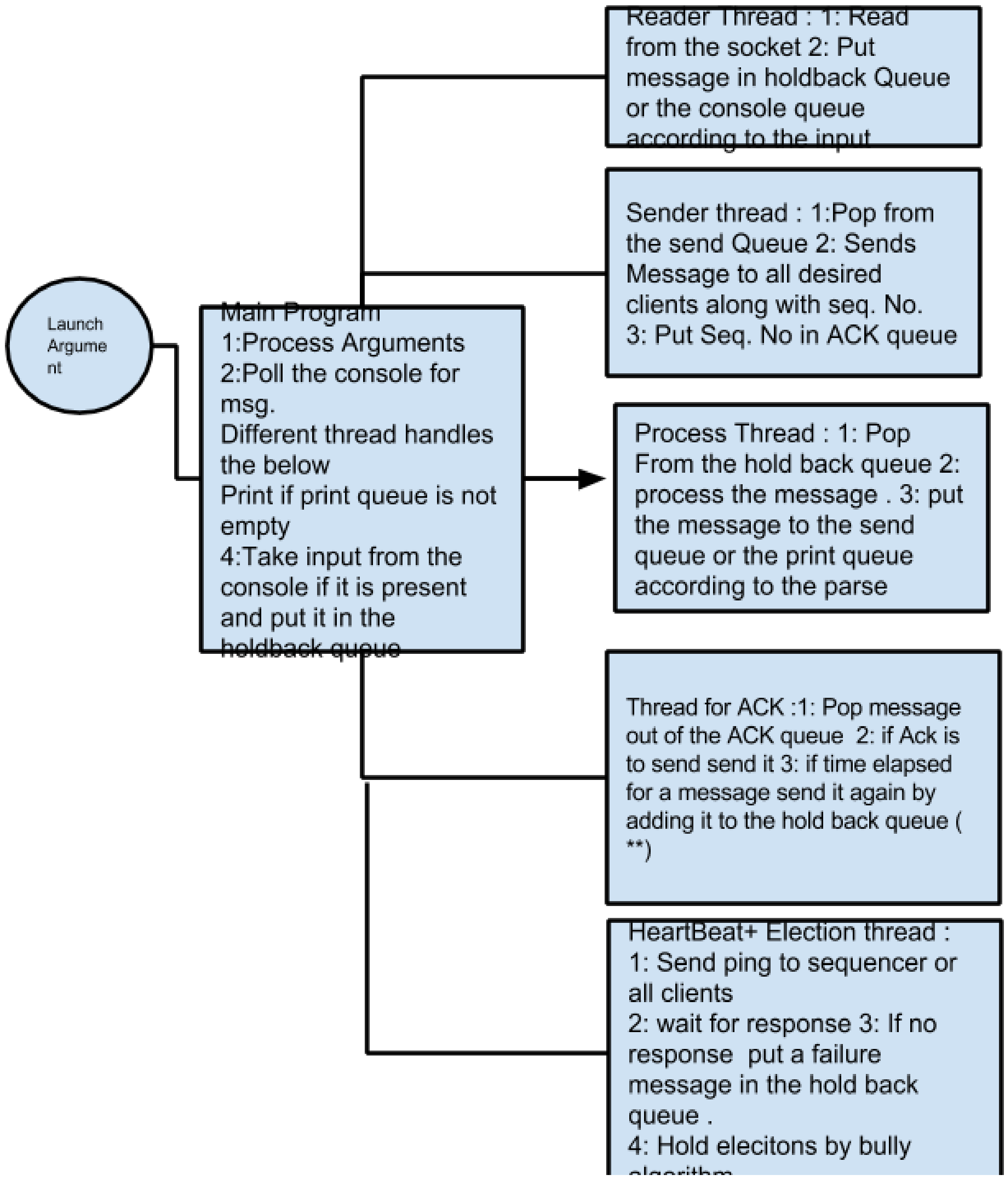
a. The terminal input from the user is inputed into a send queue .
b. From the send queue the message is sent to the sequencer using the UDP protocol.
c. Waits for an ACK signal from the Sequencer that the message was received then the next message in the send queue is sent out or else the same message is retransmitted.

iv. Receiving messages from the sequencer.

a. The messages received from the sequencer is put in the holdback queue.
b. These messages have the sequence number in the message which is parsed and put in the Message structure according to the message type.
c. And according to the sequence number the messages are sent out to the print queue. This point we also make sure there is no message reordering happening by making sure the messages sent out from the holdback queue are in accordance with the sequence number.
d. The print queue takes the sequenced messages from the holdback queue and prints to the console.

v. Sequencer handling messages coming from the users :

a. the sequencer puts a sequence number in order in which it receives messages from the chat group. we make sure that the messages that are being received from the group gets lower priority over the messages being entered by the sequencer himself.
b. the sequencer send the messages to everyone in the group .
c. sequencer waits for ACK from everyone in the group by checking if it received an ACK signal from all the users by checking the list successAckList.
d. Only when it has received a ACK from all the users only then it sends out the next message in the sequence.



Main Program
1:Process Arguments
2:Poll the console for msg.
Different thread handles the below
Print if print queue is not empty
4:Take input from the console if it is present and put it in the holdback queue

Reader Thread : 1: Read from the socket 2: Put message in holdback Queue or the console queue according to the input

Sender thread : 1:Pop from the send Queue 2: Sends Message to all desired clients along with seq. No.
3: Put Seq. No in ACK queue

Process Thread : 1: Pop From the hold back queue 2: process the message . 3: put the message to the send queue or the print queue according to the parse

Thread for ACK :1: Pop message out of the ACK queue 2: if Ack is to send send it 3: if time elapsed for a message send it again by adding it to the hold back queue (**)

HeartBeat+ Election thread :
1: Send ping to sequencer or all clients
2: wait for response 3: If no response put a failure message in the hold back queue .
4: Hold elecitons by bully algorithm

Duties :
We will following pair programming for the whole project at any given point we will make sure at least two of use meet and do the work. The design for the project was done over couple of sessions where all of us met. The testing of the project will also involve each one of us writing necessary scripts or testing as necessary. Extra credit work will be further divided after the initial basic functionalities is working.

Anand Sriramulu	Basic functionalities of the project which involves messages being exchanged in the group with only the user who creates the chat as the sequencer.
Jitesh Gupta	Handles the case where the sequencer will fail and conduct a election. Implementation of the election algorithm.
Arpit Panwar	Managing the Acknowledgment being sent and the heartbeat signals being exchanged to make sure everything is working accordingly.

Time Frame

March 28th	Finalize the protocol and the basic outline of the program along with the data structures required and the number of threads that will be implemented
April 4th	Basic functional UDP chat model without support for the sequencer to be crashing
April 8th	Implementation of a election algorithm and supporting various error and corner cases.
April 12th	Testing and debugging and make sure all the necessary functionality of the chat service is maintained.
April 15th	Extra credit implementation of traffic control and fair queuing . We are planning

	on integrating extra credits of 5.3,5.4 and 5.5 as part of our initial design itself.
April 20th	If possible implement decentralized total ordering.