CIS 505 Project 3 Milestone 2

Design Specification

I. Basic Group Chat Functionality

1. Start a New Chat

User can use dchat USER to create a group chat, and if the chat starts successfully, the user will become the leader and the user's client will be listening for messages on the IP address and UDP port.

2. Join an Existing Chat

dchat USER ADDR:PORT will join the USER to an existing chat, in which a current member is listening on IP address ADDR and port PORT. This command will send a JOIN message to the member. If this current member is not the leader, the member will tell user the IP and port of leader, then the user will contact leader directly and send a JOIN message. After leader receives such a message, he will add this new user to the current users list and multicast this new user's join (by a NEW USER message) to every other client in the chat.

After receiving a NEW USER message, every client will save this new user's information locally. And leader will send to this new user the information about current users in the chat by USER LIST message for him to save on his local machine, including users' IP address, port number, join time and number of messages already sent.

We assume that the leader or the connected member is active at this moment. So when there's no response in a period of time, we can conclude that there are no active clients in the chat, and then just tell the user the chat is inactive.

3. Global Ordering Chat Messages

Leader is acting as a sequencer to ensure totally ordered multicast.
Every chat message from clients is sent to the leader first, and leader will reply an OK message to confirm that it has received this message. If the sender has not received this OK message, it will send his message again. After sending message for three times without response, we can assume that the leader has crashed and this client will send an ELECTION message to all other clients to invoke a leader election process.

After receiving chat messages from clients, the leader will assign a sequence number (1, 2, 3, …) to every message based on the timestamp it receives the message and then multicast it among the group.

As for other clients, they will display the received messages in the order of sequence numbers. If a client realizes that a previous message is missing, it will put the newly received message in the hold-back queue until all previous ones are delivered (that is, displayed on his screen). And if it hasn't received the particular message for a period a time, it will send a REQUEST message to leader asking leader to send this message to it again.

4. Leave or Crash

We'll create a thread to send messages between leader and clients every 10 seconds to see whether they are still active or not. If the leader or a client doesn't reply for three times consecutively, we will assume that it has died.
If a user is inactive, the leader will multicast a LEAVE message to all other clients, and then all of them will remove this user from the current users list.

II. Leader Election

1. Assumptions

1) Centralized Solution

A central sequencer will be chosen from among the current group members, and will be responsible for sequencing messages, multicasting chatting messages. The user who starts the chat will serve as the sequencer at first.

2) Server Crash

If any member of the current group sends a message to the current server for three times and has received no reply from current server, this will be recognized as "current server crash".

3) Election Criteria

When the current leader leaves the chatting group due to personal or network reasons, new leader will be chosen among remaining members and who has sent the most messages in this group so far will become the new leader.

2. Election Algorithm

Suppose a process P detects a failure/crash of the current leader:

- P broadcasts an ELECTION message to all other processes to tells them about "current server crash", and indicates that they need a new leader.
- Since data records are stored at each client of the group, everybody knows who has sent the most messages in this group, and then everybody knows who will be the next leader.
- Everybody replies an OK to P (the finder) indicating that they have already known the leader is crashed.
- P waits for the new leader's reply:

  If P receives the new leader's OK reply,

  P will wait a certain amount of time for that new leader to broadcast itself as the new leader;

  else if P doesn't get the new leader's reply,

  P will send ELECTION message to the new leader as a reminder;

  if P doesn't get the new leader's reply after three times,

  P will multicast "ELECTION" again, and look for the next new leader, who will be the one that sent the most messages in this group except the crashed new leader, then P will loop above steps again to wait for the replies.
- New leader broadcasts itself as the new.
- New leader waits for all other processes' replies:

  If new leader has received all other processes' replies,

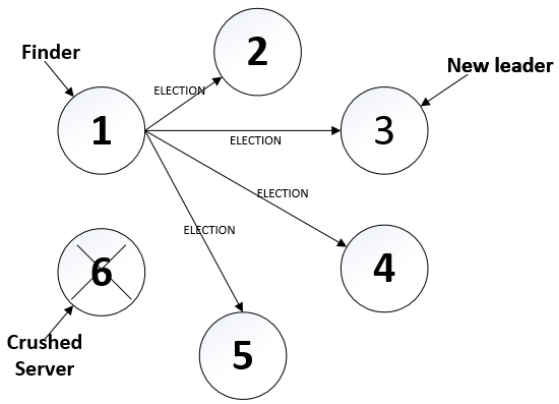
  Election ended;

  else

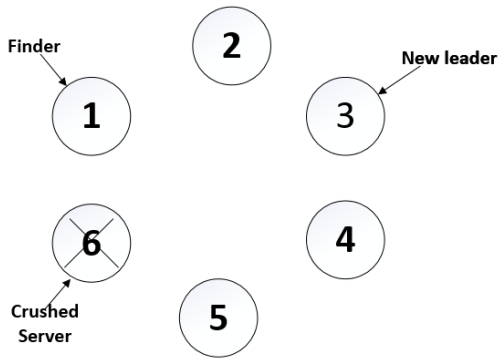
  New leader sends "NEW LEADER" message to those processes that didn't reply.

3. Election algorithm example

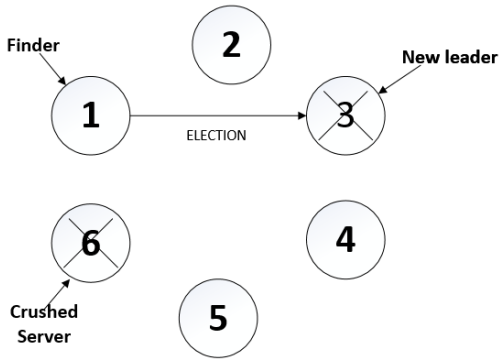
1) Suppose Client 1 finds that the current leader crashed, then it multicasts ELECTION to other members. Let's say Client 6 is the current leader.



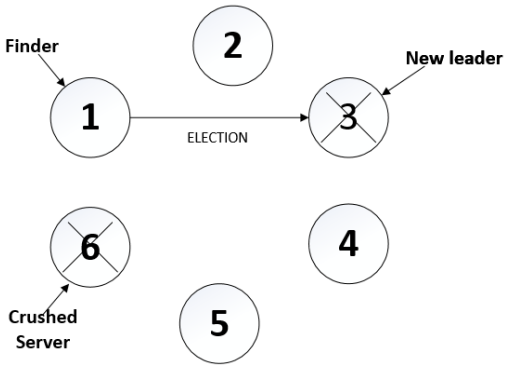
2) Suppose Client 3 has sent the most messages in this group so far, then everybody remaining in this group knows that Client 3 is the new leader. Client 1 waits for reply from Client 3.



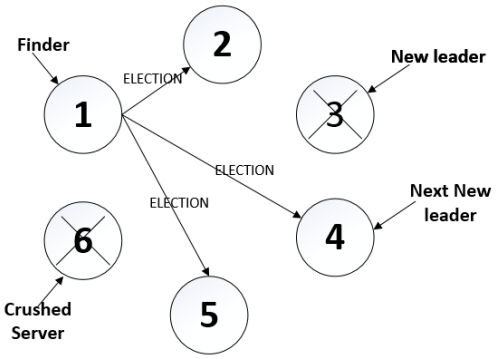
3) Client 3 crashed and doesn't reply Client 1, and then Client 1 waits until timeout, and sends ELECTION message the second time to Client 3 to inform that Client 3 is the new leader.



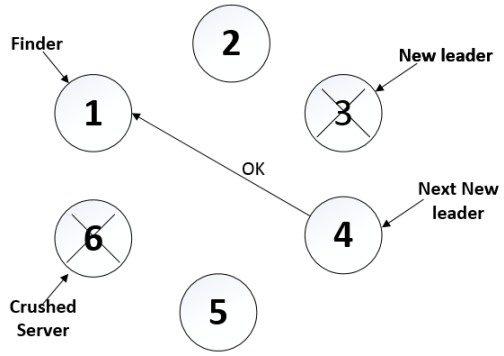
4) Client 3 crashed and doesn't reply to Client 1 the second time.
5) Client 1 waits until timeout, and sends ELECTION message the third time to Client 3.



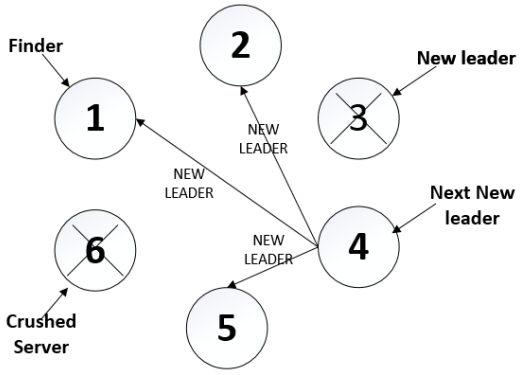
6) Client 3 crashed and doesn't reply to Process 1 the third time.
7) Client 1 hasn't received reply from Client 3 for three times and assumes that Client 3 died and multicasts ELECTION to remaining members to find the next new leader.



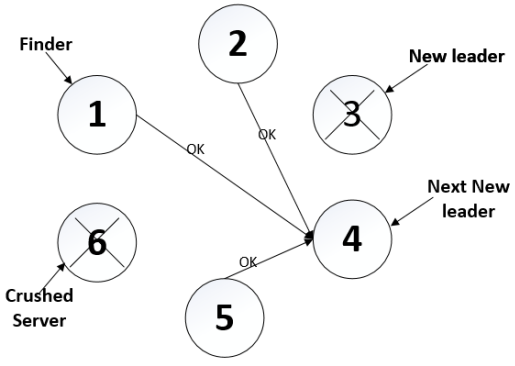
8) Client 4 is the next new leader, and it replies to the finder. (Assume that Client 4 has sent the most messages in this group except Client 3.)



9) Client 4 broadcasts itself as New Leader, and waits for replies.



10) All remaining clients reply OK to Client 4, and then Election ends.



III. Error Detection

1. Lost Messages

There are three kinds of message lost. First is when the sender sends a message to the receiver and the message lost before the receiver receives. Second is the receiver has some network errors so that it cannot receive message. Third is after the receiver receives the message successfully, but the sender doesn't receive the confirmation message from the receiver. It is hard to distinguish these three kinds of lost. We use timeout mechanism on sender. If the sender doesn't receive the confirmation message from the receiver in a certain time, the sender will send the message again. But if it still doesn't receive confirmation message from the receiver in the certain time, we assume that the receiver is out of network. In this situation, we think receiver has left the group. Next time, when the receiver is in the network again, it needs to send join request.

2. Re-ordered Messages

The leader has assigned a sequence number for each incoming message based on the message arrive order and sent these messages along with the sequence number to the members. The members can re-order these messages based on the sequence number and display it in order. If there are some sequence numbers missing after ordering, the member will not display any message which has bigger sequence number than the missing ones. For example, if a member receives the message with sequence number 8, but it hasn't received the message with sequence number 7. The member side will wait for a certain time. After receiving the message with sequence number 7 and displaying it, the message with sequence number 8 will be displayed. If it still hasn't received the message after the certain time, the member will send a message to the leader to request the message with sequence 7.

3. Duplicated Messages

Every message that a sender sent includes sender name, message content and send time on sender machine. If the receiver receives two or more messages with the same sender name, the same content and the same sender time, we think these messages are duplicated. The receiver only deals with one of them and ignore others.

4. Sender Network Error – Disconnection

If sendto() function returns -1, it means that the sender has local network errors. When it is back into network, it will send JOIN message again.

IV. Data Structure & Message Format

1. Data Structure

We define three kinds of data structures to store at each client:

1) Message: {sequence number, sender name, content}.
2) Member: {name, IP, port, join time, number of messages}.
3) Leader: {name, IP: port}.

2. Message Type and Format

Here are the message types and formats we'll use:

Type	Number	Name	Content
JOIN	NULL	New user's name	New user's port number
NEW USER	Message #	New user's name	New user's IP & port number
USER LIST	NULL	NULL	All users' information
CHAT	Sequence #	Sender's name	Message
ELECTION	Message #	NULL	NULL/NEW LEADER CRASH
LEAVE	Message #	User's name	NULL
OK	NULL	NULL	NULL
REQUEST	NULL	NULL	Sequence # of missing message
NEW LEADER	Message #	New leader's name	NULL

Conditions of each message type:

1) JOIN: New user sends to leader to join the group.
2) NEW USER: Leader sends to all other clients to inform of the new user's join.
3) USER LIST: Leader sends to the new user the information of current users.
4) CHAT: Leader sends to clients the chat messages among the group.
5) ELECTION: Finder broadcasts to invoke new leader election. If the new leader crashes, the content will be NEW LEADER CRASH, then the leader will be elected among all clients except the old and the new leader.
6) LEAVE: Leader multicasts about a user's leave or crash.
7) OK: Leader replies to client who sends a message to him, or clients reply to finder to confirm the reception of ELECTION.
8) REQUEST: A client sends to leader to request a particular message.
9) NEW LEADER: New leader multicasts to all clients.

Message number and sequence number are two different kinds of order numbers. Sequence number is only assigned to chat messages, and message number is used to record the ordering of multicast messages, including notification messages and confirmation messages to NEW LEADER.