CIS505 Final Project Report

name	pennkey
Yang Cao	yangcao
Yuexi Ma	yuexi
Muruo Liu	muruoliu

1. Design Decision

This project aims to implement a chat program based on the distributed multicast protocol we designed. The underlying network protocol is UDP, which is an unreliable unicast datagram delivery protocol. The basic algorithm regarding sending messages is illustrated as follows.

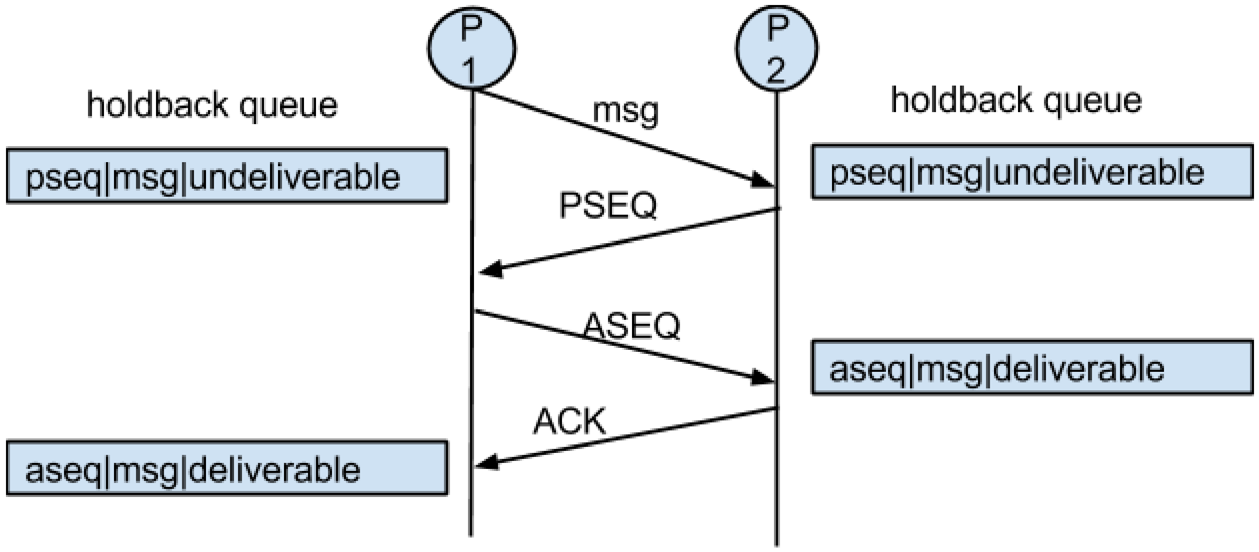

Figure 1. delivery of a total ordering message

When a process P1 wants to multicast a message to members of the group. It sends a DATA message to every other process in the group. The process that received this message will propose a sequence number PSEQ and send it back to the sender. After receiving all proposed numbers, the sender generates an agreed number and, again, sends it to every other member of the group. The process that received this ASEQ message will update their hold back queue. And if the message at the head of the queue is marked deliverable, it will be delivered immediately.

We choose to design a decentralized multicast protocol. Since there is no sequencer in the group, the total ordering should be maintained with more traffic. This is a trade off between network traffic and bottleneck of the sequencer.

2. Limits

Note that there are limits.

First, the name of the user can not be more than 25 characters.
Second, the number of simultaneous chat users can not be more than 200.
Third, the maximum message size is 10000 characters.

3. Protocol Analysis

UDP is unreliable. The protocol put lots of effort in handling network errors to make the system robust. Sometimes throughput and Latency are compromised to make the protocol more robust and maintain total ordering. This section states details about the protocol.

3.1 Messages Format

The message has the following structure:

message type	process id (ip:port)	message id	data
DATA	process id	message id	message data
PSEQ	process id	message id	proposed sequence number
ASEQ	process id	message id	agreed sequence number
ACK	process id	message id	
JOIN	process id	message id	
LIST	process id	message id	list of all processes (ip:port#name ip:port#name...)
NEW	process id	message id	proposed sequence number
LEAVE	process id	message id	
CHECKALIVE	process id	message id	
CONFIRM ALIVE	process id	message id	

3.2 Network Errors and Recovery

3.2.1 Message Loss

We use ACK messages to handle message losses. If the sender's thread did not receive the ACK, PSEQ, or ASEQ message it expects, it will resend the message in a fixed interval, which, in our case, is set to 500 msec. If after sending the message 15 times, it will think that the client has crashed.

3.2.2 Message Re-ordered

We use sequence number to maintain the order. There are two types of messages that need total ordering, which are NEW and DATA messages. When a NEW message is delivered, a client is considered to be added in the group. We maintain the order of messages about clients being added just like the normal chat messages. The algorithm is depicted in Figure 1 above. PSEQ is the proposed sequence number which equals to max(PSEQ,ASEQ)+1. And the sender of the message colloects all PSEQ from other clients and chose the biggest number as ASEQ, and multicast to them.

3.2.3 Duplicated Message

We record the messages we received in a map data structure. Before actually handle the received message, we first check if the message has already been received. The key of the map data structure is the 'type:ip:port:messageid' string. Duplicated messages are ignored unless they suggest a response is needed.

3.2.4 Client Crash

There are many issues that could lead to a crashes. To name a few, network errors, operating system issues, and a power off can all cause a crash. When a client crashed, we would like to notify other clients in the group. This is done through the hearbeat mechanism. Every client will periodically send hearbeat messages to other clients. If a client did not receive the heartbeat messages from a client, the cilent will be considered dead and be deleted from the member list. The client will also traverse the hold-back queue to find if there are pending messages of that client and ask other clients whether they have received ACK or ASEQ messages from the dead client.

4. Extra Credit

4.1 Message Priority

We devide messages into two classes, one with higher priority and one with lower priority. Messages regarding membership updates have a higher priority, so they are handled with a thread immediately. These messages include the PSEQ, ASEQ, ACK, LIST, JOIN, DIE. But NEW message, which contains the information about the new client, is put into the hold-back queue. That's because we would like the message "Someone just joined our chat group" is shown to all clients in the same order. So NEW and DATA messages are put into the same hold back queue and wait to be delivered at the appropriate time.

4.2 Encrypted chat messages

We use ASCII code rotation to encrypt the message.

Encryption function is:

x+ROTATE>ASCII?x+ROTATE-ASCII+START:x+ROTATE

Decryption function is:

x-ROTATE<START?x-ROTATE+ASCII-START:x-ROTATE

4.3 Decentralized total ordering

Instead of using the centralized sequencer, we use a decentralized total ordering algorithm, using the clients as participants. The elected leader is not used here. Details of the algorithm is illustrated in the Figure 1, the Design Decision section and the Protocol Analysis section.

5. User Manual

You can use 'make' to generate the executable 'dchat'.

5.1 Start a new chat

```
> dchat Bob
```

To start a chat, you need to specify your name. The system will generate an ip:port address for you.

5.2 Join a chat

```
> dchat Amy 10.0.2.15:45523
```

To join a chat, you need to specify the ip:port of one group member as well as your name. If you received no response from the member, you failed to join the group. Otherwise, you can see the list of the group members and start chatting.

5.3 Chat

```
> Hello
```

To chat with other group members, you can type in the command line, hit “enter” button, and the message will be multicast to all group members including yourself.

5.4 Exit

```
> EOF
```

To exit the group chat, you can hit “control D”.