DistChat - Demo 4

Team 7

### Change Log

* Removed all legacy material (pre-demo 3)
* Expanded on all aspects of the new design
* Added diagrams for visual representation of the system’s design
* Added formal protocol

### System Design

The system will use two different levels of interactions: Client-to-Server and Server Chain. DistChat is planned to be a closed system where all processes are made by the developers with no third-party applications. The system will be asynchronous meaning that all processes use a logical clock that is modified based on particular events; in this case receiving packets.

#### Client-to-Server:

Client employs UDP/IP to send messages (aimed at a particular chat room) using an ***add*** request to the server that will then place the message into a database using an ***insert*** command. Clients will become aware of what ***groupID*** to send to the server through ***create*** or ***join*** requests. ***Create*** allows the user to create a new chat room. ***Join*** on the other hand allows the user to join an existing chat. In both cases the server will either send an ***ack*** message to the client containing the chat room’s ***groupID*** or an ***error*** message.



After a client sends an ***add*** request, an ***ack*** will be sent from the server to the client to show that the message has been received. If no ***ack*** has been received from the server the client will resend the ***add*** request again after a defined timeout period.





Client will poll the server for new messages ***request*** and if there are any new messages then the server will send the messages to the client ***update***. ***msgID*** will be used to determine the state of the client in relation to the chatroom. So if Client has ***msgID*** = 4 in ***groupID*** = 1 while the database has ***msgID*** = 10 in ***groupID*** = 1, then we know that the Client is behind by 6 messages and will need to be updated. The polling is expected to be automated on a fixed time. It is not expected for the client to be responded to every poll.

###### Fault tolerance:

servers will simply not care if clients have fallen or have returned; their relationship is strictly transactional. Clients on the other hand will be provided a list of IP address of every server so if a client is unable to connect to one server they will simply move on to the next. If no server responds within a reasonable period of time, then the application will return an error statement in relation to a failure in connecting and re-attempt to communicate with the servers. Realistically getting the list of IP addresses would involve communicating with a DNS server, but for the purpose of this course the list of IP addresses may be hardcoded or provided in the command line

###### Client-To-Server Protocol:

<send> ::= <add> | <request> | <join> | <create>

<receive> ::= <update> | <ack> | <error>

<add> ::= “add”<newline><msg><groupID><newline>

<ack> ::= “ack”<newline><ackMessage><newline>

<ackMessage> ::= <msgAck> | <joinAck> | <createAck>

<msgAck> ::= <msg><groupID><msgID>

<joinAck> ::= <groupID>

<createAck> ::= <groupID>

<request> ::= “request”<newline><groupID><msgID><newline>

<join> ::= “join”<newline><groupName><newline>

<create> ::= “create”<newline><groupName><newline>

<error> ::= “error”<newline><errorMessage><newline>

<update> ::= “update”<newline><msgArray><groupID><newline>

<newline> ::= \r?\n

<msg> ::= String

<groupID> ::= int

<msgID> ::= int

<errorMessage> ::= String

<msgArray> ::= array of <msg> and <msgID>

The ***send*** format is for all messages sent from the client to the server while the ***receive*** format are for all messages sent from the server to the client.

#### Server Chain (Active Replication):

Servers will act as peers in this group that is managed by a group manager and a group communicator. Servers will receive messages from clients using the above Client-to-Server interaction. These messages will be assigned (by the client) to a particular chat room ***add(msg, groupID)***. The server will then add this message to its database. After, the server will IP-Multicast the message to its peers. The peers will then add the message to their respective databases. Employing IP-Multicast may result in some messages being lost, but this is considered an acceptable loss. Alternatively R-Multicast may be employed, but there is a significant bandwidth cost. As well, messages may not always appear in the same order i.e. server 1 may update its client with message 1 then message 2, but server 2 may update its client with message 2 then message 1 (causal ordering); this is considered acceptable.



###### Server Chain Fault Tolerance:

If a server falls, the group manager will reform the chain. The group manager learns of failure through keep alive requests that servers must respond to. Failure to respond within a specified timeframe will result in the group manager to order the failed server’s predecessor to change its message path from the failed server to the next server in the change. Ex. assume there are 4 servers where server 1 passes to server 2 who passes to server 3 whom passes messages to server 4 who finally passes messages to server 1 to finish the chain. If server 4 fails, then the group manager will send a message to server 3 to start passing its messages to server 1 reforming the chain.

###### Server Chain Protocol:

<send> ::= <add> | <passTO> | <stillAlive>

<receive> ::= <ack>

<add> ::= “add”<newline><msgArray><vectorTimeStamp><newline>

<passTo> ::= “passTo”<newline><ipAddress><newline>

<stillAlive> ::= “stillAlive”<newline><newline>

<newline> ::= \r?\n

<ack> ::= “ack”<newline><newline>

<msgArray> ::= array of <msg> and <msgID>

<msg> ::= String

<msgID> ::= int

<vectorTimeStamp> ::= vector of type int

<ipAddress> ::= IP address

The ***send*** format is employed by servers sending messages to their next peer while the ***receive***format is used by the groupManager to determine if a server is still active or not (there is still no guarantee that fault detection will always be successful due to the asynchronous nature of the system).