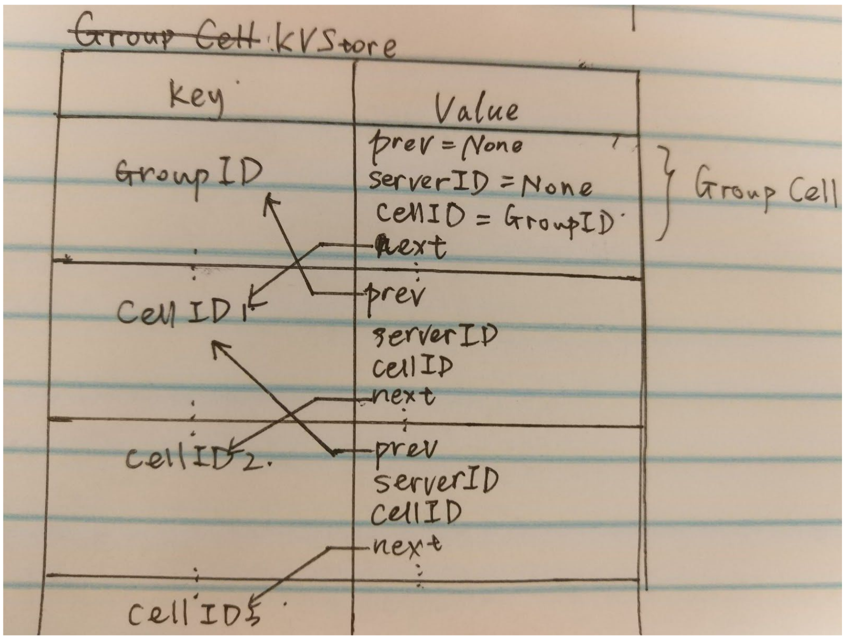
Project Report

System Design

1. Data Structure

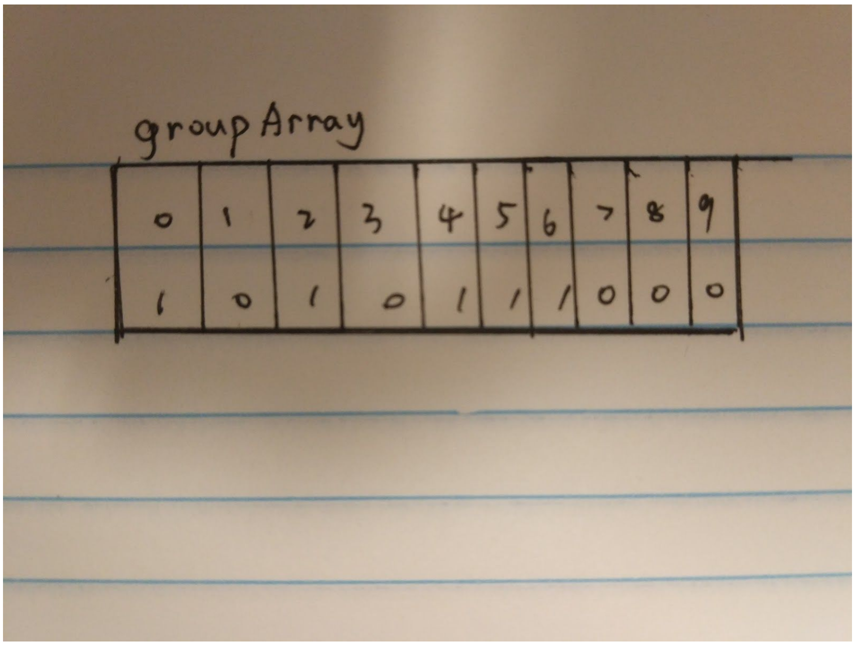


~~Group Cell~~ kVStore

key	Value

GroupID	prev = None
	serverID = None
	cellID = GroupID
	next

Cell ID1	prev serverID cellID next
cellID2.	prev serverID CellID next
cellID5	



groupArray

0	1	2	3	4	5	6	7	8	9
1	0	1	0	1	1	1	0	0	0

2. Message

There are 3 kinds of command()

After receiving Prime(), the group is initialized. We use to assign a group leader to each group.

33.3% of the probability that the command is to JoinGroup()

another 33.3% of the probability that the command is to LeaveGroup()

and the left 33.3% of the probability that the command is to SendMessage()

If an actor is told to JoinGroup():

It will first check the groupArray to see which group it's not in. If groupArray(i) == 0, then the actor is not in group i. Else if groupArray(i) == 1, then the actor is in the ith group.

So we randomly choose a i and see if groupArray(i) == 0. If so, we keep this i, else we try another i. Then, we tell the group leader to JoinG(), with the groupID of the group I choose and myNodeID. And, I will changeGroupArray(), to set groupArray(i) to 1.

If a leader is told to JoinG(), it will first writeJoin() to write the KVStore and then readJoin() to print who is in that group.
In writeJoin(), we first create a new GroupCell. We use groupIDtoBigInt() to generate an unused key with the nodeId in writeJoin()'s parameter.Then the group leader use groupID to locate the first node of this group in KVStore. Then, it will keep checking whether this node is the last actor in this group with the pointer "next" until it finds the last one. We then modify the next pointer of the last node and the prev pointer of the new GoupCell. We directWrite() both nodes into the KVStore.

The principle of LeaveGroup() is similar to JoinGroup()

If a leader is told to sendMulticast() to a group, the group leader will send multicast messages to its group member

Questions:
We have one group leader for each group. So we pass the operations to the group leader, and the operations on group members are all conducted by that group leader.

1. **Do actors ever receive messages originating from a given actor out of order? What if the messages are forwarded through an intermediary?**
No. Because each AKKA Actor has a mailbox, which receives messages from its own FIFO queue.
No. If A send a message to B and B forward it to C, the following is the process: A will first send the message to the group leader of A and B (A and B are in the same group), the group leader will then send the message to B. Then B will forward the message to the group leader of B and C, the group leader will send the message to C. In each step, the message is sent in order, so the result is in the same order.
2. **What if two actors multicast to the same group? Does each member receive the messages in the same order?**
Yes. According to our experiment design, if two actors A and B multicast to the same group, each actor will send only one message (to tell the group leader to send multicast messages) to the mailbox of the group leader in order, and the group leader will process the message in order of the arrival of the 2 messages, for example, A then B. So the group leader will first multicast A's message to its group members, then multicast B's message to its group members, so each member in the group will receive the messages in order of message A and message B.

3. **Do actors ever receive messages for groups "late", after having left the group?**
No. If a node want to leave one of his group, first it will send a message to his group leader to request for leaving, then the group leader process the message and delete the member from the group membership list. In addition, if one node wants to send a multicast message, it first send a message to the group leader, then the group leader process the message and send multicast message. So if a node has left a group, it means that leader process its leaving request message

before the multicast request message. So the node cannot receive multicast messages from this group any more.

4. **How does your choice of weights and parameters affect the amount of message traffic?**

When we increase the burst size, more operations (including read,write and send messages) are processed.

When we increase the TTL(maximum number of forward numbers per message), more multicasts are sent between servers.

5. **How can you be sure your code is working, i.e., the right actors are receiving the right messages?**

We print out a line for each message type the server received. Besides, we add variables into class Stats to record the message type and their numbers. We compare the number of messages sent and received by a group leader to make sure the code is working correctly.