Asynchronous Systems - Assignment 3

Muthukumar Suresh

1 Question 1

1.1

The algorithm only specifies that the that acceptors will eventually choose a value and that a learner will eventually learn a value. It talks nothing about the proposers.
Any proposer who proposes a lower proposal and it arrives late for more than half the acceptors then it is put in an infinite wait.

1.2

To avoid this, we can introduce :

1. Timeout - which if the proposer does not recieve a reply from more than a half acceptors, it timesout and restarts with a higher proposal.
2. Preemption - Acceptors help the proposers by sending a preempt message indicating a higher number msg has been received.
3. `http://www.slideshare.net/baotiao/paxos-38862630` talks about liveness lock, where each process tries to out propose each other.
4. Having multiple proposers (leaders) makes paxos need 2 rounds. To avoid this and the liveness lock, We have 1 leader and once the leader gets accepts from half the acceptors, it chooses that value. (page 9, "Paxos Made Simple")

2 Question 2

2.1

The main problem with paxos is that a lot of state information is kept at acceptors and leaders. As the paper says, some of the state information is necessary, e.g., the replicas need to maintain the state information for multiple slots and until all the slots have been decided the algorithm cant start a new round but [solution - after all slots have been decided, then the replica can discard the old state].

2.2

To solve the problem of acceptors and leaders, the following solution is proposed. The leader and acceptor maintain the decision for the slot for which they propose and accept respectively until the slot is decided and half of the replicas have learnt it then they can discard the state.

3 Question 3

3.1

Add timeout- I used the distalgo timeout function in elif of the proposer to skip await after a timeout. Preemption-acceptors send reset message to proposer to help proposer reset.

A driver process runs the 4 algorithms adn checks for liveness. Liveness condition:

- Only a value that has been proposed may be chosen, `Proposals send a message to the driver about the numbers proposed and we check if the chosen value is one of them.`
- Only a single value is chosen, `we see that all learners have the same value learned when the learners send to the driver.`
- A process never learns that a value has been chosen unless it actually has been. `We check for a match of the acceptor and learner values and see if they match in the driver.`

3.2

To test performance, we measure 2 things amount of process time used by acceptors, learners and proposers and the response time- time for the process to start and for it to finish.
Also, we measure these benchmark criterias by varying it against 3 parameters - message loss,message delay and wait time between rounds.
To implement the wait time between the rounds, we make sure that the proposer sleep for the time needed for the duration specified before trying another time.
To implement message delay, we make sure that before sending any message we sleep for the designated time.
To implement message loss, we make sure that we only send to a fraction of acceptors, simulating message loss.
Point to note: there are limitations to how accurately we can inplement these criterias. To test for a good set of values, we need to have high number of runs and varying a large number of parameters, but if we maintain a reliable channel distalgo will put forward an error too many files open and also the paxos implementation does not require reliable channel so we have chosen an unreliable channel.
So when we implement message delay and message loss, we are not factoring in the contributions made by the underlying network itself. So the artificially induced message loss needs to be taken with a grain of salt.

3.3 Implementation and running

To run type this huge command:
python -m da main.da p a l n r d w tp tl ml de wa
python -m da main.da 5 3 2 2 0.3 1 1 3 60 3 3 3
p - proposers
a - Acceptors
l - learners
n - number of rounds r - message loss fraction (0 - 1)
d - delay for message (seconds)
w - wait time for round
tp - time out for proposer in seconds
tl - timeout for learner
ml - number of values tested for message loss fraction (int)
de - number of values tested for delay messages(int)
wa - number of values tested for wait time between rounds(int)

3.4 result

You should see 6 graphs pop up and they each show the graph of varying 1 of the 3 parameters averaged oved the other two. I would suggest tinkering with the last 3 parameters 1ith 1 value high and other 2 low. e.g., 1 1 5 or 2 5 2. To get a good smooth graph.

You should see that sometimes huge spikes can be seen, when message loss and preemption takes place. Also, no timeout and no preemption seems to work well because the process are locked and are not contributing to the chaos of the leaders.
Overall the results are as epected where increasing message loss increases the time taken. Varying other parameters give varying effects depending on how the sysem as a whole behaves.

3.5 system specs

2.5 GHz i5 processor, Apple Mac 2012, 8 GB ram.