**SHIVANSHI GUPTA**

**112079392**

**Assignment – 4**

**QUESTION 1.**

**(1.1)**

1. Process P1 starts with Number N1 as the proposal and sends it to acceptors. Let’s call this PROPOSAL N1.
2. Meanwhile, another process P2 may send a proposal with Number N2>N1 to all acceptors. We assume that process1 faces some delay due to different locations of different nodes and latencies may be different. Let’s call this PROPOSAL N2.
3. Due to latency, the PROPOSAL N2 gets accepted and it completes its phase 1. And PROPOSAL N1 will get rejected as N2>N1.
4. Before Process P2 could go to phase 2, Process1 issues another proposal with N3>N2.
5. Process P1 gets the acceptance from the acceptors and complete its phase1.
6. Process P2 then performs phase 2 and gets rejected again.

This can go on for forever and the system will violate the liveness conditions. This is how the problem in liveness problems in the algorithm arises. Several competing proposers on the same paxos run can cause contention.

**(1.2)** Possible methods for solving liveness:

1. We can elect a distinguished proposer i.e. have only one proposer. It would be the only one which issues the proposal. If elected distinguished proposer communicates with the acceptors successfully, the protocol will surely guarantee liveness. Also, the proposer if uses the proposal number greater than the already used one, it will never be hindered as it will learn about the highest number with which it got abandoned/rejected and may try again with an even more higher number.

2. We can also create exponential back-off time so that exponential back-off will leave enough time for any of the proposals to go through entirely. This can make a process wait till another one gets its accept request completed and as the consensus would have reached the paxos run will be over.

**QUESTION 2.**

**(2.1**) The 2 problems we have are:

1. If there are multiple proposers, all wrking at the same time we risk them of having conflicts and some of them may have to restart. Liveness issues lead to proposers starting from the beginning and thus would eventually slow down the system. More conflicts would lead to more loading. If there are multiple clients which are concurrently talking to servers and all are trying to propose a value for the same log slots, the system would definitely crash.
2. The 2 RPC’s i.e “accept” and “prepare” for each value chosen lead to a lot load and may ultimately slow down the system.

**(2.2)** What we would do to solve performance problems?

1. We make a “LEADER” i.e we make a single server as the only proposer and all the client requests are pointed to that particular leader only.

HOW TO CHOOSE LEADER?

We select the server with highest ID as the leader. Every server will send a heartbeat message to every other server. If a server does not get any information from the server with higher id for sometime, it will appoint itself as leader and it will accept requests form client and act as proposer and acceptor both.

If the server is not the leader, it rejects the request and asks the client to talk to the leader. And the server just acts as acceptor.

1. We can eliminate Prepare RPC’s. What we can do is we will go with one round of prepare for the leader and it will prepare the whole log, not just a single entry. And once it is done, it can create many log entries just by using the accept round. It will not have to create prepare again. With this we almost half the number of RPC’s we need to issue.

HOW TO DO THAT?

We need proposal numbers to be sure that old proposal numbers are blocked out and also to find out any already chosen values. We can make the proposal number denote the whole log rather than just a single entry. If we do this, 1 round of prepares will lead to blocking of whole of the log and not just one entry.

The acceptor can also send “No more accepted” flag if no more proposals are accepted for log beyond the current one. For an acceptor giving out this flag, we don’t need to send anymore prepares to that acceptor. Once the leader gets “NO more accepted” from majority, there is no need for prepare RPC and just ACCEPT can be issued.