CSCI/ECEN 5673: Distributed Systems
Spring 2017
Homework 3
Due Date: 03/16/2017

Please submit a hardcopy of your answers in class on Thursday, March 16, 2017. BBA students may submit a PDF copy of their answers via the submission link provided on Moodle. Write your answers in the space provided. Please DO NOT use any extra space. The space provided is sufficient for answering the questions.

Topics covered: Replicated state machines, Totem, Paxos, Raft.
Lecture Sets: five, six, seven and eight.

Honor Code Pledge: On my honor, as a University of Colorado at Boulder student, I have neither given nor received unauthorized assistance on this work.

Name: Ishan Shah

1. In the Totem group communication system, explain the problem that occurs if two nodes fail in quick succession. How does Totem recover from this type of failure?

When a node fails (first failure), the processor invokes the protocol for the formation of a new ring. Processor collects info about operational processors and failed processors and broadcast that info in the proc_set and fail_set fields of a Join message. Upon receiving the Join Message from other participating processors, processor updates the its my_proc_set and m_fail_set.

If a second node fails in quick succession, other processors wouldn't receive a Join message from that node so they start broadcasting Join messages with the updated sets which shows that the second node that failed is moved to the m_fail_set from the my_proc_set. Consensus is met when my_proc_set and my_fail_set are equal to proc_set and fail_set of all Join messages from every processor i.e. they all agree on 2 failures.

Consider a distributed system with the following failure model (referred to as DS failure model):

(1) a node may suffer a crash failure;
(2) a node may send messages but not receive messages; and
(3) a node may receive messages but not send messages.

2. Explain how Raft will perform under DS failure model. What changes in Raft design will you make to tolerate failures with in the DS failure model?

Raft uses random timeouts. Each time a failure occurs, every server has a random timeout before it starts election which ensures that there won't be a split vote every time. Raft will take crash or message loss scenario as a failure and a new election or epoch will be initiated.

Now, to improve on the existing scheme for the DS failure model, we can do 2 things:

1) Keep counters on servers for this DS failure model that tracks the same message received. For example, if a server is failing to receive but keeps sending the same append request than counters on another server will timeout because it will keep receiving same request from the failed server so then the server that detected the same messages will start a new election.

2) Another way is to leverage 3 phase commit. Here the leader first sends replica to other servers and once it receives ACK from majority, it sends out commit message. We can add a pre-commit functionality using pre-commit message sent to majority before sending the commit message.

3. Can Paxos be improved under DS failure model? If it can be improved, explain how. If not, explain why not.

In my opinion, there can be some changes made to improve Paxos under the DS failure model. We can leverage the 3 phase commit idea for the Paxos implementation. To achieve that, instead of sending ("decide",v) to all learners, system tries to send it to at least k learners and then tries to send it to all learners.

Another way is to implement a checkpoint method. Let's assume that for every x increases in proposal version number n, the system goes into a global sync mode where, all the processes will come to a consensus regarding their command list. In this phase, instead of using quorum, system actually retries for y times to get a response from a failed process, and if it still doesn't get any response than that process would be eliminated from all future broadcasts. If the process wants to rejoin than it would have to go through the original joining process again. But, the 3 phase commit idea sounds more feasible because this solution would require more overhead and blocking than 3 phase commit would.