1

a)

If a process q gets a message and sends it directly to the application without reliably multicasting to the message to the group, then it atomicity is lost since the reliable multicast ensures that every non-faulty process has received the message. So without reliably multicasting to the group before sending to the application, it is possible that some non-faulty processes got the messages while others did not.

A possible scenario is that q sends the message to the application and then reliably multicasts it to the group. It is possible that a process p could have crashed between the time the message was sent to the application and when it was multicast to the group In this scenario, q sent the message to the application but p never received it.

b)

Totally ordered - Message is reliably multicast to group and each server stores messages in a queue whose messages are sorted by Lamport timetamp

FIFO - Reliably multicast the message to g, but at the receiving end, assign sequence numbers to the messages received so that order may be preserved

Causal - Start with FIFO multicast and use vector timestamps

2

a)

A candidate wins an election if it gets a majority of the votes from the servers in the cluster. Since a majority is required then there can be no ties so only one leader is elected.

b)

Scenario (e) could have happened if the server was leader for term 4 and got 2 commands and crashed before sending them, and was down for a few terms

Scenario (c) could have happened if the server was the leader for term 6 and then crashed after getting the command but before sending it to any of the servers

Scenario (d) could have happened if the server was leader for 7 and got 2 commands and crashed before sending them

c)

An entry is committed only after it has been replicated on a majority of the servers in a cluster. To win an election, a candidate must get votes from a majority of servers in the cluster and within this group of servers, at least one will have every committed entry.

d)

From step (3), “…the voter must have accepted the committed entry from leaderT before voting for LeaderU.

Step (4) then says that the voter still stored the entry (from leaderT) when it voted for leaderU, since every intervening leader contained the entry.

The assumption was that the Leader Completeness property did not hold, which means that all leaders after T do not have the entry.. However, those before and including leader do have the entry. The statement that every intervening leader contained the entry could mean those leaders up to and including T that do have the entry.

e)

The broadcast time (time to send RPCs in parallel) must be much less than the election timeout (the time a server will wait for communication from a leader), so leaders can send the heartbeat messages needed to keep elections from starting. The election timeout needs to be less than the mean time between failures so that the algorithm can proceed. If the election timeout is longer than the mean time between failures, then there will be a long wait after a leader failure and before the new leader is elected.

3

Project Proposal: A general, distributed Map-Reduce framework (Option #1)