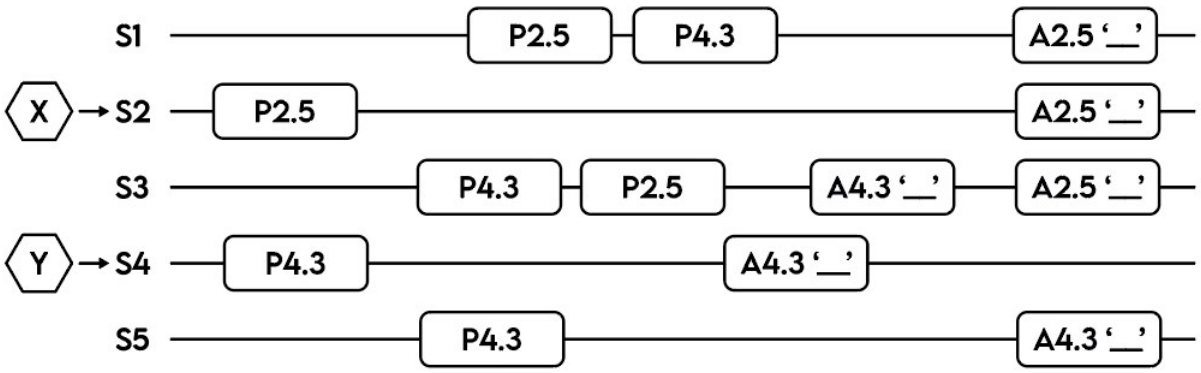
CS582 Distributed Systems

Quiz 4 Solution (Both variants were practically the same)

1. (6 points) Your TA Ahmed has developed a small Go program for simulating consensus in the Basic Paxos protocol running on five nodes: S1, S2, S3, S4, and S5. In his implementation, every Prepare and Accept message is stamped with a unique identifier: **ServerID.RoundNumber**. In terms of ordering, RoundNumber takes precedence, and then ServerID.

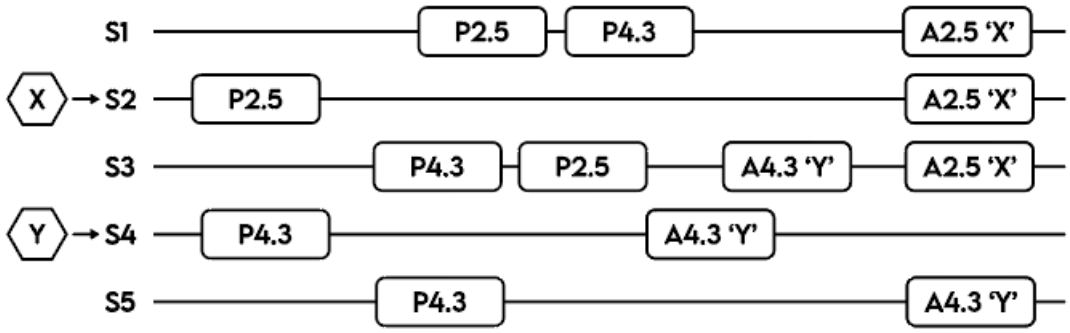
   Your TA Ayain runs his first simulation and observes the following:

   - **P2.5** means that a Prepare message was received by a server from 'S2' with round number '5'
   - **A4.3 'Y'** means that an Accept Message was received by a server from 'S4' with round number '3' containing value 'Y'



   (a) (3 points) Trace the remaining simulation and fill in the values that are received for acceptance at on the diagram.

   Solution:



   0.5 marks for each correct label

   (b) (1 point) Looking at the situation (as represented in the diagram), Ayain notices that most nodes agree on the same value. How will be remaining nodes learn of the agreed-upon value?

   Solution:
   When those servers next propose their own own value, their proposal will never be able to reach a majority of servers without at least reaching either S1, S2, or S3. In that case, S1, S2, or S3 will respond with the acceptedValue of 'X', leading to a global consensus.
   (No partial marking)

   (c) (2 points) Ayain runs the simulation again but this time notices that the program is not terminating (i.e. a consensus is not being reached). What could be the problem with Ahmed's implementation? How can he fix it?

   Solution:
   The protocol has livelocked **(1 mark)**
   Fixed by having a randomized delay before restarting so other proposers can finish choosing // Use a leader election algorithm to elect only one proposer to prevent competing proposers **(1 mark)**

2. (4 points) Danish and Bilal have been hired for building Meta's next global payment processing platform, MetaPay. The transactions involve updating balances in accounts, where it is critical to ensure consistency. In other words, either a transaction is fully committed (successful) or fully aborted (failed), with no partial updates.

   The system will operate under the following constraints:

   - **Node failures are possible** due to server crashes, network partitions, or other unpredictable events.
   - **The system must ensure strong consistency (linearizability)** even in the face of failures, preventing issues like double-spending or inconsistent account balances.
   - **The system must provide availability as much as possible**, minimizing downtime during failures or network issues.

   (a) (2 points) Bilal suggests that the Two-Phase Commit protocol can be deployed on all nodes with the node based in the United States being the coordinator. By using the Two-Phase Commit protocol, there would be no partial updates and all updates made would be linearizable. Do you think Bilal's proposal should be implemented? Explain.

   Solution:
   Bilal's proposal should not be implemented **(1 mark)**
   2PC can block if any one of the node goes down, leading to zero availability **(1 mark)**

   (b) (2 points) Danish suggests that the Basic Paxos protocol can be deployed on all nodes so that a transaction is committed on a majority of nodes. Stale reads can also be avoided by having a sufficiently large read quorum (i.e. reading from more than half the nodes and returning the most recent value) instead of directly reading from a user's nearest datacenter. Do you think Danish's proposal should be implemented? Explain.

   Solution:
   Danish's proposal should be implemented **(1 mark)**
   Paxos can tolerate node failures to a certain extent, ensuring that the service is relatively more available as long as a majority of the servers are up **(1 mark)**