IV Raft

Consider the Raft replication system described in *In Search of an Understandable Consensus Algorithm* by Ongaro and Ousterhout. Suppose you have a five-server Raft system. Here's the state of the servers' logs. The notation T.N means the Nth log entry from term T. The servers are about to choose a new leader.

```
S1: 3.1 4.1
S2: 3.1 3.2
S3: 3.1 5.1
S4: 3.1 5.1 5.2
S5: 3.1
```

9. [6 points]: Could any replica have already executed the operation in log entry 5.1? If yes, explain how this could have happened. If no, explain why it could not have happened.

Answer: No. An operation cannot execute if it is not in a majority of logs.

10. [6 points]: Could operation 3.2 be committed in the future? (Assume that the client does **not** re-transmit the operation.) If yes, how could that happen? If no, why not?

Answer: No. 3.2 can only be committed if S2 becomes the next leader (since any other leader will cause S2 to delete 3.2). S2 can't become a leader because there is no majority in which S2 has the most up-to-date log.

11. [6 points]: Could operation 4.1 be committed in the future? (Assume that the client does **not** re-transmit the operation.) If yes, how could that happen? If no, why not?

Answer: Yes. S1 can become the next leader with a majority from S1, S2, and S5.

12. [6 points]: A classmate points out that read-only operations could be executed much faster if the Raft leader alone executed them and sent the result back to the client, without sending such operations to the followers. After all, read-only operations don't modify the state, so executing them on the followers is a no-op. Explain why this idea would lead to incorrect behavior.

Answer: That would allow a leader in a network partition to continue to reply to client requests even after a new leader is elected. The old leader might then serve stale data, since it would not know about the latest write operations.

Imagine that you are using Raft to replicate a file server, so that the state being replicated is the file system stored on disk, and executing a client operation involves updating the file system on disk. Suppose a client operation has been committed to the Raft log in position 20. The leader has replied to the client, and the client never re-sends the request.

13. [6 points]: A follower crashes and re-starts. Could Raft give the operation in position 20 to the follower's state machine for execution twice, once before the crash and once after? That is, could the "apply log[lastApplied] to state machine" in the paper's Figure 2 Rules for Servers ever happen twice for the same log entry on the same replica? If yes, explain how that could happen. If no, explain why it couldn't happen.

Answer: Yes. Figure 2 says lastApplied is in volatile state, so the re-starting server won't know which operations it has already applied to its on-disk state.