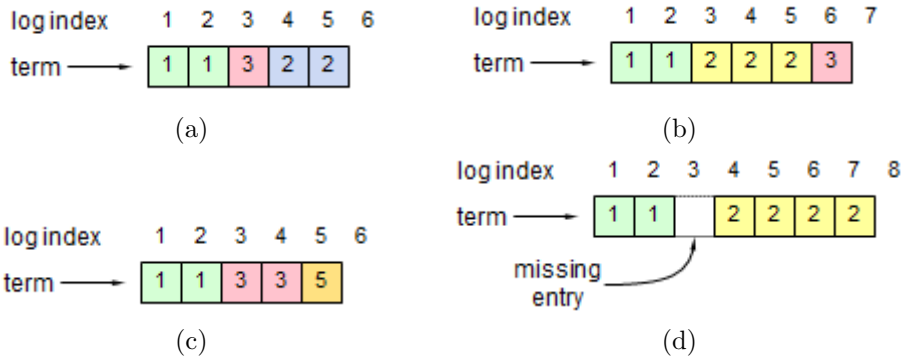
CS4224/CS5424 Quiz 8: Raft Consensus

Wk 9, Sem 1, 2023/24

All questions are based on Ongario and Ousterhout's Raft user study.

1. Each figure below shows a possible log configuration for a Raft server (the contents of log entries are not shown; just their indexes and terms). Considering each log in isolation, could that log configuration occur in a proper implementation of Raft? If the answer is "no," explain why not.



Solution:

(a) No: terms increase monotonically in a log. Specifically, the leader L that created entry (4,2) could only have received (3,3) from a leader with current term ≥ 3, so L's current term would also be ≥ 3. Then it could not have created (4,2).

(b) Yes.

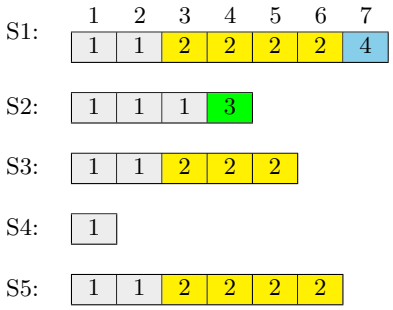
(c) Yes.

(d) No: logs cannot have holes. Specifically, leaders only append to their logs, and the consistency check in AppendEntries never matches a hole.

2. The figure below shows the state of the logs in a cluster of 5 servers (the contents of log entries are not shown; just their indexes and terms) where the current leader is S1.

   (a) Which of the servers could be elected as the new leader if S1 were to fail?

   (b) Which log entries will be present in the logs of all future leaders?



Solution:

(a) S2 and S5.

(b) Since entry (1,1) is present in all the logs, the entry will never be overwritten by any future leader; thus, (1,1) will be present in the logs of all future leaders.

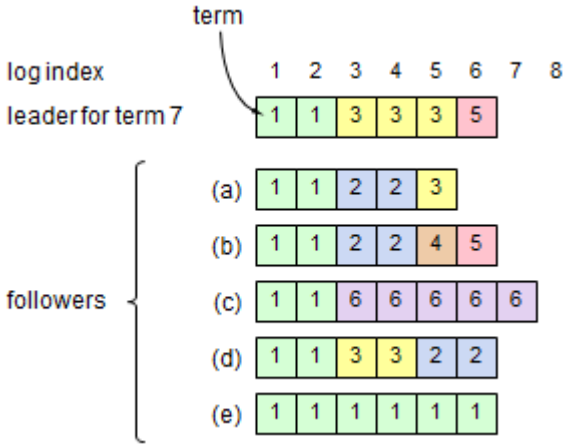
Observe that S4 can not be elected as the next leader should S1 fail as the logs of all the other servers are more complete than S4's.

Any future leader among $\{S1, S2, S3, S5\}$ will contain the entry (2,1). If S4 were to to be eventually elected as the leader in some future term T, the entry (2,1) must be present in S4's log by term $T-1$ in order for any of $\{S1, S2, S3, S5\}$ to vote for S4 to become leader in term T. Thus, (2,1) will also be present in the logs of all future leaders.

Note that even though the entries (3,2), (4,2) and (5,2) are present in a majority of the logs, they could be overwritten by some future leader. For example, S2 could be elected as the next leader after S1 and overwrite these three entries. Thus, the entries (3,2), (4,2) and (5,2) are not guaranteed to be in all future leaders' logs.

Therefore, only entries (1,1) and (2,1) are present in the logs of all leaders.

3. Consider the figure below, which displays the logs in a cluster of 6 servers just after a new leader has just been elected for term 7 (the contents of log entries are not shown; just their indexes and terms). For each of the followers in the figure, could the given log configuration occur in a properly functioning Raft system (i.e., wrt the leader's log)? If yes, describe how this could happen; if no, explain why it could not happen.



Solution:

a) No. Entry (5,3) uniquely identifies a log prefix (by the AppendEntries consistency check), but this follower has entry (5,3) and a different log prefix before it than the leader.

b) No. Entry (6,5) uniquely identifies a log prefix (by the AppendEntries consistency check), but this follower has entry (6,5) and a different log prefix before it than the leader.

c) Yes. Since we can't say much about the other servers in the cluster, this server could have been leader in term 6 with a starting log of (1,1), (2,1) and could have written a bunch of entries to its log and not communicated with the leader of term 7. This assumes that entries (3,3), (4,3), (5,3), and (6,5) were not committed in term 5, which is possible.

d) No. Terms increase monotonically in a log. Specifically, the leader L that created entry (5,2) could only have received (4,3) from a leader with current term ≥ 3, so L's current term would also be ≥ 3. Then it could not have created (5,2).

e) Yes. For example, (e) is the leader for term 1 and commits entries (1,1) and (2,1), then becomes partitioned from the other servers but continues processing client requests.

4. Suppose that a hardware or software error corrupts the nextIndex value stored by the leader for a particular follower. Could this compromise the safety of the system? Explain your answer briefly.

> **Solution:** No.
> If the nextIndex value is too small, the leader will send extra AppendEntries requests. Each will have no effect on the follower's log (they will pass the consistency check but not conflict with any entries in the follower's log or provide any entries to the follower that the follower didn't already have), and the successful response will indicate to the leader that it should increase its nextIndex.
> If the nextIndex value is too large, the leader will also send extra AppendEntries requests. The consistency check will fail on these, causing the follower to reject the request and the leader to decrement nextIndex and retry.
> Either way, this is safe behavior, as no critical state is modified in either case.

5. Suppose that you implemented Raft and deployed it with all servers in the same datacenter. Now suppose that you were going to deploy the system with each server in a different datacenter, spread over the world. What changes would you need to make, if any, in the wide-area version of Raft compared to the single-datacenter version, and why?

> **Solution:** We'd need to set the election timeouts higher: the expected broadcast time is higher, and the election timeout should be much higher than the broadcast time so that candidates have a chance to complete an election before timing out again. Similarly, the leader timeouts should also be increased. The rest of the algorithm does not require any changes, since it does not depend on timing.

6. Recall that each server stores three pieces of information on its disk: its current term, its most recent vote, and all of the log entries it has accepted.

   (a) Suppose that a follower F crashes, and when F restarts, its most recent vote has been lost. Is it safe for F to rejoin the cluster (assuming no modifications to the algorithm)? Explain your answer.

   (b) Now suppose that a follower F's log is truncated during a crash, losing some of the entries at the end. Is it safe for F to rejoin the cluster (assuming no modifications to the algorithm)? Explain your answer.

Solution:

(a) No. This could allow a server to vote twice in the same term, which would then allow multiple leaders per term, which breaks just about everything.
For example, with 3 servers (S1, S2 & S3):

- All three servers S1, S2 & S3 are initially followers in term 1.
- S1 & S2 timeout about the same time and each of them tries to elect itself as the leader for term 2.
- S3 receives S1's vote request and votes for S1.
- S1 acquires votes from S3 and itself to become the leader for term 2.
- S3 restarts and forgets that it had voted in term 2; S3 remains in term 1
- S3 receives S2's vote request and votes for S2
- S2 acquires votes from S3 and itself to become the second leader for term 2.

Now S1 and S2 can commit distinct entries in term 2 with the same index and terms but different values.

(b) No. This could allow a committed entry to not be stored on a majority of the servers, which would then allow some other entry to be committed for the same index.
For example, with 3 servers (S1, S2 & S3):

- S1 becomes leader in term 2 and appends index=1, term=2, value=X on itself and S2.
- S1 sets its commitIndex to 1 and returns to the client that X is committed.
- S2 restarts and loses the entry from its log.
- S3 (with an empty log) becomes leader in term 3, since its empty log is at least as complete as S2's.
- S3 appends index=1, term=3, value=Y on itself and S2.
- S3 sets its commitIndex to 1 and returns to the client that Y is committed.

7. It is possible for a leader to continue operating even after other servers have decided that it crashed and elected a new leader. The new leader will have contacted a majority of the cluster and updated their terms, so the old leader will step down as soon as it communicates with any of these servers. However, in the meantime it can continue to act as leader and issue requests to followers that have not yet been contacted by the new leader; furthermore, clients may continue to send requests to the old leader. We know that the old leader cannot commit any new log entries it receives after the election has completed, since it would need to contact at least one of the servers in the electing majority to do this. But, is it possible for an old leader to execute a successful AppendEntries RPC that completes the commitment of an old log entry that was received before the election started? If so, explain how this could happen, and discuss whether or not this will cause problems for the Raft protocol. If this cannot happen, explain why.

Solution: Yes. This can only happen if the new leader also contains the entry being committed, so it will not cause problems:

- Let L1 and L2 denote the old and new leaders, respectively.
- Let M denote the set of servers that have voted L2 to be the new leader.
- Clearly $L1 \notin M$ and $L2 \in M$.
- Let F denote the follower that L1 successfully appended its old entry E after the election of L2 as new leader.
- Since F did not reject L1's AppendEntries RPC, F must not have received L2's RequestVote and AppendEntries messsages. That is, $F \notin M$.
- For E to be committed after L1's successful AppendEntries RPC to F, L1 must have replicated E to a majority of servers after appending E to F's log. This implies that E must exist in at least one of the servers in M. For L2 to be elected by M, L2's log must be at least as complete as the log of each of the servers in M; thus, L2's log must contain E.

Here's a concrete example of this happening with 5 servers (S1 to S5):

- S1 with an empty log becomes leader for term 2 with votes from S1, S2, and S3.
- S1 completes appending index=1, term=2, value=X to itself and S2.
- The servers become partitioned into two clusters $\{S1, S3\}$ and $\{S2, S4, S5\}$.
- S2 with index=1, term=2, value=X in its log becomes leader for term 3 with votes from S2, S4, S5.
- S1 completes appending index=1, term=2, value=X to S3.

At this point, S1 has completed commitment of index=1, term=2, value=X, even though it is no longer the current leader.