IV Raft

Refer to Ongaro and Ousterhout's *In Search of an Understandable Consensus Algorithm (Extended Version).*

6. [7 points]: In the paper's Figure 8, at time (b), S5's last log entry has a term of 3. A friend of yours suggests that the term in S5's last log entry should instead be 2, since S5 got votes only from S3 and S4 (and itself), and those servers all have 1 at the ends of their logs. Explain what it is that forces S5 to be leader for term 3, and not term 2.

Answer: When S1 became leader for term 2 (at time (a)), it exchanged RPCs with at least a majority of peers, which caused those peers to set their currentTerm's to 2. This majority must have included at least one of S3, S4, or S5. S5 could not then become leader for term 2, because at least one server in whatever majority voted for it must have known that the latest term number was 2.

Suppose Figure 2's RequestVote handling code were changed to omit lastLogIndex and lastLogTerm from the Arguments, and item 2 were changed to:

```
2. If votedFor is candidateId,
   or (votedFor is null and term >= currentTerm),
   grant vote.
```

7. [7 points]: Explain why this change can cause two Raft peers to apply different entries at the same index.

Answer: With this change, a server whose log is missing committed/applied entries could become the leader. It will force other peers' logs to be the same its log, and thus cause the system to forget about committed operations. It could then receive, commit, and apply different entries at those same log indices.

Figure 13 describes a series of steps that the InstallSnapshot RPC handler must follow, including:

```
6. If existing log entry has same index and term as snapshot's last
   included entry, retain log entries following it and reply
```

Recall that Raft's safety guarantee is that if two servers commit an entry at a particular index, it must be the same entry.

8. [7 points]: Consider what would happen if this step were omitted (i.e. we unconditionally proceeded to step 7, discarding the entire log upon receiving an InstallSnapshot RPC). Would this modified protocol violate Raft's safety guarantees? If not, explain why not. If it would, describe an execution of the protocol that results in an observable safety violation.

Answer: Yes, it would violate Raft's safety guarantees. The discarded log entries may be committed, but known only to a bare majority. Discarding them might cause them to be known only to a minority. A new leader that lacked those entries could be elected without votes from that minority, causing committed log entries to be entirely forgotten, and different entries could be applied in their place on some peers.

Suppose that a correct Lab 2 implementation is running on a cluster of machines, and a cosmic ray flips some bits in the memory of the leader. This question asks about possible corruptions and how they affect Raft safety. Recall that Raft's safety guarantee is that if two servers commit an entry at a particular index, it must be the same entry.

9. [7 points]: Consider the scenario where the cosmic ray flips bits in the 'nextIndex[]' array, replacing some values with other arbitrary values. Assume that code handles out-of-bounds values by clipping to the size of the log. Could an execution where this occurs result in an observable safety violation? If not, explain why not. If it could, describe an execution of the protocol that results in an observable safety violation.

Answer: This won't cause a safety violation. If nextIndex is too low, the leader may send log entries that the follower already has, which the follower will ignore. If nextIndex is too high, this may cause the leader to not send new log entries to a follower, and thus perhaps not be able to commit new commands; this may hurt liveness but not safety.

10. [7 points]: Now, consider a scenario where the cosmic ray flips bits in the 'matchIndex[]' array, replacing some values with other arbitrary values. Assume that the code handles out-of-bounds values by clipping to the size of the log. Could an execution where this occurs result in an observable safety violation? If not, explain why not. If it could, describe an execution of the protocol that results in an observable safety violation.

Answer: This can cause a safety violation, by causing the leader to think that a majority of peers have stored a log entry when the really haven't. The leader may commit that entry; if it then crashes, the new leader may not be aware of the entry, and it could commit a different entry at the same index.