B Proof of Safety

SC1. Validity: *If a server decides on a log L then L only contains proposed commands.*

By inspection of the pseudo-code shown in Figure 4b, any decided command must come from either the `log` or the `buffer` of some server ①. And as seen in ⑥, the `buffer` and the `log` contains only commands from clients. Furthermore, the properties of the session-based FIFO perfect link do not invent any new commands. □

SC3. Integrity: *If a server decides on a log L and later decides on L' then L is a strict prefix of L'.*

The invariant P2c is central for the proof of SC3. Recall from §4.2 the invariant P2c:

> For any v and n, if a proposal with sequence v and number n is issued, then there is a majority-set S of acceptors such that sequence v is an extension to the sequence of the highest-numbered proposal less than n accepted by the acceptors in S.

Sequence Paxos maintains P2c with log synchronization in the Prepare phase. Any new leader must first collect a majority of promises to adopt the most updated log, which is guaranteed to include any chosen sequence. Only after adopting the most updated log and synchronizing it with the promised followers can a leader propose new commands that extend the log.

Proof by induction.
We are going to prove the property SC3 using induction on the sequence of decisions by any server. We denote the i^{th} decision by a server as $SC3(i)$.

<u>Base case:</u> Show that $SC3(1)$ holds, that is the property holds on the first decision of a server.

Initially, all servers have an empty log. And any log is an extension to the empty log. Thus, if any servers decide it will be an extension to the empty log and the base case is satisfied.

<u>Inductive step:</u> We want to show that for any server s if $SC3(i)$ holds for any $i < j$, then $SC3(j)$ also holds for s.

By the induction hypothesis for a particular $i < j$, $SC3(i)$ holds. Furthermore, we assume that log L was the i^{th} decision of a server s. We perform a case analysis on the role of s.

(1) Leader. We assume s is the leader of a round n. Since L is already decided in a lower round, it is guaranteed that at least one server among any majority has L as a prefix. In the Prepare phase ④, s adopts the log with the highest round number, denoted by L_{max}. Note that L_{max} includes the latest chosen log, since it is the most updated log in the majority. Given the invariant P2c, it is guaranteed that L_{max} must have L as a prefix. The leader s adopts L_{max} and extends it. Thus, any subsequent decision j by s in this round extends L_{max}, which in turn extends L and $SC3(j)$ holds.

(2) Follower in the Prepare phase. We assume s is a follower in round n and has received a ⟨PREPARE⟩ message with n from the leader. Since L is decided by s, the suffix in the ⟨PROMISE⟩ sent by s is guaranteed to include any possibly missing entries of L at the leader. Given this and the invariant P2c, the adopted log L_{max} at the leader must have L as a prefix. L_{max} is then adopted by s when handling ⟨ACCEPTSYNC⟩ ⑤. Given the FIFO property of the communication, any later accepted entries extends L_{max}. Thus, any subsequent decision j by s is an extension of L and $SC3(j)$ holds.

(3) Late follower after the Prepare phase. We assume s is a follower in round n and has received a ⟨PREPARE⟩ message with n from the leader. However, now we assume that the leader has already completed the Prepare phase in round n. In this case, L is already a prefix at the leader. Thus, what is sent in the ⟨ACCEPTSYNC⟩ and the following ⟨ACCEPT⟩ messages is guaranteed to extend L at s. Any subsequent decision j by s is an extension of L and $SC3(j)$ holds.

The property holds for all possible roles of a server, establishing the inductive step □.

C Correctness of Reconfiguration

A correct reconfiguration from c_i to c_{i+1} must preserve the invariants of Sequence Paxos (§4.2) across the different configurations. It is therefore crucial that no entry can be decided in c_i after a SS has been chosen. Thus, any new requests are dropped by the leader if SS is in the log, as seen in steps P5 of ④ and 1 of ⑥ in Figure 4b. Furthermore, a server can only start its Sequence Paxos component of c_{i+1} when it has all the log entries from c_i. Its log when c_{i+1} starts is therefore the same as when c_i stopped. As a result, a reconfiguration is just a continuation of Sequence Paxos where the round number conceptually increments from $r = (i,b)$ to $r = (i+1,b)$, where i is the configuration number and b is the ballot number provided by the leader election in respective configuration. Thus, the invariants of Sequence Paxos are preserved. □
SC2. Uniform Agreement: *For any two servers that decided logs L and L' respectively then one is the prefix of the other..*

Proof by contradiction.
Assume that L is not a prefix of L' and vice versa. Since both logs are decided, there are two majorities Q and Q' that have

decided L and L' respectively. As any two majorities intersect, there must be at least a server $s \in (Q \cap Q')$ that has decided both L and L'. However, as shown by the proof of SC3, s can only decide on a log incrementally. Thus, L must be a prefix of L' or vice versa, a contradiction. $\square$