CS 582: Distributed Systems

Raft (Cont'd)

Dr. Zafar Ayyub Qazi

Fall 2024

Acknowledgments: Slides use some material from John Ousterhout (Stanford University)

Specific learning outcomes

By the end of today's lecture, you should be able to:

- ❑ Describe how safety can get violated if Raft servers immediately switch to a new configuration
- ❑ Explain the two-phase process of joint consensus in Raft for configuration changes
- ❑ Analyze how Raft's joint consensus ensures Safety while allowing clients to make commits during configuration changes
- ❑ Analyze, evaluate, and compare Raft's performance
- ❑ Analyze how Raft can provide linearizability

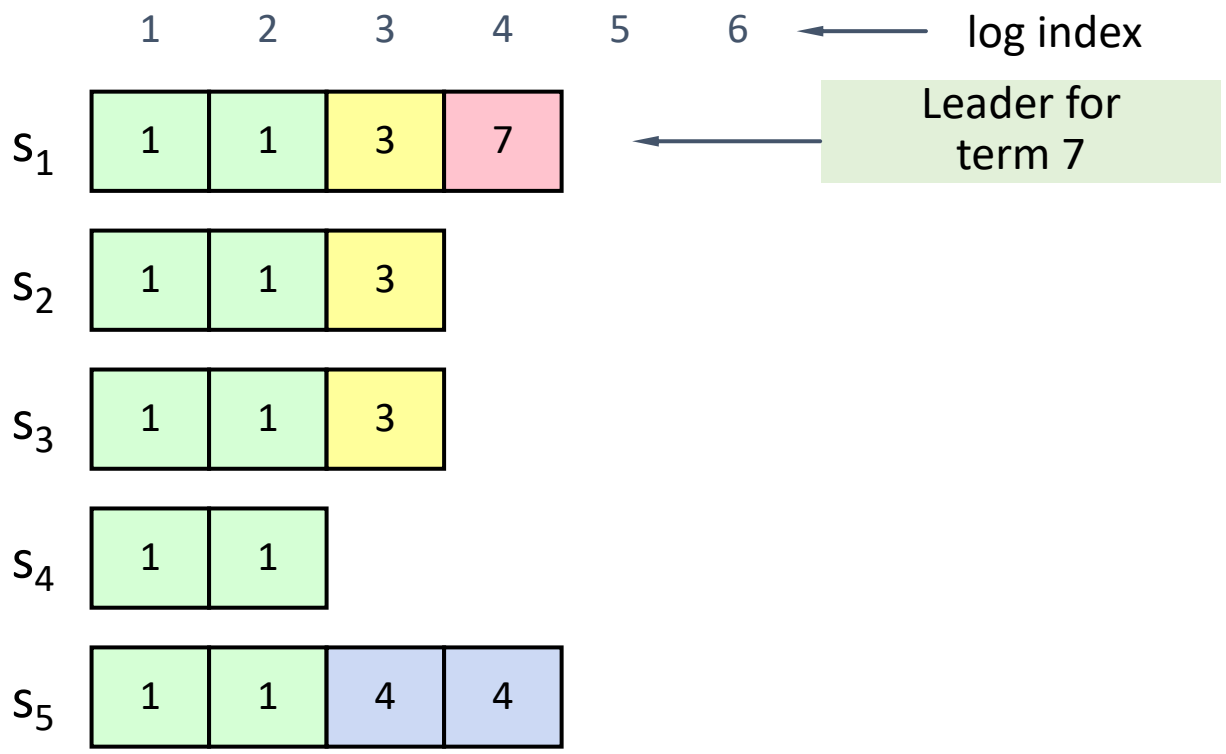
Raft Summary

1. Leader election
   - Select one of the servers to act as leader
   - Detect crashes, choose new leader
2. Normal operation (basic log replication)
3. Safety and consistency after leader changes
4. Neutralizing old leaders
5. Client interactions
6. Configuration changes:
   - Adding and removing servers

Raft Summary

1. ~~Leader election~~
   - ~~Select one of the servers to act as leader~~
   - ~~Detect crashes, choose new leader~~
2. ~~Normal operation (basic log replication)~~
3. ~~Safety and consistency after leader changes~~
4. ~~Neutralizing old leaders~~
5. ~~Client interactions~~
6. Configuration changes:
   - Adding and removing servers

Example


1
2
3
4
5
6
log index
Leader for term 7
s_1
1
1
3
7
s_2
1
1
3
s_3
1
1
3
s_4
1
1
s_5
1
1
4
4

Cluster Membership Changes

- Up until now, we have assumed a fixed configuration
  - The set of servers participating in the consensus algorithm is fixed

- In practice, it will occasionally be necessary to change configuration
  - For example, to replace servers when they fail or
  - Change the degree of replication

- This could be done manually (bring down the cluster and change configuration files), however,
  - This could leave the cluster unavailable during the changeover
  - And in general, manual steps, risk operator error

Automating Configuration Changes

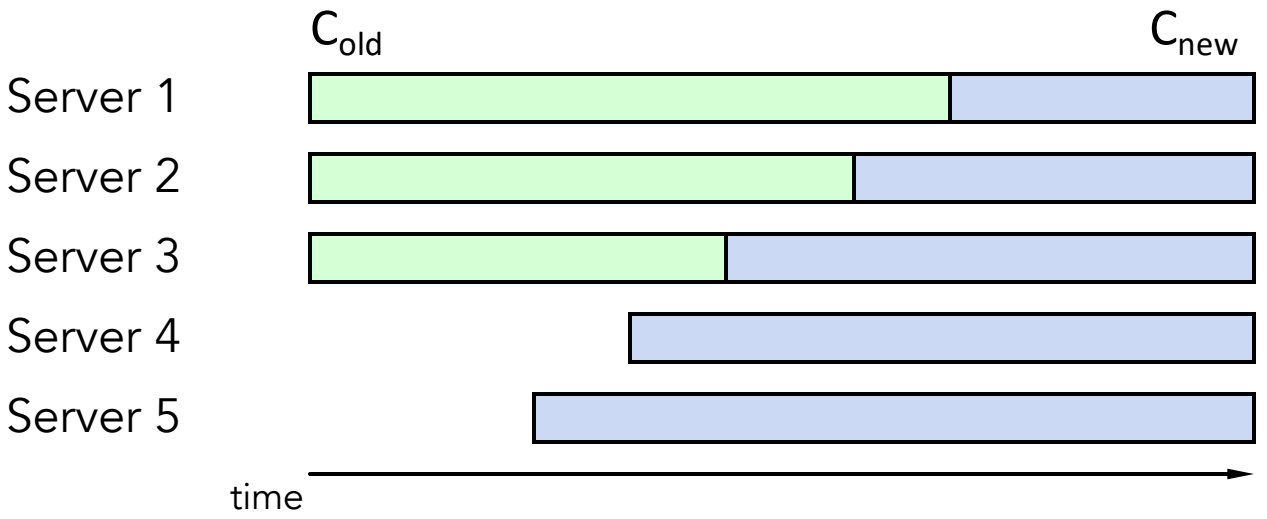
- Goal: allow the cluster of replicas to continue operating normally during changes

Strawman Solution

- Simply switch to another configuration
  - Operator issues a configuration change to all servers in the cluster

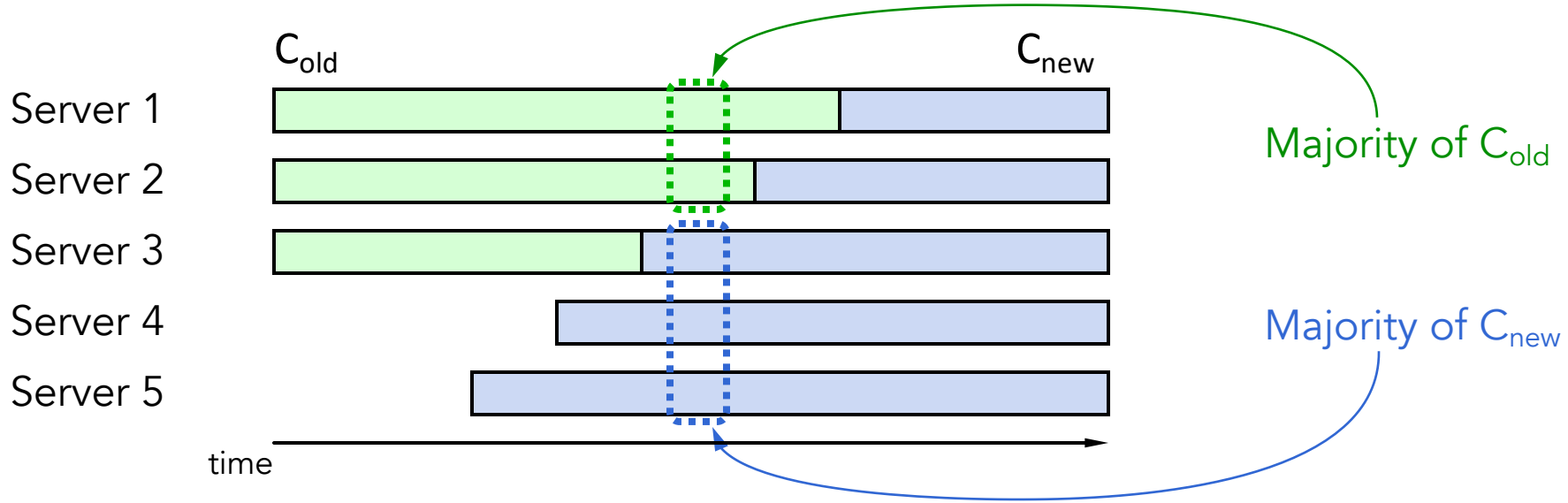
Problem with the Strawman Solution

Cannot switch directly from one configuration to another:
conflicting majorities could arise



Why not simply switch to another config?

Cannot switch directly from one configuration to another: conflicting majorities could arise



Preserving Safety During Config Changes

- Safety: there must be no point during the transition where it is possible for two leaders to be elected for the same term

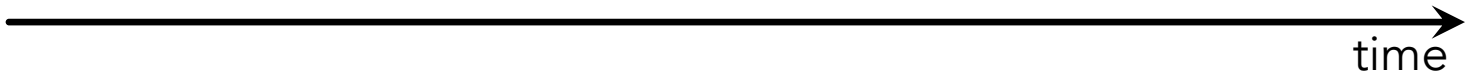
Goal: Allow individual servers to transition between configurations at different times without compromising safety

How does Raft solve this problem?

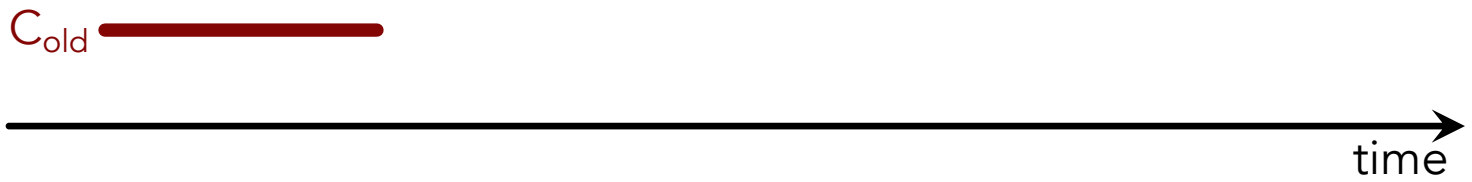
Joint Consensus

- Raft uses a 2-phase approach:
  - Intermediate phase uses joint consensus (need majority of both old and new configurations for elections, commitment)
  - Configuration change is just a log entry; applied immediately on receipt
  - Once joint consensus is committed, begin replicating log entry for final configuration
- Allows individual servers to transition between configurations at different times without compromising safety
  - Also allows the cluster to continue servicing client requests throughout the configuration change

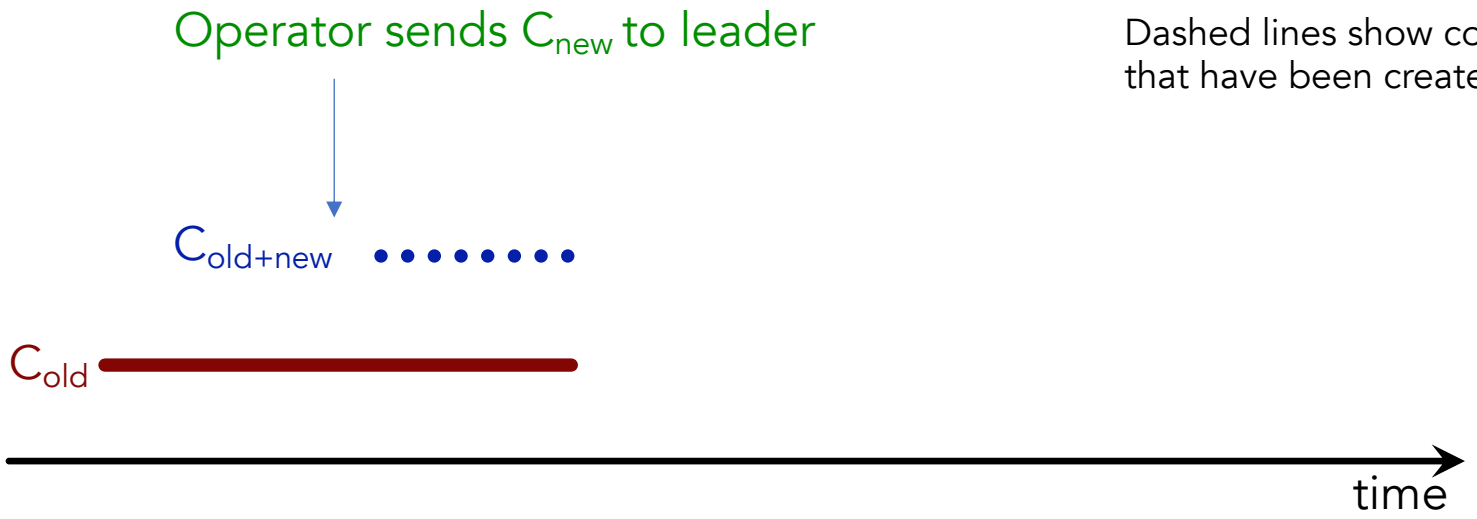
Timeline for a Configuration Change



Timeline for a configuration change



Timeline for a configuration change

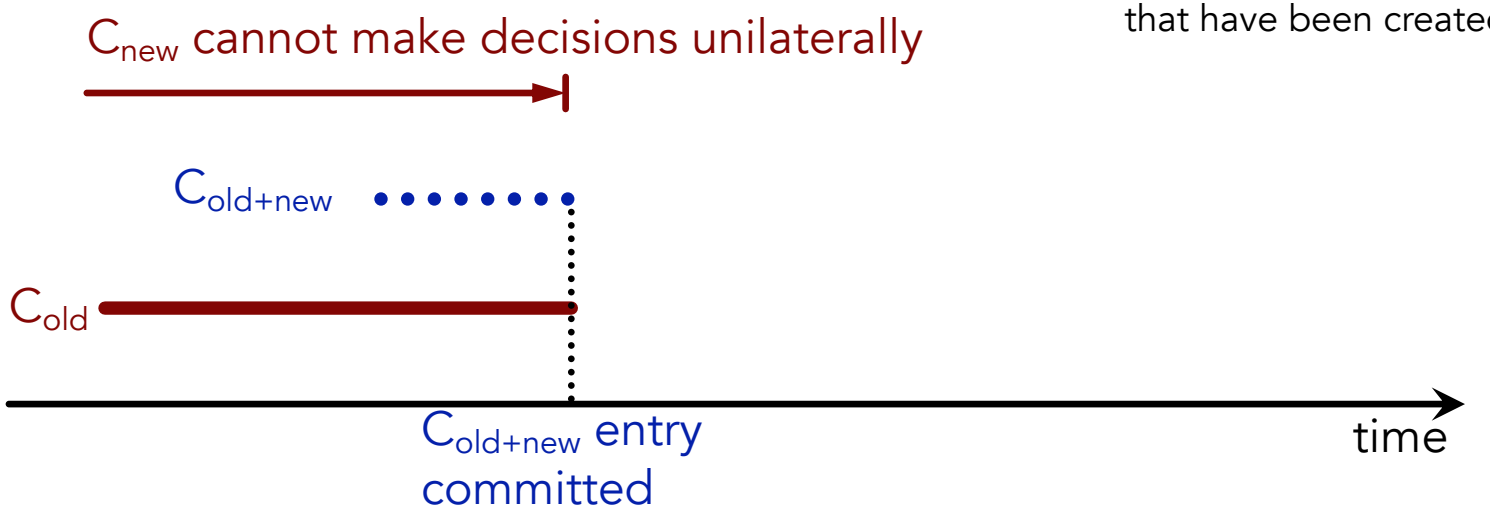


Dashed lines show configuration entries that have been created but not committed

1. Leader receives a request to change the configuration from C_{old} to C_{new}
2. Leader first stores the configuration change as a special log entry $C_{old+new}$

Timeline for a configuration change

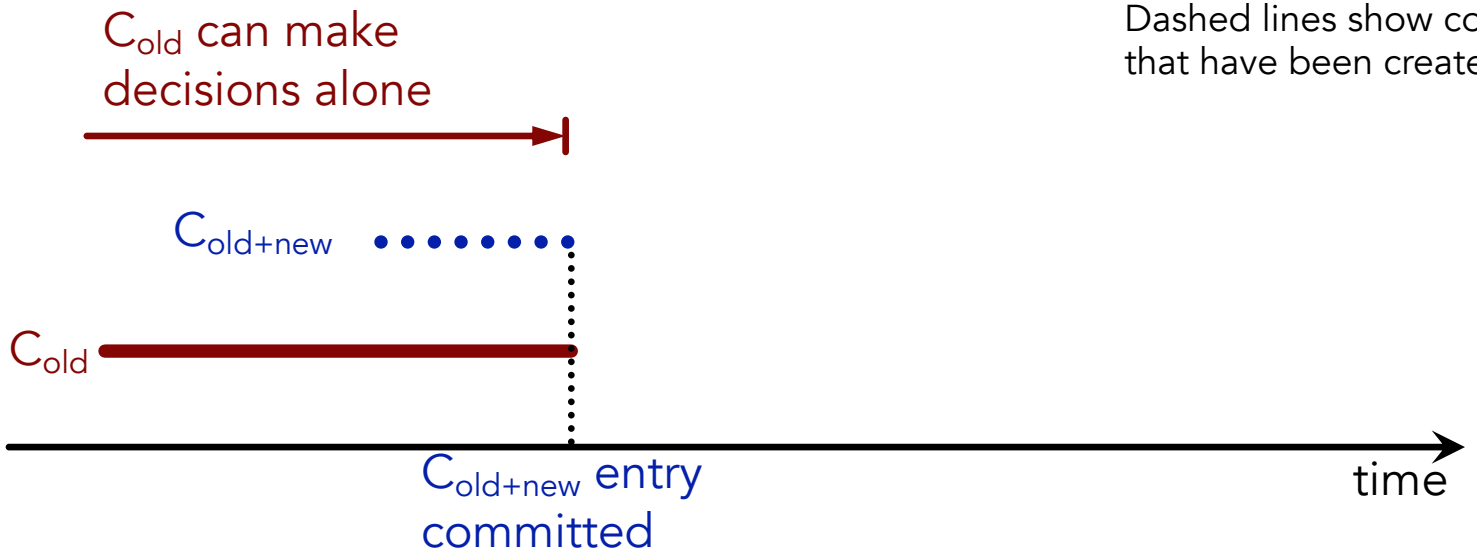
Dashed lines show configuration entries that have been created but not committed



1. Leader receives a request to change the configuration from C_{old} to C_{new}
2. Leader first stores the configuration change as a special log entry $C_{old+new}$
3. Commits $C_{old+new}$ (a majority of C_{old} and a majority of C_{new})

* Once a server adds a new configuration, it uses that configuration for all its future decisions, regardless of whether the configuration entry is committed

Timeline for a configuration change

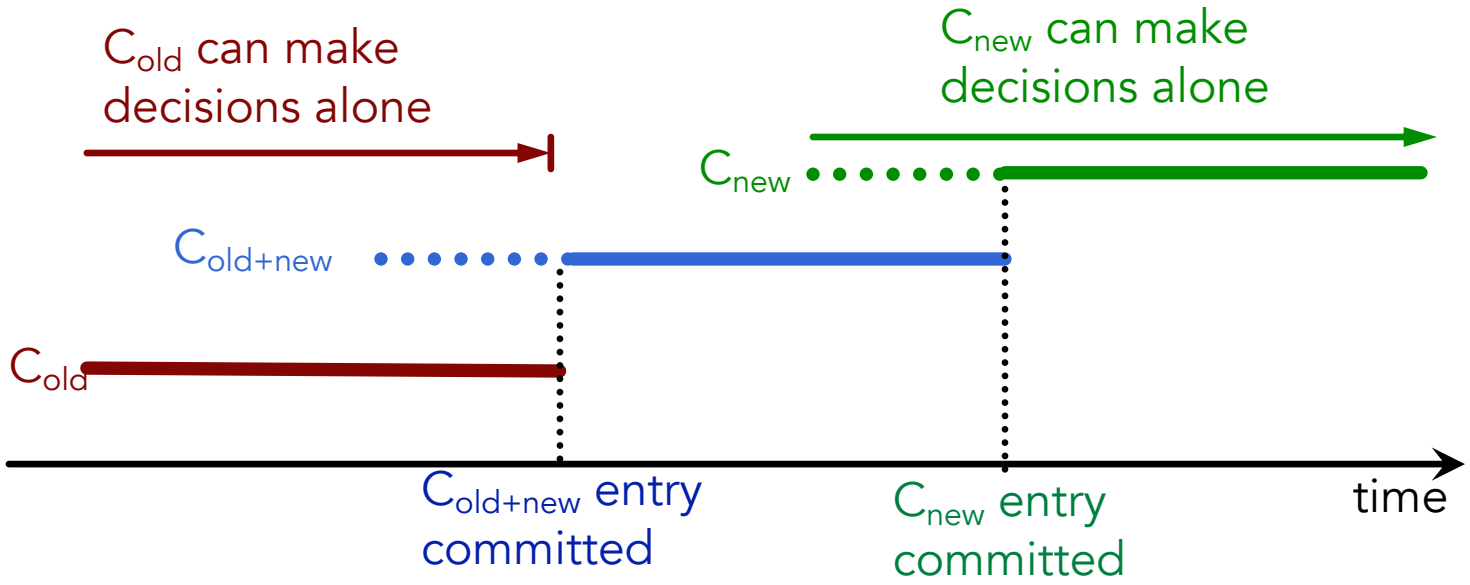


Dashed lines show configuration entries that have been created but not committed

1. Leader receives a request to change the configuration from C_{old} to C_{new}
2. Leader first stores the configuration change as a special log entry $C_{old+new}$
3. Commits $C_{old+new}$ (a majority of C_{old} and a majority of C_{new})

* Once a server adds a new configuration, it uses that configuration for all its future decisions, regardless of whether the configuration entry is committed

Timeline for a configuration change



1. Leader receives a request to change the configuration from C_{old} to C_{new}
2. Leader first stores the configuration change as a special log entry $C_{old+new}$
3. Commits it to $C_{old+new}$ (a majority of C_{old} and a majority of C_{new})
4. Once $C_{old+new}$ is committed, then the leader creates the C_{new} entry and commits it

There is no time when C_{old} and C_{new} can both make unilateral decisions → can't have two separate consensus → ensures safety

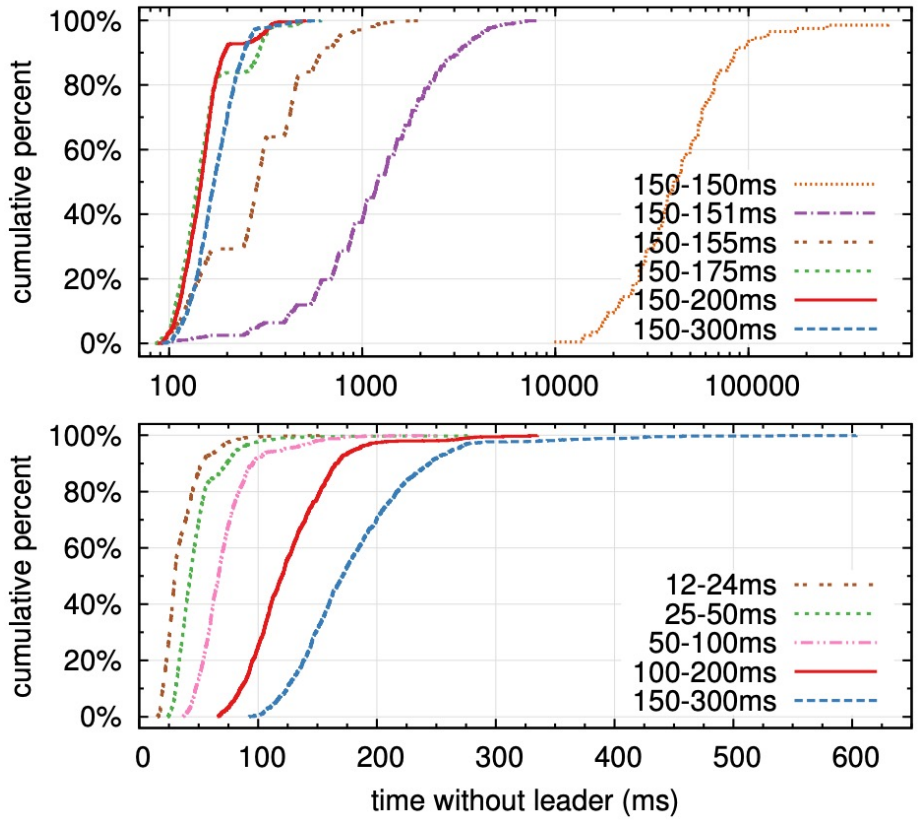
Other issues (Read section 6 & 7 of paper)

- New servers may not initially store any log entries
  - If they are added to the cluster in this state it could take a while for them to catch, during which time it may not be possible to commit new log entries
- Cluster leader may not be part of new configuration
- Removed servers can disrupt the cluster
- Log may grow without bound
  - As the log grows it occupies more space and takes more time to replay

Raft Summary

1. Leader election
   - Select one of the servers to act as leader
   - Detect crashes, choose new leader
2. Normal operation (basic log replication)
3. Safety and consistency after leader changes
4. Neutralizing old leaders
5. Client interactions
   - Implementing exactly-once semantics
6. Configuration changes:
   - Adding and removing servers

Raft Performance



Impact of adding randomness to election timeouts

Impact of reducing election timeouts

Discussion

- Can Raft provide linearizability?

Next Class

- Multi-Paxos