How Unreliable Computers Can Usually Agree (Sort Of)

A Brief Tour of the Raft Algorithm

Laura Hampton
@incunabulista

OSCON 2018

Introduction to Consensus

Computers are terrible

- Computers prone to failure
- Unreliable networks
- Availability
- How to avoid involving people when things fail

Effective Consensus Algorithms

- Consistency between participant state machines
- Tolerant of failure of 1 or more participants
- Tolerant of unreliable networks and network partition

Features of Consensus Algorithms

- Agreement
- Validity
- Termination

CAP Theorem

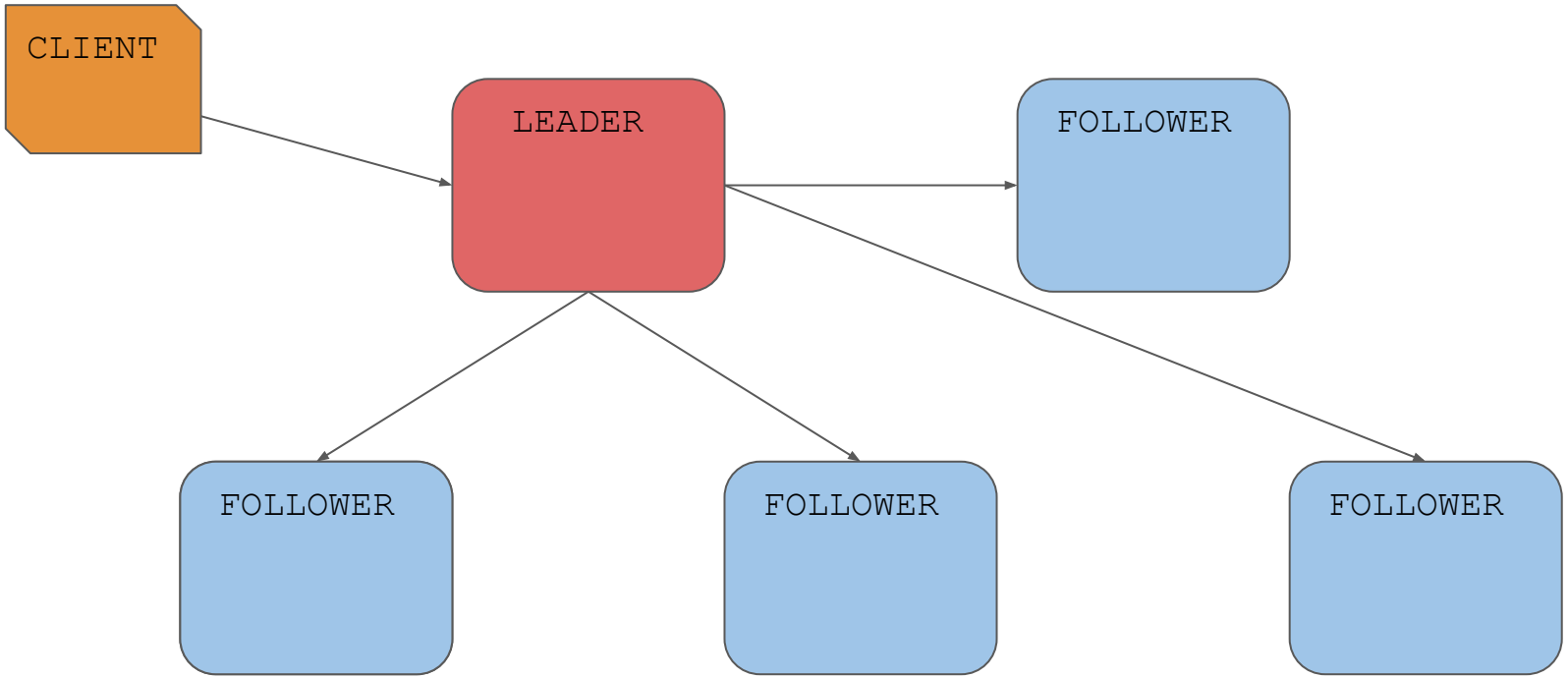
- Consistency
- Availability
- Partition Tolerance

What Could Possibly Go Wrong?

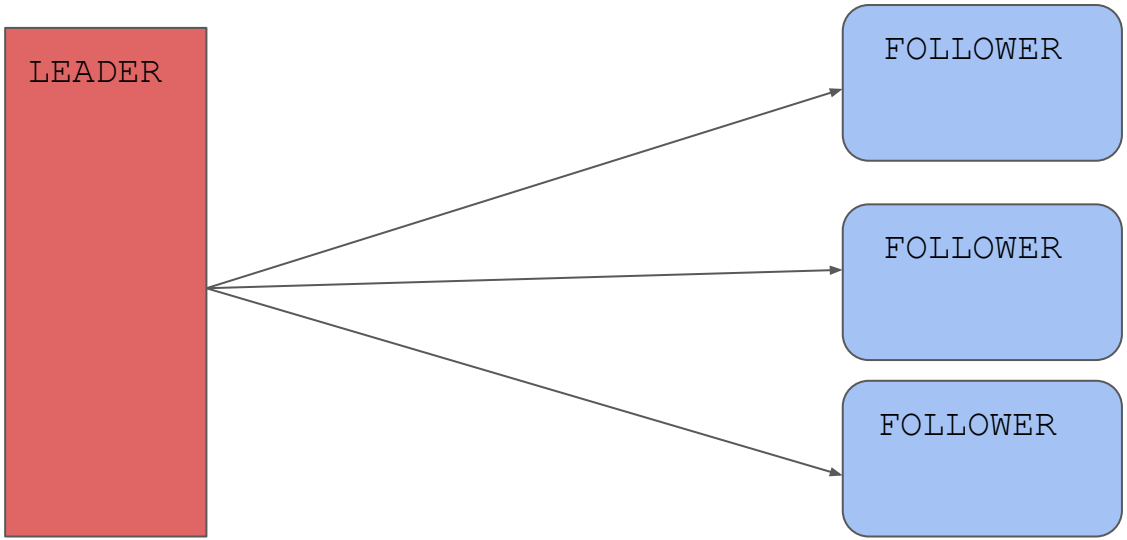
- Fail-stop
- Fail-recover
- Network partition
- Byzantine failure

Byzantine Failure

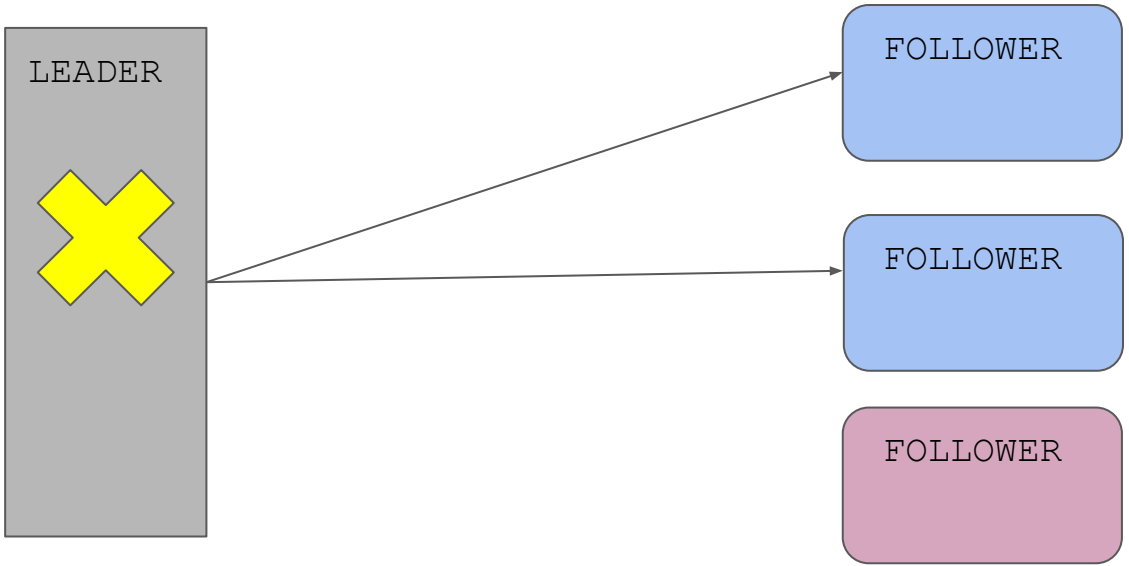
Terms



2-Phase Commit

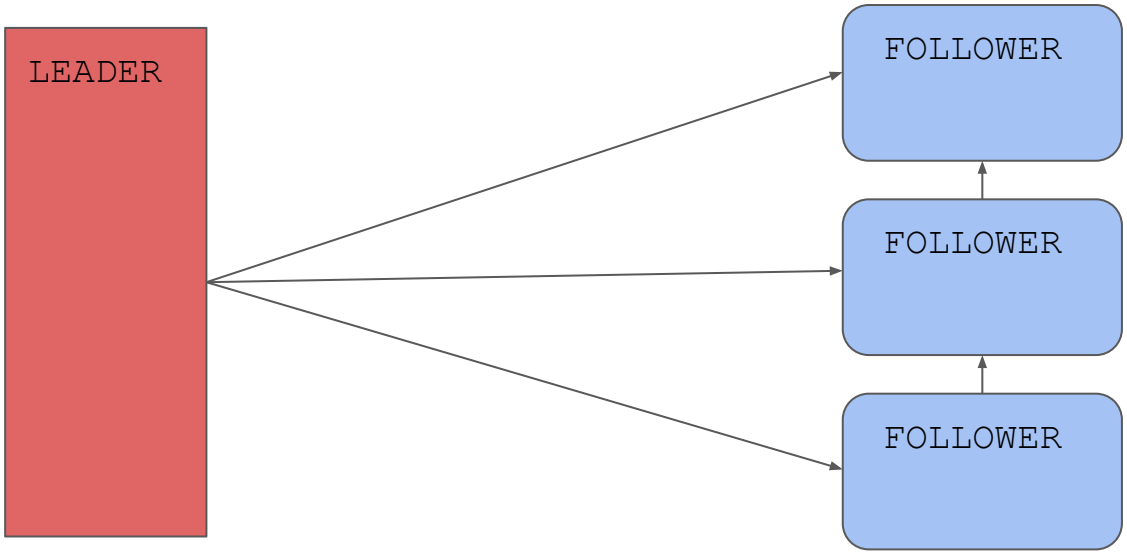


2-Phase Commit - Leader Fail-Stop

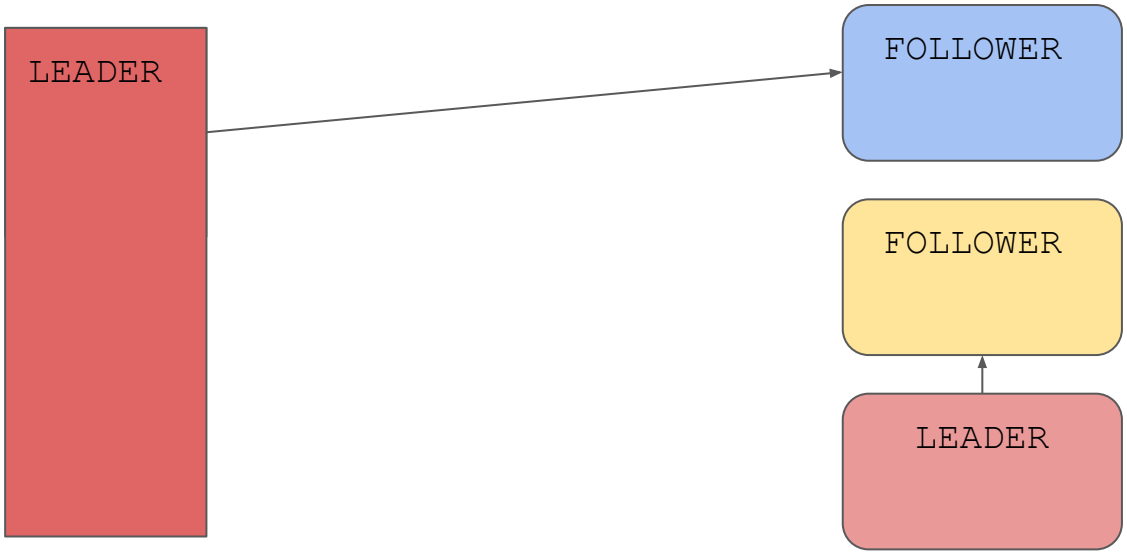



A PAGER GOES OFF

3-Phase Commit



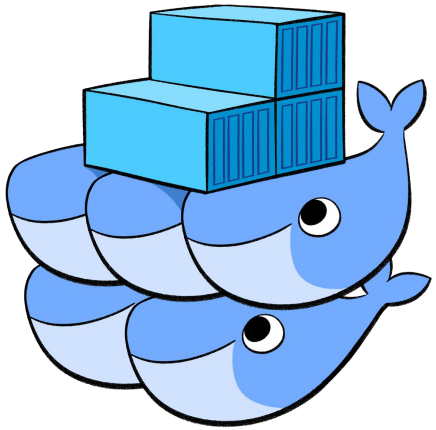
3-Phase Commit: Fail-Recover




A PAGER GOES OFF

Raft in Theory

Uses for Raft

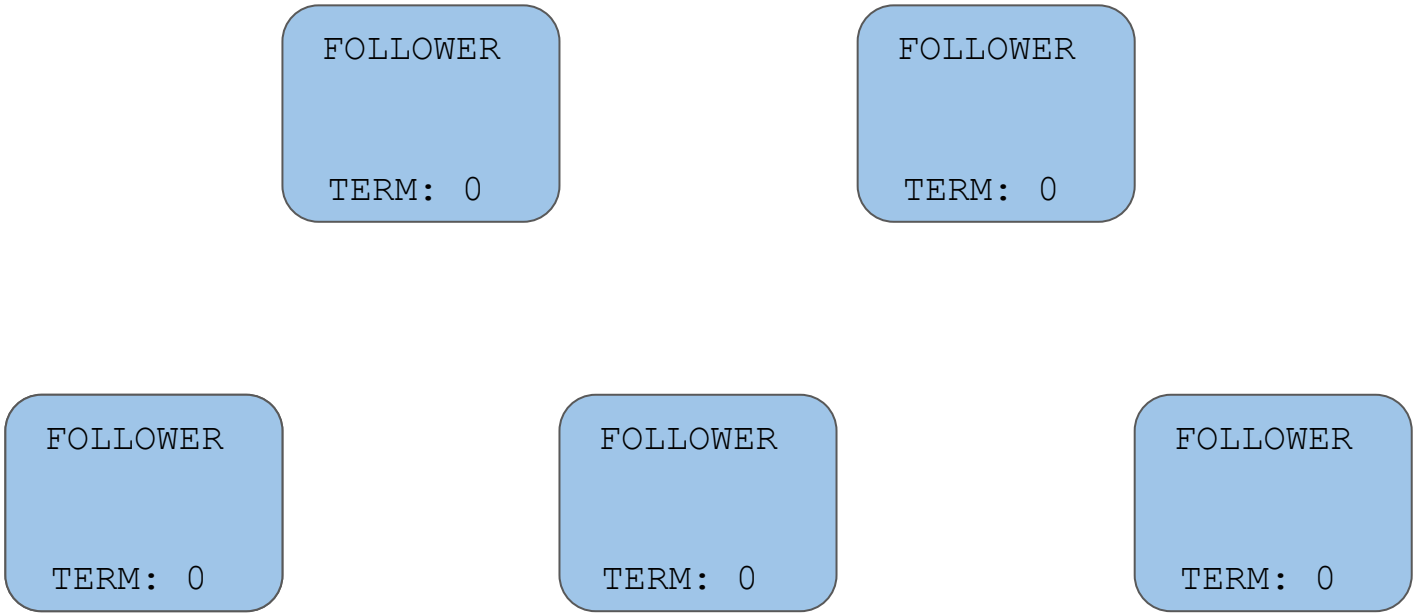


etcd

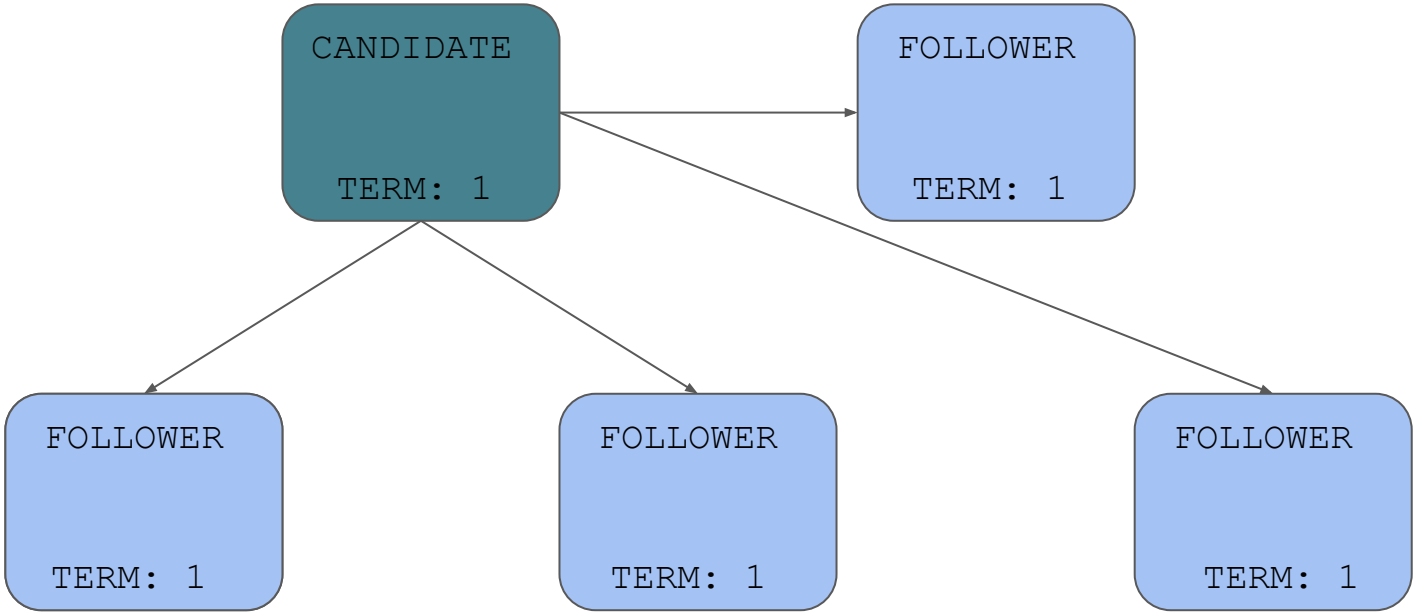
mongoDB®

Election and Log Propagation

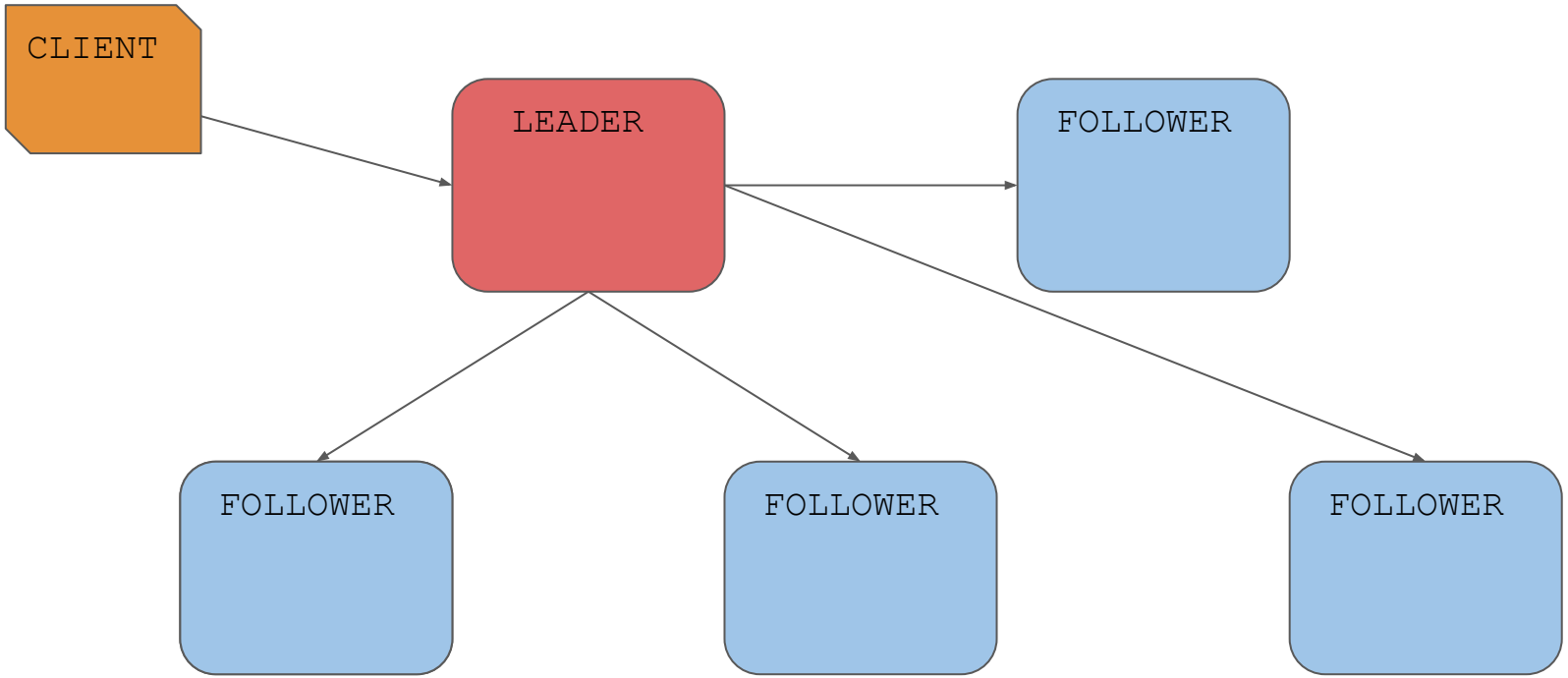
Raft Cluster



Leader Election



Log Replication



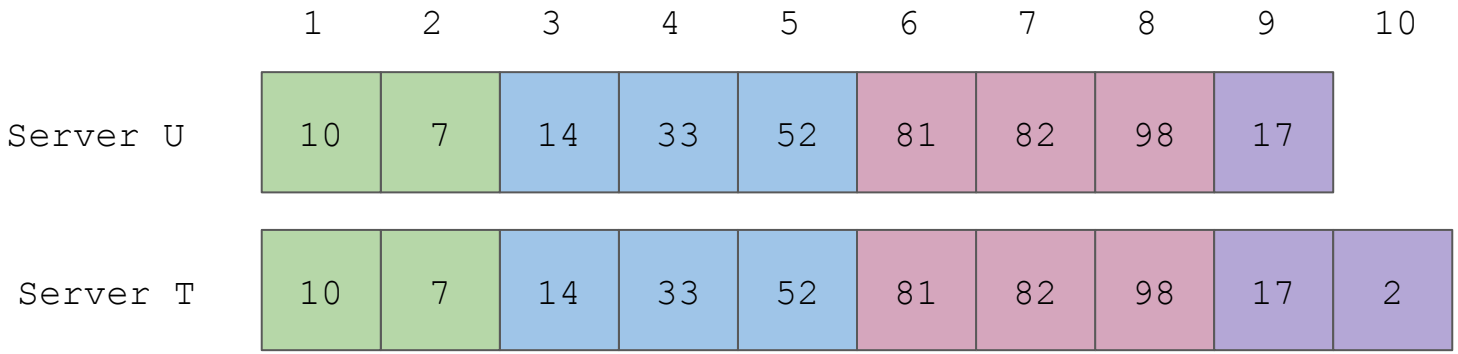
Vote Message

```
Vote(
    status = self.status,
    sender_id = self.id,
    sender_log_length = len(self.log),
    recipient_id=recipient,
    term = self.term,
    vote=None #None to request, 1 vote yes, 0 vote no
     )
```

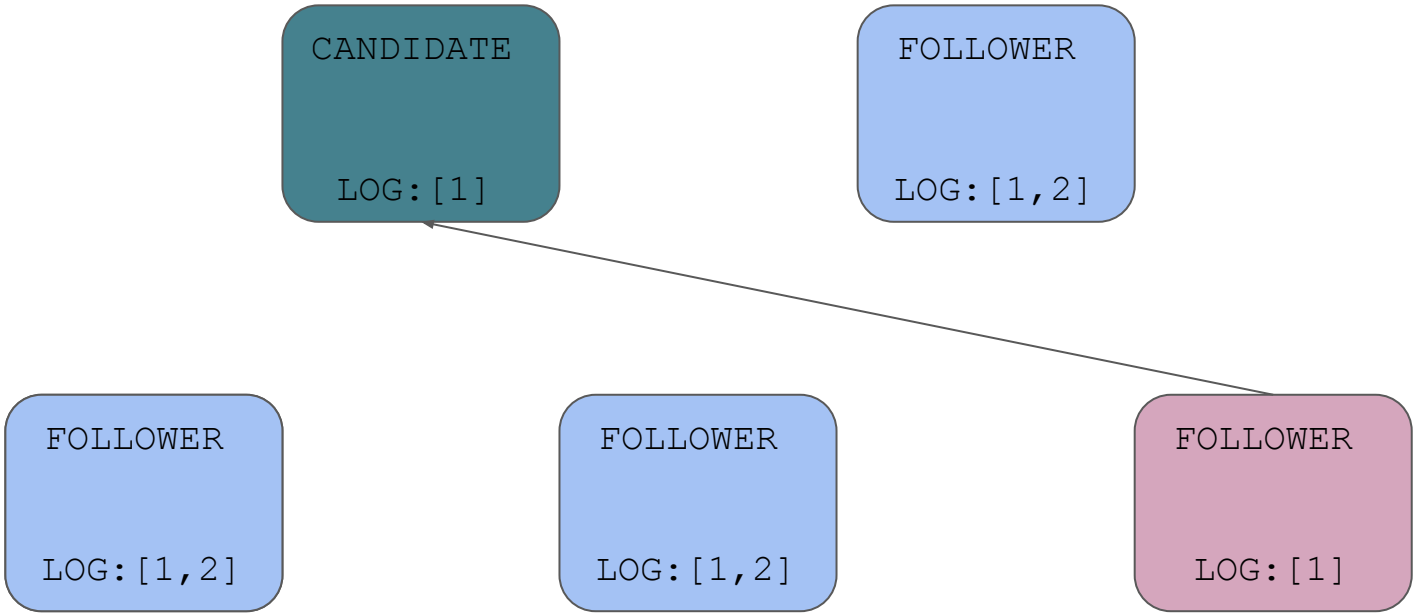
Add Entries Message

```
AddEntry(
    status = self.status,
    sender_id = self.id,
    recipient_id=recipient,
    term = self.term,
    commit = False,
    success = None,#False to reject, True if logs match
    last_log_index = self.last_log_index,
    last_log_entry = self.last_log_entry, #None for heartbeat
    new_log_entry = self.new_log_entry #None for heartbeat
     )
```

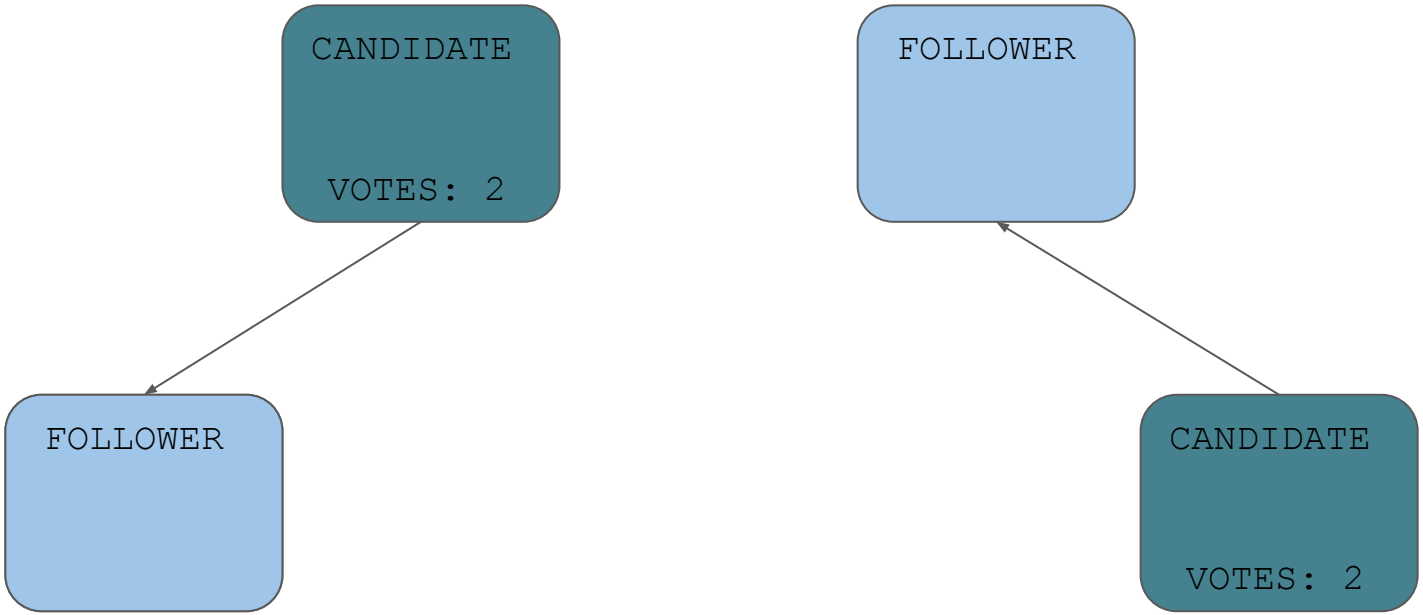
Log Safety


1
2
3
4
5
6
7
8
9
10
Server U
10
7
14
33
52
81
82
98
17
Server T
10
7
14
33
52
81
82
98
17
2

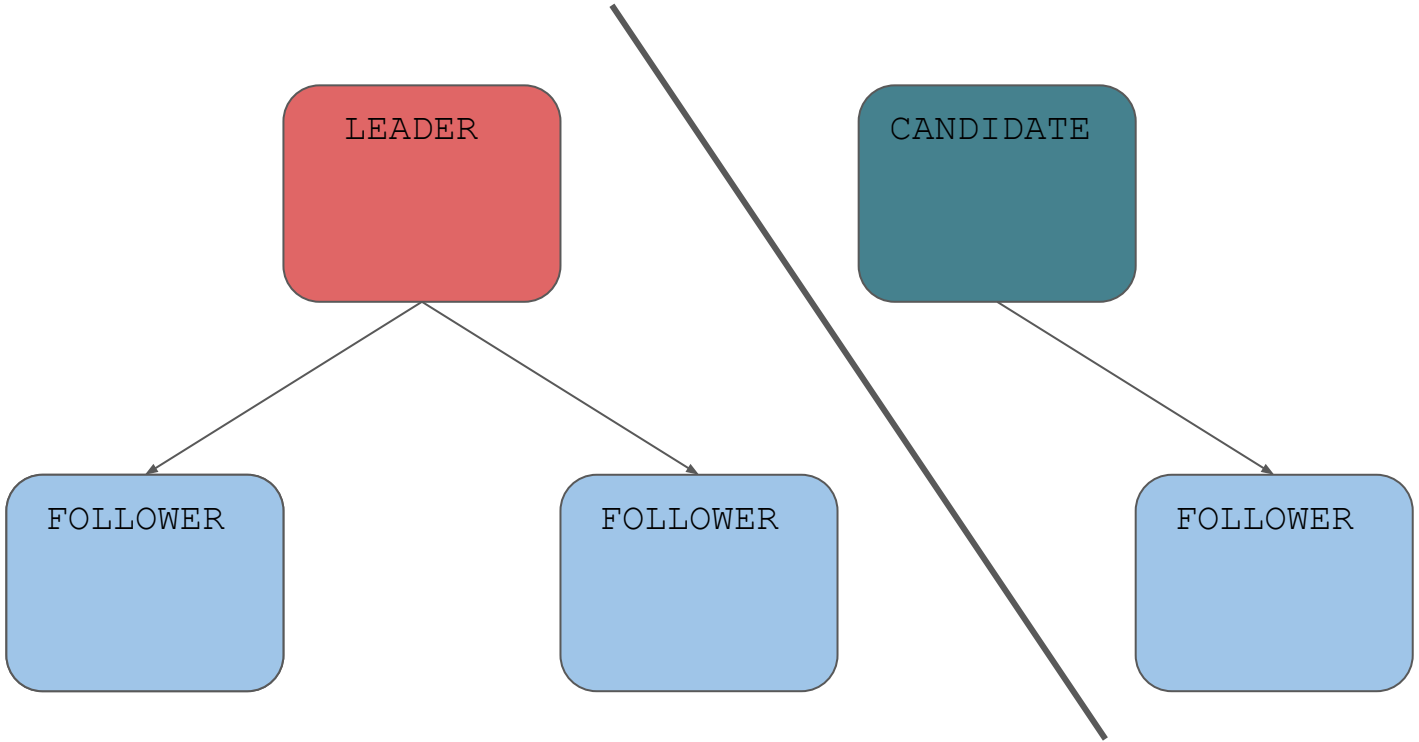
Leader Completeness



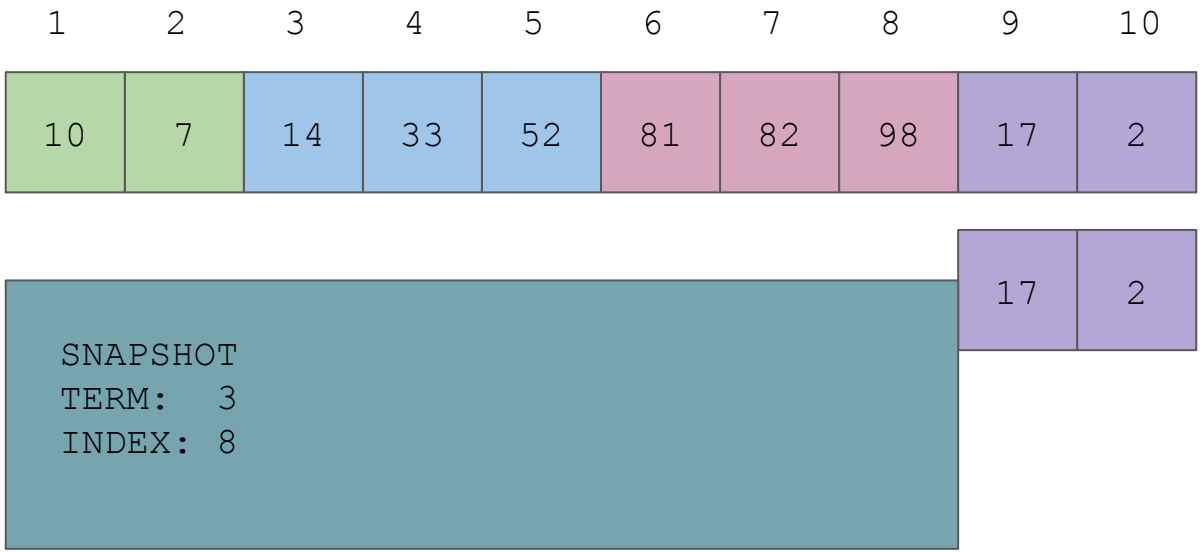
Split Vote



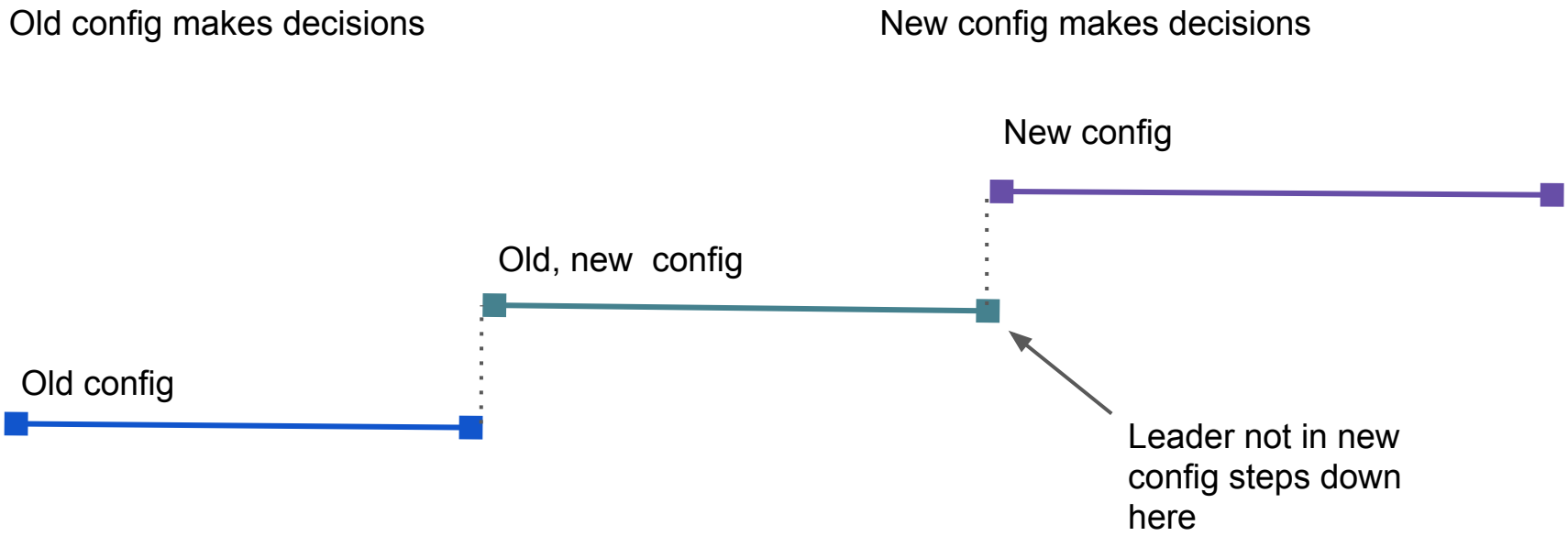
Network Partition



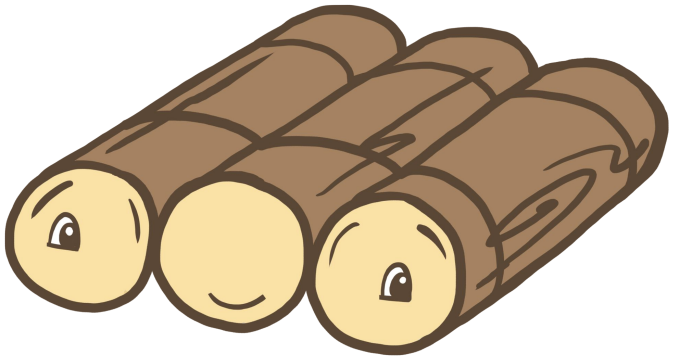
Log Compaction



Membership Changes



Success!



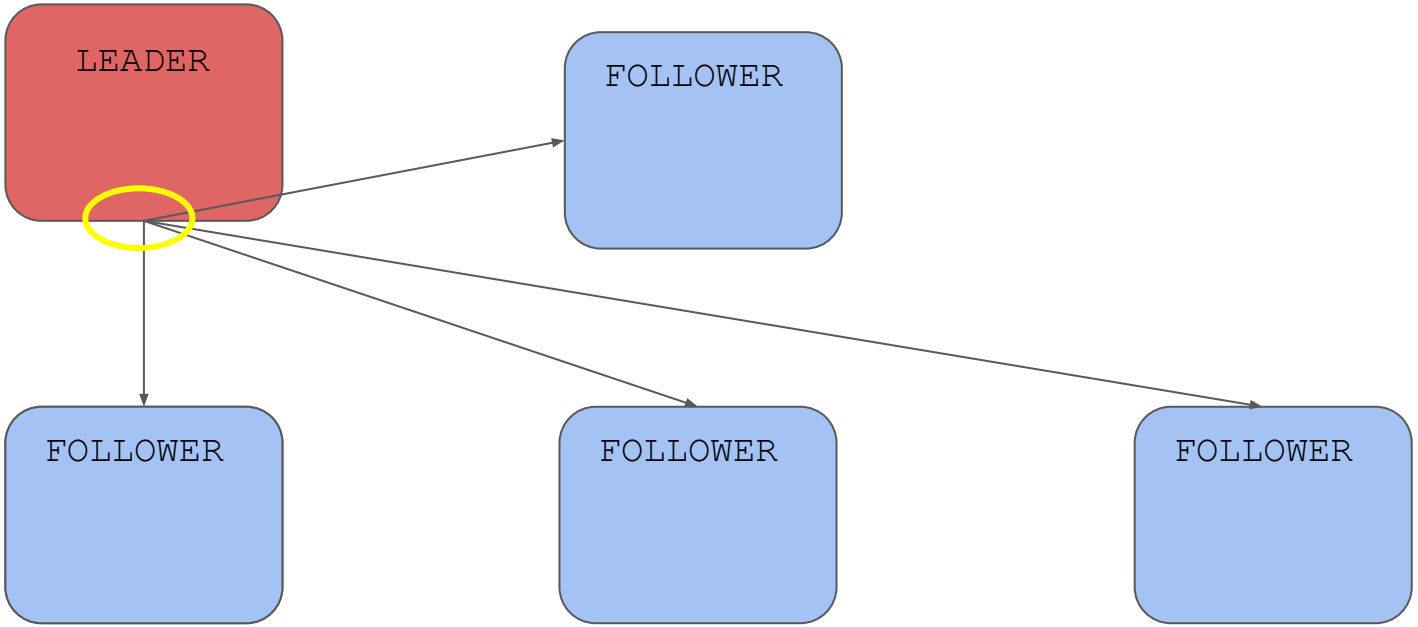
A Brief Trip to a Greek Island

Raft in Practice

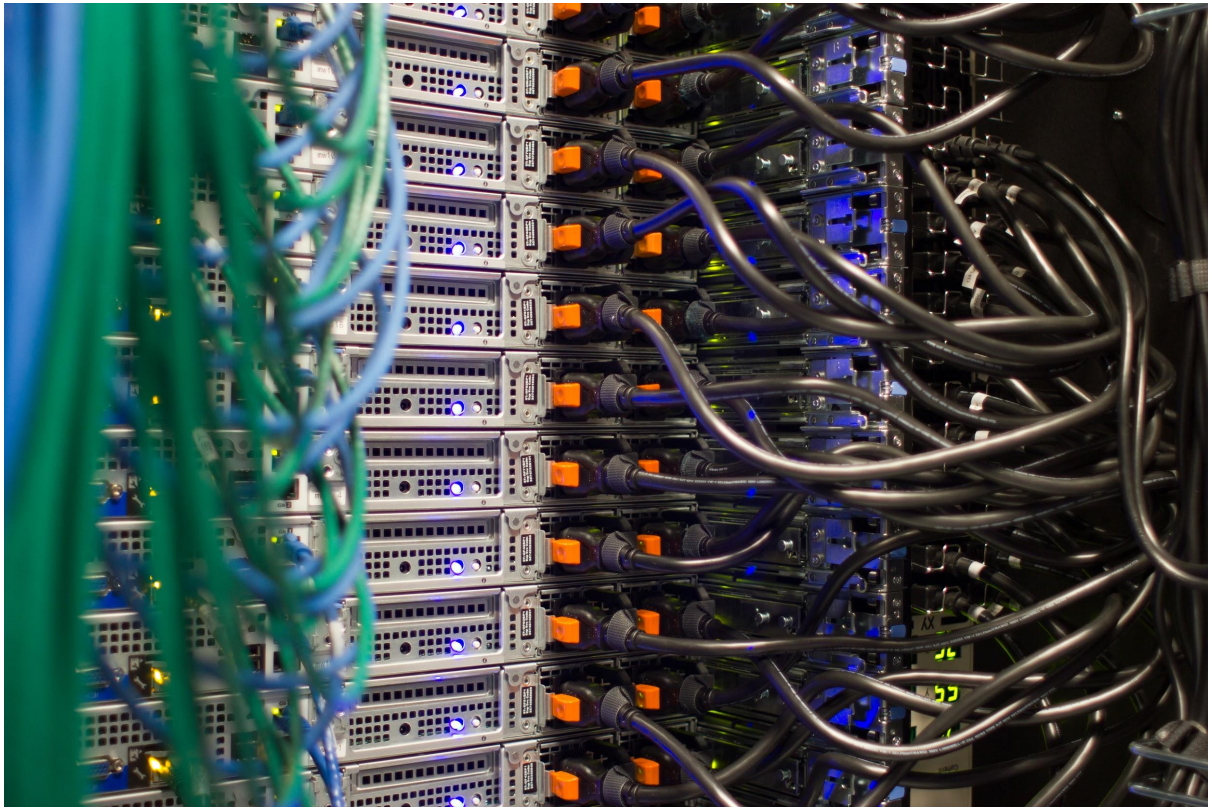
Setting up a Cluster

- Reliability
- Maintenance schedule
- Risk
- Performance
- Cost

Performance Considerations



Where to Locate Replica Nodes

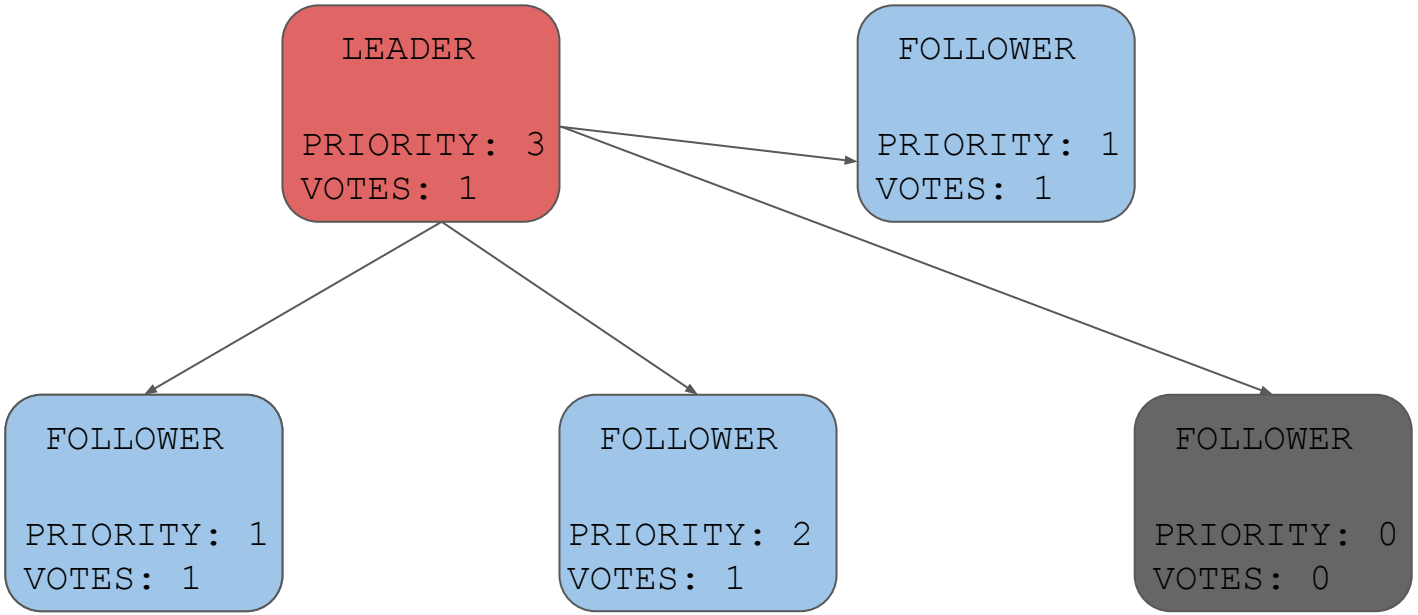


Raft and etcd

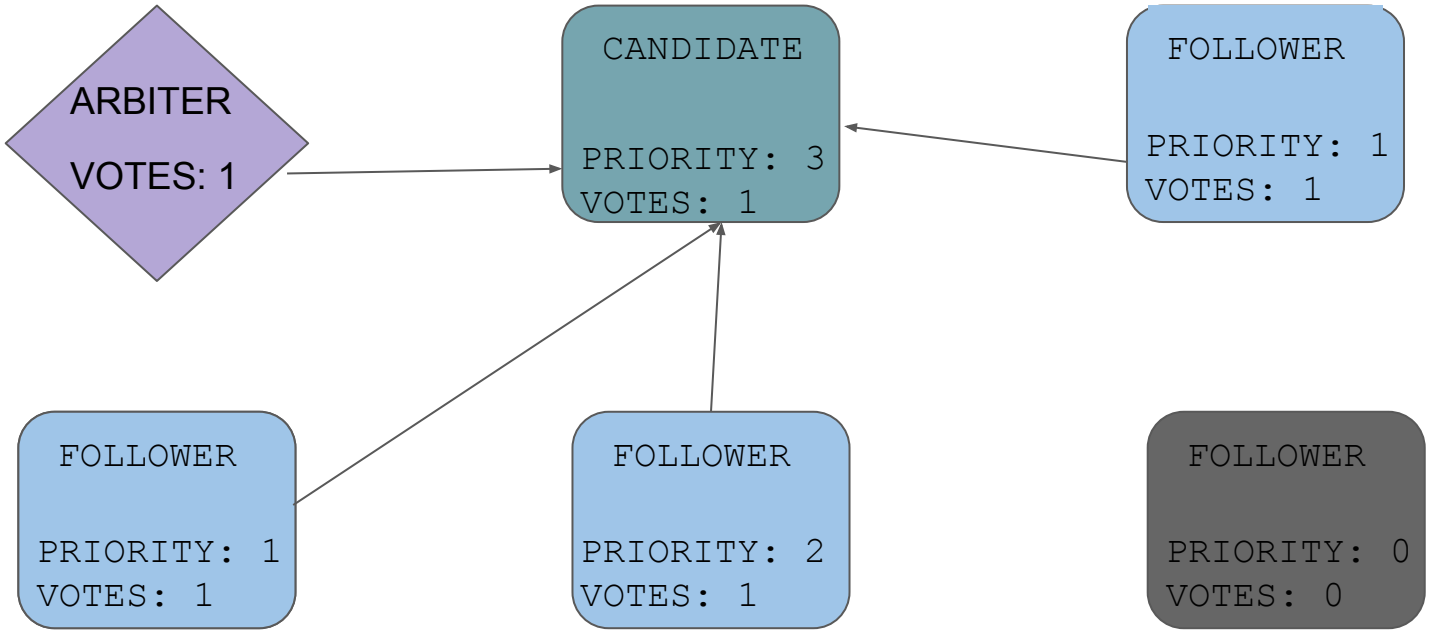
Raft as Implemented in MongoDB

- Priority
- Arbiters
- Chaining
- Delayed members

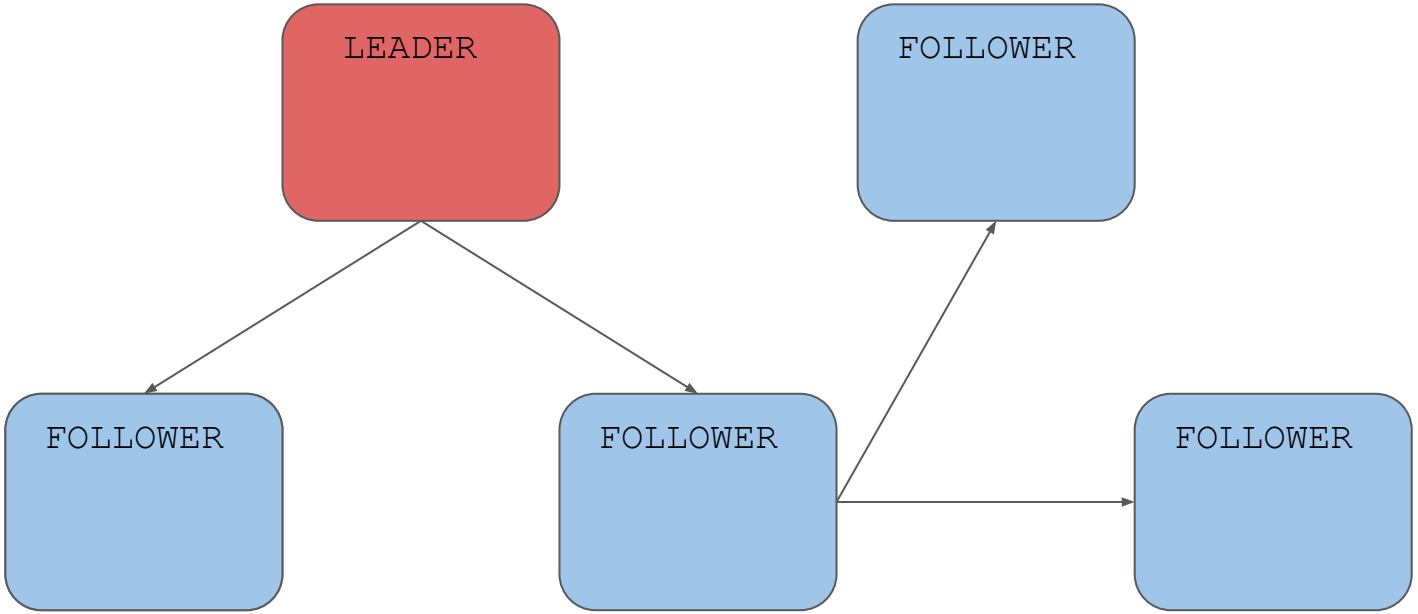
Priority


LEADER
PRIORITY: 3
VOTES: 1
FOLLOWER
PRIORITY: 1
VOTES: 1
FOLLOWER
PRIORITY: 1
VOTES: 1
FOLLOWER
PRIORITY: 2
VOTES: 1
FOLLOWER
PRIORITY: 0
VOTES: 0

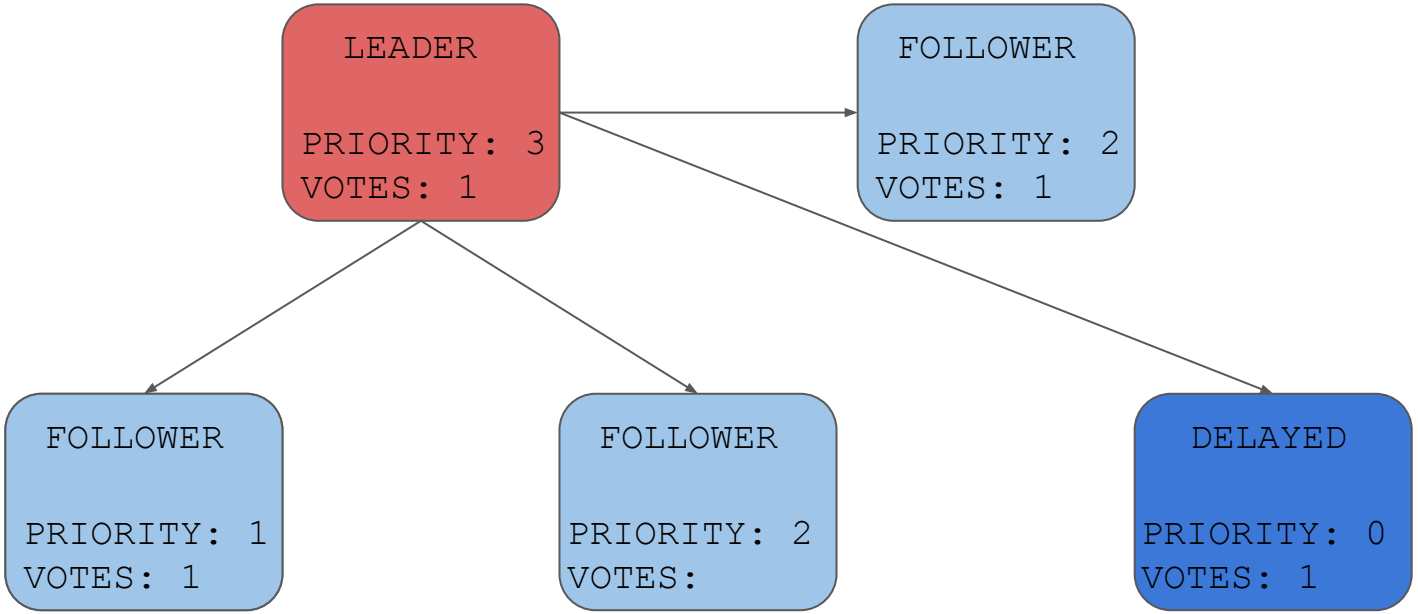
Arbiters


ARBITER
VOTES: 1
CANDIDATE
PRIORITY: 3
VOTES: 1
FOLLOWER
PRIORITY: 1
VOTES: 1
FOLLOWER
PRIORITY: 1
VOTES: 1
FOLLOWER
PRIORITY: 2
VOTES: 1
FOLLOWER
PRIORITY: 0
VOTES: 0

Chaining



Delayed Participants

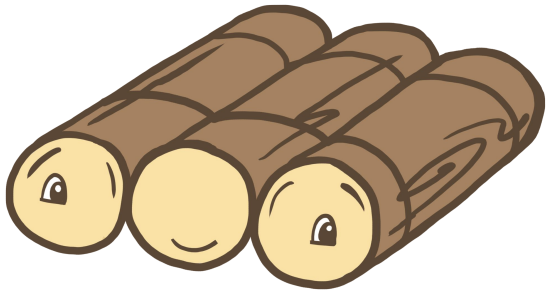


Resources

Raft paper: https://raft.github.io/raft.pdf

Raft visualization with explanations: http://thesecretlivesofdata.com/raft/

FLP Paper: https://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf



Thank you