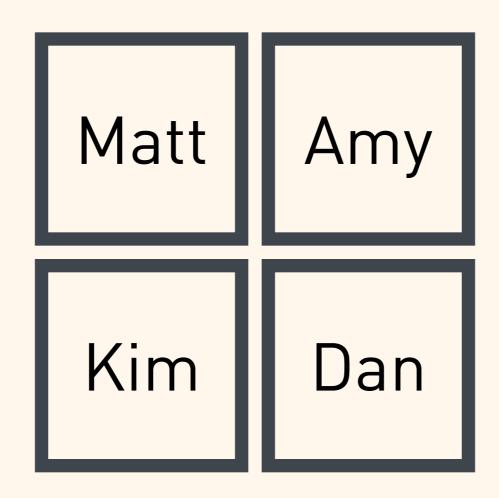
Raft

Consensus for Rubyists

ExampleBig Nerd Ranch

Coffee Snob Scenario

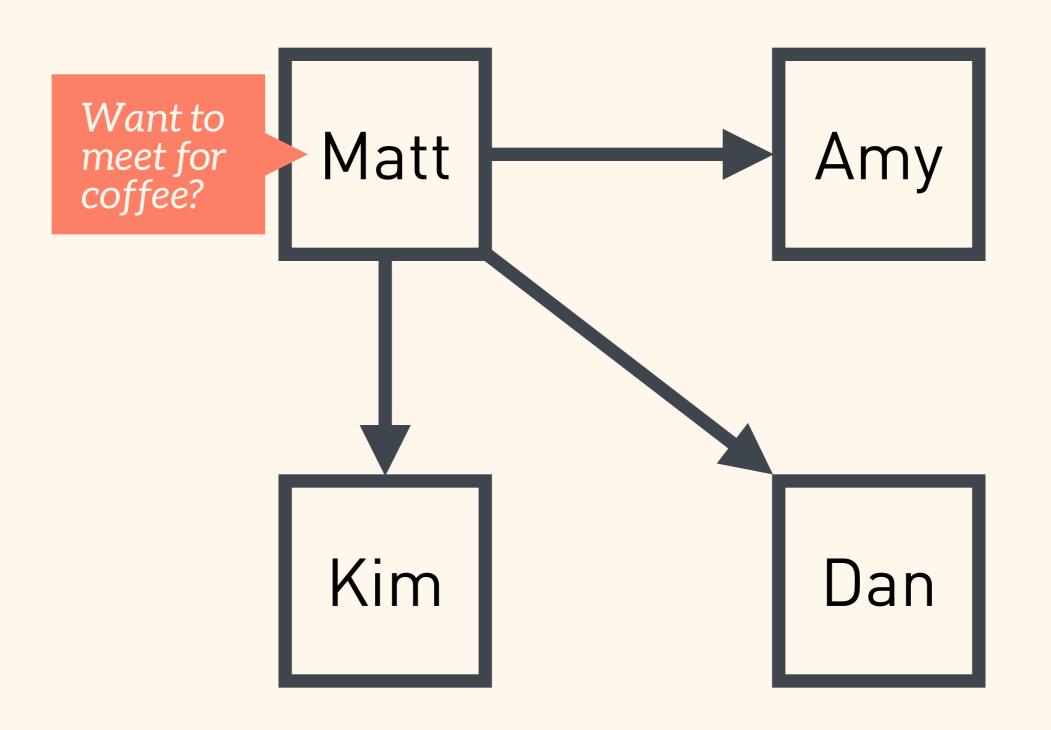


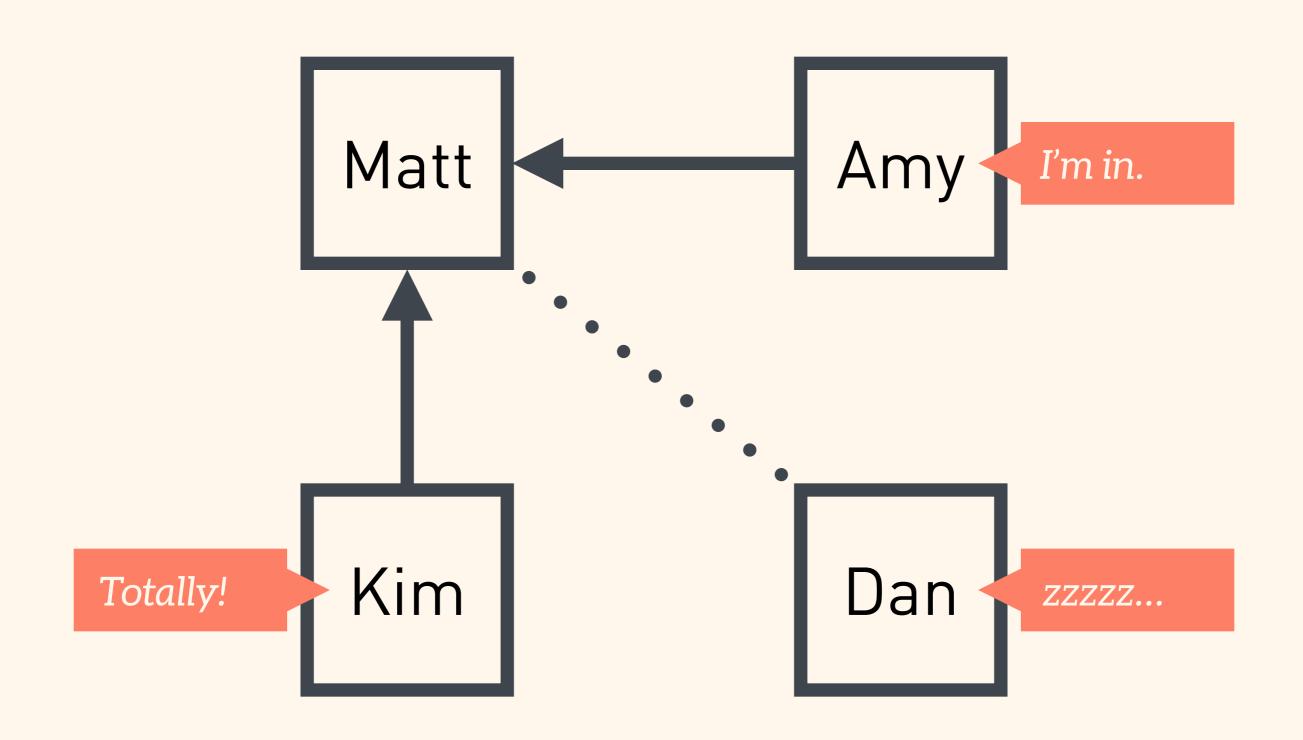
Matt

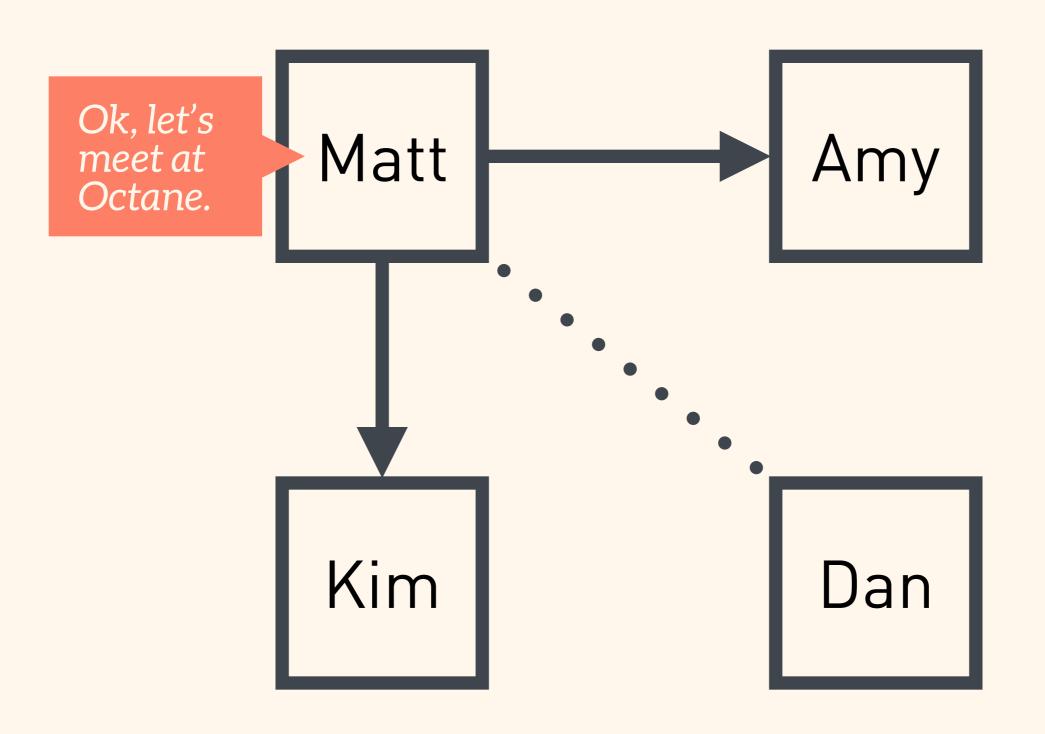
Amy

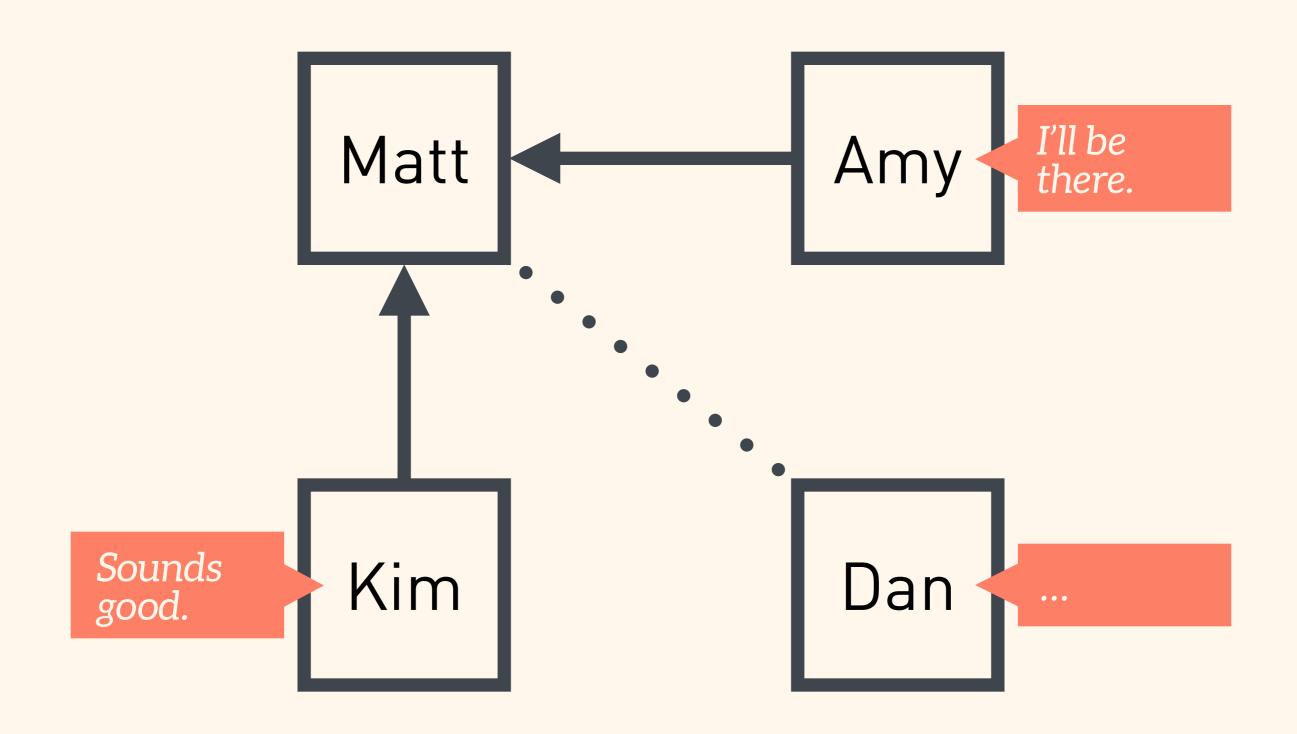
Kim

Dan









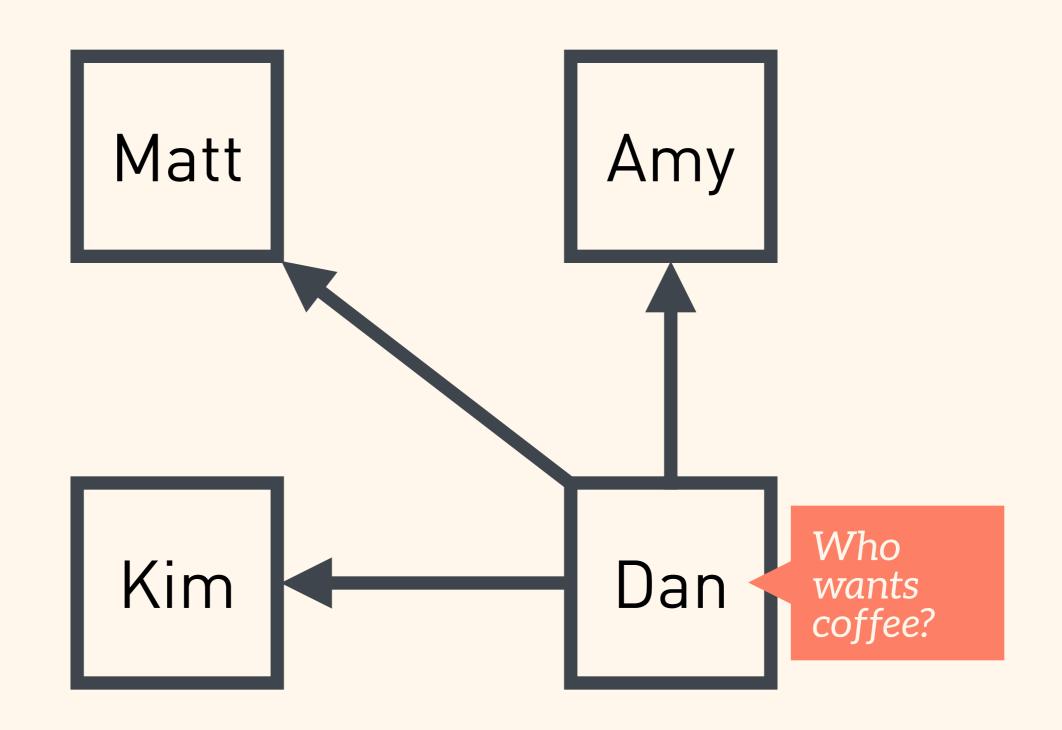
Matt

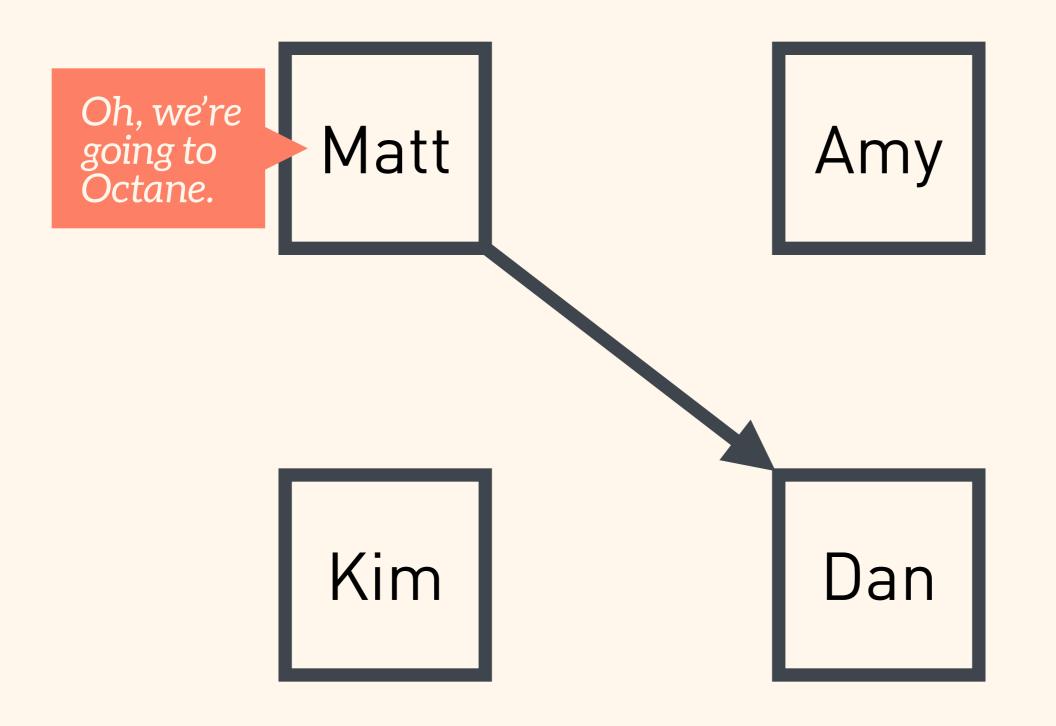
Amy

Kim

Dan

*wakes up





Delicious Coffee Consensus

What did we just do?

- Leader election
- State replication
- Partition tolerance

Now write an algorithm.

Humans > Computers > Congress

Mhy Ruby

Multiple db servers?
Multiple app servers?
Multiple clients?

Multiple db servers? **Distributed**Multiple app servers?
Multiple clients?

Multiple db servers? **Distributed**Multiple app servers? **Distributed**Multiple clients?

Multiple db servers? **Distributed**Multiple app servers? **Distributed**Multiple clients? **Distributed**

If you build webapps, you build distributed systems.

Raft

@ongardie Diego Ongaro and John Ousterhout

con-sen-sus

/kən'sensəs/

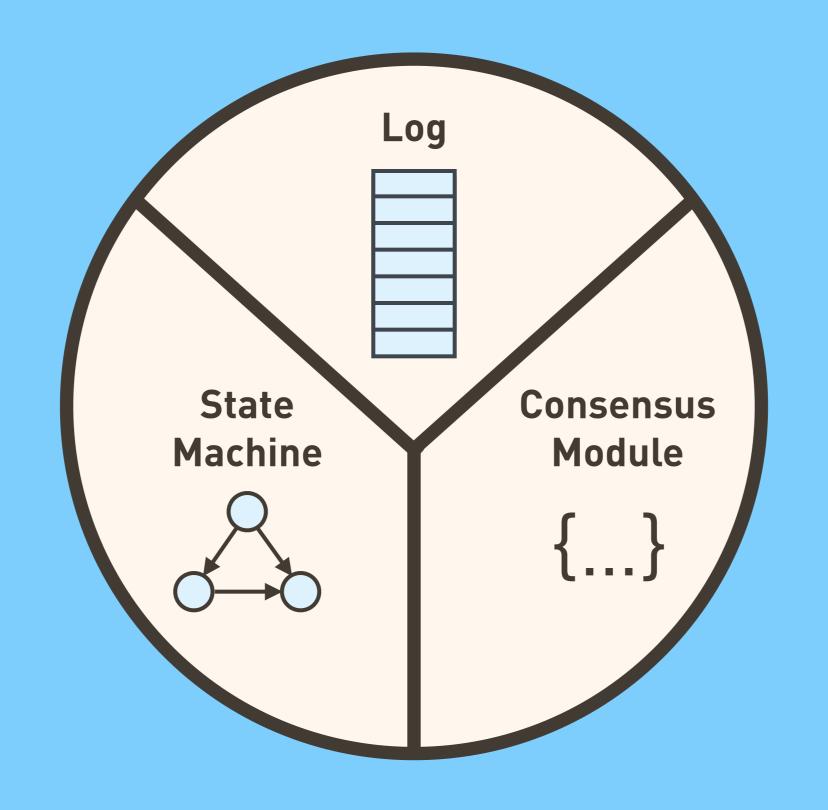
Agreeing upon state across distributed processes even in the presence of failures.

When should I use it?

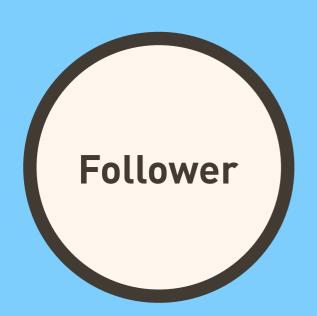
- Distributed locks
- Configuration
- Background jobs

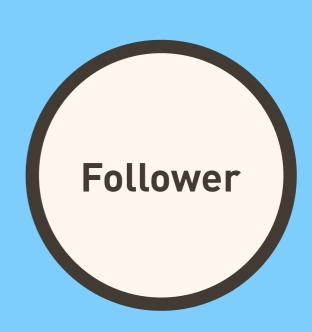
Why not Paxos?

Laying the Groundwork

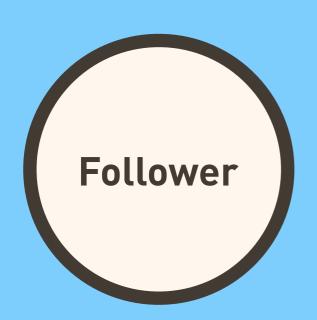


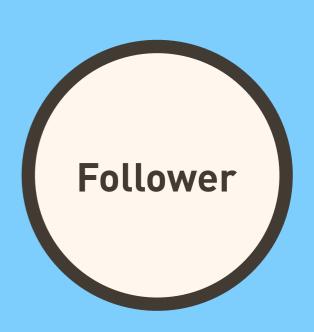


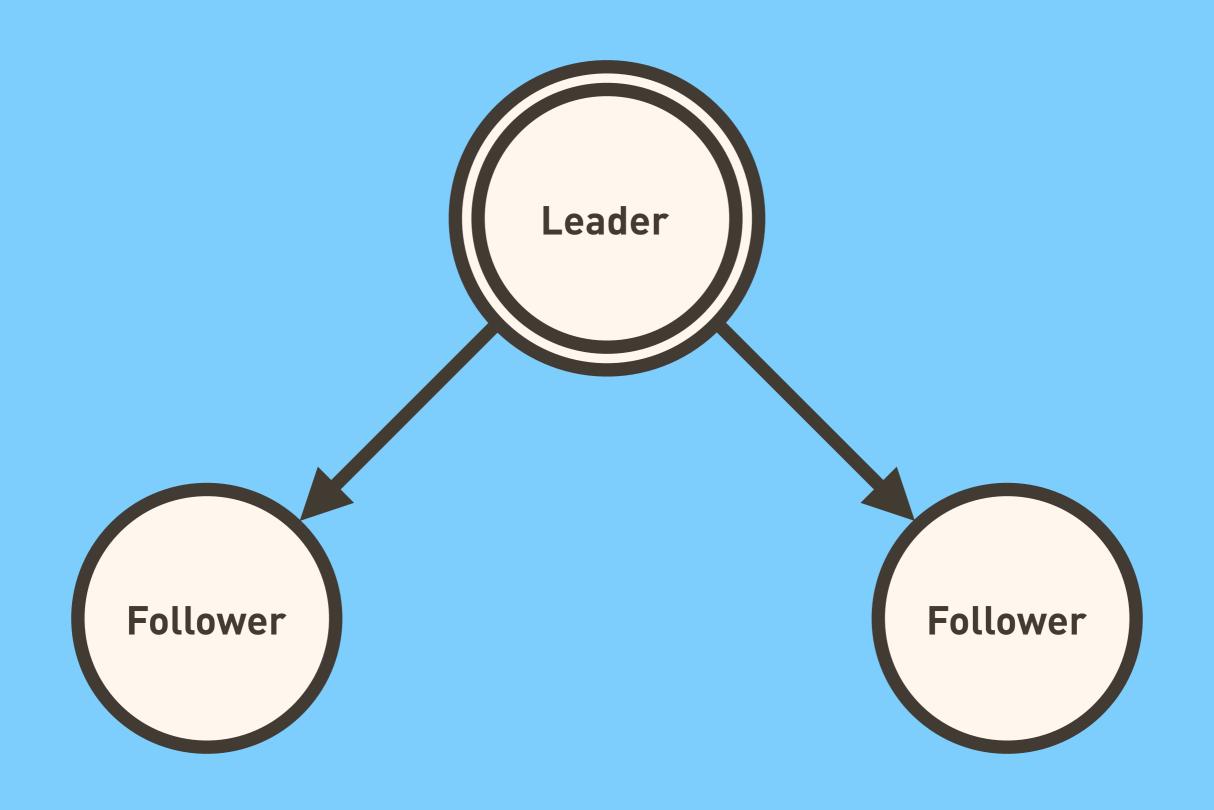


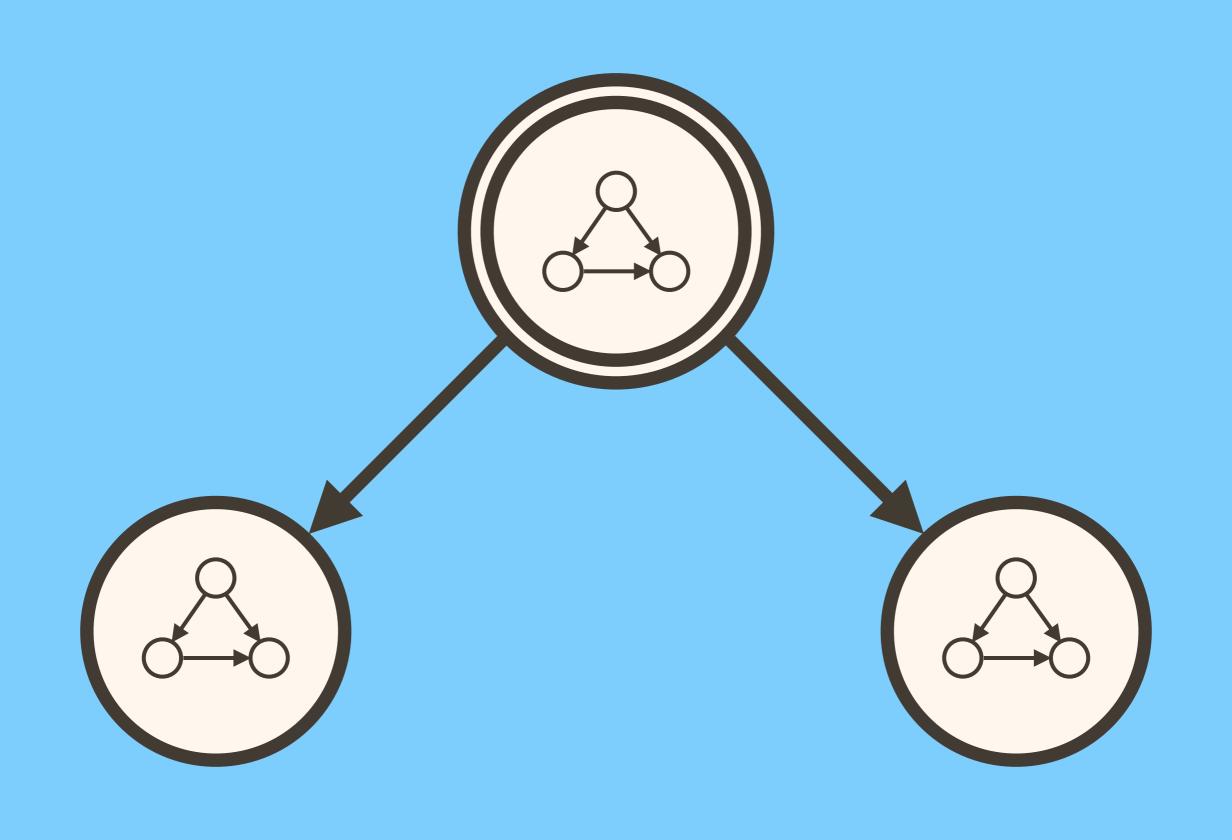












Leader Election

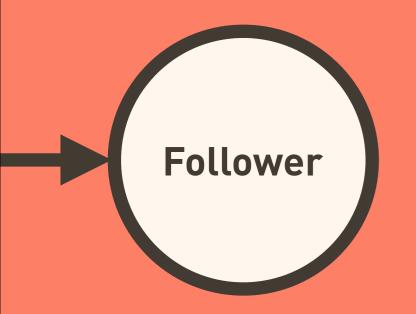
Log Replication

Safety









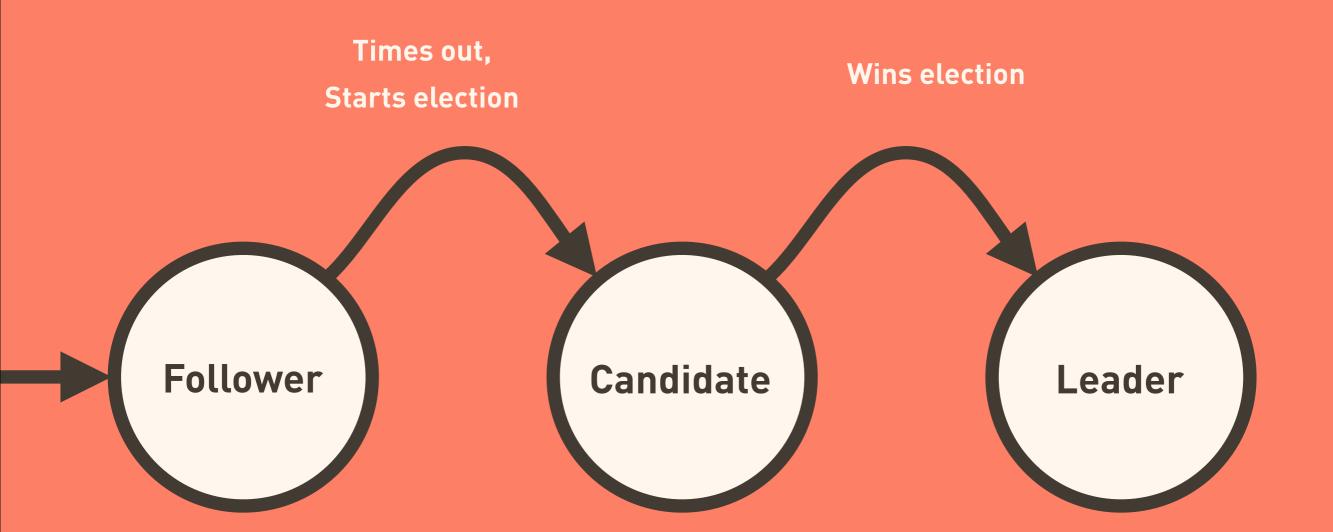


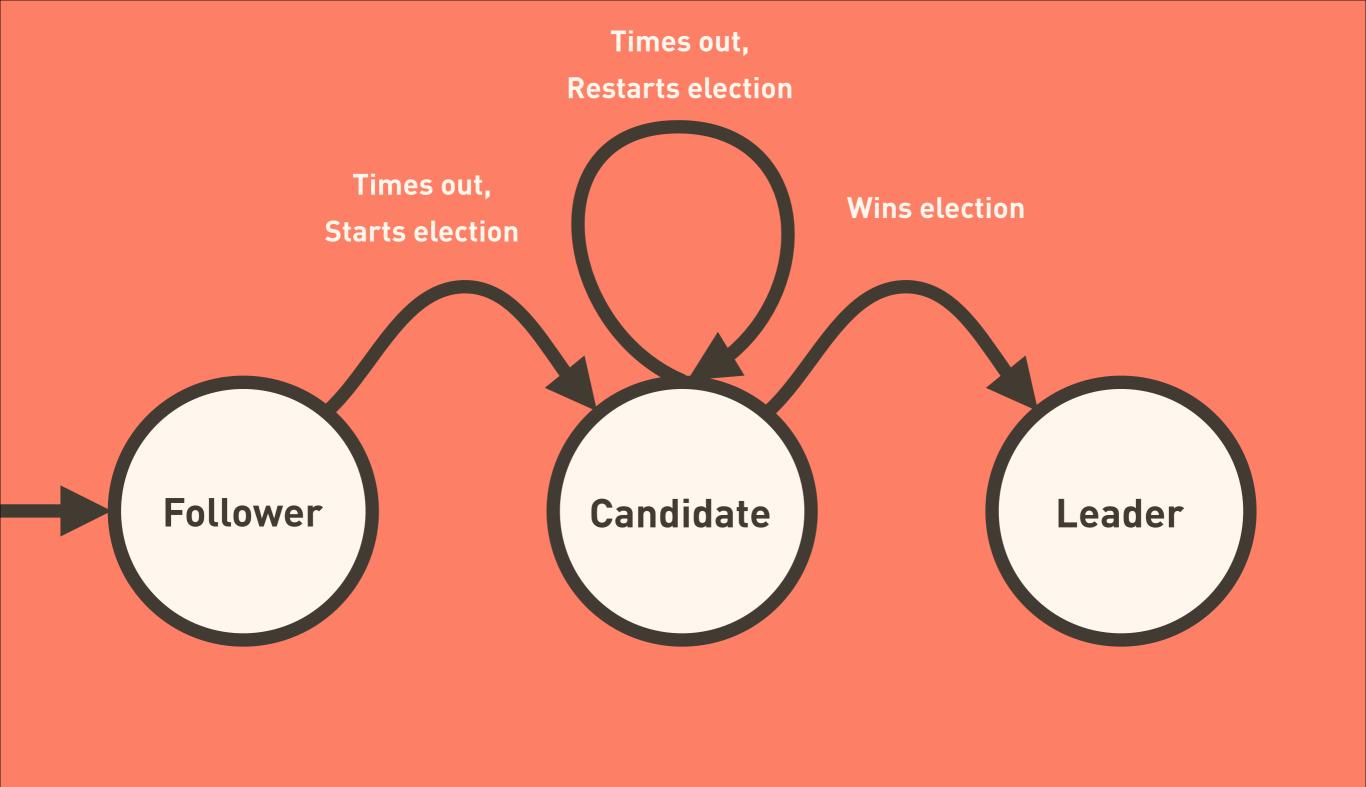


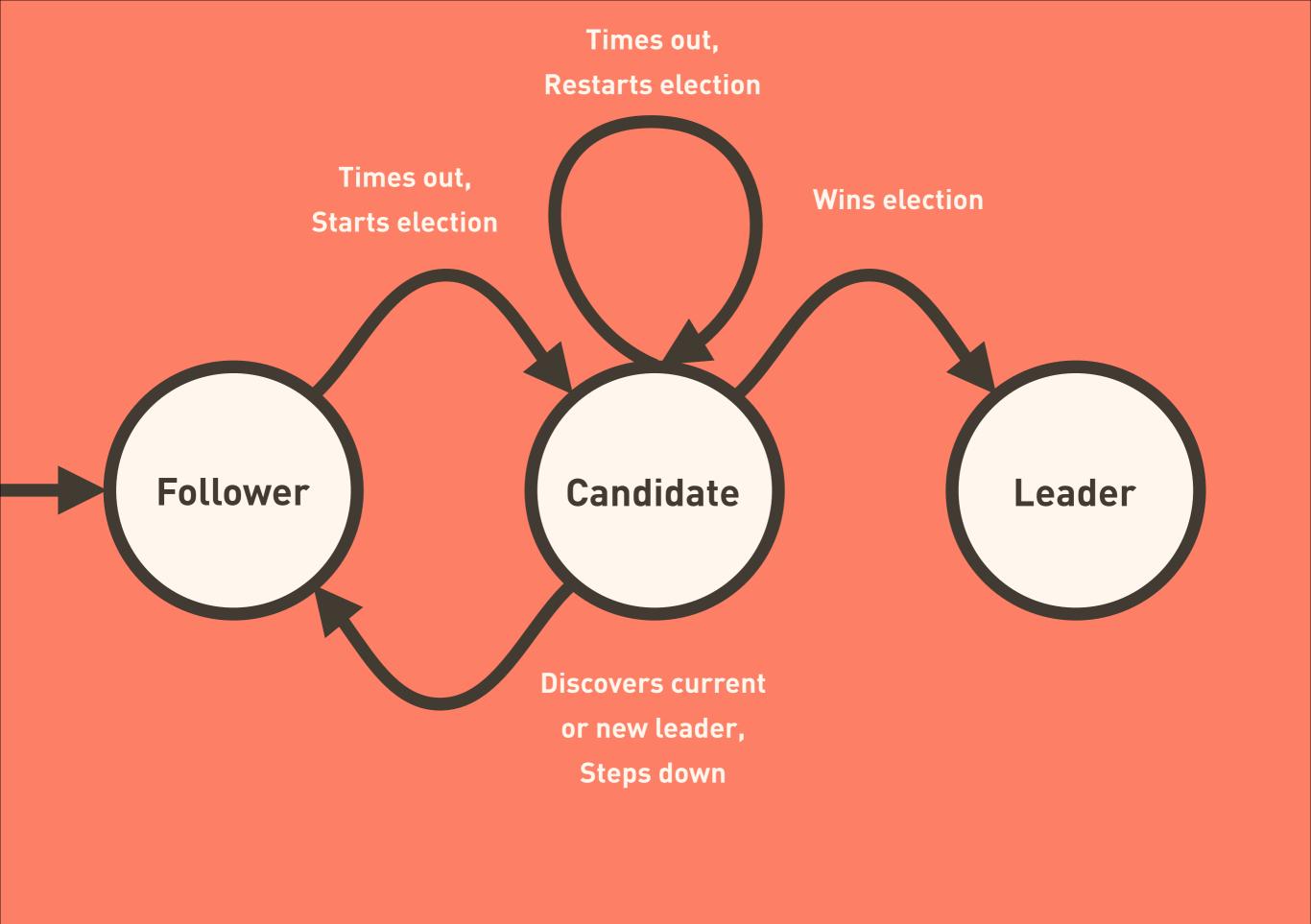
Times out,
Starts election

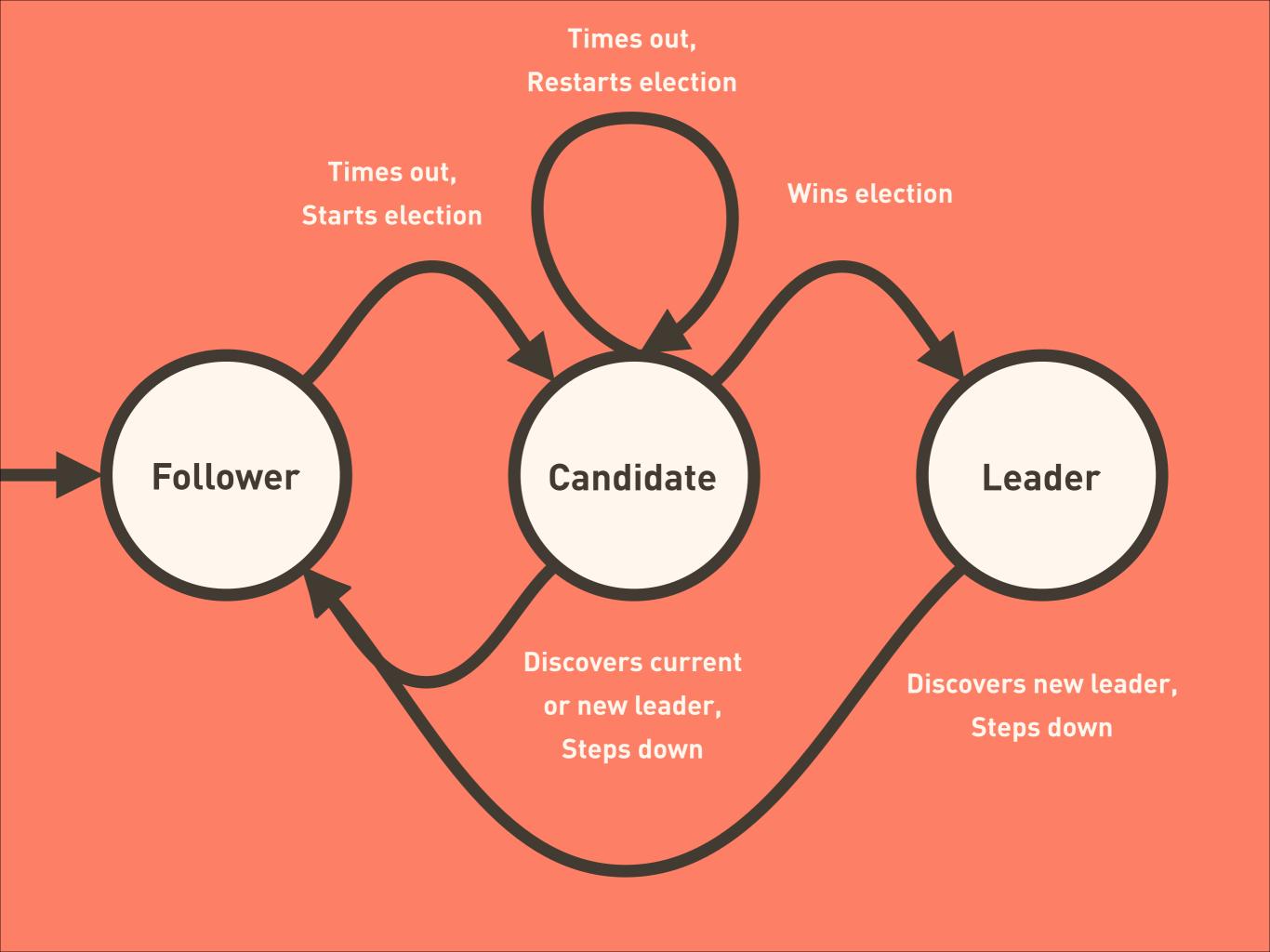
Candidate

Leader





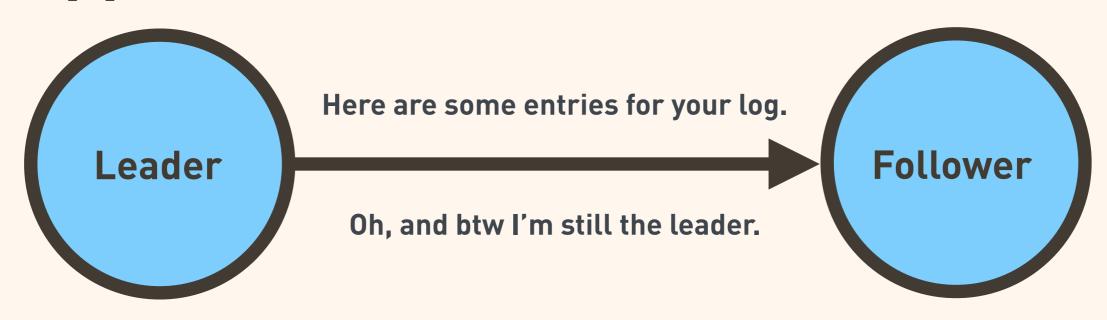




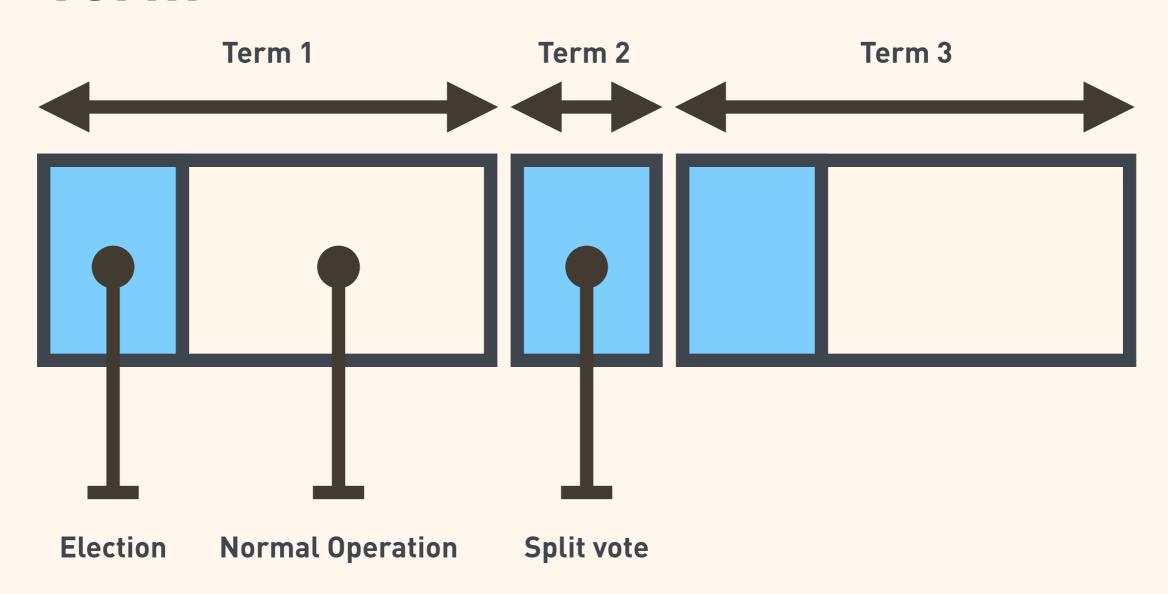
RequestVote

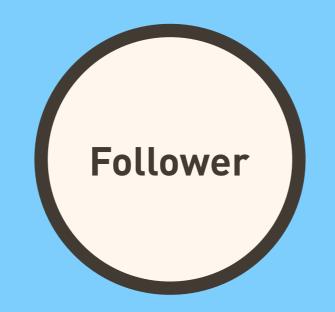


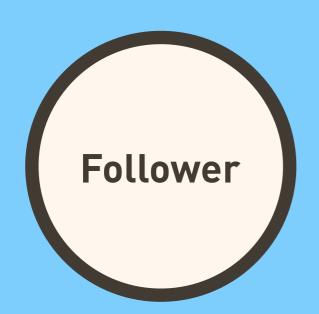
AppendEntries

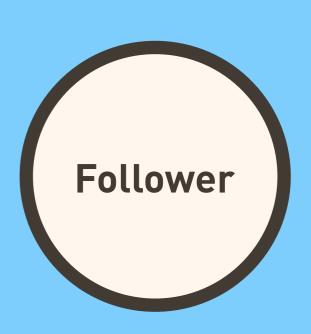


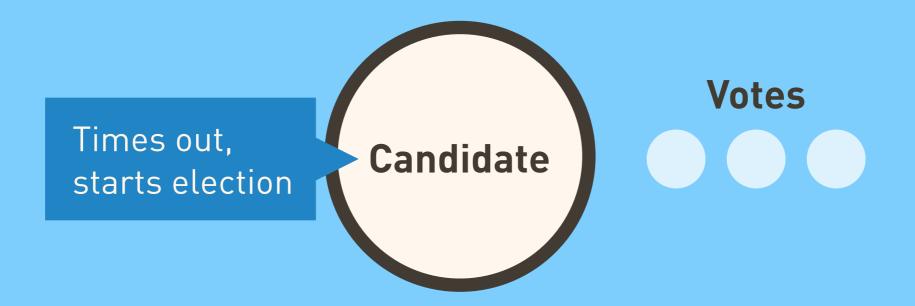
Term



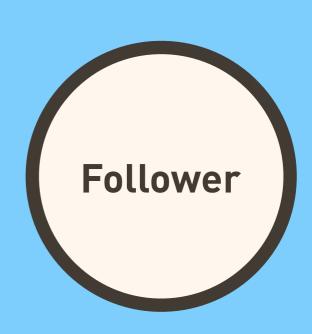


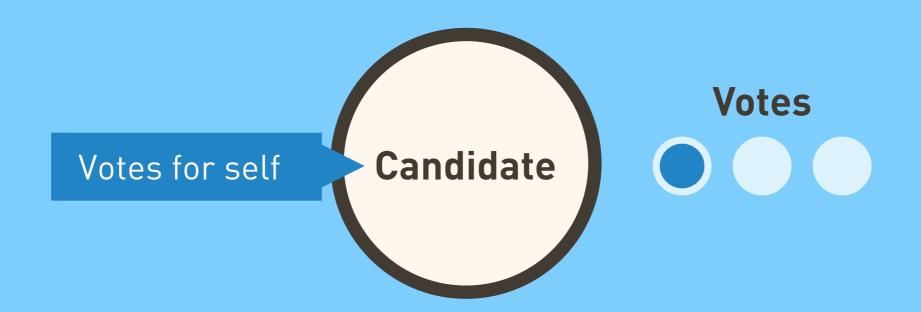




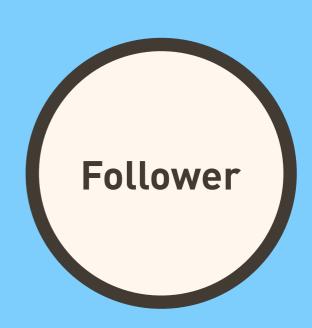


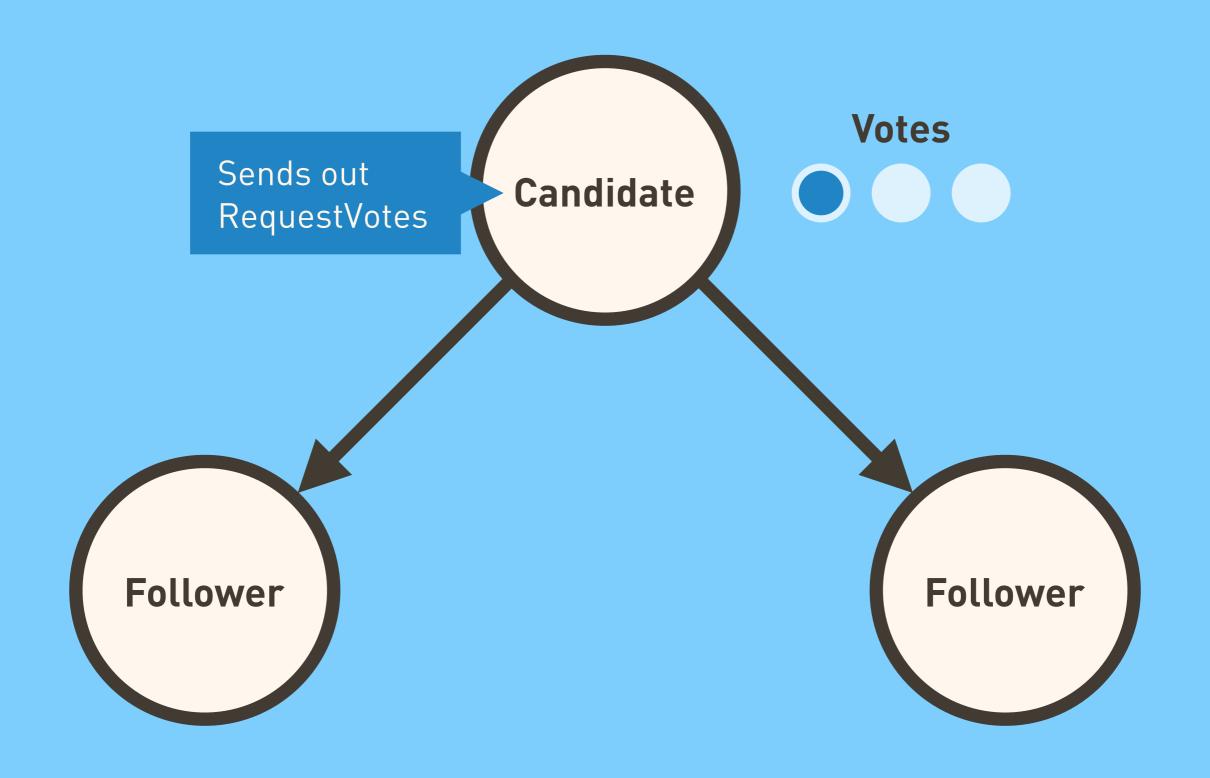


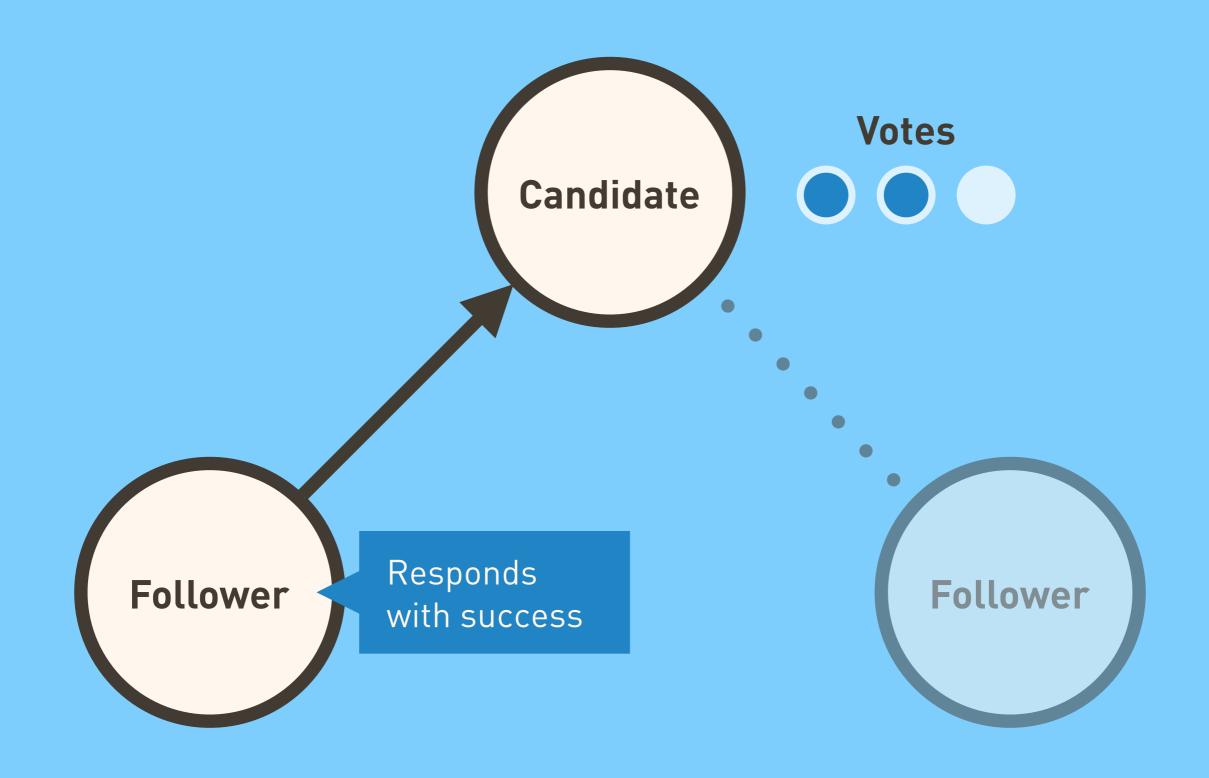


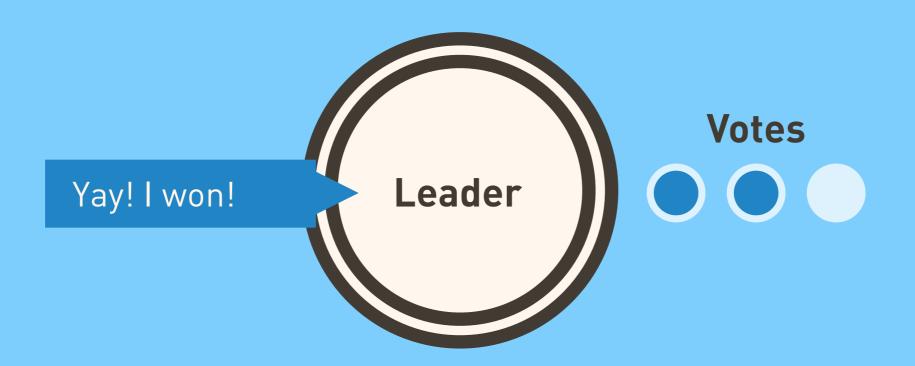














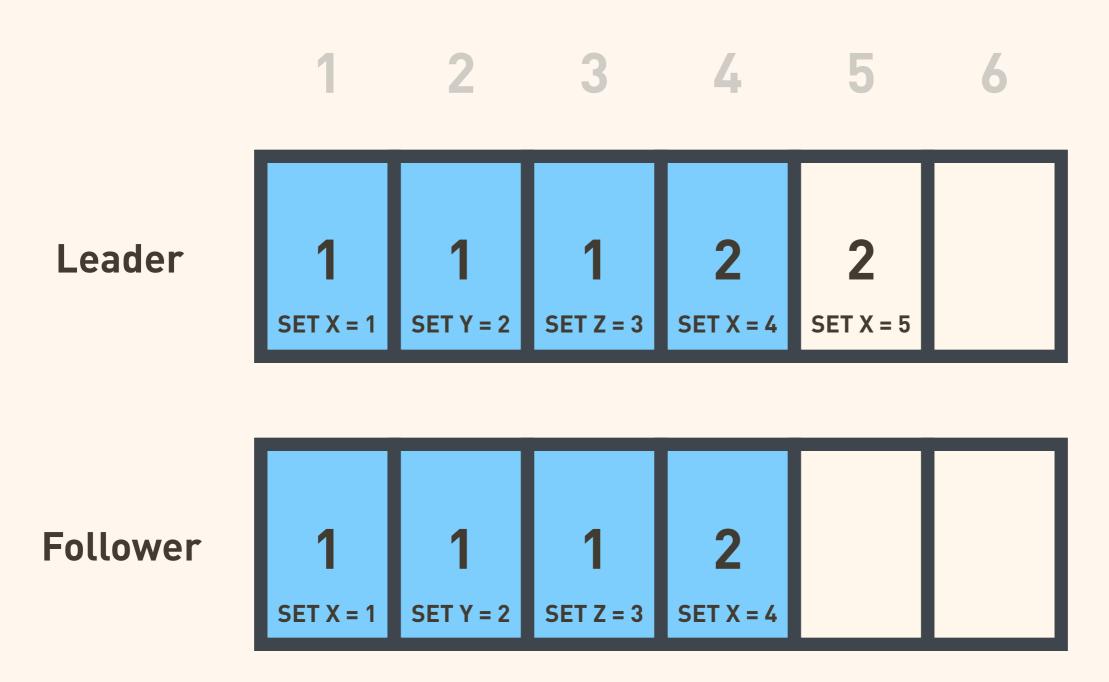


Leader Election

Log Replication

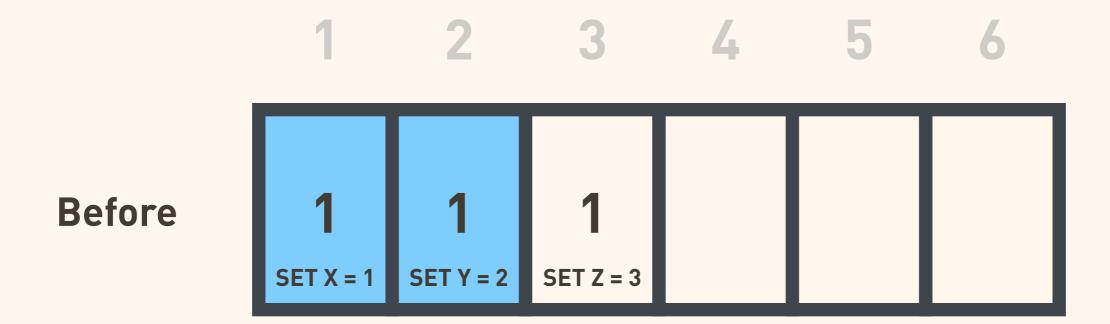
Safety

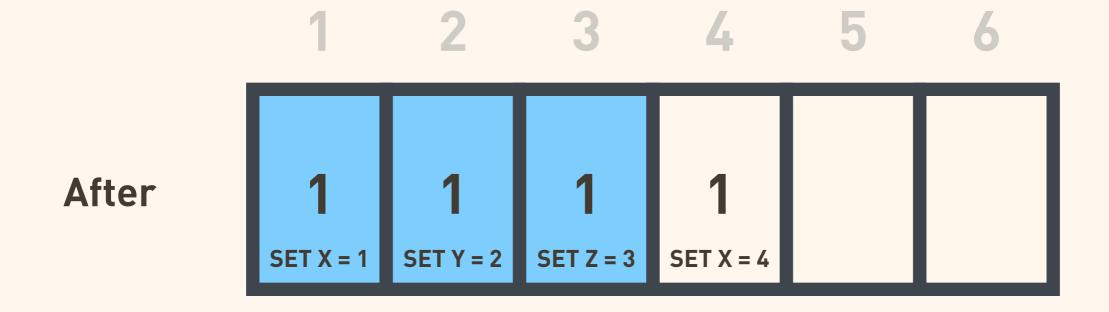
Log Entries Explained



APPEND ENTRIES EXAMPLE

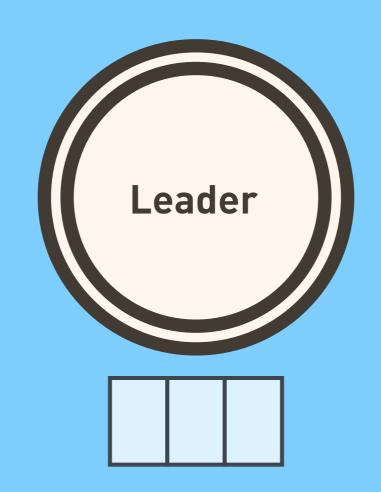
```
:entries => [{ 4 => 'SET X = 4' }],
:term => 1,
:prev_log_term => 1,
:prev_log_index => 3,
:leader_commit => 3,
:leader_id => '192.168.1.101/7238'
```

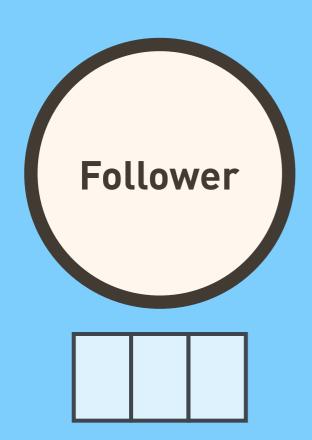


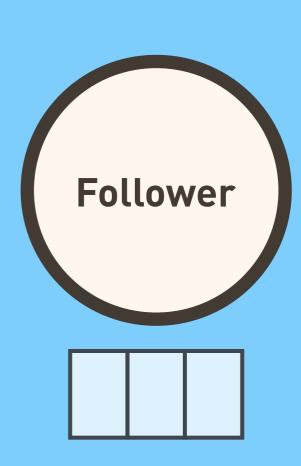


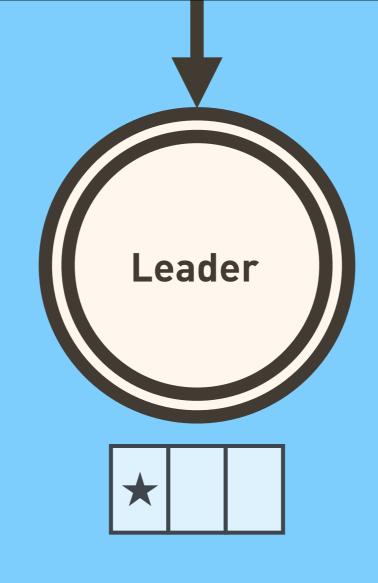
Log entries are always committed in the same order and are never uncommitted.

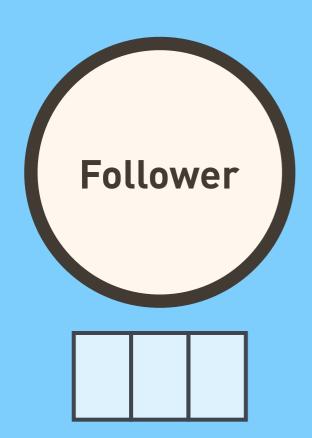
Happy Log Entry

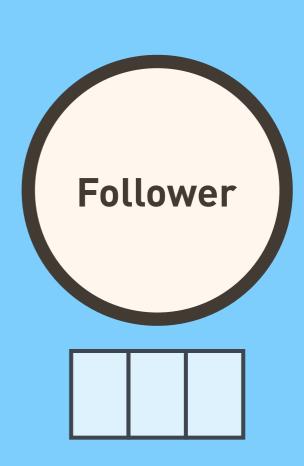


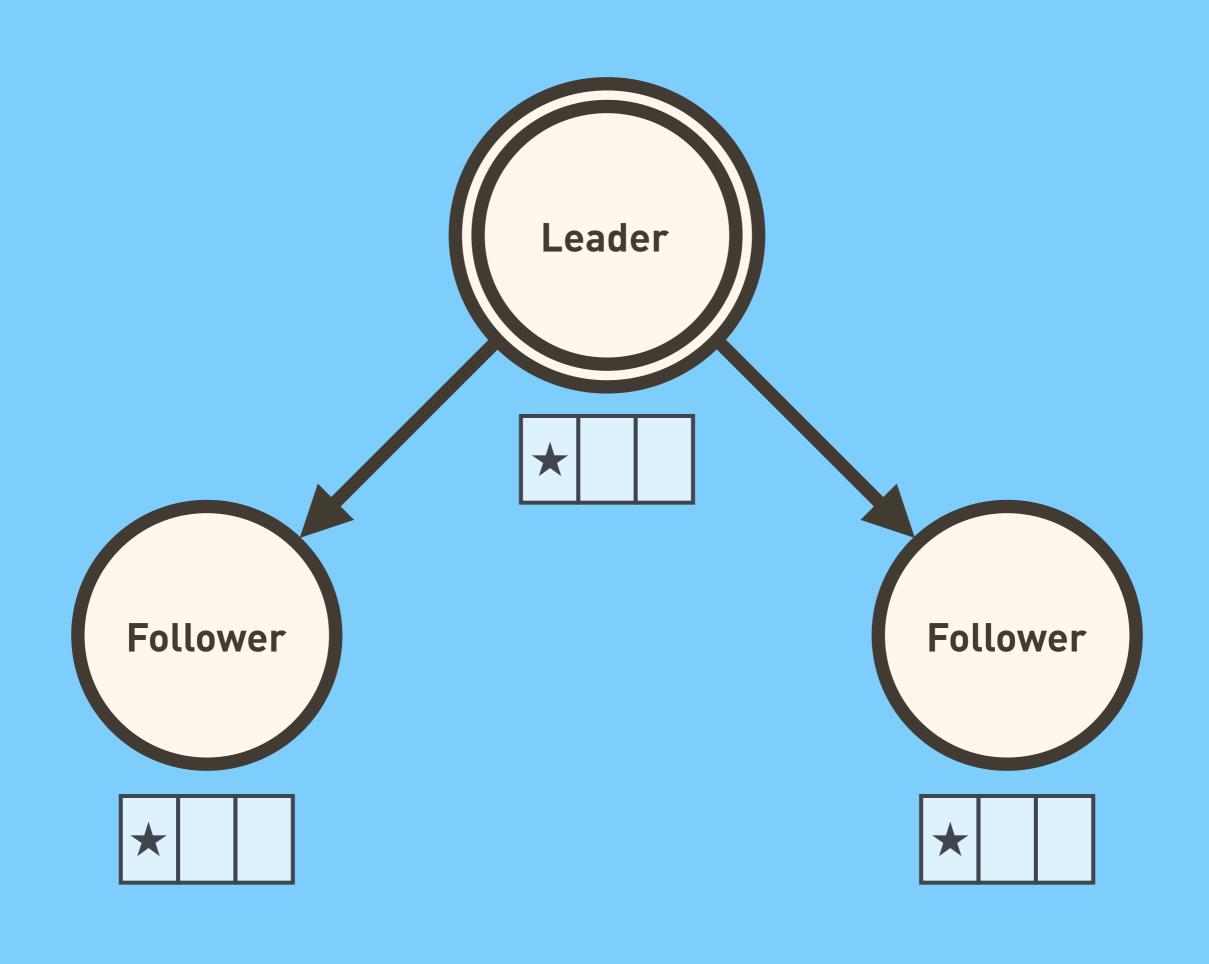


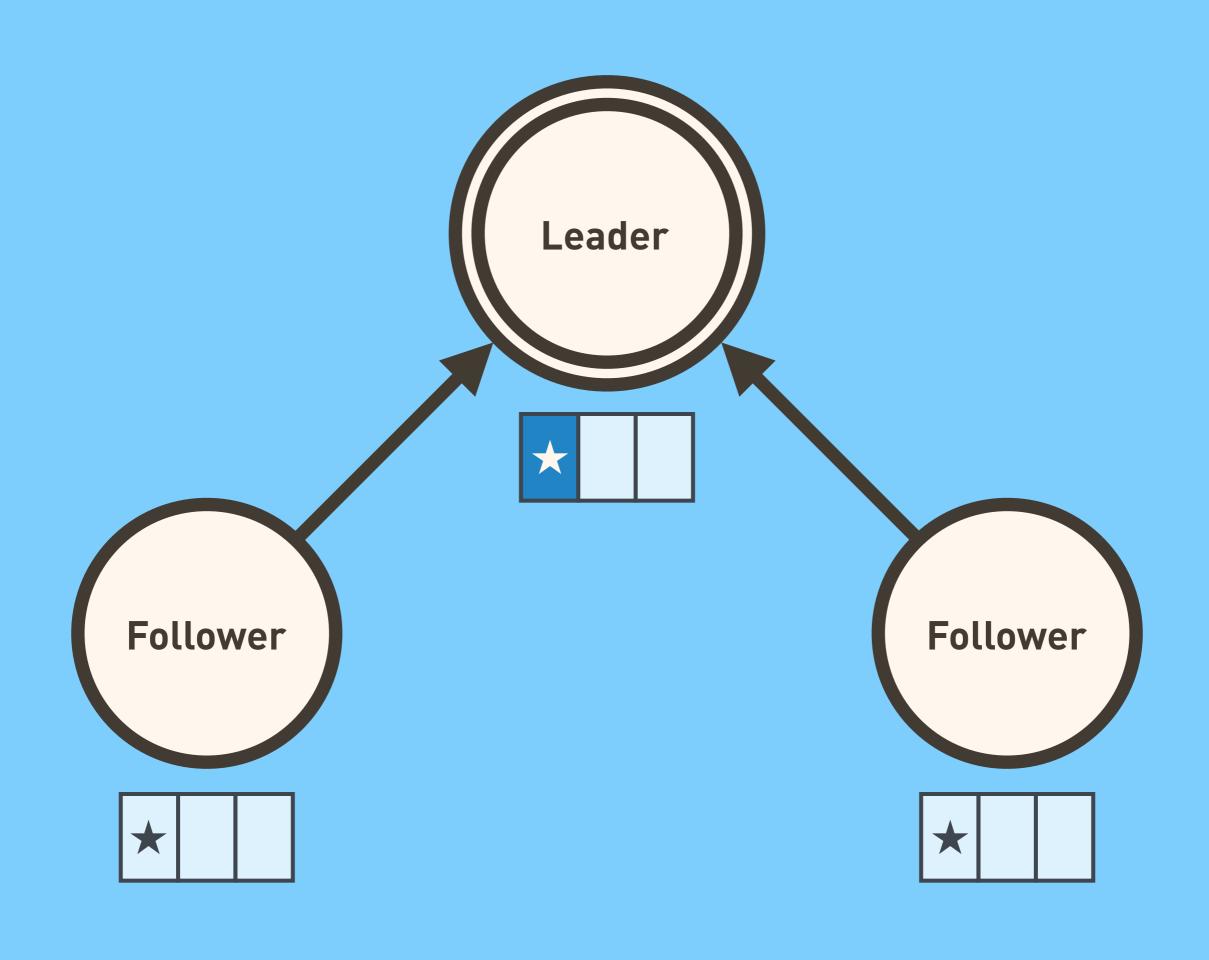


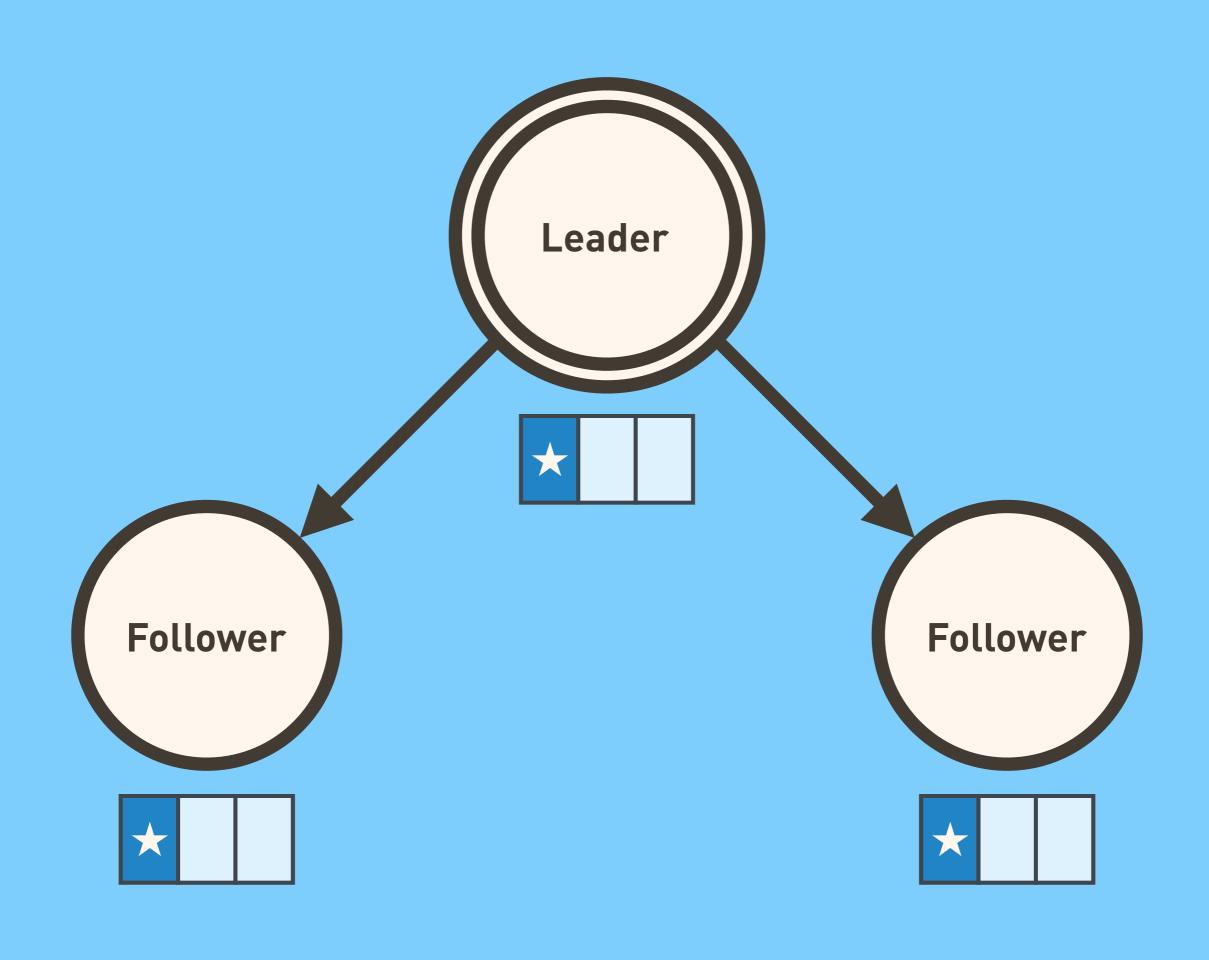




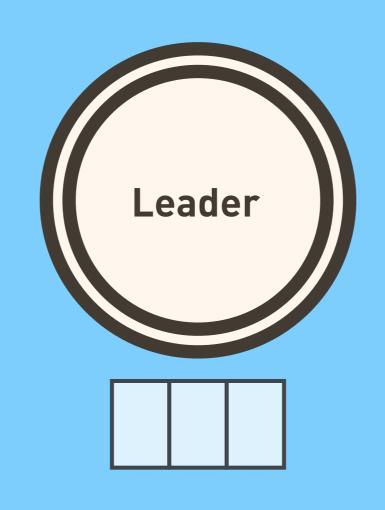


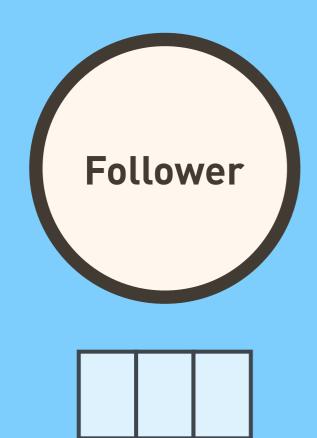


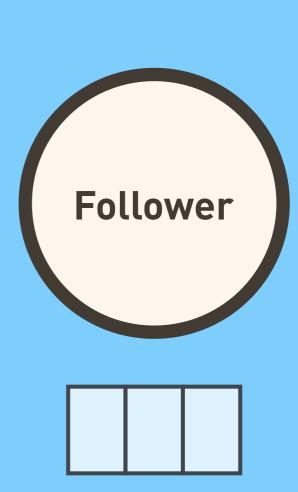


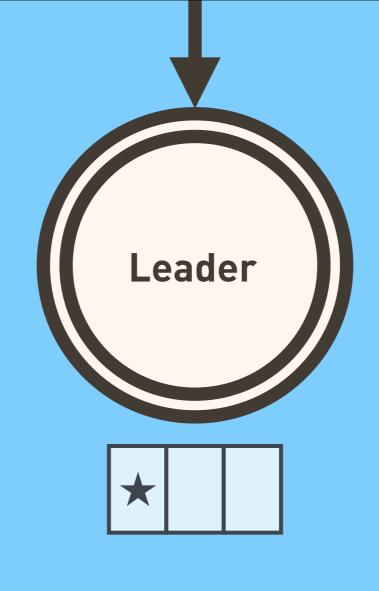


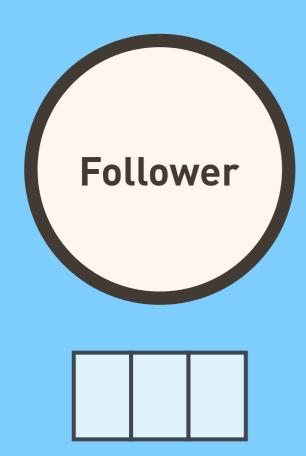
Sad Log Entry

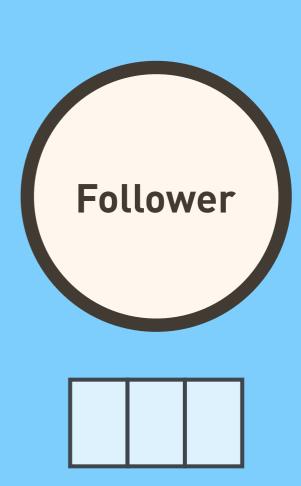


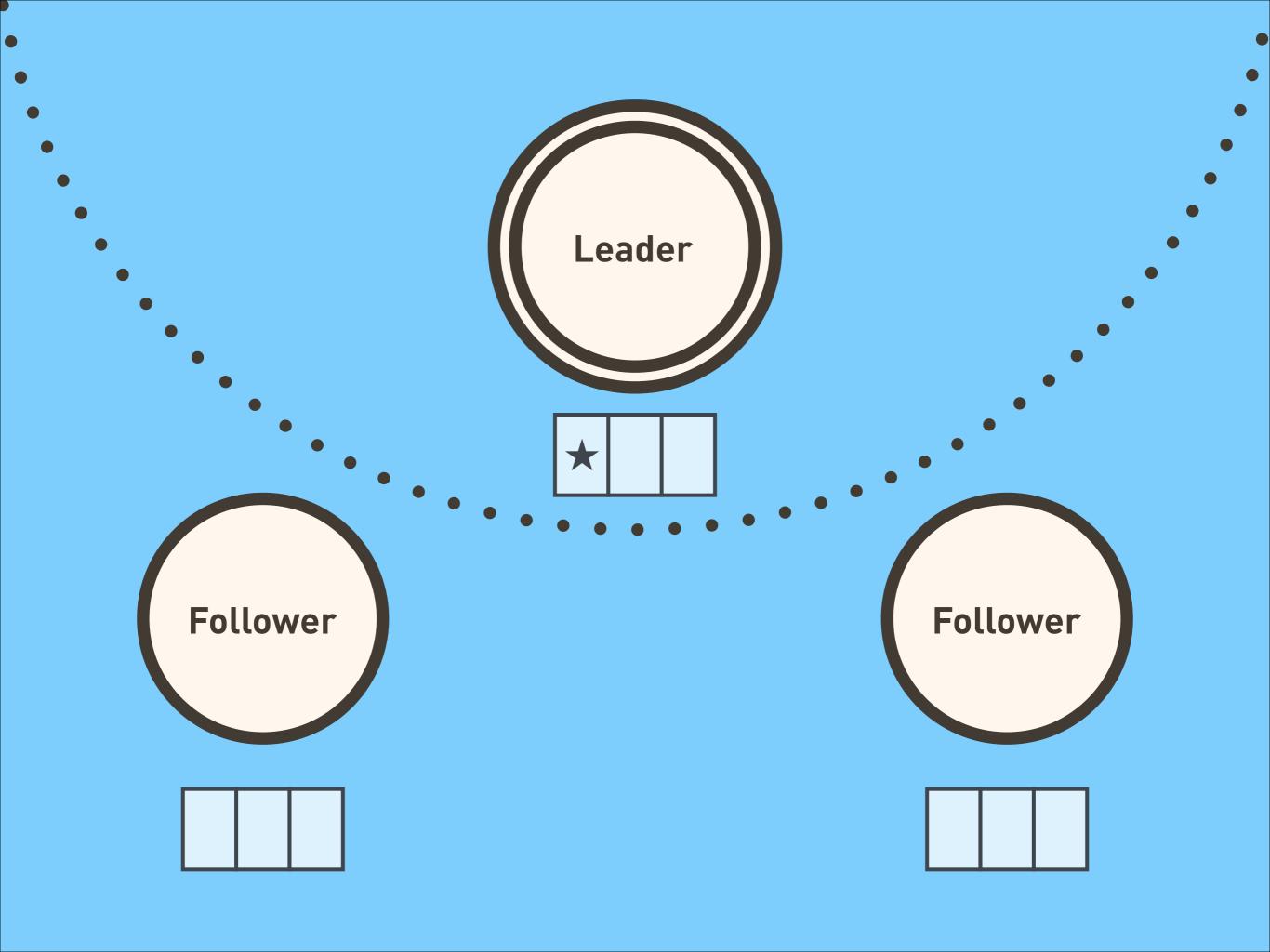


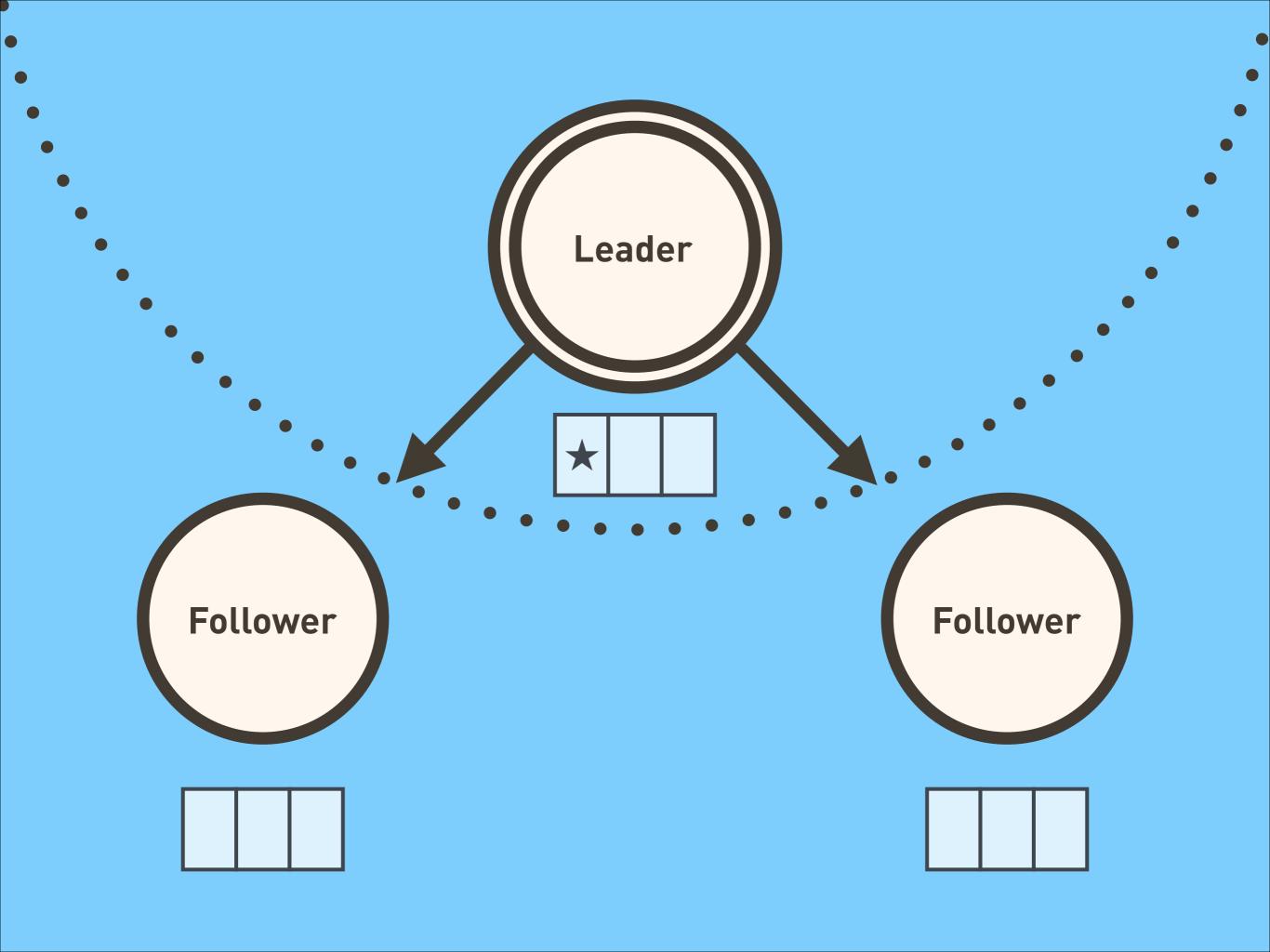


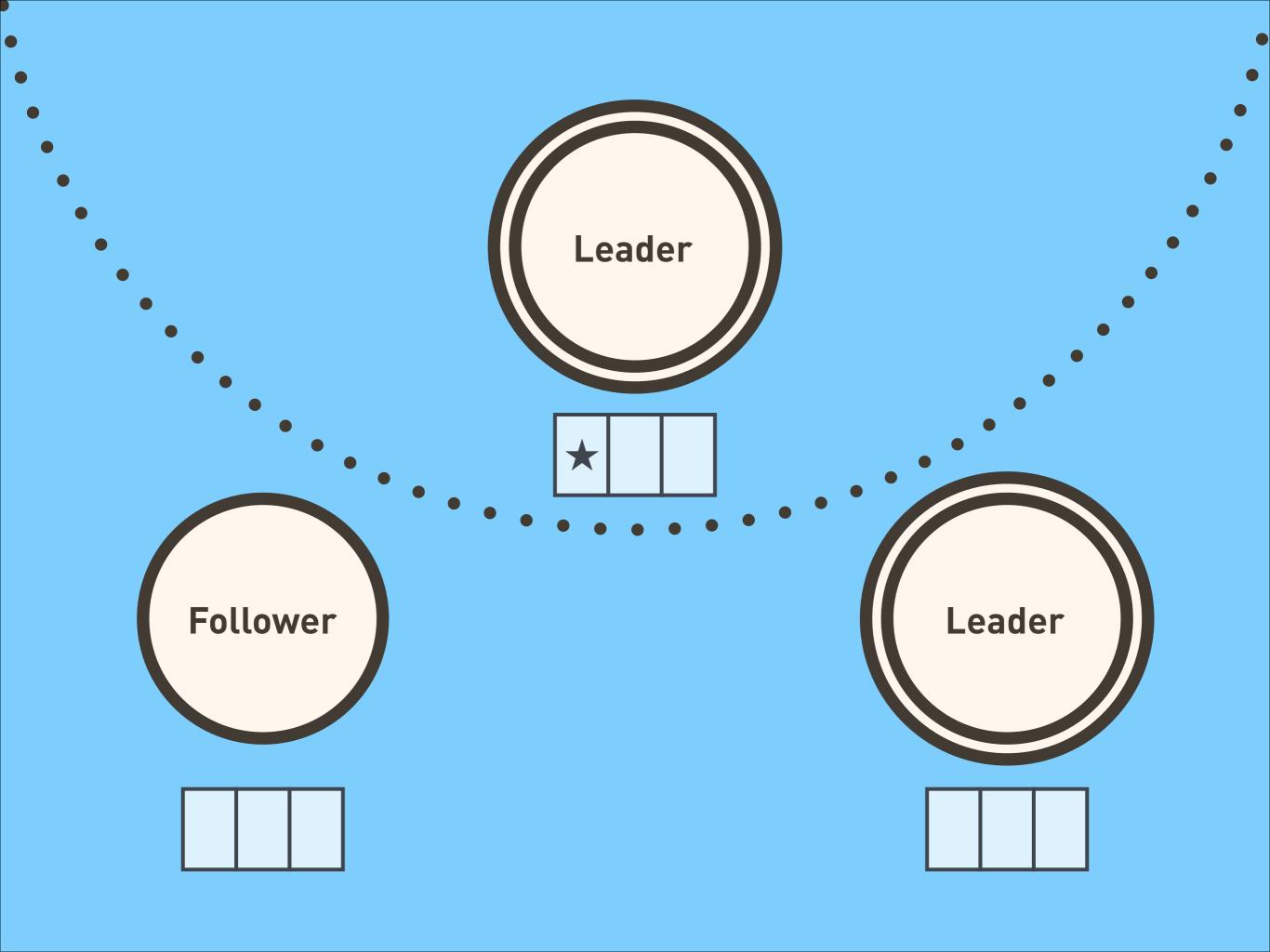


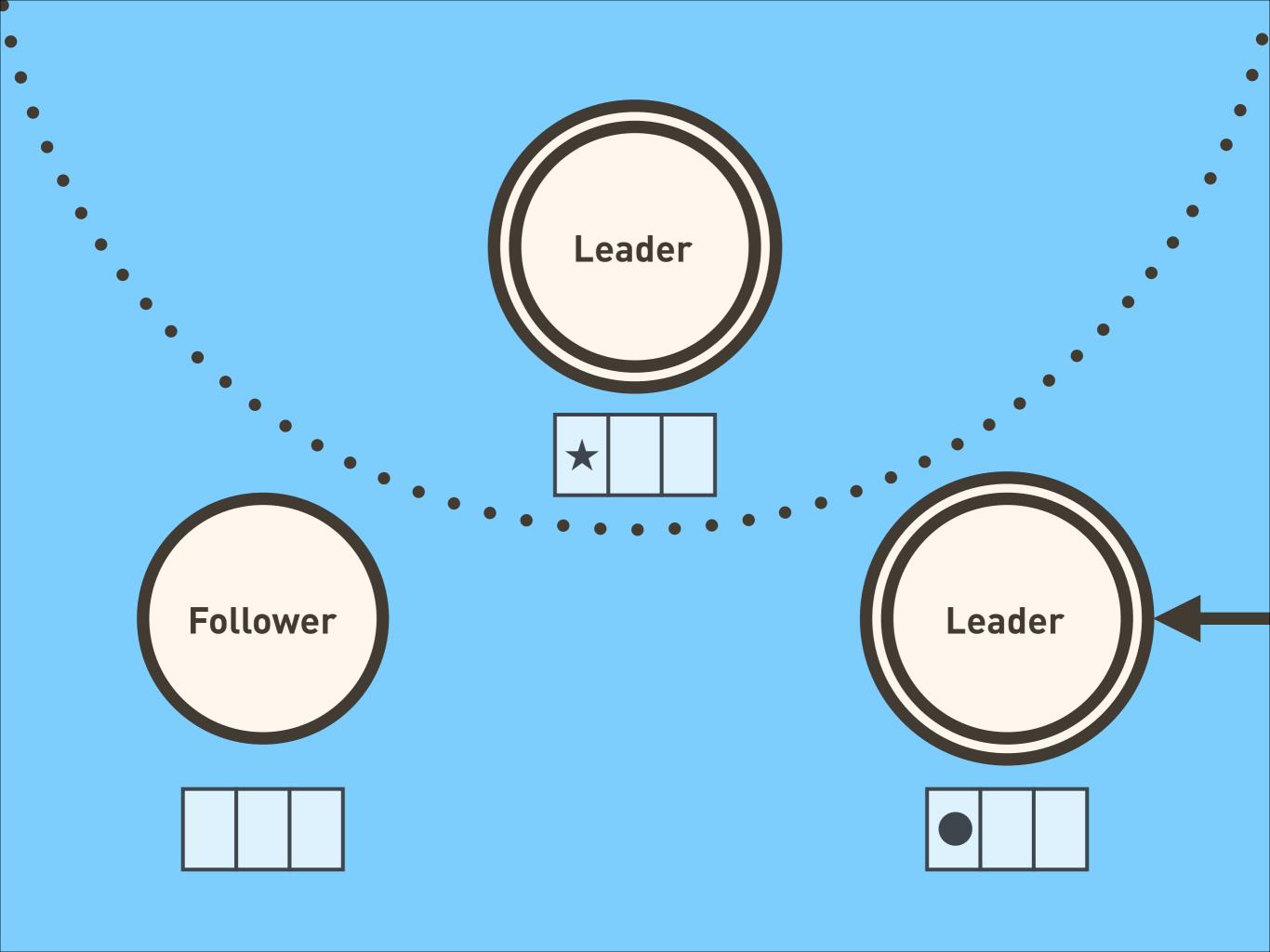


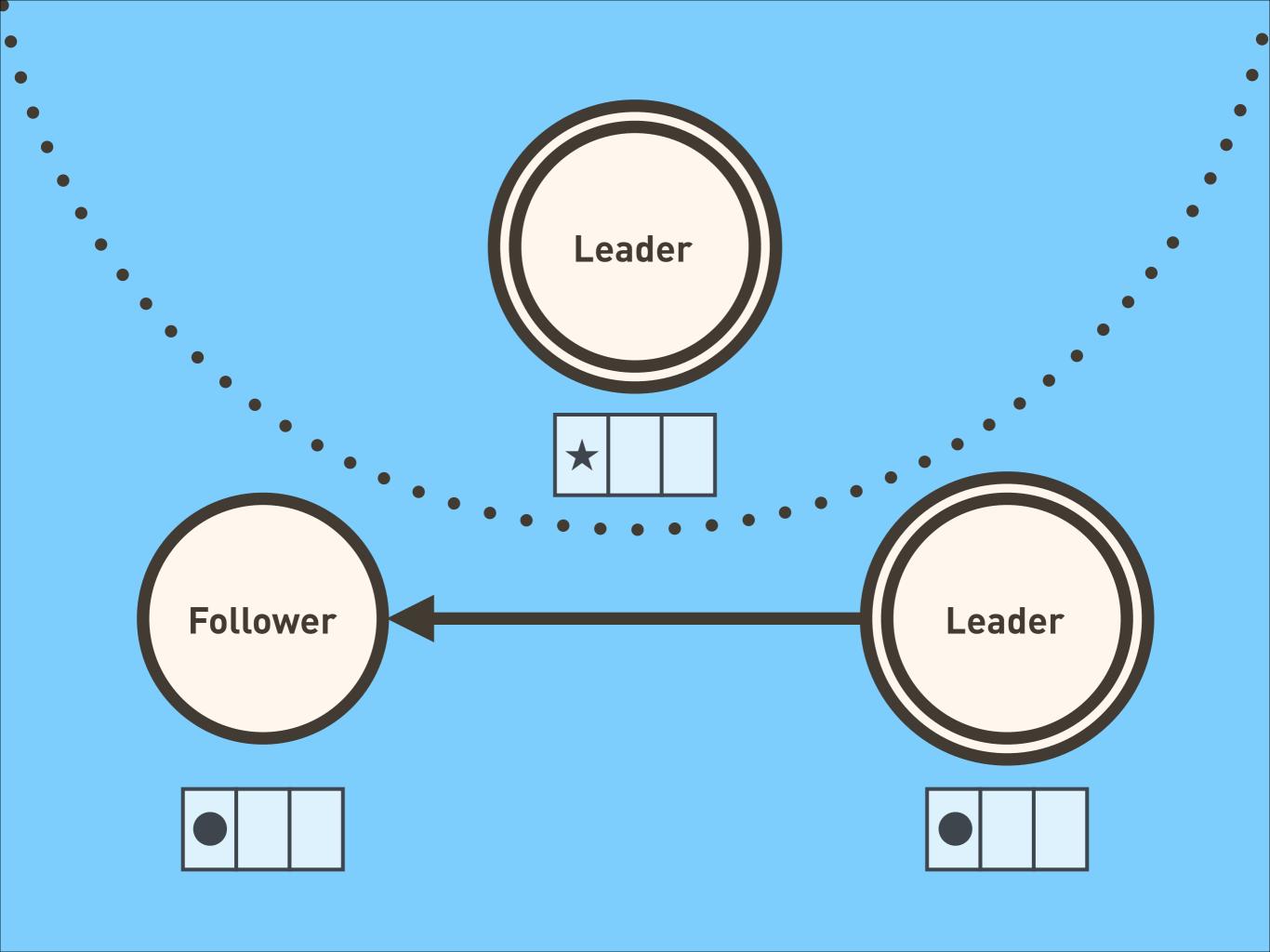


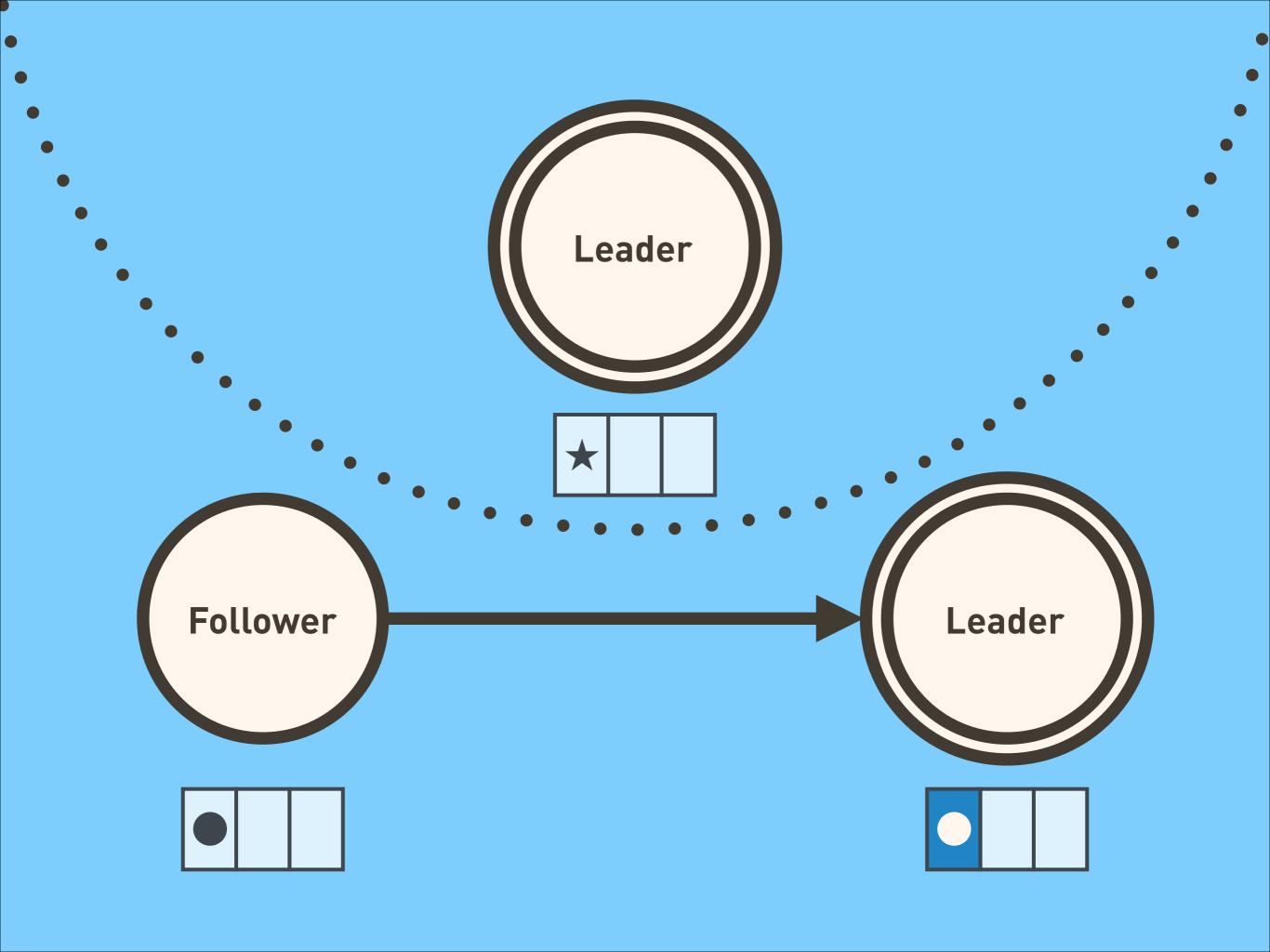


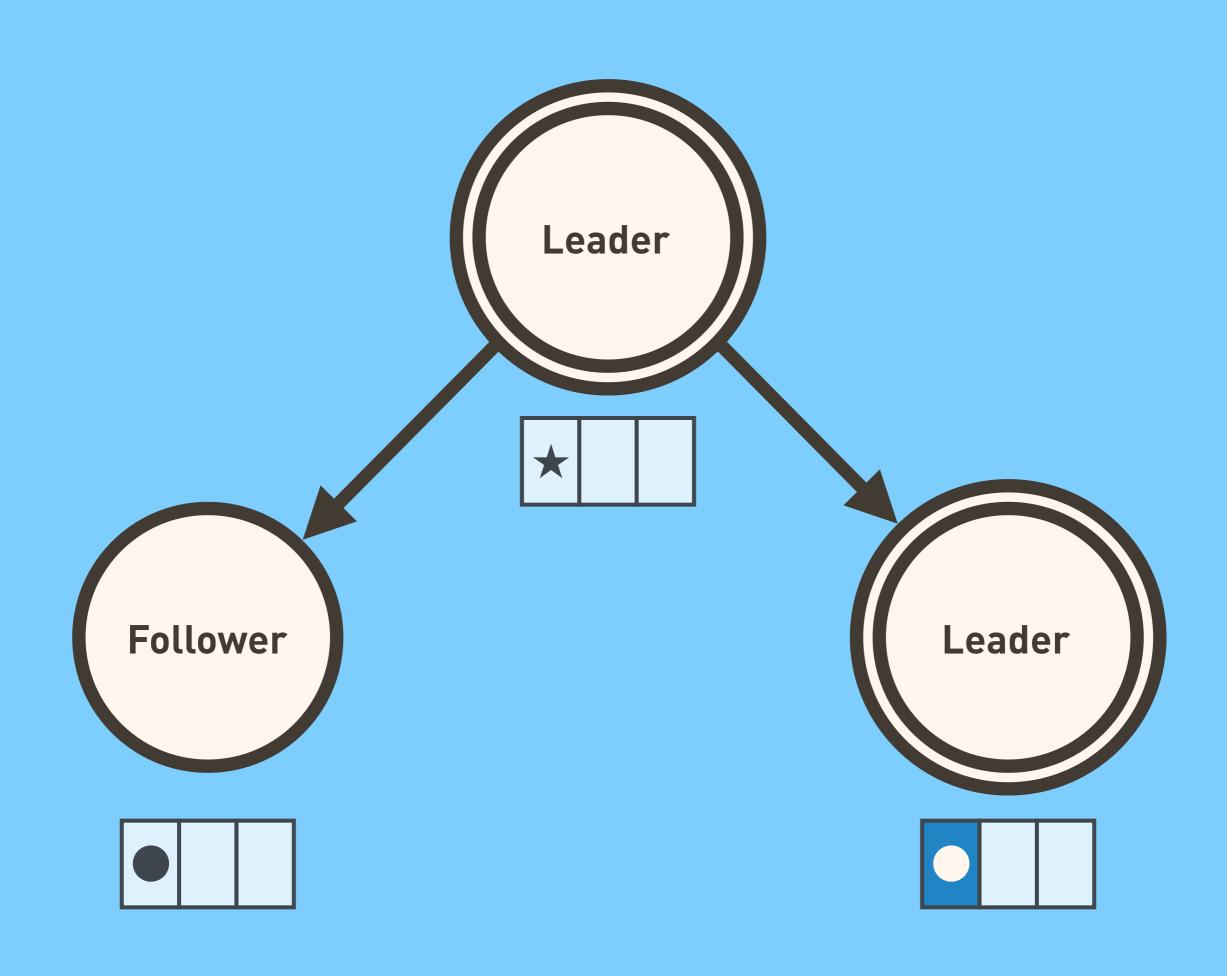


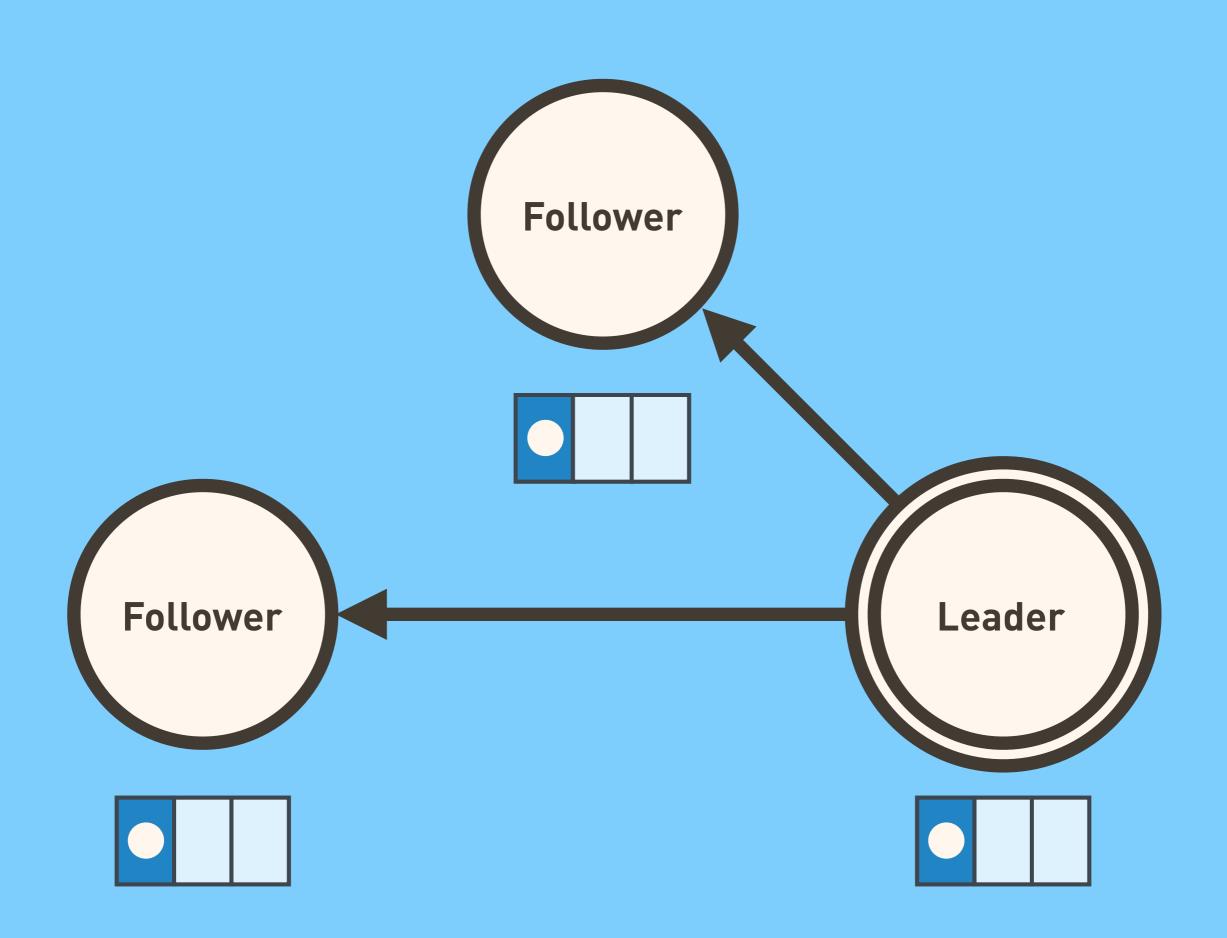




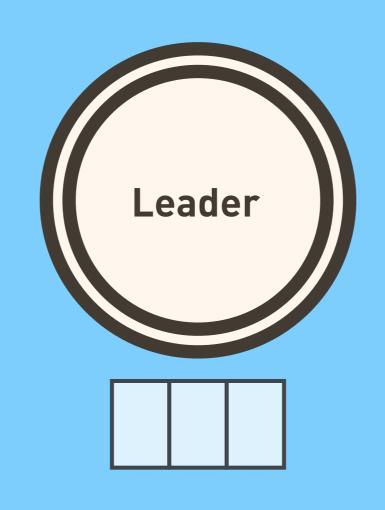


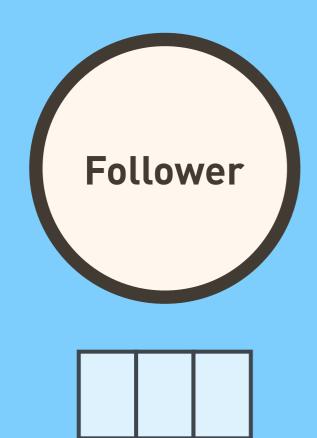


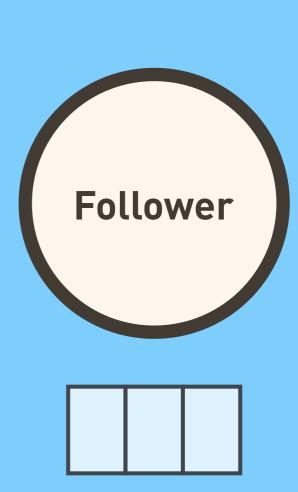


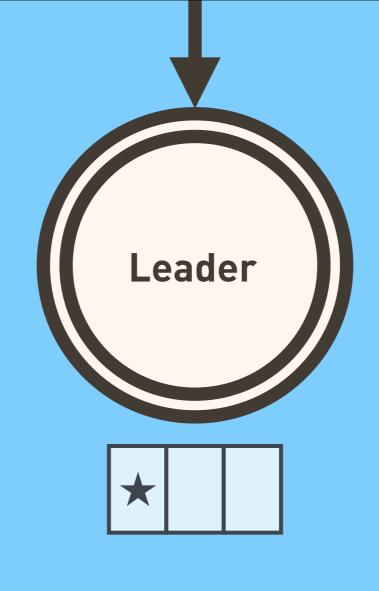


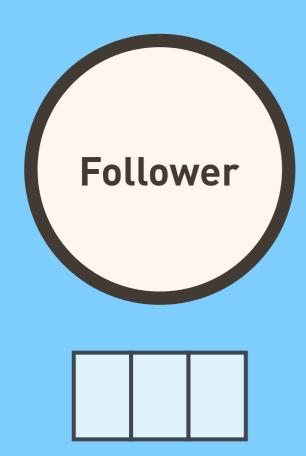
Facepalm Log Entry

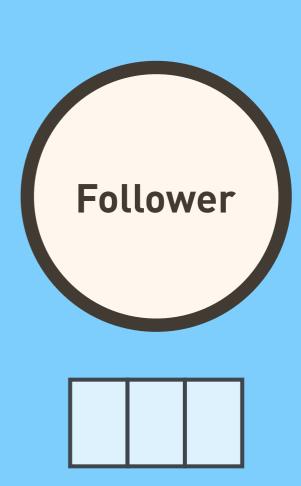


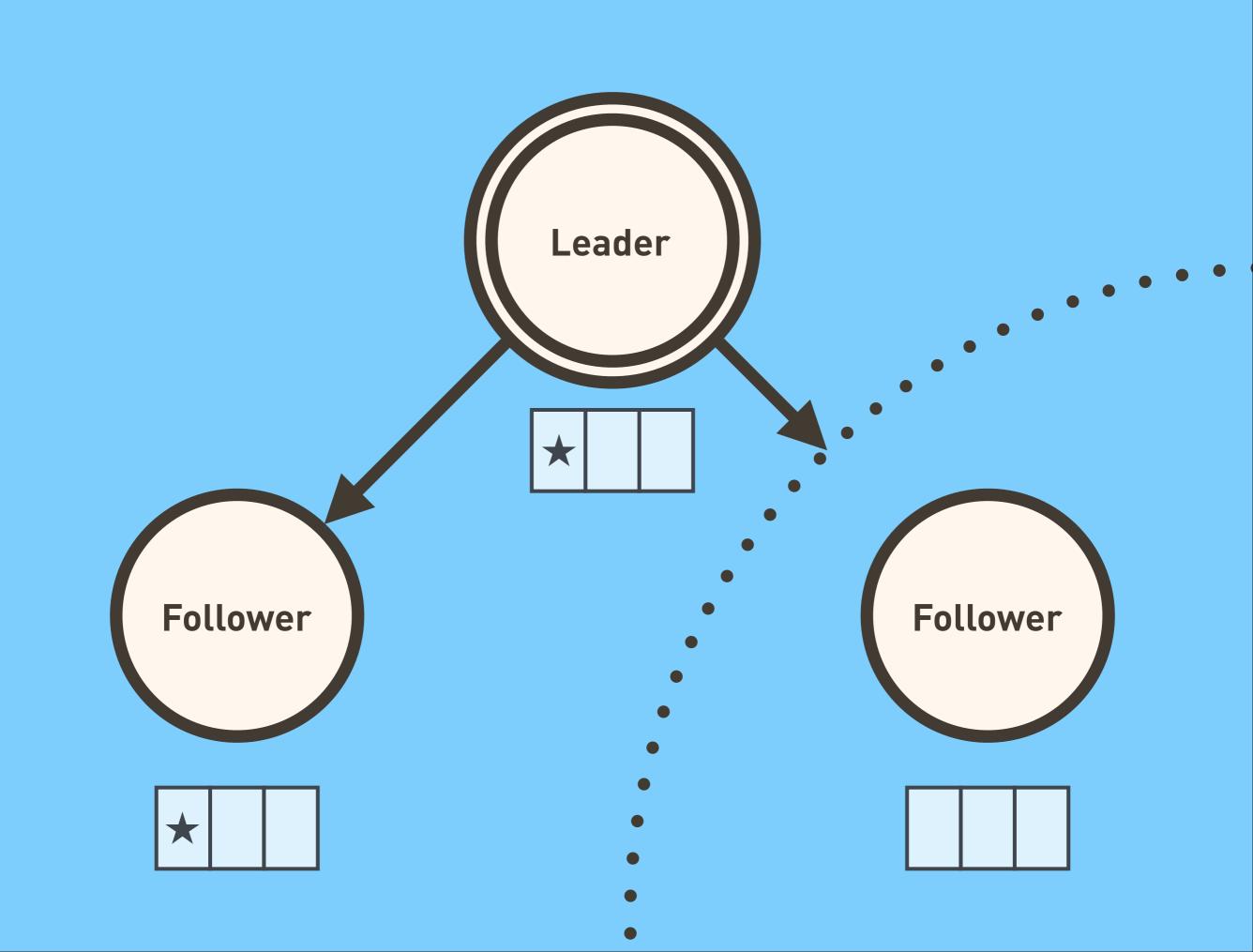


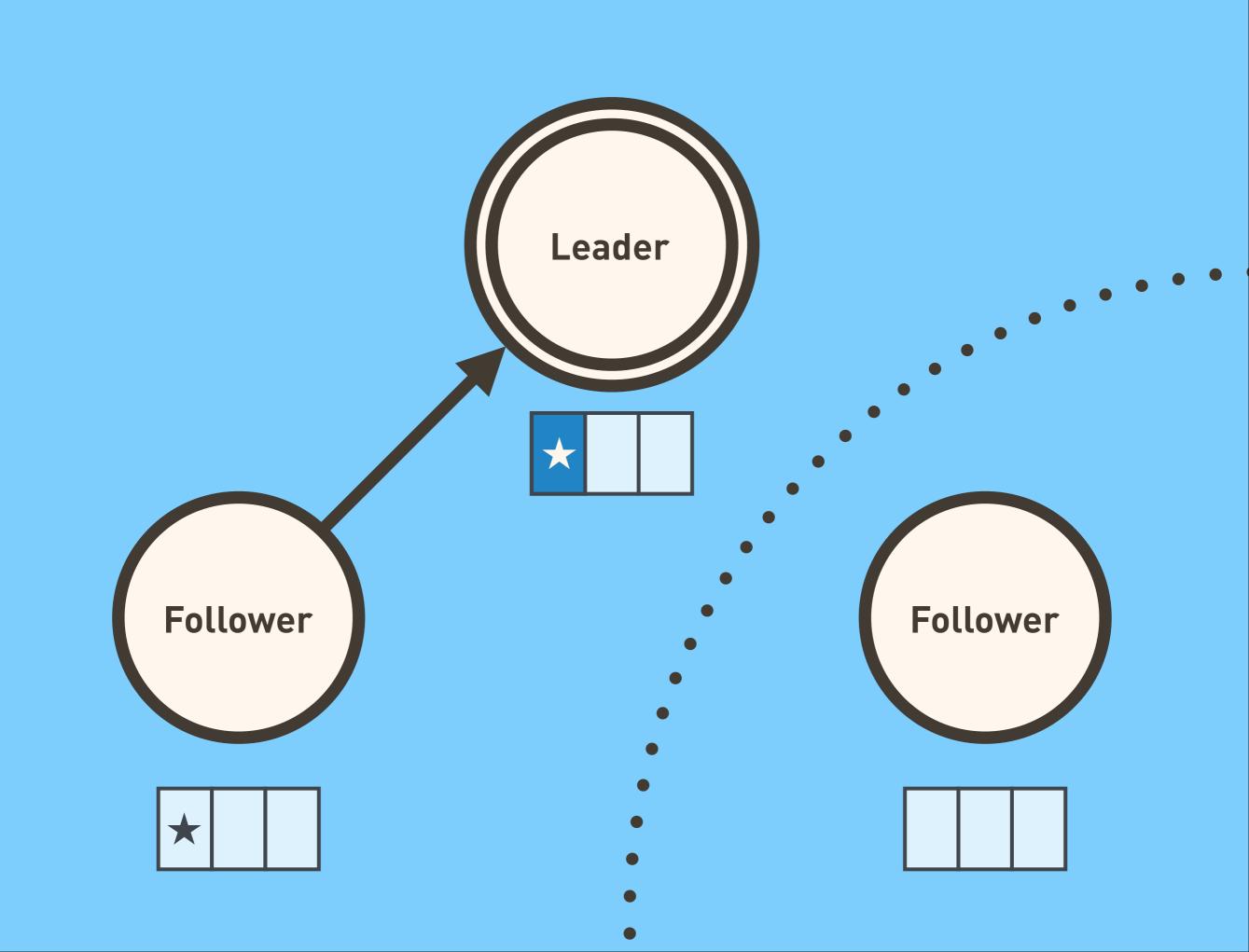


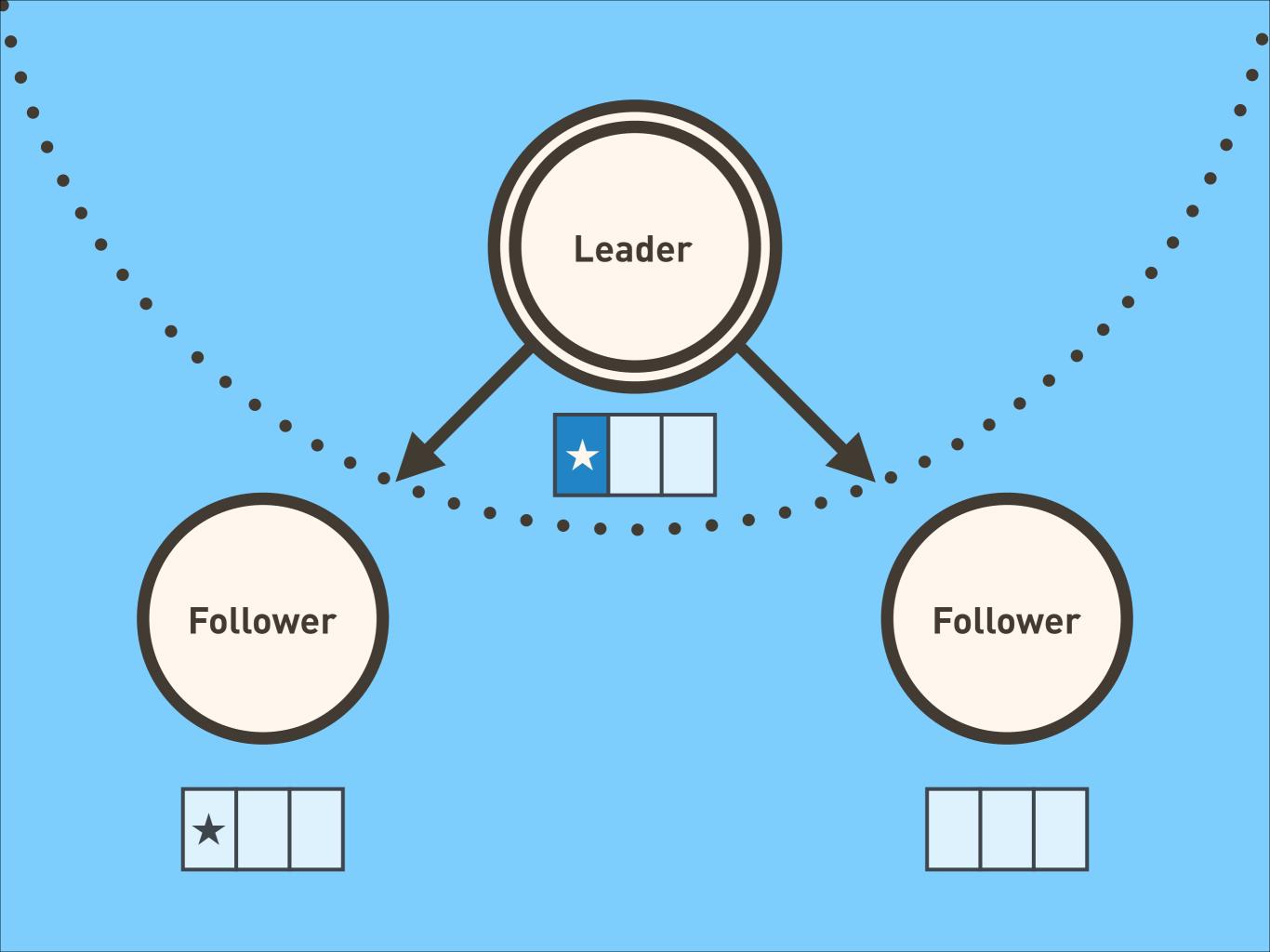


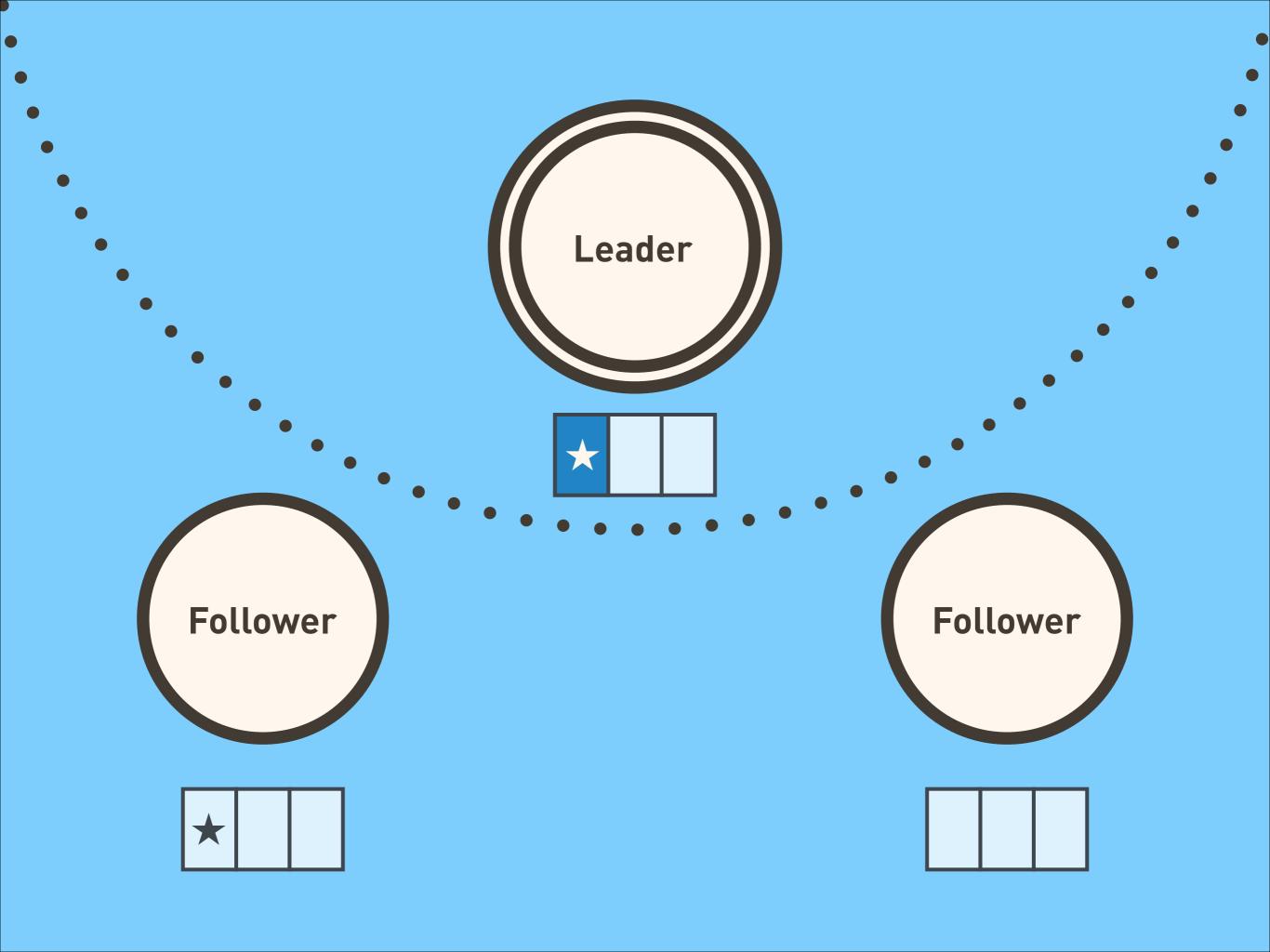


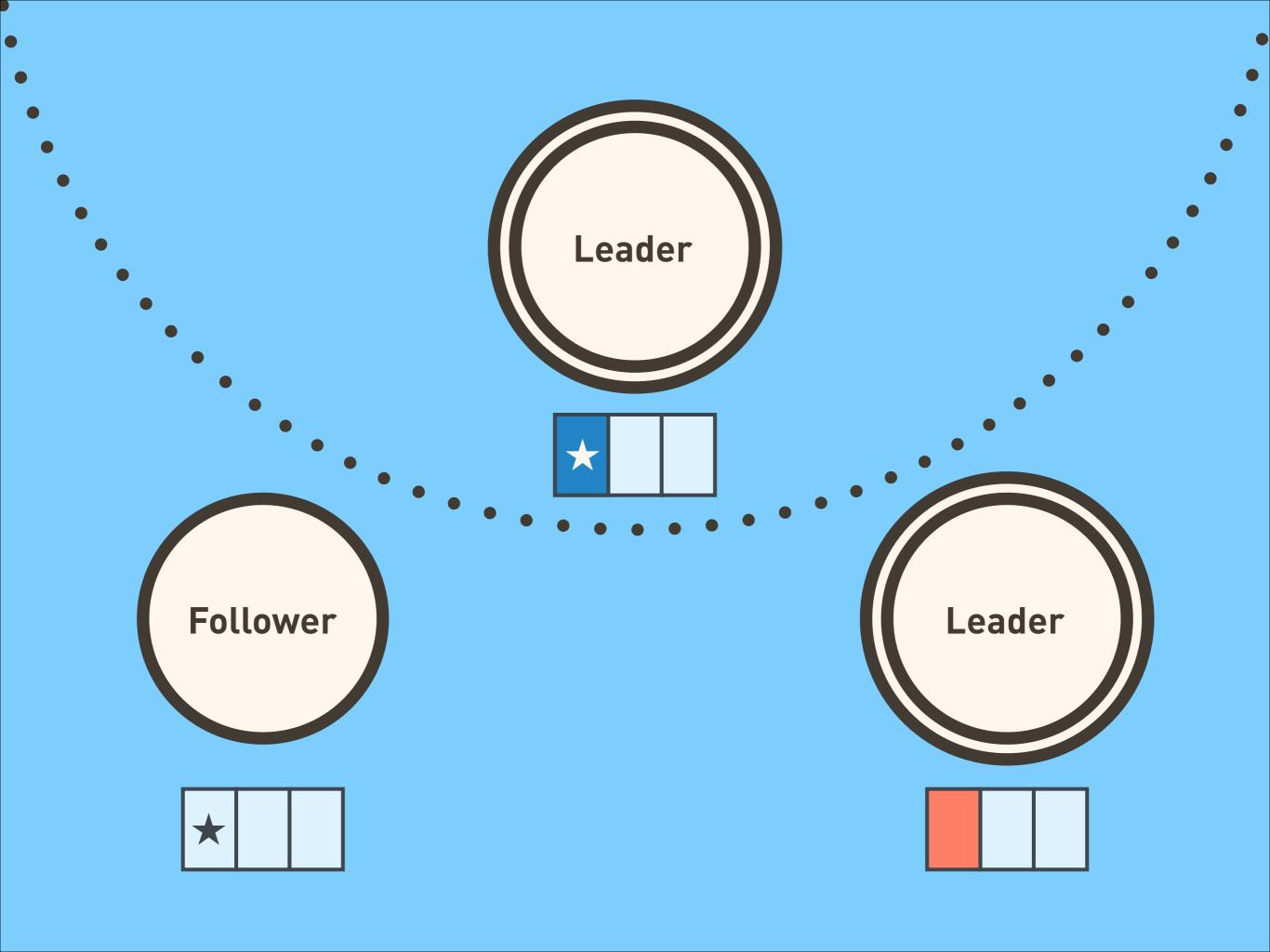












How do we guard against losing log entries?

Leader Election

Log Replication

Safety



Only cast votes for nodes with logs that contain at least as many entries as your own.

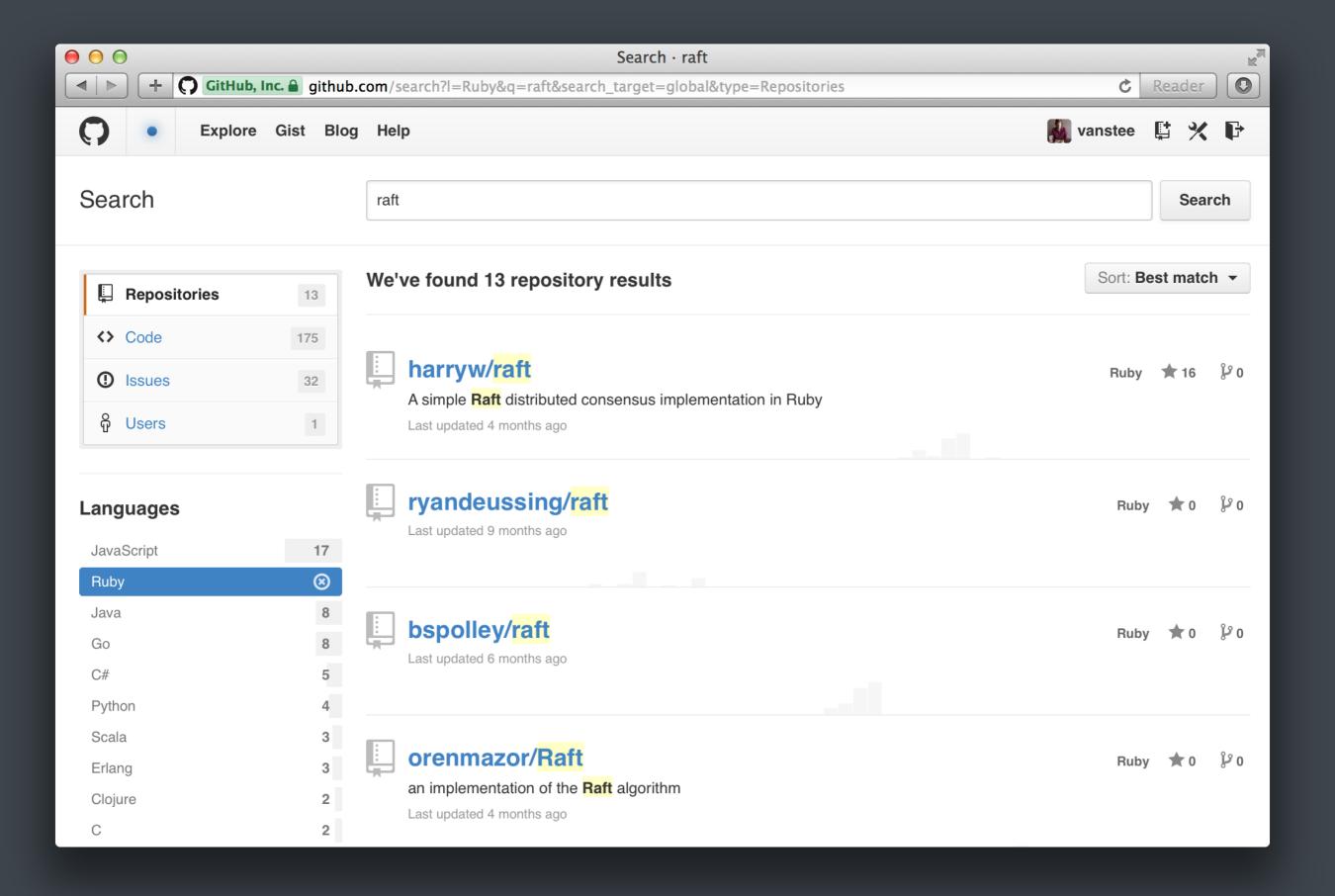


New leaders must commit a log entry form their new term before committing old entries.

Bonus Round

- Cluster changes
- Log compaction
- Client specifics

Mhy Ruby



Ruby is great for expressing complex problems.

Then why doesn't the academic community love it?

- Community
- Understandability
- Learning Material

- Read Papers
- Go To Conferences
- Talk To People

Raft Paper

Raft Implementations

Raft Website

ThinkDistributed

Ricon

Google Scholar

Question Time

Make sure I repeat your questions.