Raft

The Understandable Distributed Consensus Protocol

@benbjohnson

What is Distributed Consensus?

Distributed = Many nodes

Consensus = Agreement

Distributed = Many nodes

Consensus = Agreement

Data Replication

Leader Election

Distributed Locks

A Really Short History Of Distributed Consensus Protocols

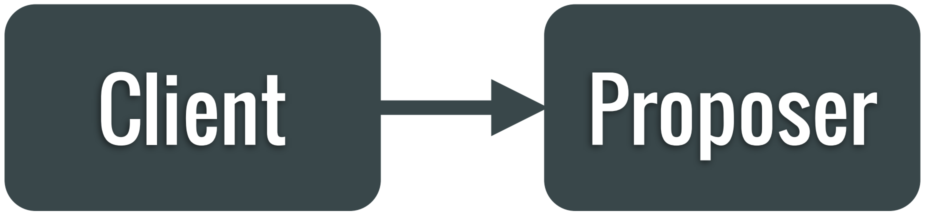
A Really Short History Of Distributed Consensus Protocols

Paxos (1989)

Paxos In A Nutshell

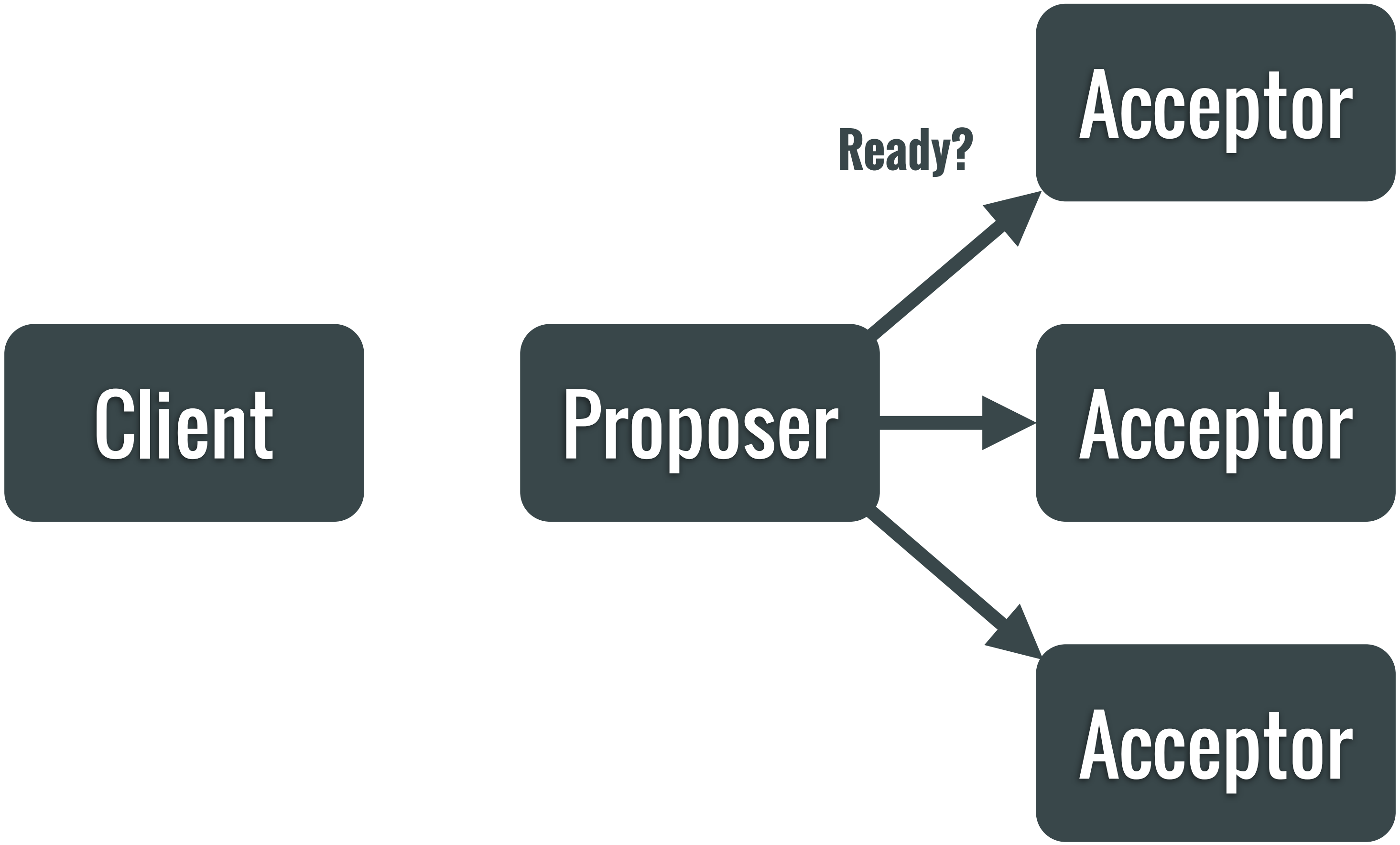
Paxos In A Nutshell

Paxos In A Nutshell



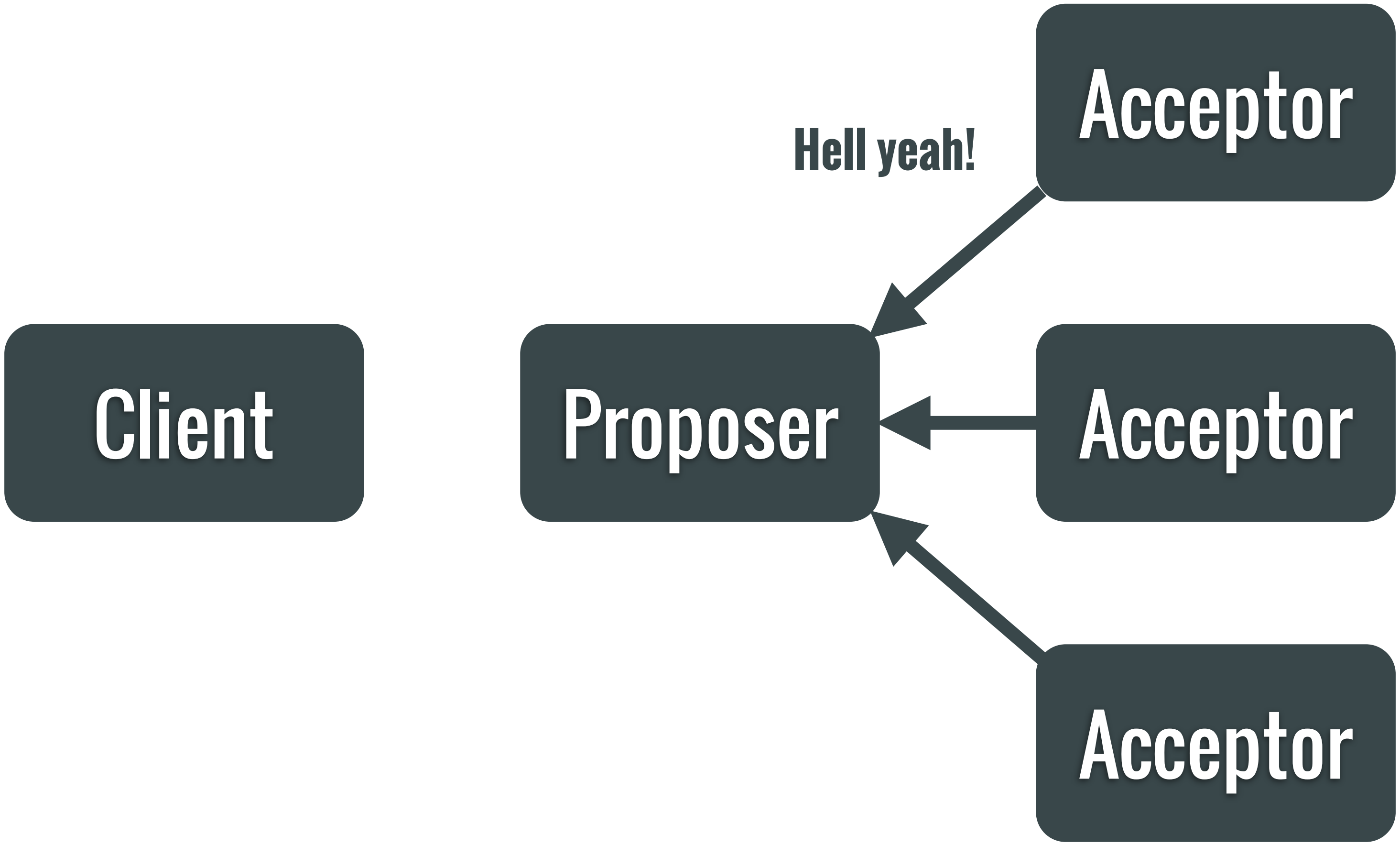
Client requests change to system

Paxos In A Nutshell



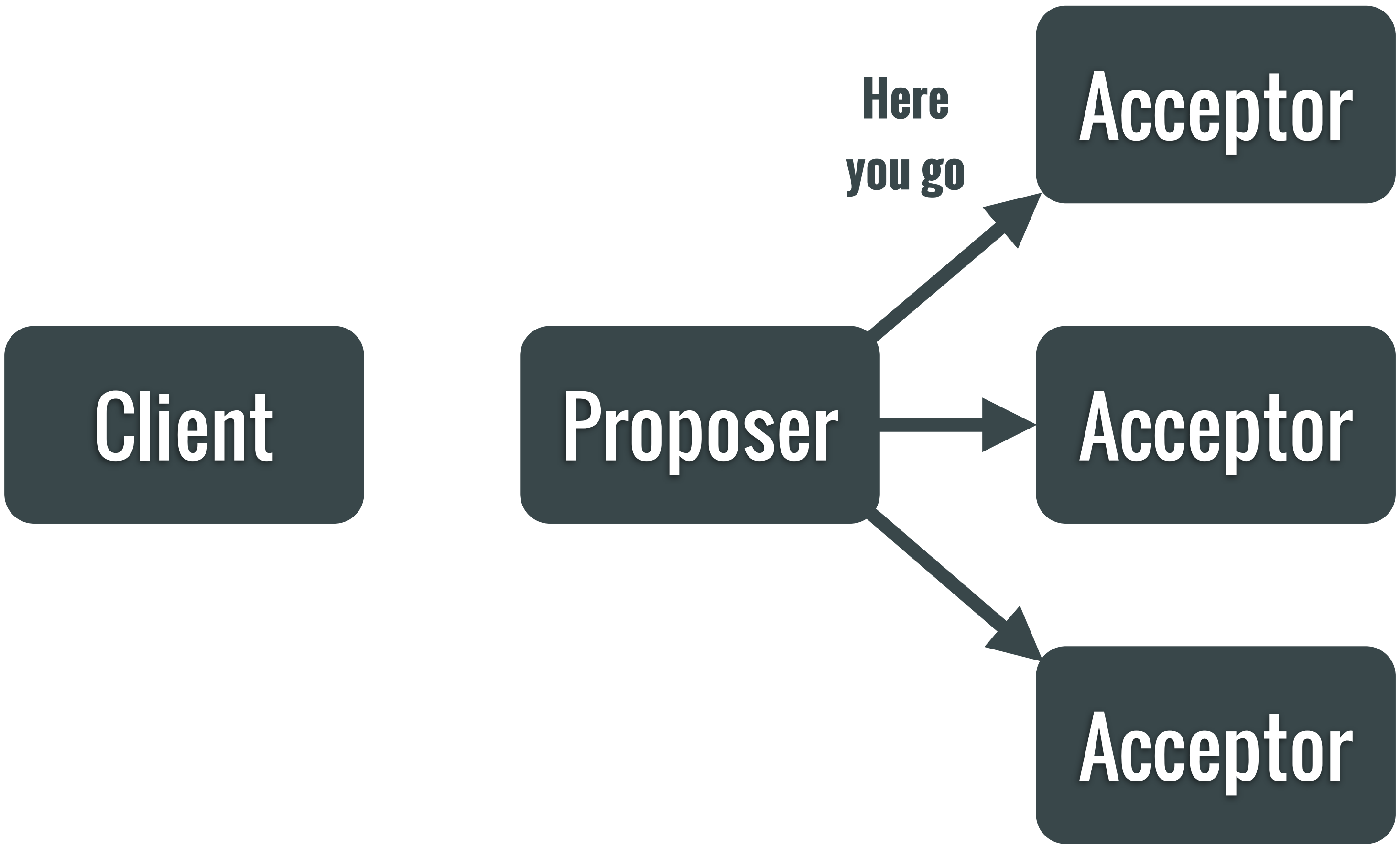
Proposer tells Acceptors to get ready for a change

Paxos In A Nutshell



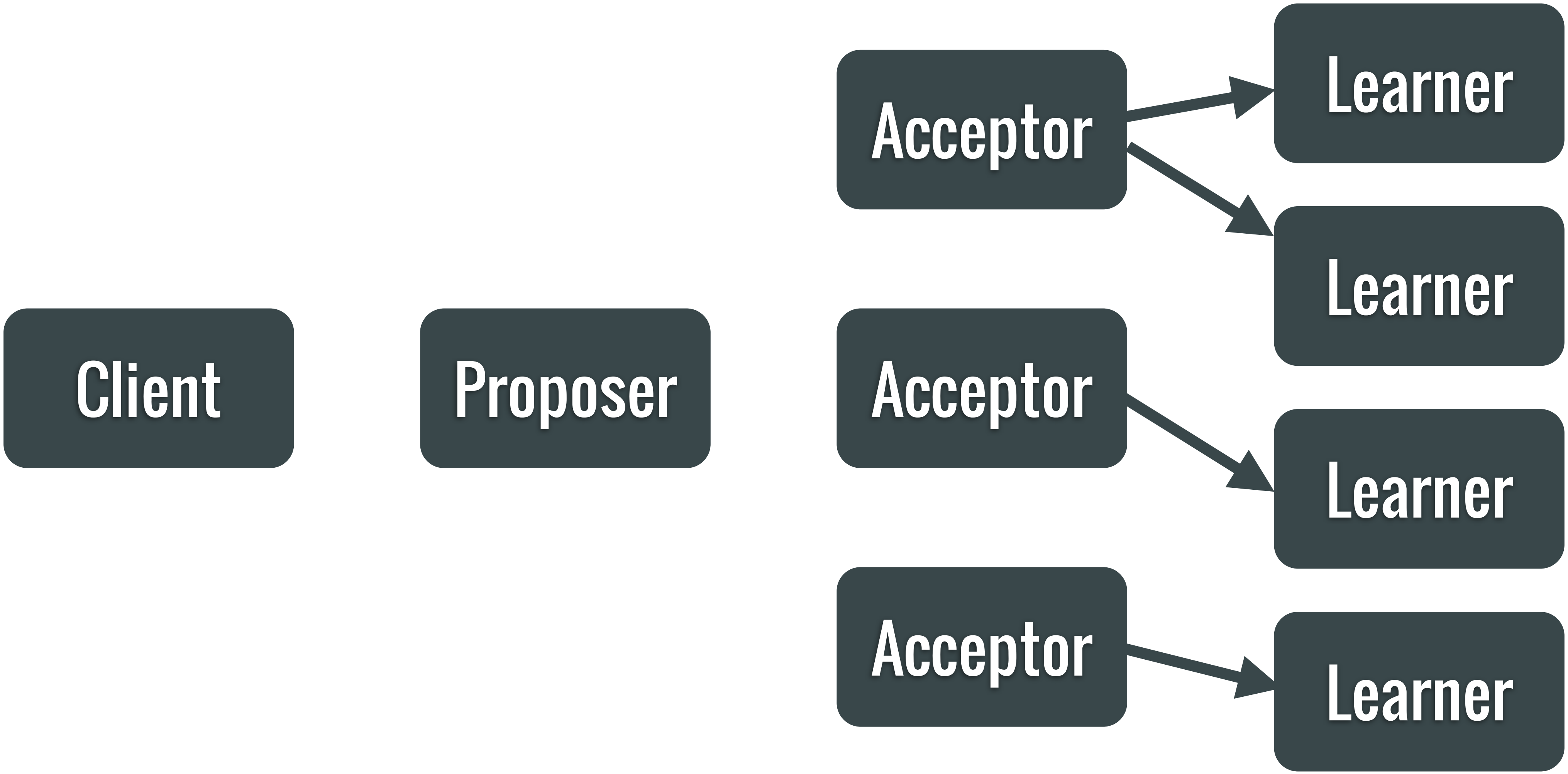
Acceptors confirm to Proposer that they're ready

Paxos In A Nutshell



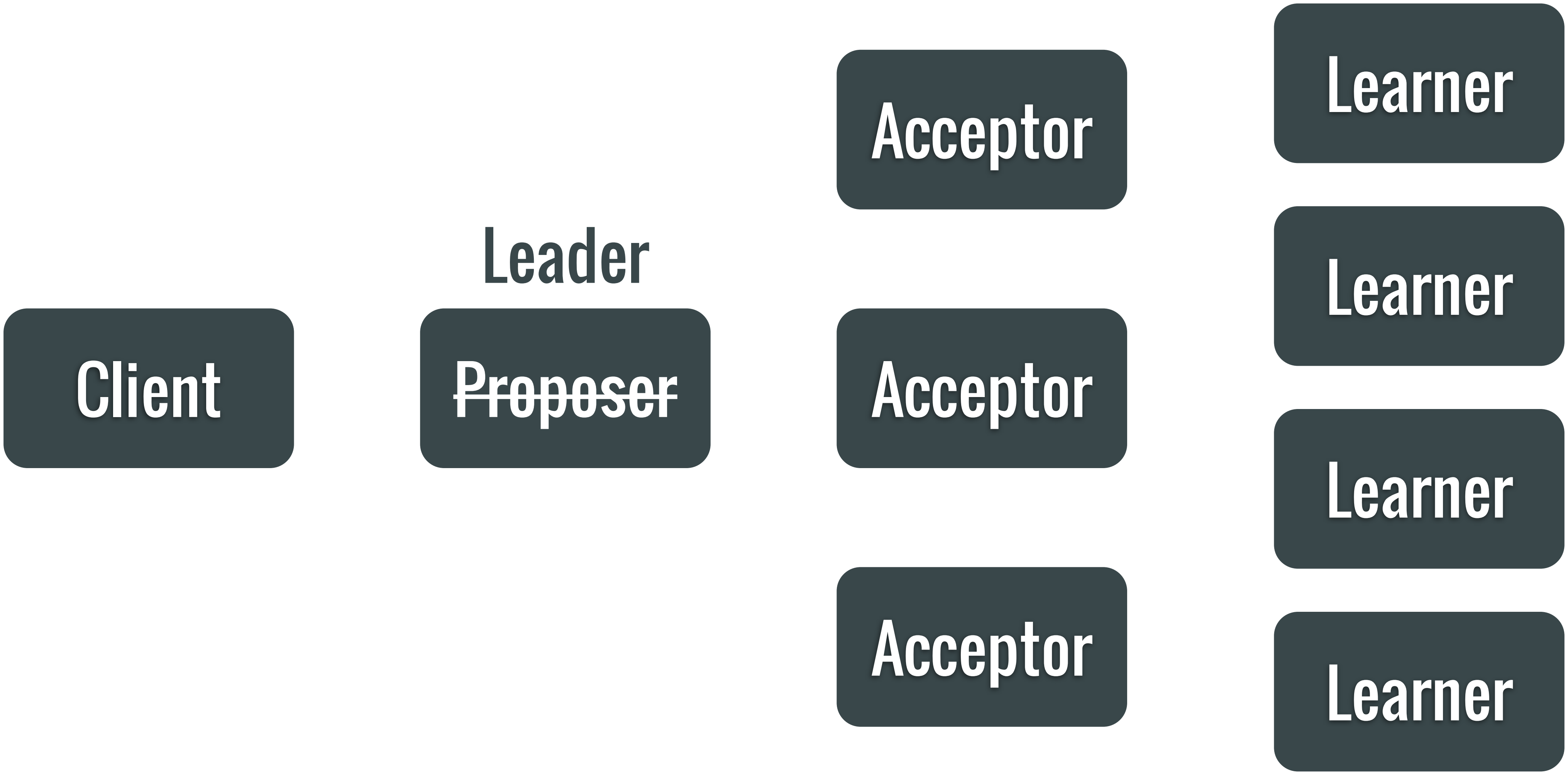
Proposer sends change to Acceptors

Paxos In A Nutshell



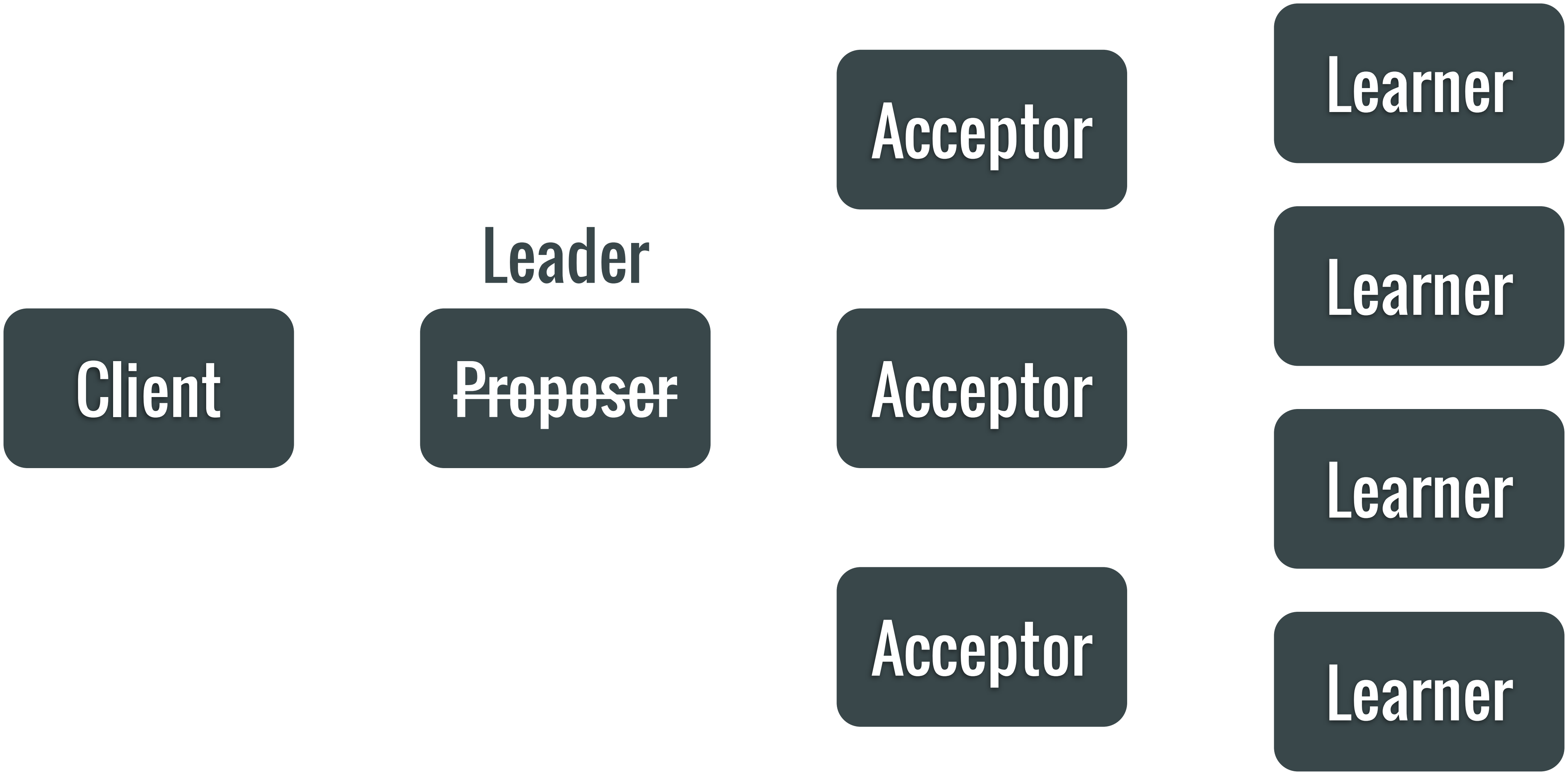
Acceptors propagate change to Learners

Paxos In A Nutshell



Proposer is now recognized as leader

Paxos In A Nutshell



Repeat for every new change to the system

Fun Raft Facts

Created By:

Diego Ongaro

Ph.D. Student

Stanford University

Diego Ongaro

Ph.D. Student

Stanford University

John Ousterhout

Professor of Computer Science

Stanford University

28 Implementations across various languages

In Commercial Use

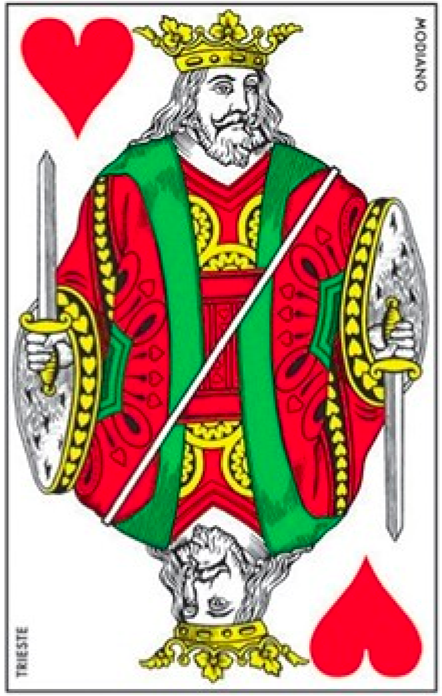
CoreOS

(etcd)

go-raft

Raft Basics

Three Roles:

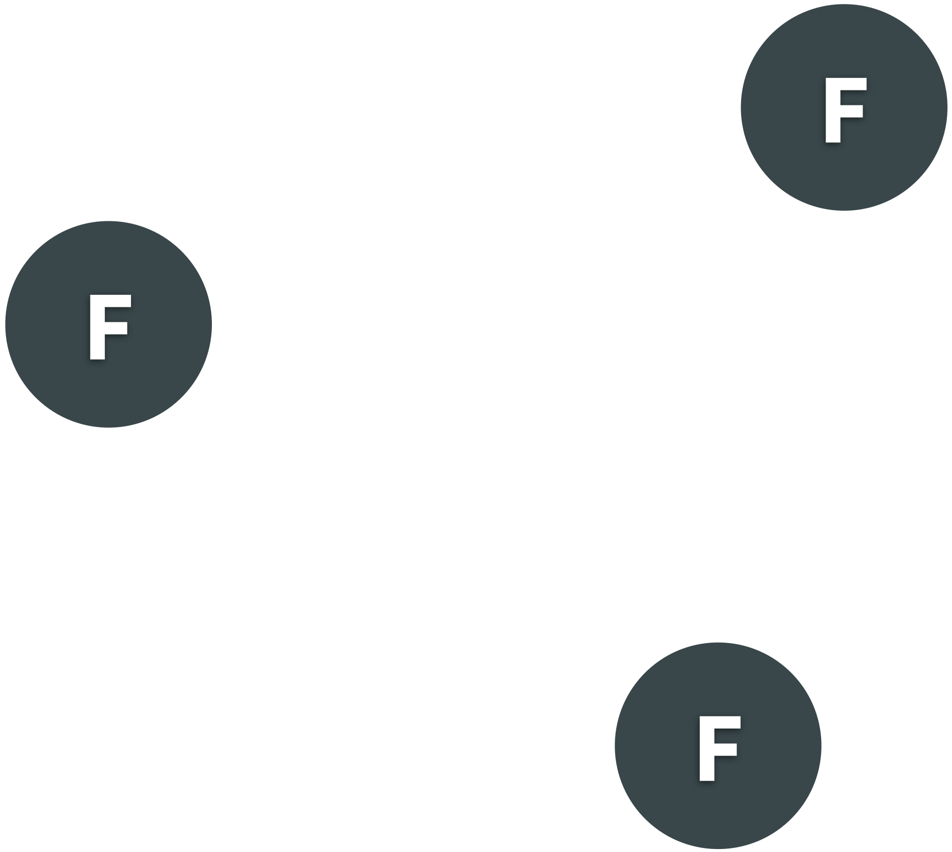


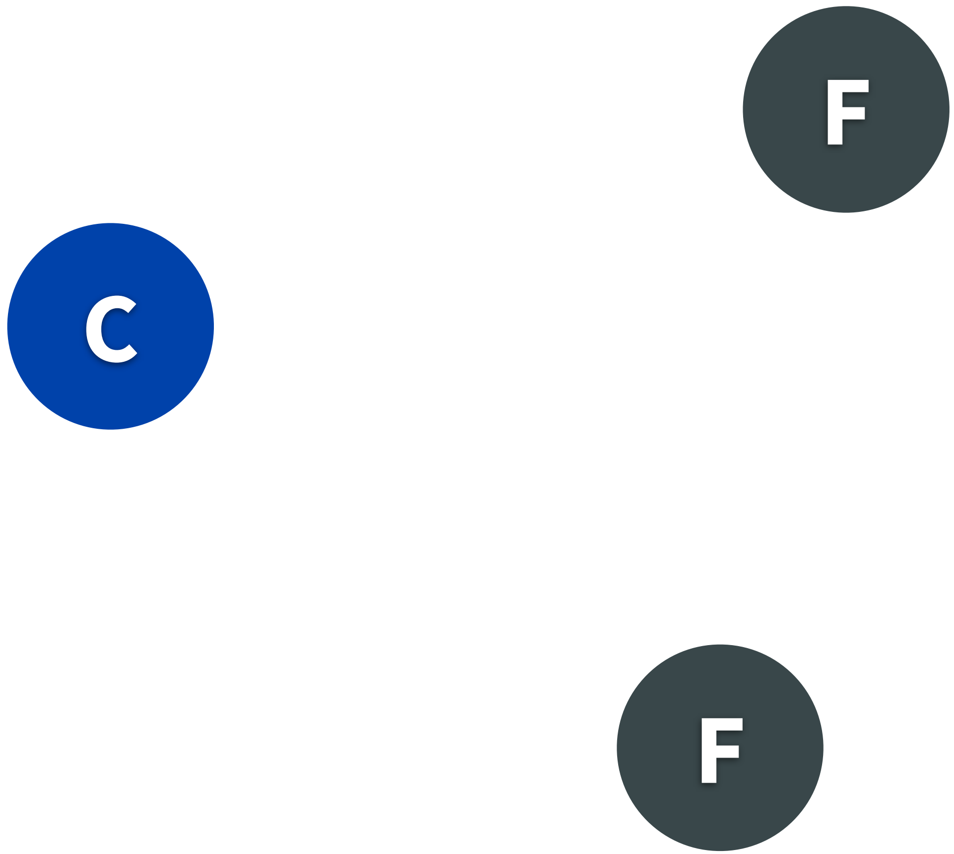
The Leader

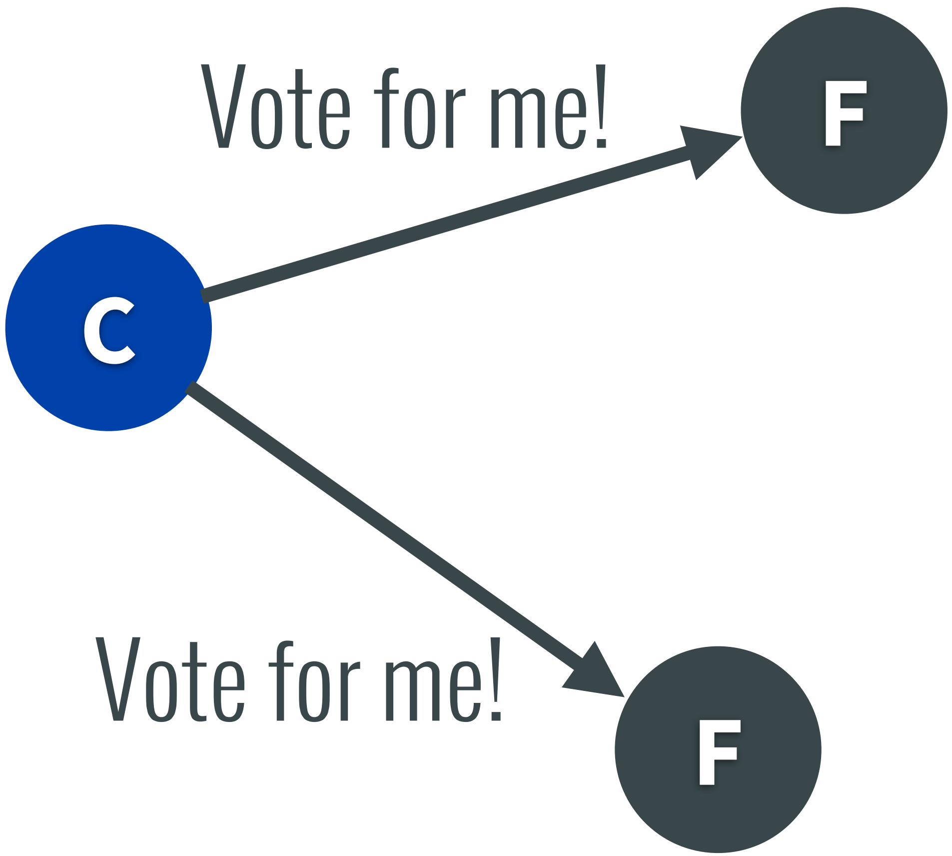
The Follower

The Candidate

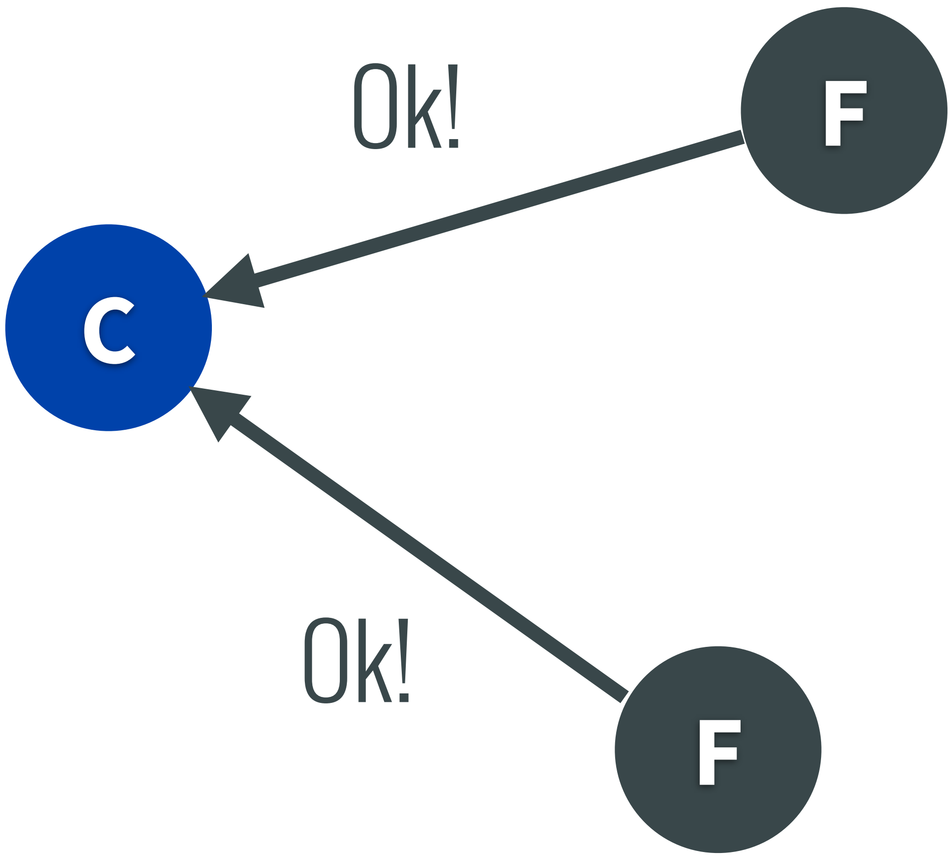
High-Level Example:

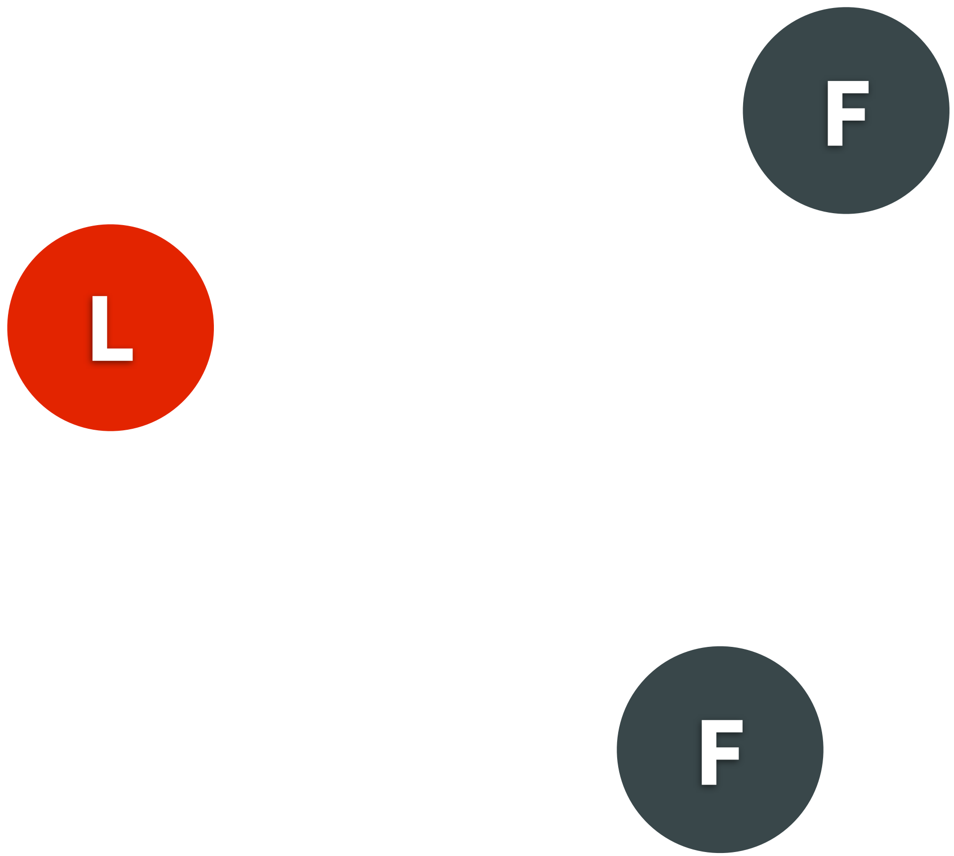

F
F
F

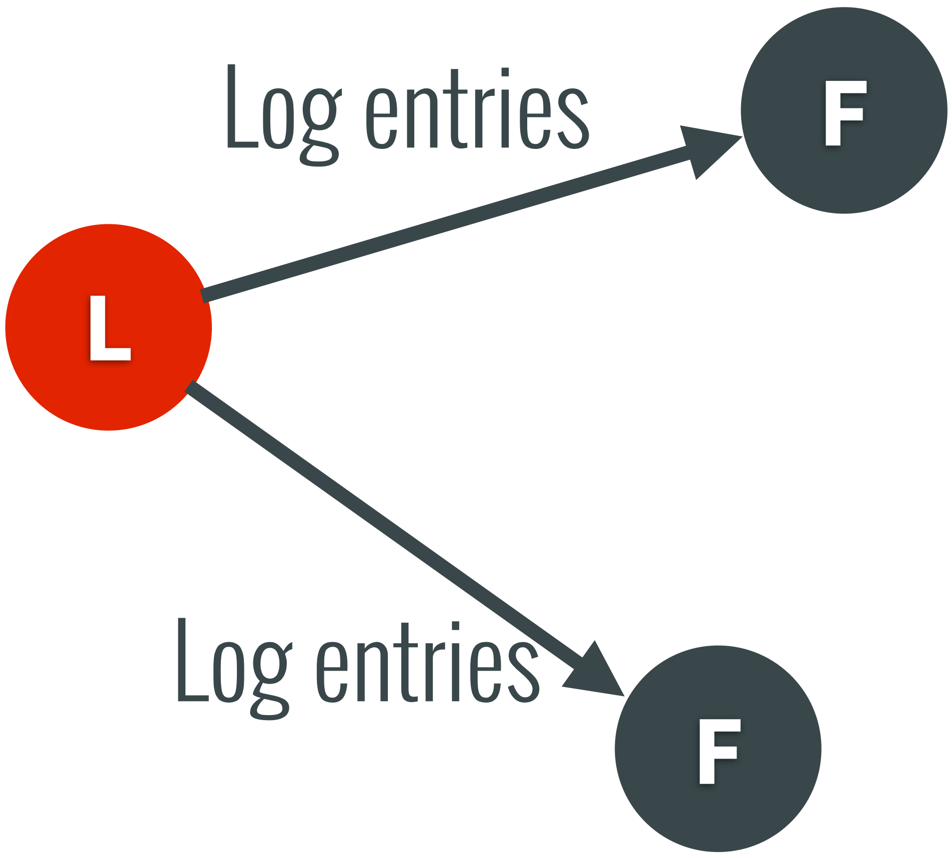

F
C
F

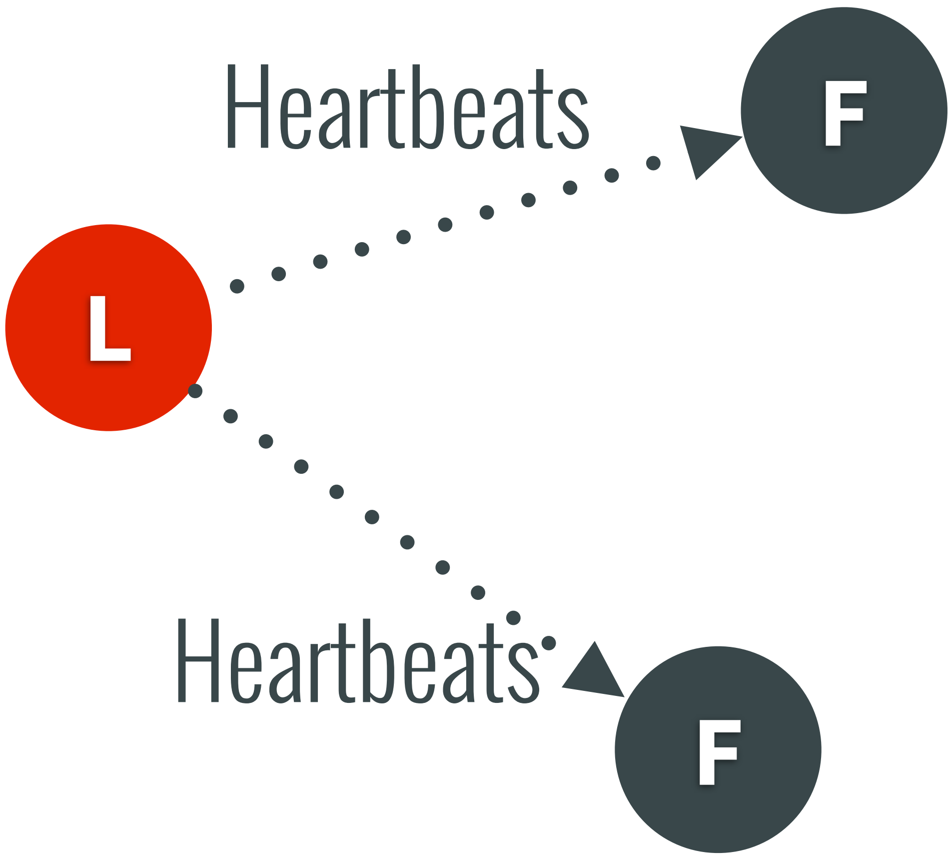


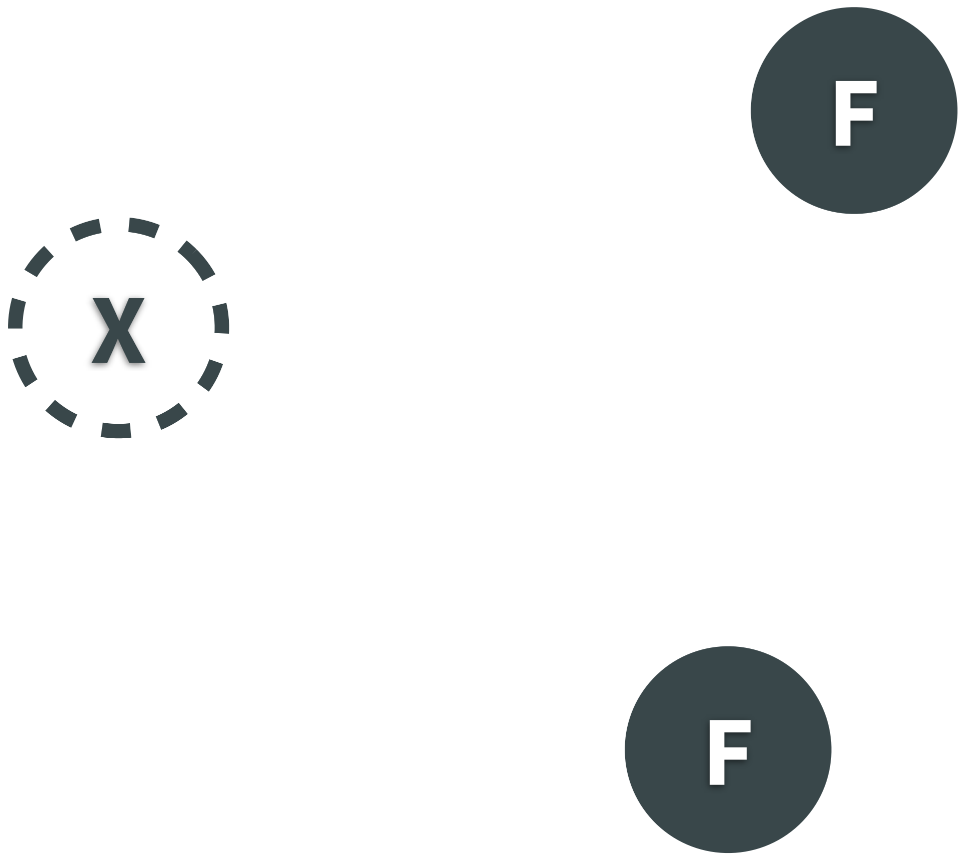
Vote for me!
F
C
Vote for me!
F

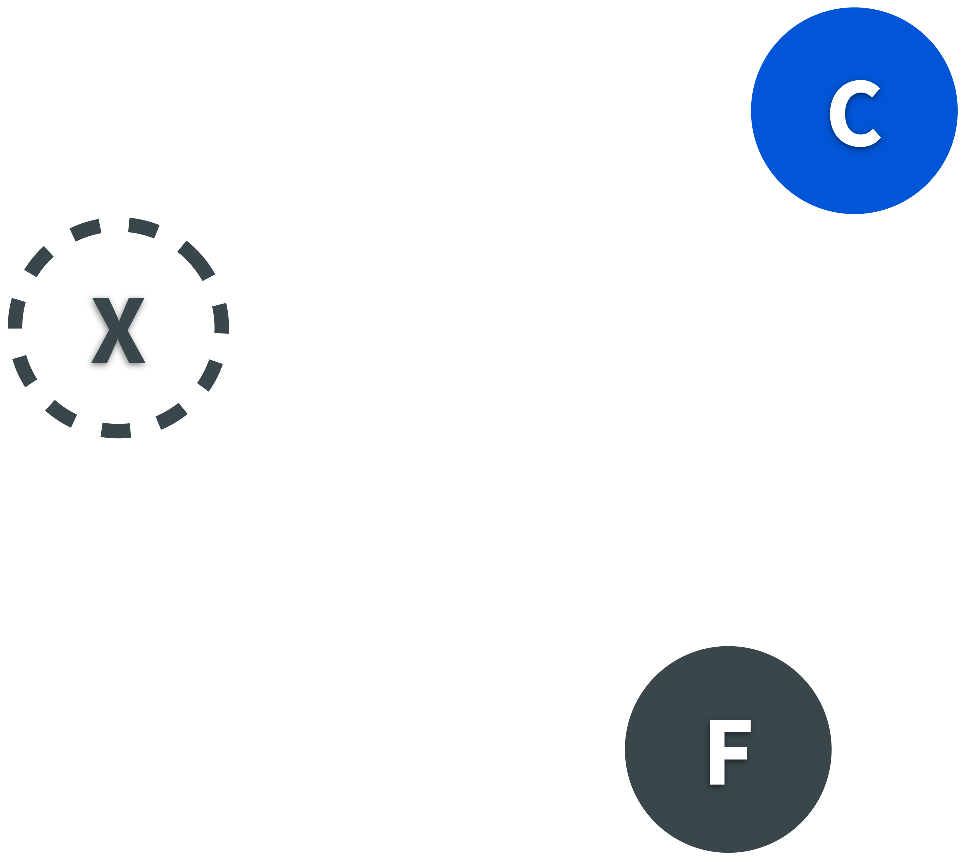

OK!
F
C
OK!
F

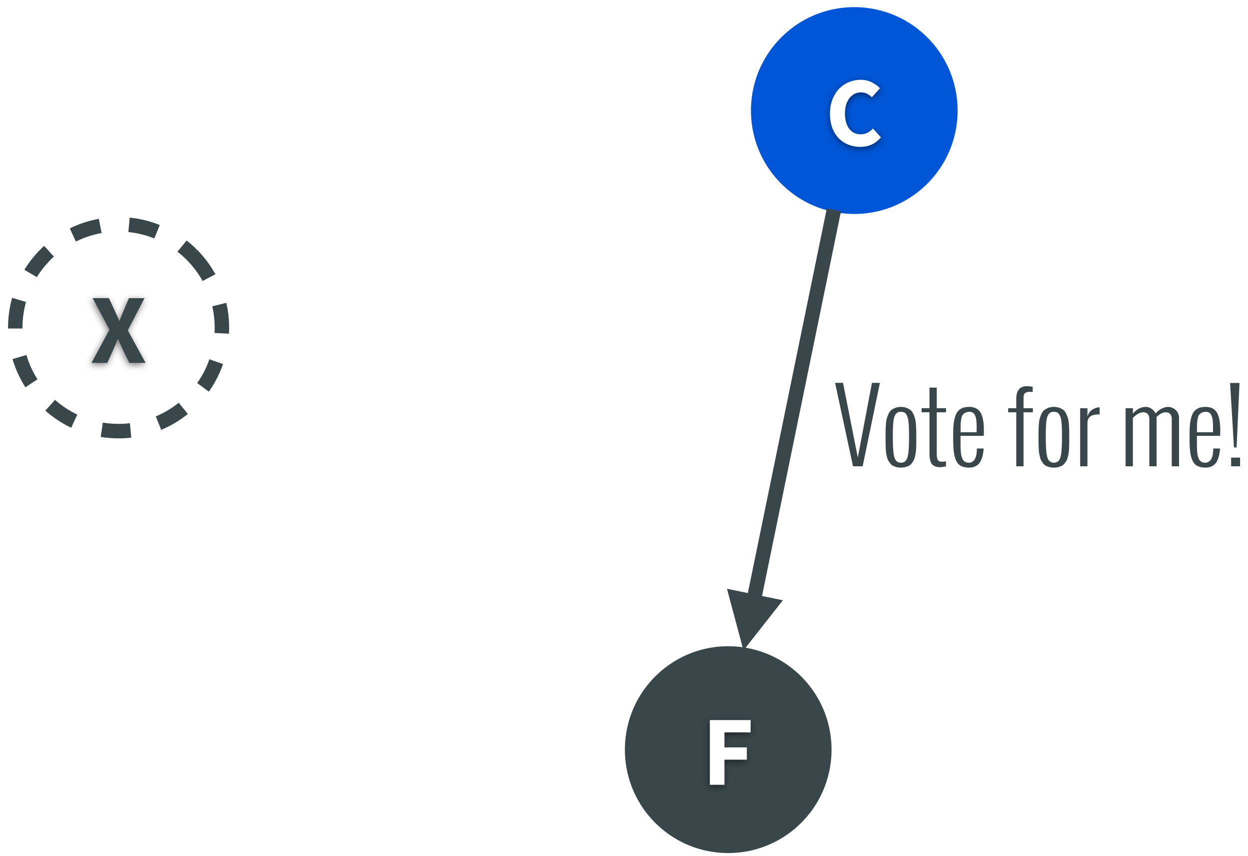

F
L
F

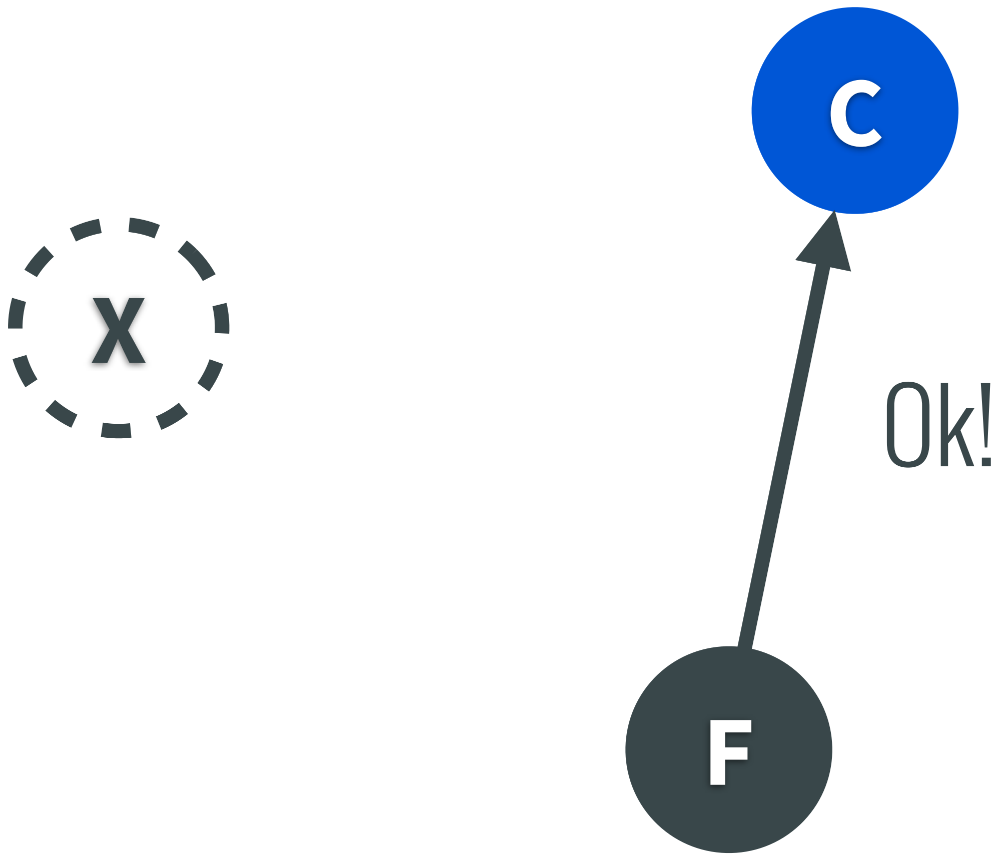

Log entries
F
L
Log entries
F


Heartbeats
F
L
Heartbeats
F

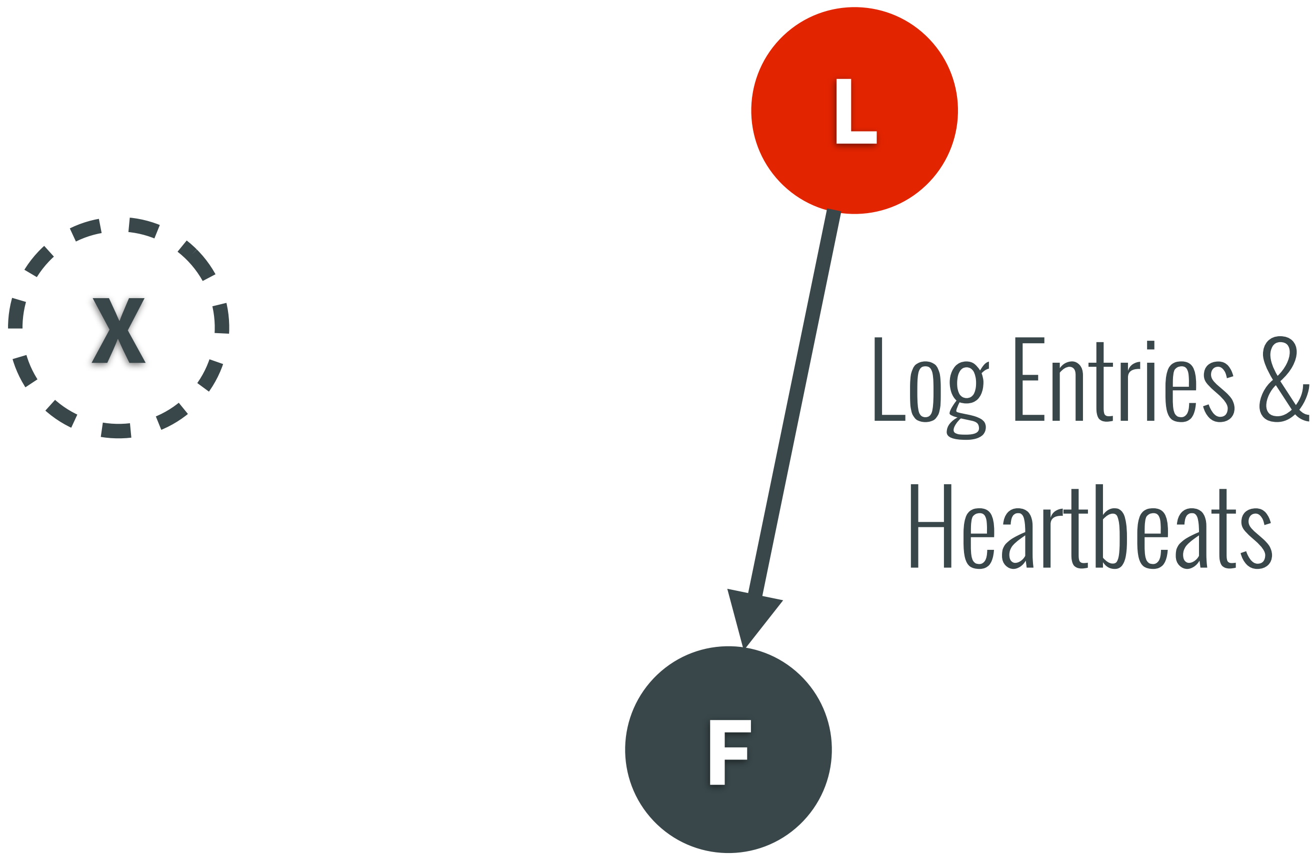

X
F
F


C
X
F


C
Vote for me!
X
F

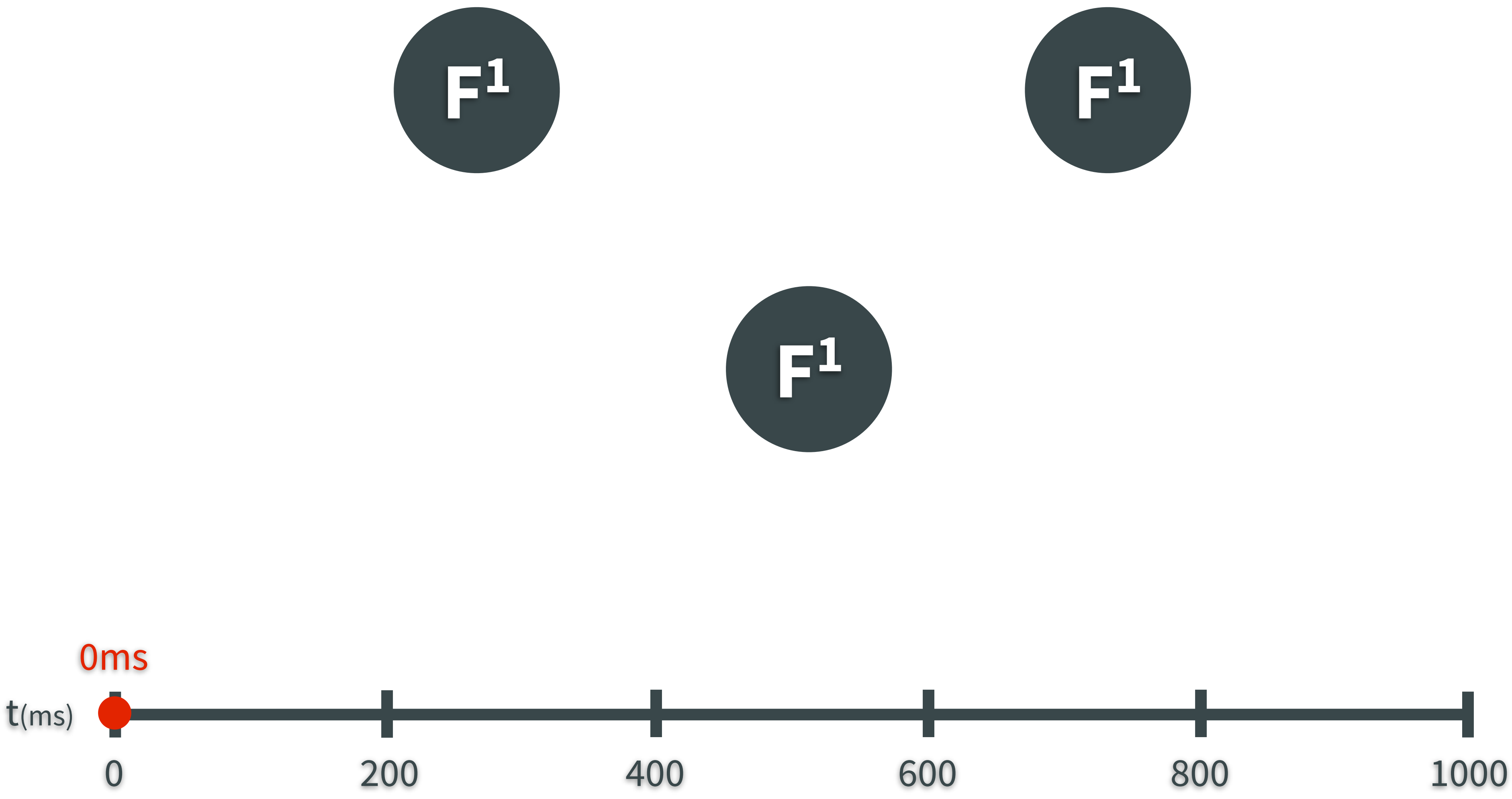


X
C
Ok!
F


L
X
F
Log Entries &
Heartbeats

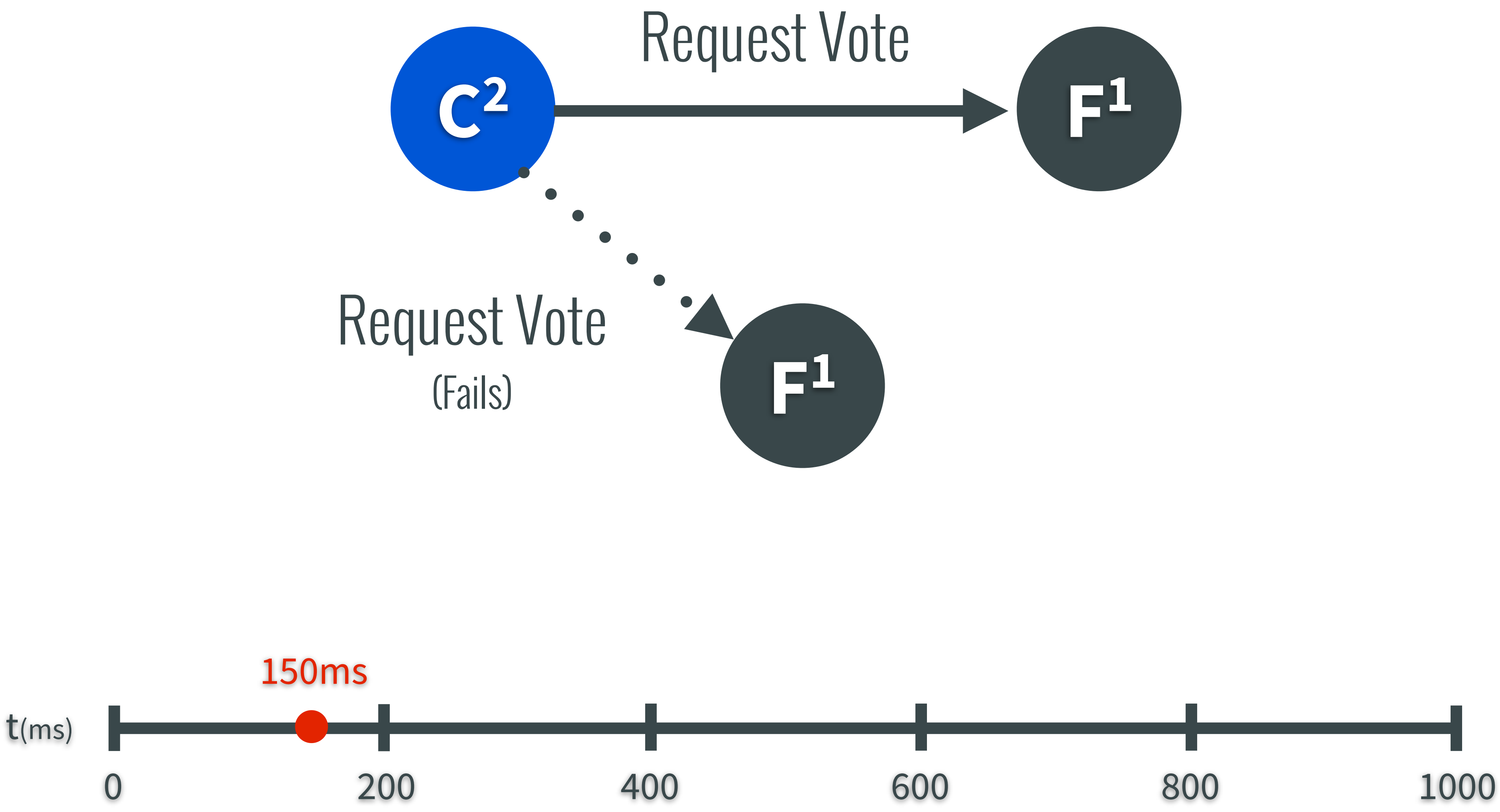
Leader Election

Leader Election



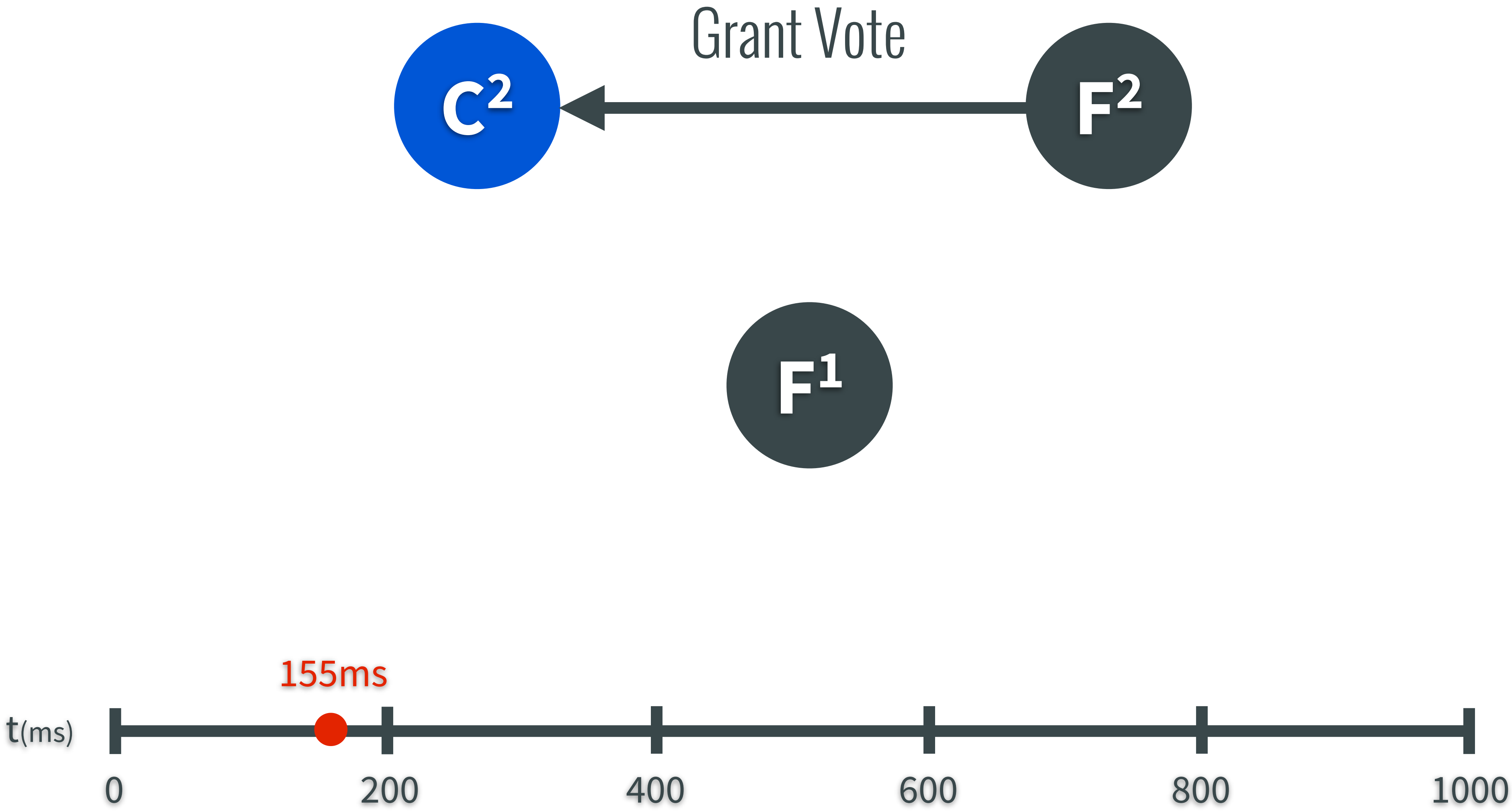
Leader Election

One follower becomes a candidate after an election timeout and requests votes



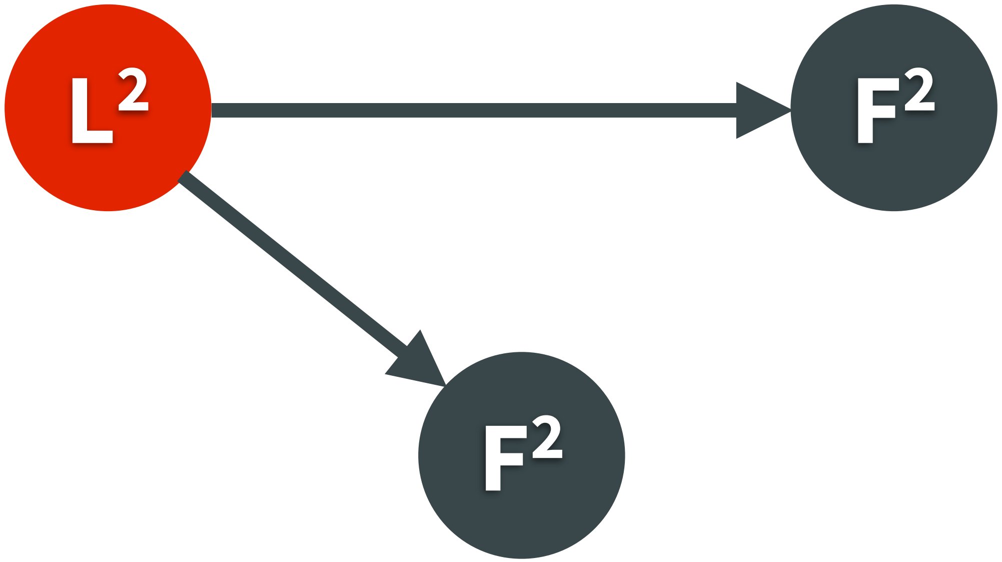
Leader Election

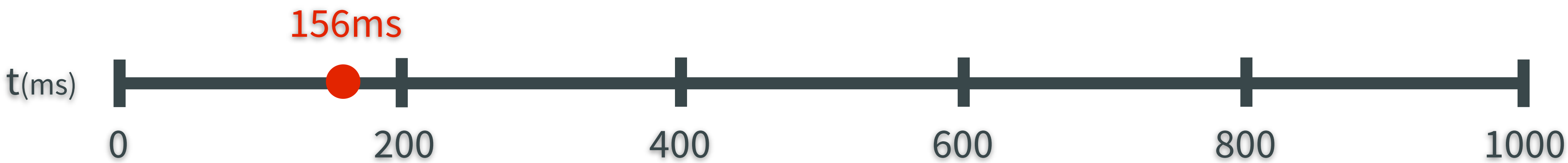
Candidate receives one vote from a peer and one vote from self



Leader Election

Two votes is a majority so candidate becomes leader

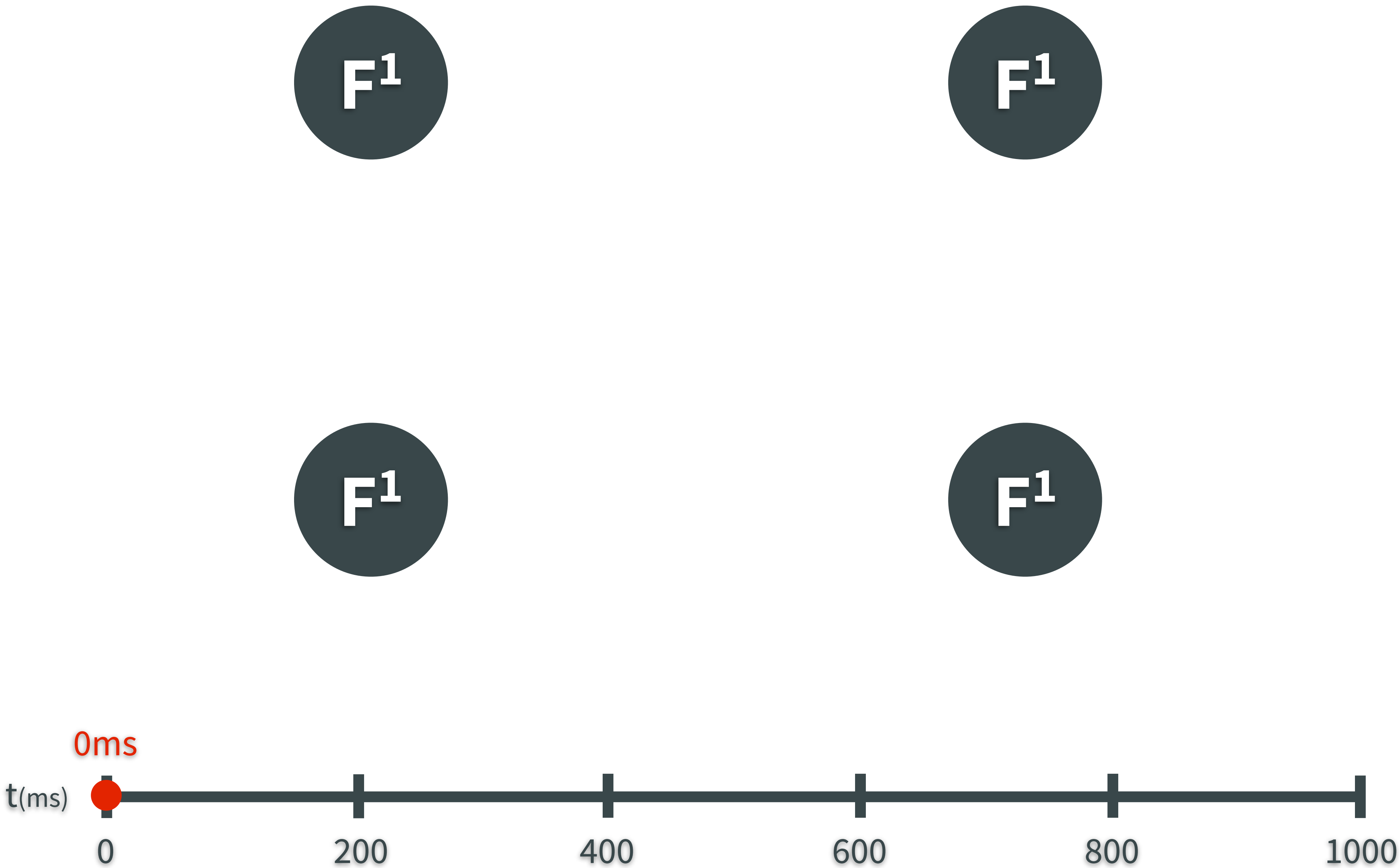




Leader Election

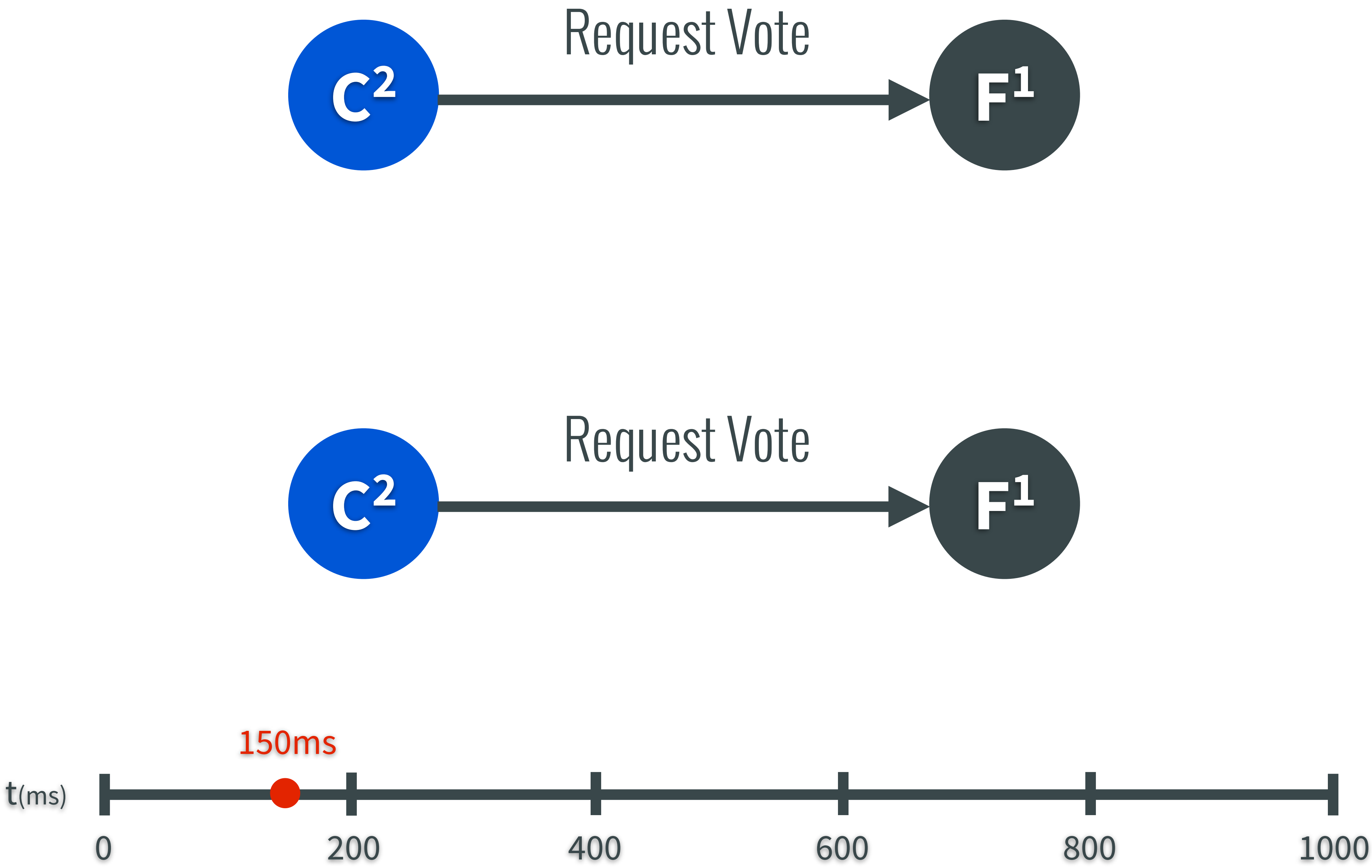
(Split Vote)

Leader Election



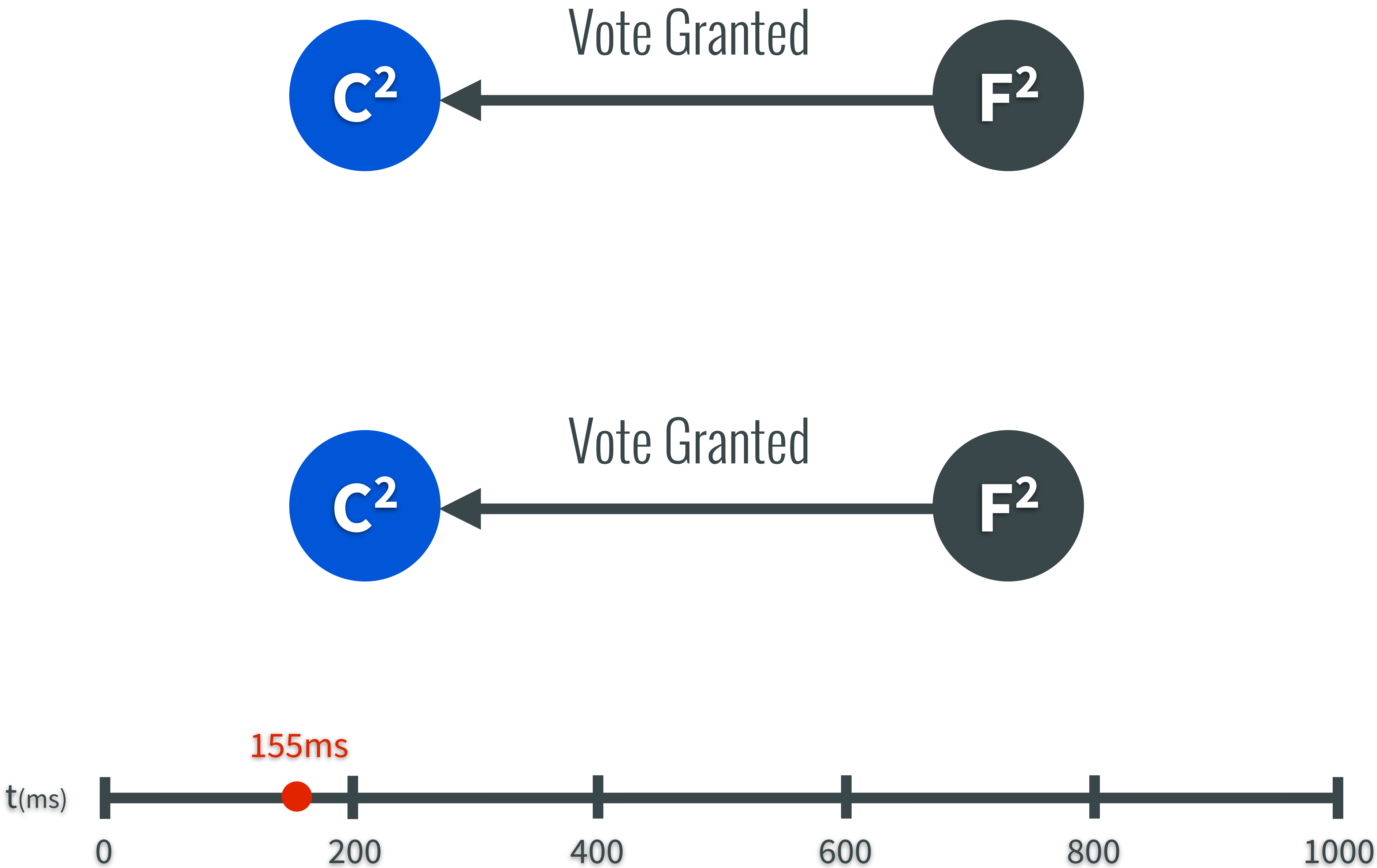
Leader Election

Two followers become candidates simultaneously and begin requesting votes



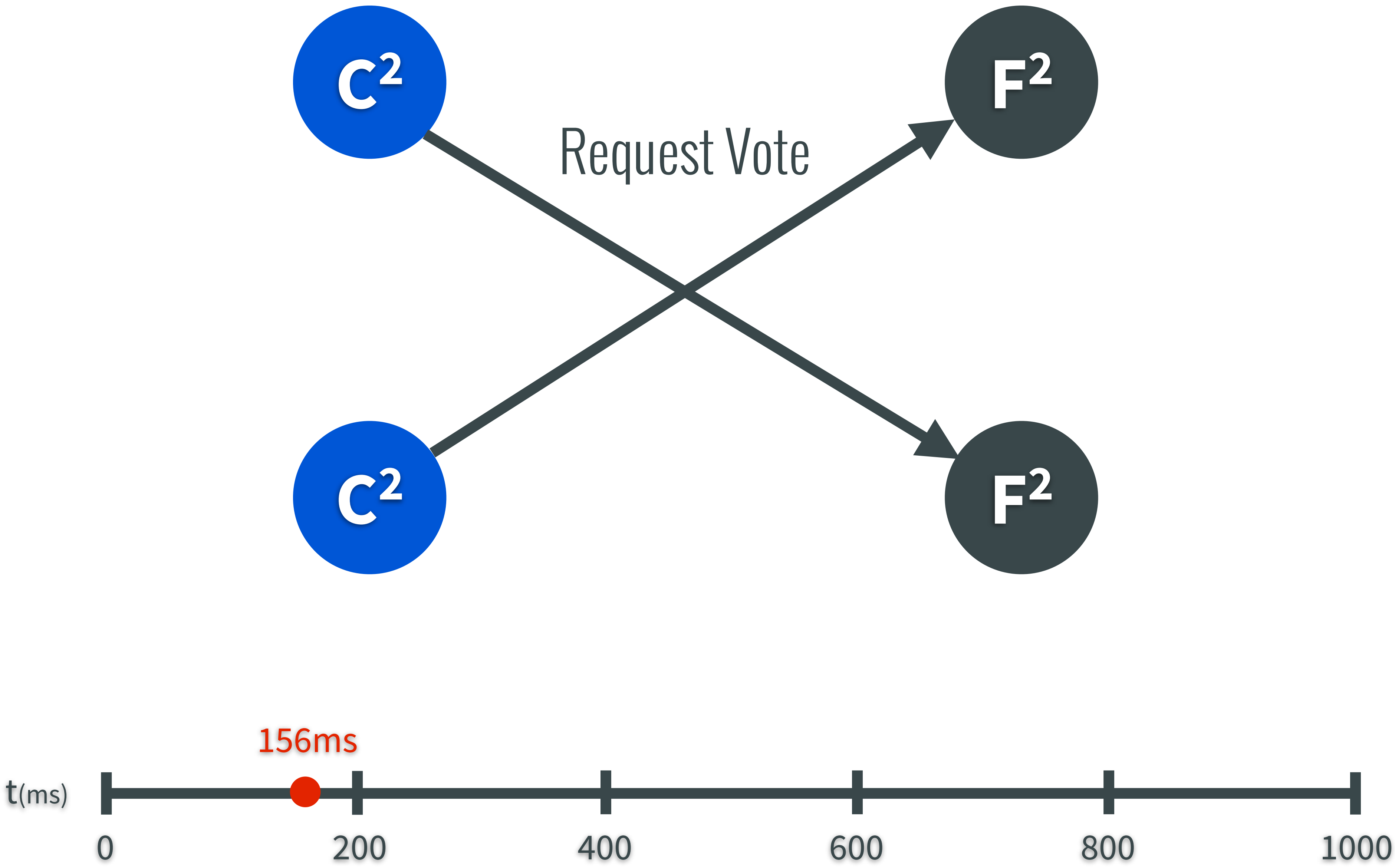
Leader Election

Each candidate receives a vote from themselves and from one peer



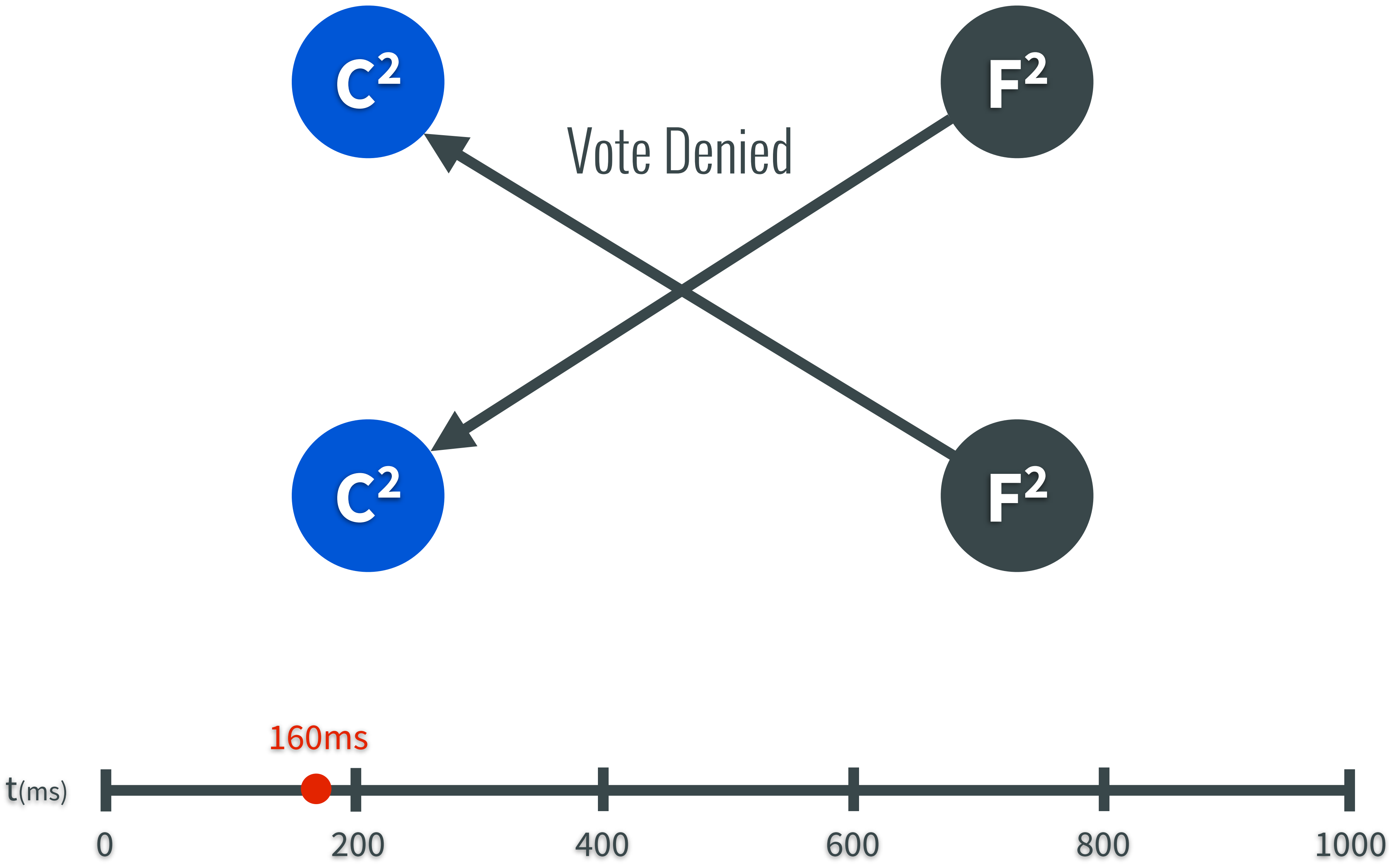
Leader Election

Each candidate requests a vote from a peer who has already voted



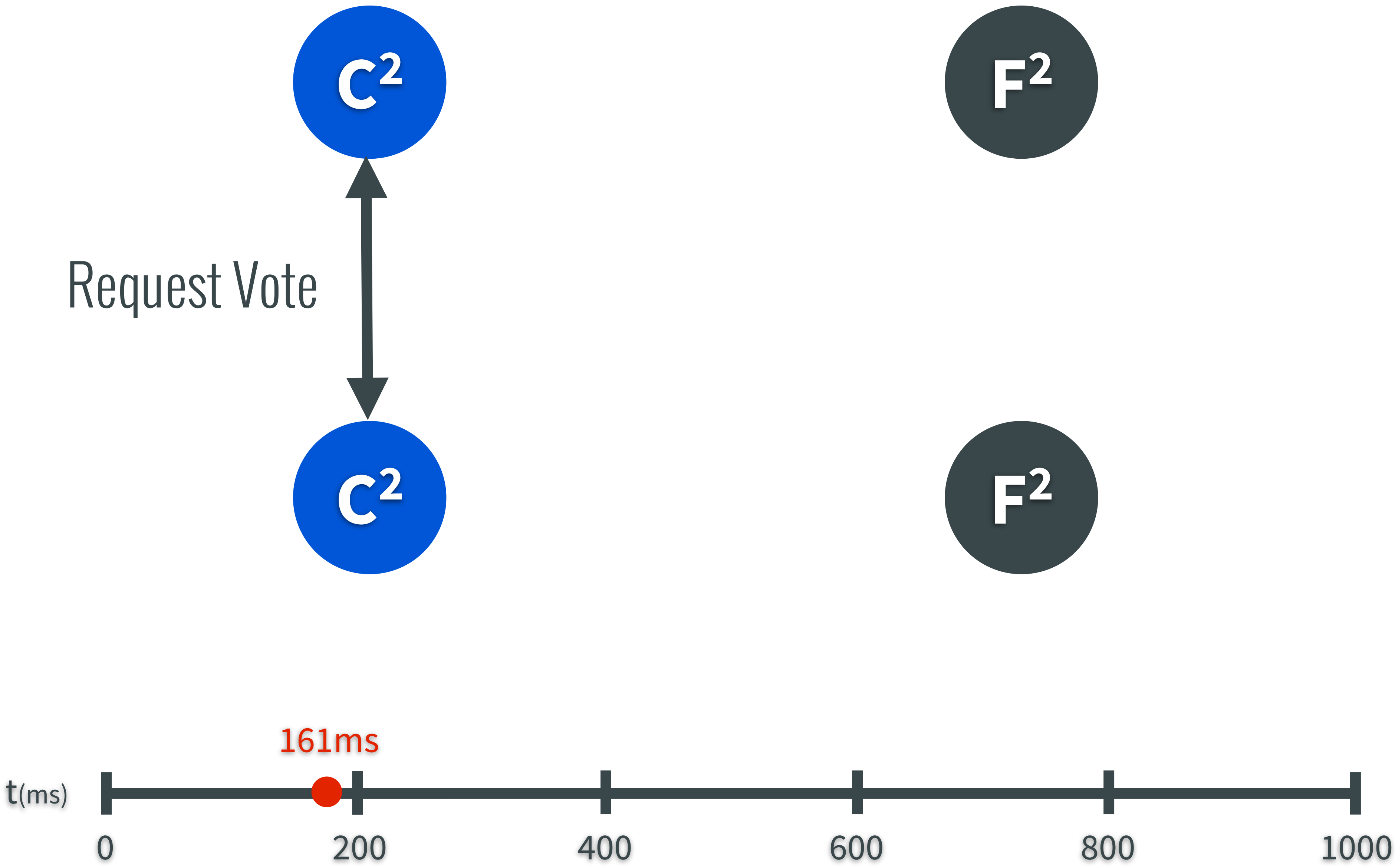
Leader Election

Vote requests are denied because the follower has already voted



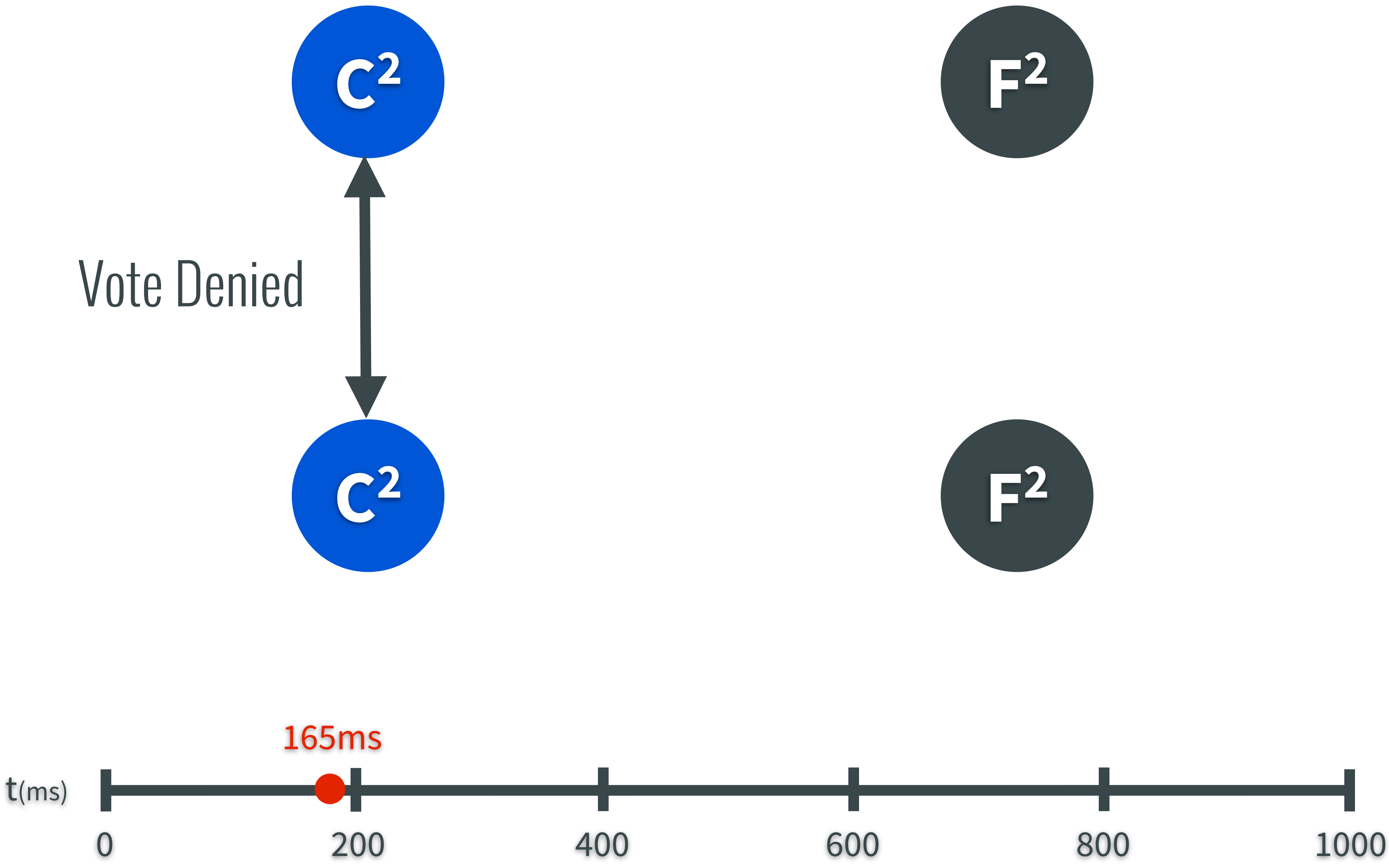
Leader Election

Candidates try to request votes from each other



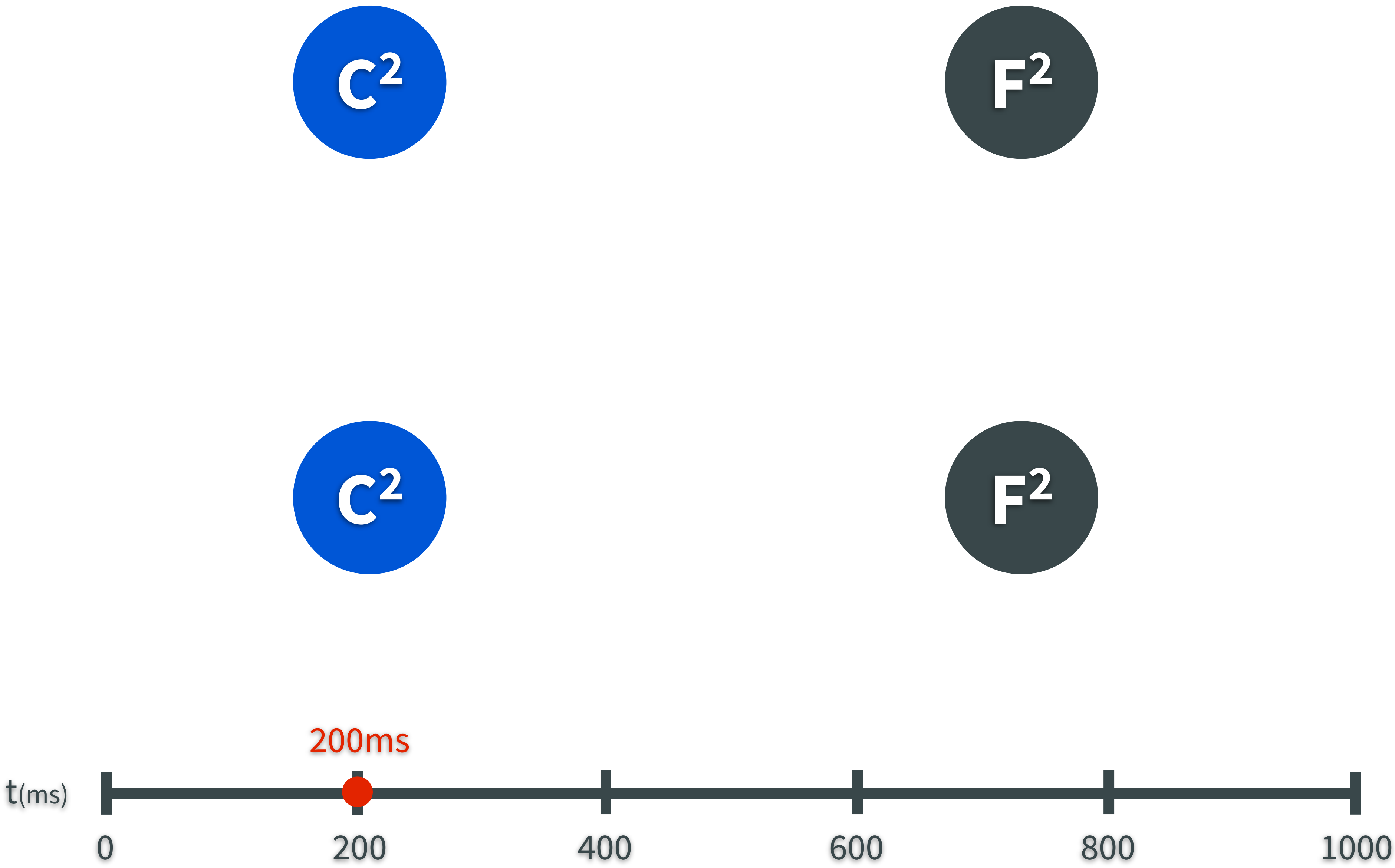
Leader Election

Vote requests are denied because candidates voted for themselves



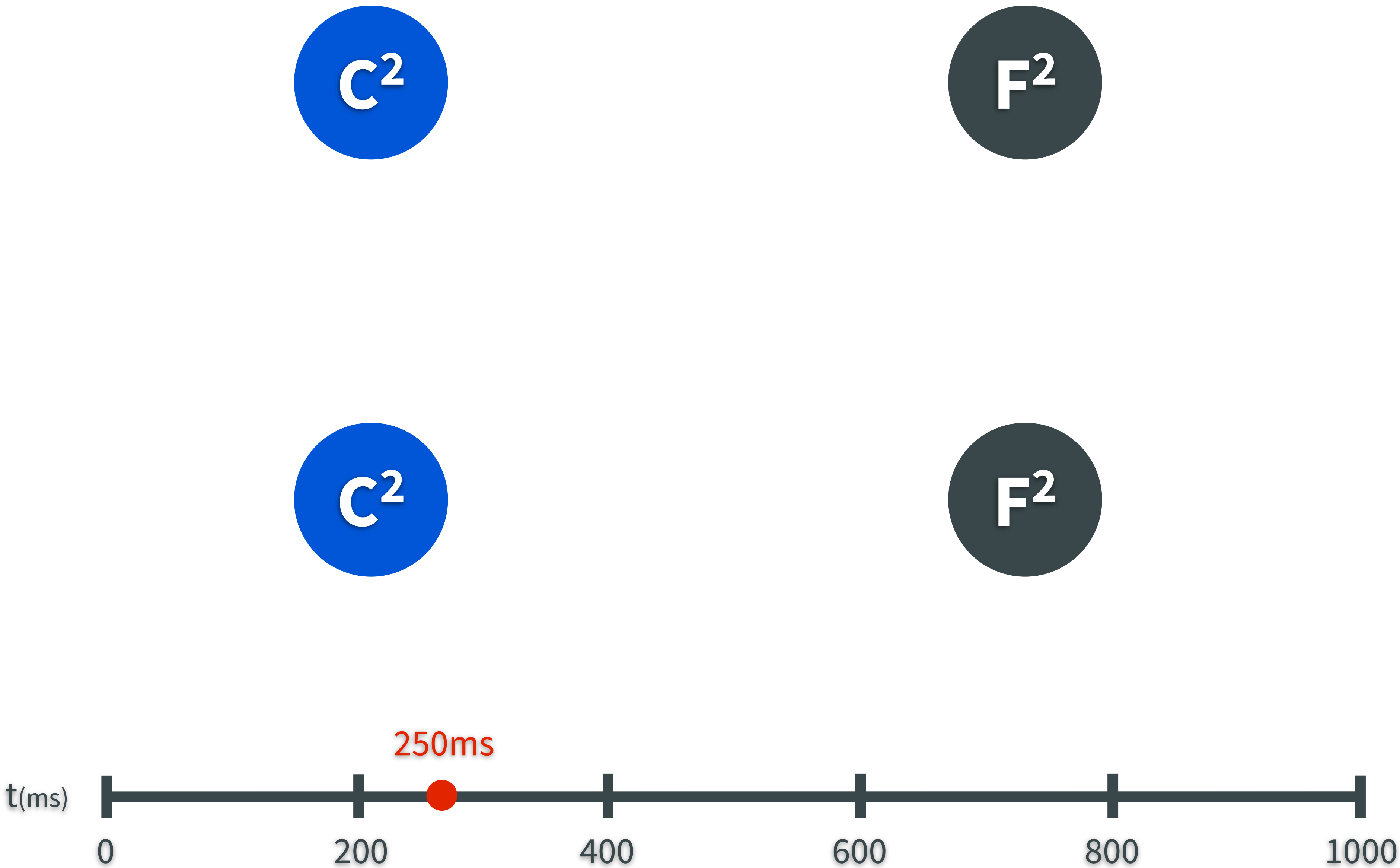
Leader Election

Candidates wait for a randomized election timeout to occur (150ms - 300ms)



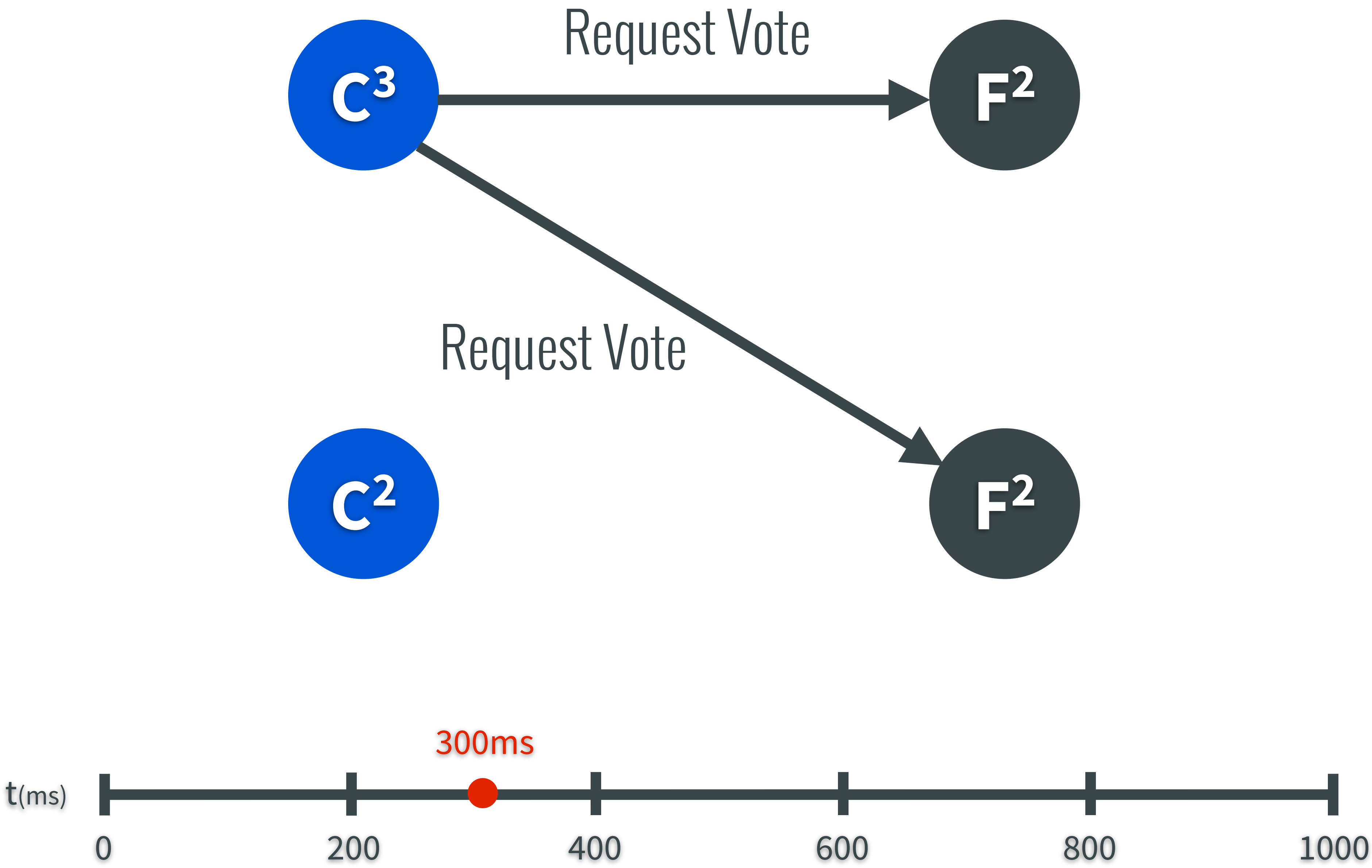
Leader Election

Still waiting...



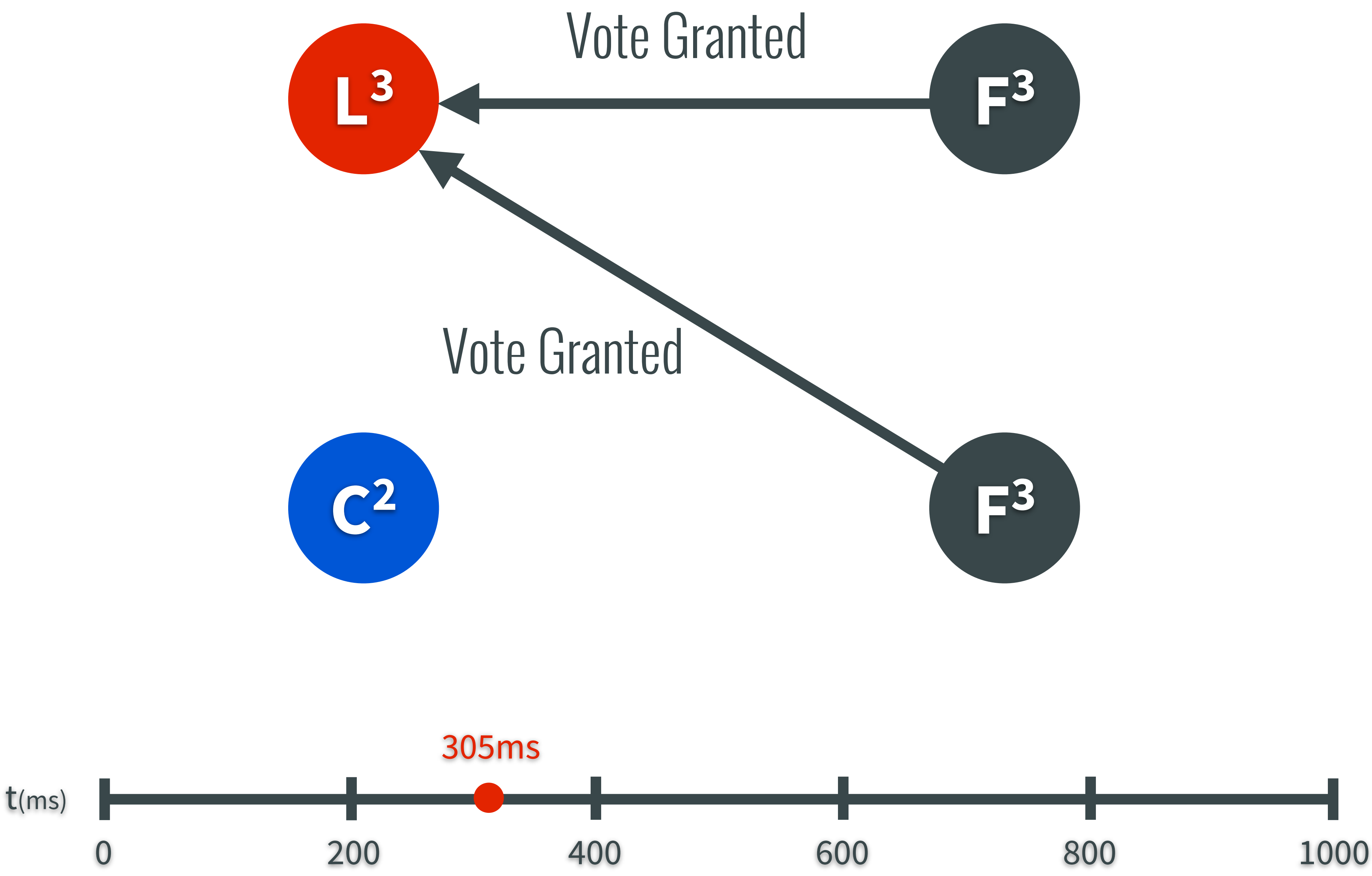
Leader Election

One candidate begins election term #3



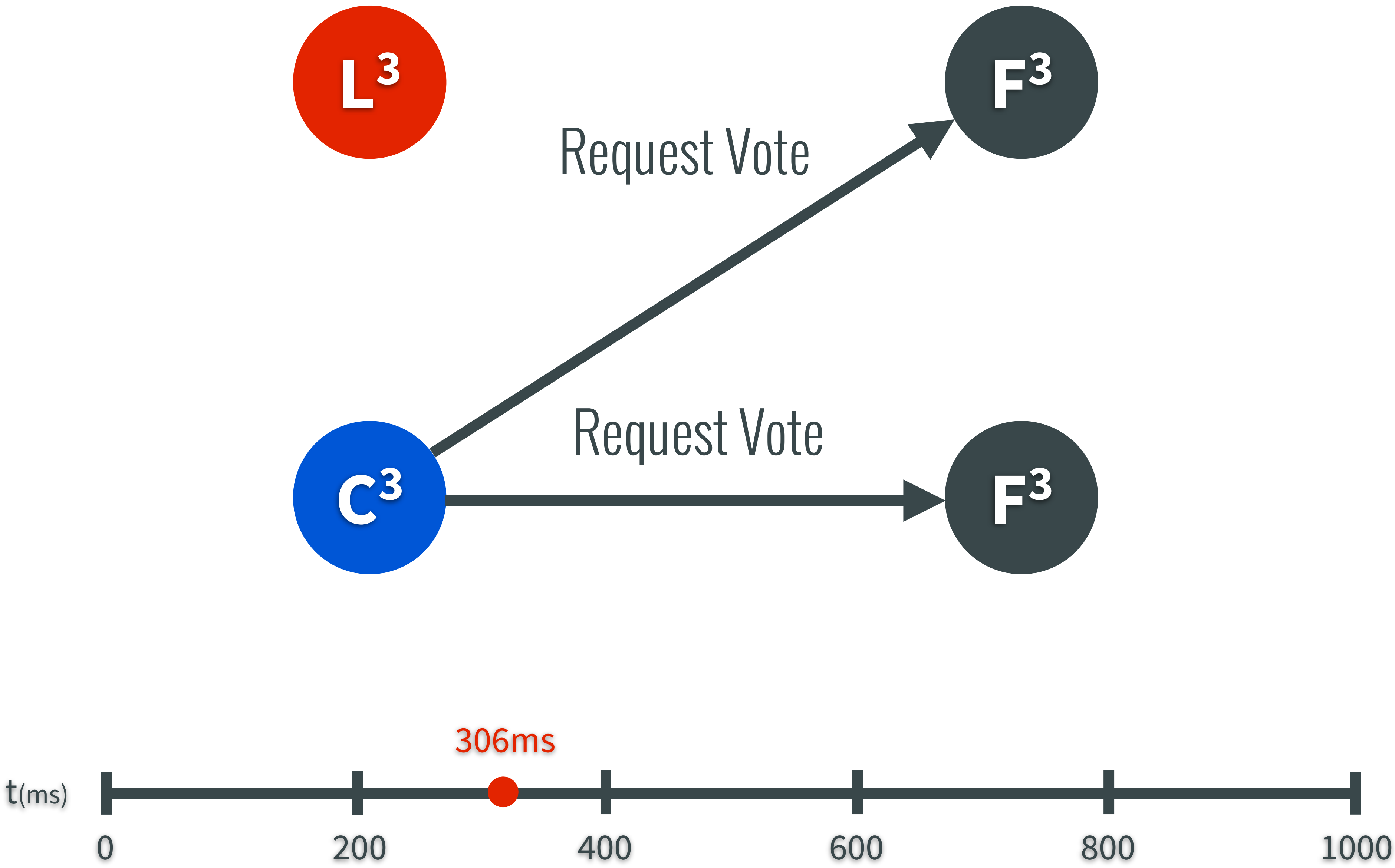
Leader Election

Candidate receives vote from itself and two peer votes so it becomes leader for election term #3



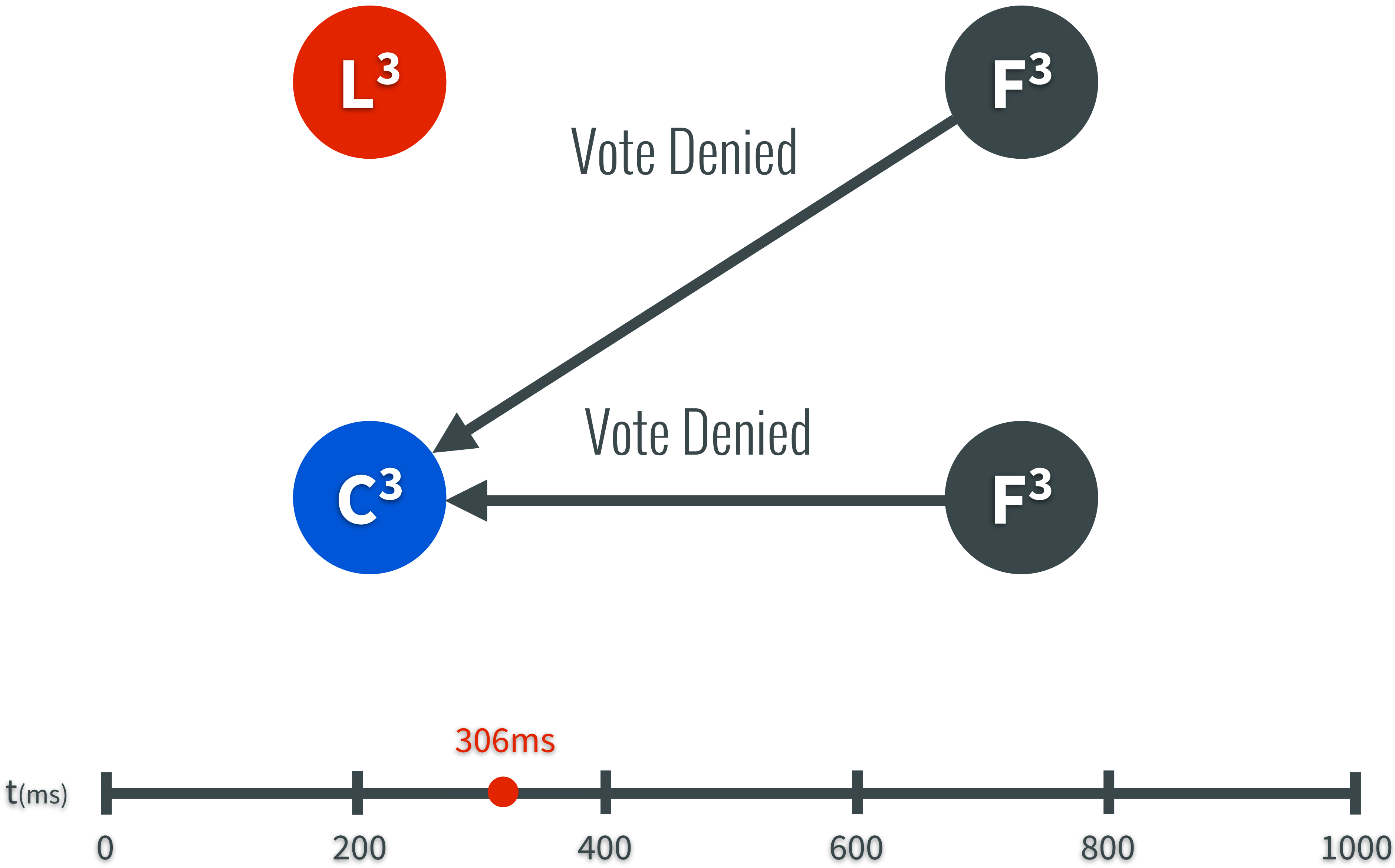
Leader Election

Second candidate doesn't know first candidate won the term and begins requesting votes



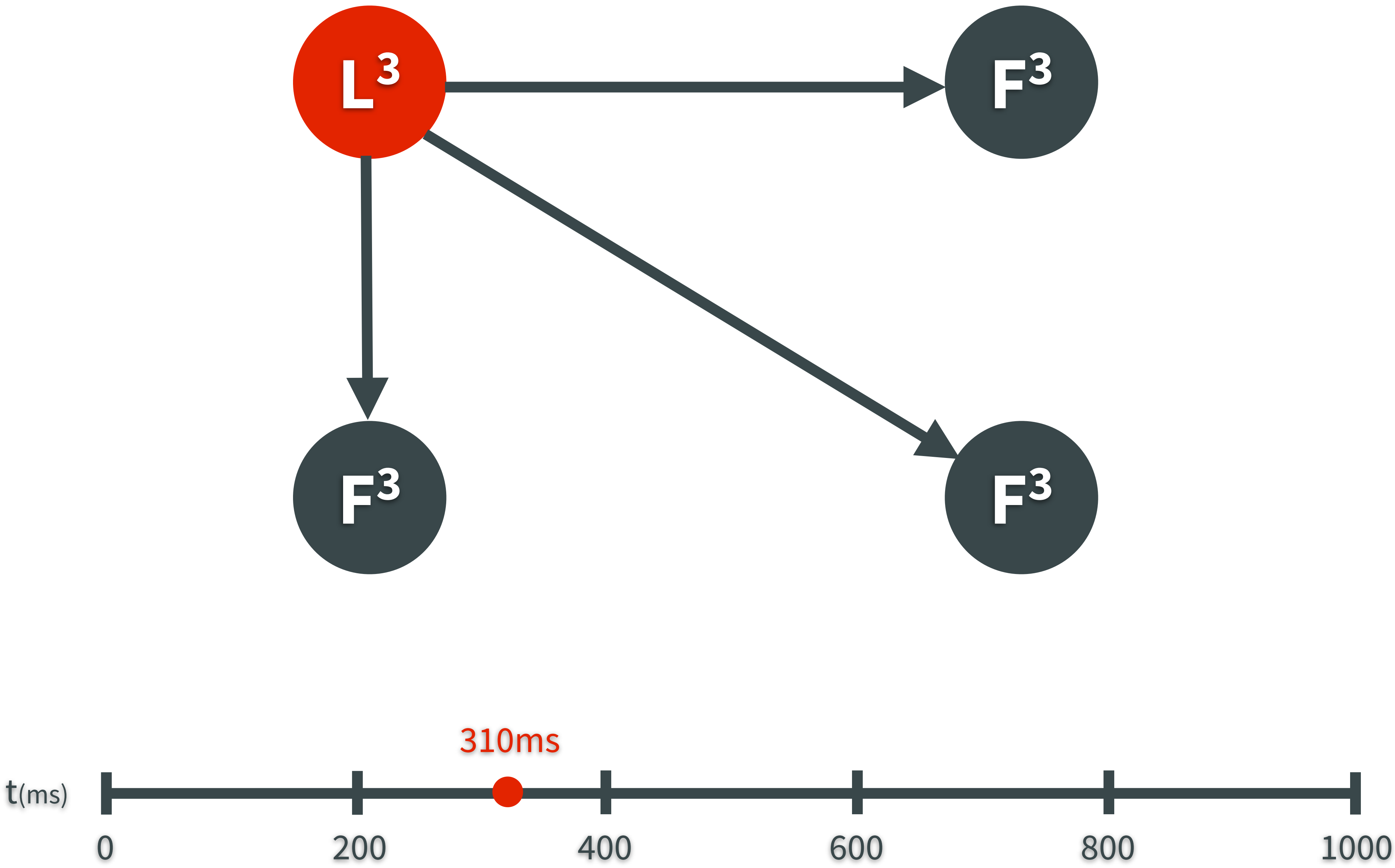
Leader Election

Peers already voted so votes are denied



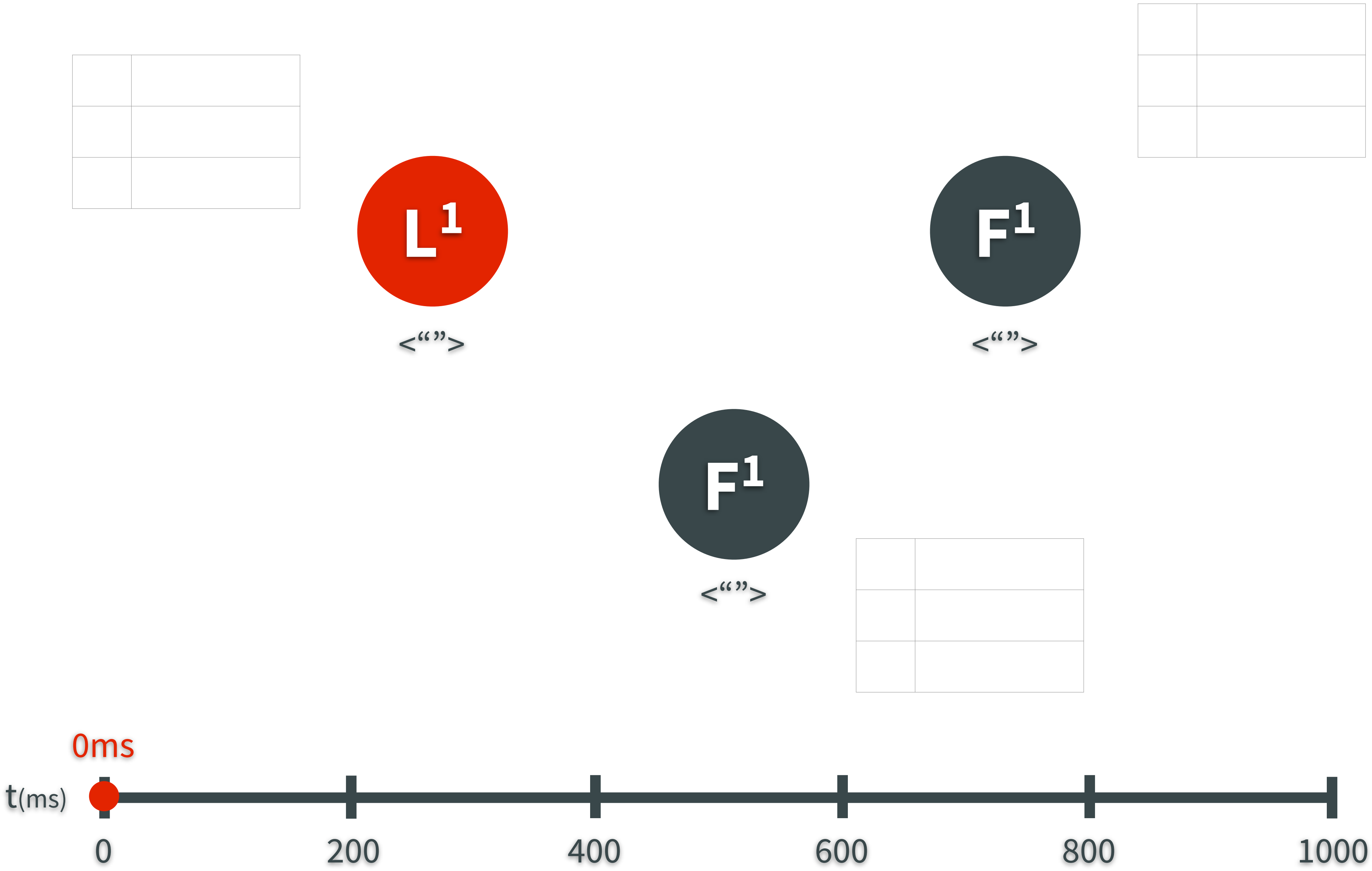
Leader Election

Leader notifies peers of election and other candidate steps down



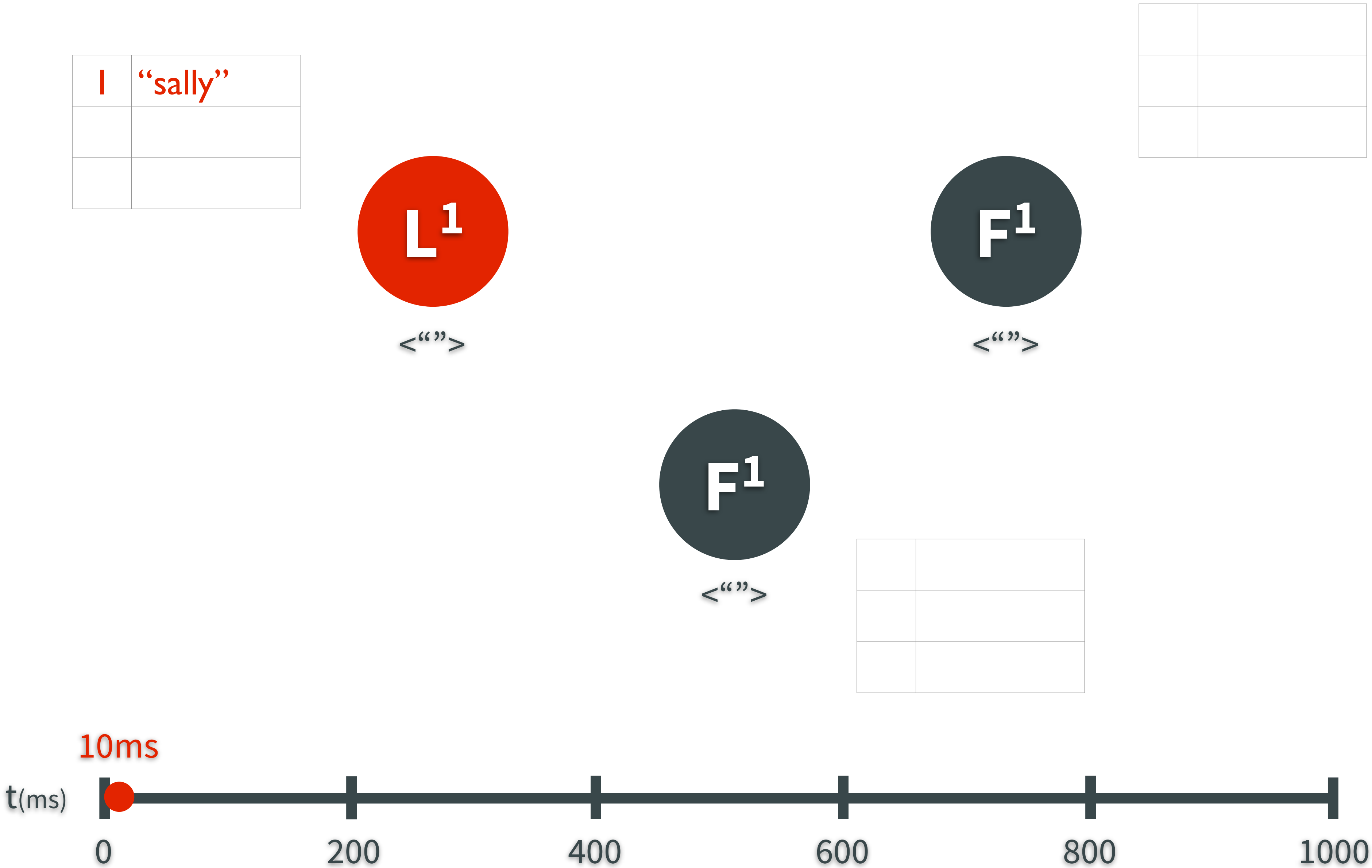
Log Replication

Log Replication



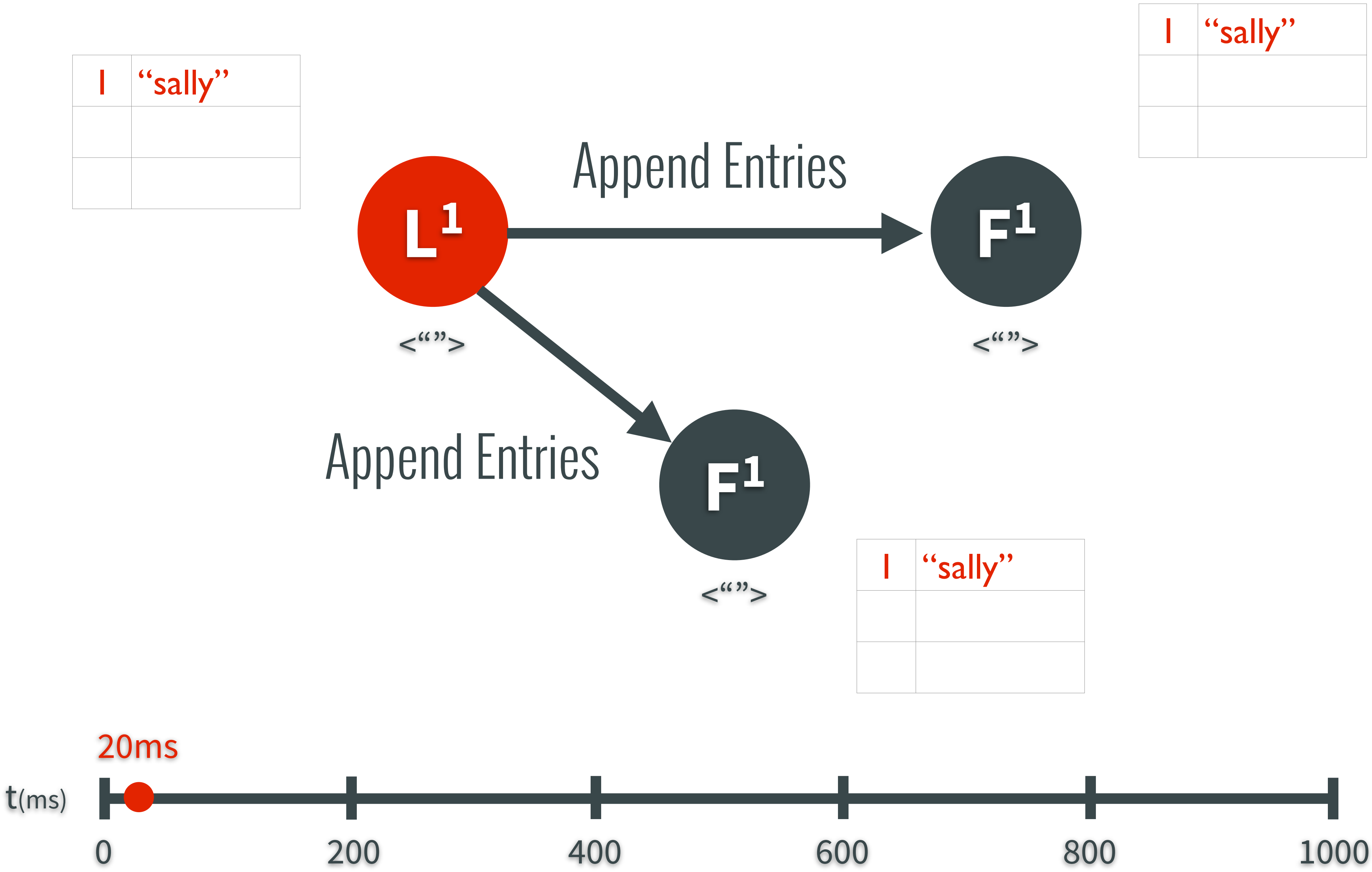
Log Replication

A new uncommitted log entry is added to the leader



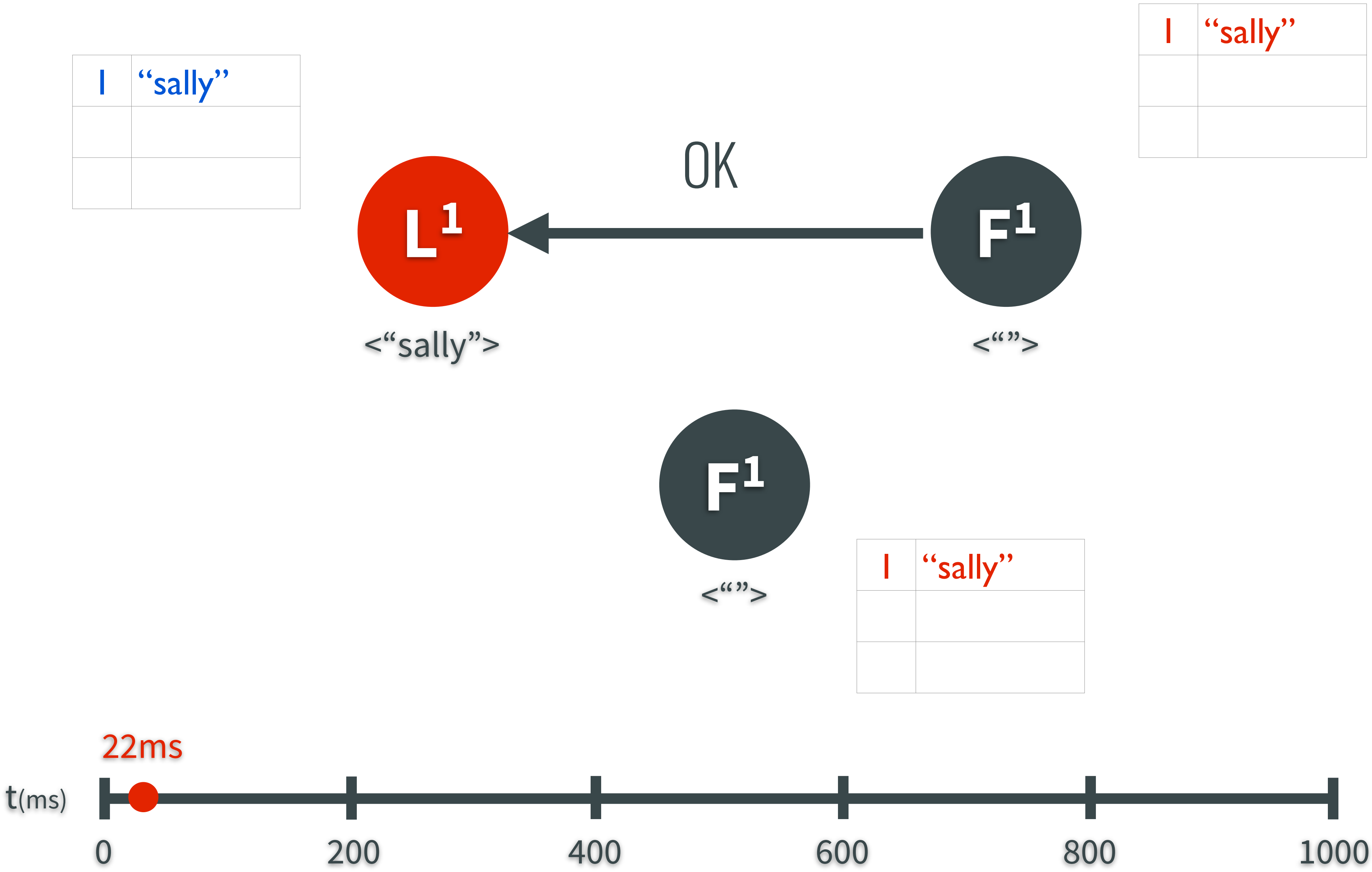
Log Replication

At the next heartbeat, the log entry is replicated to followers

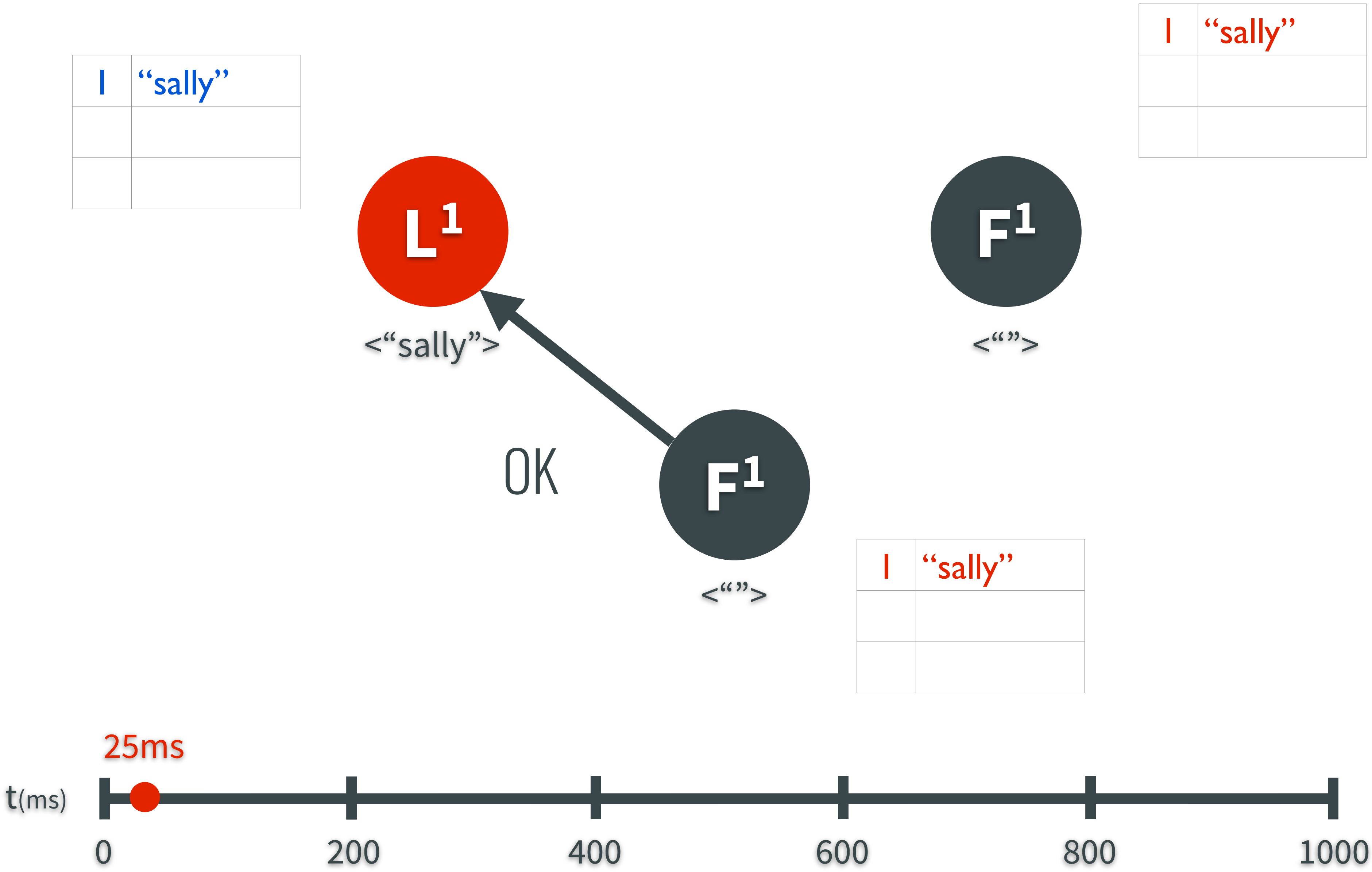


Log Replication

A majority of nodes have written the log entry written to disk so it becomes committed

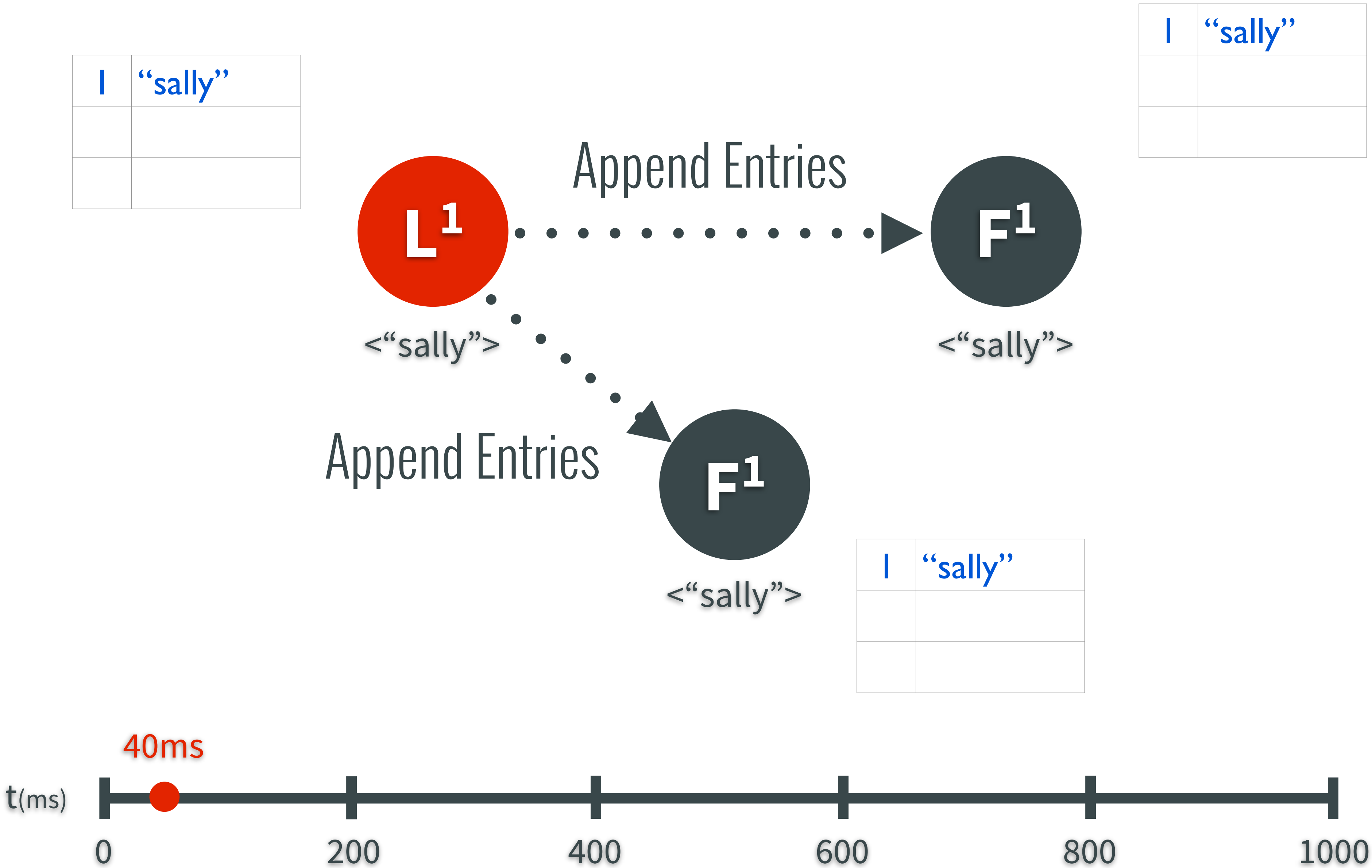


Log Replication

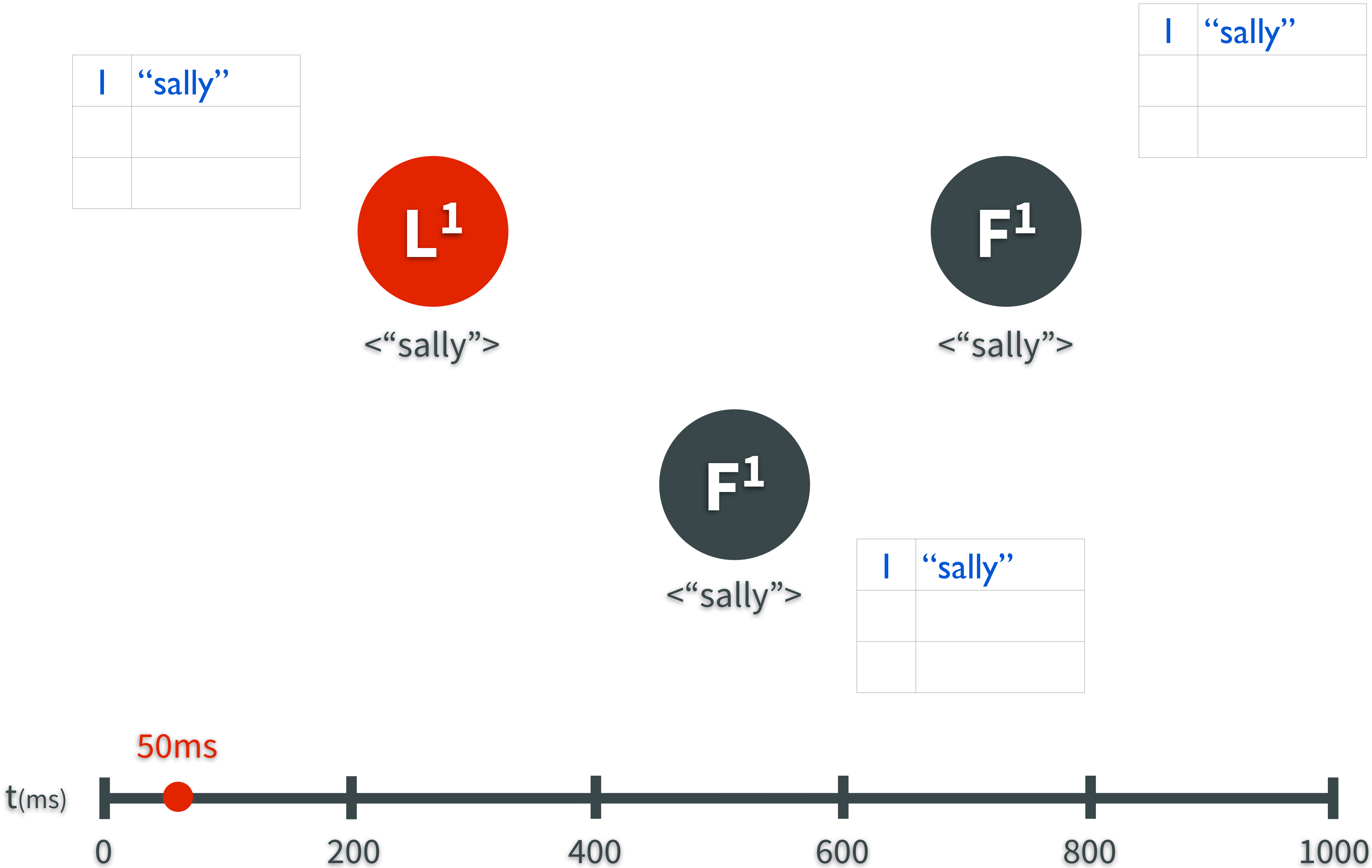


Log Replication

At the next heartbeat, the leader notifies followers of updated committed entries

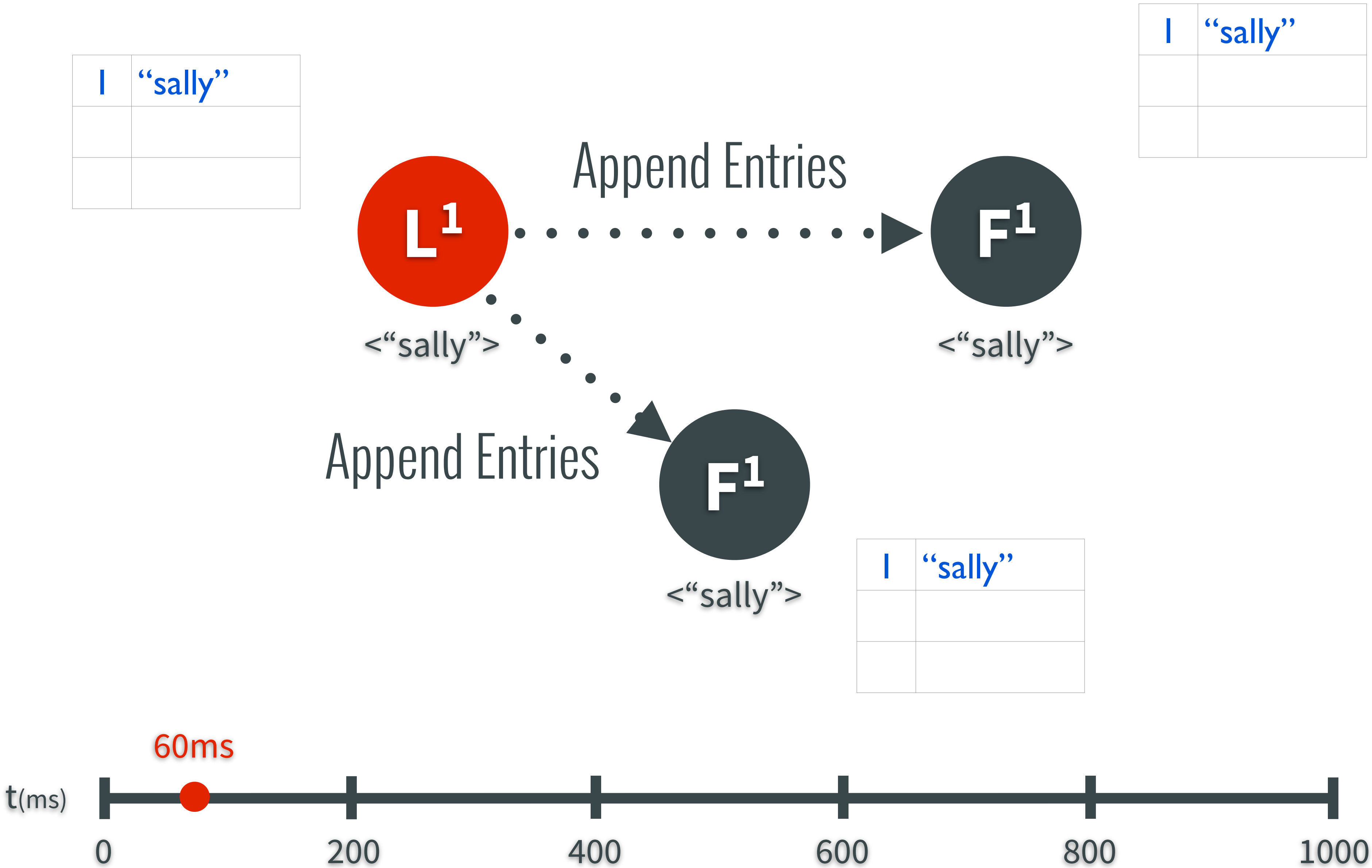


Log Replication

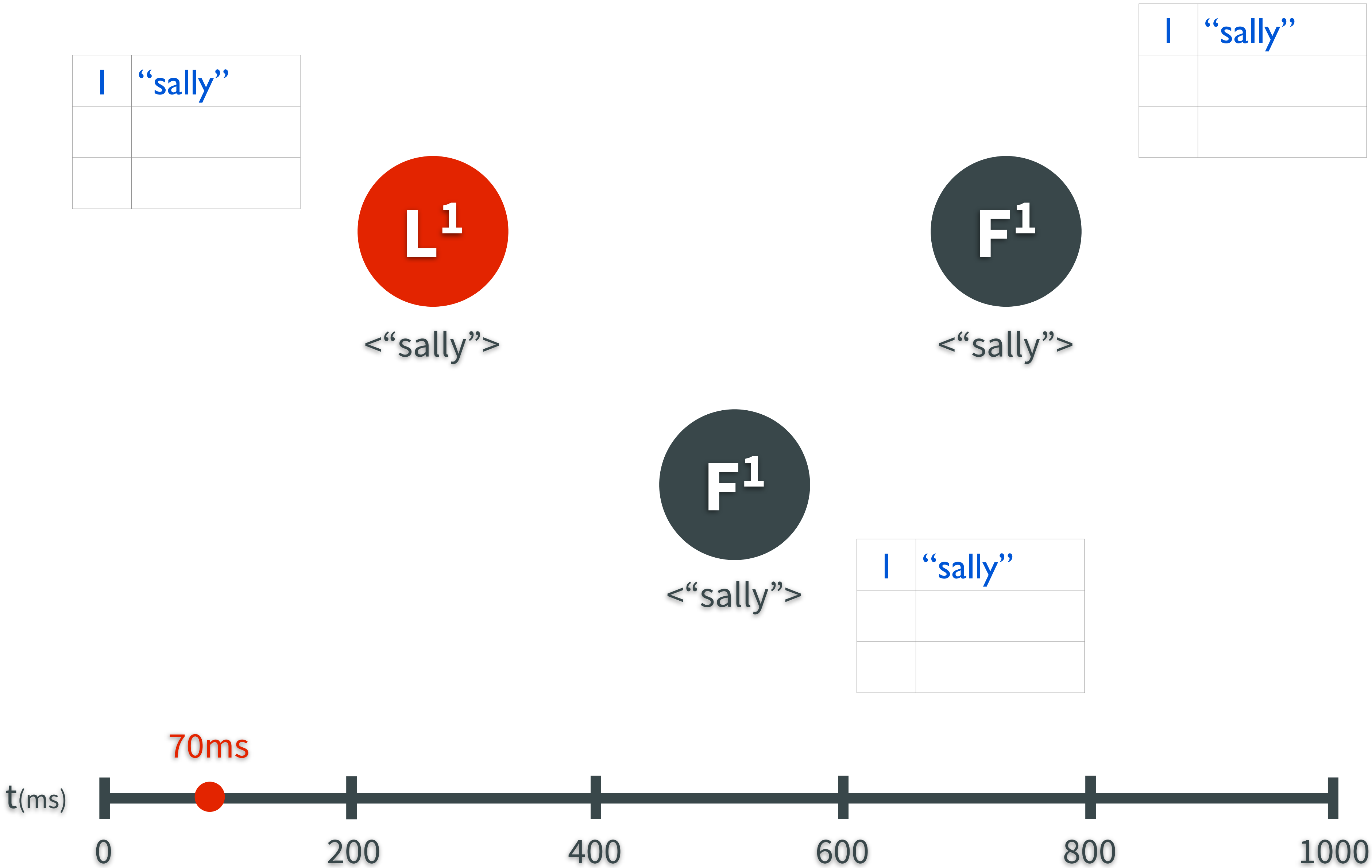


Log Replication

At the next heartbeat, no new log information is sent

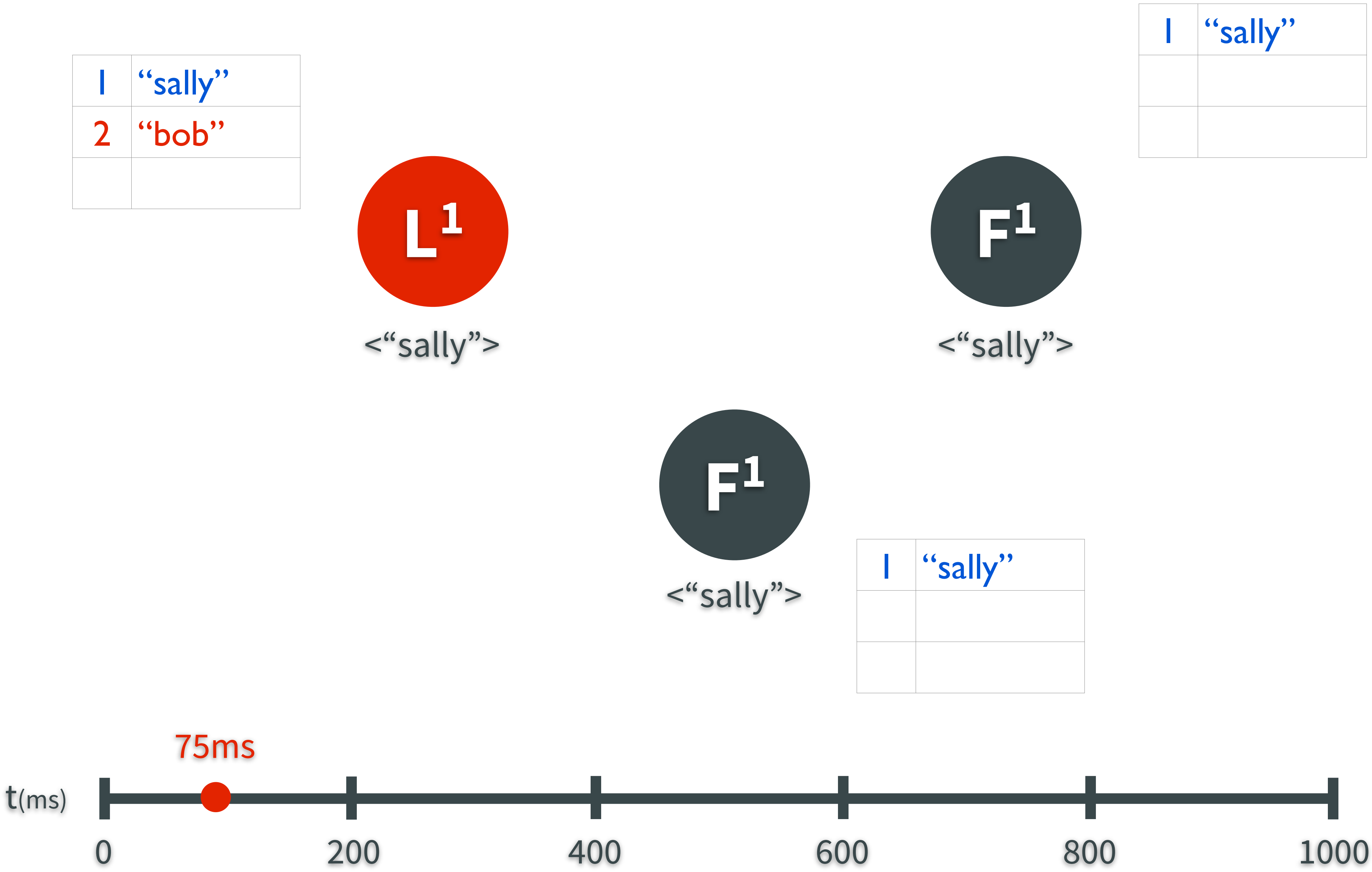


Log Replication



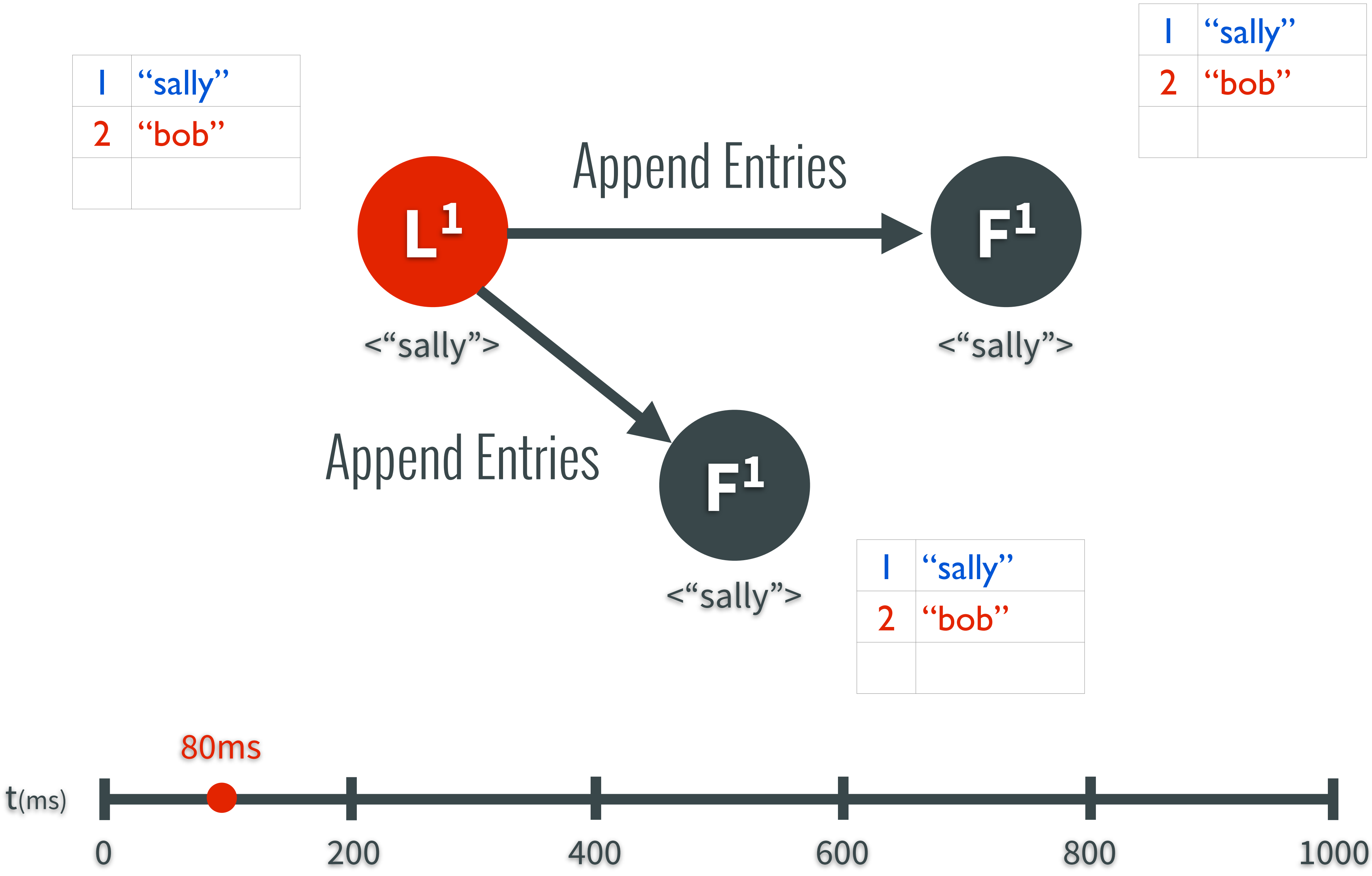
Log Replication

A new uncommitted log entry is added to the leader



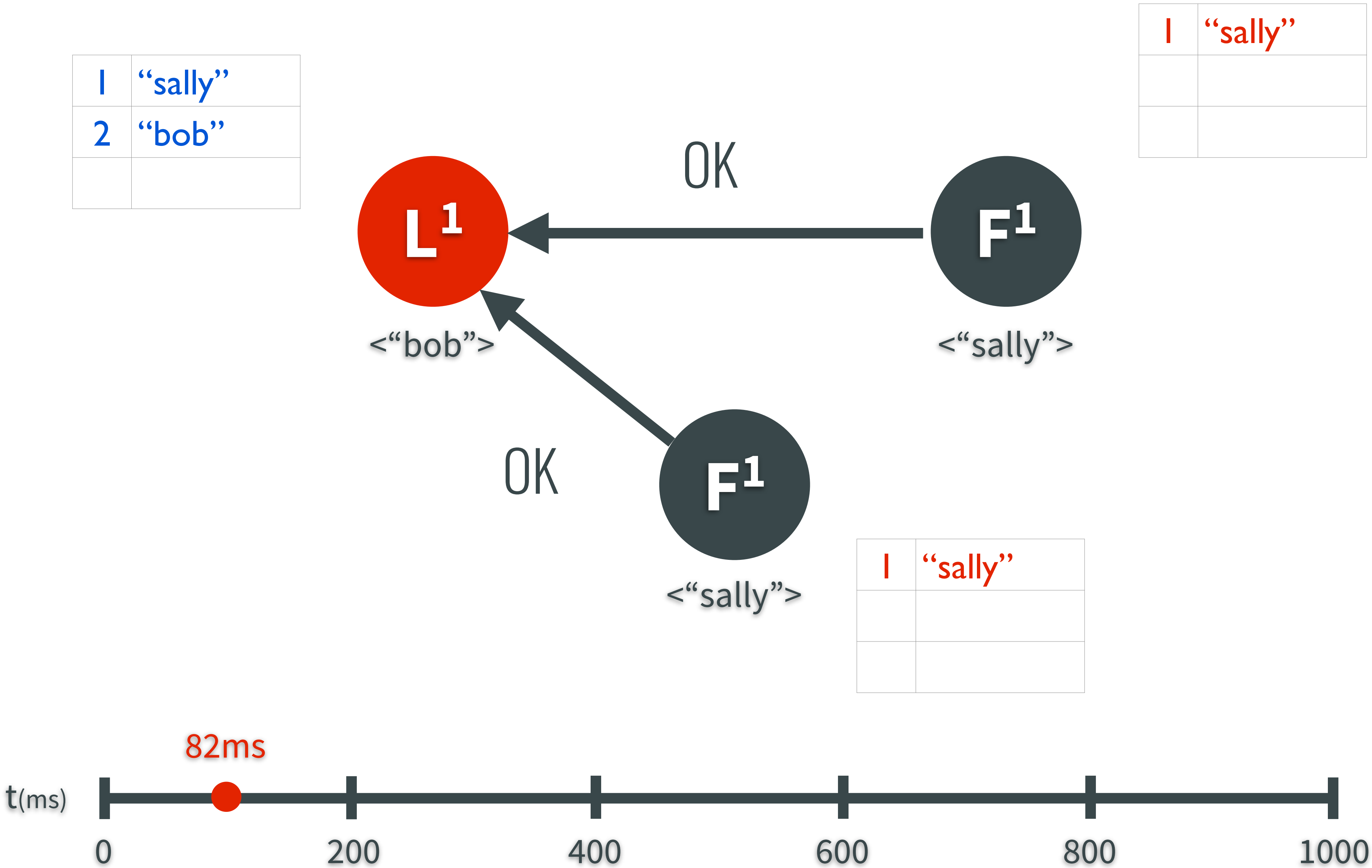
Log Replication

At the next heartbeat, the entry is replicated to the followers



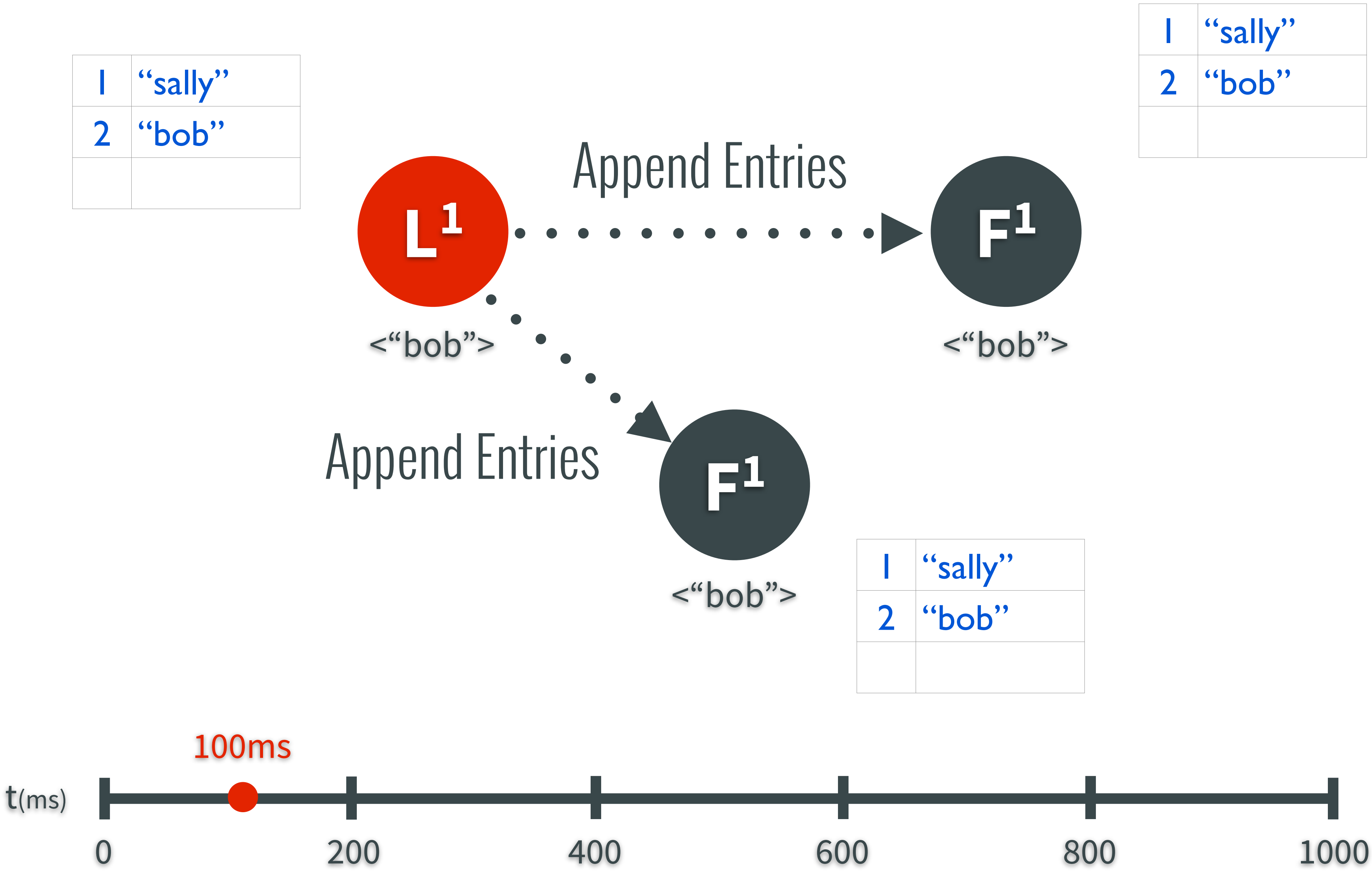
Log Replication

The entry is committed once the followers acknowledge the request



Log Replication

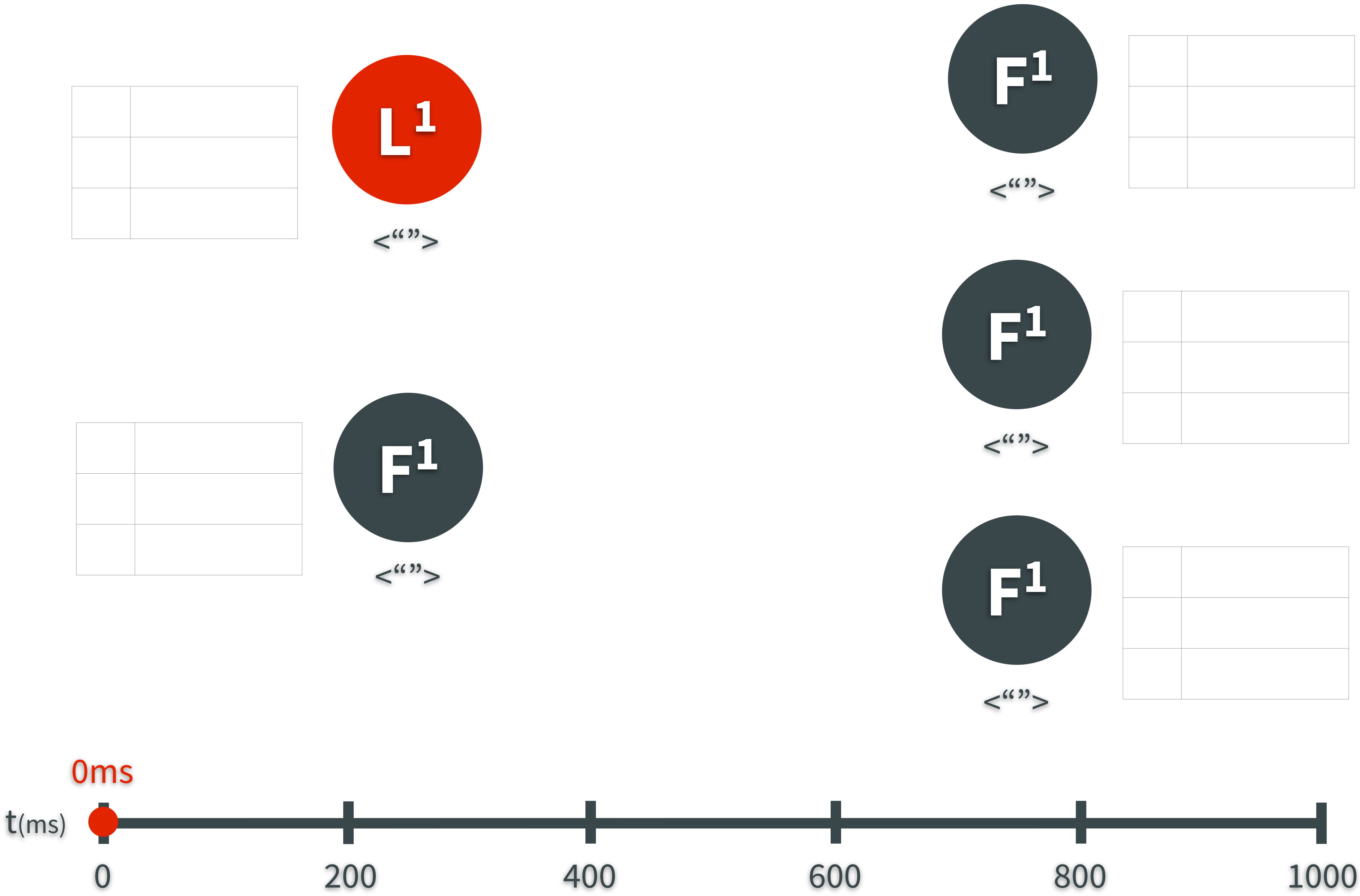
At the next heartbeat, the leader notifies the followers of the new committed entry



Log Replication

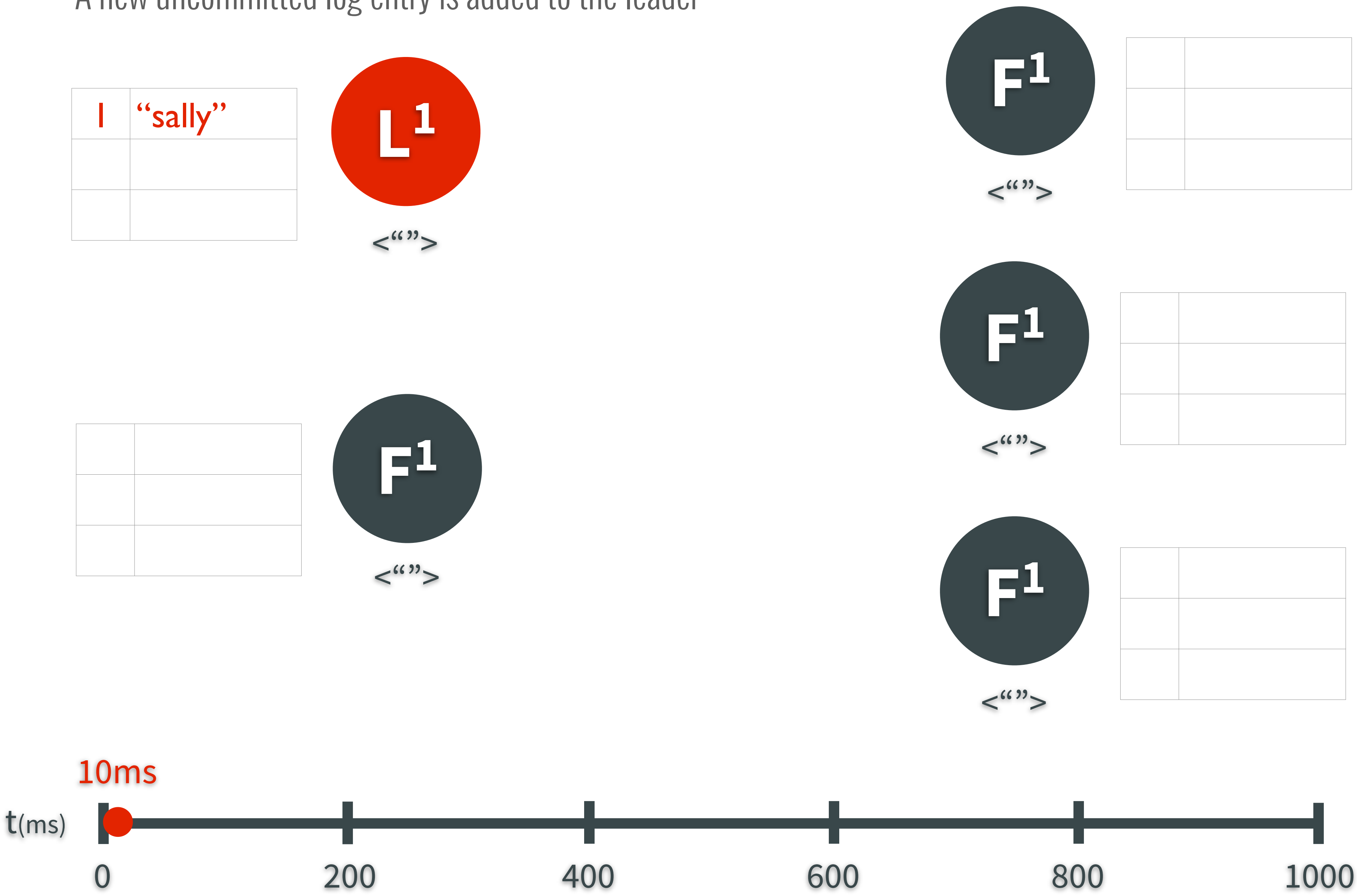
(with Network Partitions)

Log Replication



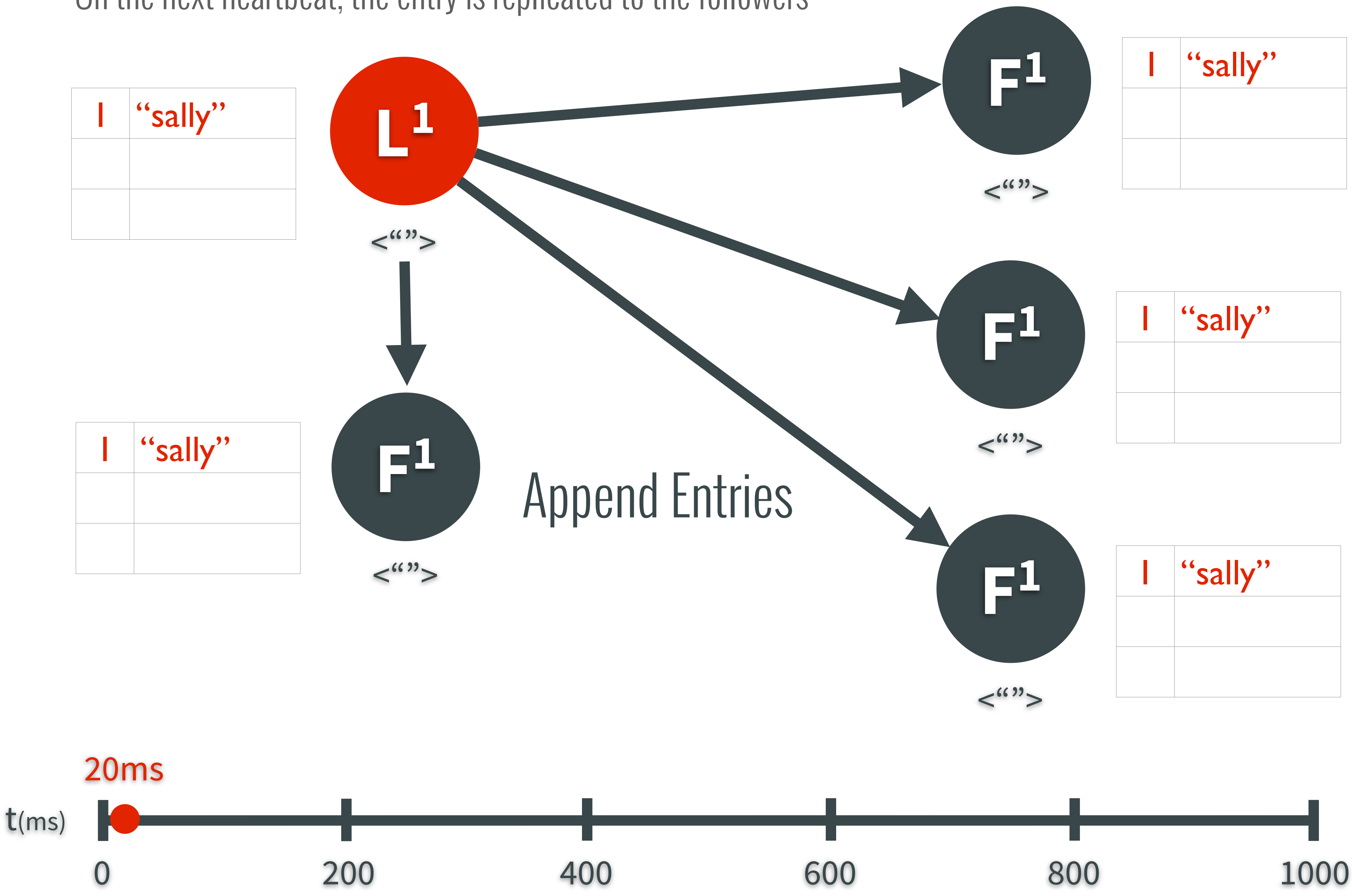
Log Replication

A new uncommitted log entry is added to the leader



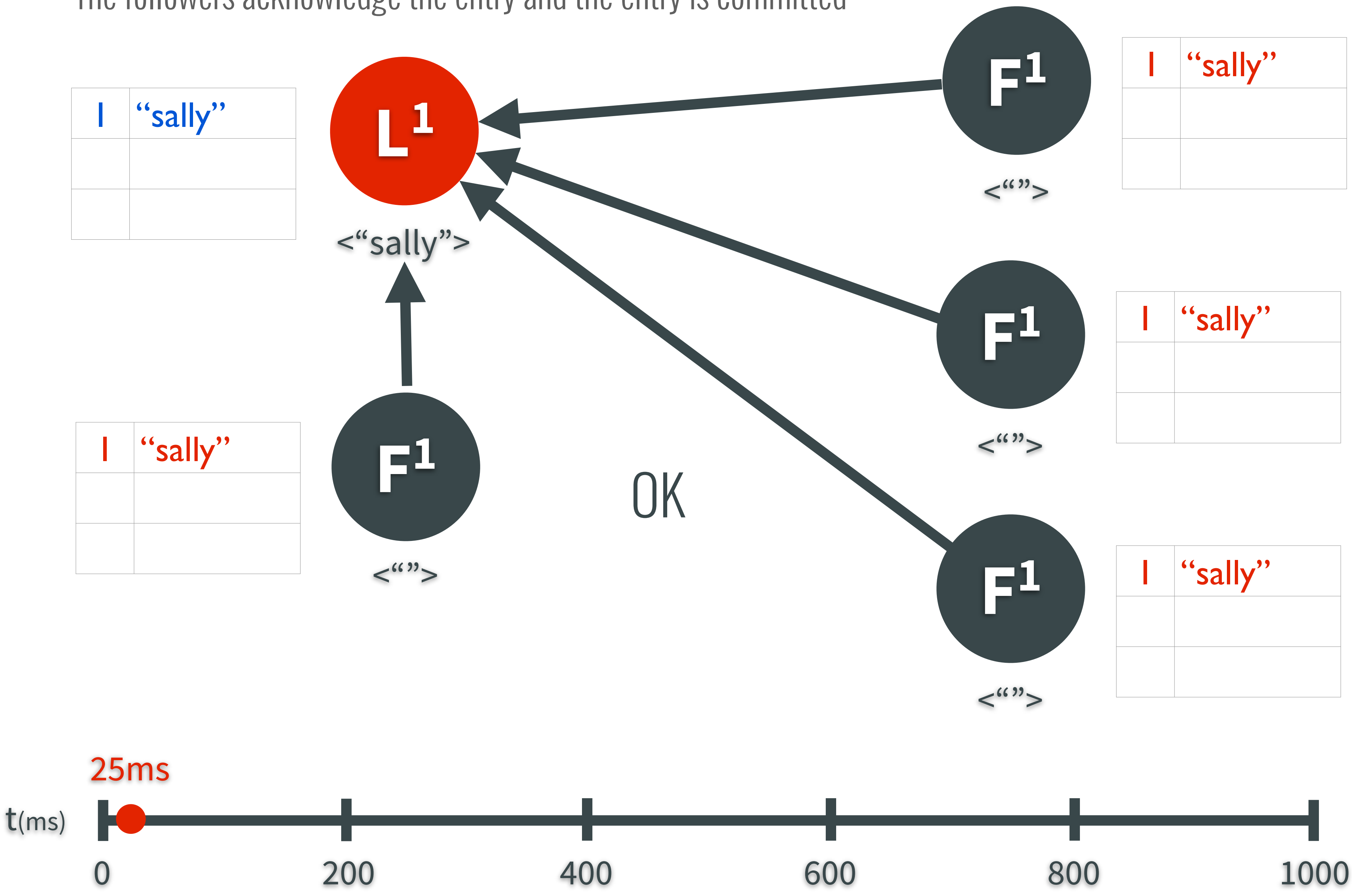
Log Replication

On the next heartbeat, the entry is replicated to the followers



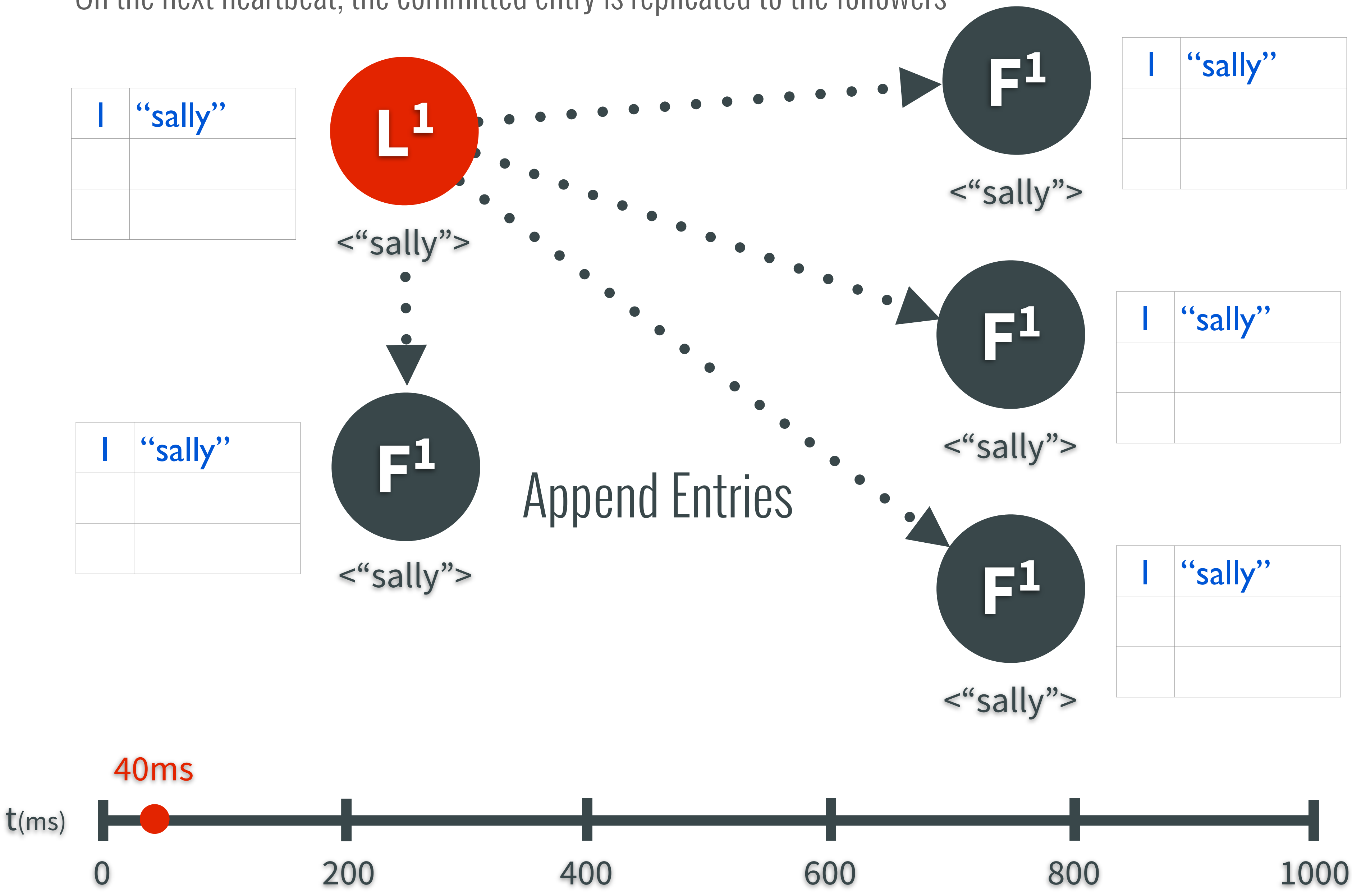
Log Replication

The followers acknowledge the entry and the entry is committed

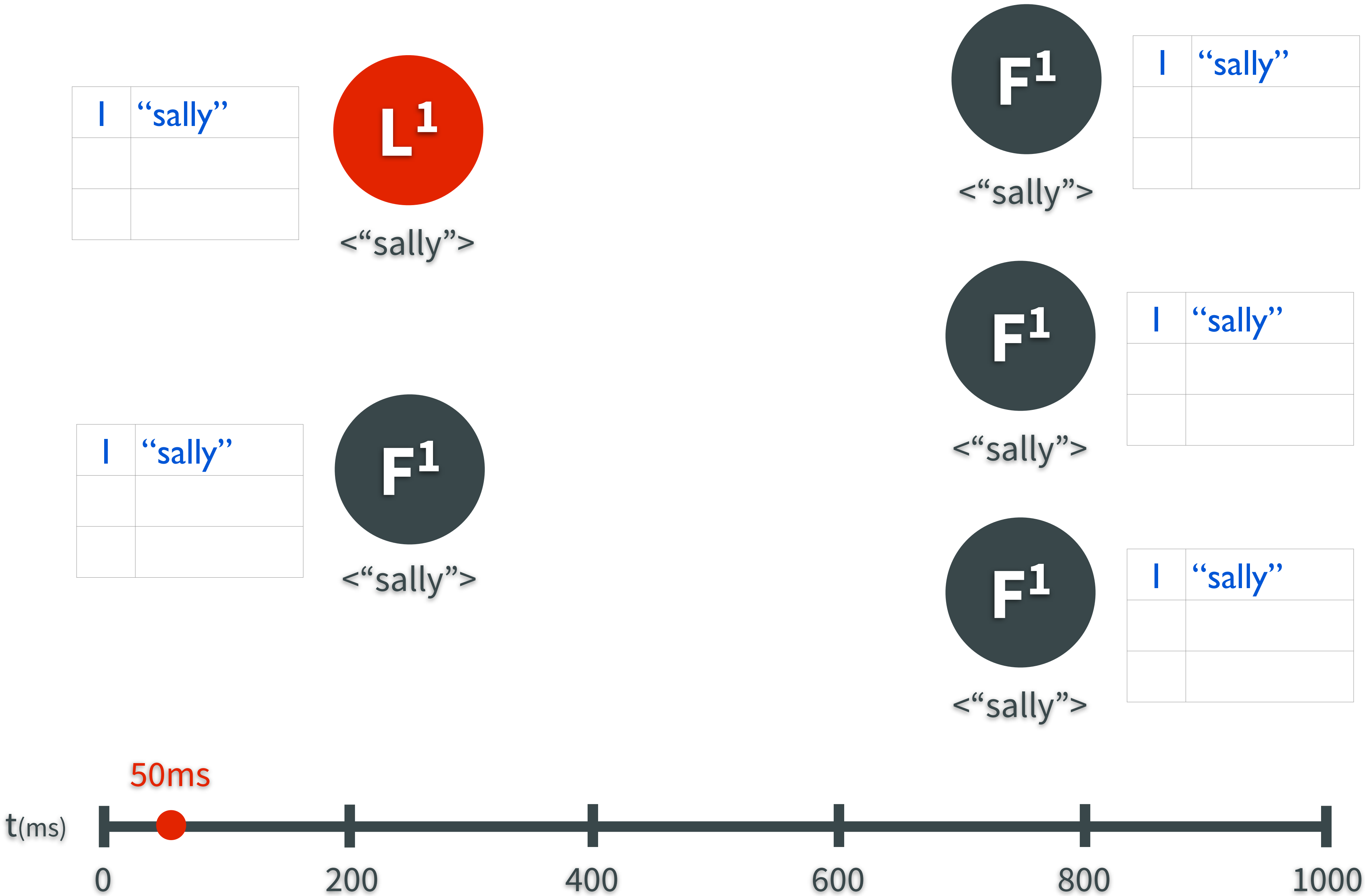


Log Replication

On the next heartbeat, the committed entry is replicated to the followers

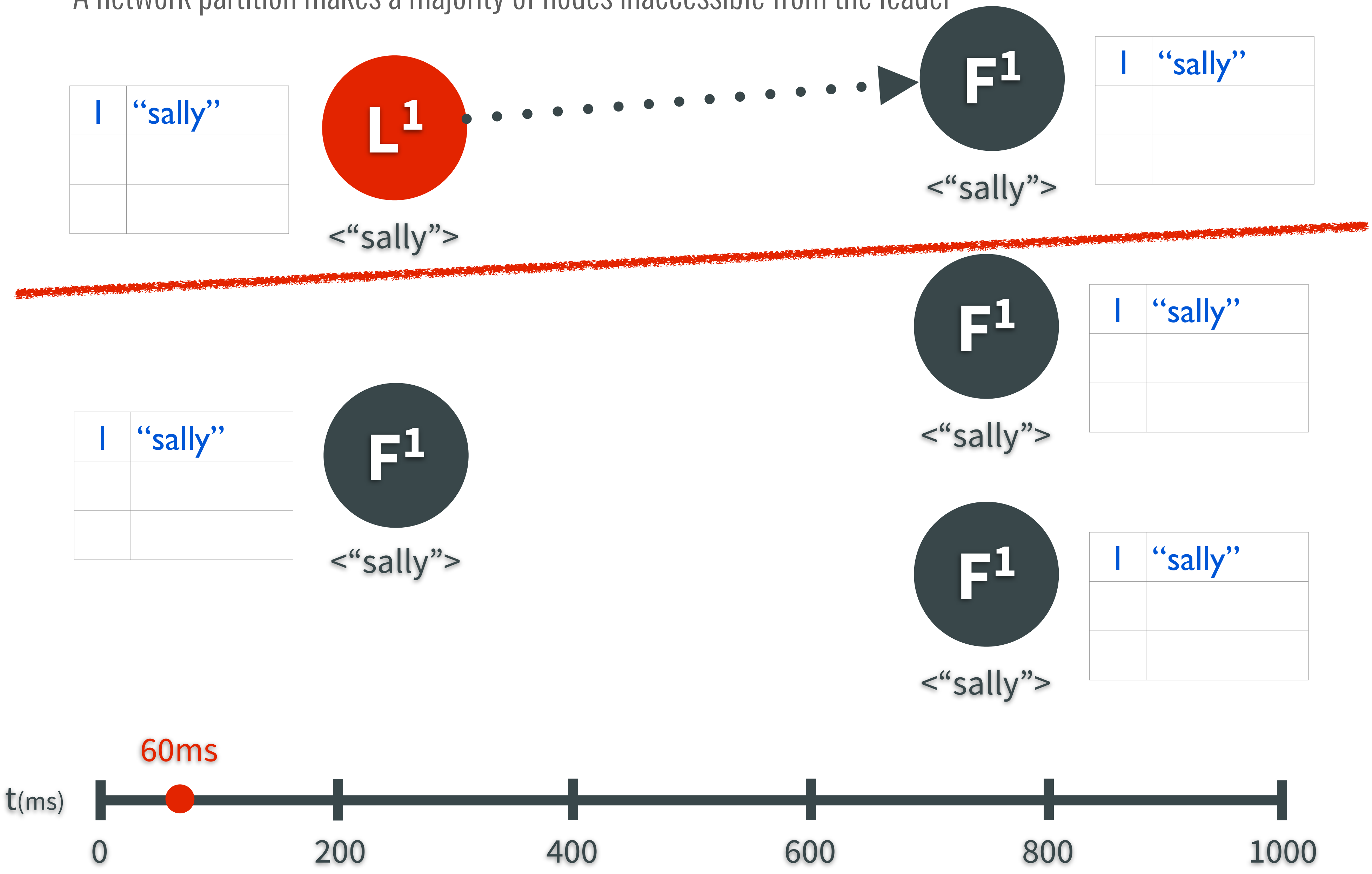


Log Replication



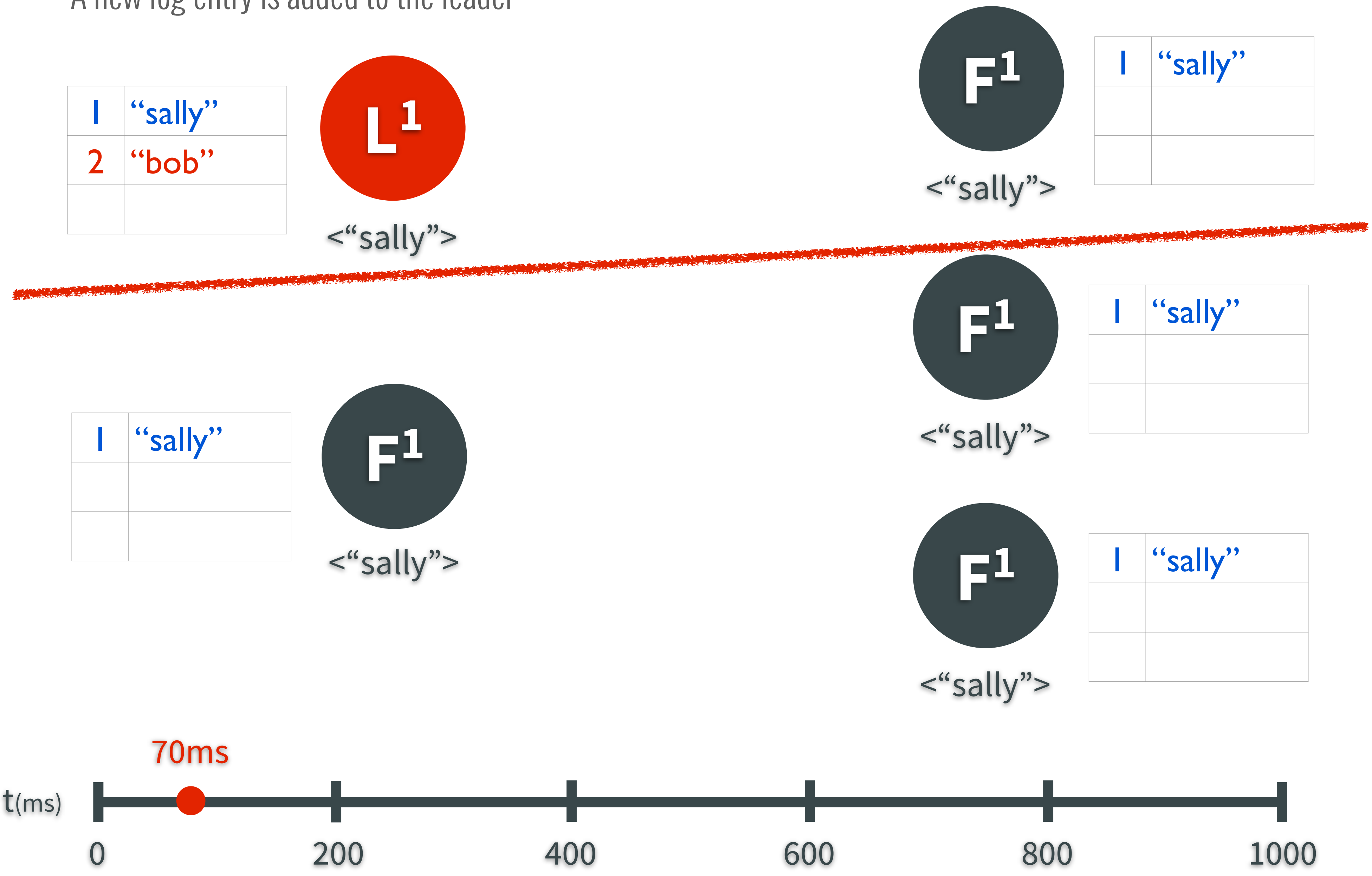
Log Replication

A network partition makes a majority of nodes inaccessible from the leader



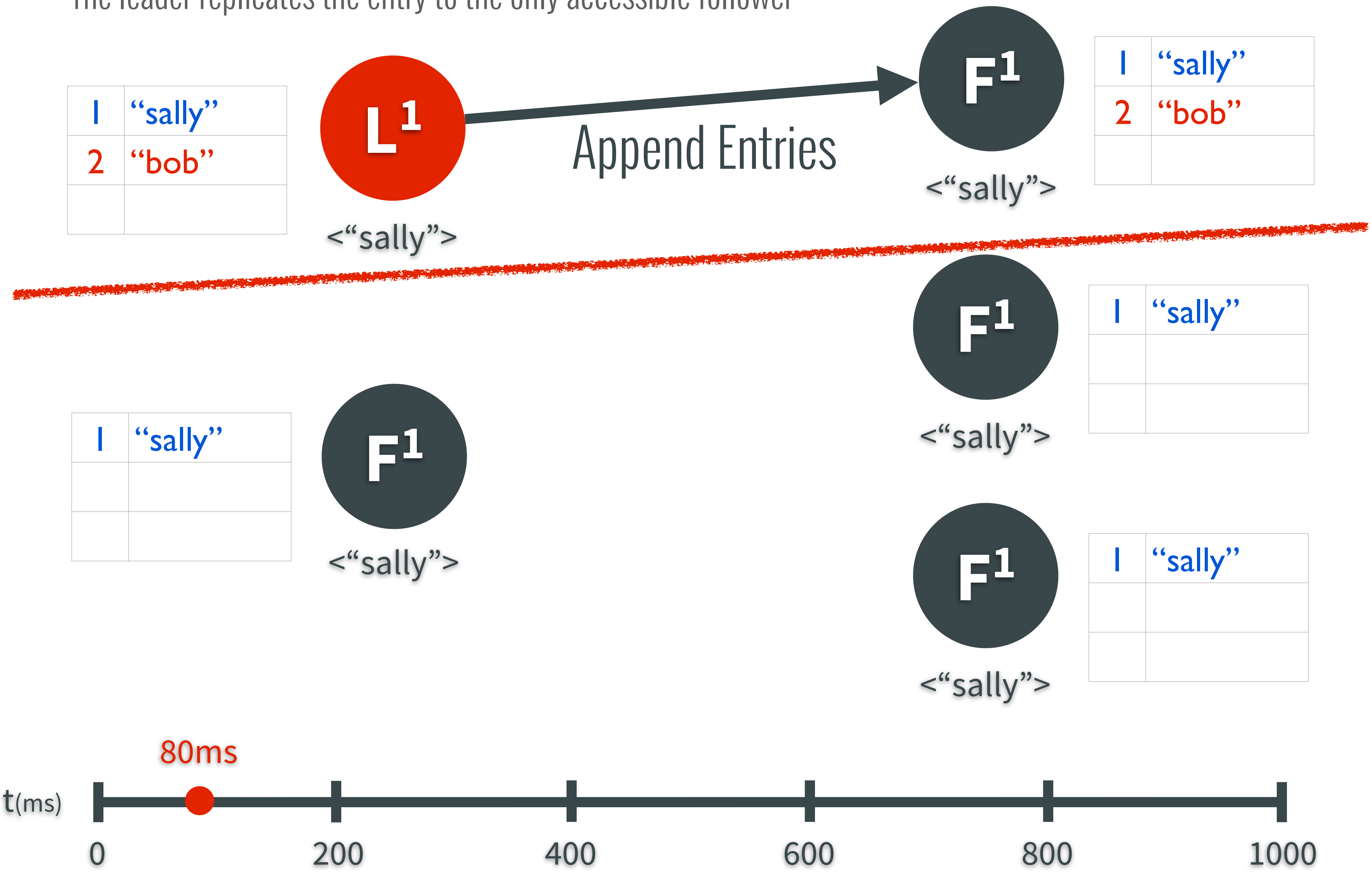
Log Replication

A new log entry is added to the leader



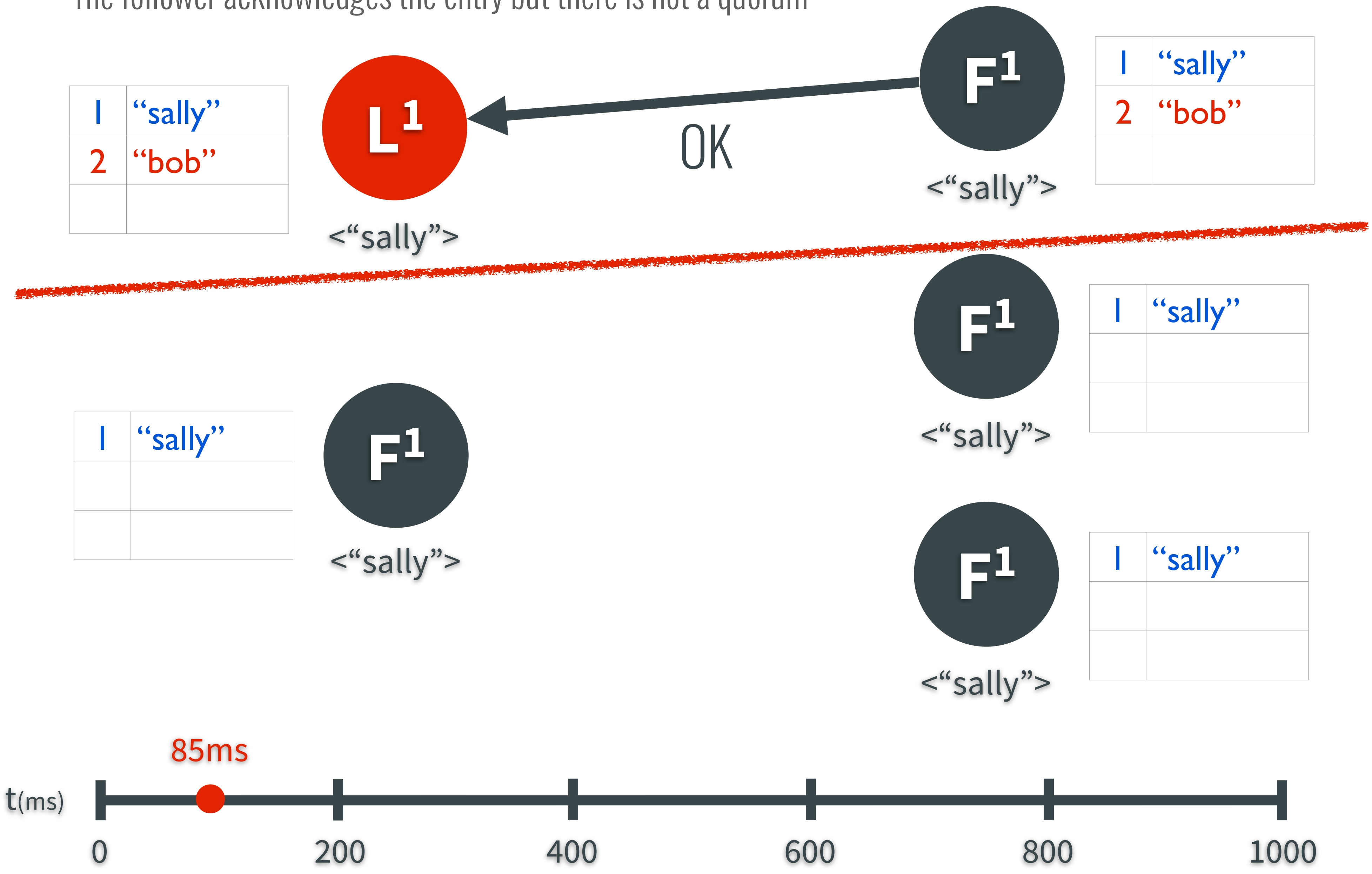
Log Replication

The leader replicates the entry to the only accessible follower

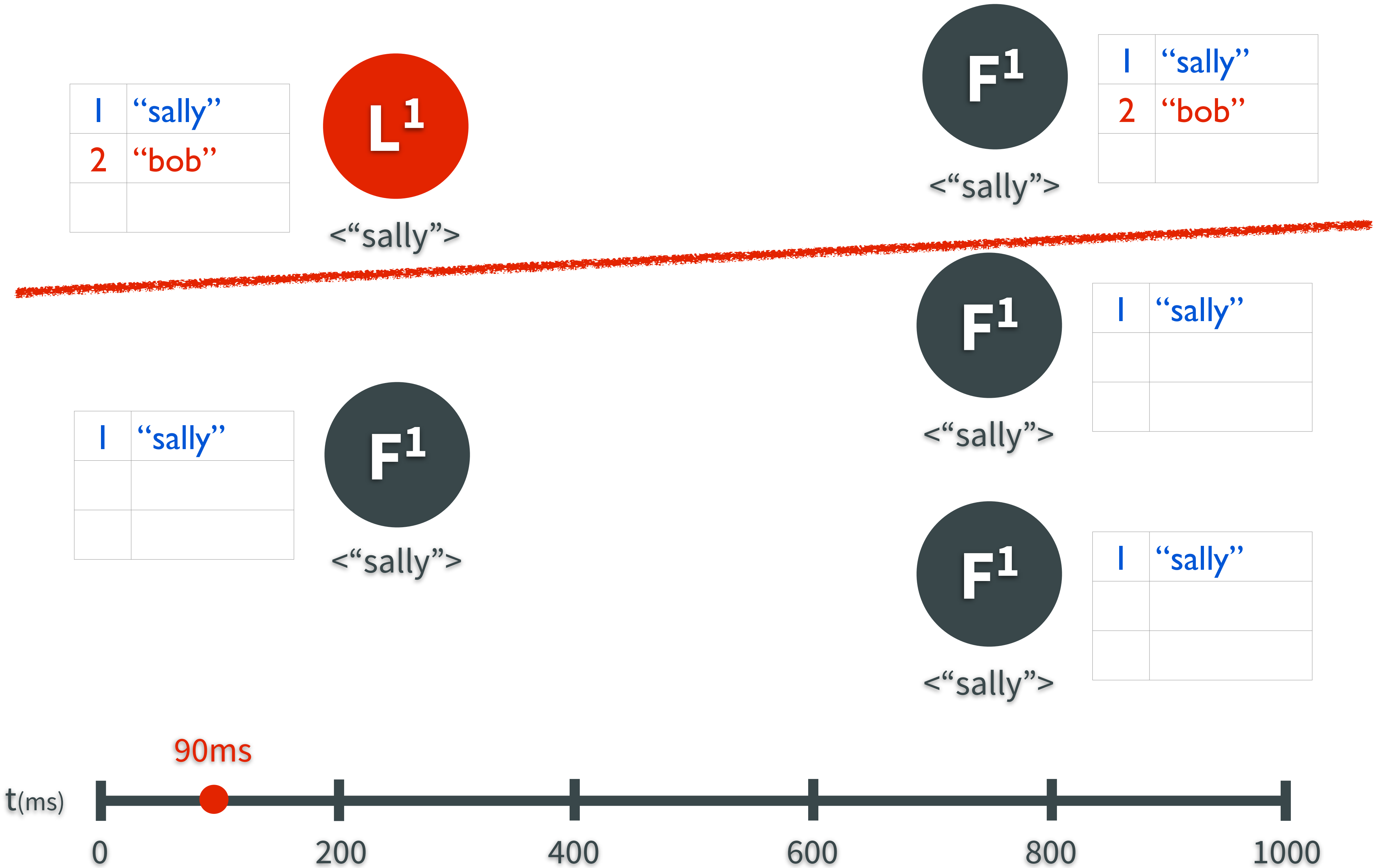


Log Replication

The follower acknowledges the entry but there is not a quorum

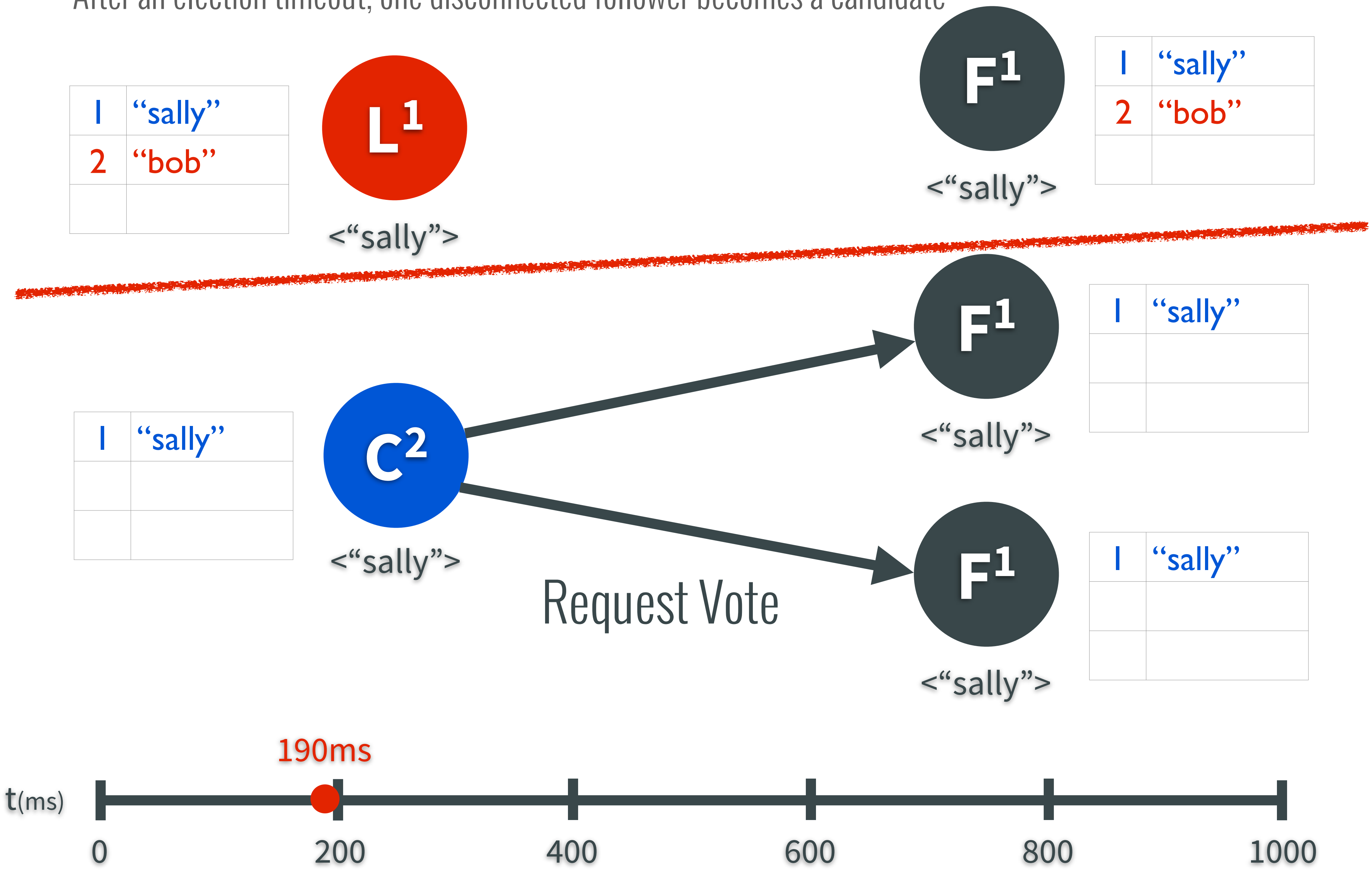


Log Replication



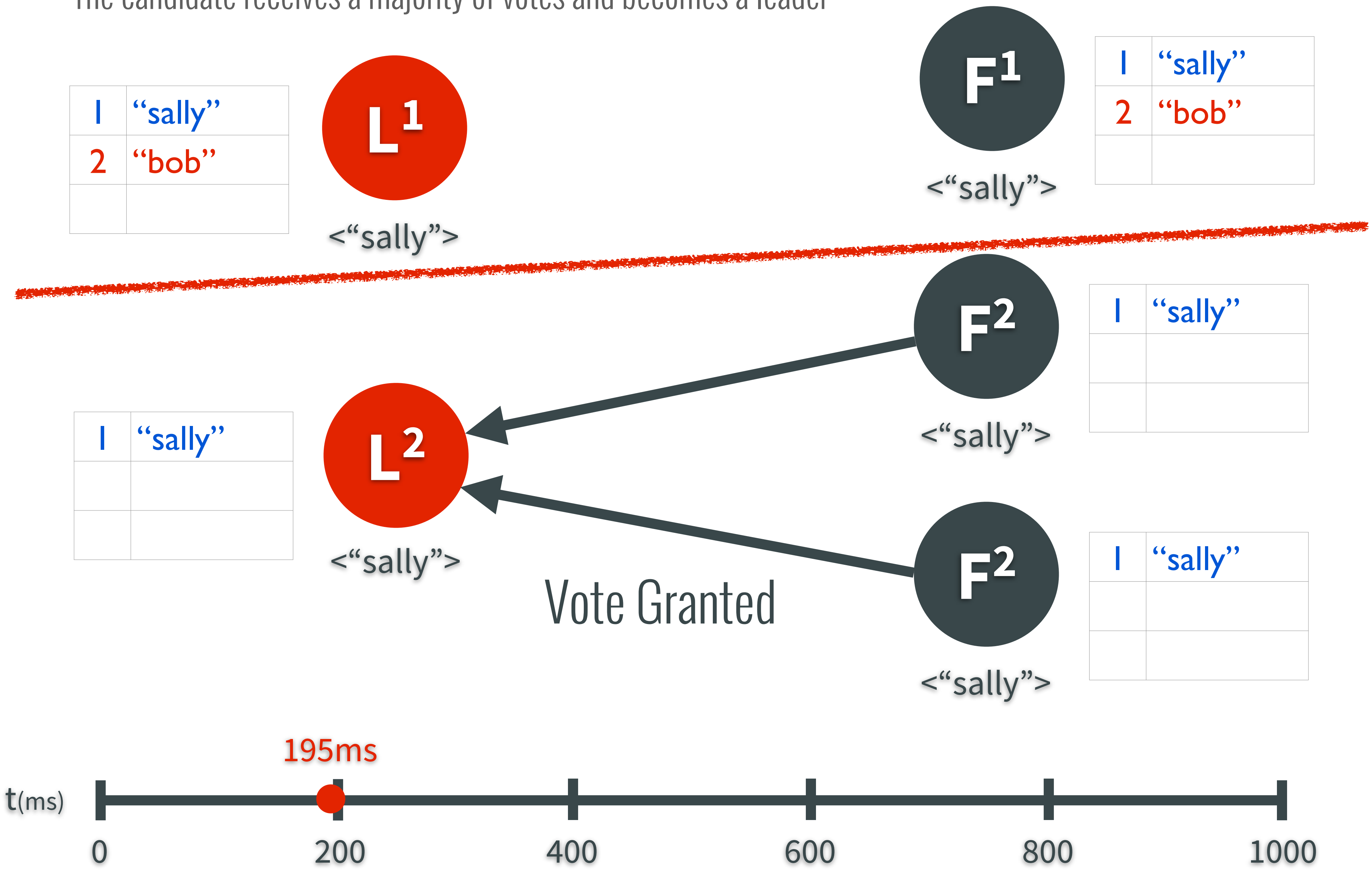
Log Replication

After an election timeout, one disconnected follower becomes a candidate

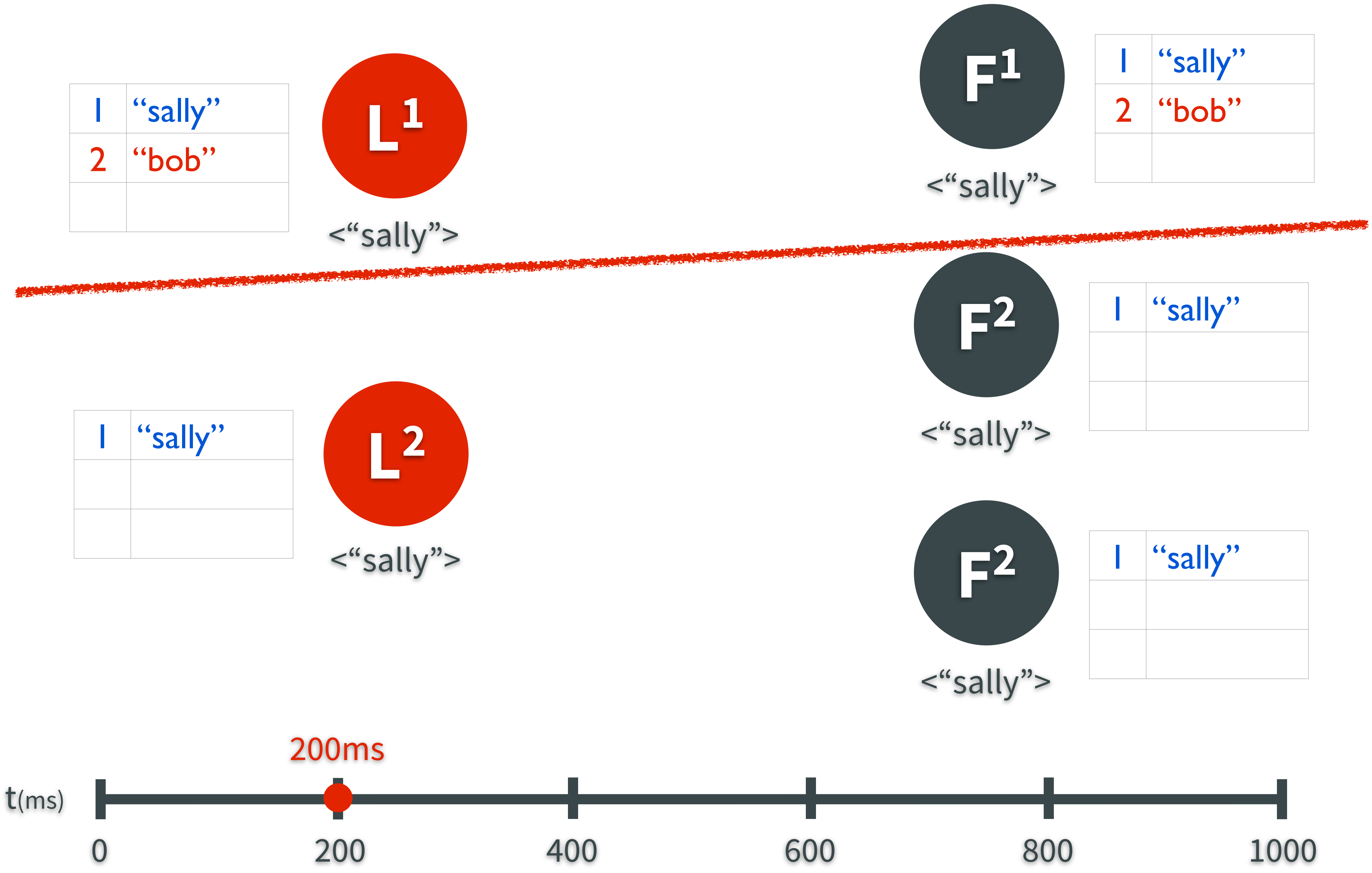


Log Replication

The candidate receives a majority of votes and becomes a leader

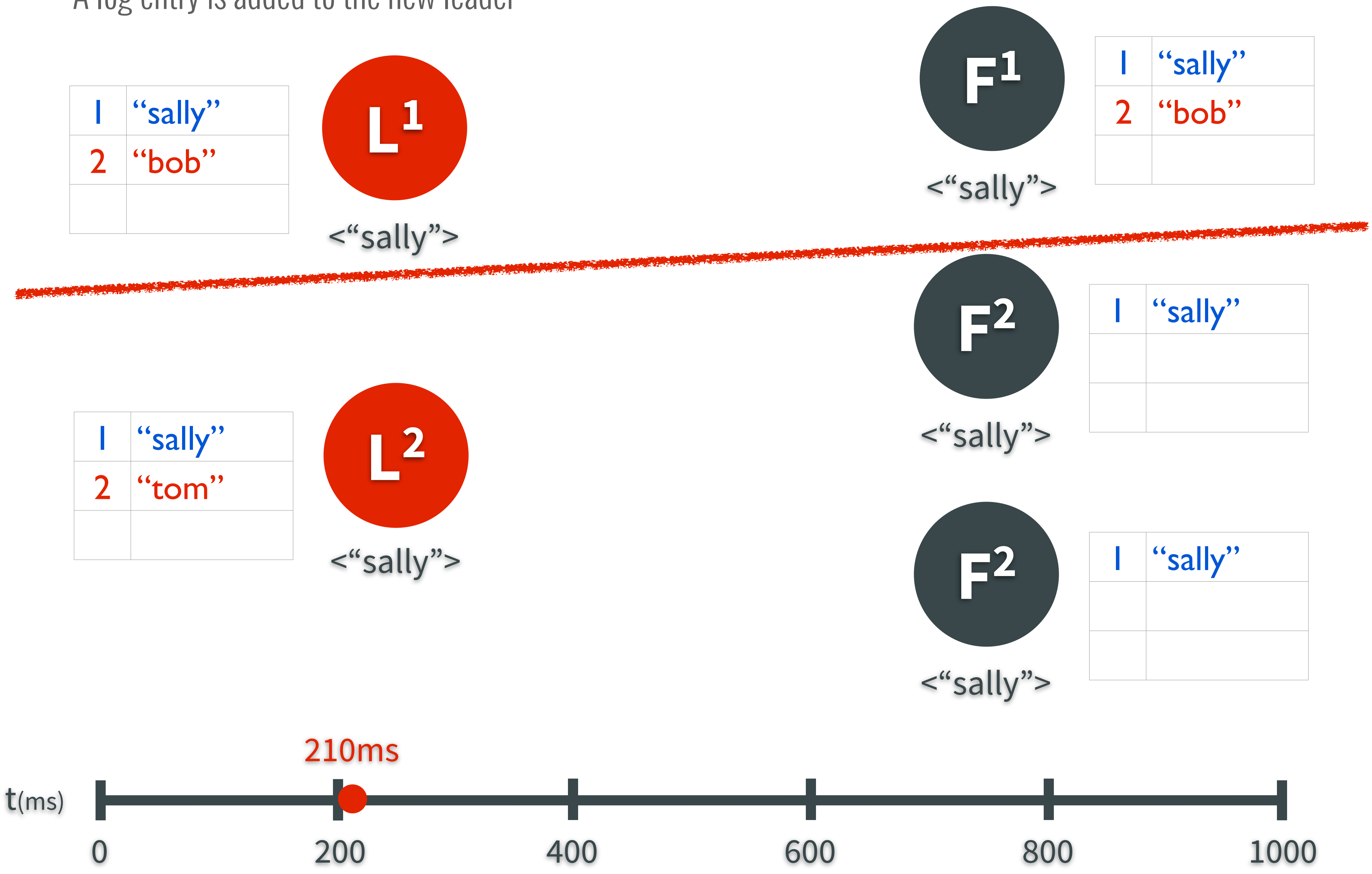


Log Replication



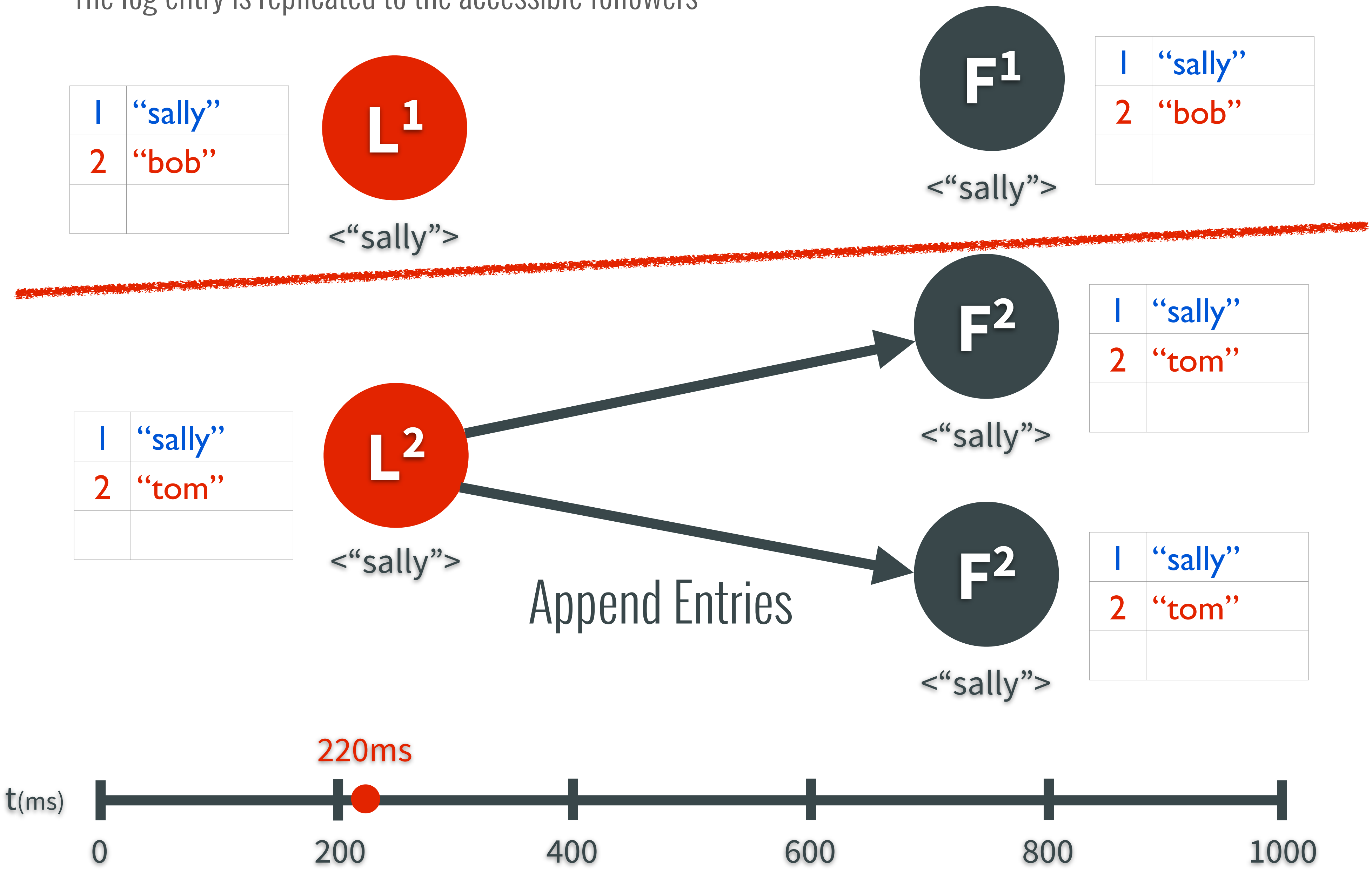
Log Replication

A log entry is added to the new leader



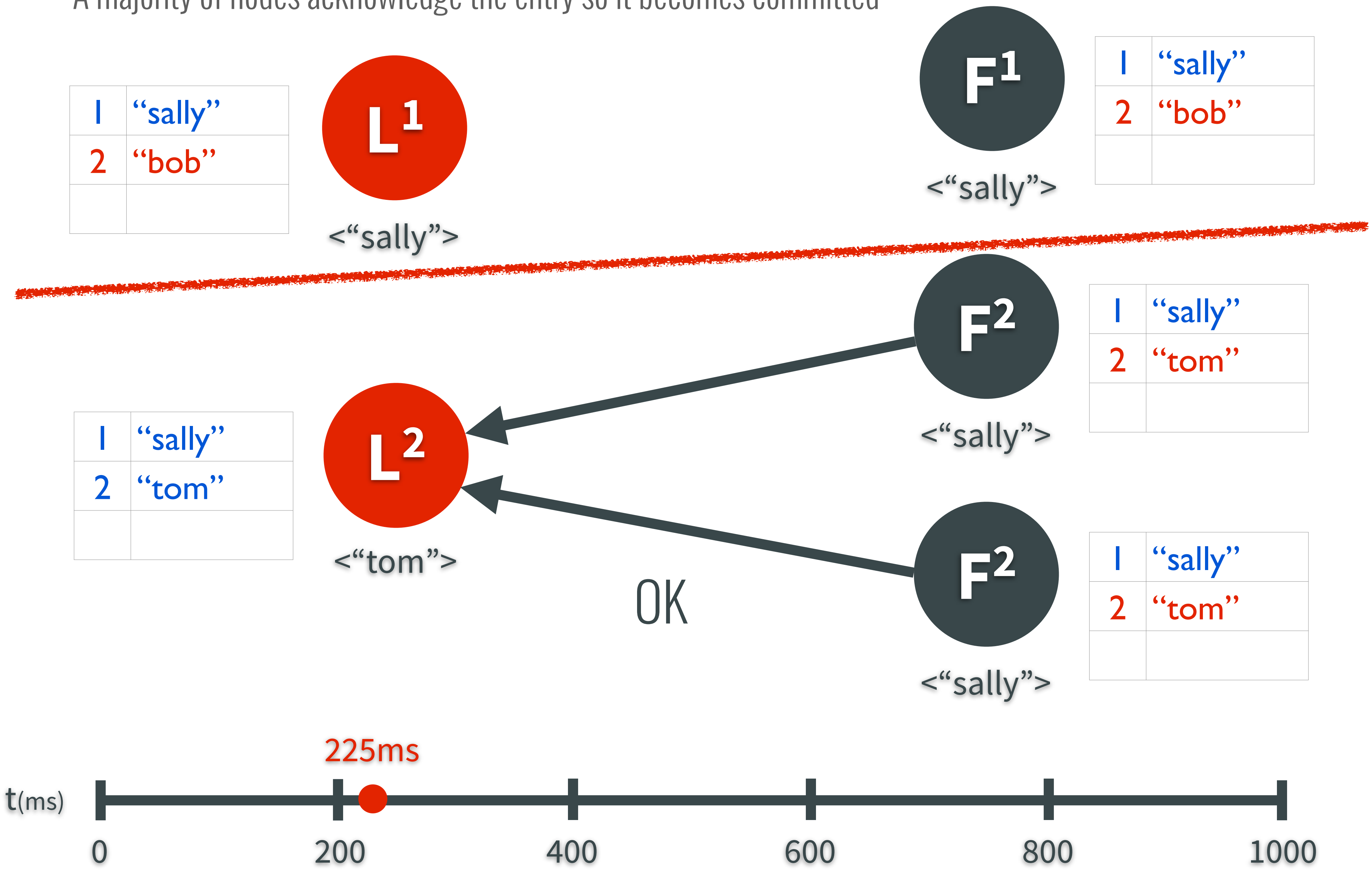
Log Replication

The log entry is replicated to the accessible followers



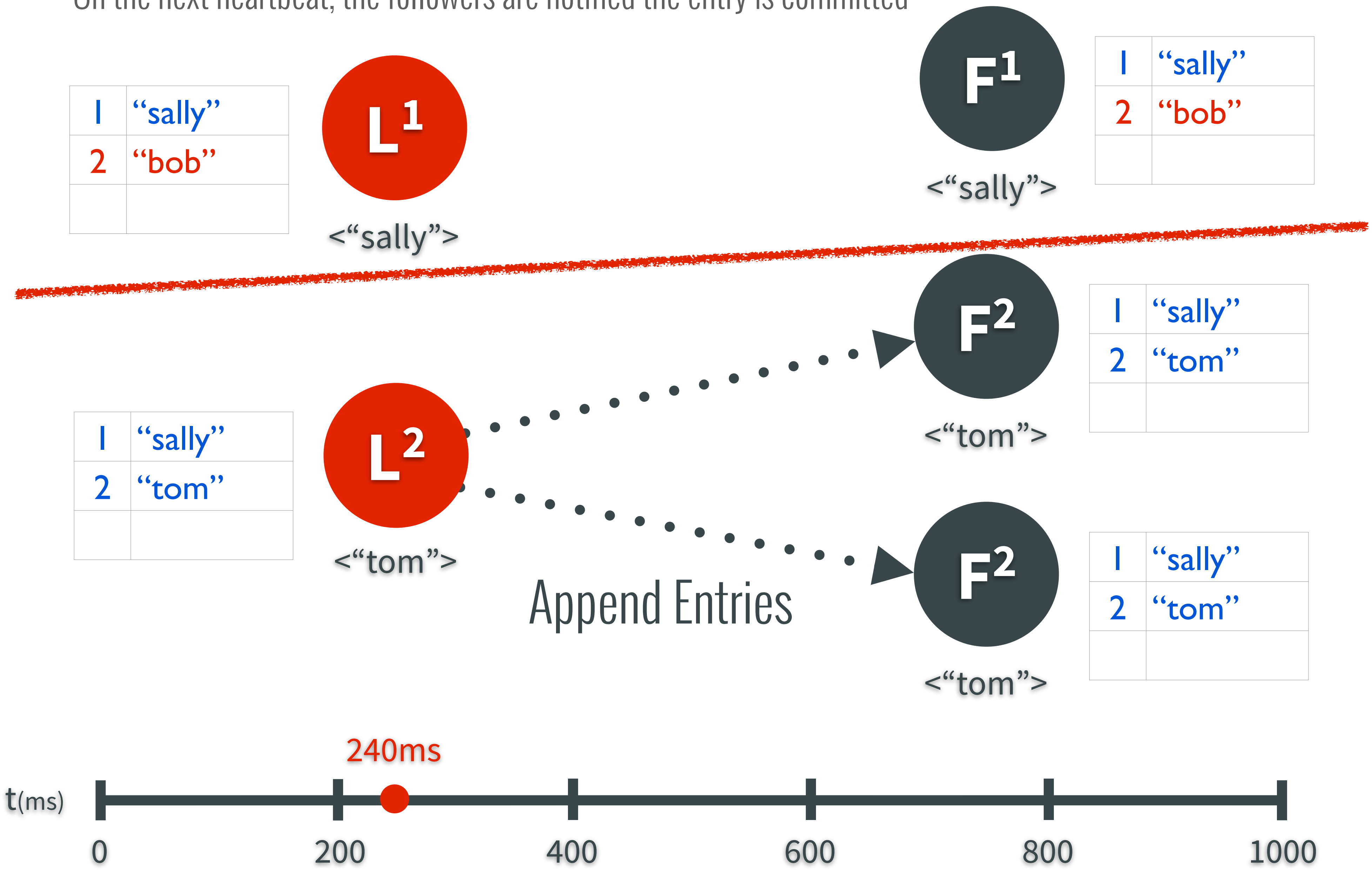
Log Replication

A majority of nodes acknowledge the entry so it becomes committed

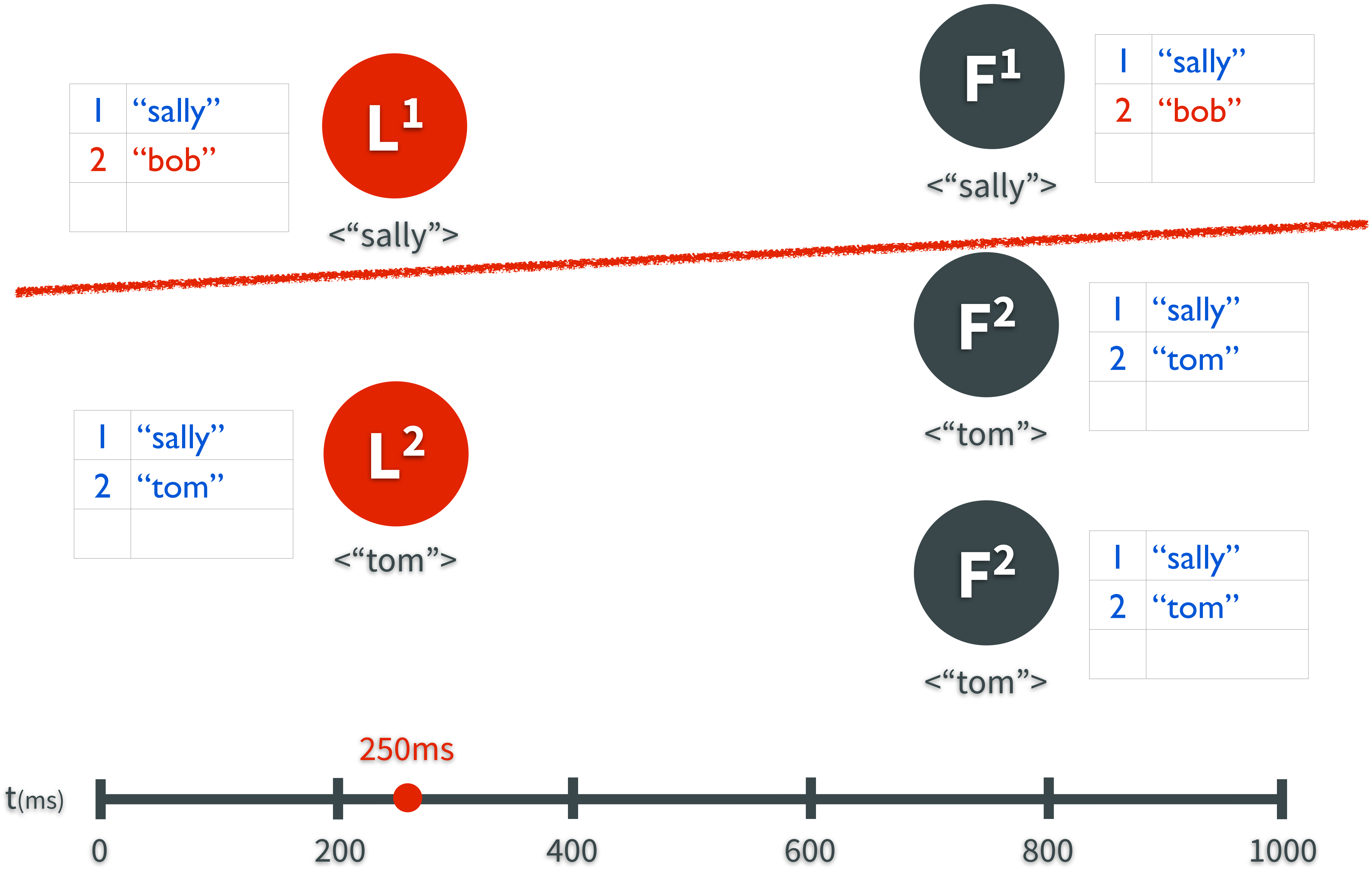


Log Replication

On the next heartbeat, the followers are notified the entry is committed

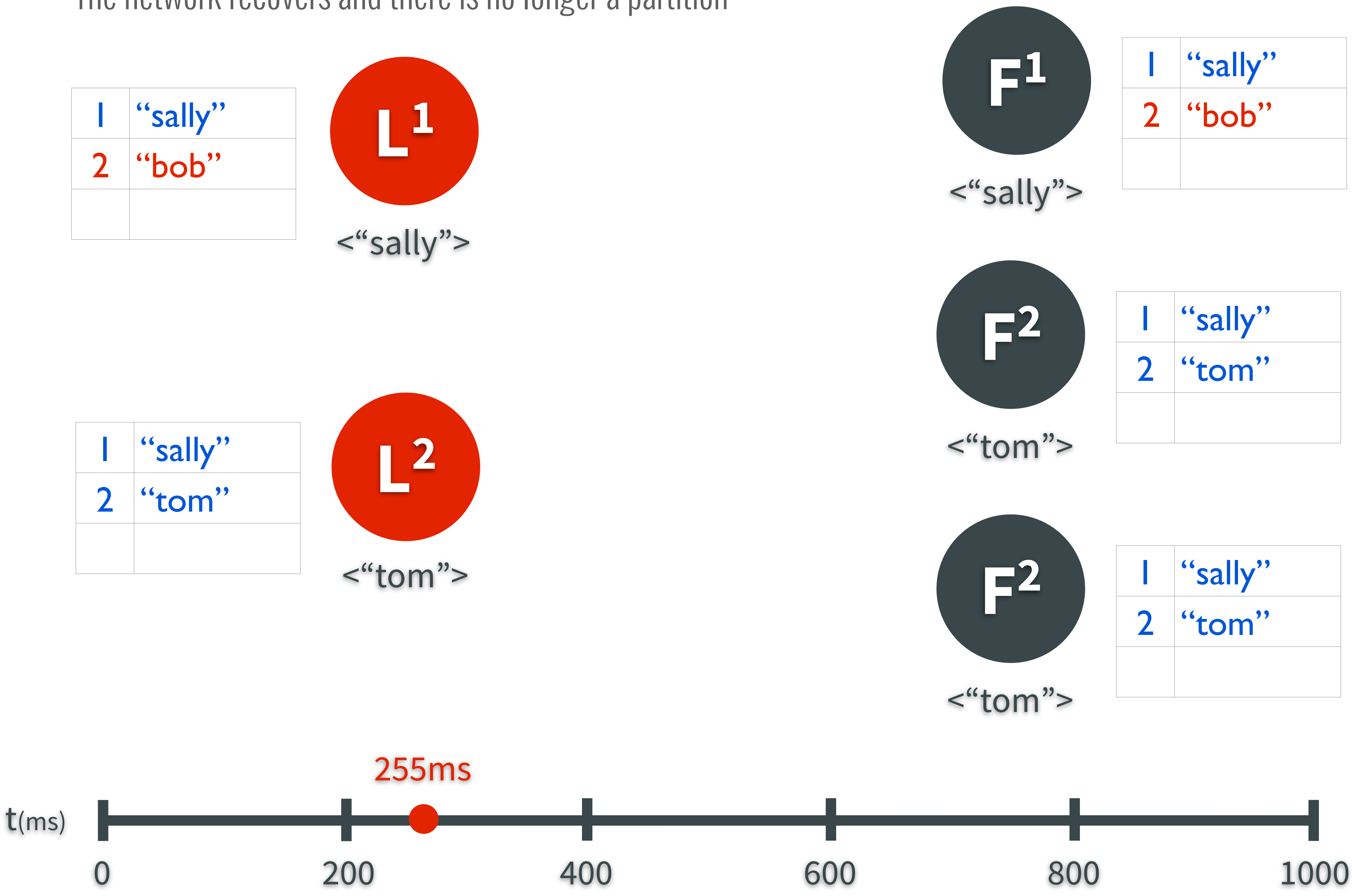


Log Replication



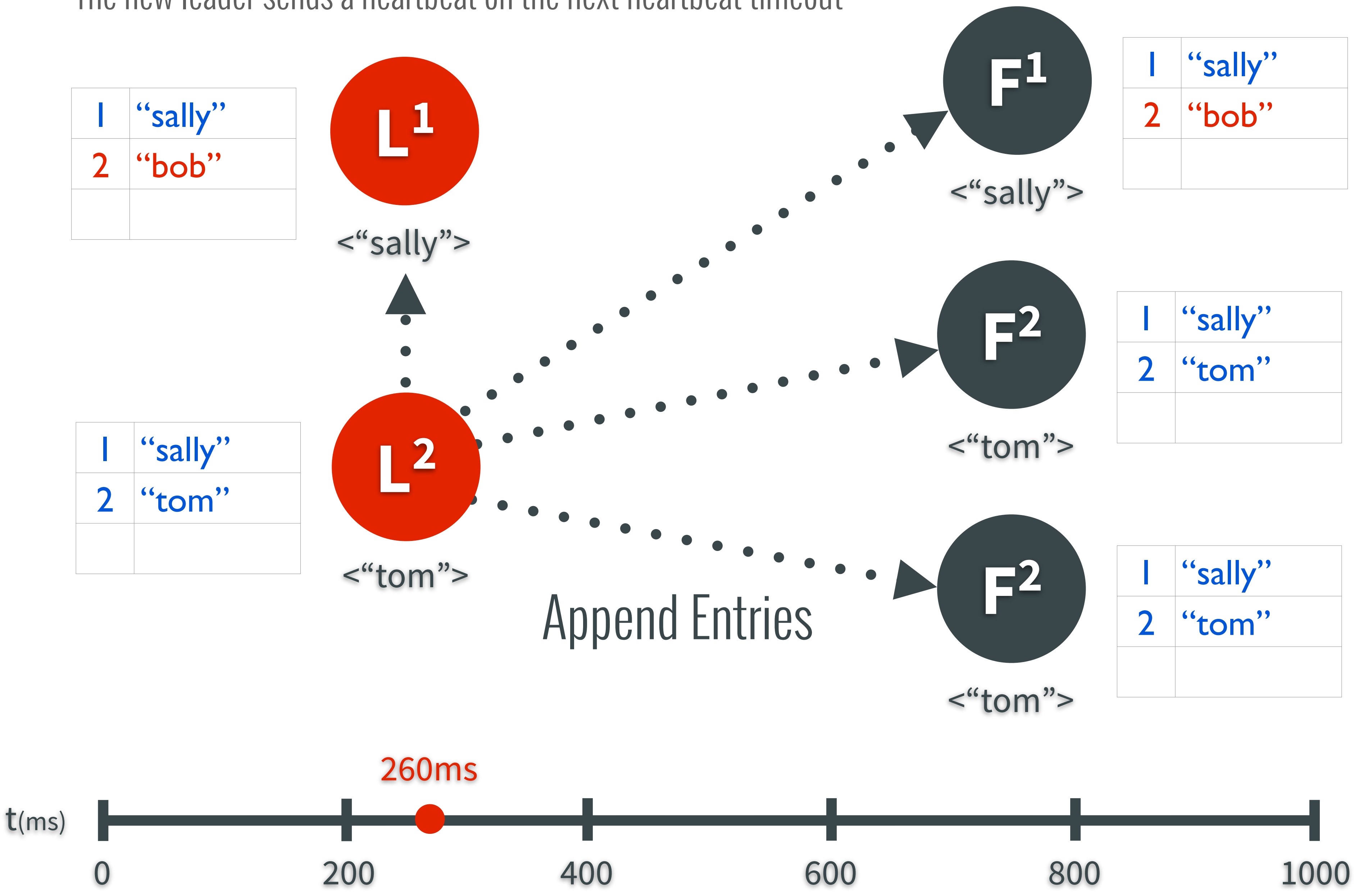
Log Replication

The network recovers and there is no longer a partition



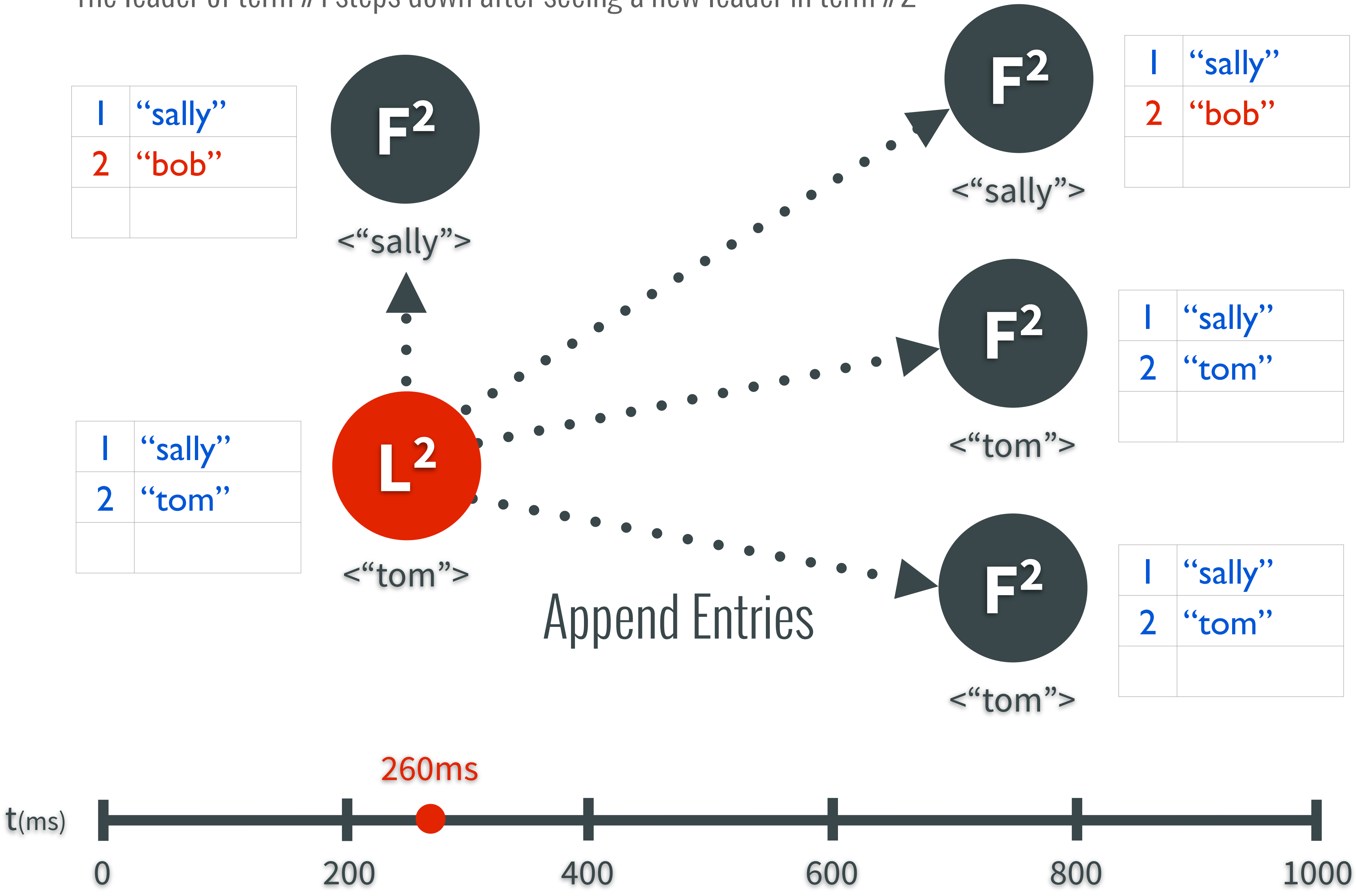
Log Replication

The new leader sends a heartbeat on the next heartbeat timeout



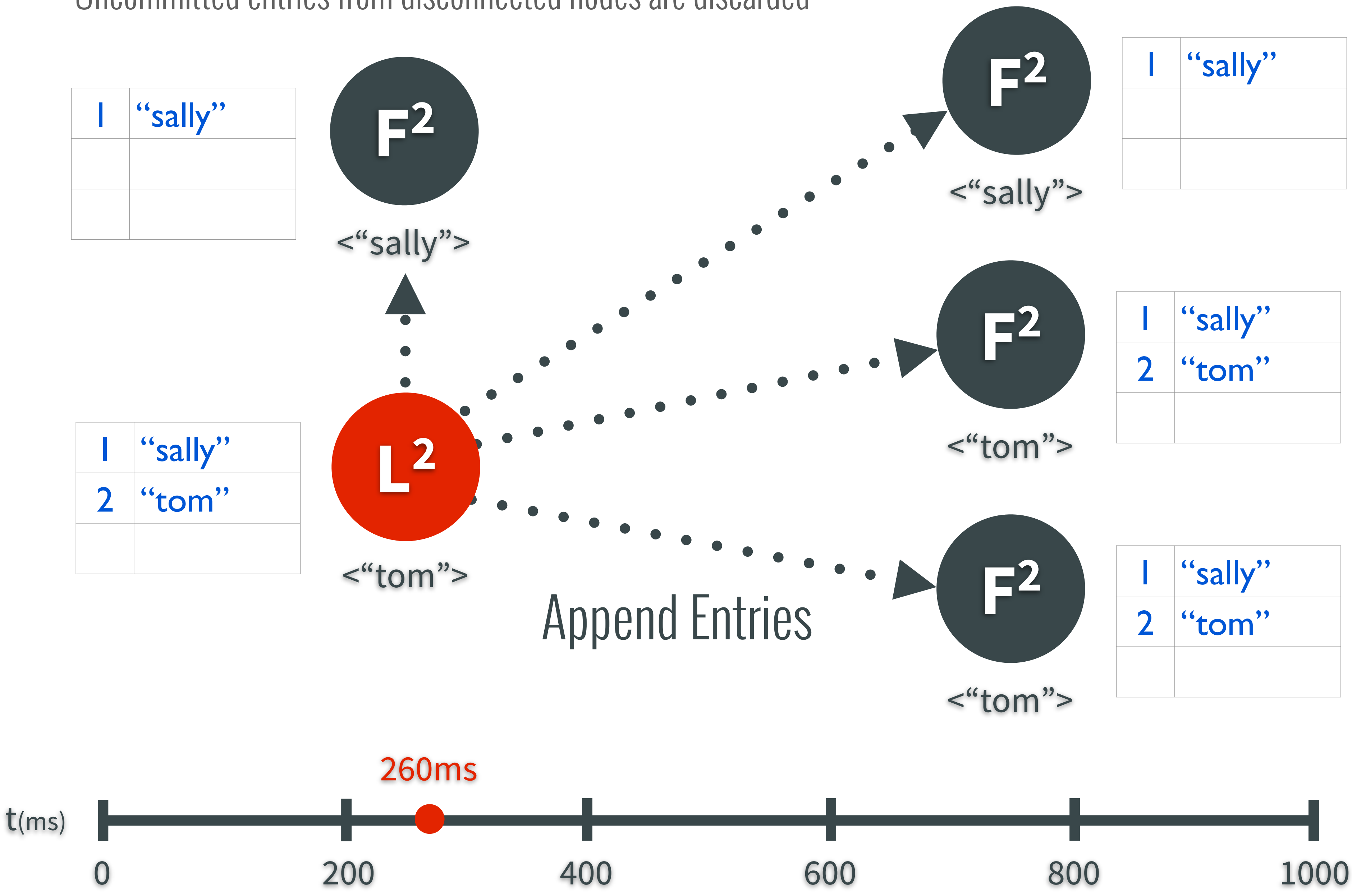
Log Replication

The leader of term #1 steps down after seeing a new leader in term #2



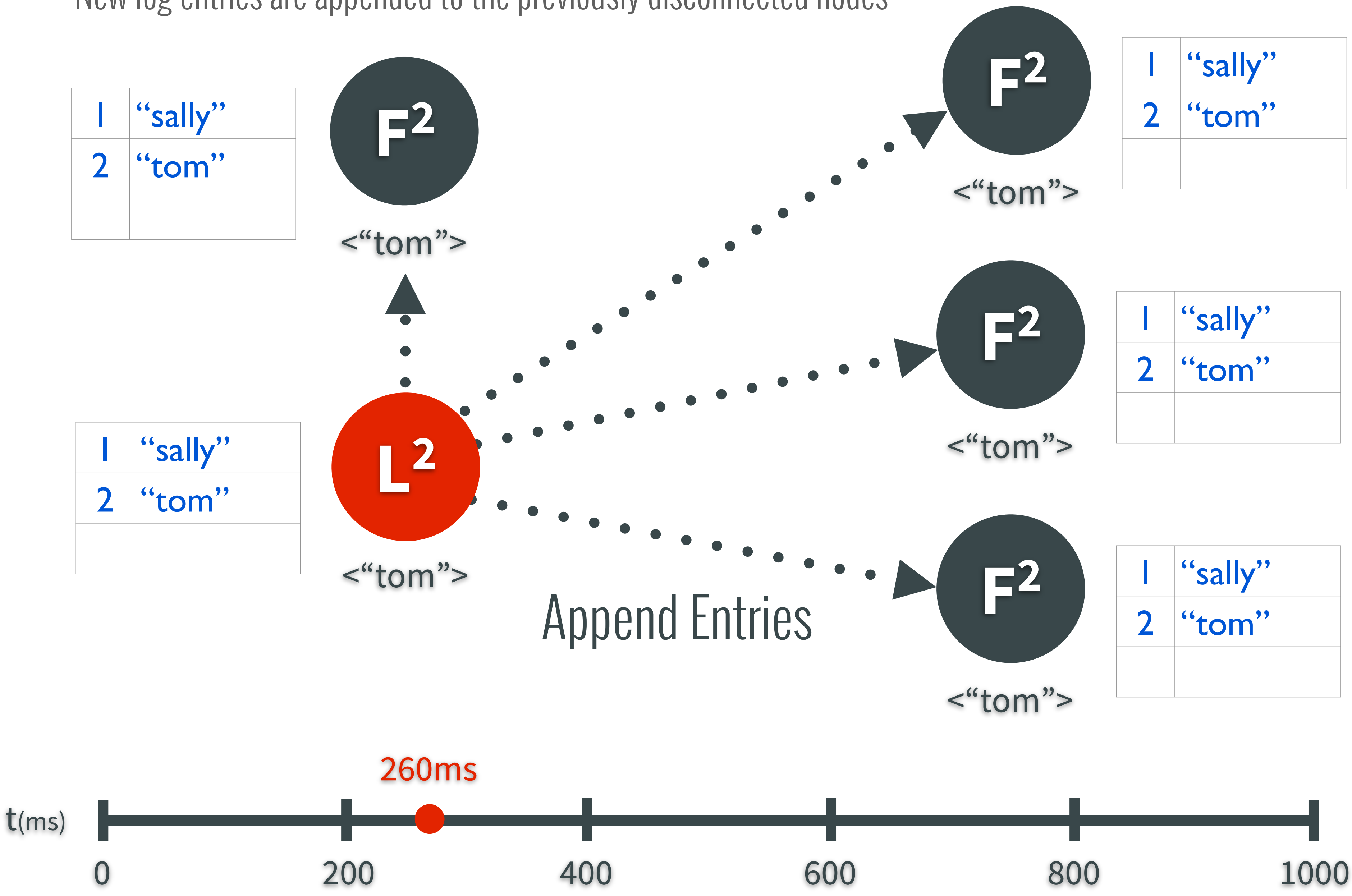
Log Replication

Uncommitted entries from disconnected nodes are discarded

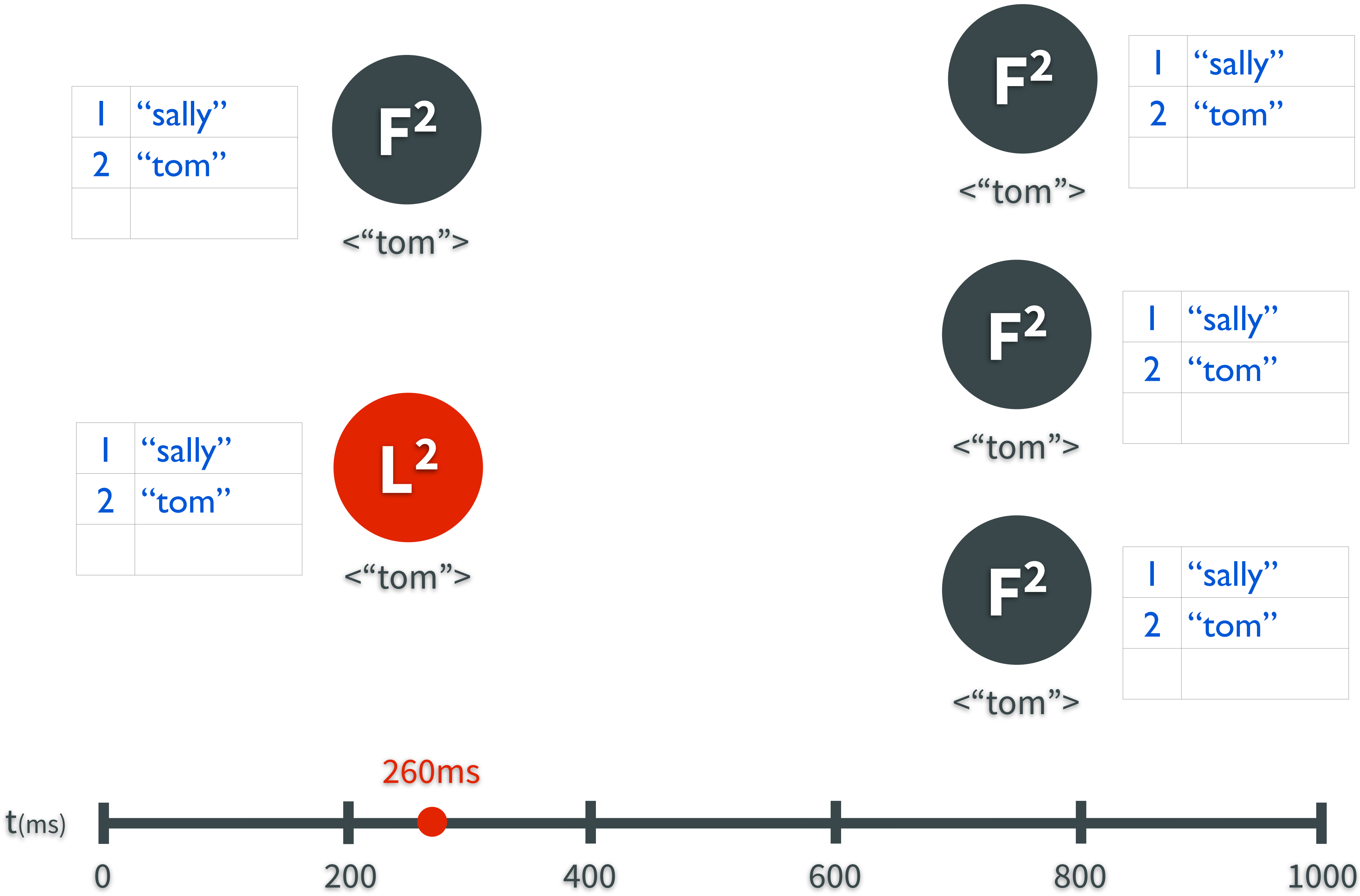


Log Replication

New log entries are appended to the previously disconnected nodes



Log Replication



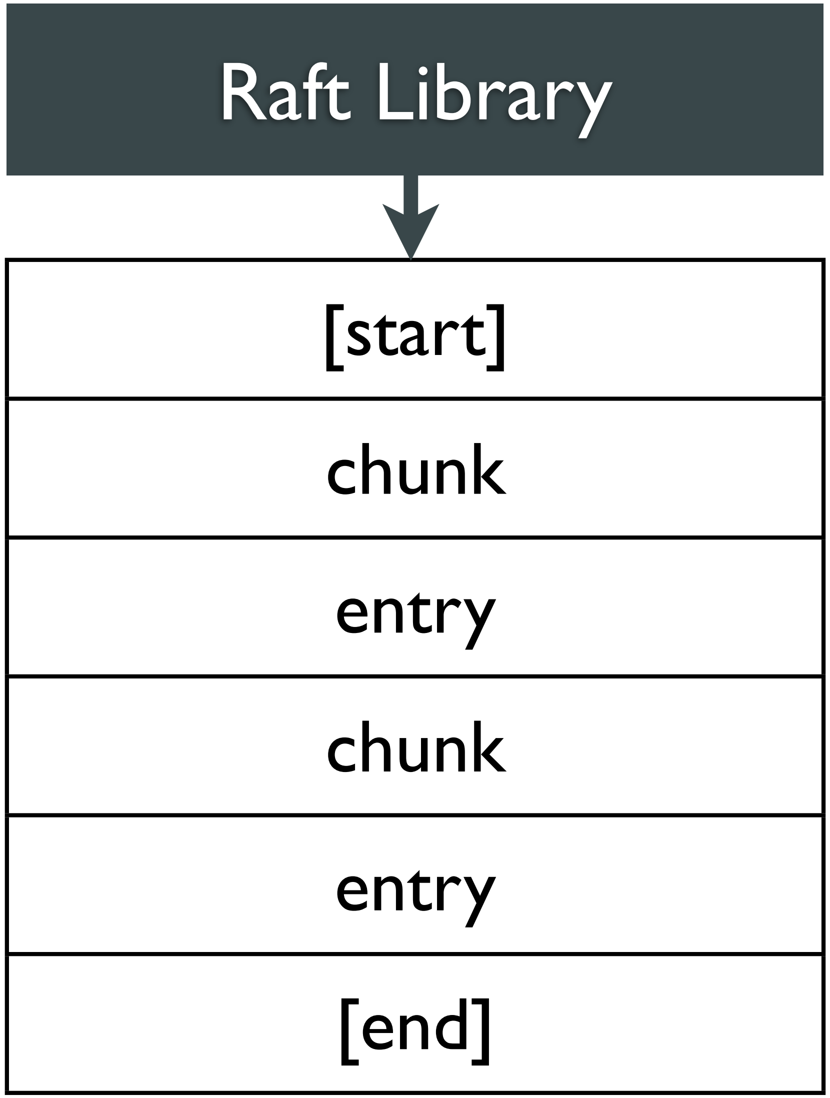
Log Compaction

Unbounded log can grow until there’s no more disk

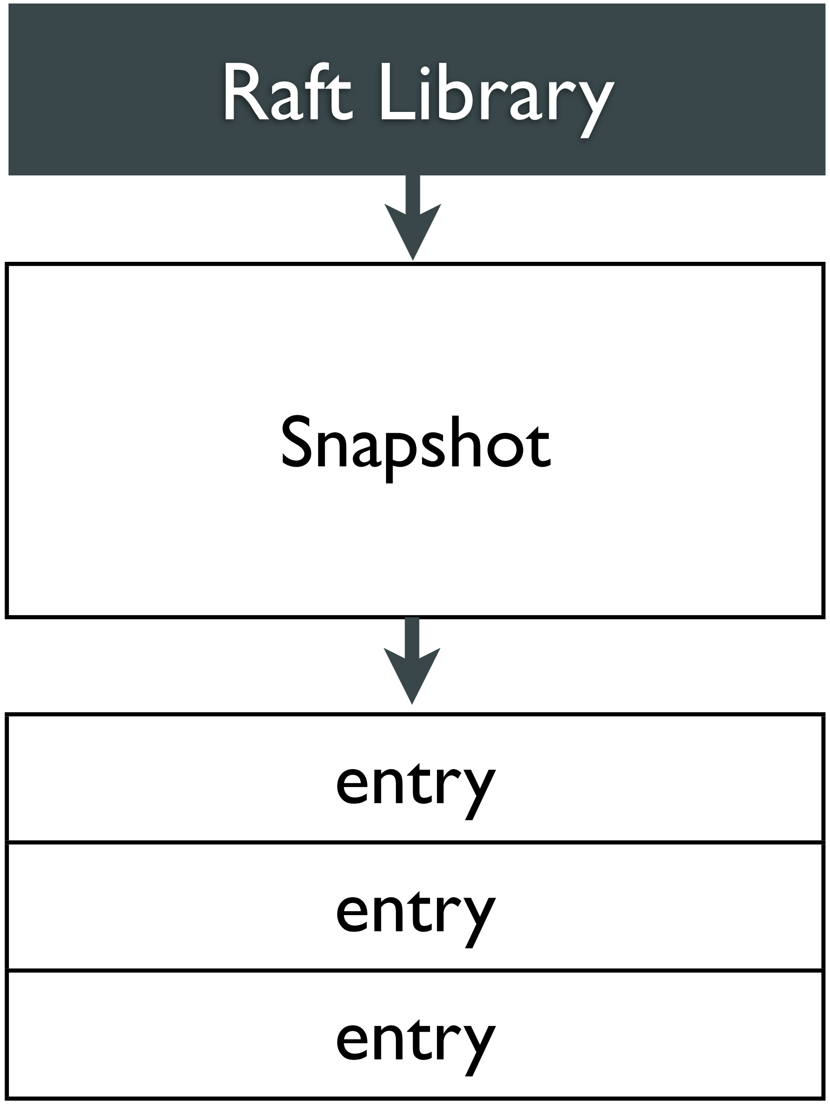
Recovery time increases as log length increases

Three Log Compaction Strategies

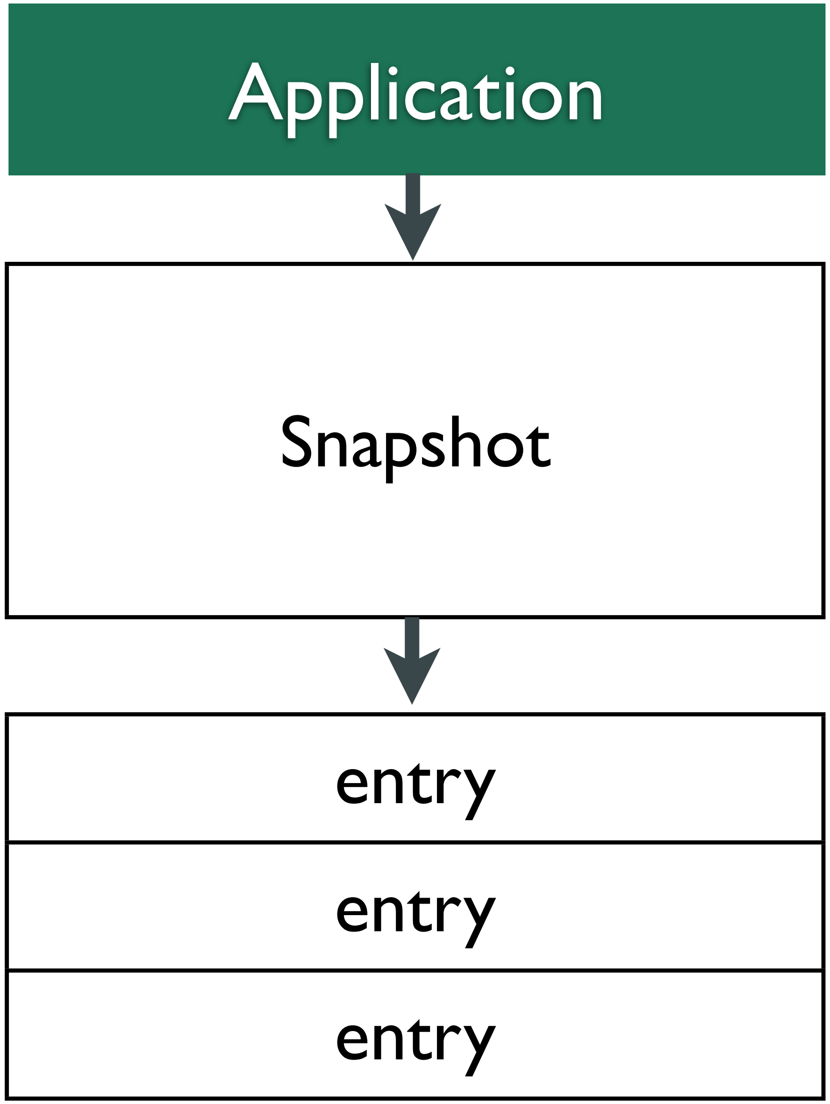
#1: Leader-Initiated, Stored in Log



#2: Leader-Initiated, Stored Externally



#3: Independently-Initiated, Stored Externally



Questions?

Twitter: @benbjohnson
GitHub: benbjohnson
ben@skylandlabs.com

Image Attribution

Database designed by Sergey Shmidt from The Noun Project
Question designed by Greg Pabst from The Noun Project
Lock from The Noun Project
Floppy Disk designed by Mike Wirth from The Noun Project
Movie designed by Anna Weiss from The Noun Project