# etcd

## Distributed environment

- Pros: Performance
- Cons: ACID - hard to comply
  - Atomicity
  - Consistency
  - Isolation
  - Durability

## etcd = /etc distributed

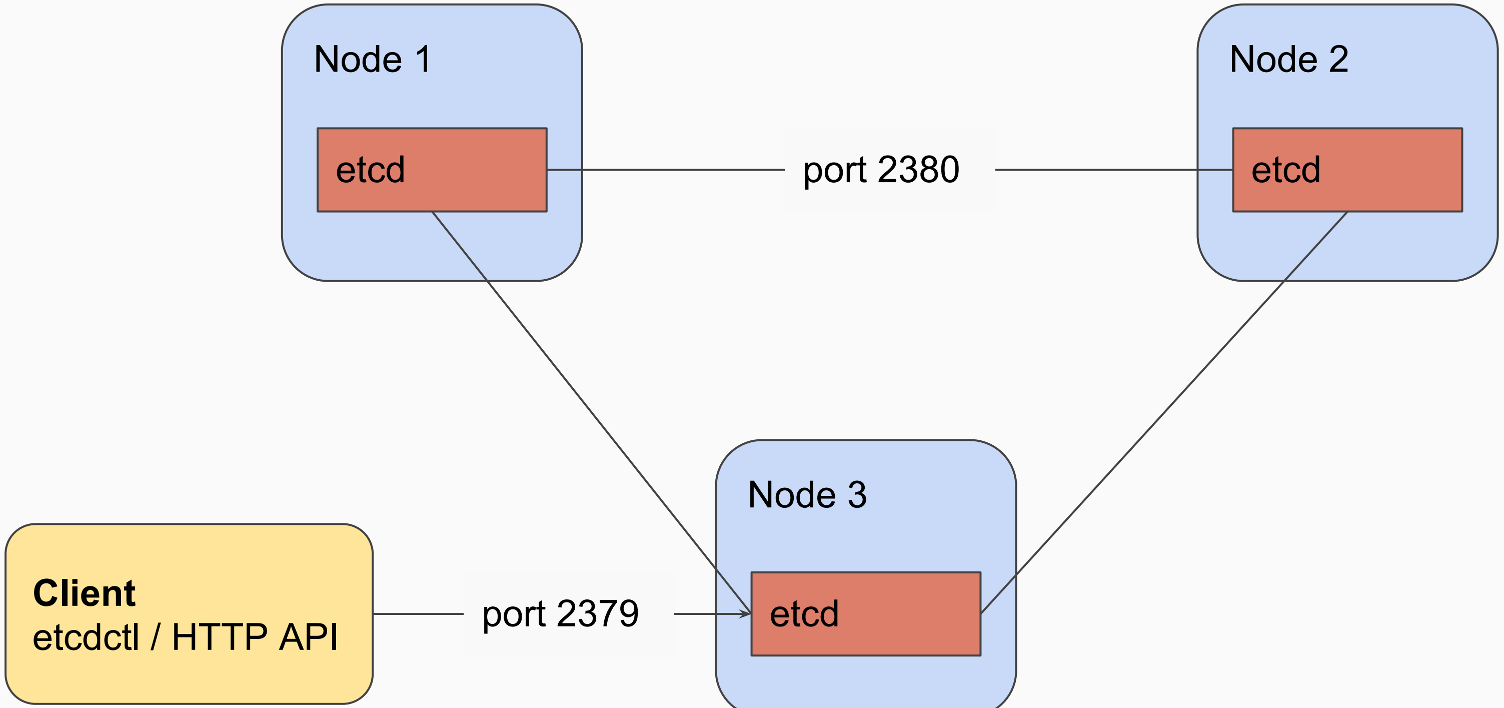
- Key-Value storage
- Consistency
- High Availability
- Failure tolerant
- Cluster Configuration

- /config
  - /database
- /feature-flags
  - /verbose-logging
  - /redesign

## etcd

- open source developed by CoreOS
- written in Go
- durable
- watchable
- exposed via HTTP
- runtime reconfigurable

## Cluster Architecture



## Basic Features - SET

Command line interface - etcdctl

```
$ etcdctl set /nosql/foo bar
bar
```

HTTP API

```
$ curl -L -X PUT http://localhost:2379/v2/keys/nosql/foo -
d value="bar"

{"action":"set","node":{"key":"/nosql/foo","value":"bar","
modifiedIndex":23995,"createdIndex":23995}}
```

## Basic Features - LIST

Command line interface - etcdctl

```
$ etcdctl ls /nosql
/nosql/foo
```

HTTP API

```
$ curl -L http://localhost:2379/v2/keys/nosql
```

{"action":"get","node":{"key":"/nosql","dir":true,"nodes":[{"key":"/nosql/foo","value":"bar","modifiedIndex":23931,"createdIndex":23931}],"modifiedIndex":282,"createdIndex":282}}

## Basic Features - GET

Command line interface - etcdctl

```
$ etcdctl get /nosql/foo
bar
```

HTTP API

```
$ curl -L http://localhost:2379/v2/keys/nosql/foo
```

{"action":"get","node":{"key":"/nosql/foo","value":"bar","
modifiedIndex":23931,"createdIndex":23931}}

## Basic Features - WATCH

Command line interface - etcdctl

```
$ etcdctl watch --recursive /web-service/backends
...
```

HTTP API

```
$ curl -L http://localhost:2379/v2/keys/web-service/backends
?wait=true&recursive=true


...
```

## Atomic Compare and Swap

Command line interface - etcdctl

```
$ etcdctl set --swap-with-value 'two' /foo three
```

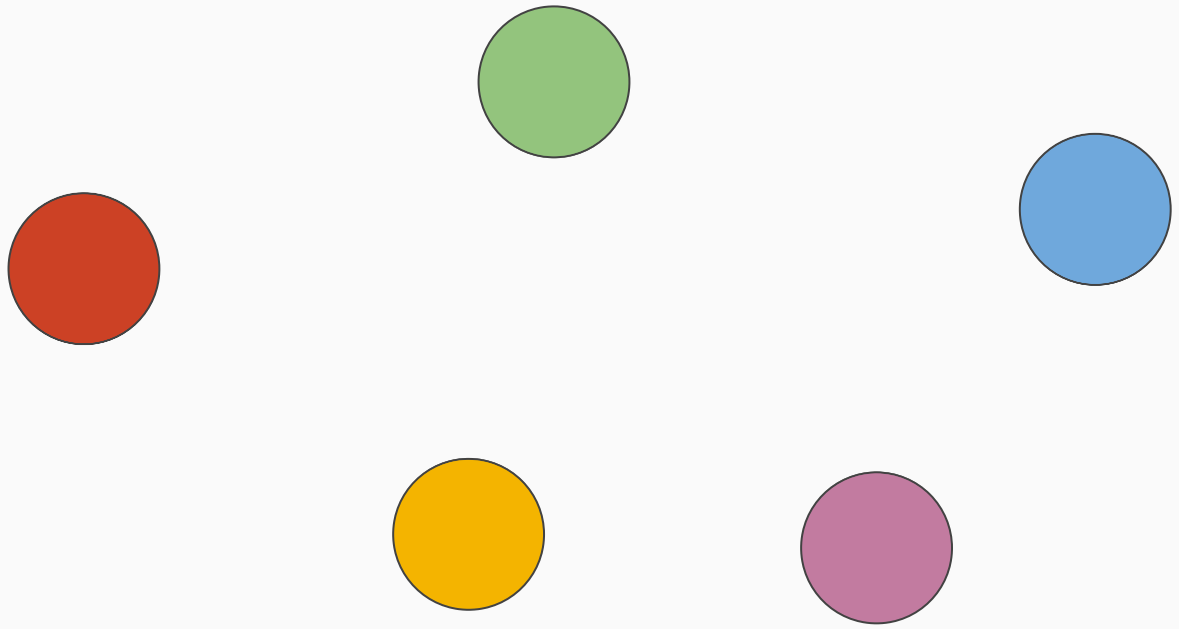
```
Error:  101: Compare failed ([two != one]) [31627]
```

HTTP API

```
$ curl http://localhost:2379/v2/keys/foo?prevValue=two -XPUT
-d value=three
```

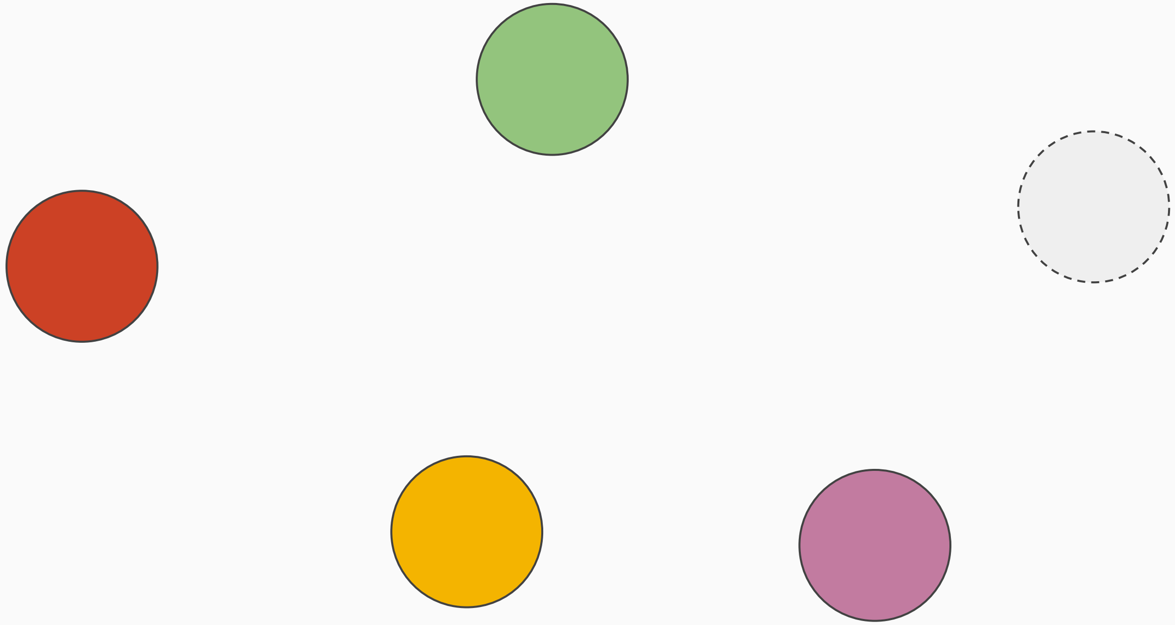
```
{"errorCode":101,"message":"Compare failed","cause":"[two !=
one]","index":31642}
```

## Cluster Availability



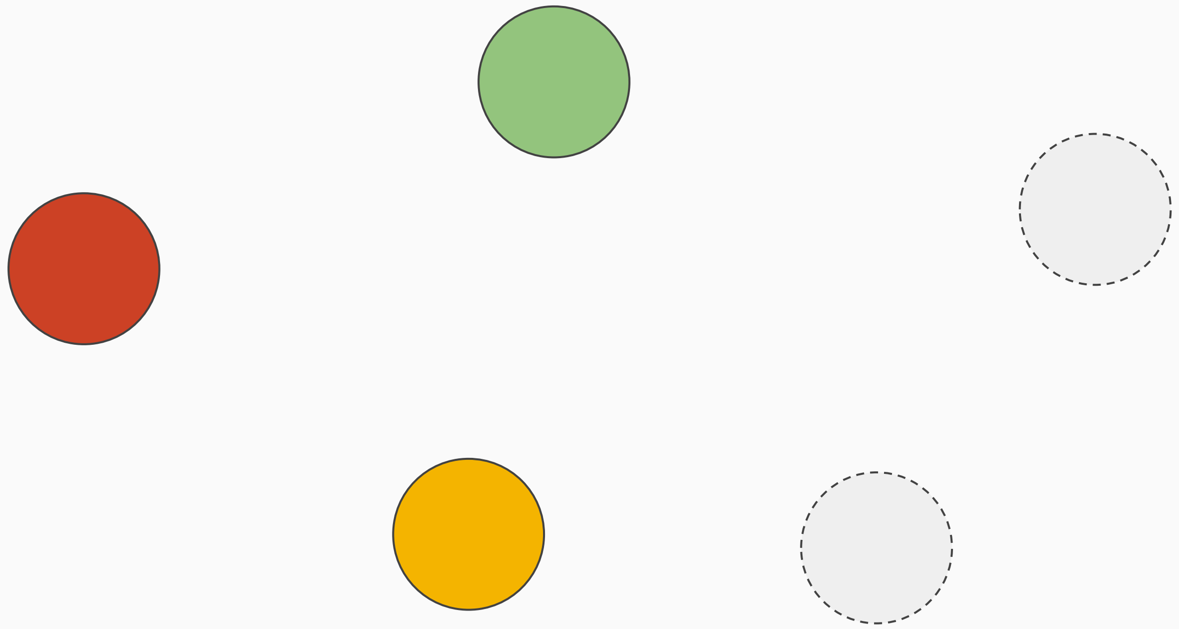
Available

## Cluster Availability



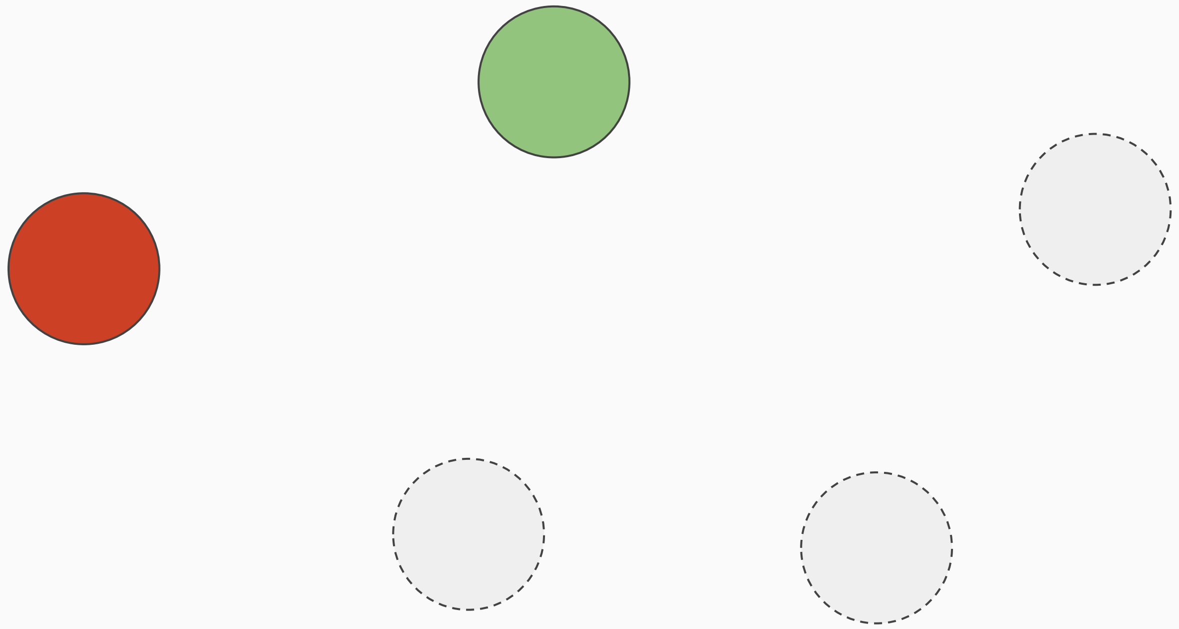
Available

## Cluster Availability



Available

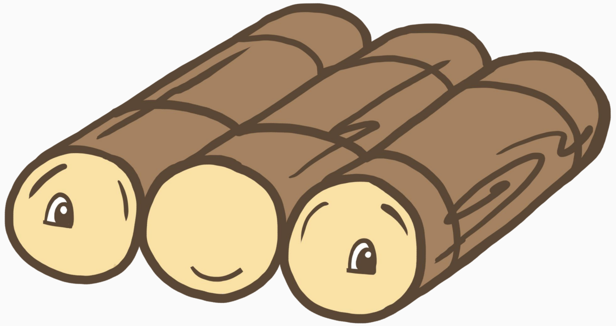
## Cluster Availability



Unavailable

## Raft

The understandable **distributed consensus** protocol



Distributed = “a lot” of nodes
Consensus = Agreement

## Raft

Data replication

Leader election

Distributed Locks

## Three roles:

## RAFT

The Leader

RAFT

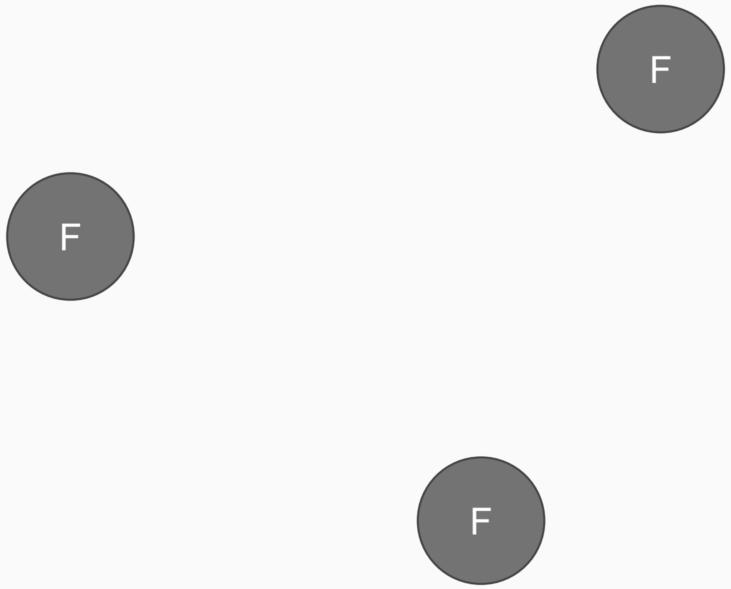
The Follower

RAFT

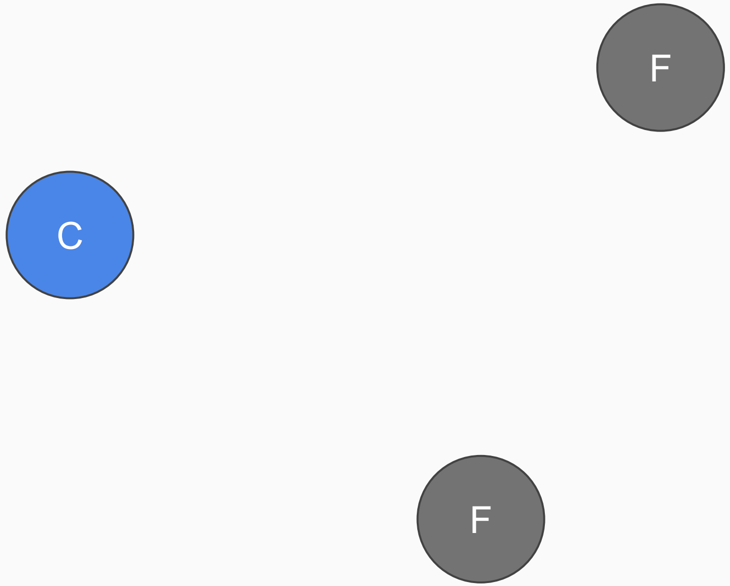
The Candidate

## High level example:
## Leader Election

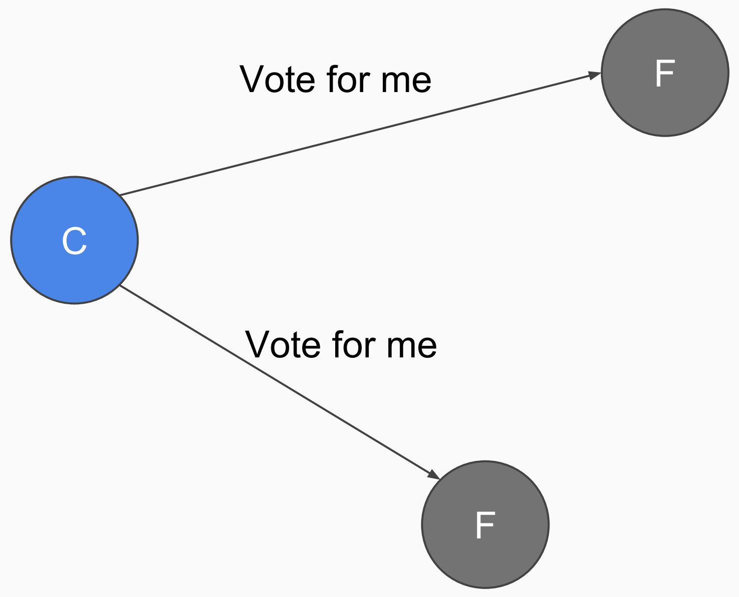
## RAFT - Leader election



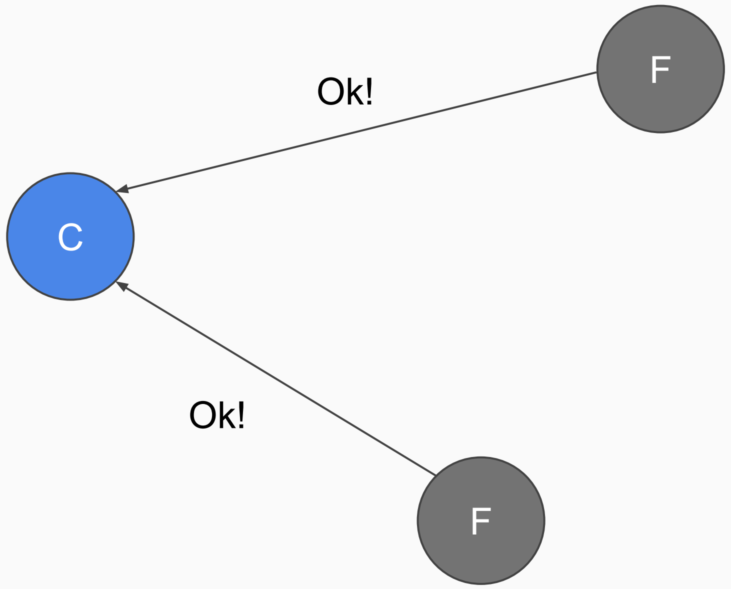
# RAFT - Leader election



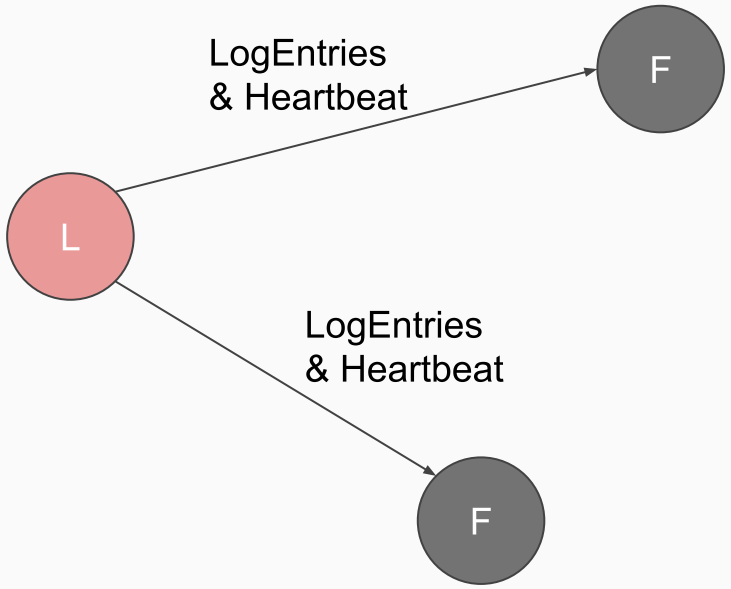
## RAFT - Leader election



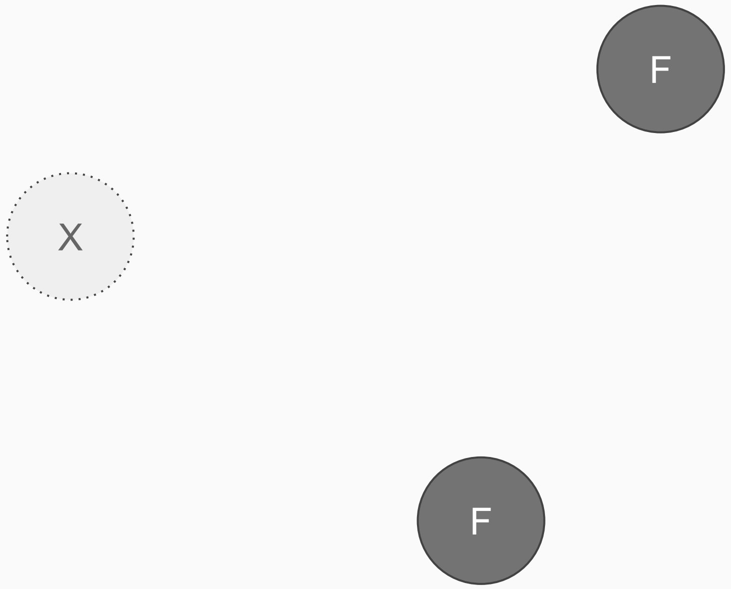
## RAFT - Leader election



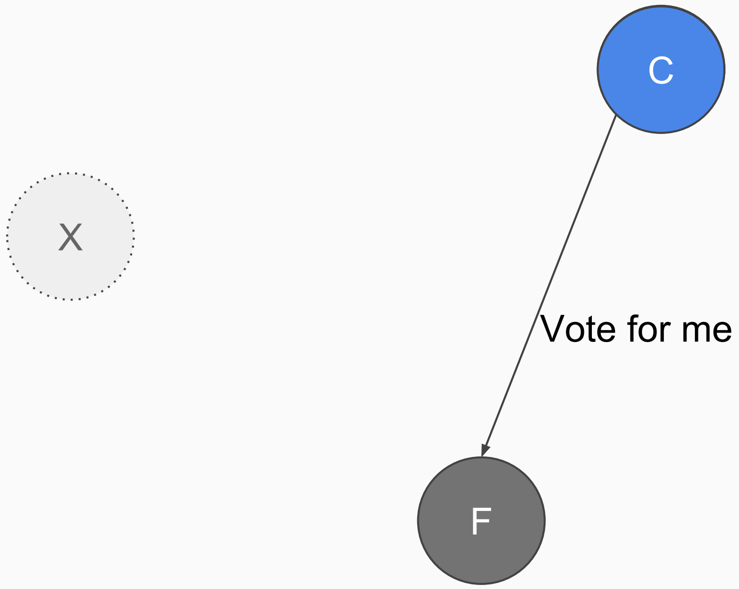
## RAFT - Leader election



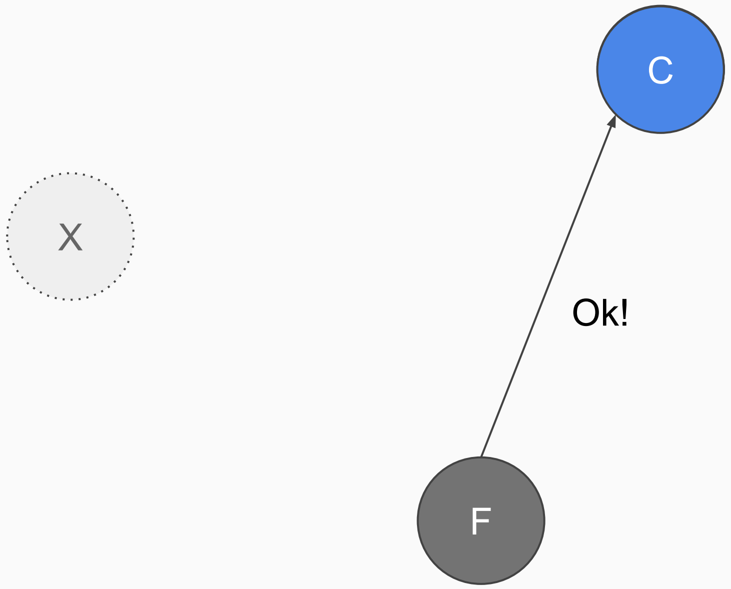
# RAFT - Leader election



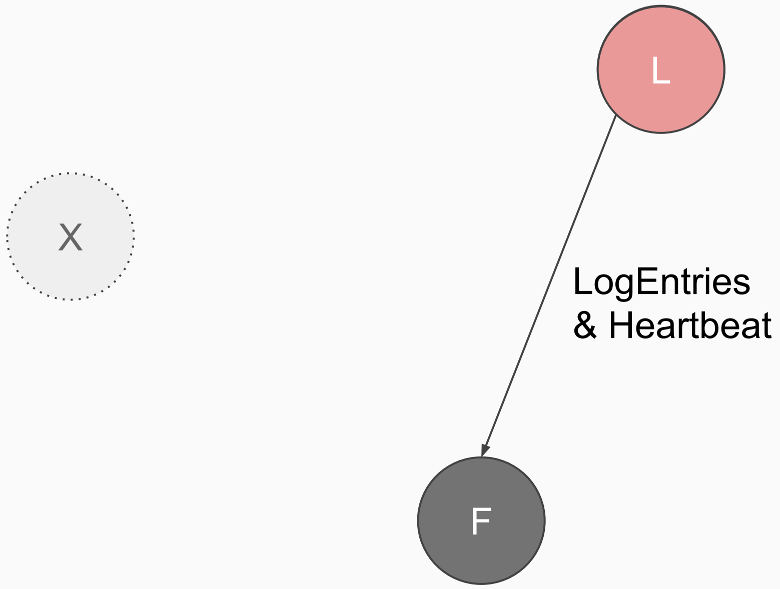
## RAFT - Leader election



# RAFT - Leader election



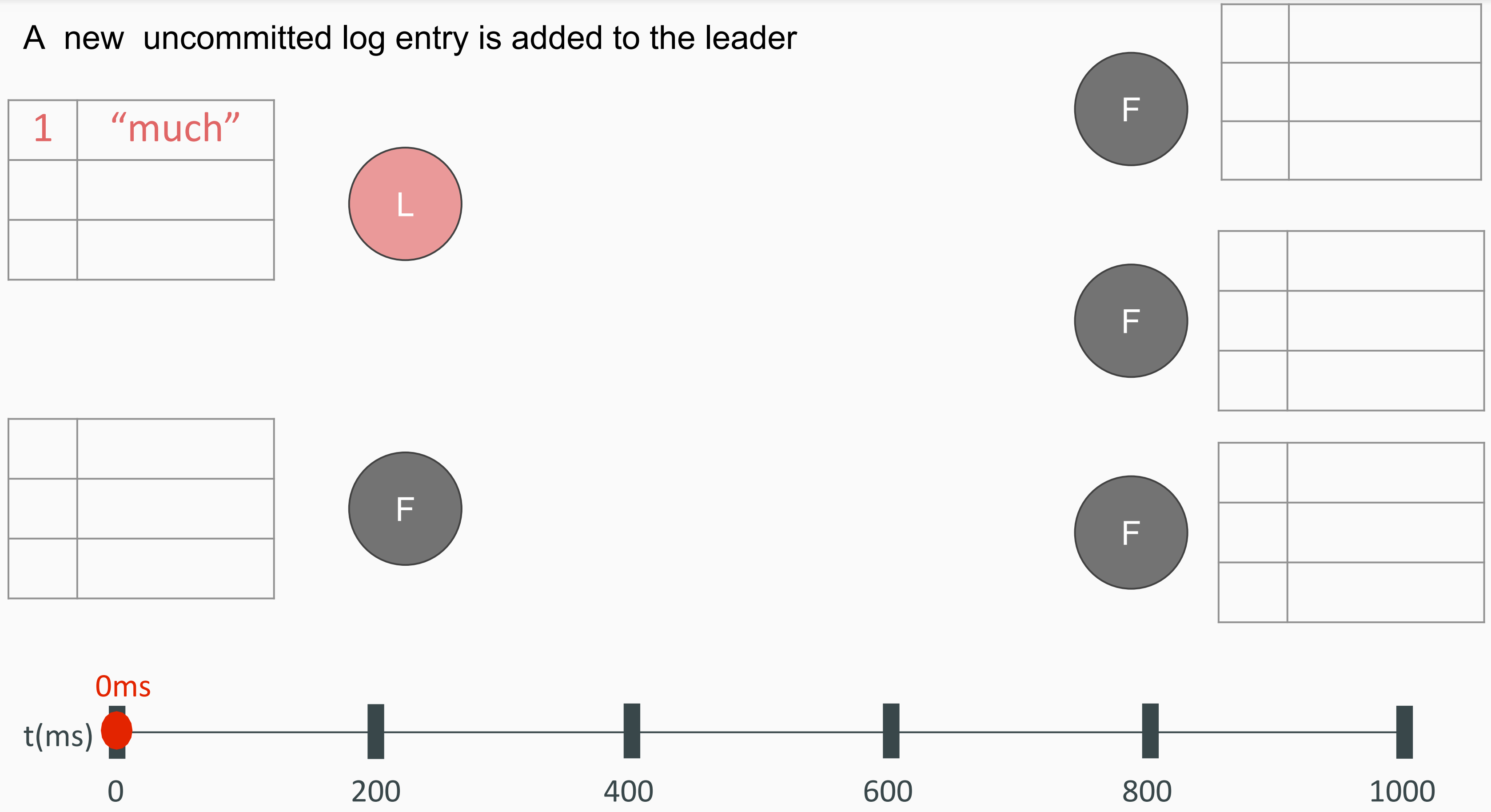
# RAFT - Leader election



## High level example:
## Log Replication
(with network partitions)

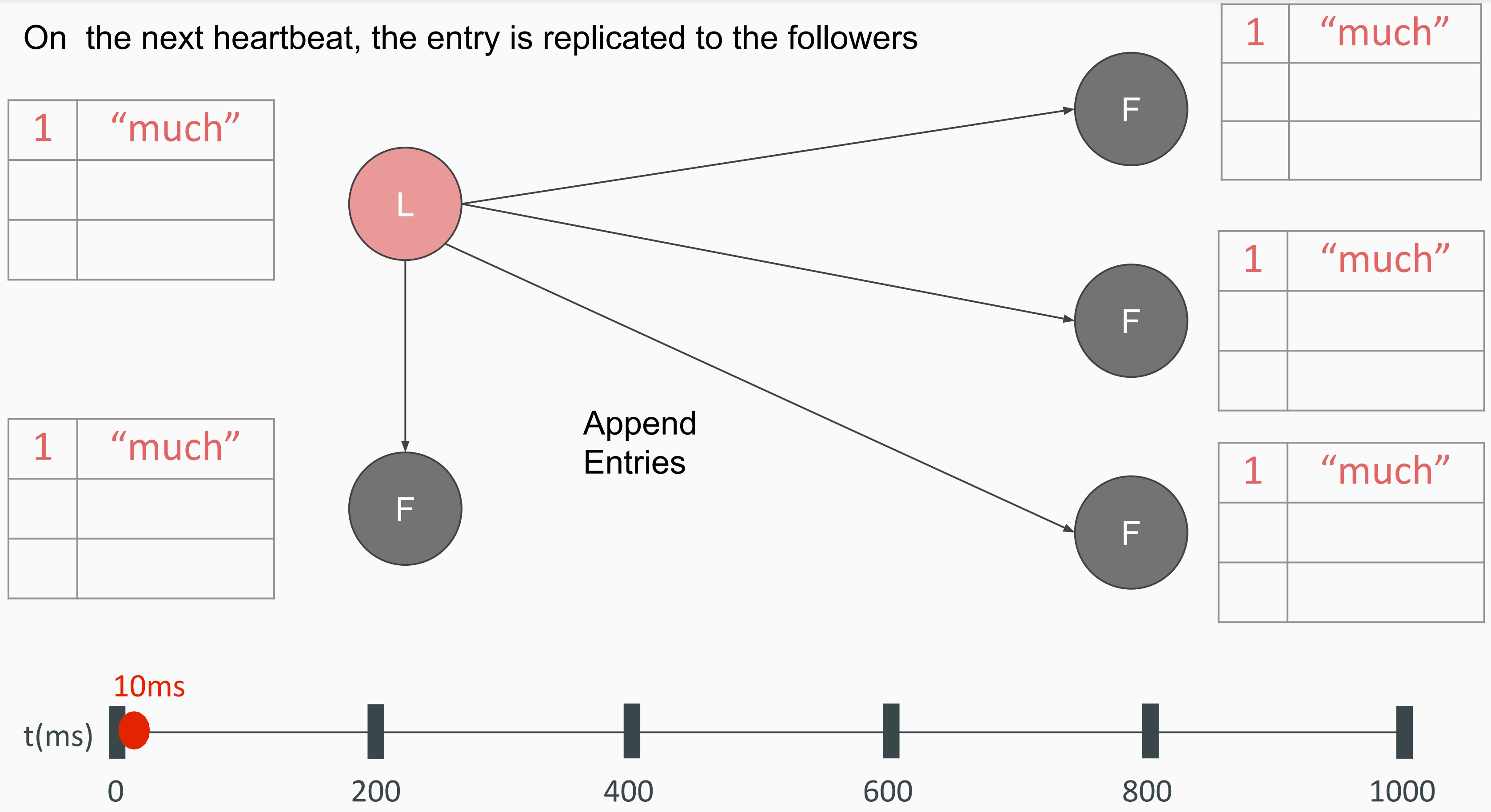
# RAFT - Log Replication

A new uncommitted log entry is added to the leader



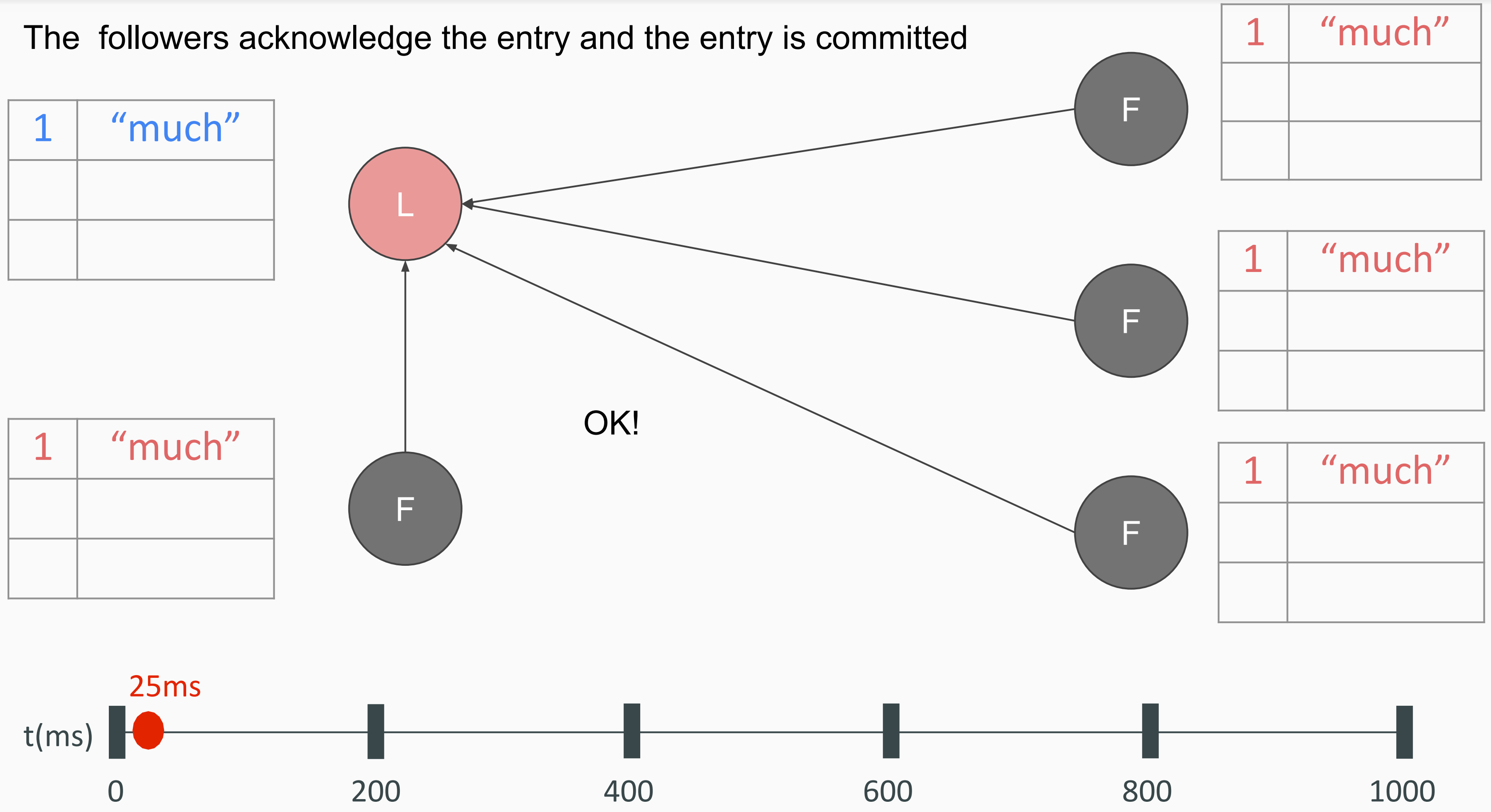
# RAFT - Log Replication

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



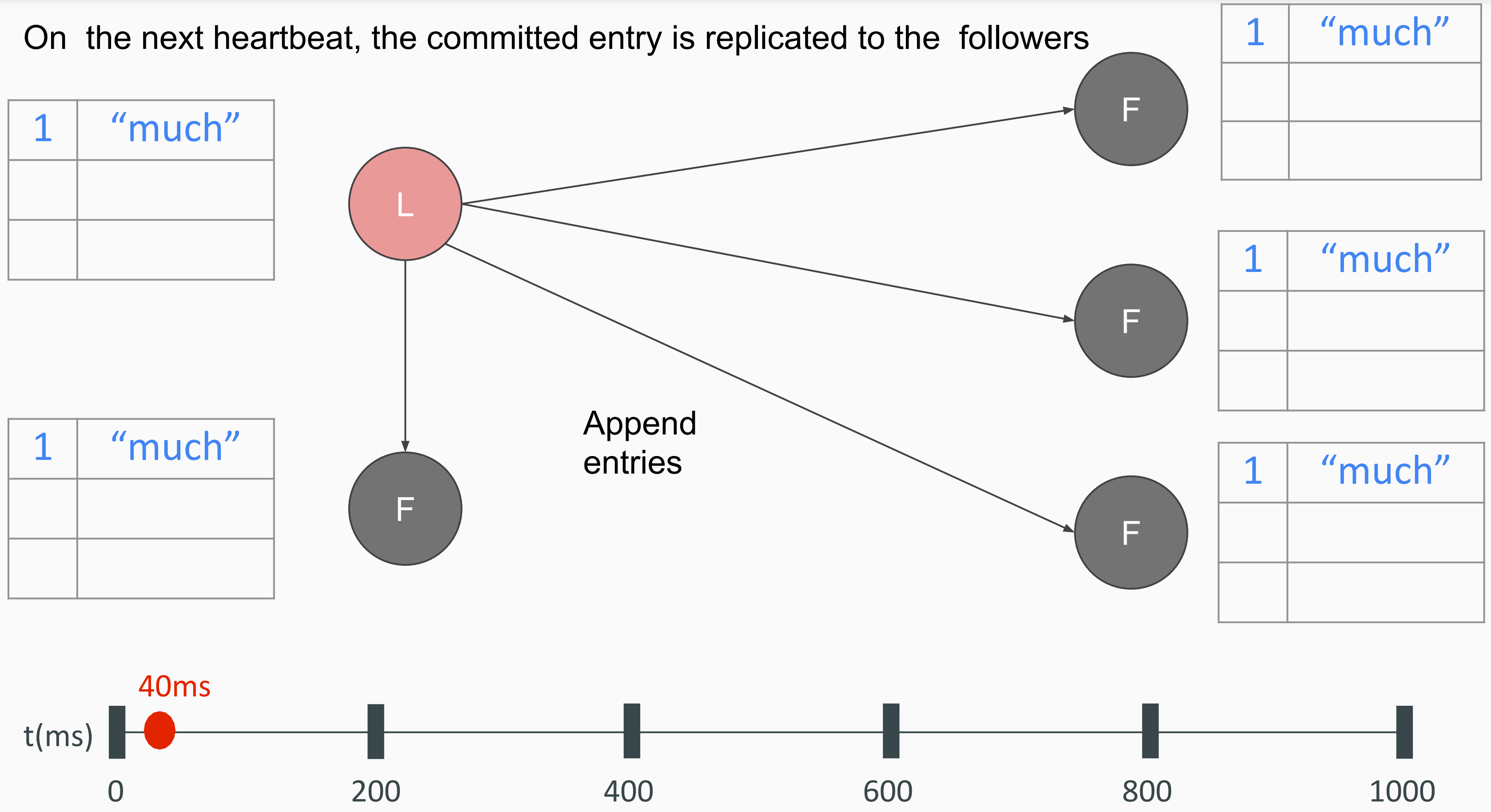
# RAFT - Log Replication

The  followers acknowledge the entry and the entry is committed



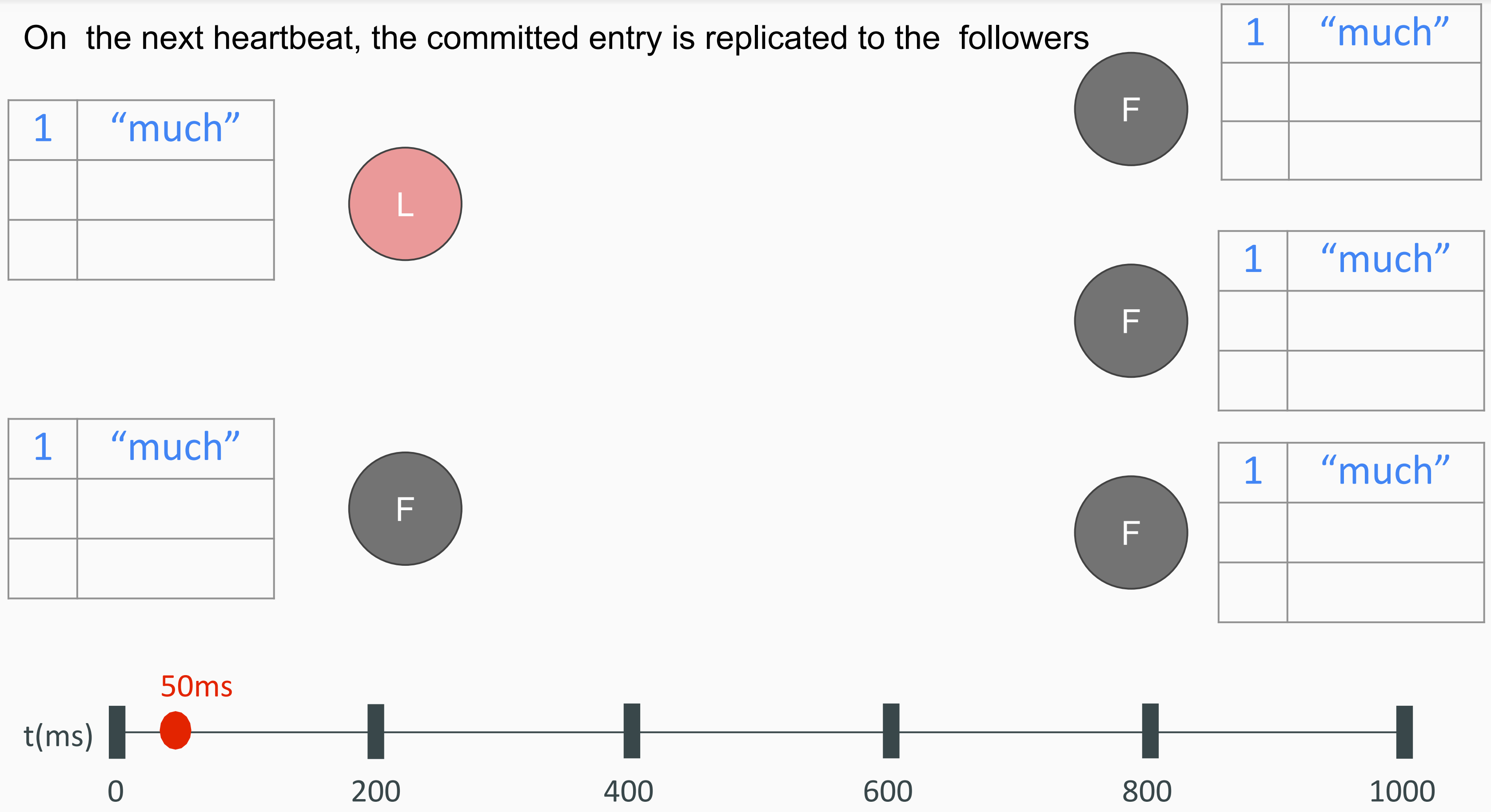
## RAFT - Log Replication

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



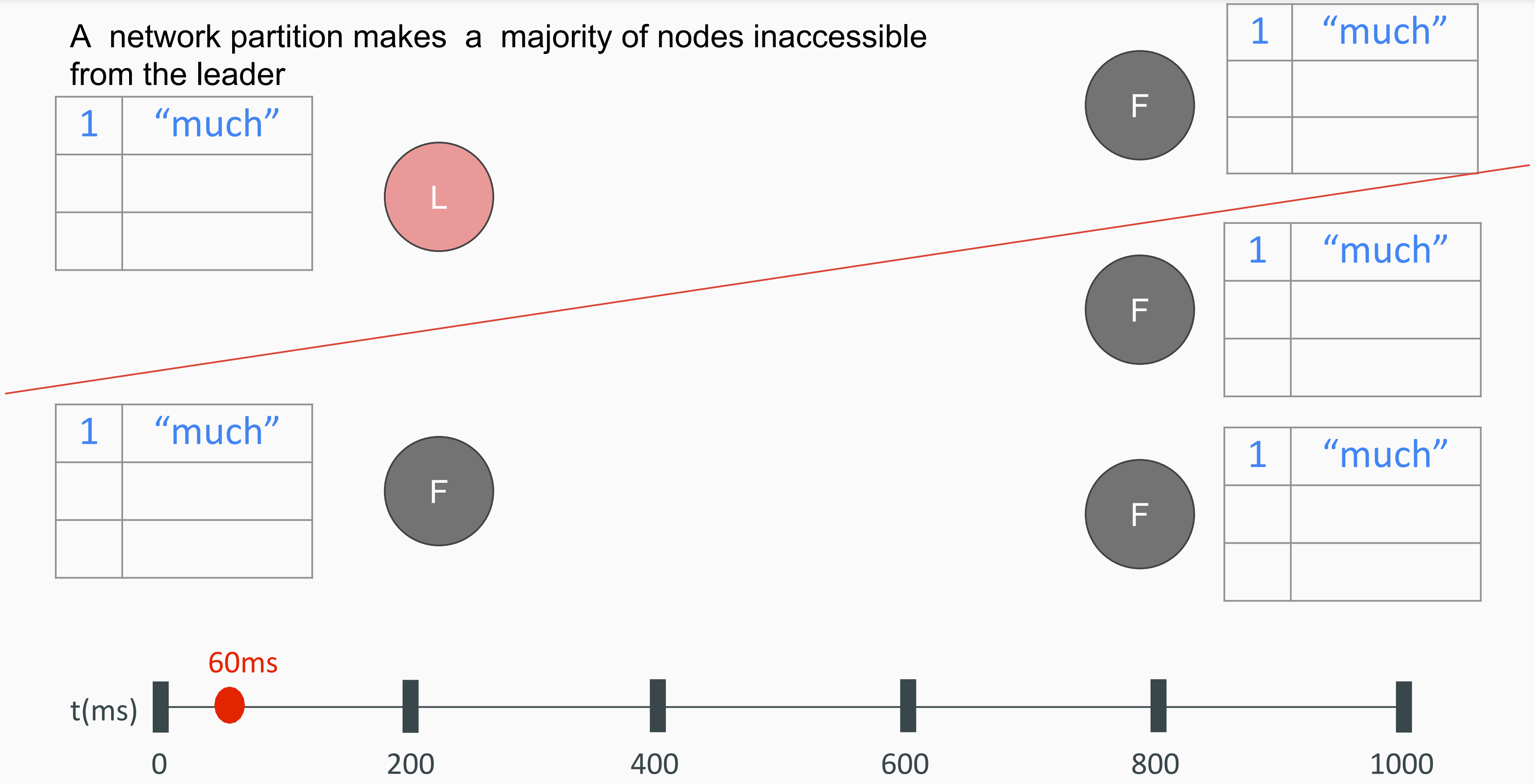
# RAFT - Log Replication

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



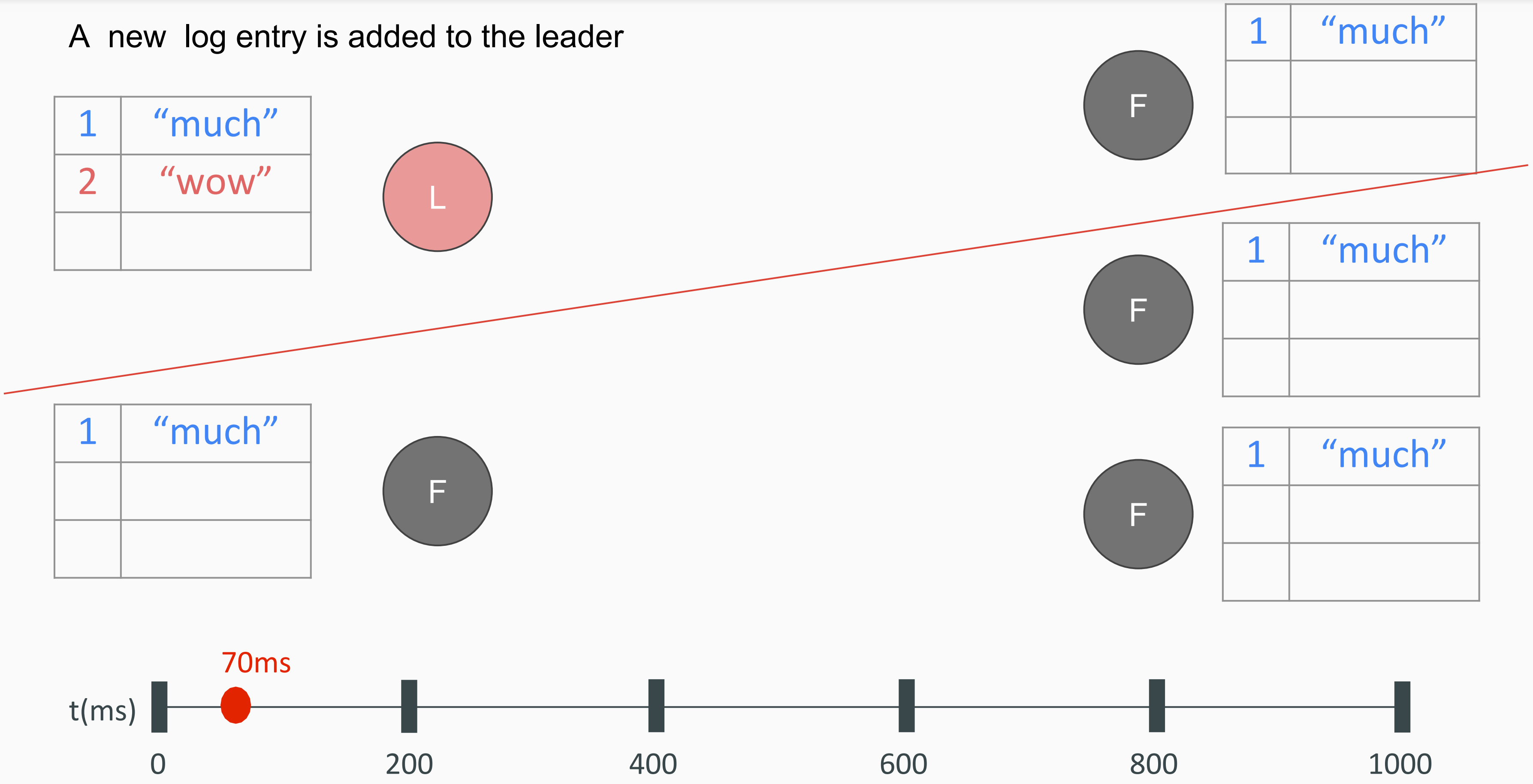
# RAFT - Log Replication

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



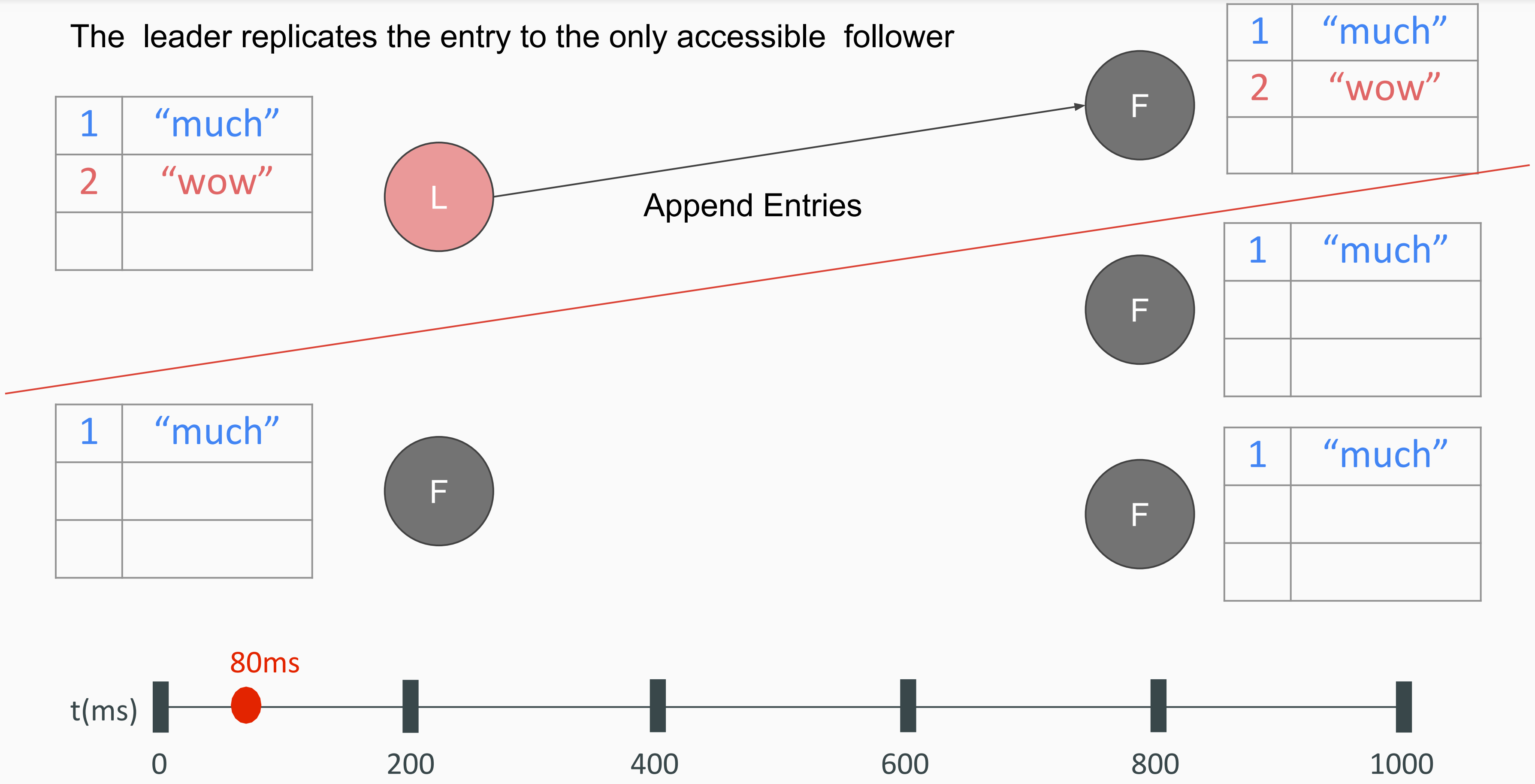
# RAFT - Log Replication

A new log entry is added to the leader



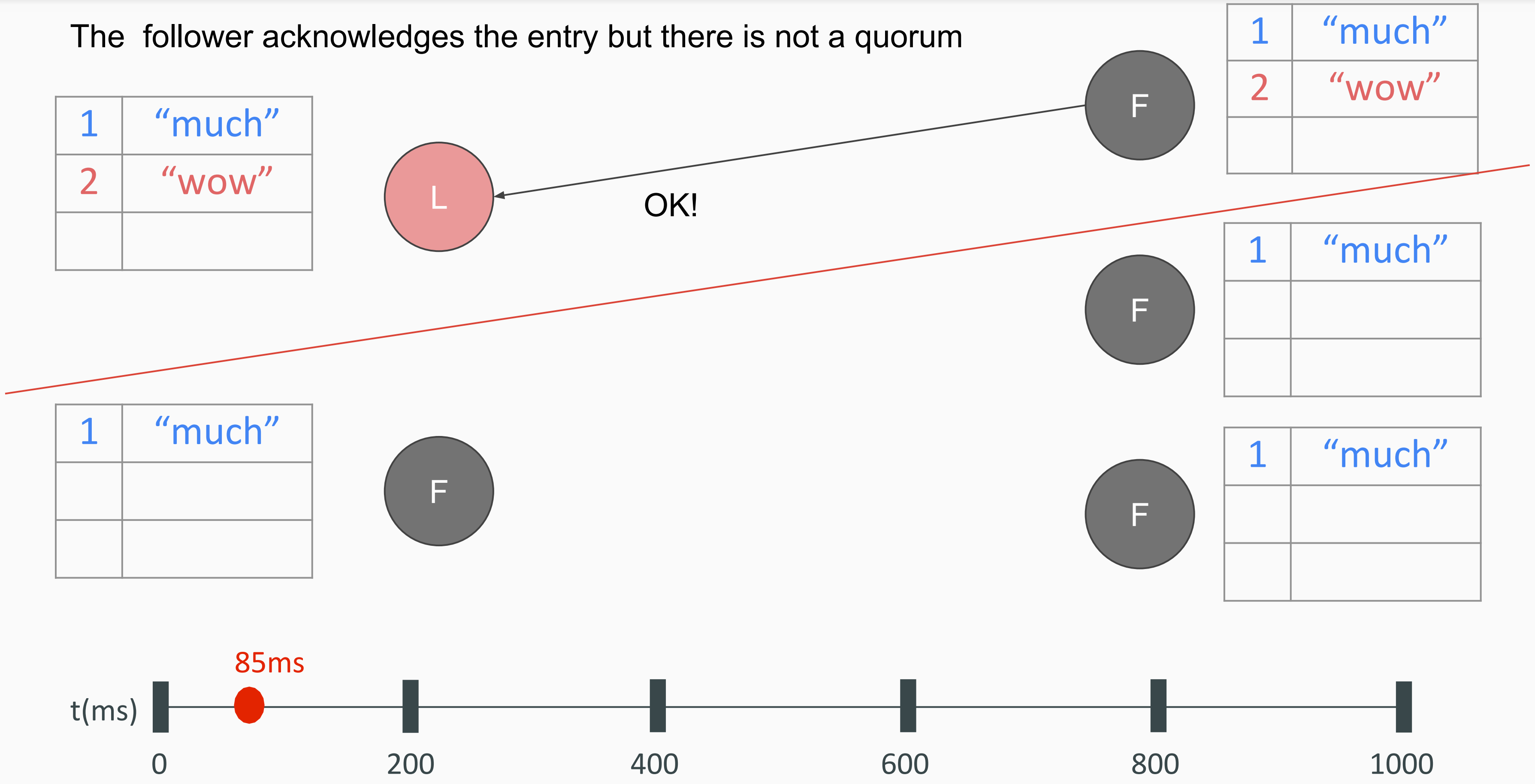
# RAFT - Log Replication

The leader replicates the entry to the only accessible follower

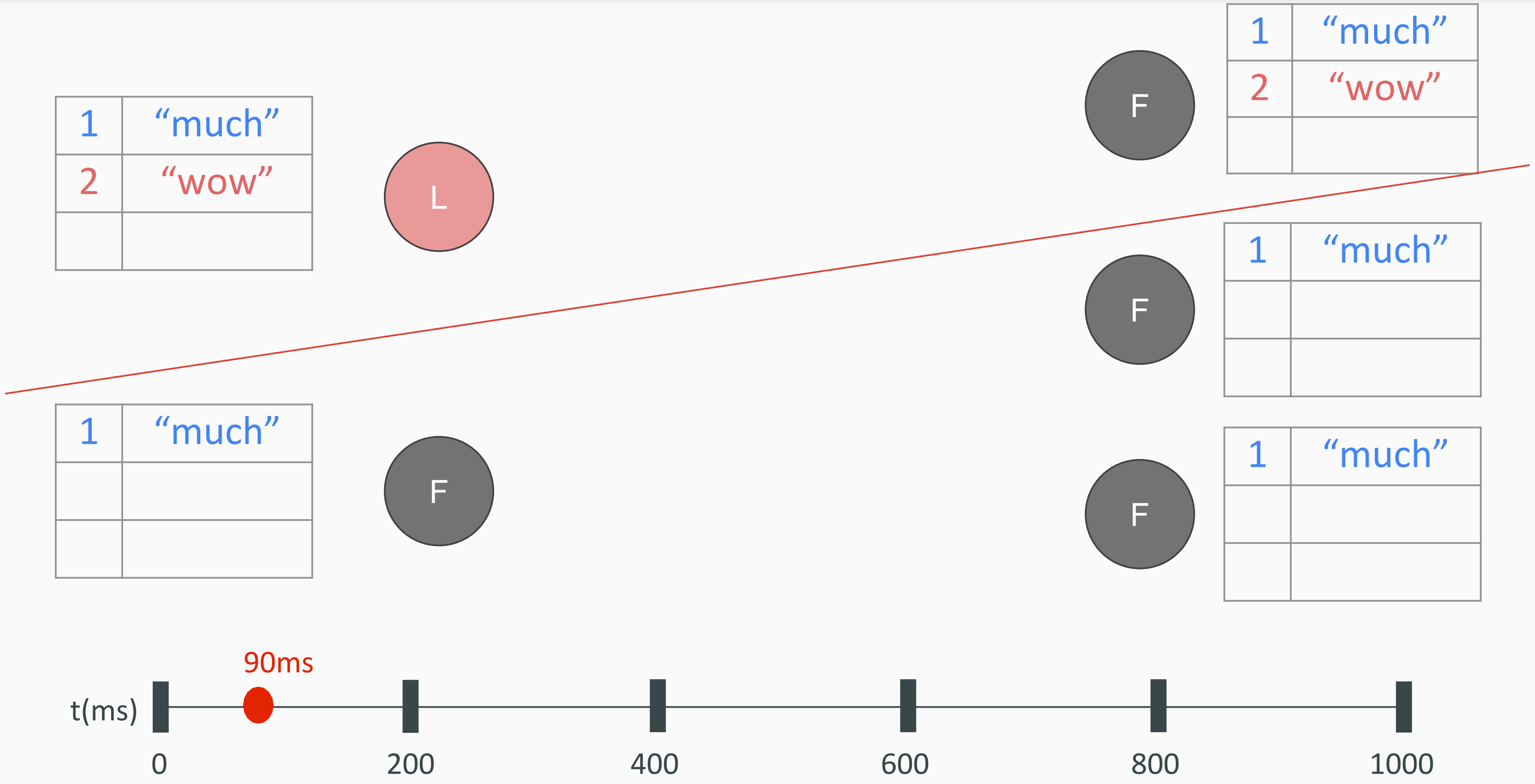


# RAFT - Log Replication

The follower acknowledges the entry but there is not a quorum

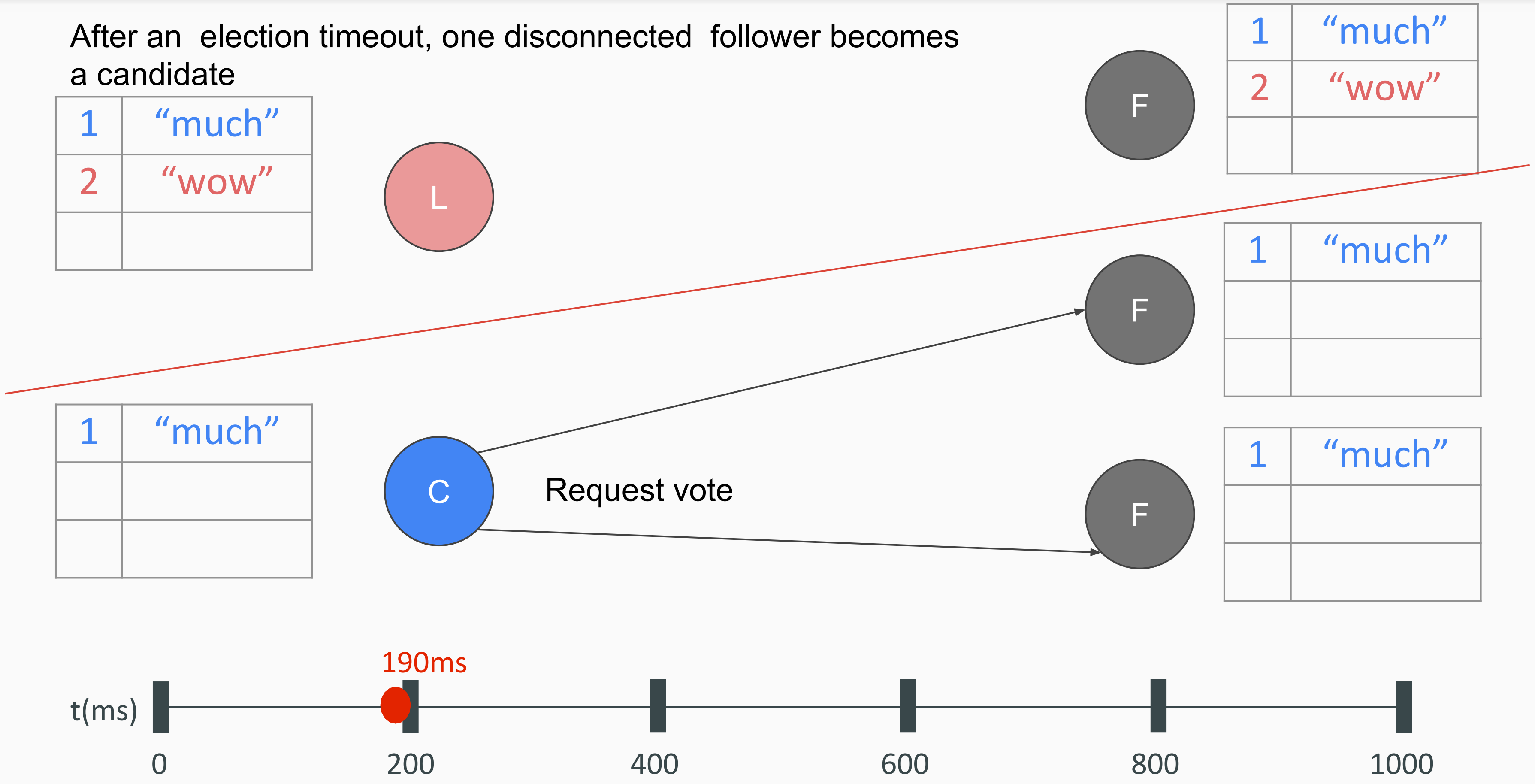


# RAFT - Log Replication



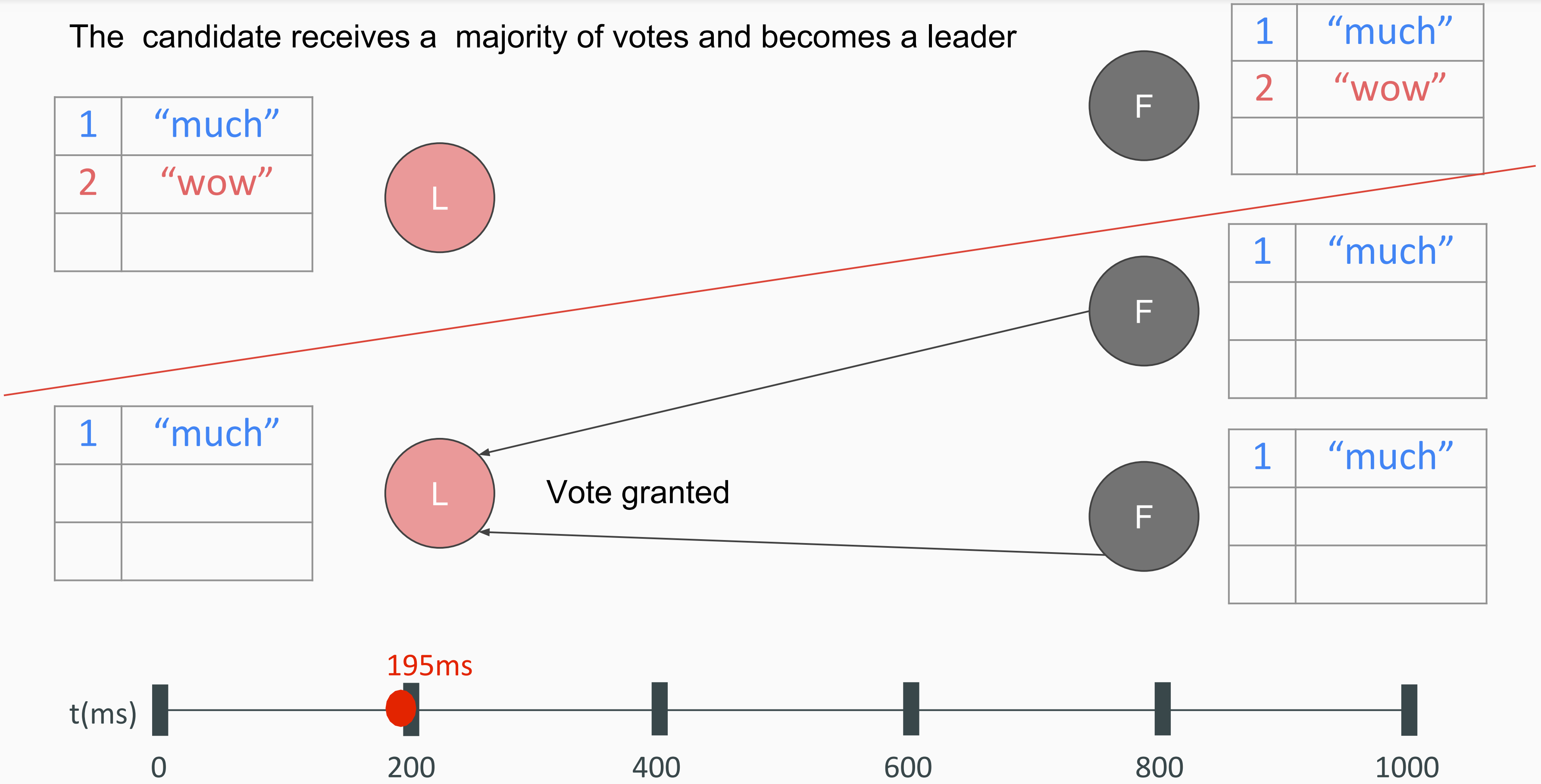
## RAFT - Log Replication

After an  election timeout, one disconnected  follower becomes a candidate

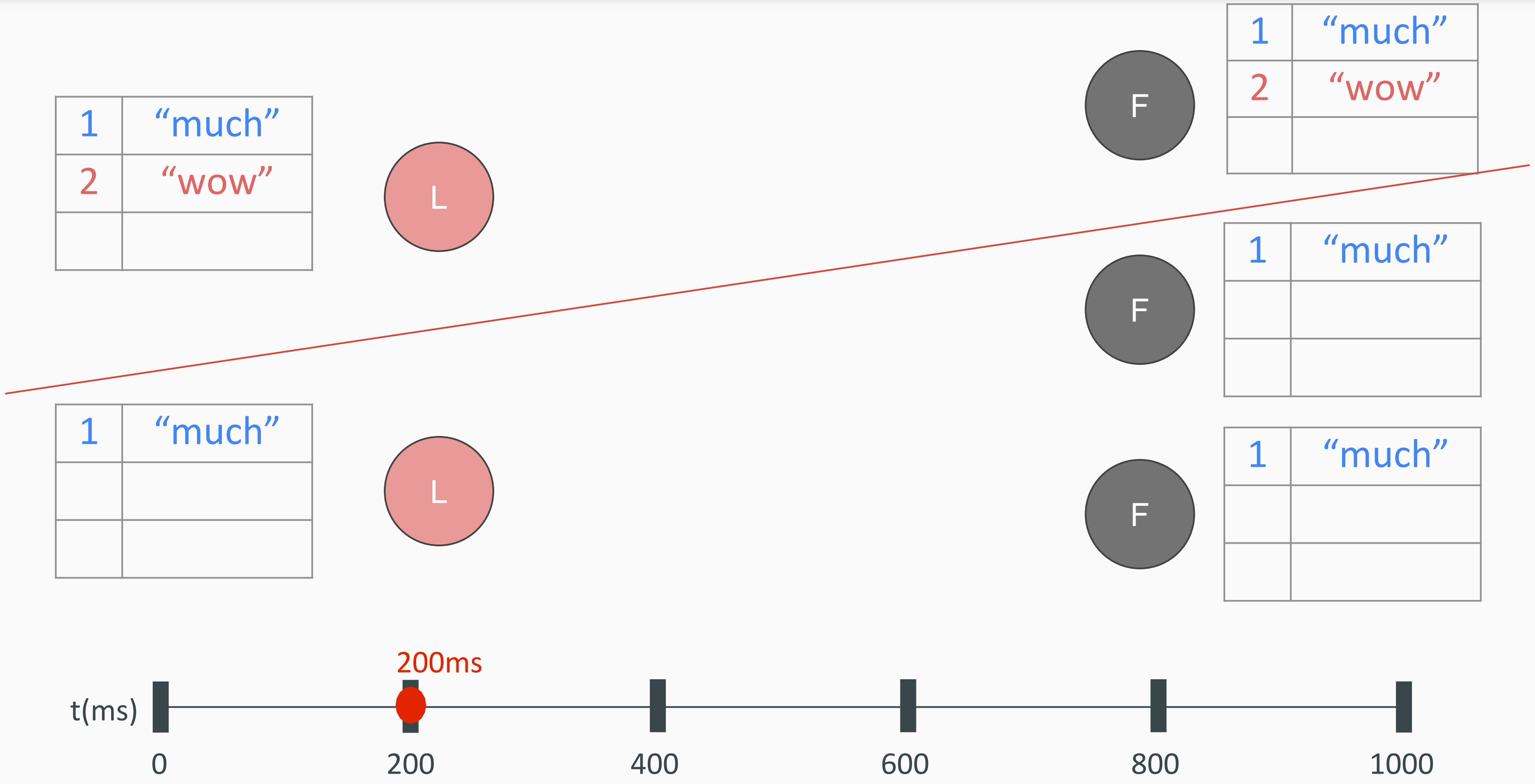


## RAFT - Log Replication

The candidate receives a majority of votes and becomes a leader

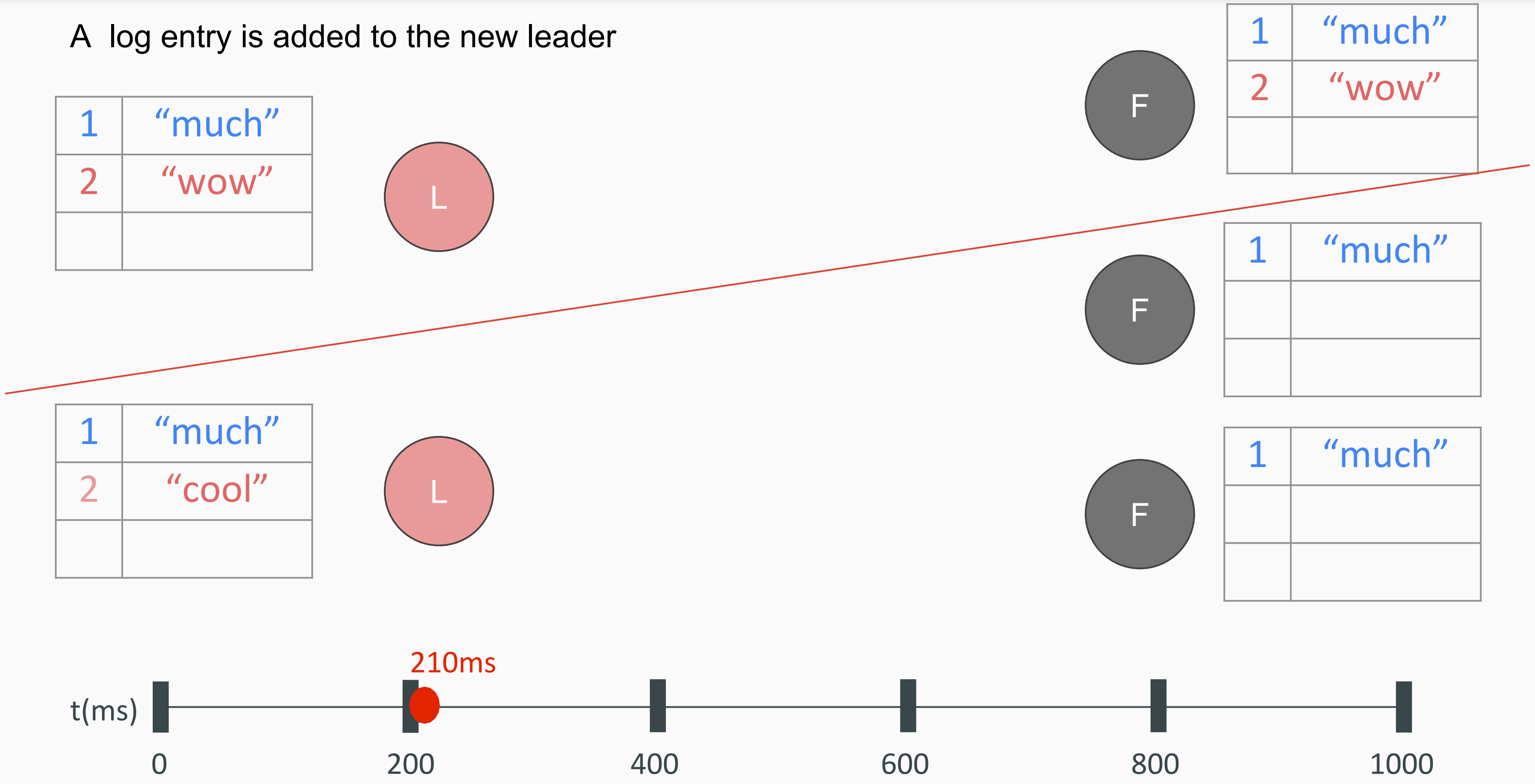


# RAFT - Log Replication



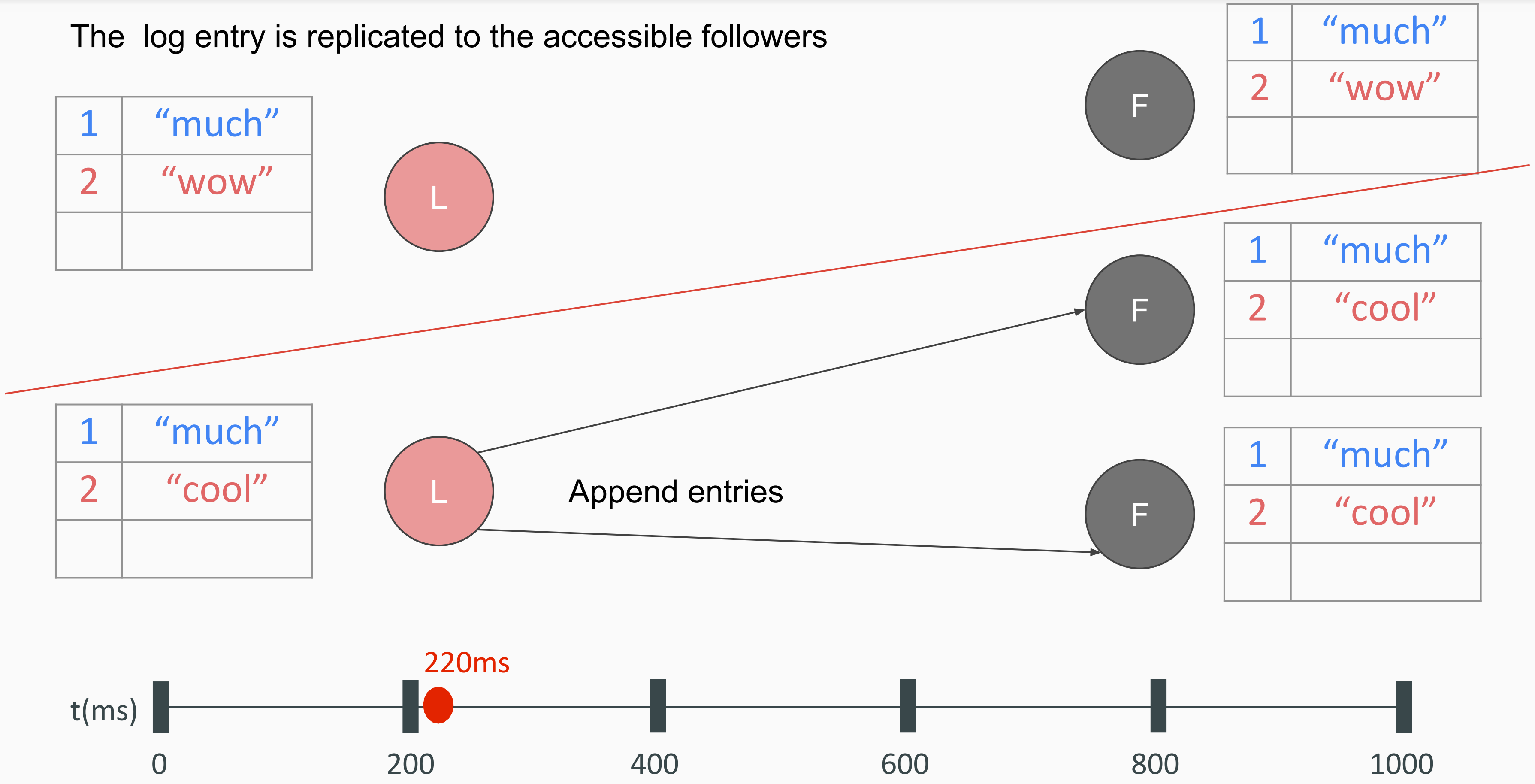
# RAFT - Log Replication

A log entry is added to the new leader



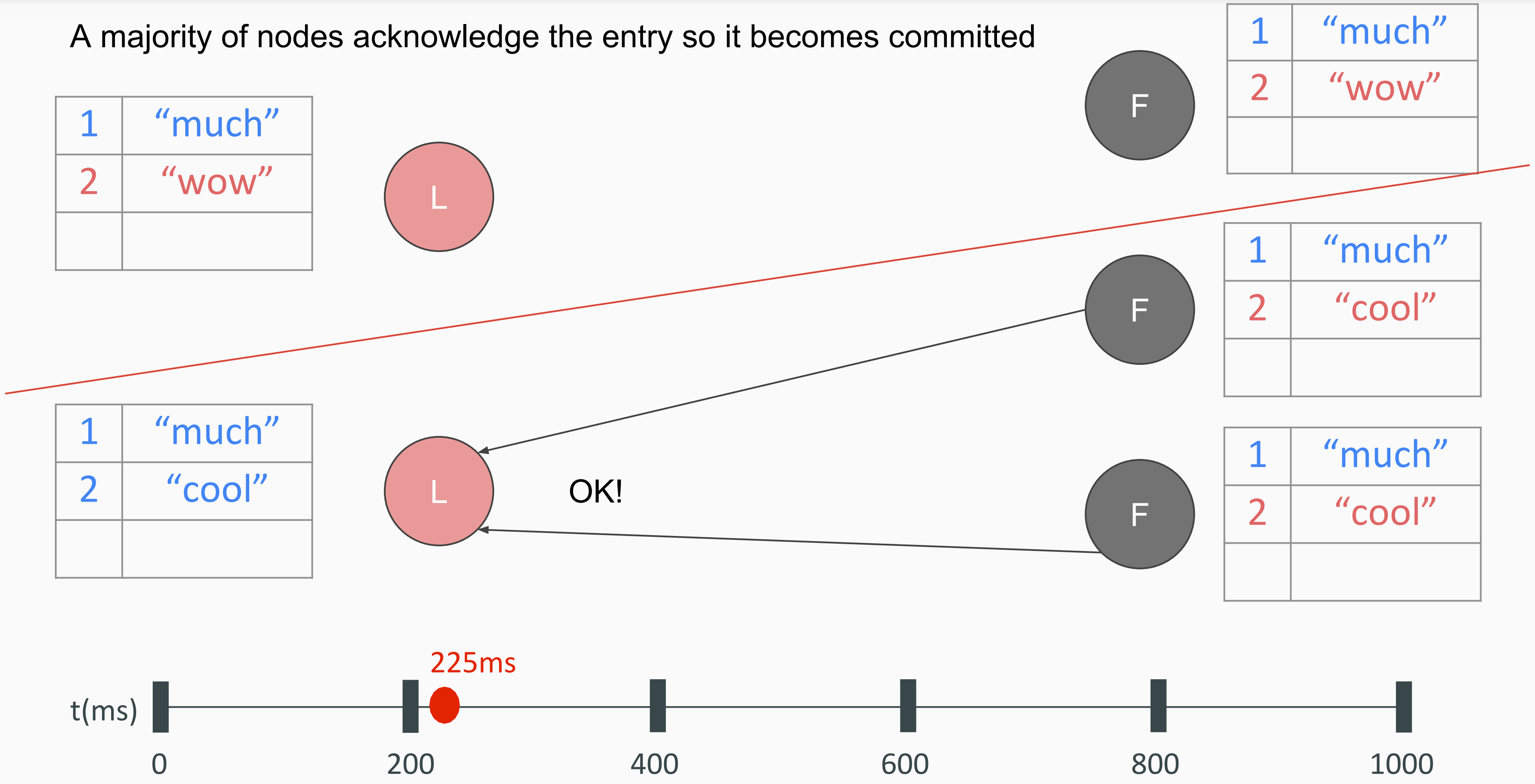
## RAFT - Log Replication

The log entry is replicated to the accessible followers



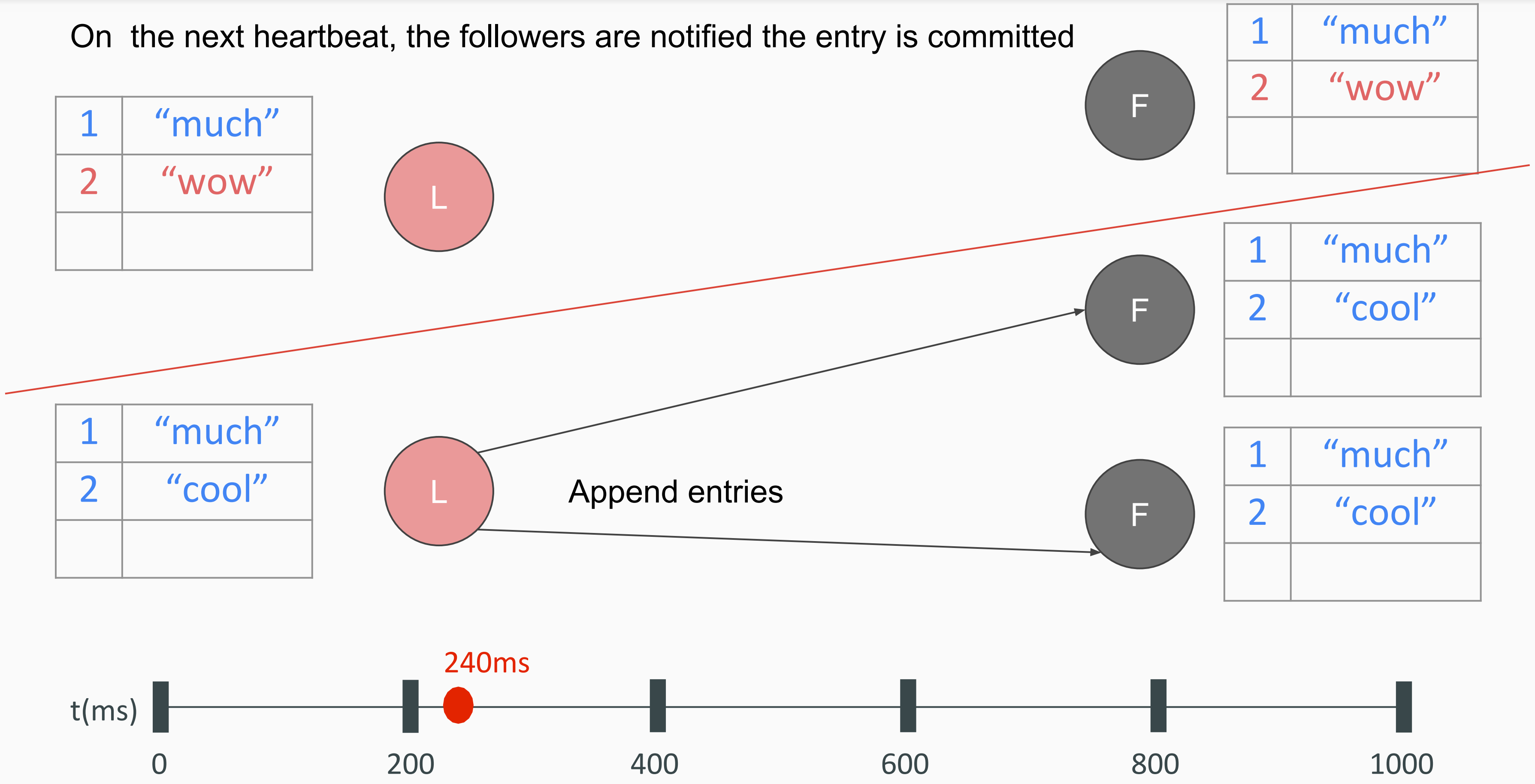
## RAFT - Log Replication

A majority of nodes acknowledge the entry so it becomes committed



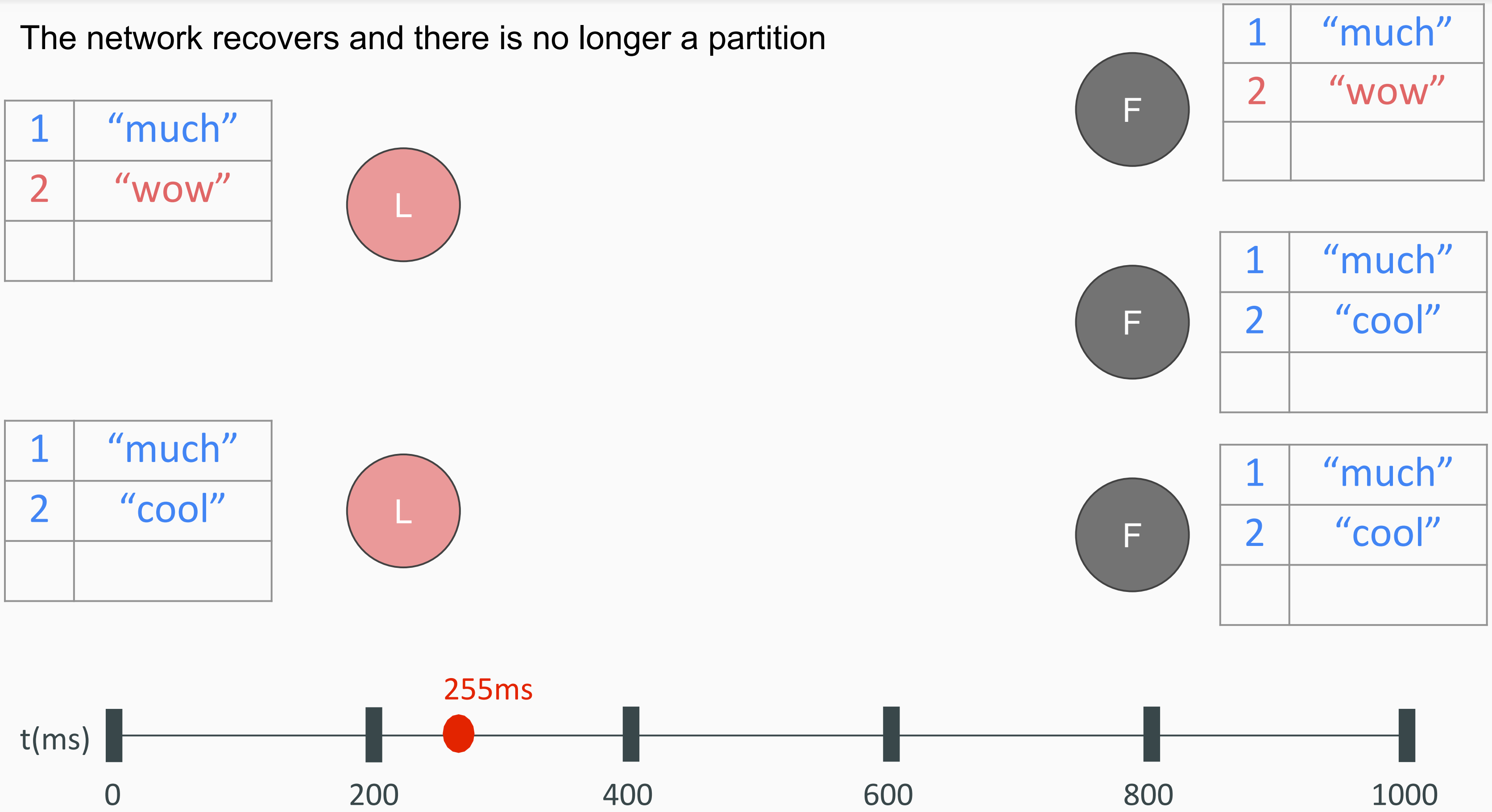
# RAFT - Log Replication

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



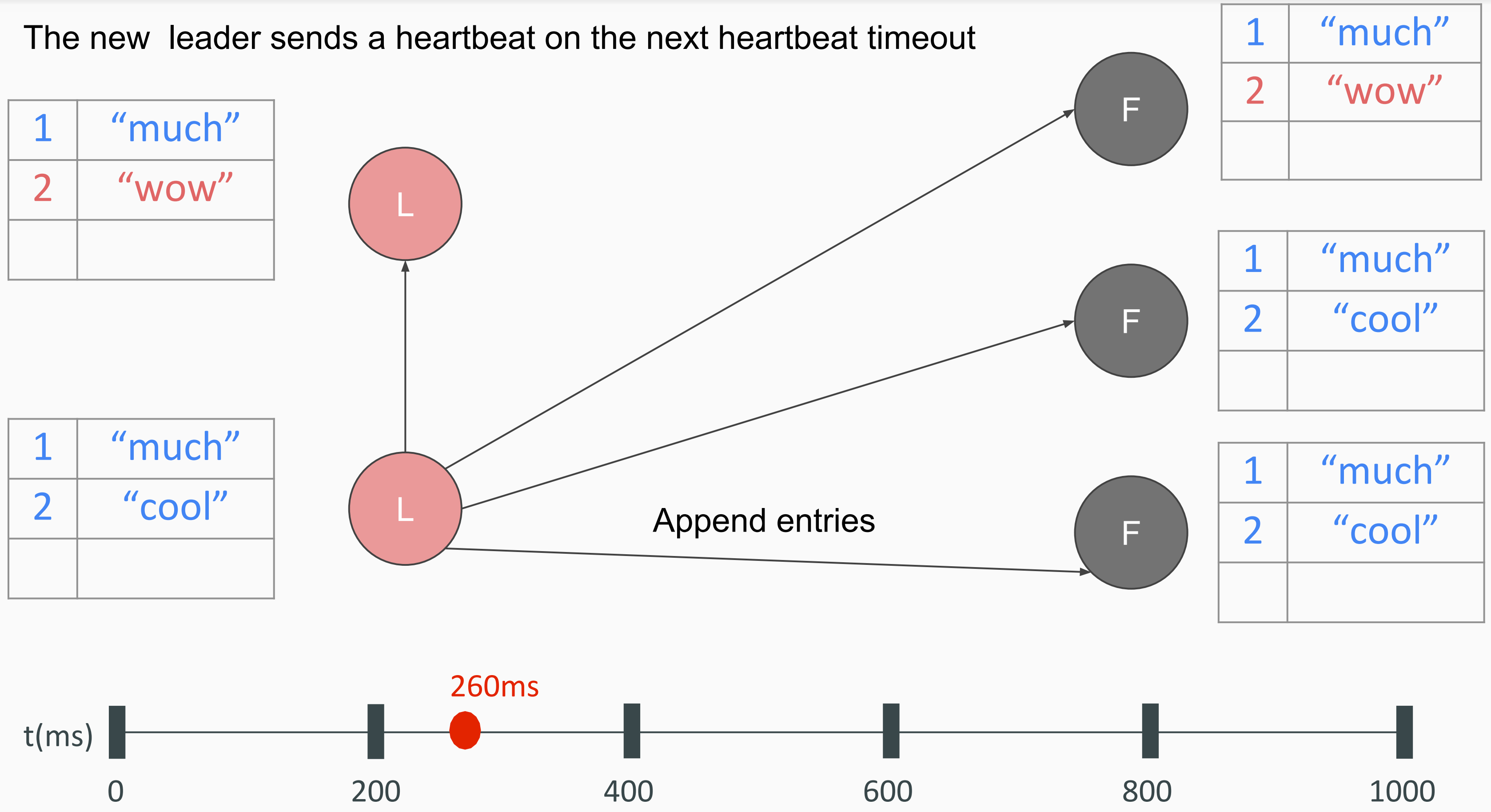
## RAFT - Log Replication

The network recovers and there is no longer a partition



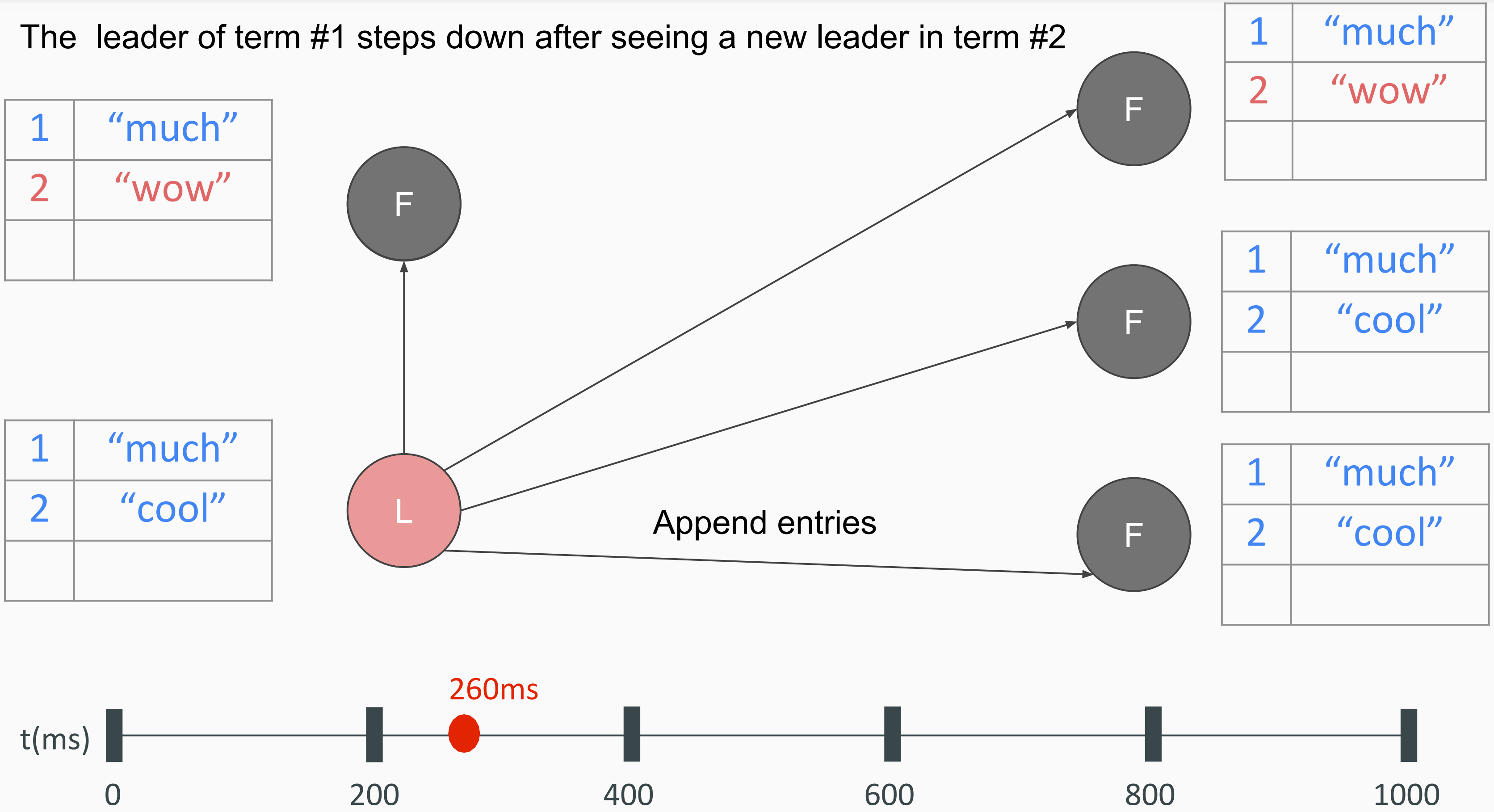
# RAFT - Log Replication

The new leader sends a heartbeat on the next heartbeat timeout



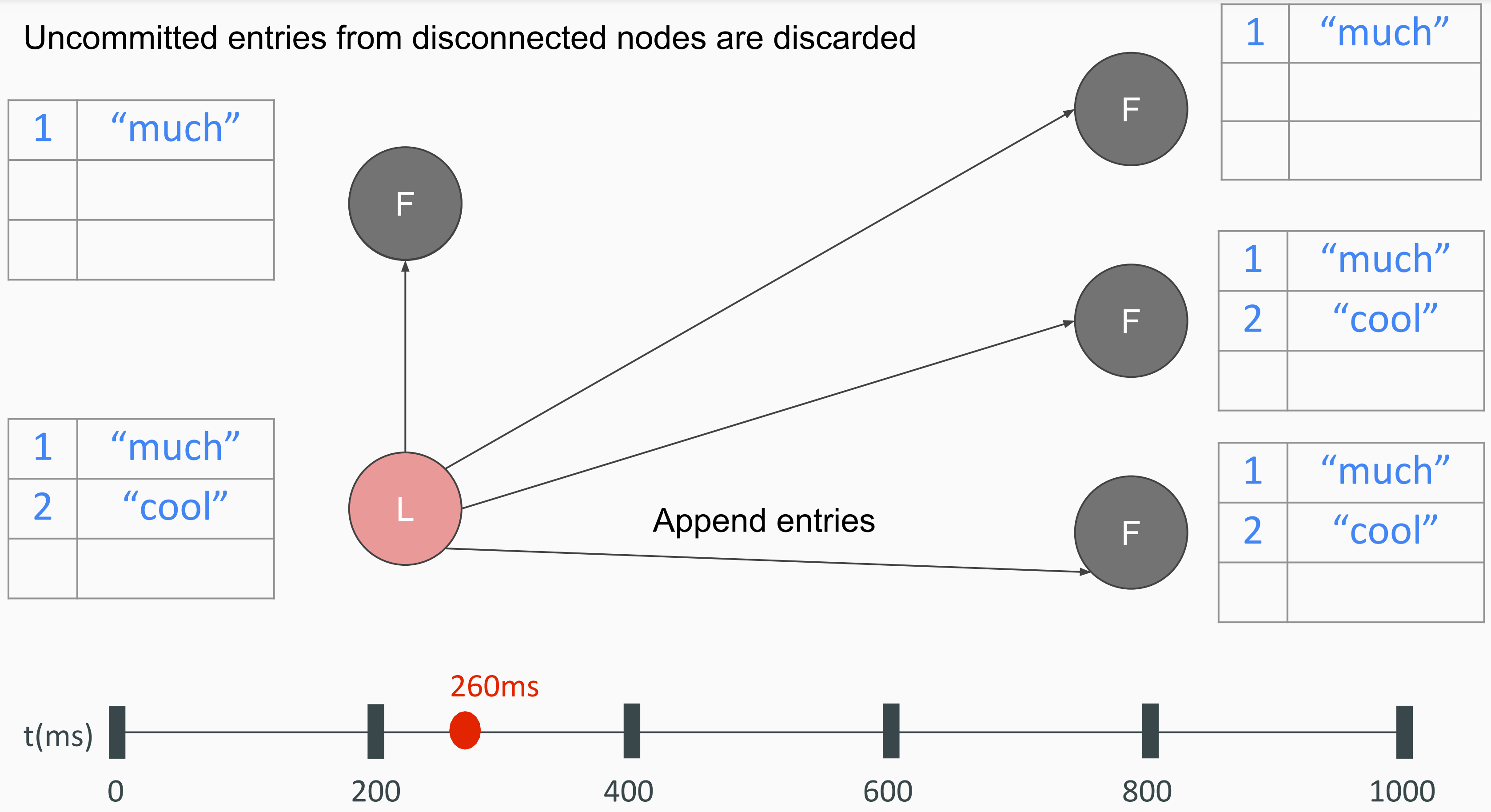
## RAFT - Log Replication

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



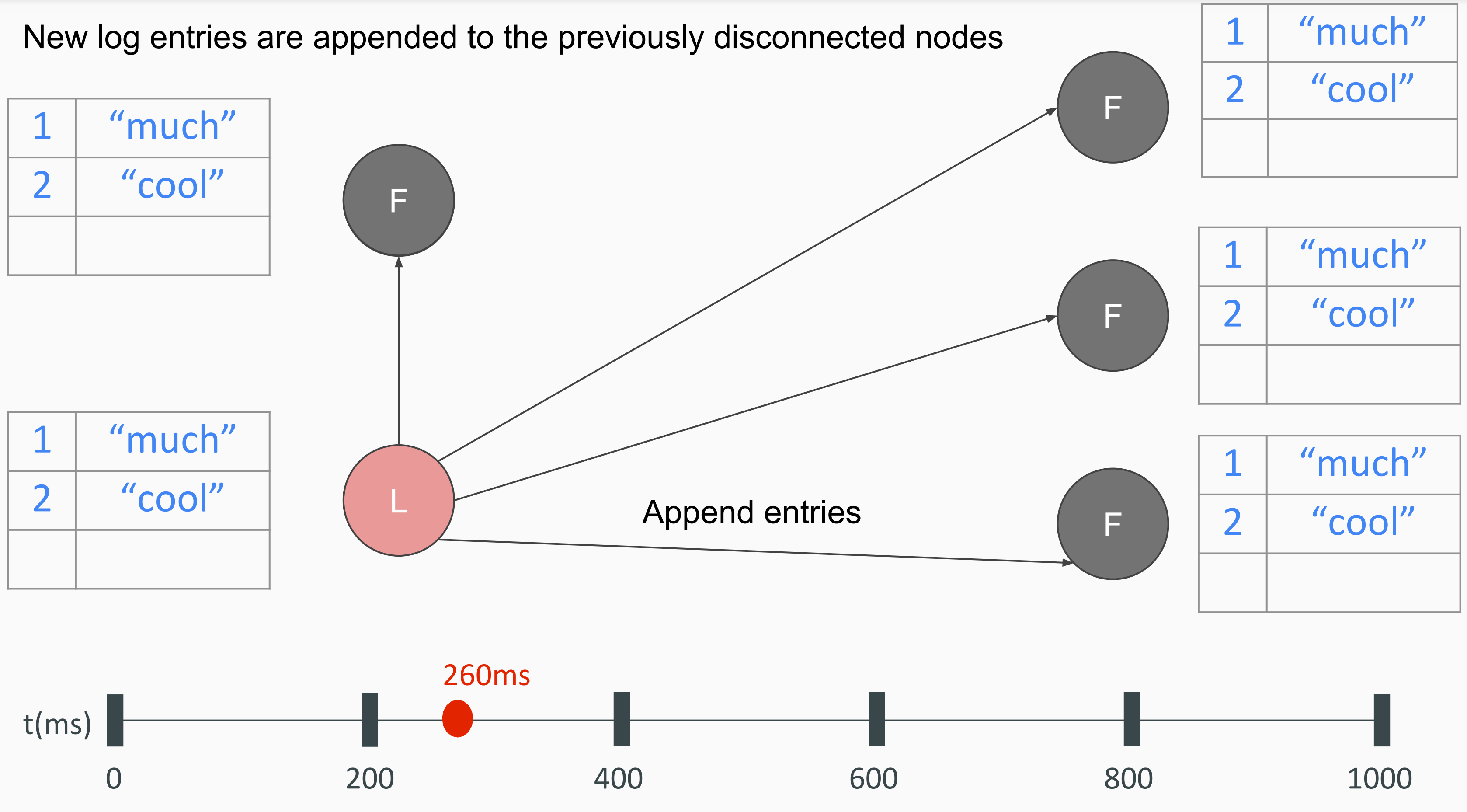
# RAFT - Log Replication

Uncommitted entries from disconnected nodes are discarded

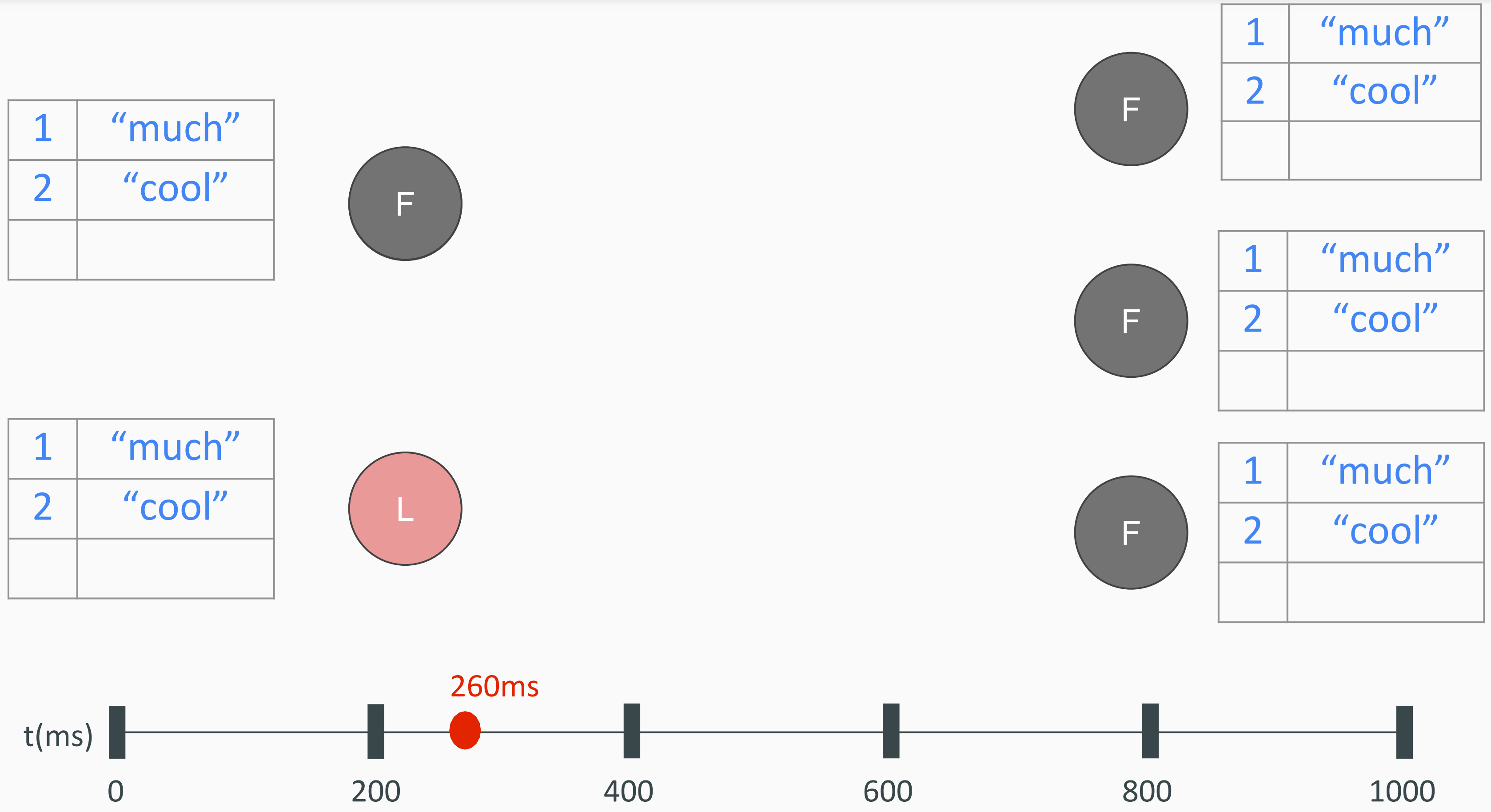


# RAFT - Log Replication

New log entries are appended to the previously disconnected nodes



## RAFT - Log Replication



SUCH WOW
VERY SYNC
SO CONSISTENT
MUCH REPLICATION
SUCH ELECTION
imgflip.com

## Bootstrapping the Cluster

- mandatory configuration options
  - **listen-peer-urls**
    default: http://localhost:2380
  - **listen-client-urls**
    default: http://localhost:2379
  - **advertise-client-urls**
    default: http://localhost:2379
  - **initial-advertise-peer-urls**
    default: http://localhost:2380

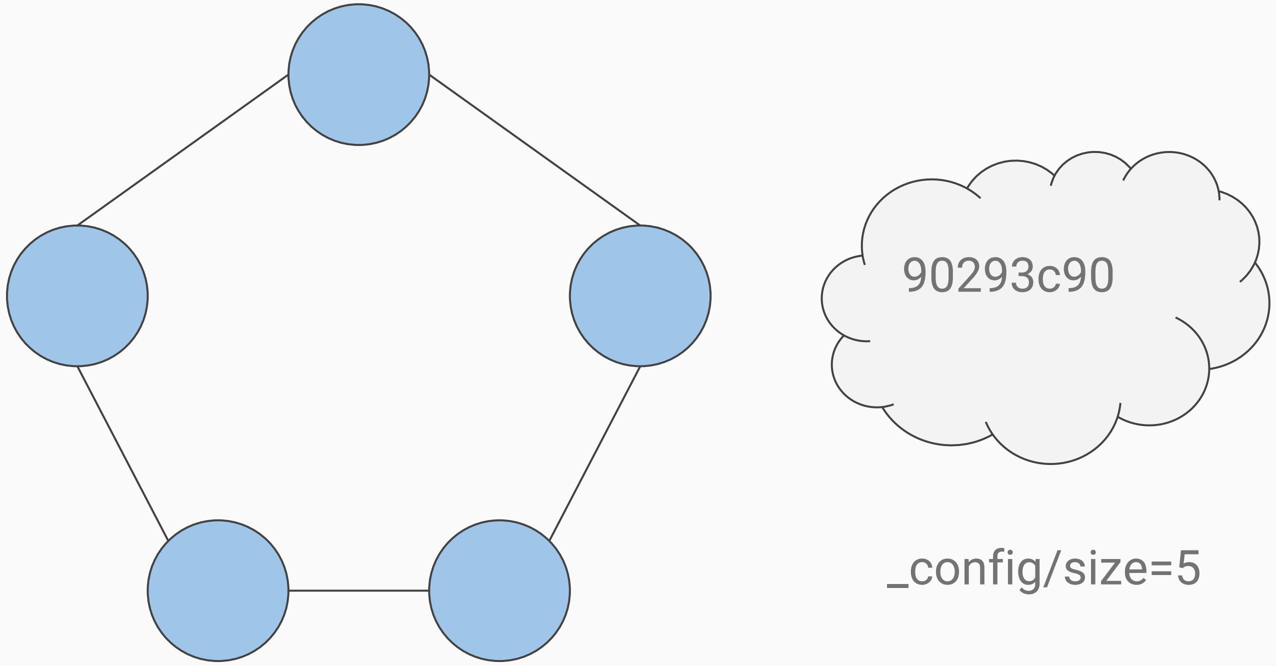
Static

- **initial-cluster**
  infra0=http://10.0.1.10:2380,
  infra1=http://10.0.1.11:2380,
  infra2=http://10.0.1.12:2380

## Bootstrapping the Cluster - Discovery URL

https://discovery.etcd.io/new?size=5

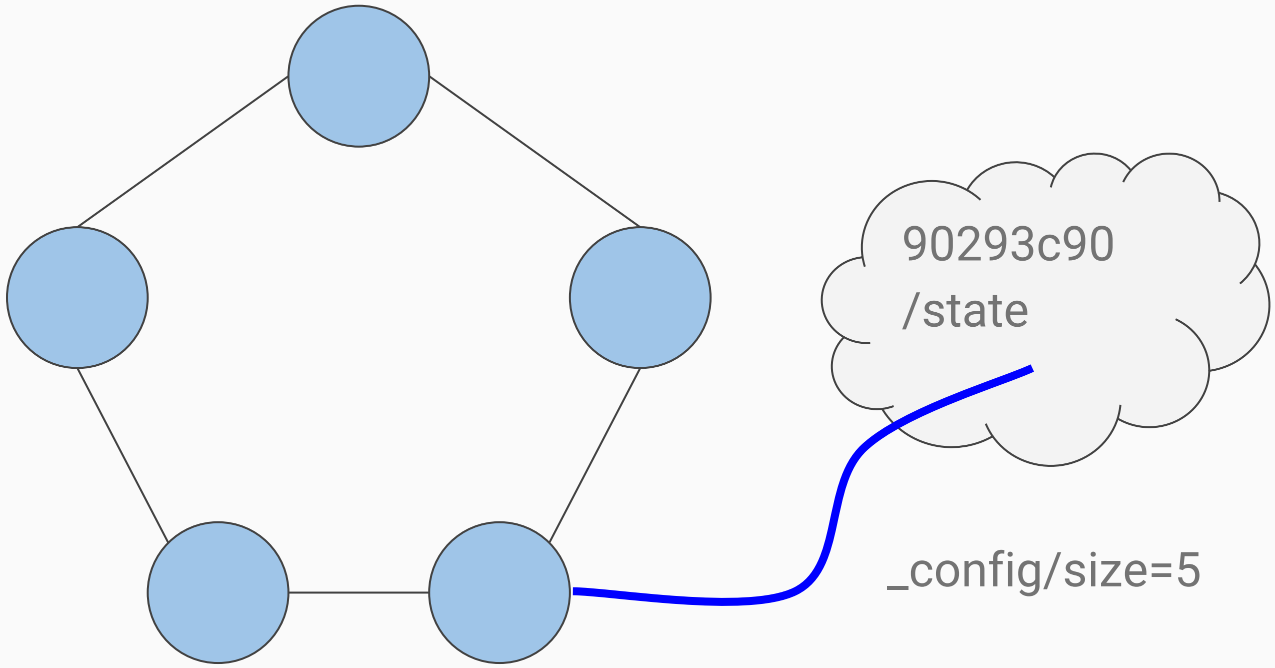
**discovery=**https://discovery.etcd.io/90293c59191021d1c27ebd9eda963f47



## Bootstrapping the Cluster - Discovery URL

https://discovery.etcd.io/90293c59191021d1c27ebd9eda963f47

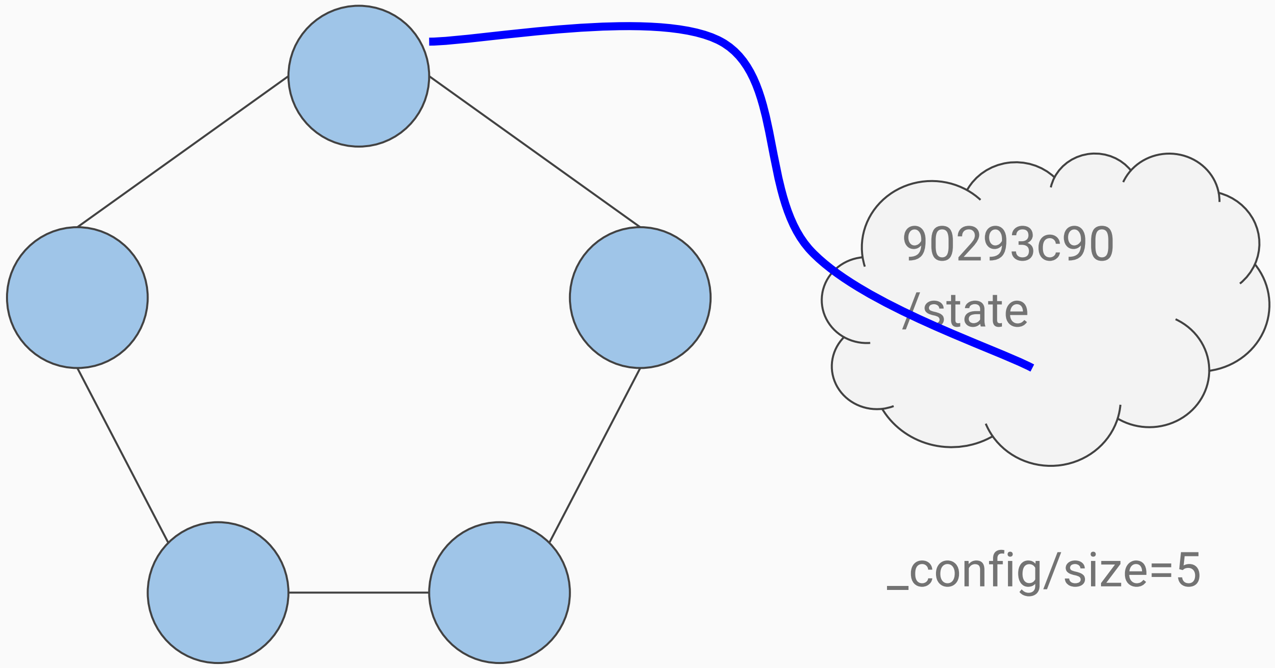
| KEY | VALUE | INDEX |
|---|---|---|
| state | started | 5890 |
| n0 | 10.0.1.10 | 5891 |



## Bootstrapping the Cluster - Discovery URL

https://discovery.etcd.io/90293c59191021d1c27ebd9eda963f47

| KEY | VALUE | INDEX |
|---|---|---|
| state | started | 5890 |
| n0 | 10.0.1.10 | 5891 |
| n1 | 10.0.1.11 | 5898 |



_config/size=5

## Bootstrapping the Cluster - Discovery URL

https://discovery.etcd.io/90293c59191021d1c27ebd9eda963f47

- When member list size meets expected value, the list is used to bootstrap every node like **initial-cluster** option in static bootstraping method
- Election process
- Cluster is ready

## Managing cluster size at runtime

**List cluster members**

```
$ etcdctl member list
```

**Add cluster member**

```
$ etcdctl member add <name> <peerURL>
```

**Remove cluster member**

```
$ etcdctl member remove <name>
```

## Comparison with similar technologies

- Zookeeper
- Consul
- Doozer

## Similarities

- Consistent and durable general-purpose K/V store across distributed system
- Based on Paxos or Raft algorithm to quickly converge to a consistent state after disconnecting one of nodes
- Paxos vs. Raft

## Zookeeper vs etcd (1)

- Zookeeper is the oldest project from compared databases
  - Mature and has big number of client bindings, tools and API’s.
  - Few years back there was no alternative
- Written in Java
  - Zookeeper is more resource hungry than any other databases

## Zookeeper vs etcd (2)

- Zookeeper is more complex than etcd
  - Harder to maintain
  - It’s harder to configure
- Zookeeper uses Zab
  - implementation of Paxos
  - Zab designed for primary-backup systems rather than for state machine replication

## Consul vs etcd (1)

- Written in GO
- Consul has more high level
  - Consul implements a full service discovery system in the library
  - DNS server interface, allowing to perform service lookups using the DNS protocol
- Uses RAFT, but different implementation than etcd
- etcd is older then Consul

## Consul vs etcd (2)

- HTTP+JSON based API, Curl-able
- Internals of consul are not public http://www.consul.io/docs/internals/index.html

## Doozer vs. etcd (1)

- Written in GO, created by Heroku before etcd
- Not developed anymore
  - has big number of forks
- Doozer implements own Paxos algorithm

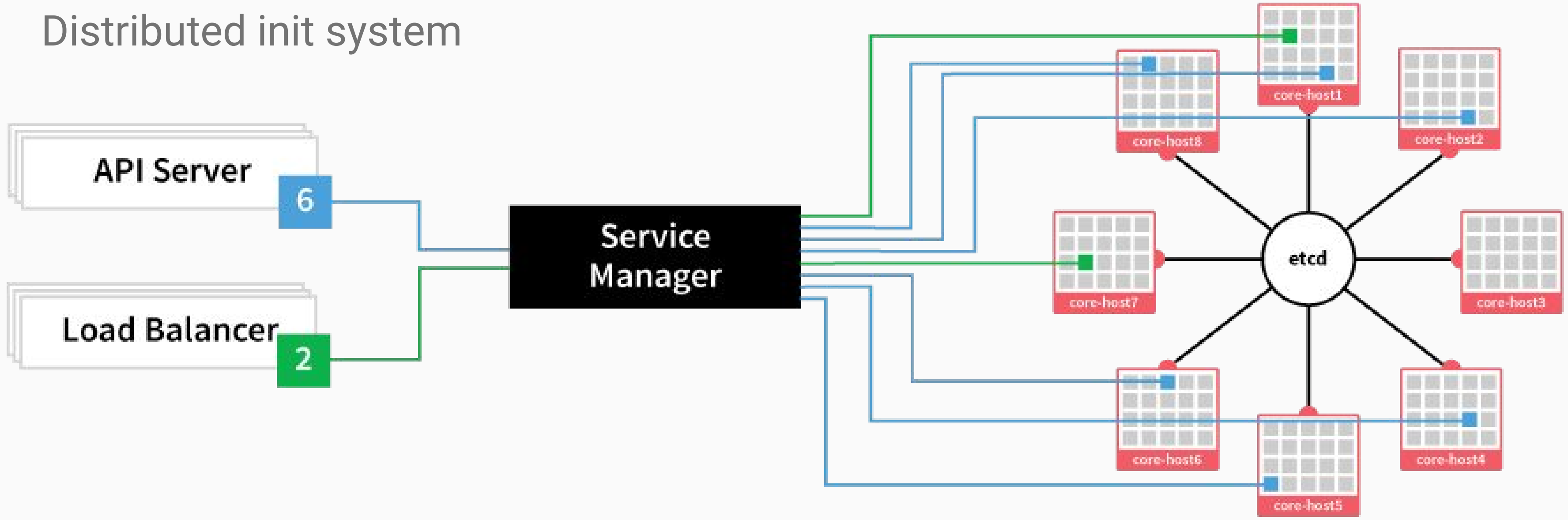
## Doozer vs. etcd (2)

- Splitted into client (doozer) and server (doozerd)
- ACL permissions are not implemented

## Who is using etcd ?

# Fleet by CoreOS

Distributed init system



## Kubernetes by Google

- container cluster manager
- **etcd** takes care of storing and replicating data used by Kubernetes across the entire cluster

## Cloud foundry

- cache for information about where and how processes are running within the container runtime
- discovery mechanism for some components.

Many more: 500+ projects on github are using etcd

## **etcd** by CoreOS

- Distributed Key-Value store
- Raft consensus protocol
- High Availability and Failure tolerant
- https://coreos.com/etcd/
- https://github.com/coreos/etcd

Thank you