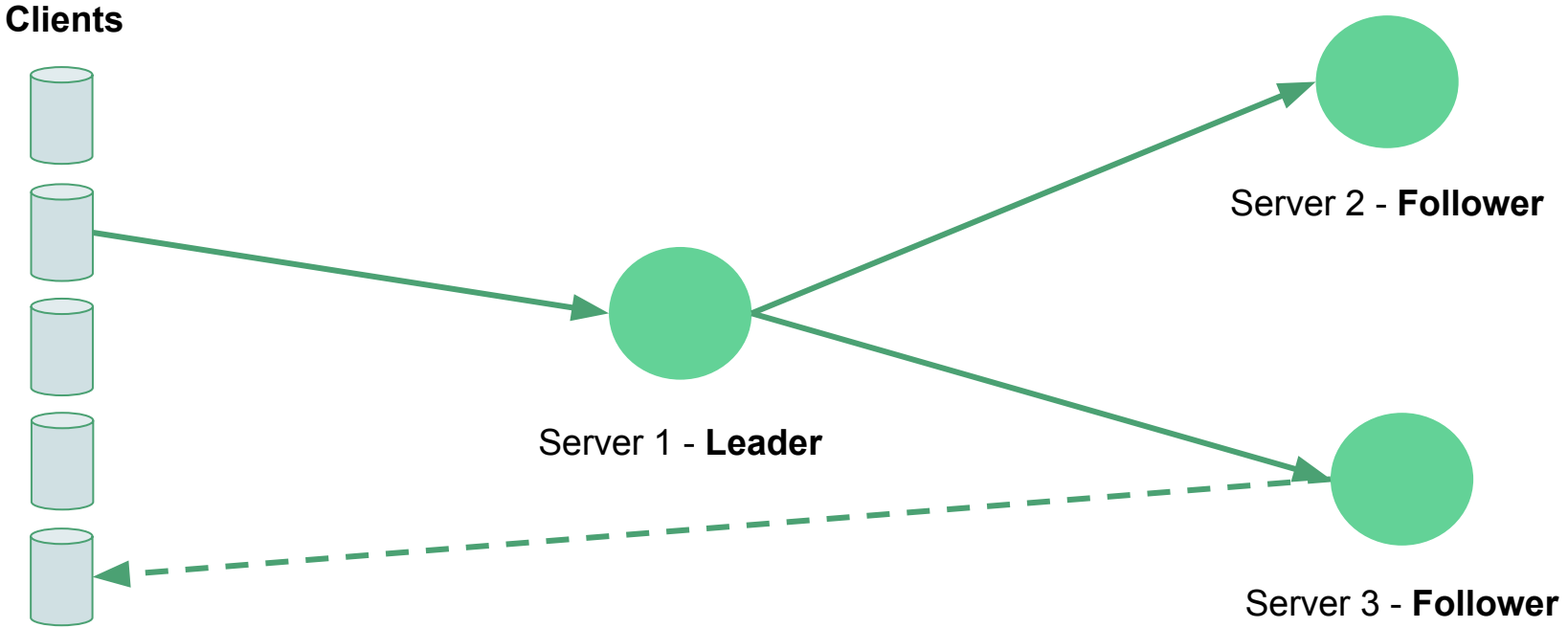
Zatt

A Python implementation of the Raft algorithm for distributed consensus

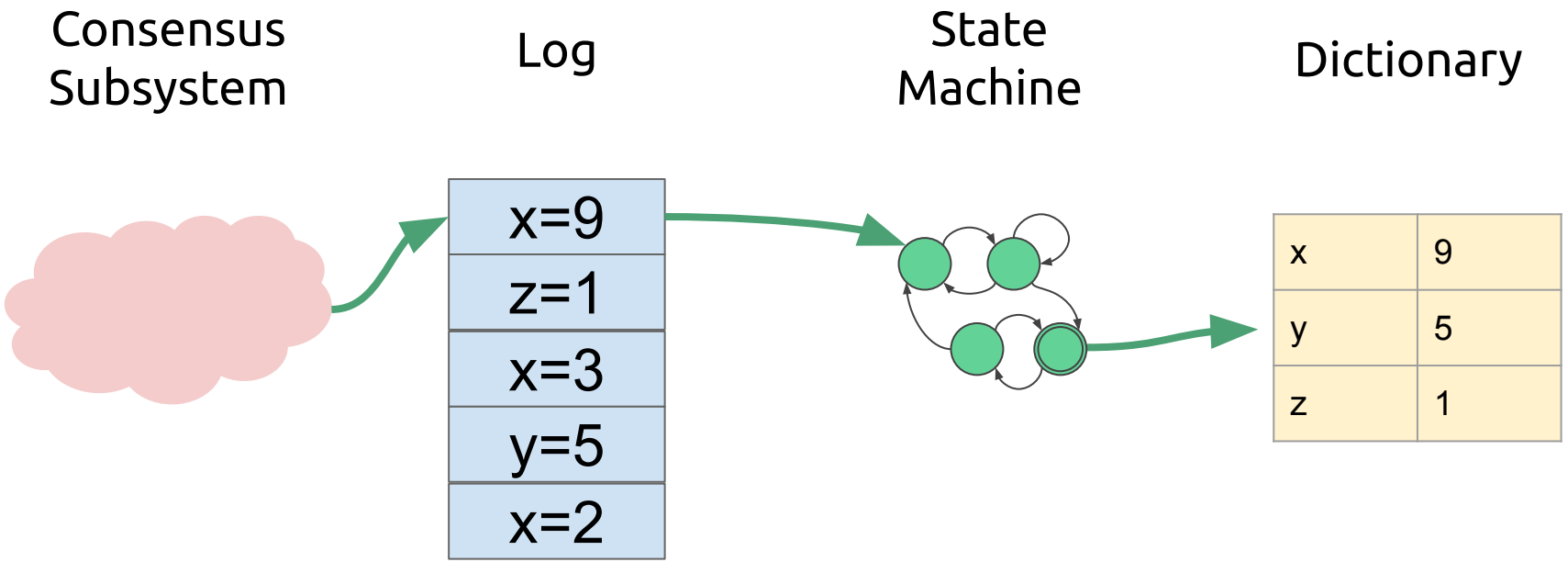
The Consensus Problem

Agree on a **shared state** in a distributed system in the presence of **faulty processes**

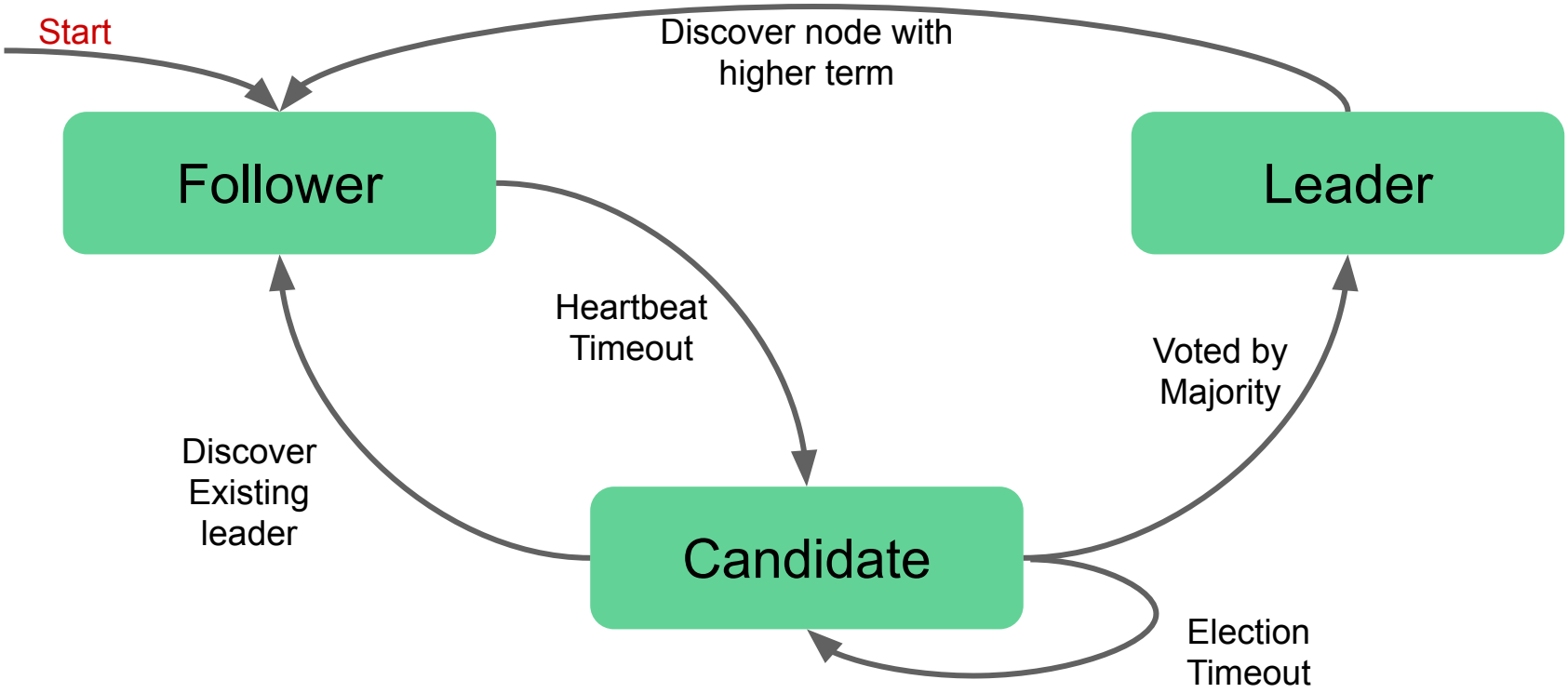
Raft Overview



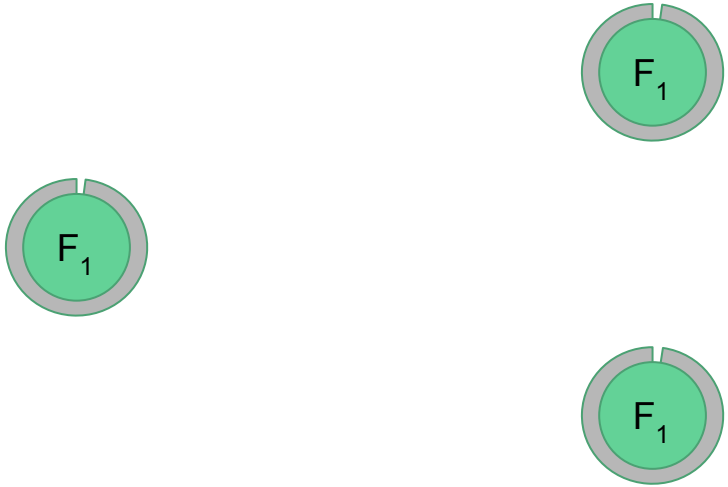
Raft Overview: State Machine Replication



Raft Overview: Election

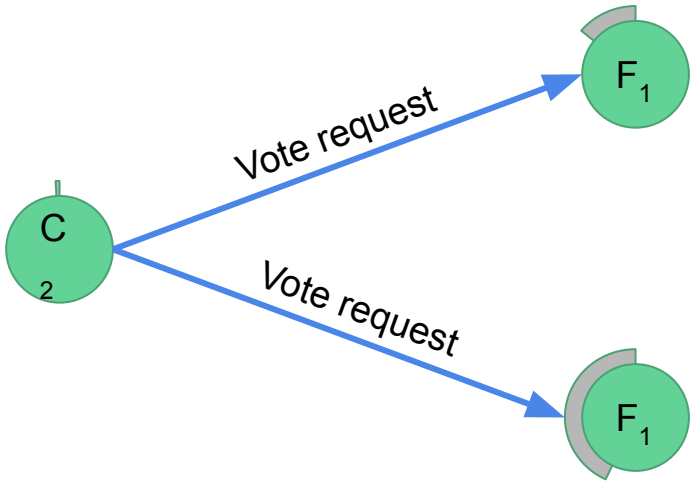


Raft: Leader Election

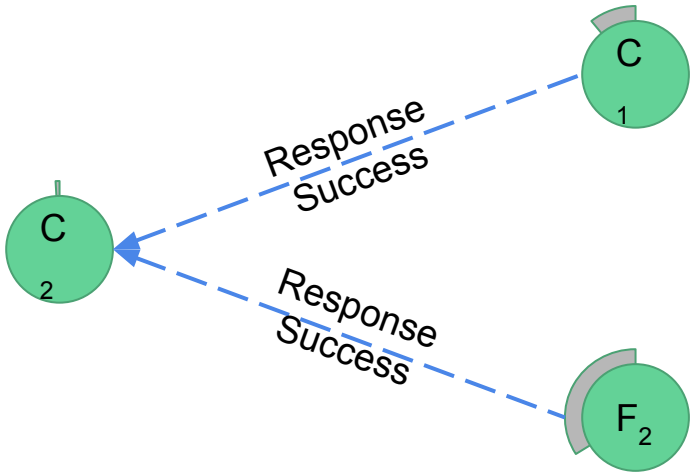


Raft: Leader Election

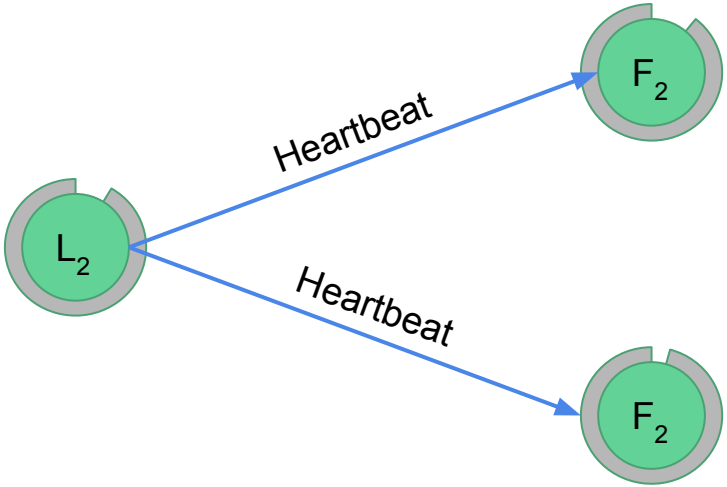
Raft: Leader Election



Raft: Leader Election



Raft: Leader Election



Implementation: Zatt

Zatt: **Client**

Client 1

```
$ python3
Python 3.6.1 on linux
>>> import zatt.client
>>> d = zatt.client.DistributedDict('node1.mycluster.io')

>>> d['hello'] = 'world'
>>>
>>>

>>> d
{'hello': 'mars'}
```

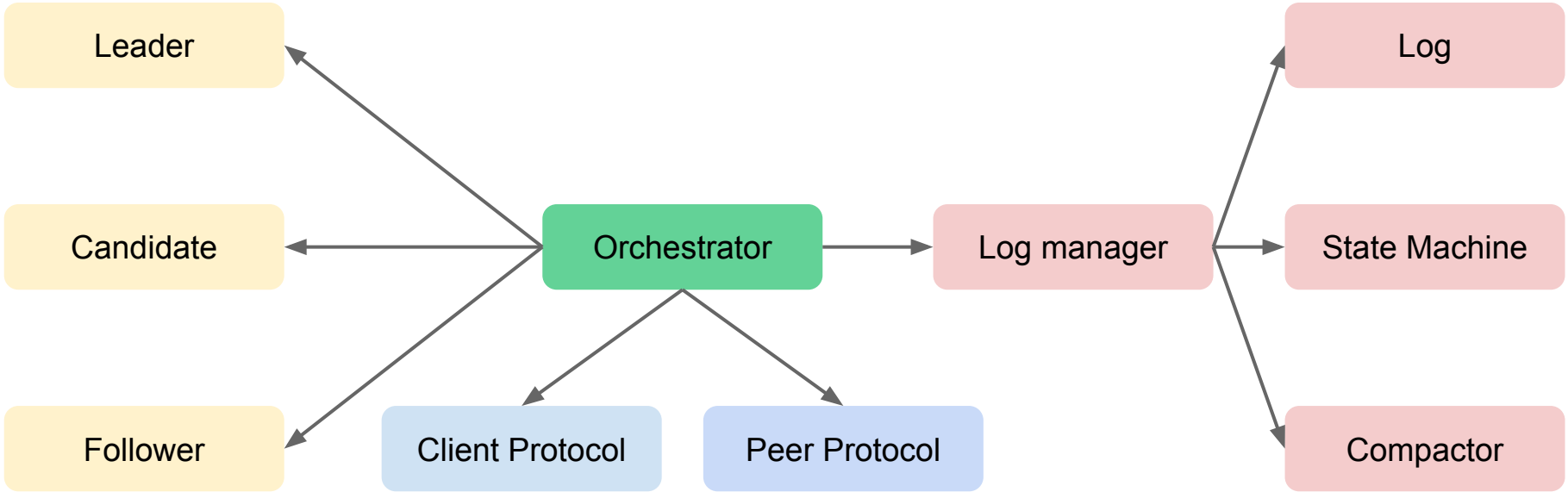
Client 2

```
$ python3
Python 3.6.1 on linux
>>> import zatt.client
>>> d = zatt.client.DistributedDict('node1.mycluster.io')

>>>
>>> d
{'hello': 'world'}

>>> d['hello'] = 'mars'
```

Zatt: Server: **Architecture**



Zatt: Architectural choices: **Concurrency**

Multiprocessing

- Independent i/o scheduling
- No GIL
- Isolation
- Message passing
- Slow spawn

Multithreading

- Shared i/o scheduling
- GIL
- Lightweight
- Sync primitives necessary

Twisted

- Green threads
- Time proven
- Python2 only
- Overkill

Asyncio

- Green threads
- Cheap context switching
- stdlib
- Non blocking i/o
- No Sync primitives

Zatt: Architectural choices: **Serialization**

	JSON	MsgPack	Cap'NProto	ProtoBuffer	FlatBuffer
Encoding	text	binary	binary	binary	binary
Append to Array	×	✓	✓	×	×
Schema	×	×	✓	✓	✓
Stream de/serialization	×	✓	✓	×	✓
Mutable state	✓	✓	×	×	×
Present in PyPi	✓	✓	✓	×	×
Zero copy	×	×	✓	×	✓

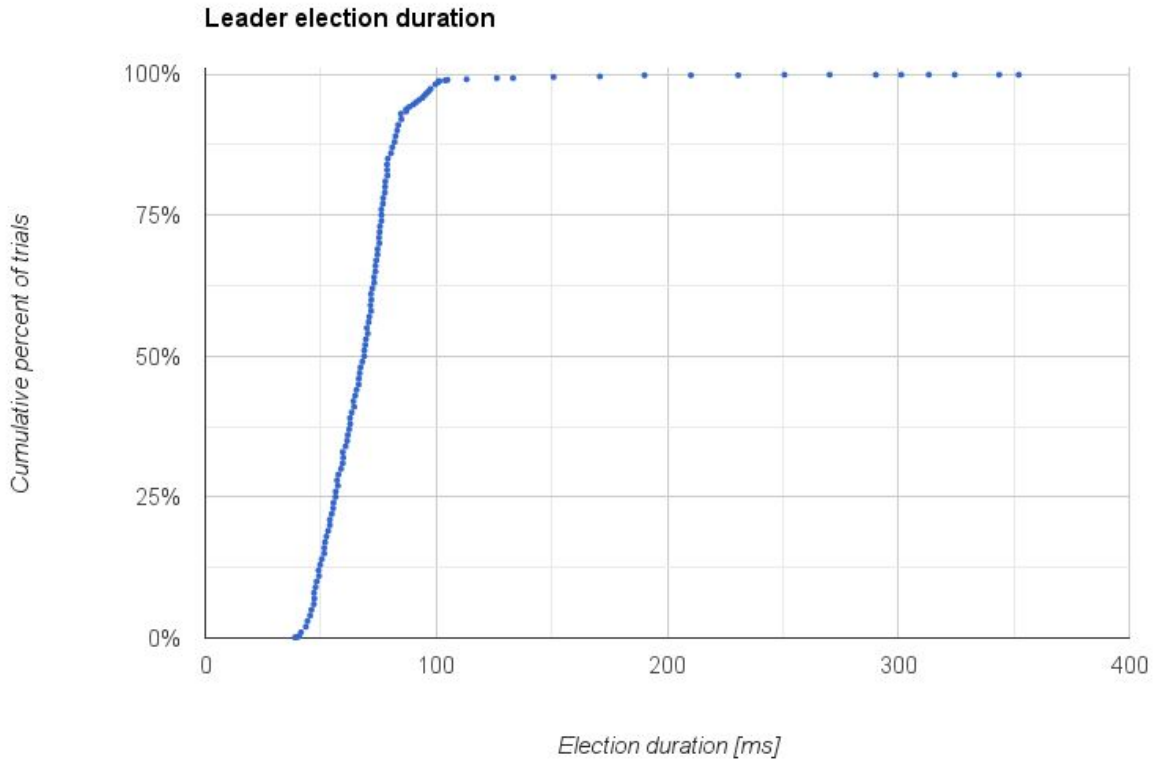
Zatt Performance: **Election Speed**

Setup

- Leader removed in stable cluster
- Cluster
  - 5 AWS m4.large
  - <4ms ping

Results

- 1000 trials
- 87% of elections under **80ms**
- 98% under 100ms



Zatt Performance: **Distributed** testing

Setup

- Used for performance evaluation
- Implemented with docker containers

6 Servers
C2L
Intel C2750
2.4 GHz
32 GB ram
5 Gbps NIC

64 Clients
C1
ARMv7
2.4 GHz
3 GB ram
1 Gbps NIC

2 Each
M4.large
Xeon E5-2666v3
2.6 GHz
32 GB ram
1 Gbps NIC

Zatt Performance: **Distributed** testing

Servers	Clients	Workers	Entries	size[B]	time[s]	req/s	Leader Throughput [KB/s]
2	4	256	1000	1000	10.94	91	91.4
2	4	256	5000	1000	15.22	657	656.9
2	4	256	10000	1000	32.01	312	312.4
2	4	100	5000	1000	11.85	422	421.8
2	4	100	5000	100	4.17	1200	120.0
2	2	100	5000	100	3.13	1597	159.75
2	2	100	10000	100	8.53	1425	424.95

Zatt: The real world

30+

stars

15+

forks

180

clones

Zatt: Future work

- UDP datagram limit workaround
  - gRPC intracluster protocol
- Unit tests
- Linearizable semantics
- Clients in multiple languages
- Observables / callbacks

Questions?

Zatt Performance: **local** testing

Setup

- Used for **profiling**, **regression testing**
- Implemented with `multiprocessing.Pool`
- Performed on laptop
  - Intel i7-4600U @ 2.10GHz
  - 8GB of DDR3 RAM @ 1600Mhz

Results

- Max throughput: **200 req/s**
  - 3 server instances
  - Peak client throughput: 64 clients
- A single client can dispatch max **100 req/s**

Invariants

- Election Safety
- Leader Append-only
- Leader completeness
- Log matching - inductive
  - Informally proven
- State machine safety
  - Formally proven

- Designed for **understandability**
  - Separate
    - leader election
    - log replication
    - membership changes
  - Strong leadership
  - Just two RPCs
  - Tested with user studies
- Formally specified and proved
- Real world **usage**
  - CoreOS/Etcd
  - RethinkDB
  - Hashicorp/Terraform

Zatt: Architectural choices: **Networking**

Intra cluster

- Uses **UDP**
- Fast
  - Acknowledgement
  - Retransmission
- No need for encryption

Client-Server

- Uses **TCP**
- Safe
- Stateful
- Supports TLS

Zatt: Architectural choices: **Client**

Refresh Policies

Always

- On every read

Lock

- Toggled by application

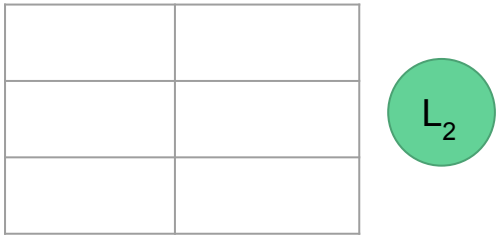
Count

- After n reads

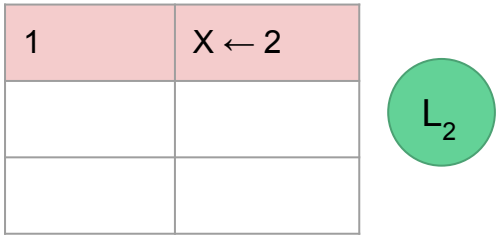
Time

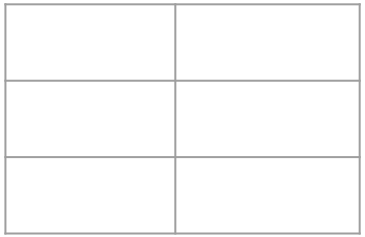
- After a given datetime.timedelta

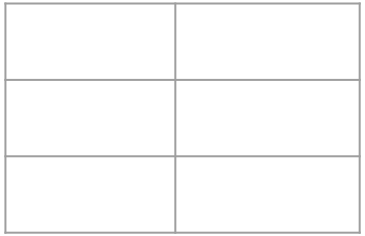
Raft: Log replication



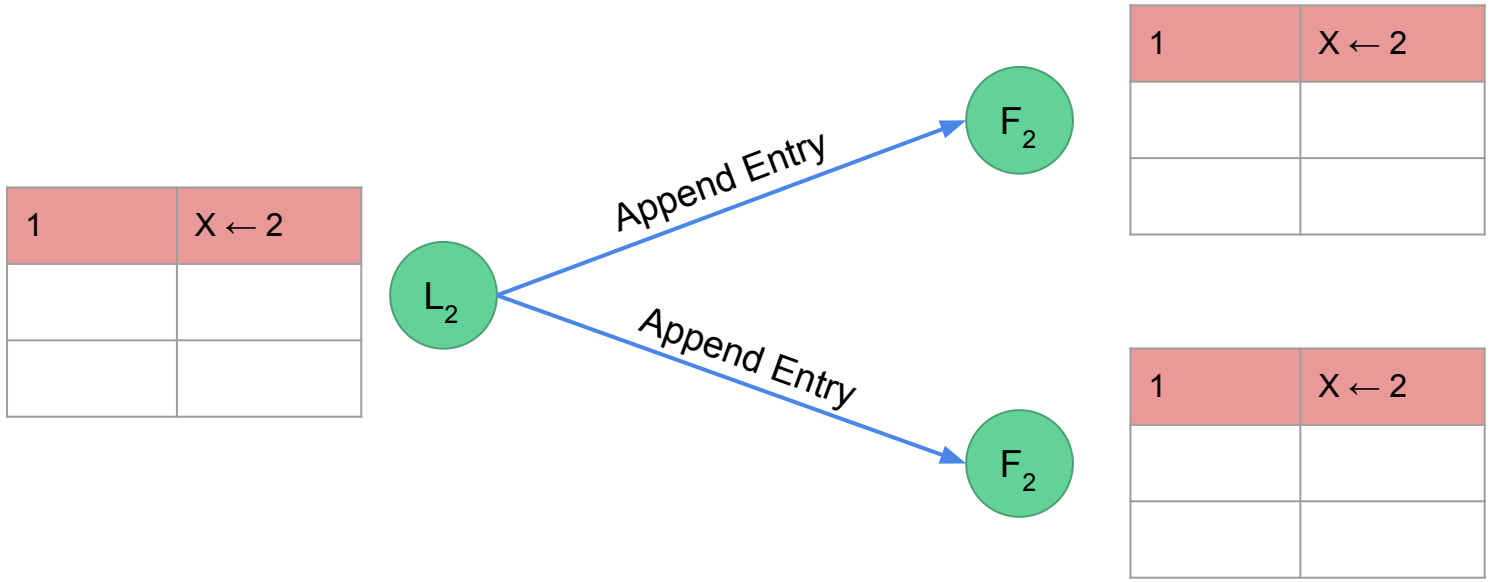
Raft: Log replication



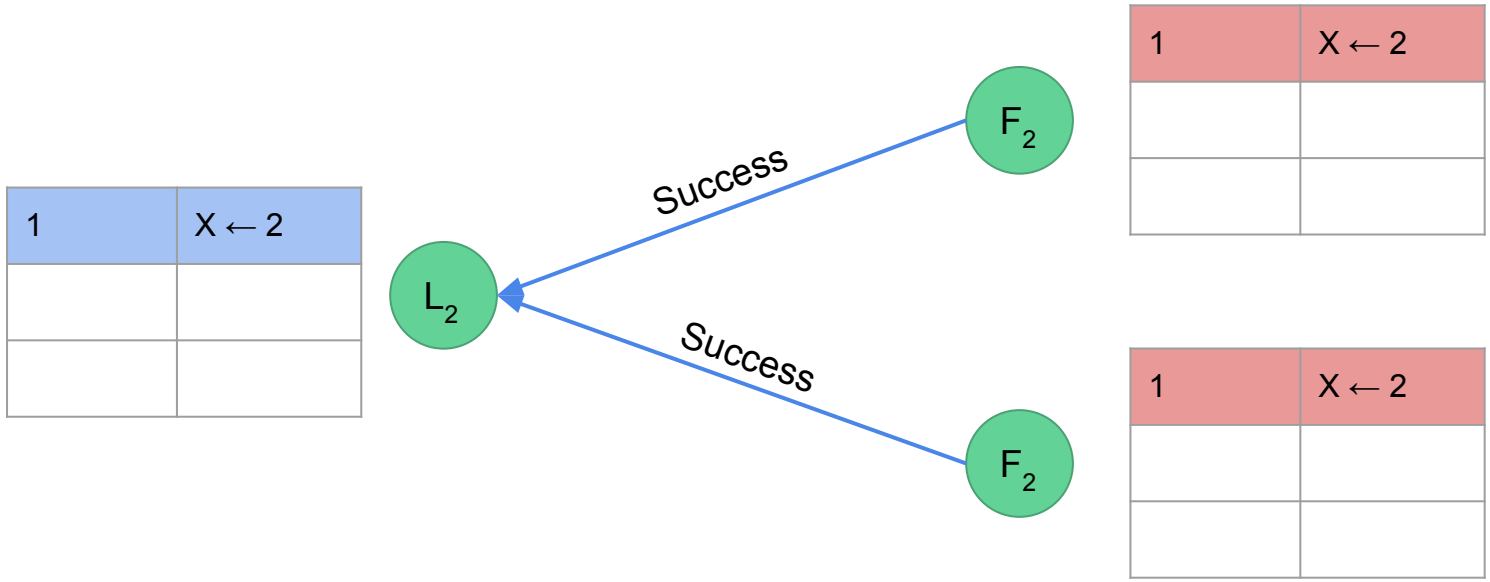




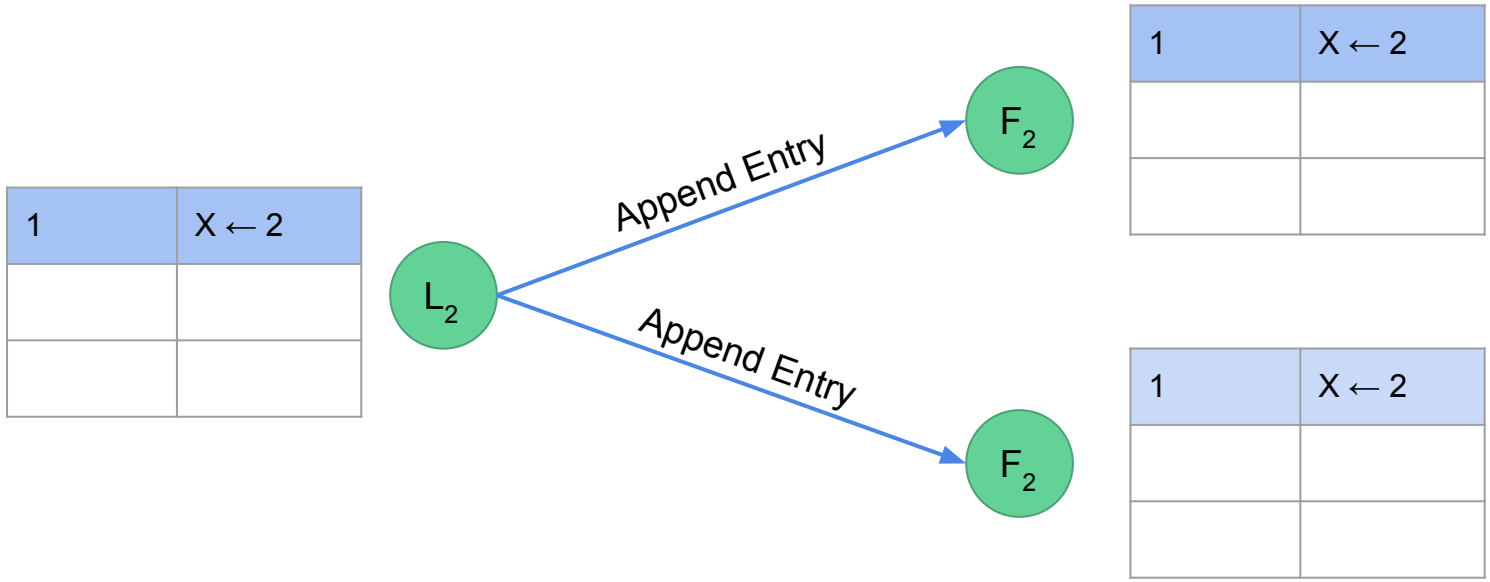
Raft: Log replication



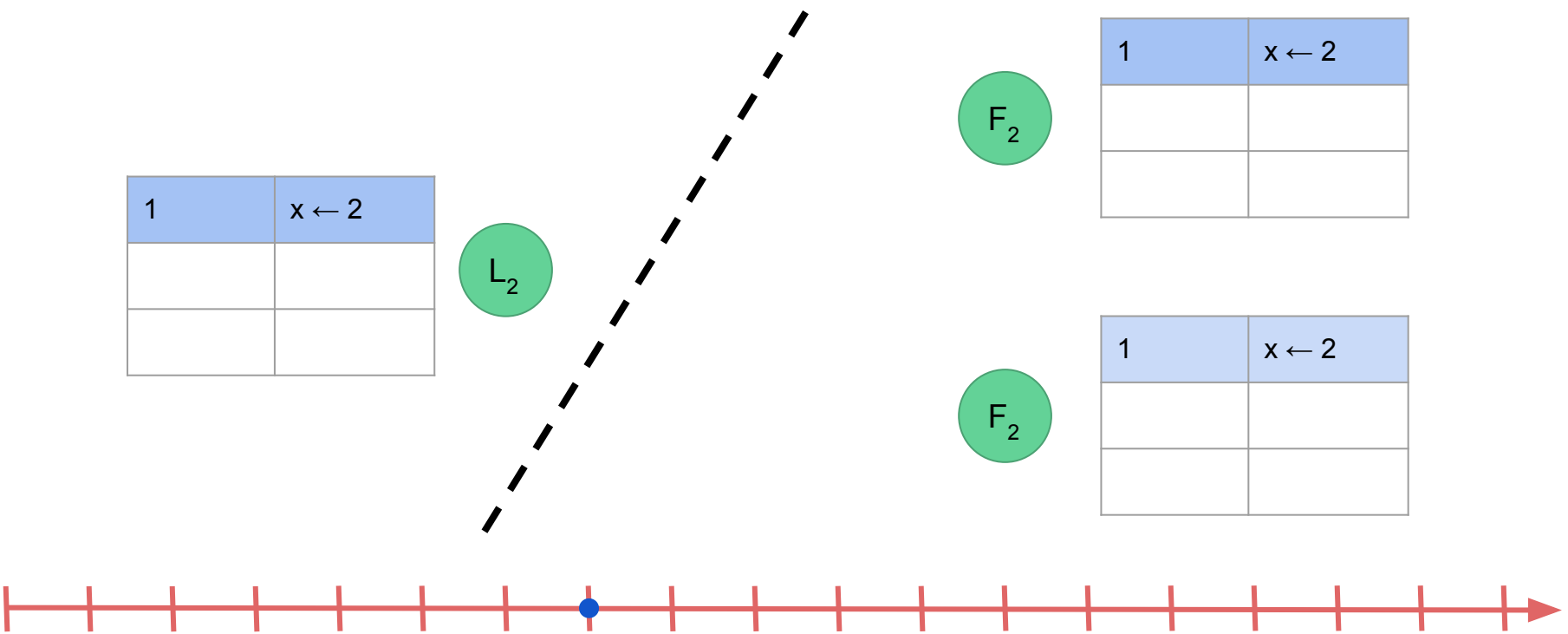
Raft: Log replication



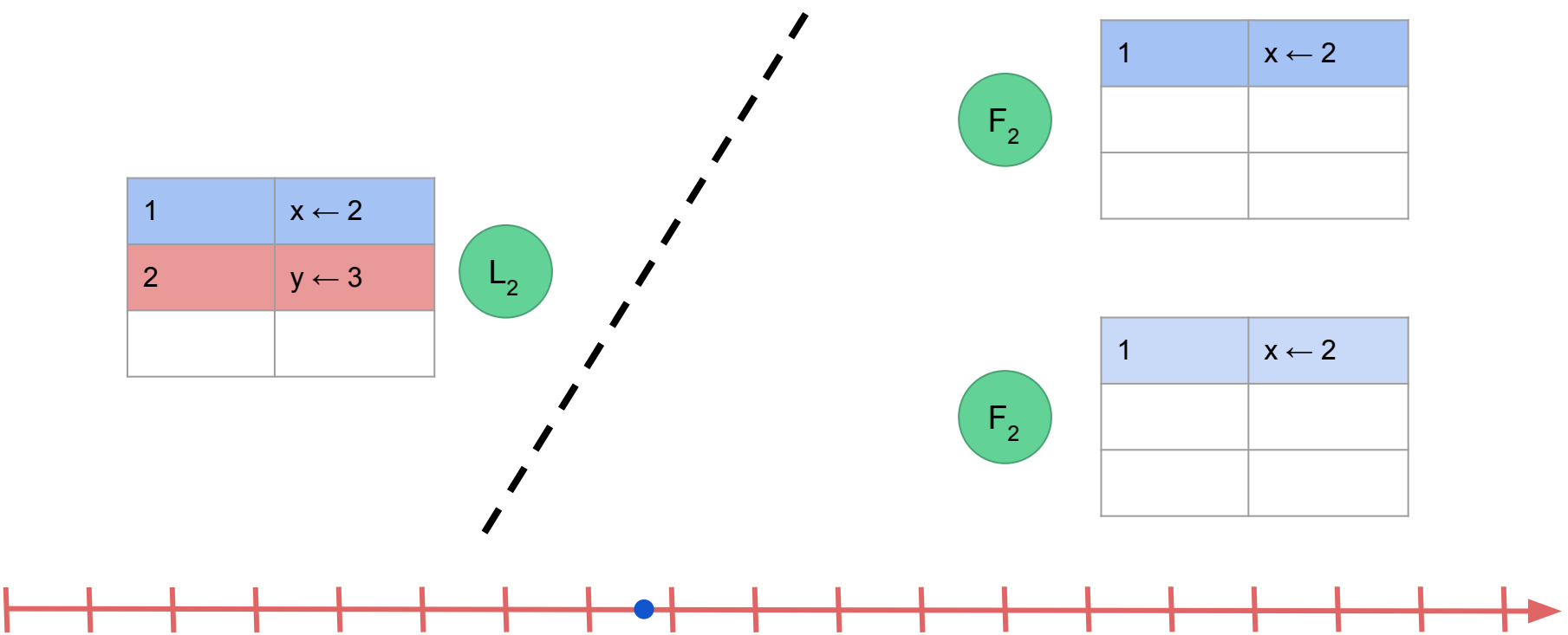
Raft: Log replication



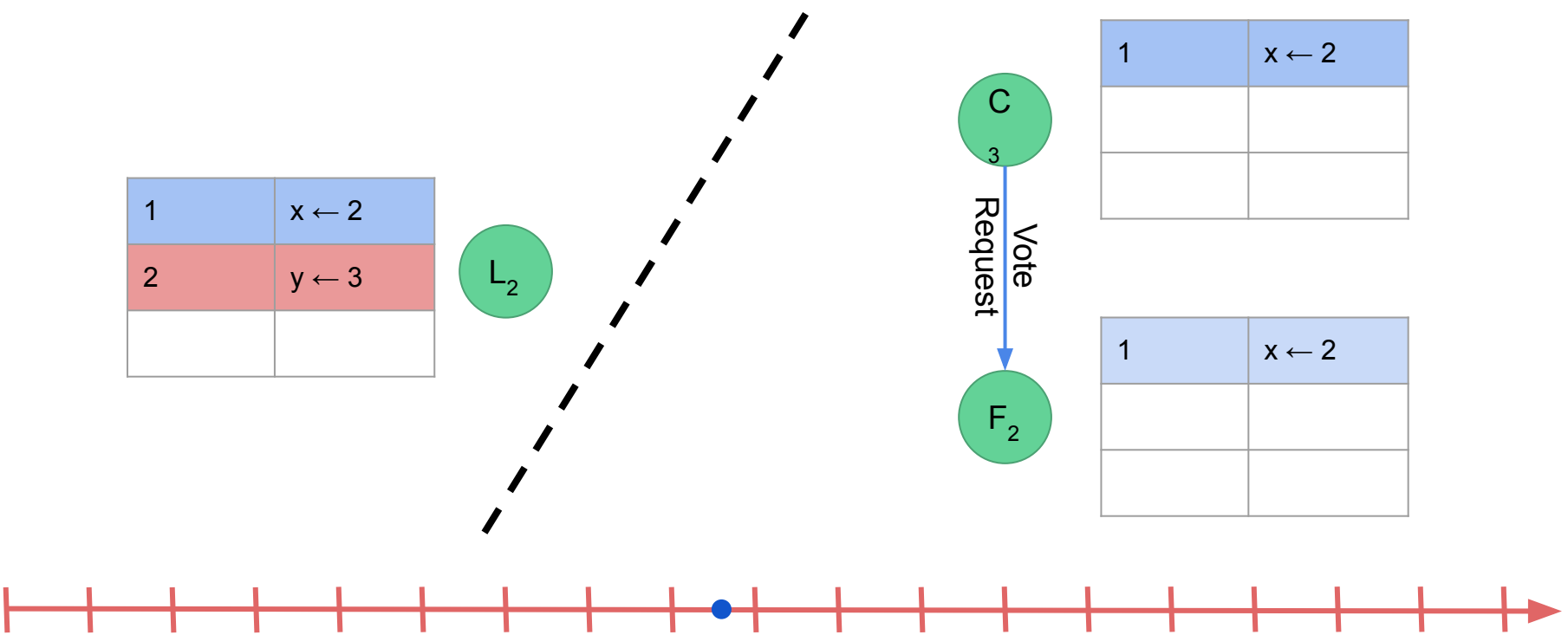
Raft: Network Partition



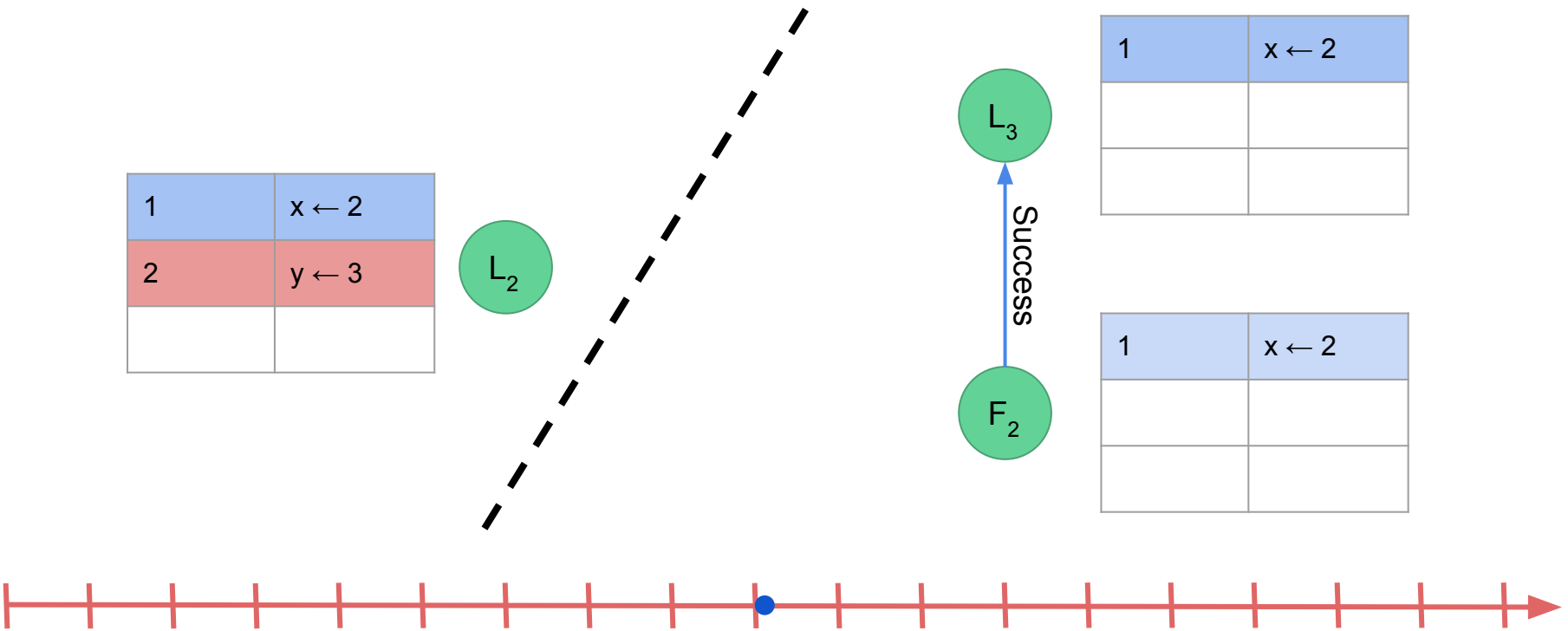
Raft: Network Partition



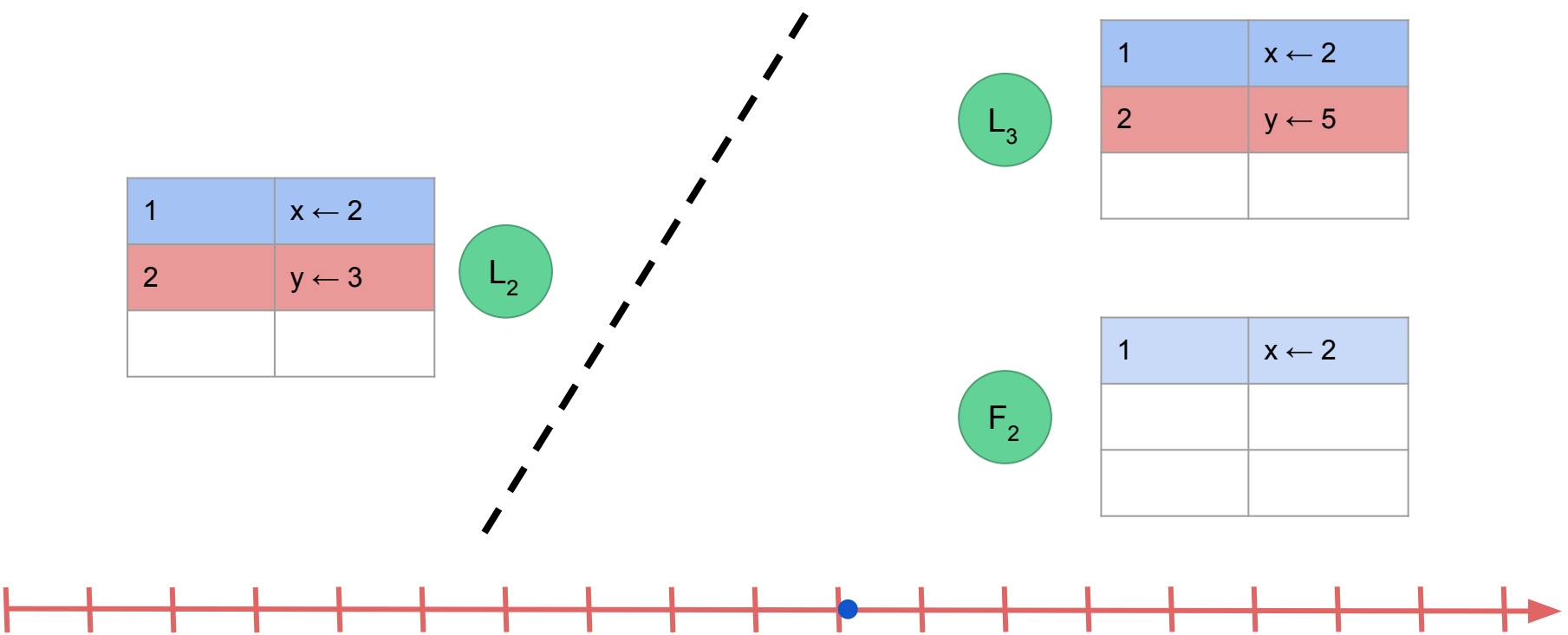
Raft: Network Partition



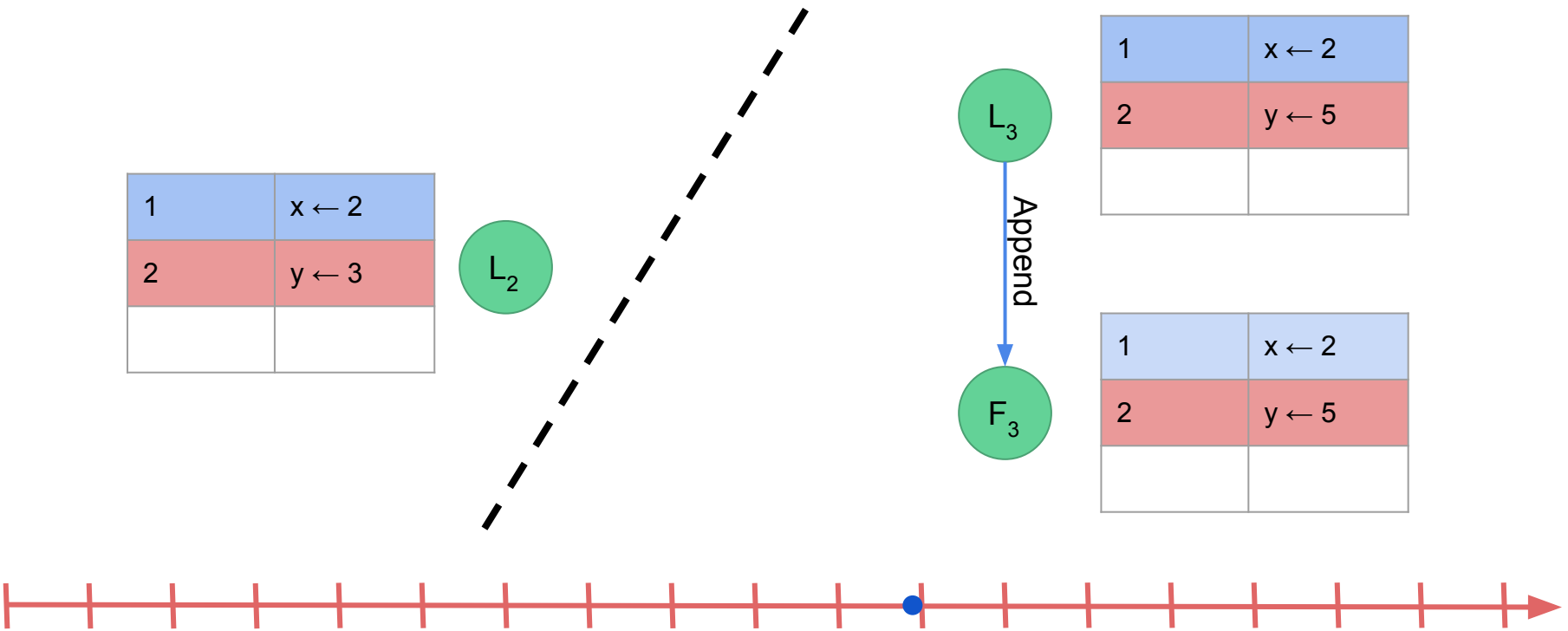
Raft: Network Partition



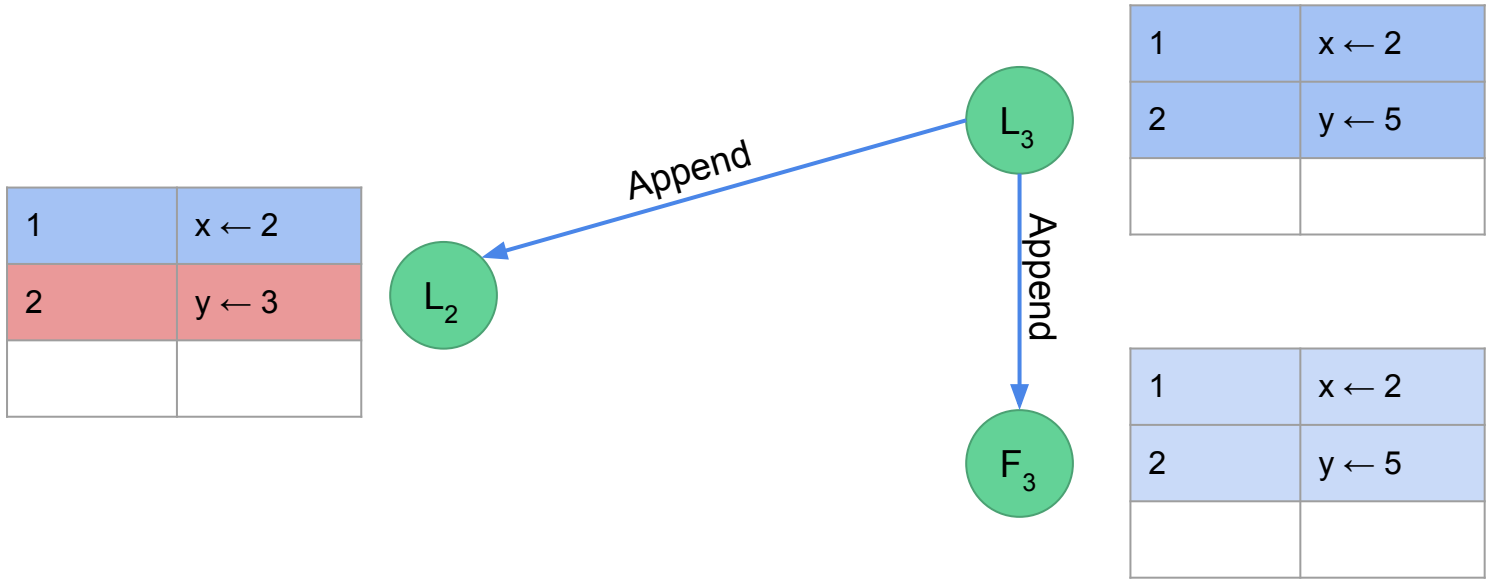
Raft: Network Partition



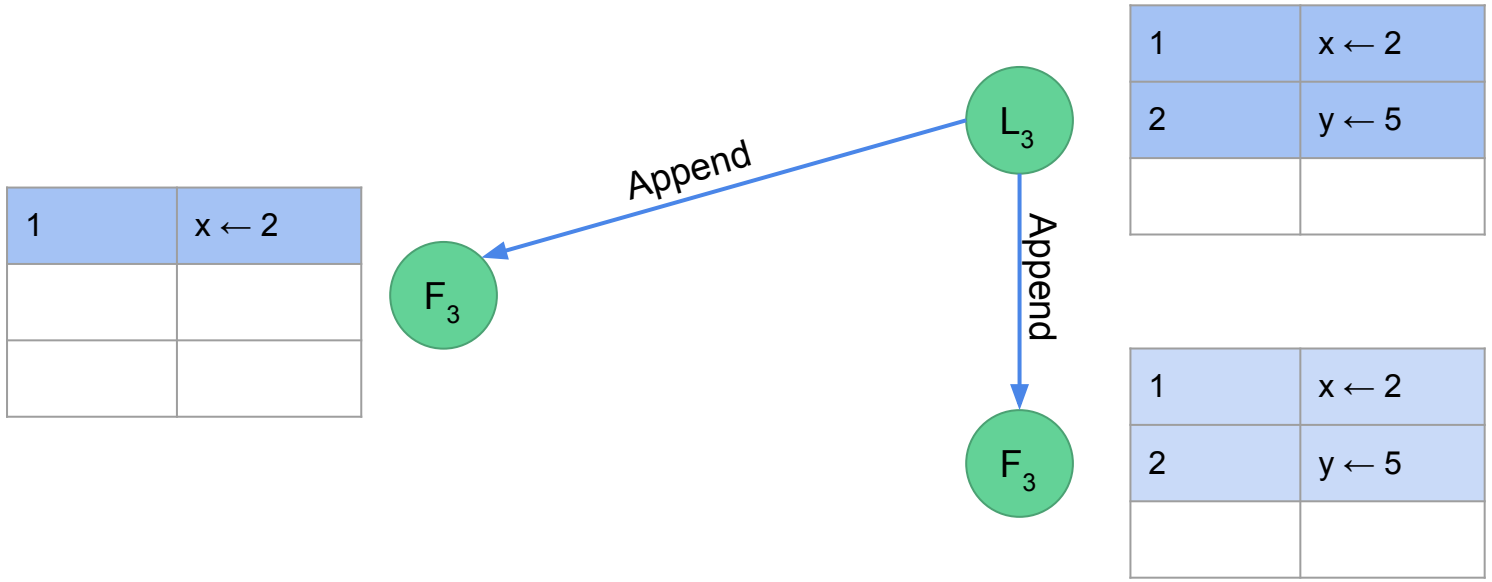
Raft: Network Partition



Raft: Network Partition



Raft: Network Partition



Raft: Network Partition