etcd: status updates

Kubernetes Meetup (Bay Area)
24 August 2016

Gyu-Ho Lee
CoreOS

Agenda

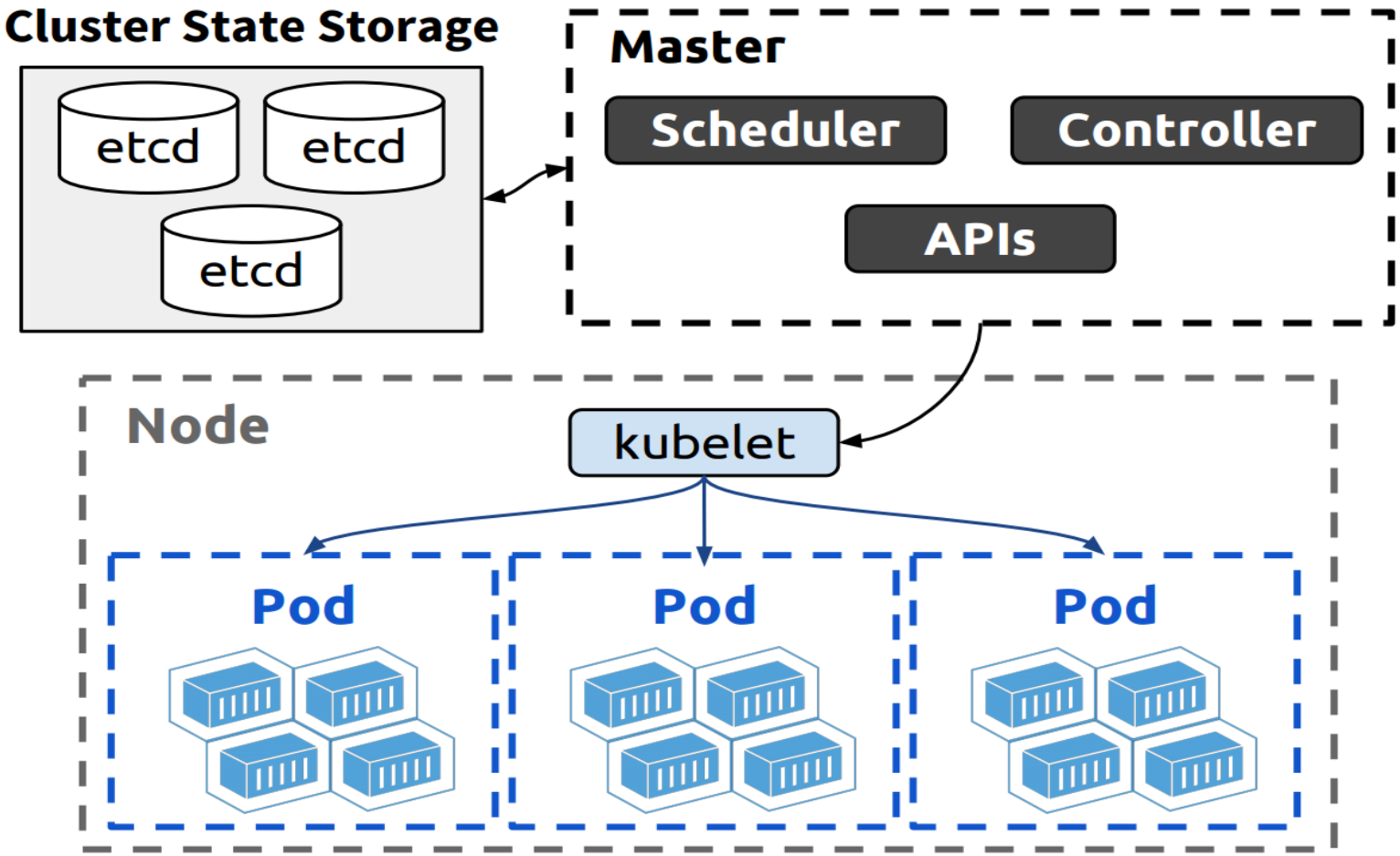
- What is etcd
- Current status (v3.0.x)
- New features (v3.1+)
- Q/A

What is etcd

- Distributed key-value store
- Open source github.com/coreos/etcd (https://github.com/coreos/etcd) (~ June 2013)
- Still new, compared to ZooKeeper (~ May 2008)

etcd + Kubernetes

- Kubernetes (https://github.com/kubernetes/kubernetes) manages cluster states in etcd (pkg/storage)



etcd API

```
cli.Put(ctx, "foo", "bar", Lease)
cli.Get(ctx, "foo")
cli.Delete(ctx, "foo")

// Transaction
kvc.Txn(ctx).
 If(clientv3.Compare(clientv3.Value("key"), ">", "abc")). // txn value comparisons are lexical
 Then(clientv3.OpPut("key", "XYZ")).                      // this runs, since 'xyz' > 'abc'
 Else(clientv3.OpPut("key", "ABC")).
 Commit()

// Watch for updates on key
ch := cli.Watch(ctx, "foo")
for res := range ch {}

// Distributed locks
mu := concurrency.NewMutex(cli, "foo")
mu.Lock()
mu.Unlock()
```

Current status (v3.0.x)

- Released in June 30, 2016
- Latest v3.0.6

Current status (v3.0.x)

(Reasonably) fast

- Write QPS 33K (vs. 3K with v2)
- Linearizable Read QPS 43K
- Serializable Read QPS 93K (vs. 45K with v2)

v3.1 will be even faster

Current status (v3.0.x)

(Super) stable

Extensive testing (unit, integration, end-to-end)

+12,000 failure injections per day

+3.5M injected for etcd v3

- kill members, leader
- network partition
- slow network
- fail points

Upcoming releases

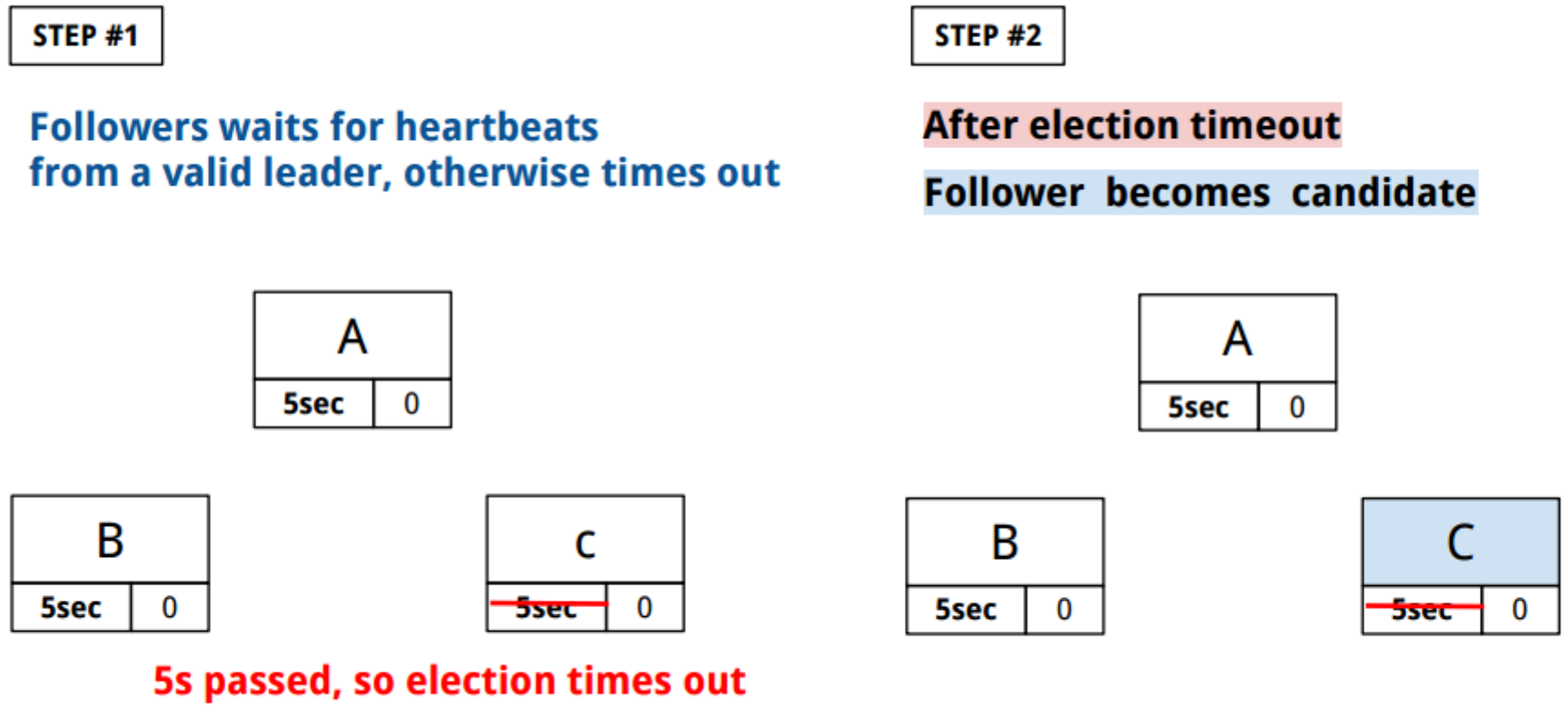
- v3.1-beta in mid-September
- GA in October

What's new in v3.1+?

Raft 101

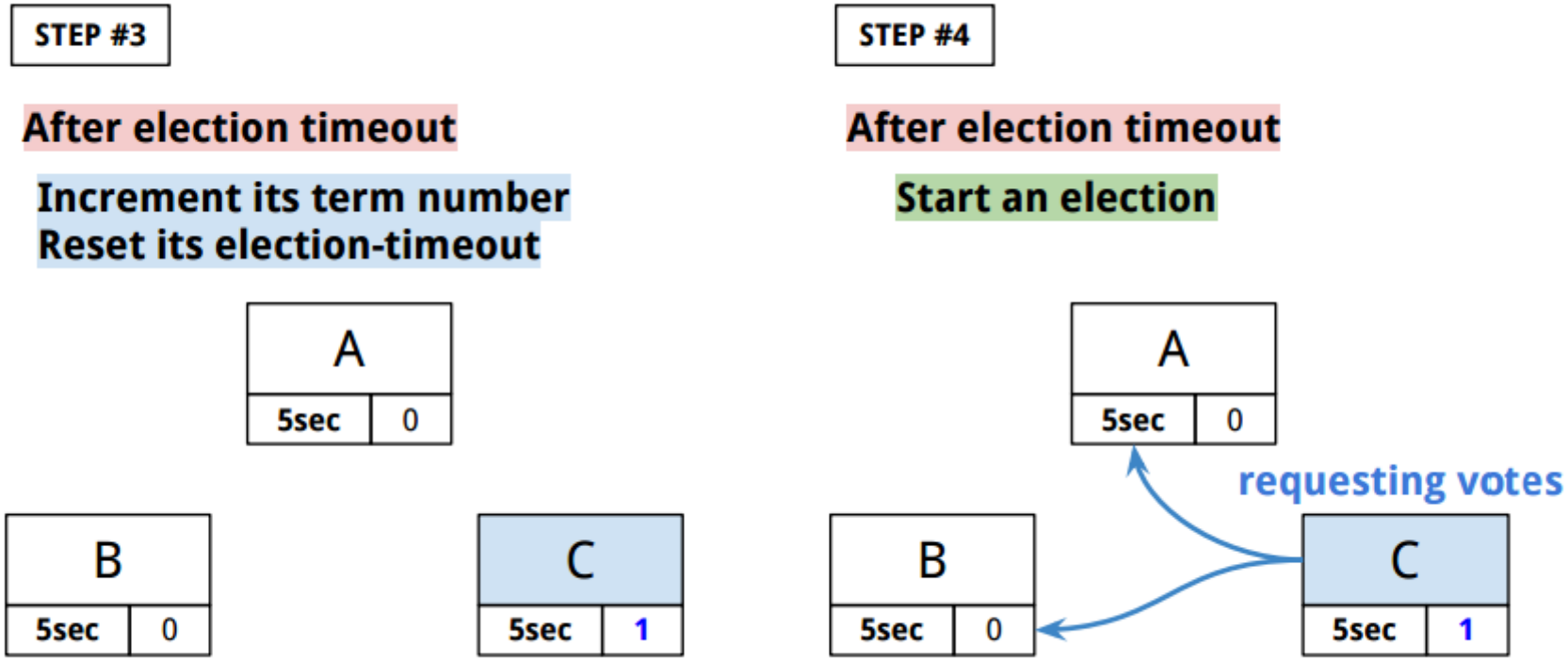
Raft leader election

- Follower becomes candidate if there's no leader within election-timeout



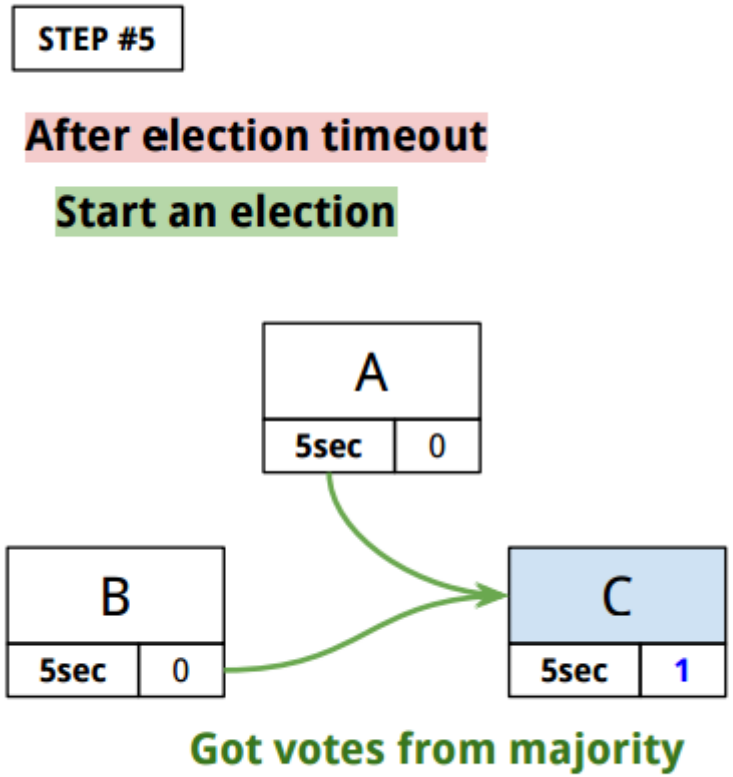
Raft leader election

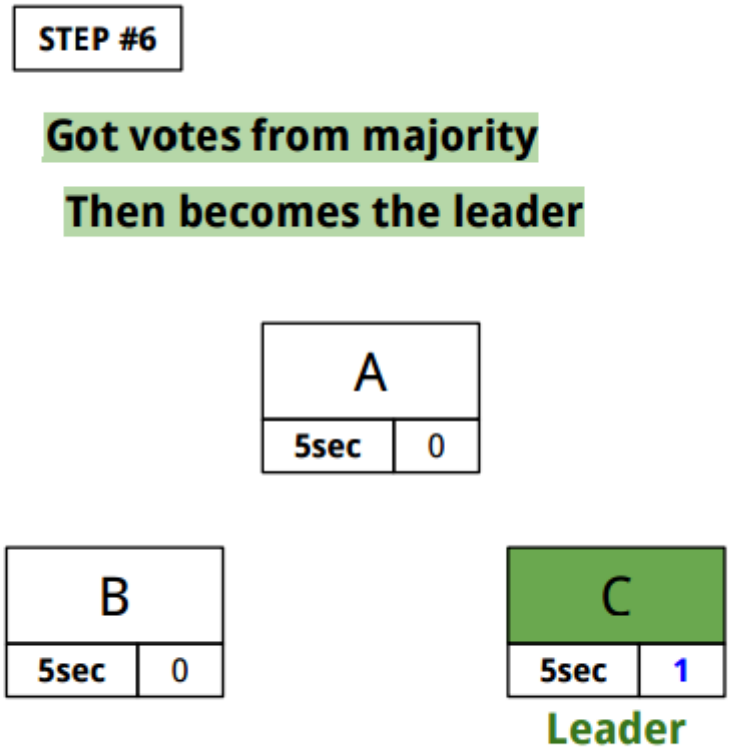
Follower starts election



Raft leader election

Got votes from majority, then becomes leader

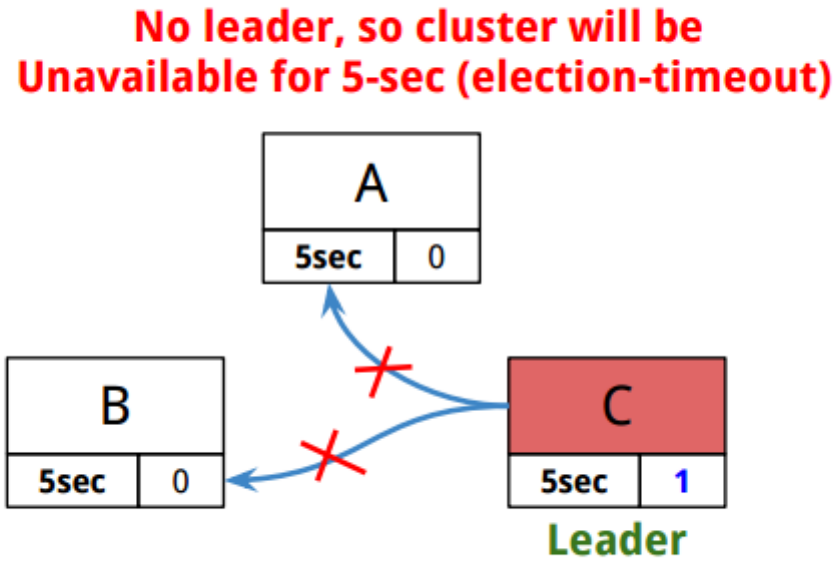




What if leader goes down?

No leader, so cluster becomes **unavailable**

Until another node election-times out, wins an election



What if leader goes down?

Demo: play.etcd.io (http://play.etcd.io)

Leadership transfer

This brief unavailability can be avoided with **leadership transfer**

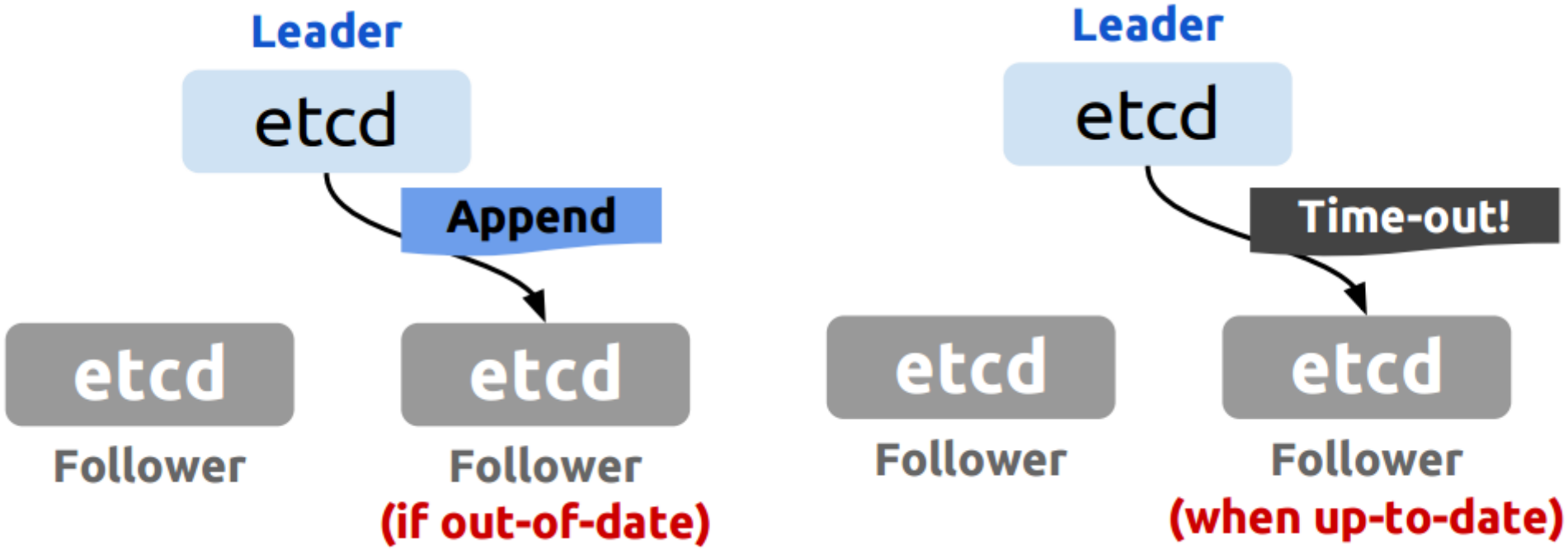
(Raft §3.10 Leadership transfer extension, p.28)

Use case

- Rolling upgrade
- Maintenance

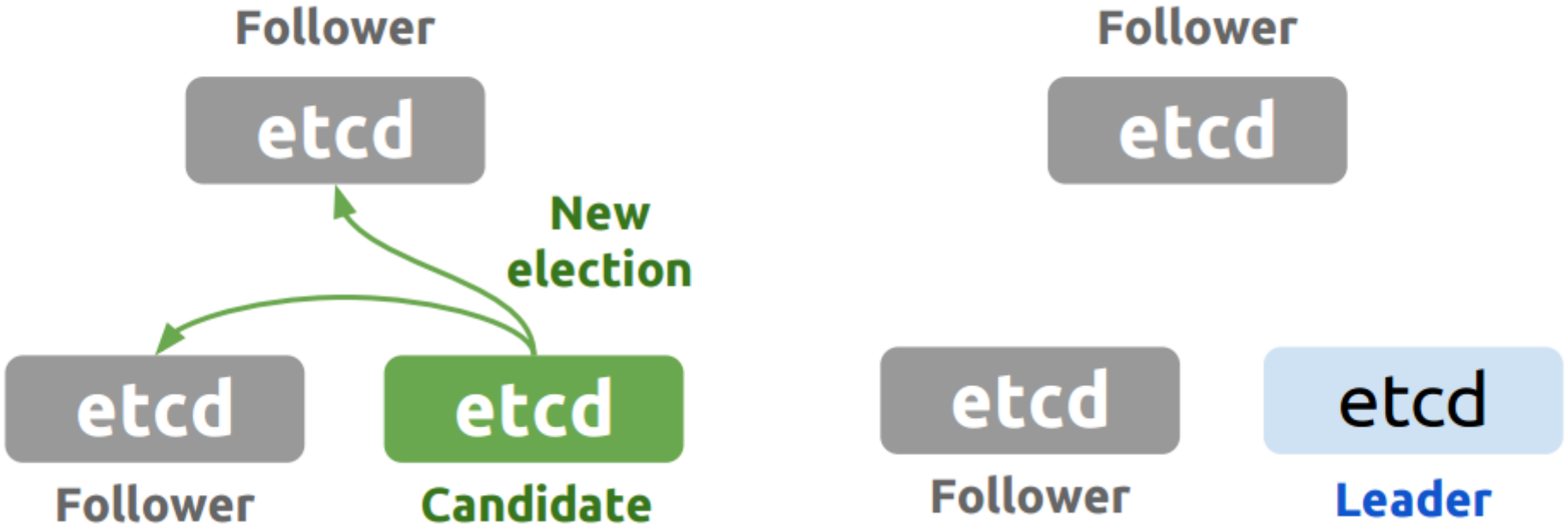
Leadership transfer

- Leader sends **MsgAppend** if transferee log is out-of-date
- Then leader sends **MsgTimeoutNow** to transferee
- **Fast-forward** transferee's clock to force **election-time-out**



Leadership transfer

- Now regular Raft leader election
- Transferee(follower) election-times-out, becomes candidate
- And gets elected as leader



Leadership transfer

etcd leader transfer is automatic (all in server side)

- etcd leader receives **SIGINT**, **SIGTERM**
- etcd leader **chooses one most stable follower as transferee**
- etcd leader **transfers its leadership** BEFORE **shutdown**

Minimum downtime +/-100ms (vs. 1~2s without leadership transfer)

Leadership transfer

- Not *"true" zero-time* leadership transfer
- Brief leader-lost while campaigning

We will make it better!

--prev-kv flag

```
etcdctl put foo bar1

etcdctl put --prev-kv foo bar2
OK
foo
bar1

etcdctl watch --prev-kv foo
foo
bar1
foo
bar2
```

Just one roundtrip to find current, previous value

No need to send another RPC to find previous value

Embedded etcd server

```
import "github.com/coreos/etcd/embed"

cfg := embed.NewConfig()
cfg.Dir = "default.etcd"

e, err := embed.StartEtcd(cfg)

<-e.Err()
e.Close()
```

etcd clients still connects via gRPC

Embedded client (https://github.com/coreos/etcd/issues/4709) planned for v3.2

Better performance

etcd v3.0 vs. **v3.1 (Go 1.7)**

- Write QPS 33K vs. **45K (+25% faster)**
- Linearizable Read QPS 43K vs. **55K (+20%)**
- Serializable Read QPS 93K vs. **110K (+15%)**

> Zookeeper v3.4.8 Write QPS *37K*

zetcd

etcd has better performance than Zookeeper

Why not serve Zookeeper requests with etcd?

Forward Zookeeper client requests to zetcd

And zetcd handles requests, data with etcd

Now we can run Kafka, Cassandra, etc. with etcd

Java client

- Led by community members; WeStudio, PPMoney, Twitter
- github.com/coreos/jetcd (https://github.com/coreos/jetcd)

Faster linearizable read (QGET)

- Linearizable read (QGET) requires quorum(majority) to agree on the value
- Serializable read doesn't go through consensus protocol, served locally

QGET provides stronger consistency, but slower

etcd v3.0 QGET QPS 40K <<< serializable-GET 100K

etcd v3.1 QGET QPS 100K == serializable-GET 100K

Faster linearizable read (QGET)

it is possible to bypass the Raft log for read-only queries and still preserve linearizability

(Raft §6.4 Processing read-only queries more efficiently, p.72)

- Leader records **current commit index** in **readIndex**
- Leader sends **readIndex** to followers
- For followers, **readIndex** is the largest commit index ever seen by any server
- Read-request within **readIndex** is now served locally with linearizability
- More efficient, avoids synchronous disk writes

gateway

kube-proxy-like etcd gateway

```
etcd gateway start --endpoints=infra.example.com --listen-addr=127.0.0.1:23790
etcdctl put foo bar --endpoints=127.0.0.1:23790
```

Static endpoint for client requests

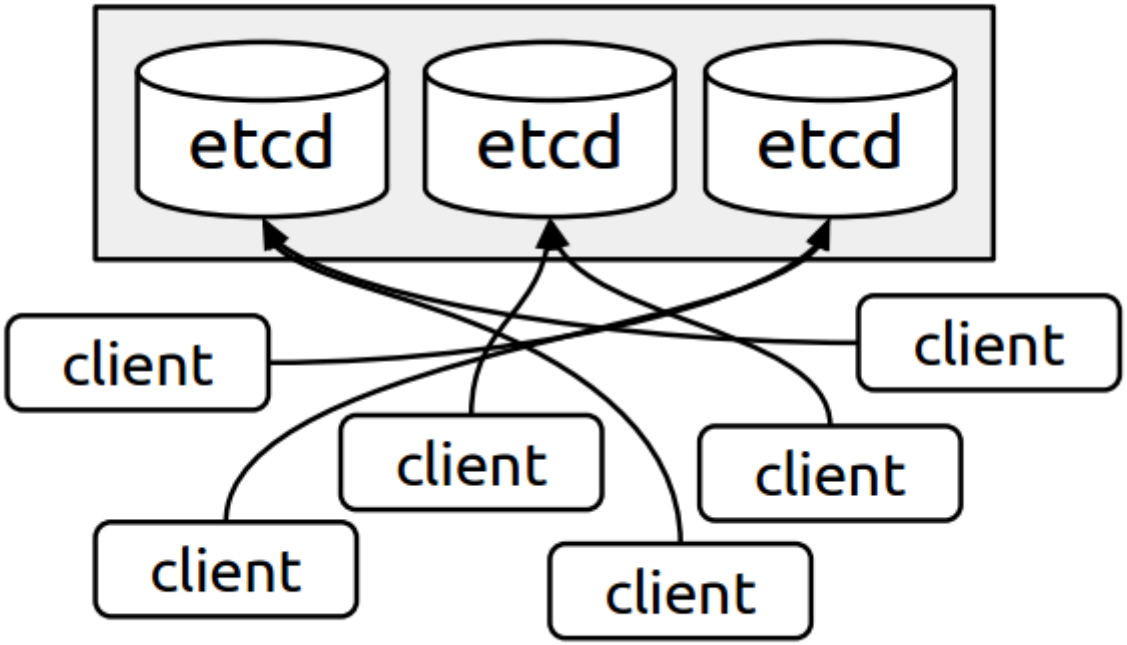
```
# even after re-configure infra.example.com
# clients still connects via same endpoint
etcdctl put foo bar --endpoints=127.0.0.1:23790
```

Not built for performance improvement

Proxy

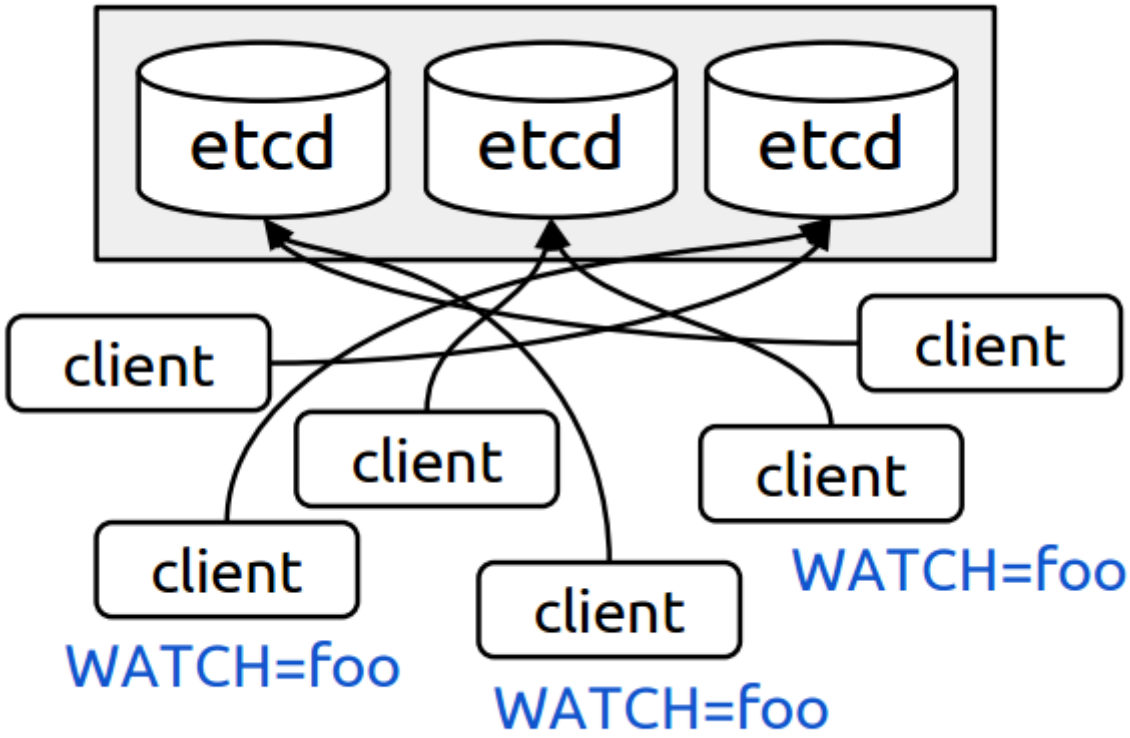
- etcd v2 proxy is just HTTP proxy (reverse proxy)
- etcd v2 proxy doesn't understand gRPC

etcd v3 clients still talk directly to etcd cluster



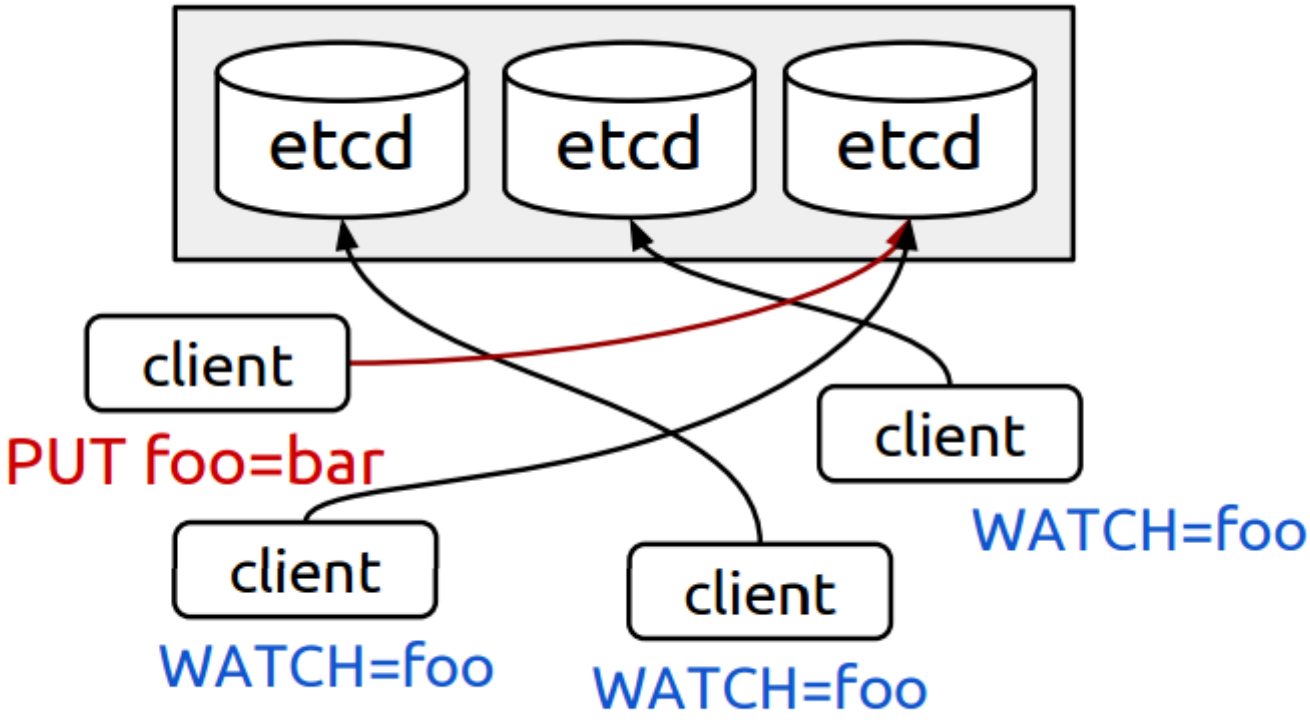
Proxy

same WATCH requests



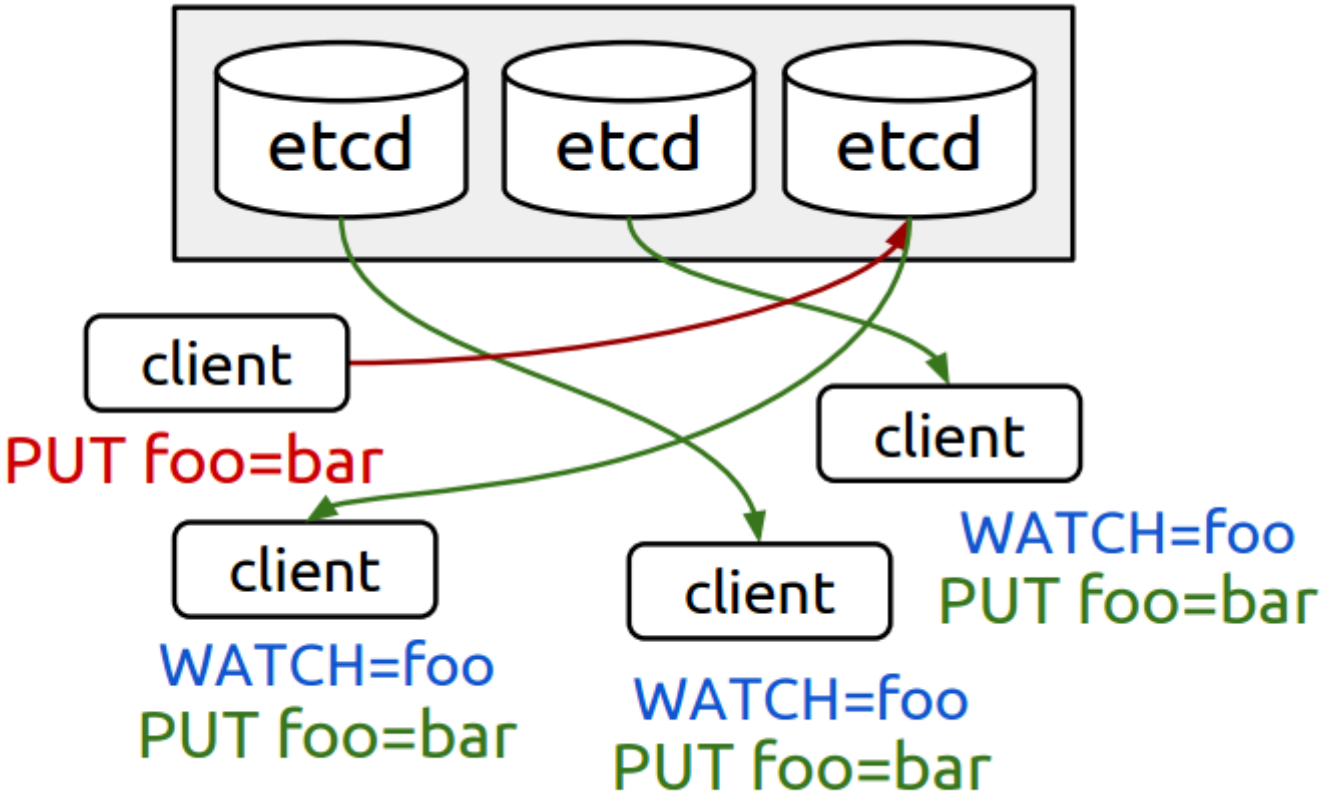
Proxy

same WATCH requests



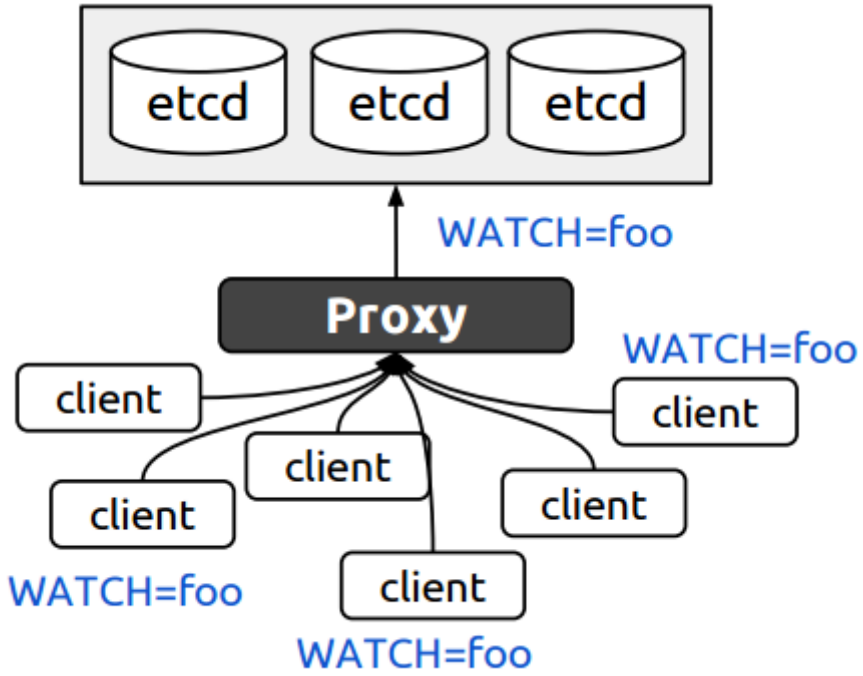
Proxy

3 different events are triggered for the same request **(This is inefficient!)**



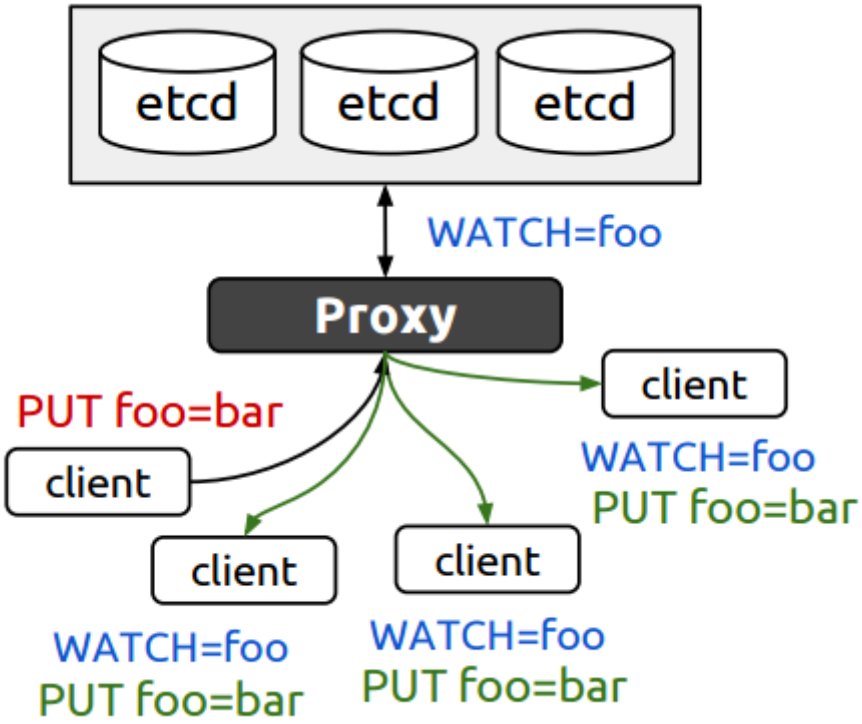
Proxy

etcd v3 proxy solves this problem by **coalescing same requests**



Proxy

Merged WATCH request



Proxy does all the hard work!

Proxy

so proxies allow a significant increase in the number of clients

A proxy cache can reduce read traffic by at most the mean amount of read-sharing

(Google Chubby paper §3.1 Proxies, p.10)

But etcd does not have much proxy use cases yet!

- etcd proxy is still in design process
- need more feedback, use case study

Thank you

Gyu-Ho Lee
CoreOS
gyu_ho.lee@coreos.com (mailto:gyu_ho.lee@coreos.com)
https://github.com/coreos/etcd (https://github.com/coreos/etcd)