The good, the bad, and the ugly of Apache ZooKeeper

Flavio Junqueira

Apache ZooKeeper Committer, PMC
Confluent

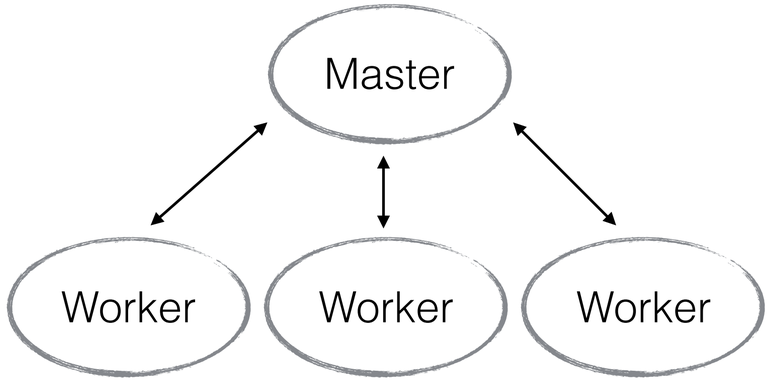
fpj@confluent.io
twitter: @fpjunqueira

What’s ZooKeeper?

Building resilient distributed systems

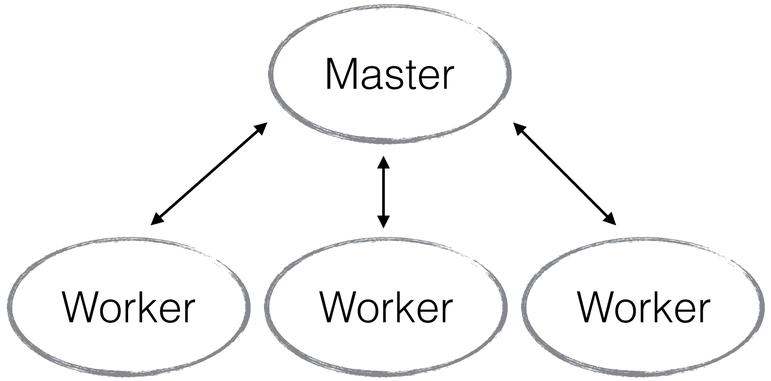
Source: Ashish via Flickr

Leader election

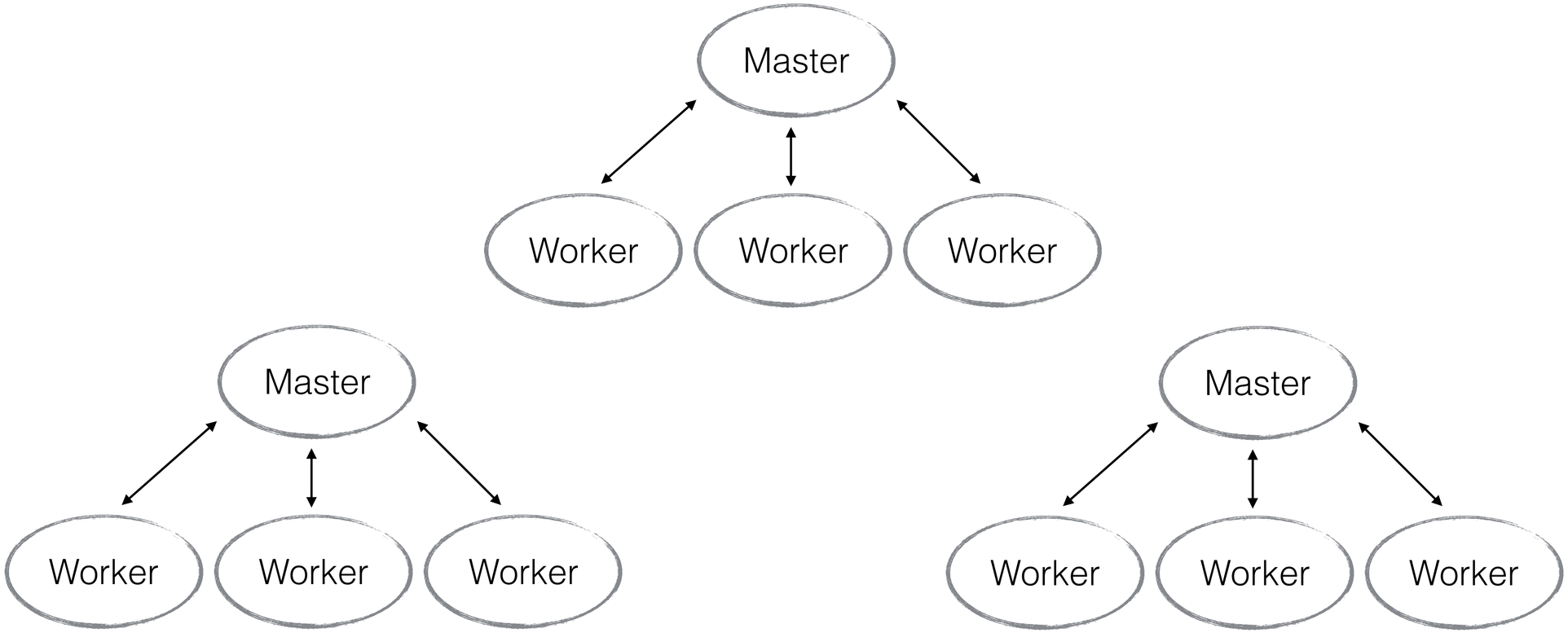


Leader election

E.g., replication



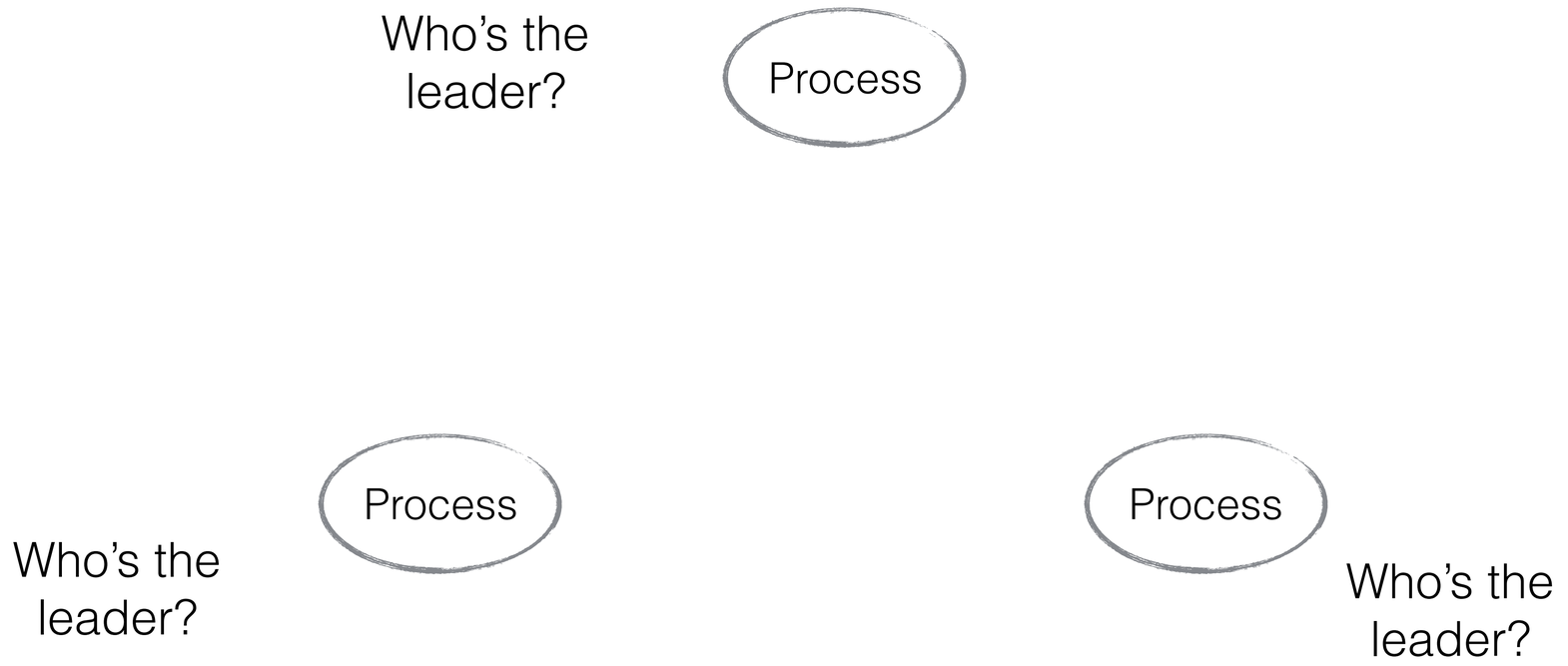
Leader election



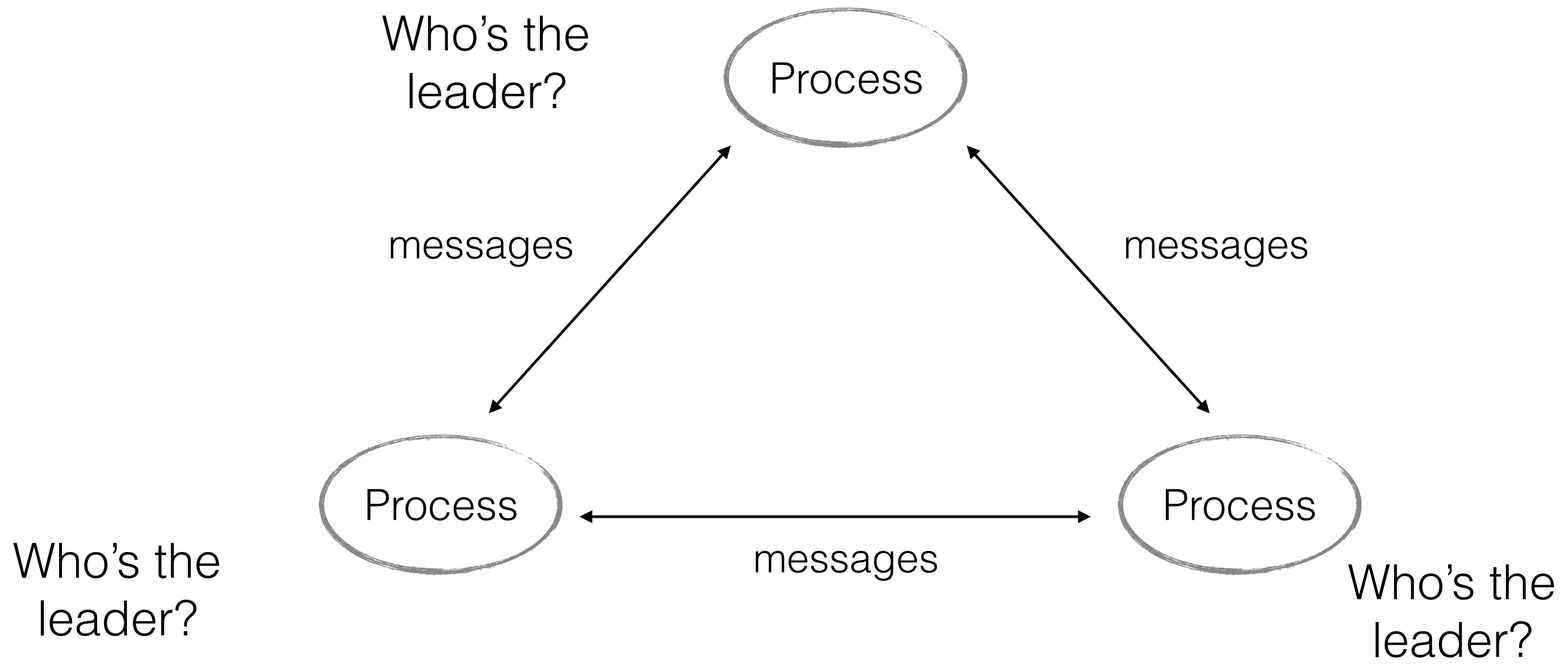
L E A D E R S H I P

Source: Ki Young Lee via Flickr

Leader election

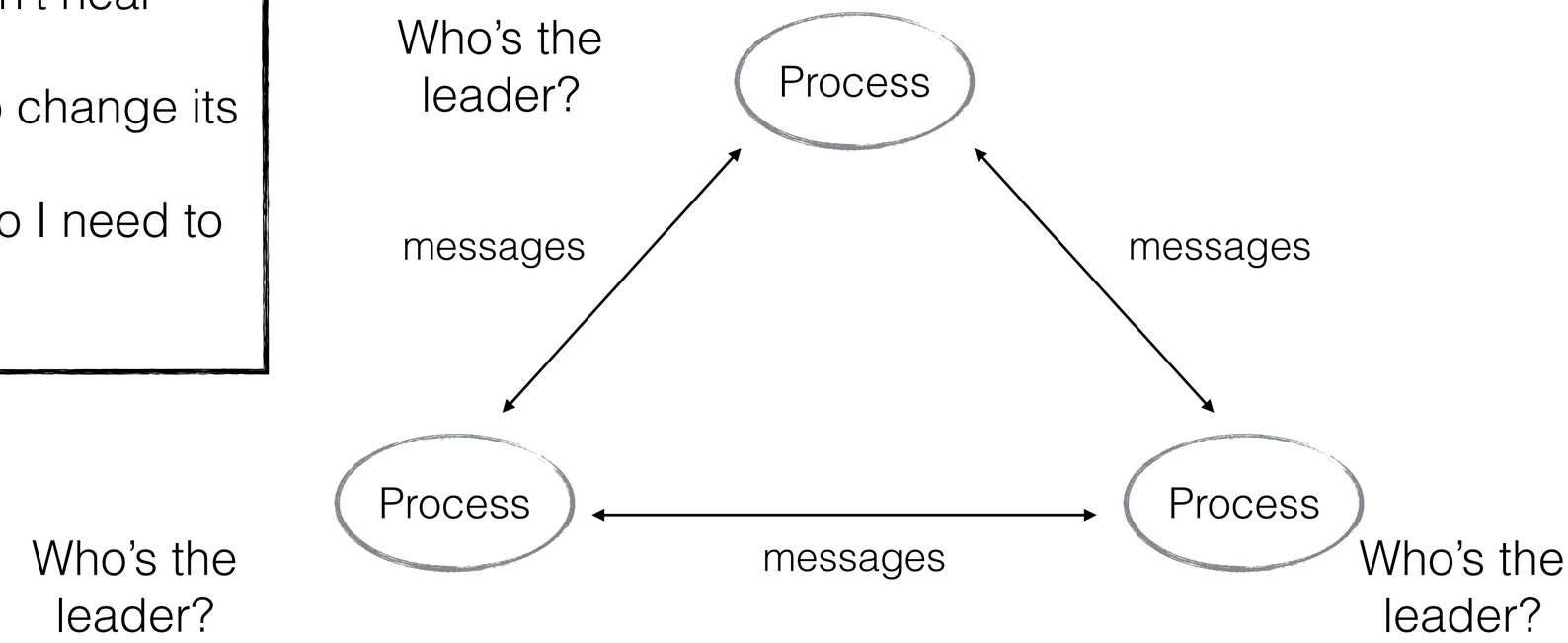


Leader election

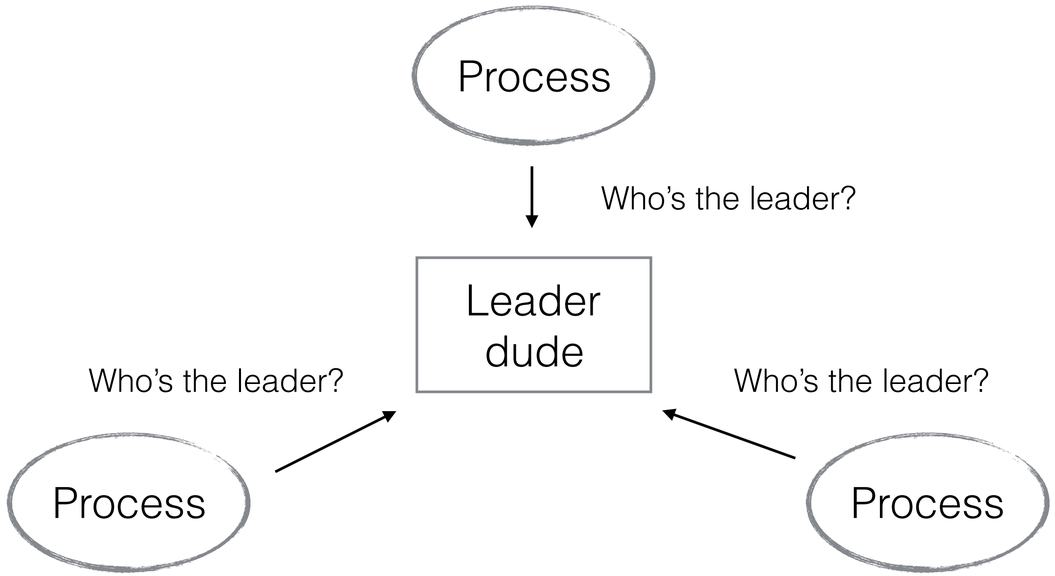


Leader election

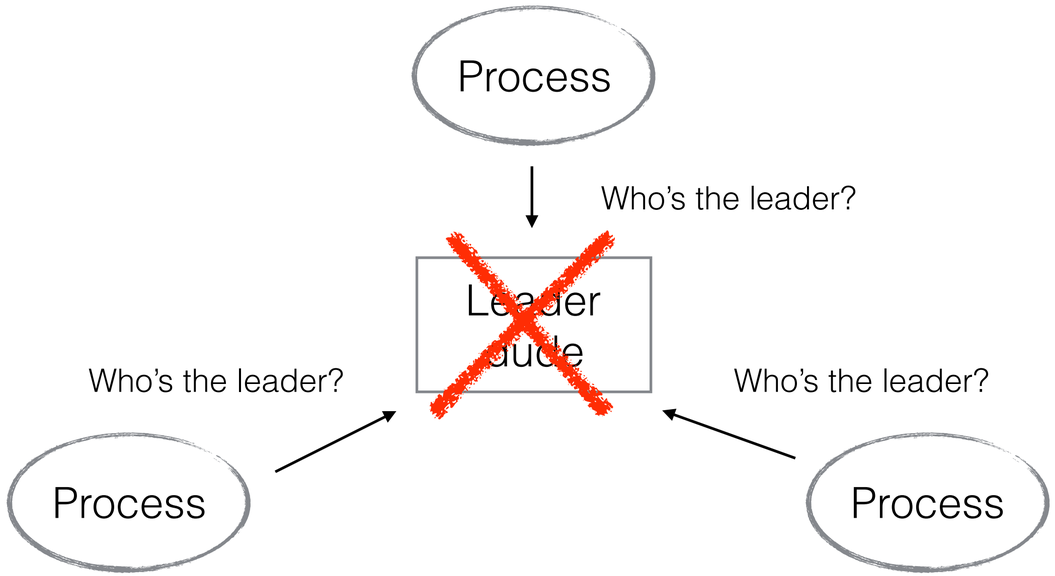
- What if a process doesn't hear from another?
- A process is allowed to change its vote?
- For how many rounds to I need to exchange messages?
- Is this even correct?



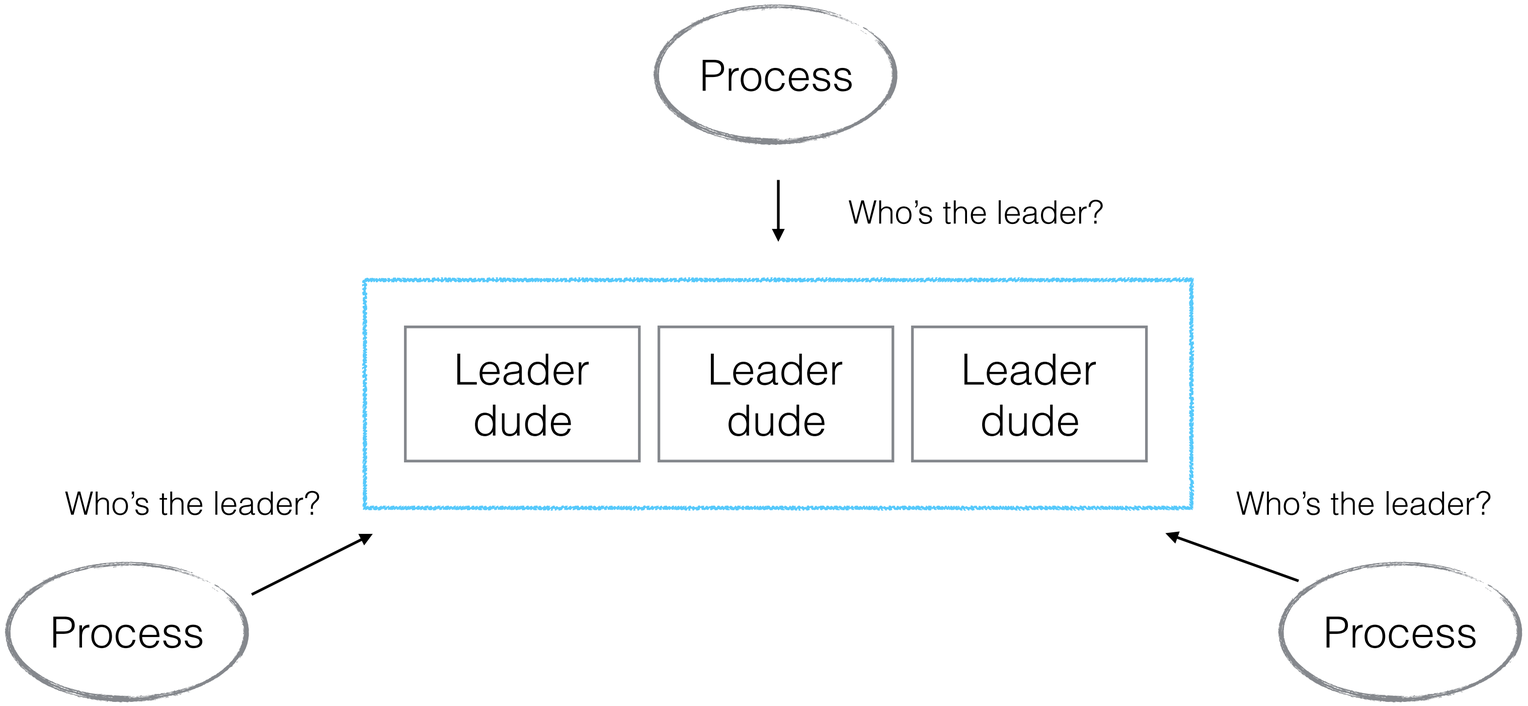
Leader election



Leader election

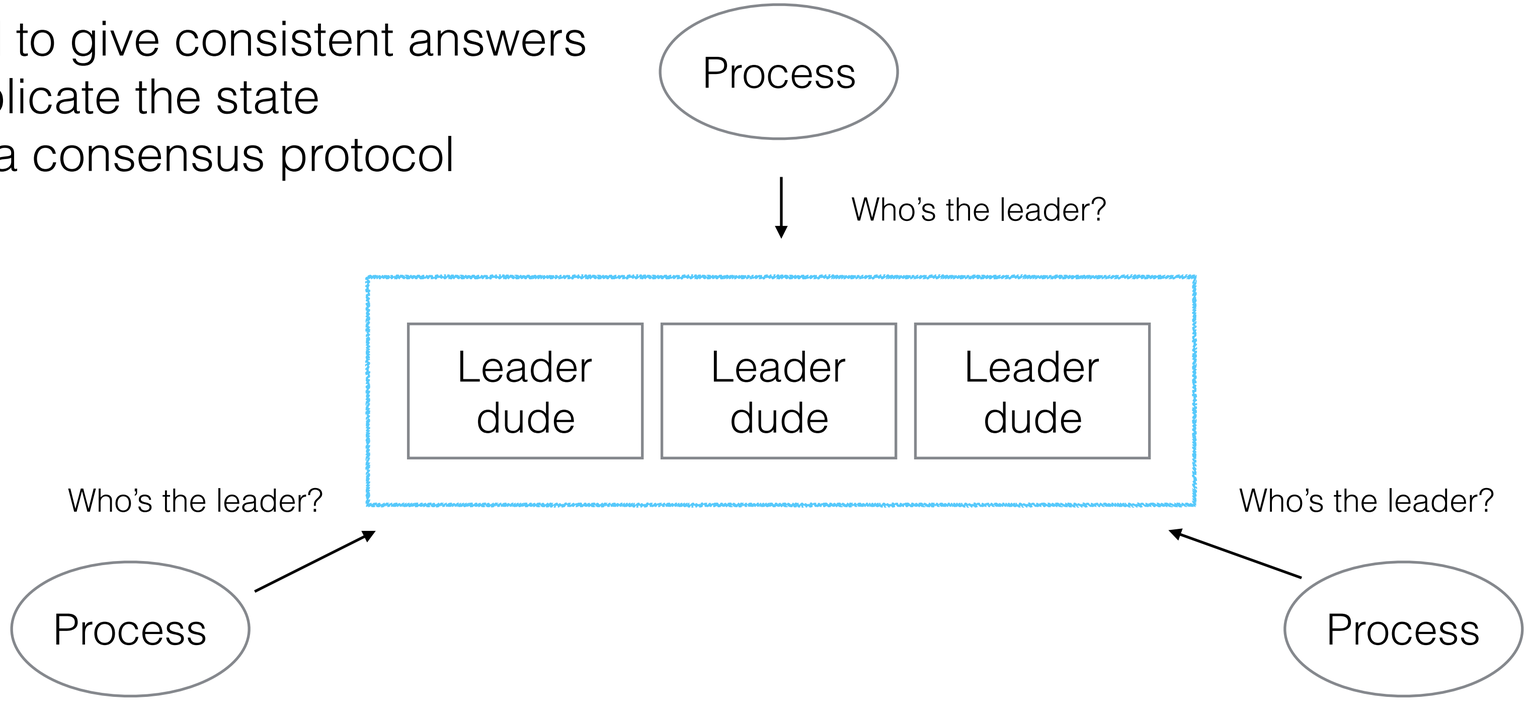


Leader election



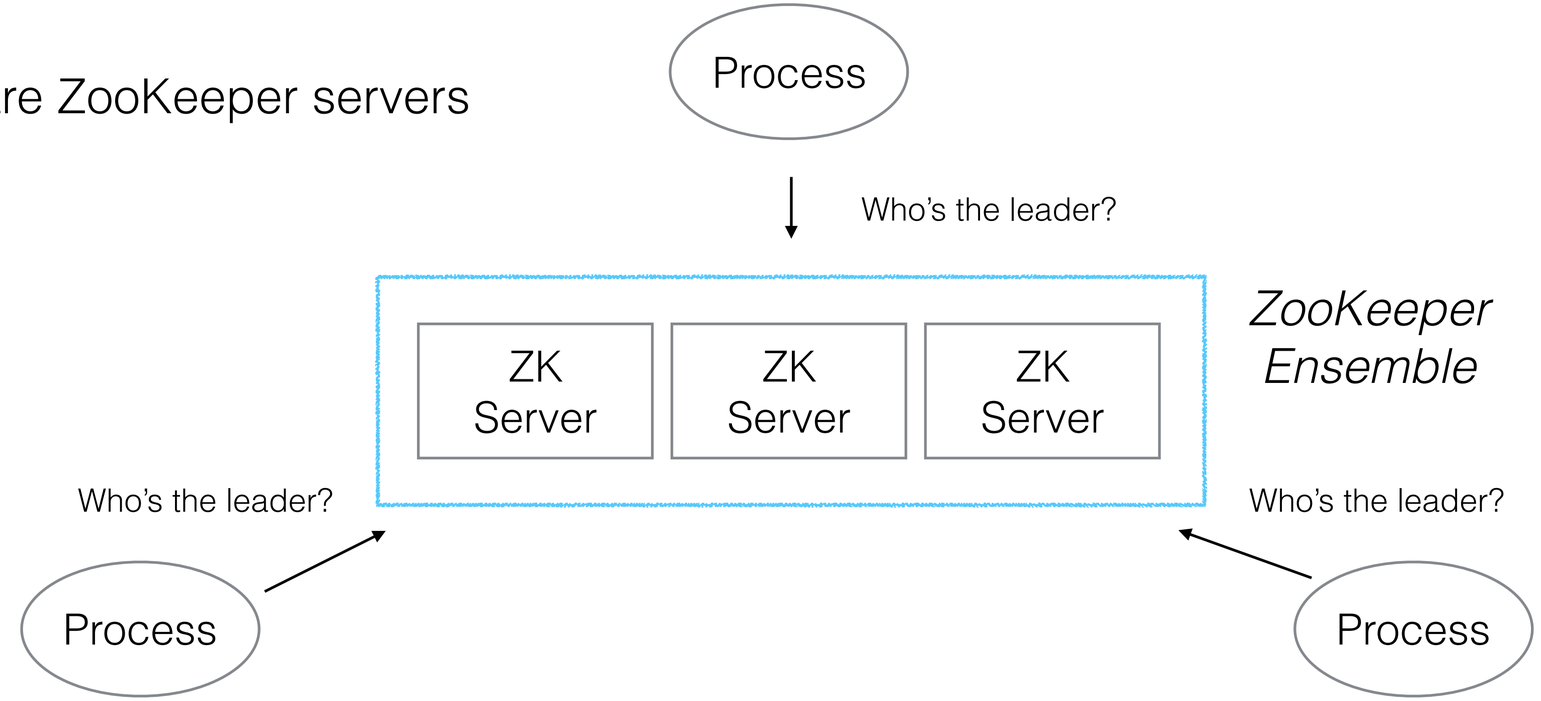
Leader election

- Replicas need to give consistent answers
- Protocol to replicate the state
- … essentially a consensus protocol



Leader election

- The dudes are ZooKeeper servers



… and more

- Membership
- Synchronization primitives
  - locks
  - barriers
  - atomic counters
  - CAS
- Configuration metadata

How does ZooKeeper work?

Basics

- Hierarchy of simple files called *znodes*
  - Persistent, ephemeral, sequential
- File-system-like API
  - Writes: `create`, `delete`, `setData`
  - Reads: `exists`, `getChildren`, `getData`
- Watches
  - Enables clients to observe changes to znodes
  - One shot, not a subscription

Recipes

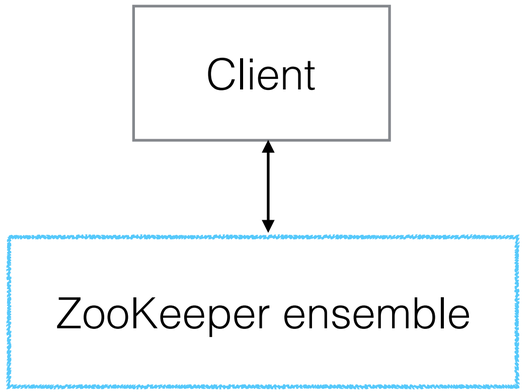
- ZooKeeper doesn’t expose primitives explicitly
- Primitives implemented using *recipes*
  - Simple algorithms based on the ZooKeeper API
  - Many have been implemented and battle-tested over time

Leader election with ZooKeeper

- Each process
  1. Creates an ephemeral znode with path `/election`
  2. If create call succeeds, then lead
  3. Otherwise, watch `/election`

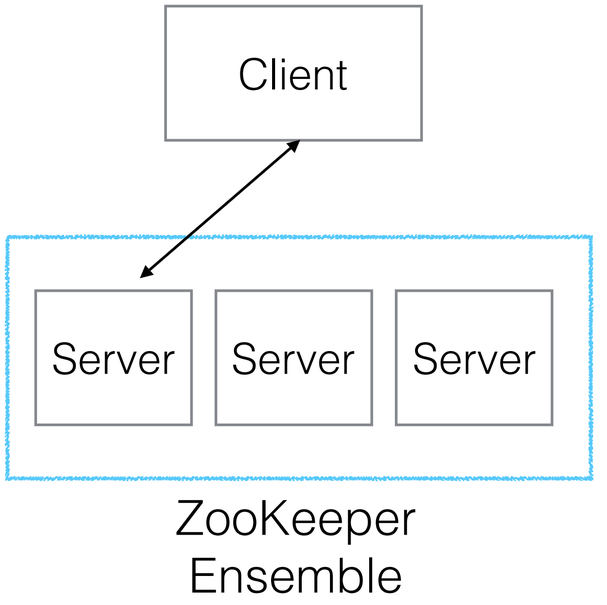
Sessions and Ephemerals

- Sessions
  - Abstraction of connection to the ensemble
  - Sessions start on a single server in an ensemble
  - Sessions can move to different servers over time
- The ensemble leader expires sessions using a timeout scheme
- An ephemeral znode is associated to a session
  - If session expires, then ephemerals automatically deleted



Sessions and Ephemerals

- Sessions
  - Abstraction of connection to the ensemble
  - Sessions start on a single server in an ensemble
  - Sessions can move to different servers over time
- The ensemble leader expires sessions using a timeout scheme
- An ephemeral znode is associated to a session
  - If session expires, then ephemerals automatically deleted



… but could we have done it ourselves?

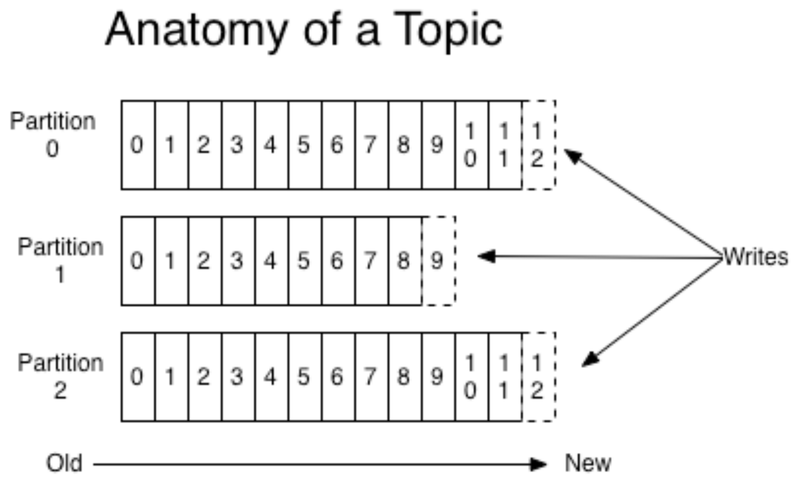
Implement your own screw driver…

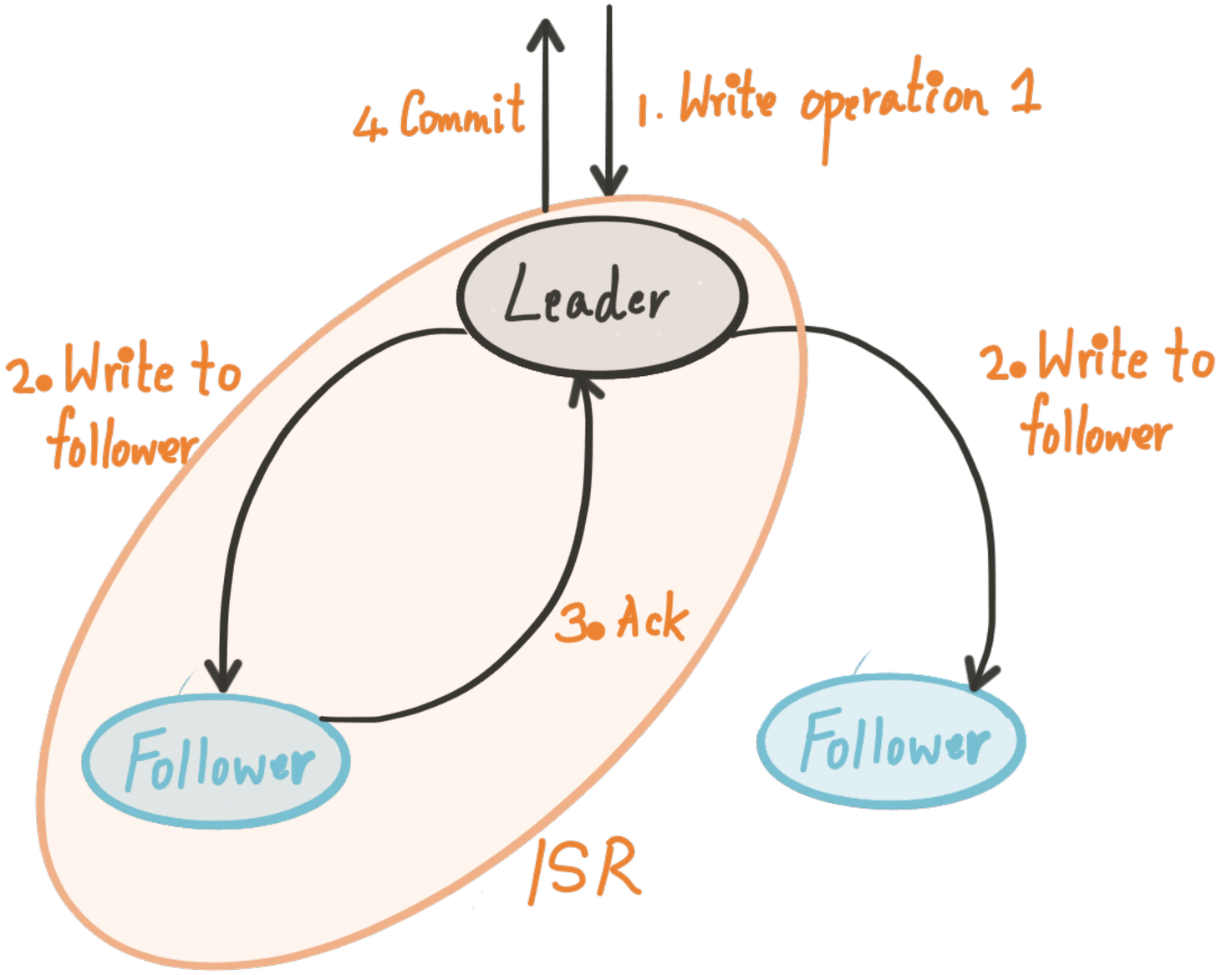
Source: Florinda Chan via Flickr

Use case: Apache Kafka Replication

Kafka basics

- Pub-sub messaging
  - Implemented as a distributed commit log
- Topics
  - App-specific element of organization
  - *E.g.*, user clicks, search queries, likes, friendship connections, tweets
- Topics are sharded into partitions
  - Each partition has a replica set

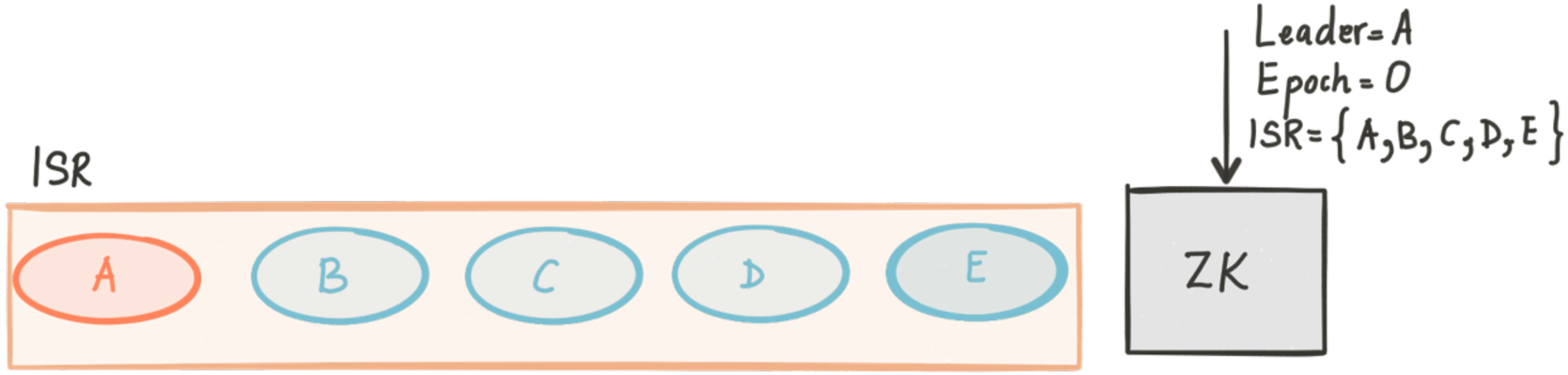



4. Commit
1. Write operation 1
Leader
2. Write to follower
2. Write to follower
3. Ack
Follower
Follower
ISR

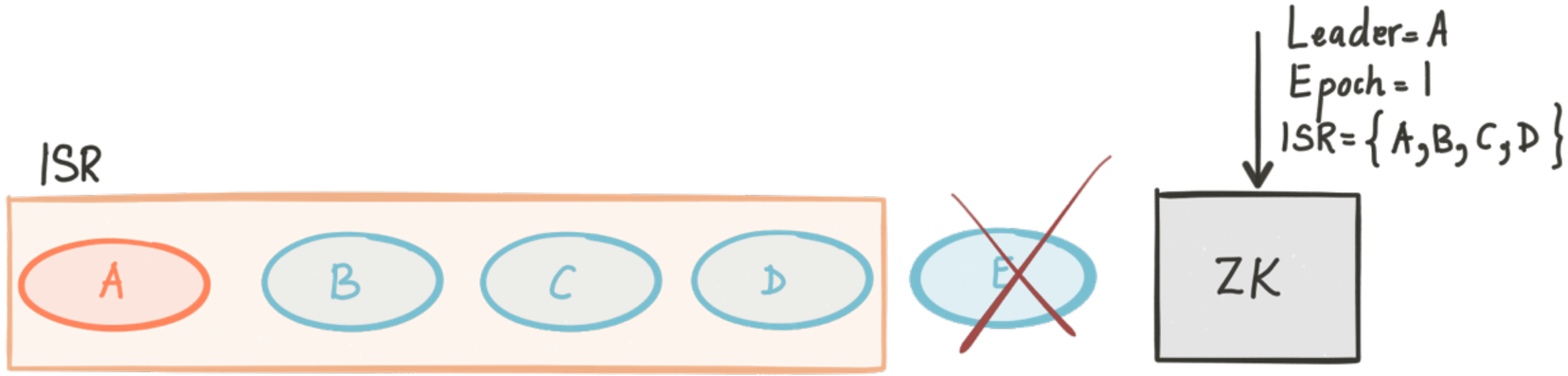
ZooKeeper

- Stores the metadata of replica groups
- Leadership and in-sync replicas

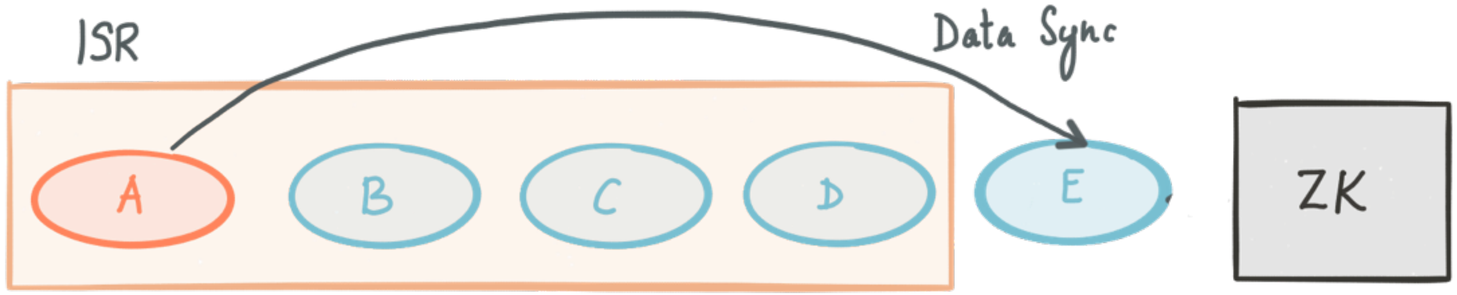
Partition replication and ZooKeeper



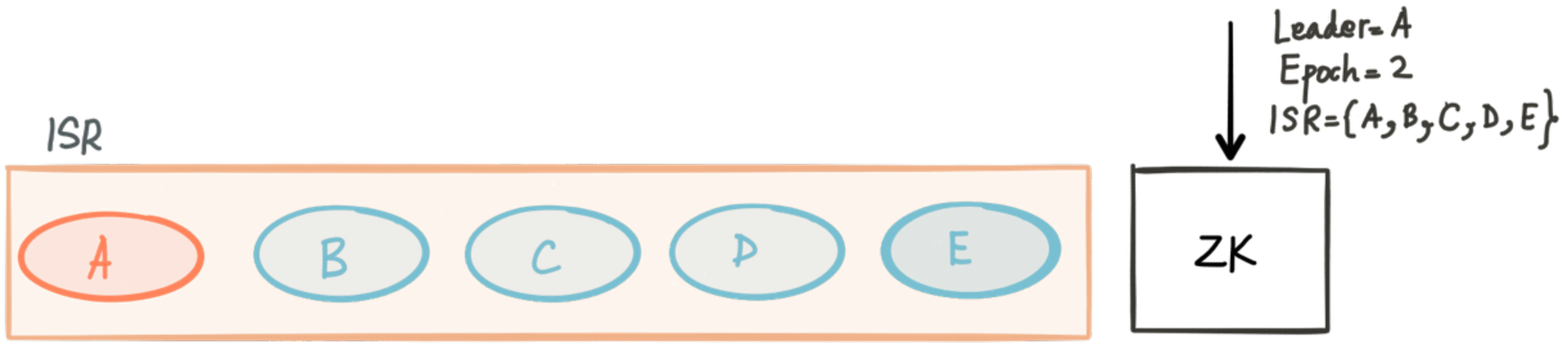
Partition replication and ZooKeeper



Partition replication and ZooKeeper



Partition replication and ZooKeeper



ZooKeeper

- Stores the metadata of replica groups
- Leadership and in-sync replicas
- Advantages
  - **Source of truth**: Precise information about the replica group
  - **Flexibility**: No need to rely on majority quorums

But why use a replicated system to build another replicated system?

Rationale

- Write throughput to ZooKeeper is bounded
  - Lower write throughput with more replicas
  - … higher read throughput though
- Management of replica groups
  - Easier with a component like ZooKeeper around

Other examples

- Apache HBase
  - Large-scale key-value store
- Apache BookKeeper
  - High-performance, distributed logging

The project

Apache ZooKeeper

- Apache top-level project
  - Since 2010
- Committers: 15
  - Across 9 different companies
- PMC members: 9
  - Across 8 different companies

http://zookeeper.apache.org

confluent

Good, bad, and ugly

- **Good**
  - What made the project successful, what users like
- **Bad**
  - What users don’t like
- **Ugly**
  - What we devs of ZooKeeper don’t like

The good

- See previous slides…
- Simple API
- It works
- Battle tested

The bad

- Dependency-phobia
- Server footprint
  - Requires additional hardware (or VMs)
- Hard to embed
  - Making operations harder
- Fat client
- Dedicated device for the txn log

The ugly

- Requests under disconnection
  - No really good way to tell if request has been executed
- Multi-tenancy
  - Security and performance isolation: ok but not stelar

Wrap up

Apache ZooKeeper

- Distributed coordination
  - Master election, membership, metadata, locks, barriers, etc
- Battle-tested in production across a number of companies
- Consider contributing
  - Subscribe to `(user|dev)@zookeeper.apache.org`
  - Check `http://zookeeper.apache.org`

We're hiring

https://jobs.lever.co/confluent