Gezähmte Wildnis:
Einführung in Distributed Coordination mit Apache ZooKeeper

Hartmut Lang, Ericsson

Java Forum Stuttgart, Juli 2012

Hartmut Lang

Senior Software Developer
Solution Architect

Network Management
and Customer Solutions
for the Telecom Industry

http://nighthacks.com/roller/jag/

The Eight Fallacies of Distributed Computing

Peter Deutsch

Essentially everyone, when they first build a distributed application, makes the following eight assumptions. All prove to be false in the long run and all cause *big* trouble and *painful* learning experiences.

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

For more details, read the article by Arnon Rotem-Gal-Oz

popular fallacy = weit verbreiteter Irrtum

The Eight Fallacies of Distributed Computing

Peter Deutsch

Essentially everyone, when they first build a distributed application, makes the following eight assumptions. All prove to be false in the long run and cause lots of trouble and *painful* learning experiences.

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

For more details, read the article by Arnon Rotem-Gal-Oz

„I know when my peers are up"

Add distribution two weeks before the release

„We only have to write this little library ..."

Use real-time clock for „happened-before"

Why do we need

distributed computing?

- Big Data
- Many CPUs
- Fault Tolerance

- Big Data
- Many CPUs
- Fault Tolerance

-> Distribution

-> Coordination

Wer zähmt uns diese Wildnis?

Apache ZooKeeper

http://zookeeper.apache.org

Apache ZooKeeper

http://zookeeper.apache.org

Hadoop

Netflix

Akka

Yahoo!

CXF (OSGi)

Camel

Apache ZooKeeper

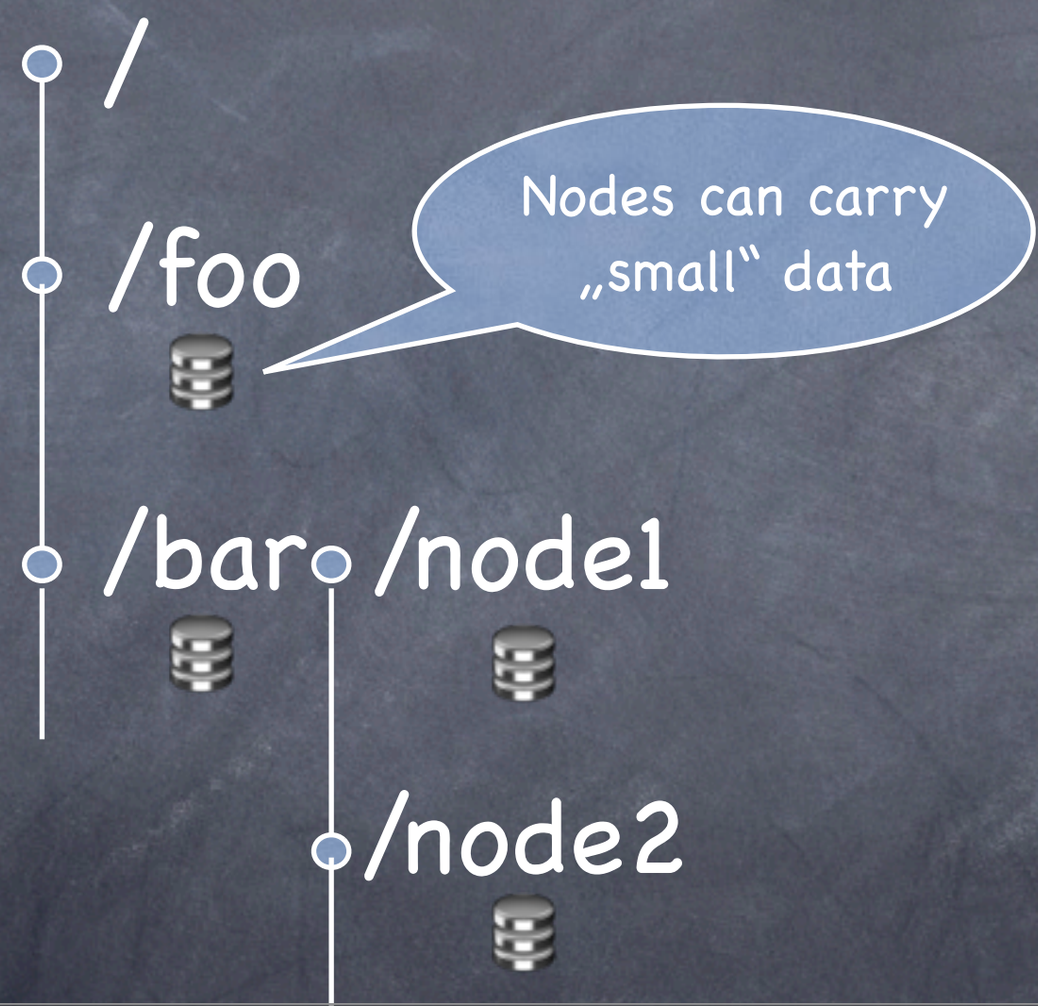
http://zookeeper.apache.org

ZooKeeper:

Service + Recipes

ZooKeeper: Service + Recipes

File-System like data nodes (znodes)



ZooKeeper manual:
„A common property of the various forms of coordination data is that they are relatively small: measured in kilobytes."

ZooKeeper: Service + Recipes

Java- & C-API:

- znodes:

  create, getChildren, exists, delete

- znode data:

  getData, setData

ZooKeeper: Service + Recipes

Java- & C-API:

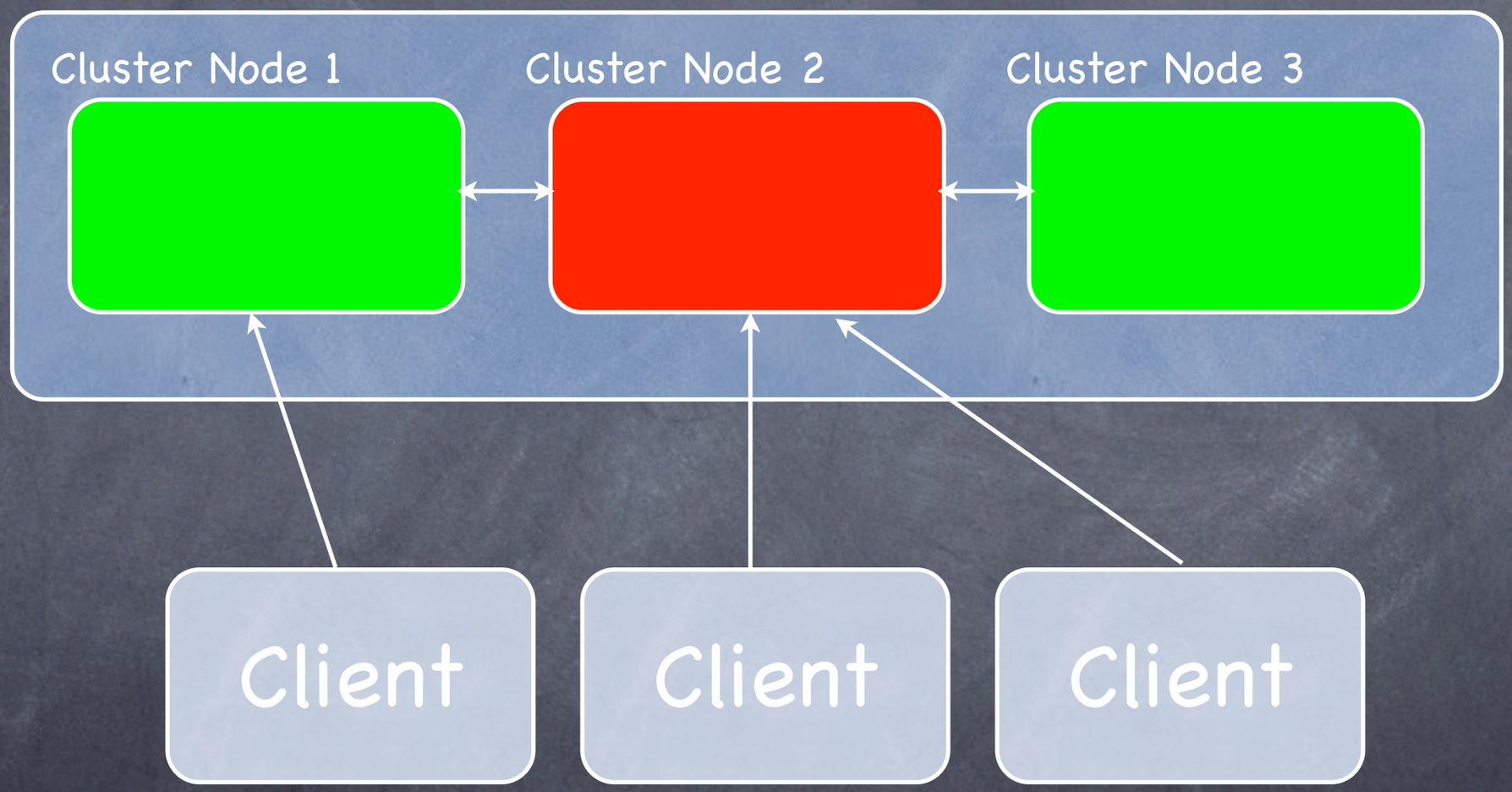
- znodes types (create-time):
  persistent or ephemeral
  sequential

- **watches:**
  one-time trigger when znode
  (data or children)
  has changed

ZooKeeper: Service + Recipes

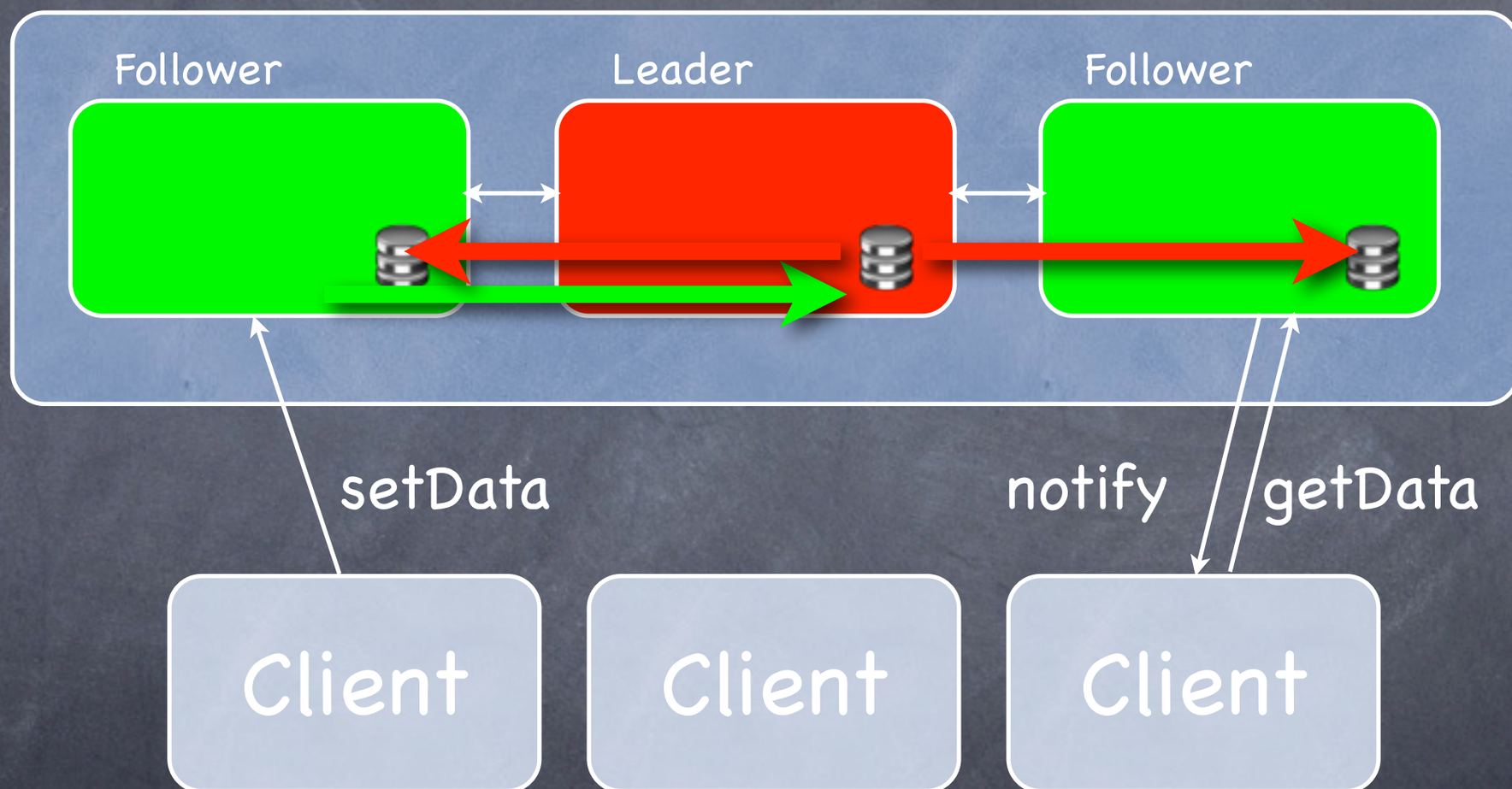
ZooKeeper: Service + Recipes

ZooKeeper Cluster



ZooKeeper: Service + Recipes

ZooKeeper Cluster



ZooKeeper: Service + Recipes

ZooKeeper Cluster

- Replicated data under leader coordination
- ZAB: ZooKeeper Atomic Broadcast
- Quorum: more than half are alive
- Failsafe with high throughput

ZooKeeper: Service + Recipes

ZooKeeper Guarantee: Sequential Consistency

- Updates from a client will be applied in the order that they were sent.
- But: there is no „total ordering“

ZooKeeper: Service + Recipes

ZooKeeper Guarantee: Atomicity - Reliability

- Updates either succeed or fail.
  No partial results.
- Once an update has been applied, it will persist from that time forward until a client overwrites the update.

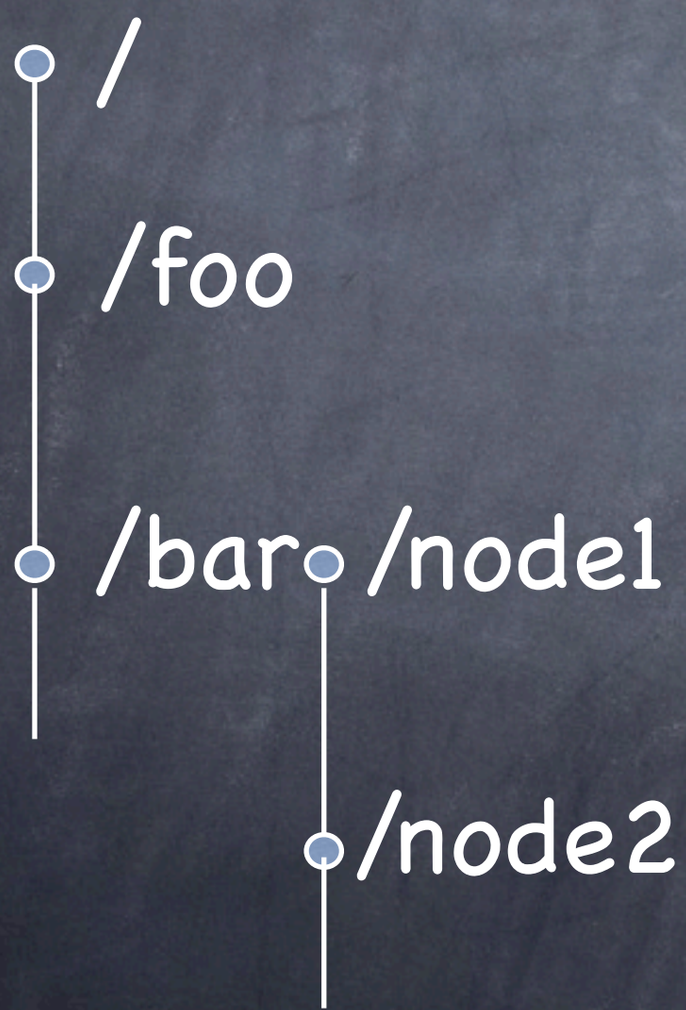
ZooKeeper: Service + Recipes

ZooKeeper Guarantee: Timeliness

- The clients view of the system is guaranteed to be up-to-date within a certain time bound.
- But: the clients do not see the same state „at the same time"

ZooKeeper: Service + Recipes

File-System like data nodes (znodes)



zkCli.sh (client shell):

create [-s] [-e] path [data] [acl]
delete [-v version] path
get [-s] [-w] path
set [-s] [-v version] path data
ls [-s] [-w] path

(with zookeeper 3.5.0)

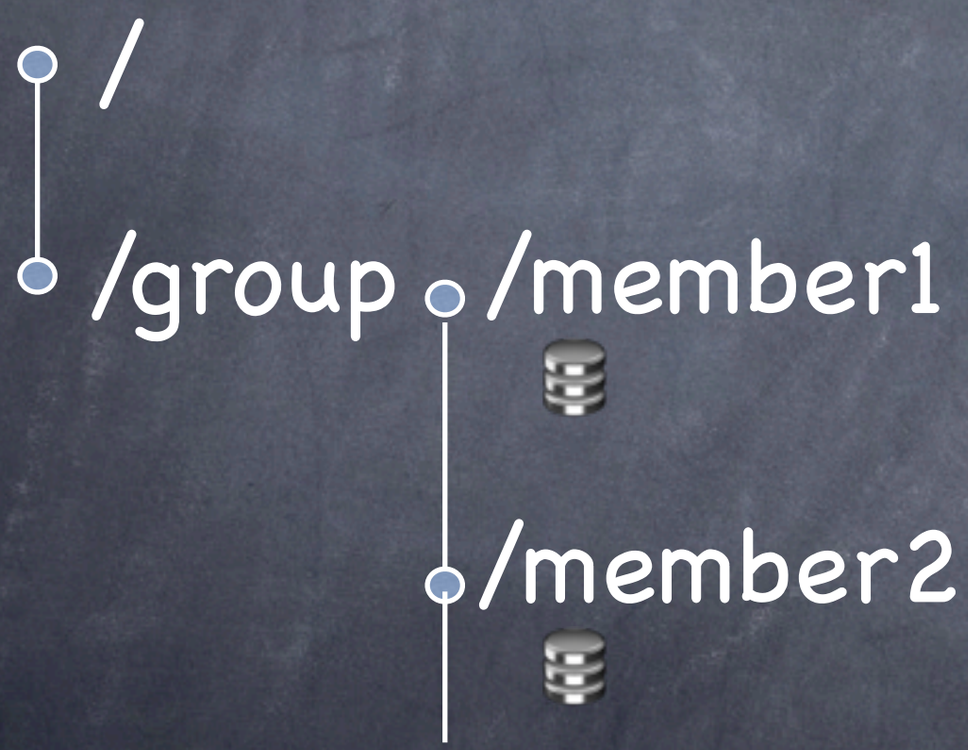
ZooKeeper: Service + Recipes

DEMO:
zkCLi.sh

ZooKeeper: Service + Recipes

- Group Membership, Name Service, Configuration
- Barriers, Queues
- Locks
- Two-Phase commit
- Leader Election

ZooKeeper: Service + Recipes

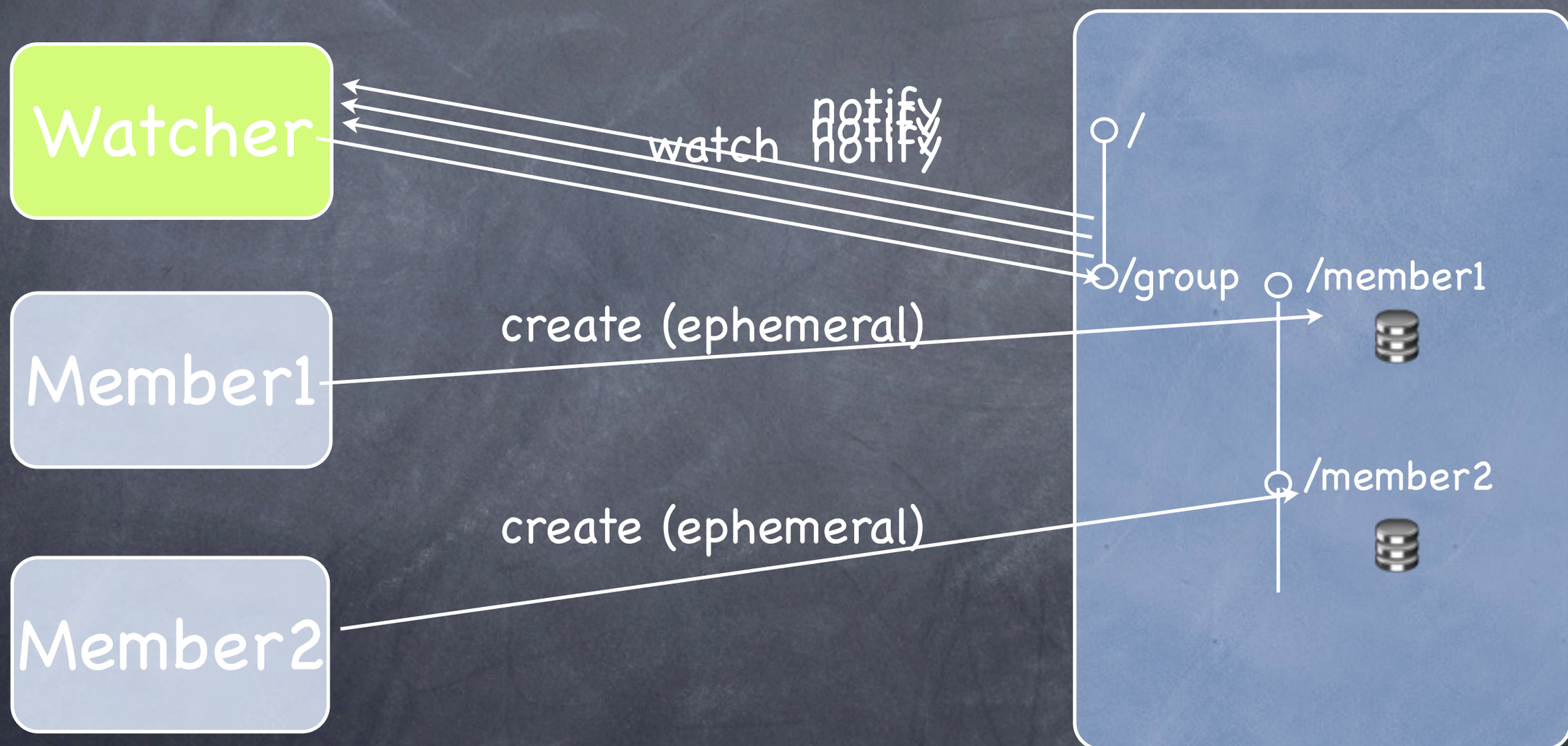


Group Membership:

- create znode for group
- create ephemeral node for group members
- watch on group/member node for changes

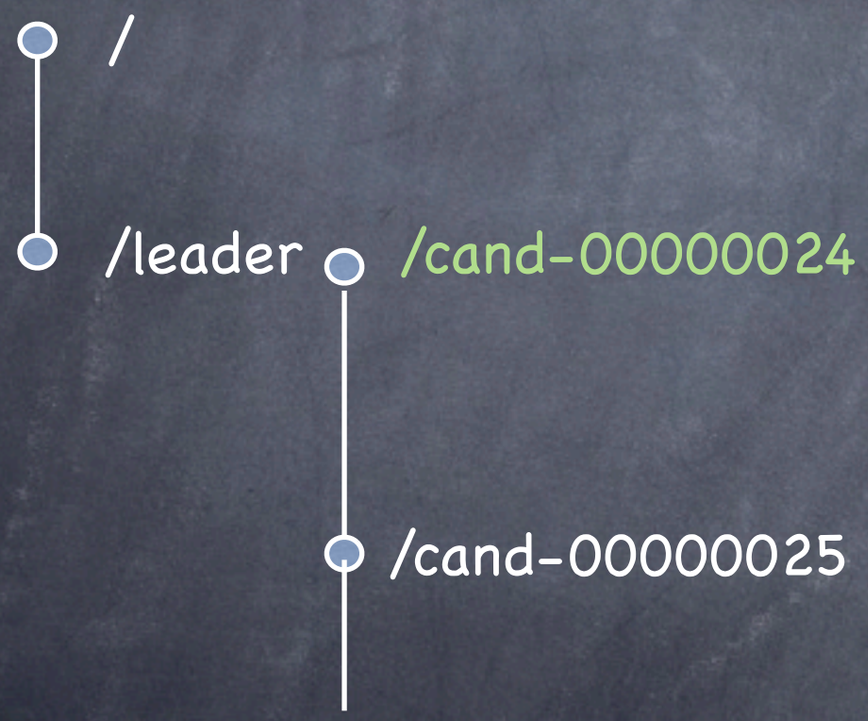
ZooKeeper: Service + Recipes

Group Membership



ZooKeeper: Service + Recipes

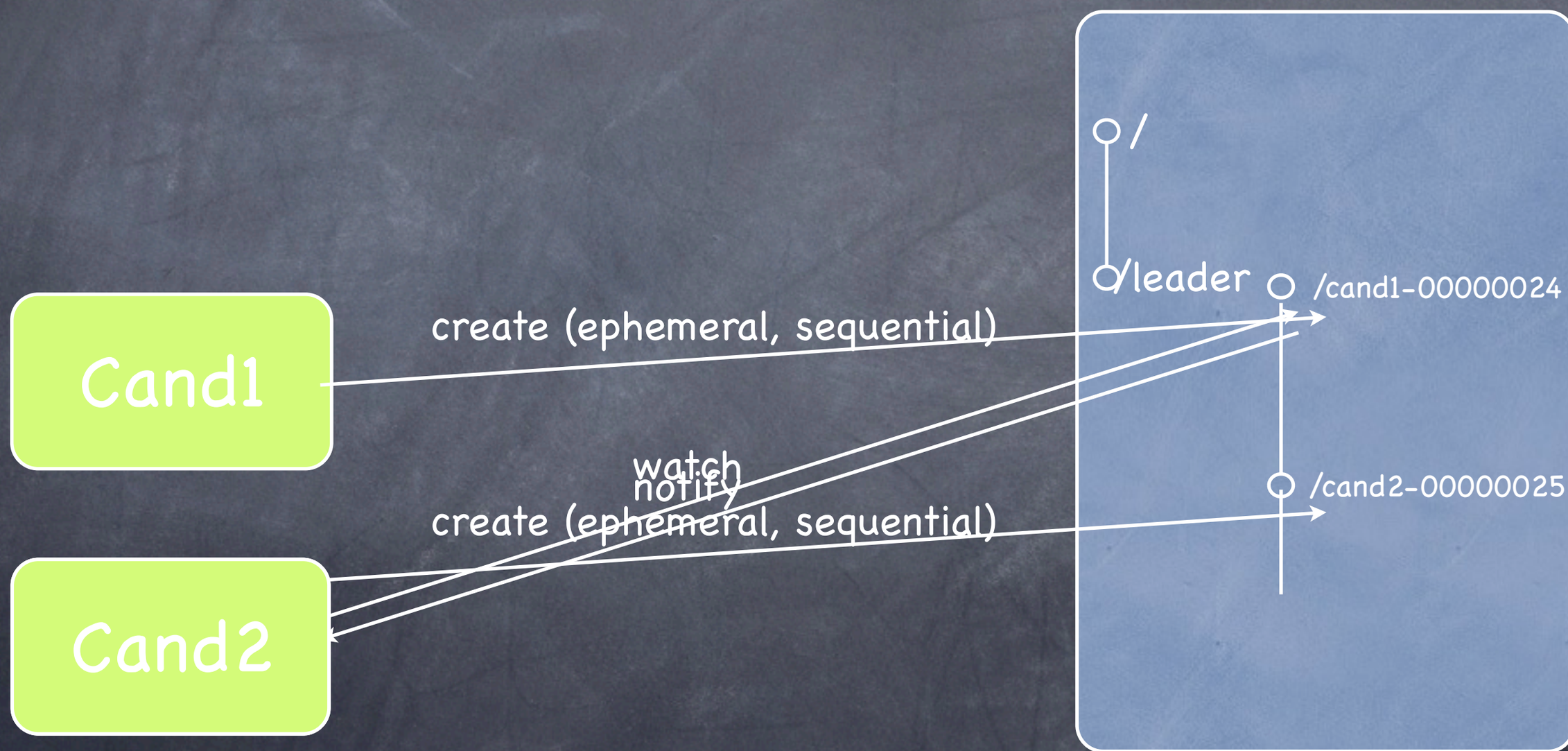
Leader Election



- create znode for group
- create ephemeral/sequential node for leader candidates
- candidates watch node with lower sequential number
- candidate with lowest number is leader

ZooKeeper: Service + Recipes

Leader Election



ZooKeeper: Service + Recipes

- Group Membership, Name Service, Configuration
- Barriers, Queues
- Leader Elec

Implementing the recipes can be challenging:
-> curator framework

curator by Netflix

ZooKeeper client wrapper and rich ZooKeeper framework

Curator n 'kyoor,ātər: a keeper or custodian of a museum or other collection - A ZooKeeper Keeper.
Curator is a set of Java libraries that make using Apache ZooKeeper much easier. While
ZooKeeper comes bundled with a Java client, using the client is non-trivial and error prone.

ZIP TAR

ZooKeeper: Service + Recipes

curator by Netflix

```
// Curator: make a client connection to ZK-cluster

CuratorFramework client =
    CuratorFrameworkFactory.builder().
        connectString(„localhost:2181“).
        retryPolicy(new RetryOneTime(1000)).
        namespace("/group").
        build();
```

ZooKeeper: Service + Recipes

curator by Netflix

```
// Curator: add a ephemeral member node

client.create().
    withMode(CreateMode.EPHEMERAL).
    forPath("/member1");
```

ZooKeeper: Service + Recipes

more Curator recipes:

- Leader Latch, ... Election
- Shared Lock, ... Read Write Lock, ... Semaphore
- Distributed Queue, ... Id Queue, ... Prio Queue
- Barrier, Counter
- ...

ZooKeeper: Service + Recipes

DEMO:

Leader Election with Curator

ZooKeeper: Service + Recipes

Who is missing in the Zoo?

Apache Camel

ZooKeeper: Service + Recipes

ZooKeeper component in Camel:

- From ZooKeeper node:

```
from("zookeeper://localhost:2181/somepath/somenode").
    to(....);
```

- To ZooKeeper node:

```
from(...).
    to("zookeeper://localhost:2181/somepath/somenode");
```

ZooKeeper: Service + Recipes

ZooKeeper component in Camel:

- Use ZooKeeper leader election as route policy (only the leader's route is started):

```
ZooKeeperRoutePolicy policy =
new ZooKeeperRoutePolicy("zookeeper://localhost:2181/someapp/somepolicy", 1);

from("direct:policy-controlled").routePolicy(policy).to("mock:controlled");
```

ZooKeeper: Service + Recipes

DEMO:
ZooKeeper camel component

ZooKeeper: Best ways to get started

- 1-node cluster and zkCli.sh (client shell)
- 3-node cluster
- Curator & Camel
- Get involved, community

ZooKeeper: Best ways to get started

> The examp

You are rig

from("zook

I update

Thanks fo

Bilgin

seems more like a writing example.

I really appreciate your help Hartmut. You have, indeed, found a bug. My test case didn't precisely replicate your situation. I updated the test so that it did (the lock node getting deleted after session expiration) and the same problem expressed. You also found the location of the bug making my job very easy ;)

Thanks again - I'll push a fix and get a new b

-Jordan

Hi Hartmut -

Great! Any improvements to the CLI are most welcome.

(...)

I would open a JIRA for each of the changes you want to work on, and we can discuss their merits there. I've commented briefly on each proposal below.

Wenn Du auch die Wildnis zähmen möchtest ...

... nimm Apache ZooKeeper!

Fragen?

Apache ZooKeeper

Curator

Camel ZooKeeper Component

A simple totally ordered broadcast protocol