

# LECTURE 10

Chubby and Zookeeper

# Implementing RSMs

2

- Logical clock based ordering of requests
  - ▣ Cannot serve requests if any one replica is down
  
- Primary-backup replication
  - ▣ Replace primary/backup upon failure
  - ▣ Relies on always available view service
  
- Paxos-based replicated service
  - ▣ Available as long as majority in same partition

# Case Studies (in brief)

3

- Chubby (OSDI 2006)
- ZooKeeper (USENIX ATC 2010)

# Chubby

4

- Internal service within Google
- Highly available coordination service

Building Chubby was an engineering effort required to fill the needs mentioned above; it was not research. We claim no new algorithms or techniques. The purpose of this paper is to describe what we did and why, rather than to advocate it. In the sections that follow, we de-

# Chubby

5

- Internal service within Google
- Highly available coordination service
  
- Two purposes:
  - ▣ Lock service
  - ▣ File system (for small files)

# Example: GFS Chunkmaster

6

```
x = Open("/ls/gfs-cell18/chunkmaster")

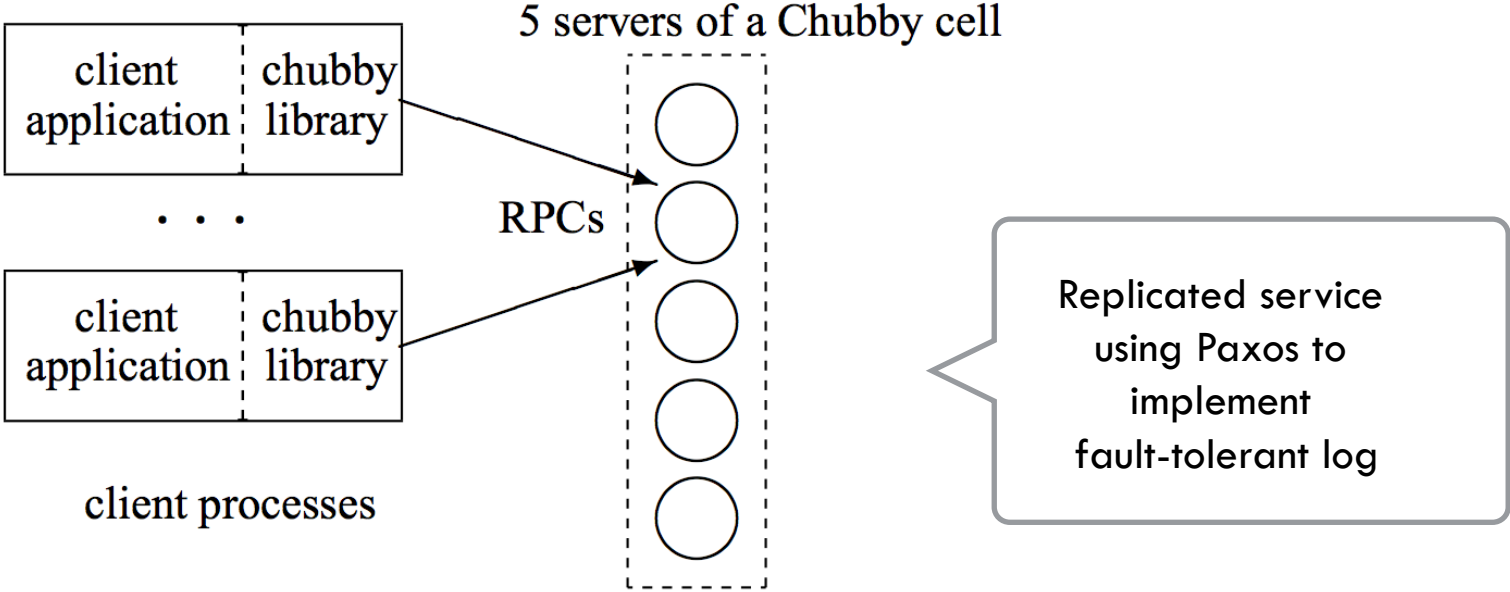
if (TryAcquire(x) == success) {
    // I'm the chunkmaster, tell everyone
    SetContents(x, my-address)
} else {
    // I'm not the master, find out who is
    chunkmaster = GetContents(x)
}
```

# Why this interface?

7

- Why not a library that implements Paxos?
- Developers do not know how to use Paxos
  - ▣ They at least think they know how to use locks!
- Need to export result of coordination outside of system
  - ▣ Example: Clients need to discover GFS master

# Chubby Design



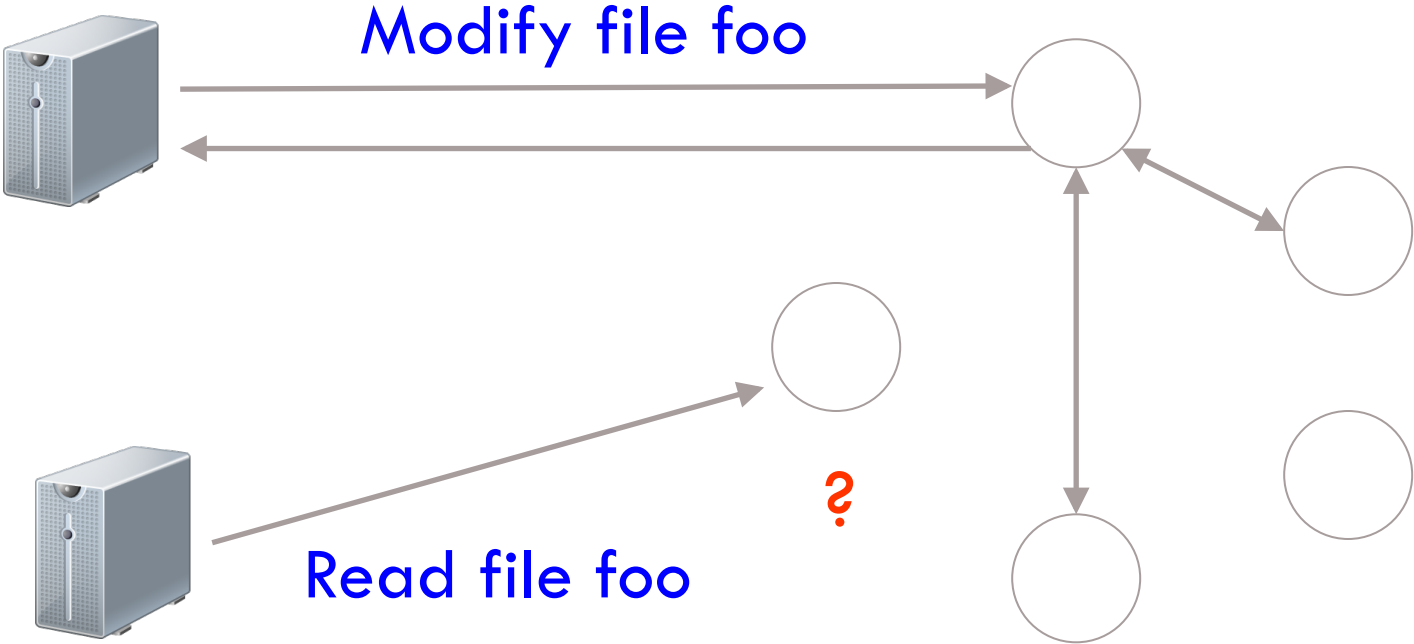


# Chubby procedure

---

- Client sends request to nearest replica.
- Replica returns Chunkmaster's address
- Client directs requests to the Chunkmaster until it says it is no longer playing that role.
- Writes propagated to replicas using consensus
- Reads handled without consensus (Safe)

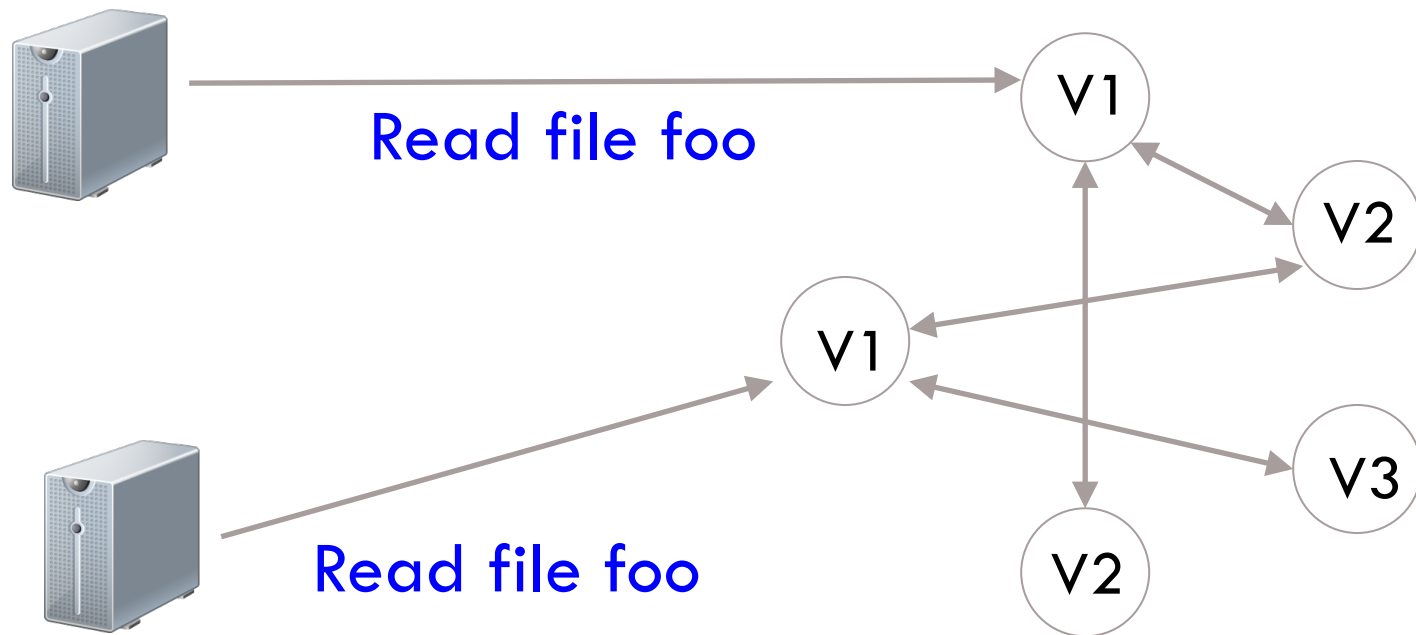
# Chubby in Action



# Reads in Paxos-based RSM

11

For every key, every replica stores (value, version)  
Return latest version out of majority of replicas



**Why do different replicas have different versions?**

Gap between Accept and Learn

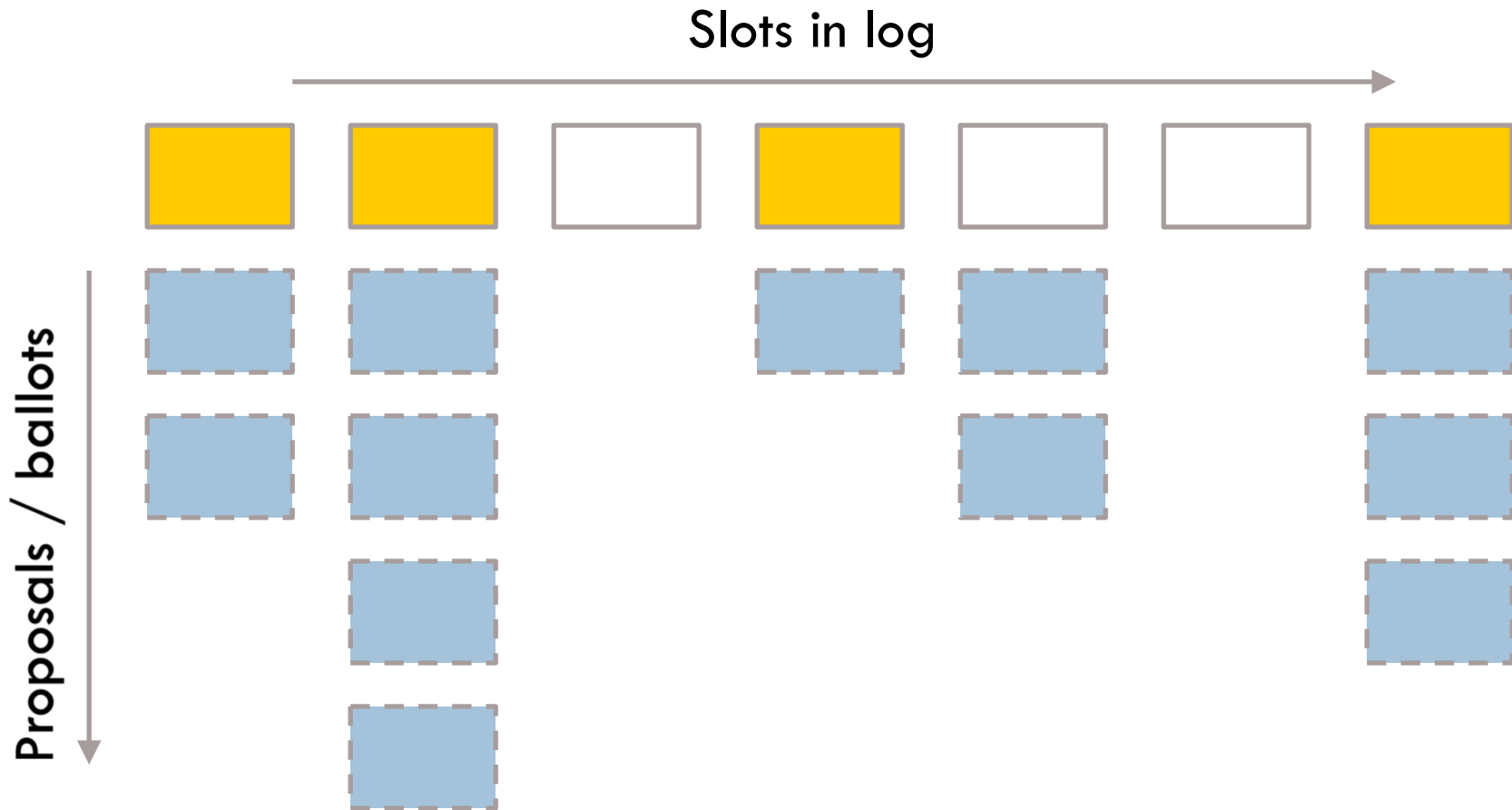
# Reads in Paxos-based RSM

12

- How to ensure linearizability?
  - ▣ A read must see the effect of all accepted writes
  
- Get read accepted to one of the slots in replicated log
  
- Every replica executes command in a slot only after executing commands in all prior slots

# RSM with Paxos

13



# Reads in Paxos-based RSM

14

- How to ensure linearizability?
  - ▣ A read sees the effect of all accepted writes
- Get read accepted to one of the slots in replicated log
- Problem: Poor performance at scale

# Reads/Writes in Chubby

15

- One of the 5 replicas chosen as the master
- Clients submit all reads and writes to master
- How to handle master failure?
  - ▣ Another replica must propose itself as master
  - ▣ New master must first “catch up”
- Master is performance bottleneck

# Client sessions

---

- A client requests a new session by contacting the master of a Chubby “cell”
- It ends the session when it terminates or if it has been idle (no open handles or no calls for a minute).
- Lease → session needs to be renewed before lease expires.
  - Master can move it forward but not backward.



# Event subscriptions

---

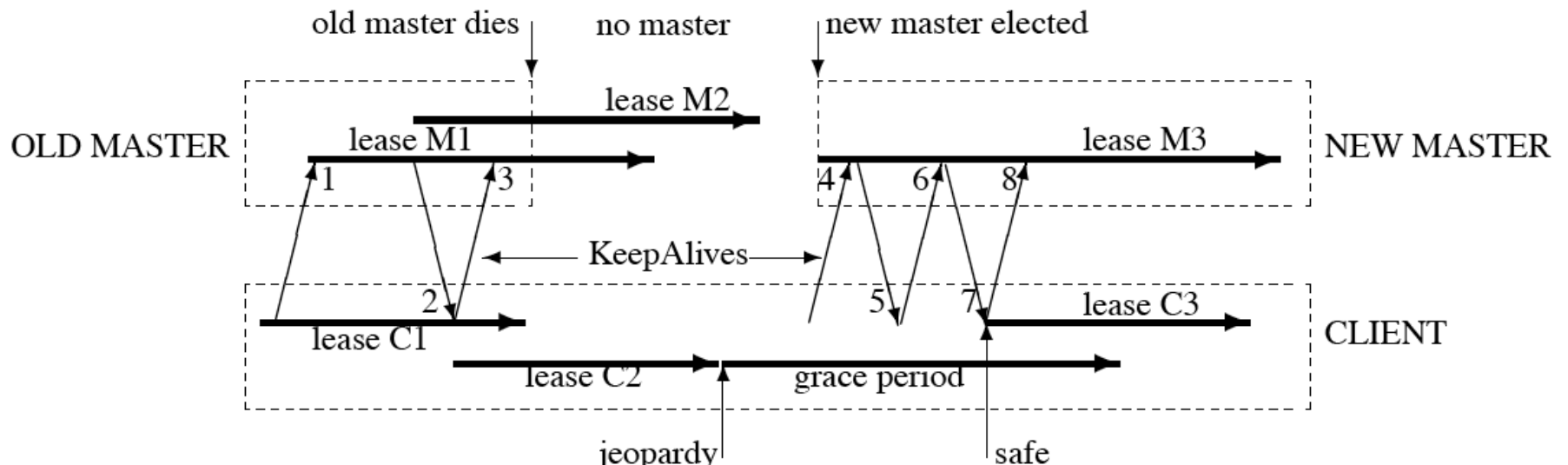
- Clients may subscribe to a range of events (e.g., file contents modified or master failure).
- Such events are delivered asynchronously via an up-call from the Chubby library.
  - ▣ Delivered after the event has taken place

# Scaling Chubby

18

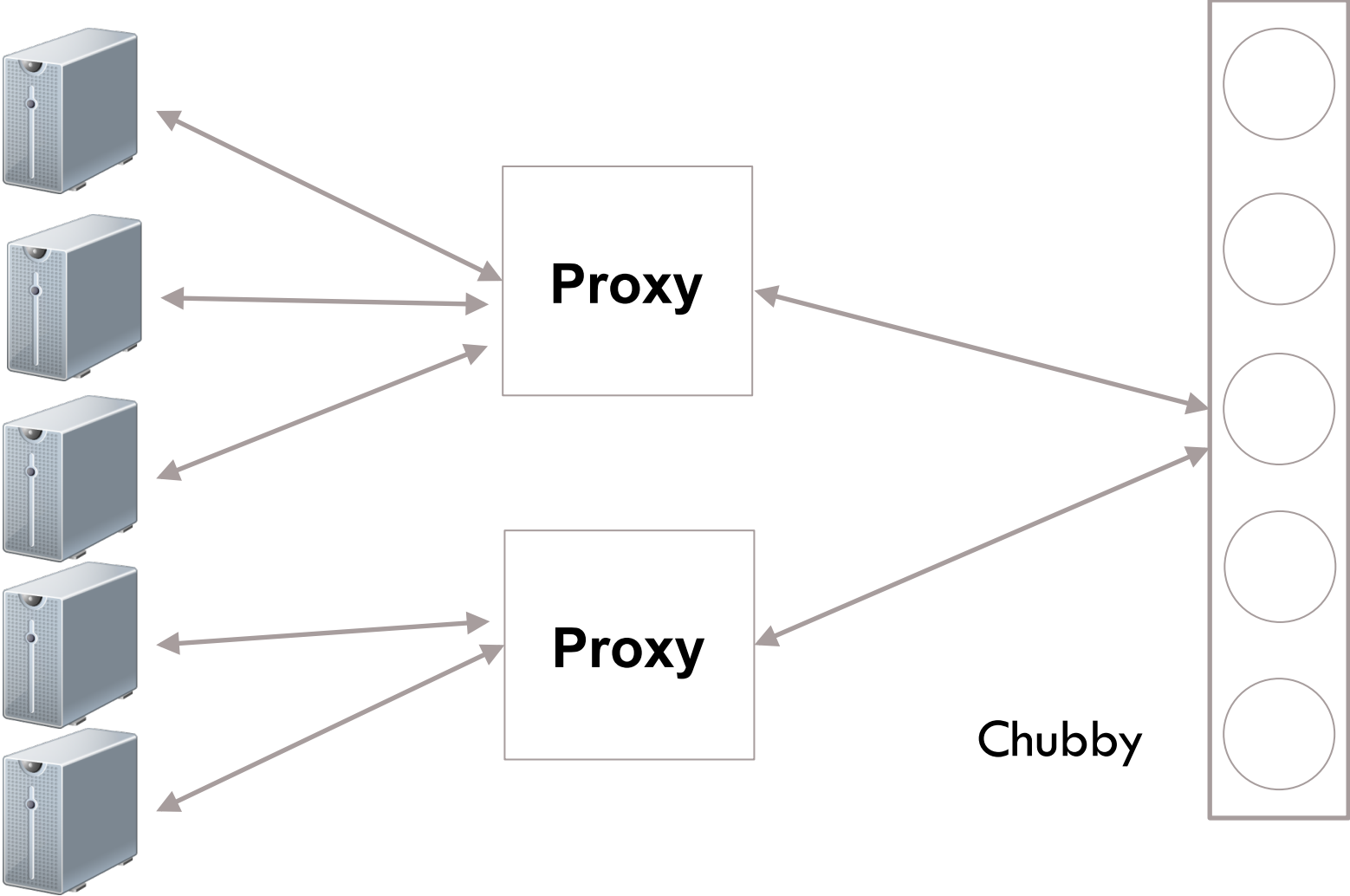
- Clients cache data they read
- Reading from local cache violates linearizability
  - ▣ How to fix this?
- Master invalidates cached copies upon update
  - ▣ Modification blocked until invalidations are sent
    - Note that caches not updated (only invalidated) – this may be unnecessary.
  - ▣ Reads proceed as usual (much more frequent than writes)
- Master must store knowledge of client caches
- What if master fails?

# Grace period for leases



- Grace period after lease expires
  - ▣ If keepalive received within grace period, state updated
- If grace period expires, the client assumes session has expired (Chubby cell inaccessible)
  - ▣ Call returns with an error
- Failed master discards state about sessions, handles and locks.

# Scaling Chubby with Proxies



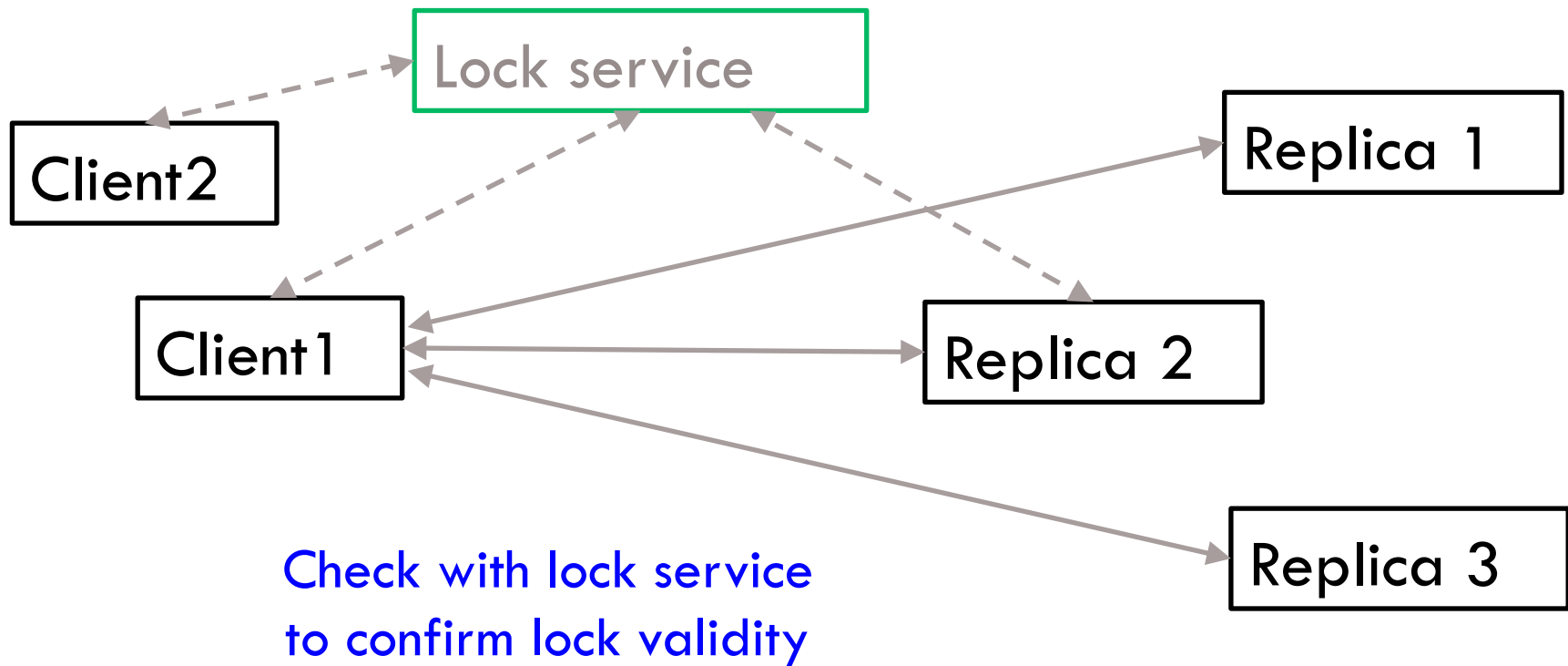
# Handling client failures

21

- What if a client acquires a lock and fails?
- Client library exchanges keep-alives with master
- Lock revoked upon client failure
- Problem?
  - ▣ Network partition not client failure
- Chubby associates lock acquisitions with sequence numbers
  - ▣ Can distinguish operations from previous lock holders

# Discrepancy in Lock Validity

22



# ZooKeeper

23

- Open source coordination service
- Addresses need for polling in Chubby
  - ▣ Example: **If you cannot acquire lock, need to retry**
- Goal in ZooKeeper: **Wait-free coordination**

# ZooKeeper: Watch mechanism

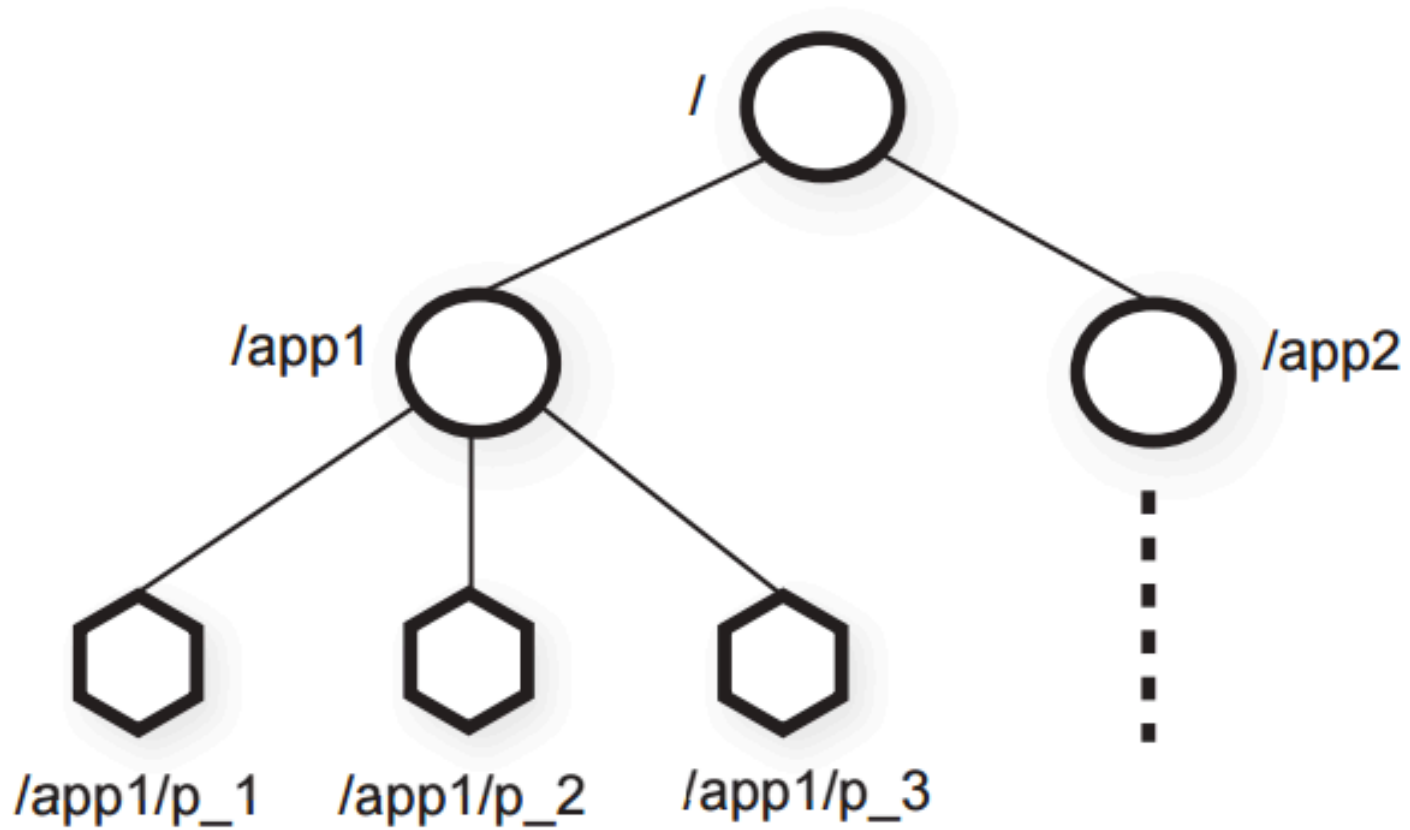
24

- Clients can register to “watch” a file
  - ▣ Watches are one-time triggers associated with a session (unregistered once triggered or when session closes)
  - ▣ ZooKeeper notifies client when file is updated (trigger)
  
- Example:
  - ▣ Try to acquire a lock by creating a file with an ephemeral flag
  - ▣ If file already exists, watch for updates
  - ▣ Upon watch notification, try to re-acquire lock
  
- Problem?
  - ▣ Herd effect
    - Many clients which get notification may vie for the lock



# ZooKeeper: Hierarchy

25



# ZooKeeper: Sequencer

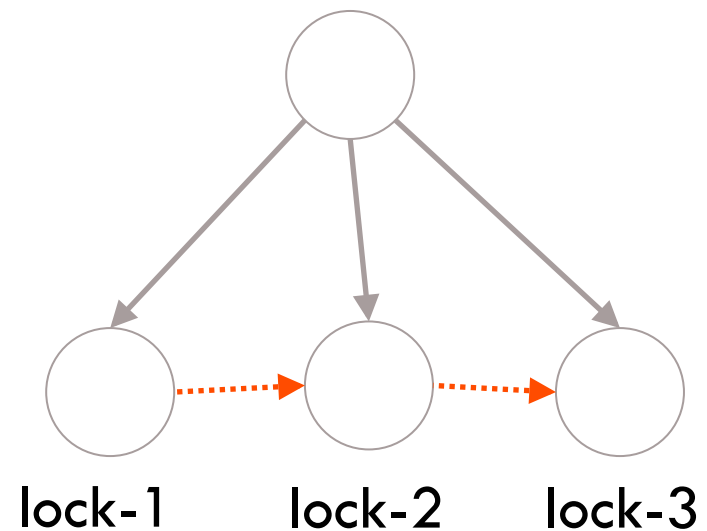
26

## Lock

```
1 n = create(l + "/lock-", EPHEMERAL|SEQUENTIAL)
2 C = getChildren(l, false)
3 if n is lowest znode in C, exit
4 p = znode in C ordered just before n
5 if exists(p, true) wait for watch event
6 goto 2
```

## Unlock

```
1 delete(n)
```



# Zookeeper:Sequencer

---

- The SEQUENTIAL flag orders the client's attempt to acquire lock with respect to others.
- If the znode created is lowest, get lock; else, wait for deletion of immediately prior znode.
- Herd effect avoided by waking up only one process when lock is released or a lock request is abandoned.
- Releasing lock: delete the proper znode.

# Chubby vs. ZooKeeper

28

- Difference between invalidation and watch?
- Invalidation:
  - Only library receives notification to update cache
- Watch:
  - Application receives notification
  - Only app knows what it needs to do