Chubby

Doug Woos

Logistics notes

Lab 3a due tonight

Friday's class is in GWN 201!

Next few papers

Three real-world systems from Google

- Chubby: coordination service
- BigTable: storage for structured data
- GFS: storage for bulk data

All highly influential, have open-source clones

- Chubby -> Zookeeper, etcd
- BigTable -> HBase, Cassandra, other NoSQL stores
- GFS -> HDFS

Chubby

Distributed coordination service

Goal: allow client applications to synchronize and manage dynamic configuration state

Intuition: only some parts of an app need consensus!

- Lab 2: Highly available view service
- Master election in a distributed FS (e.g. GFS)
- Metadata for sharded services

Implementation: (Multi-)Paxos SMR

Why Chubby?

Many applications need coordination (locking, metadata, etc).

> Every sufficiently complicated distributed system contains an ad-hoc, informally-specified, bug-ridden, slow implementation of Paxos

Paxos is a known good solution

(Multi-)Paxos is hard to implement and use

How to do consensus as a service

Chubby provides:

- Small files
- Locking
- “Sequencers”

Filesystem-like API

- Open, Close, Poison
- GetContents, SetContents, Delete
- Acquire, TryAcquire, Release
- GetSequencer, SetSequencer, CheckSequencer

How to do consensus as a service

Chubby provides:

- Small files
- Locking
- “Sequencers”

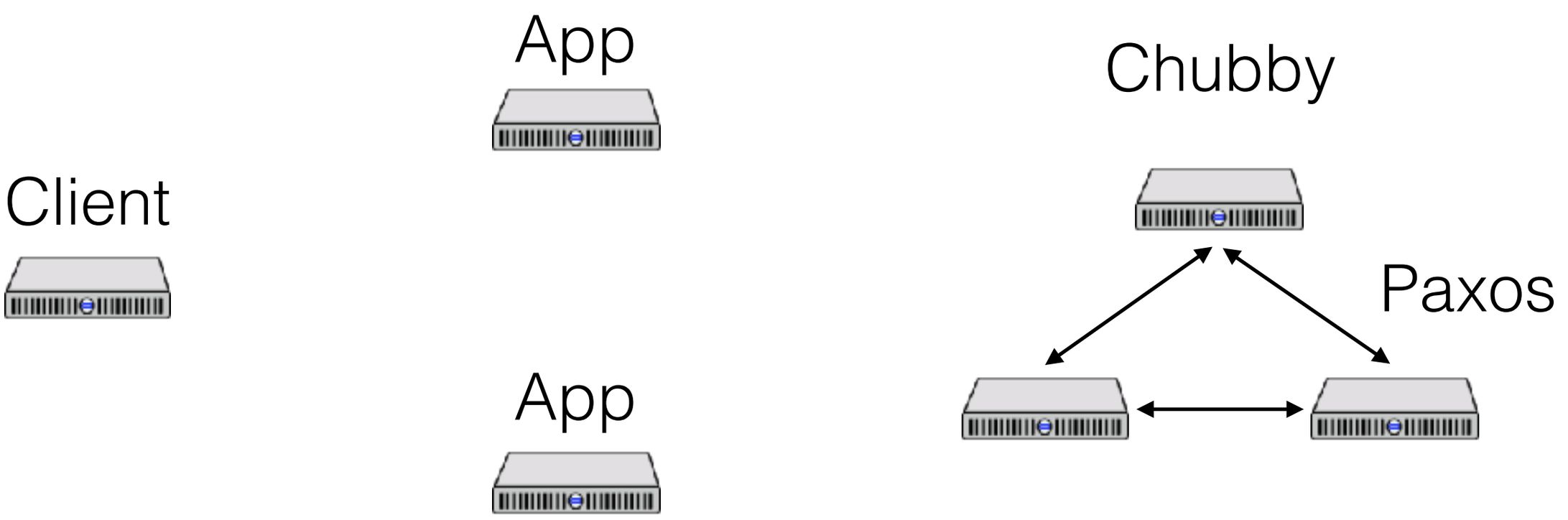
Filesystem-like API

- Open, Close, Poison
- GetContents, SetContents, Delete
- Acquire, TryAcquire, Release
- GetSequencer, SetSequencer, CheckSequencer

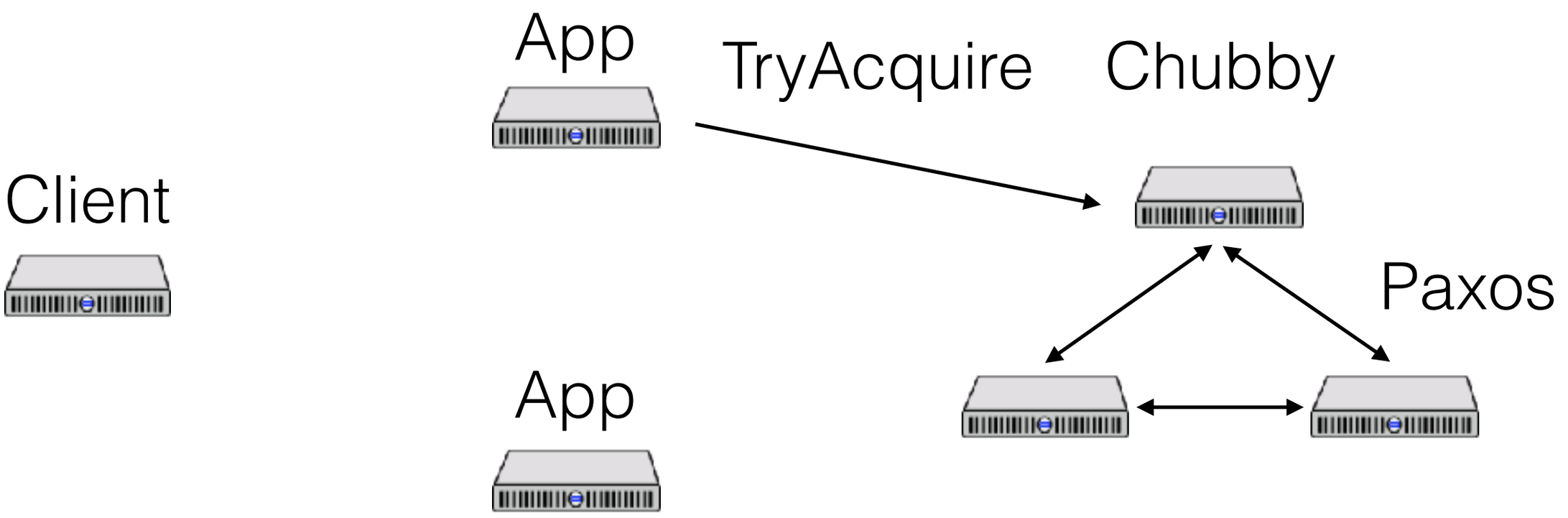
Example: primary election

```
x = Open(“/ls/cell/service/primary")
if (TryAcquire(x) == success) {
   // I'm the primary, tell everyone
   SetContents(x, my-address)
} else {
   // I'm not the primary, find out who is
   primary = GetContents(x)
   // also set up notifications
   // in case the primary changes
}
```

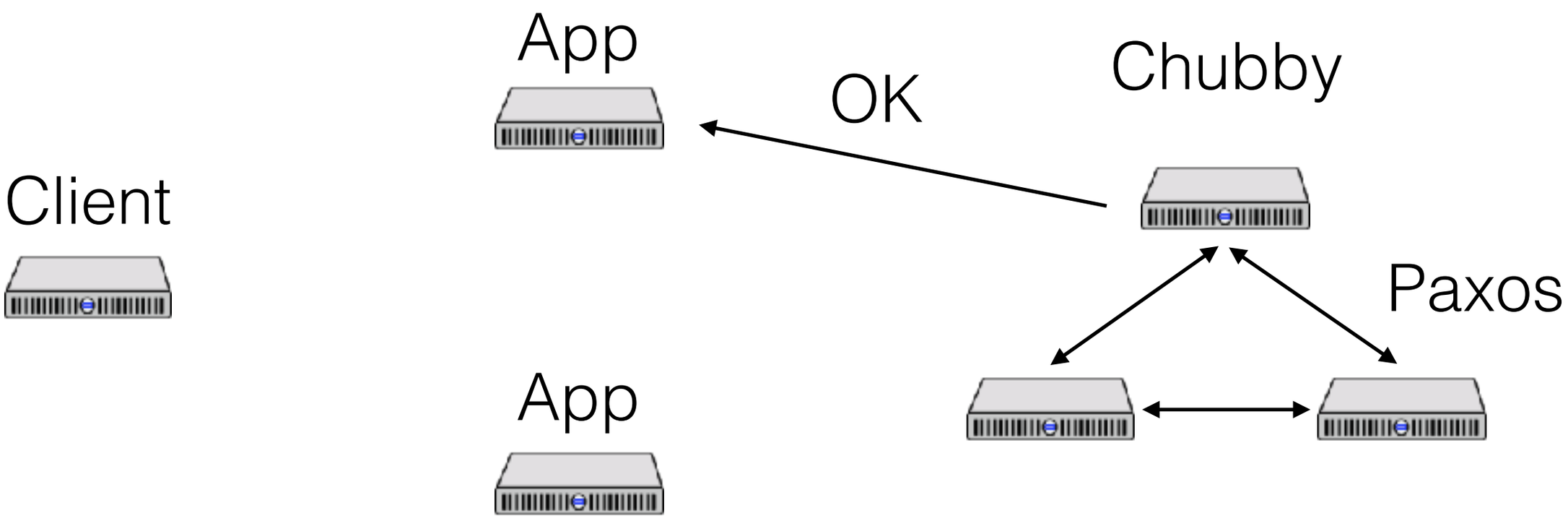
Example



Example



Example



Example

Primary

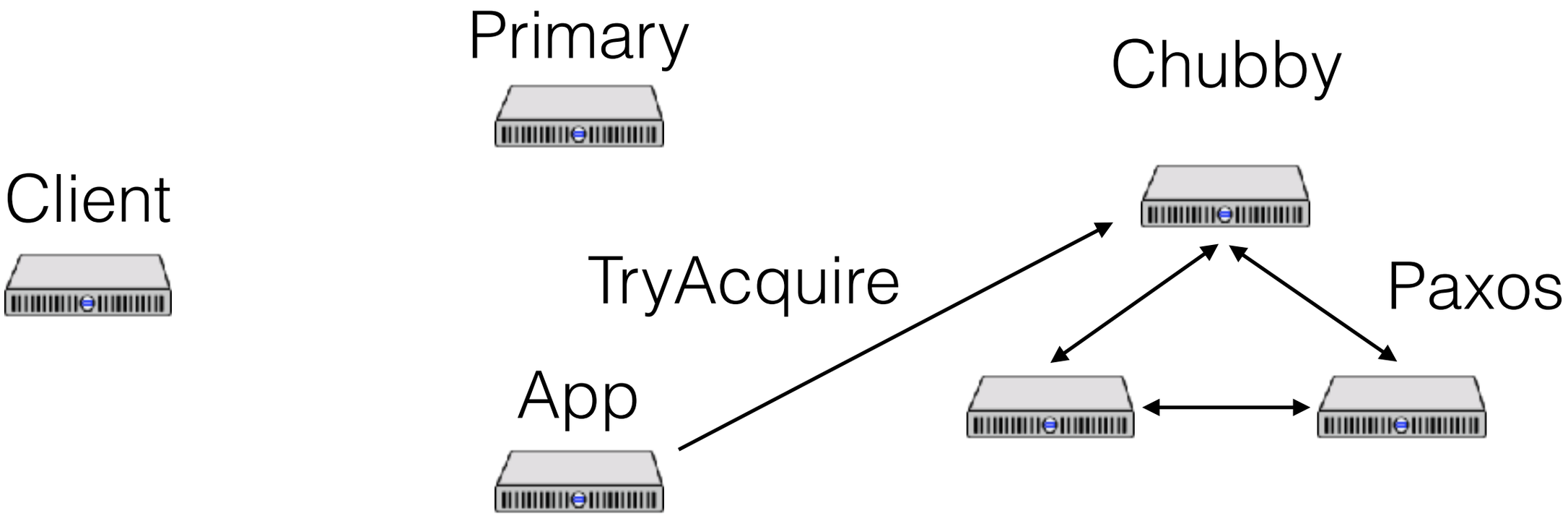
Chubby

Client

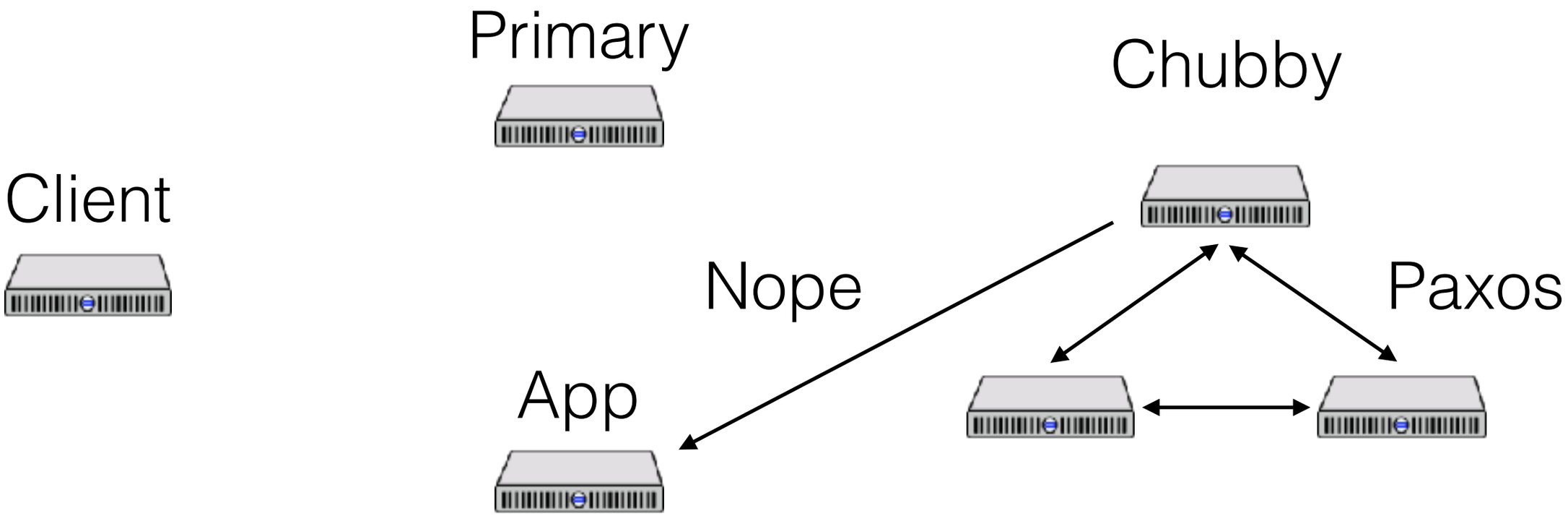
Paxos

App

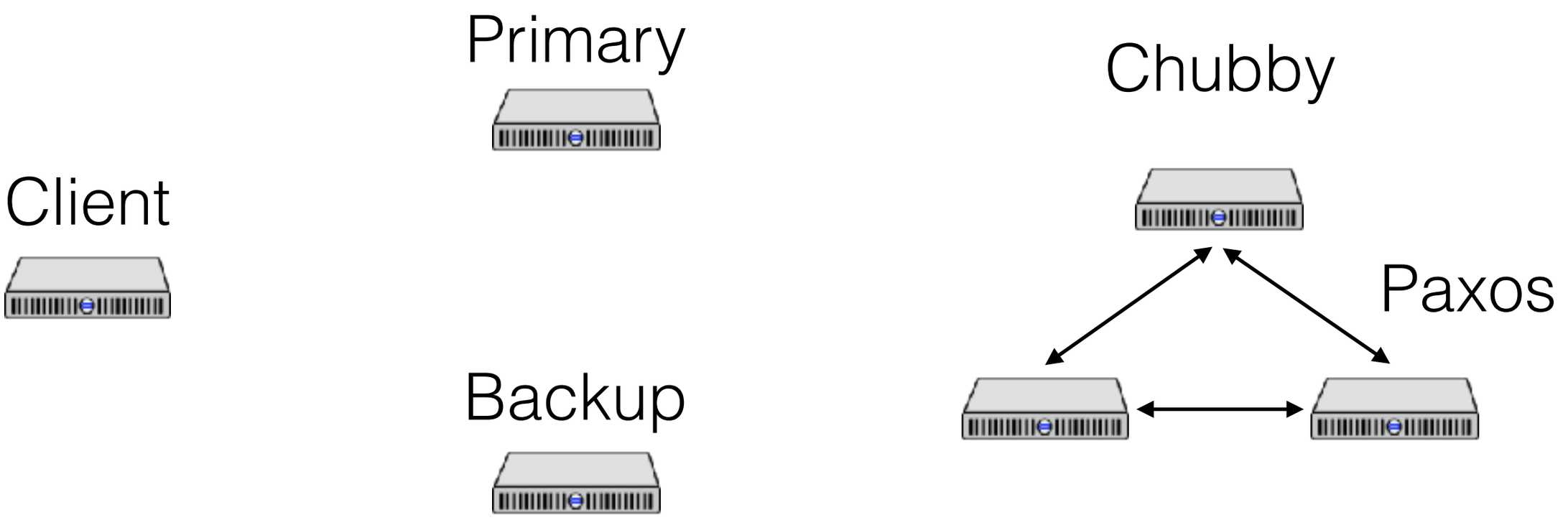
Example



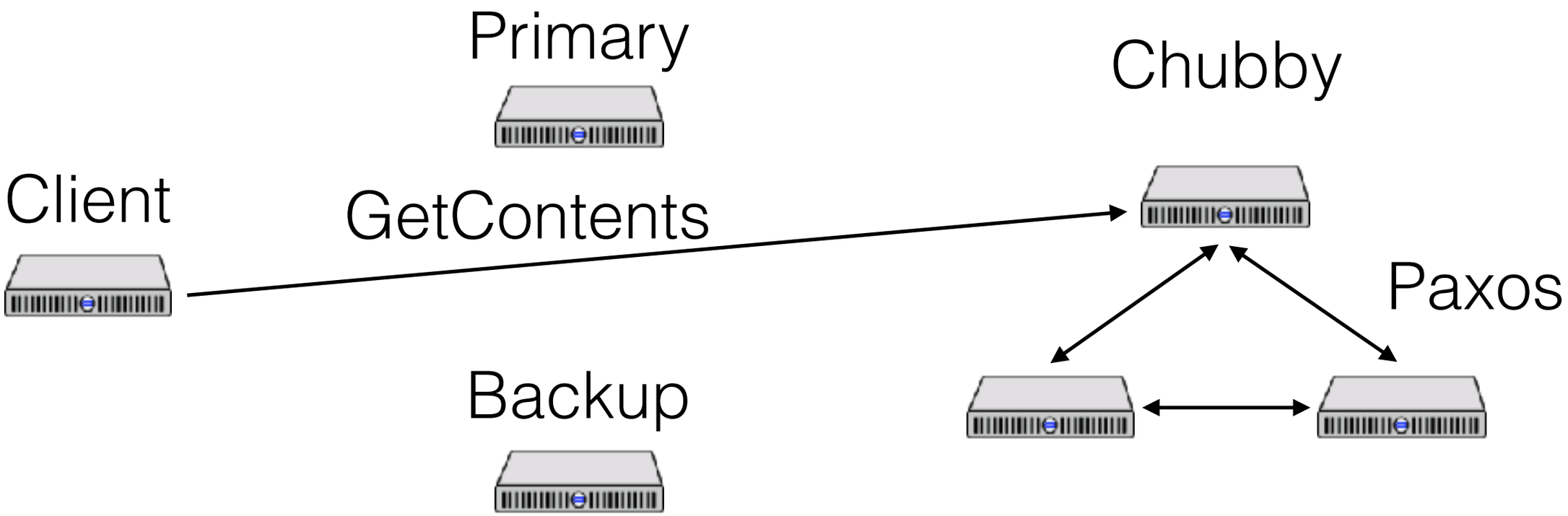
Example



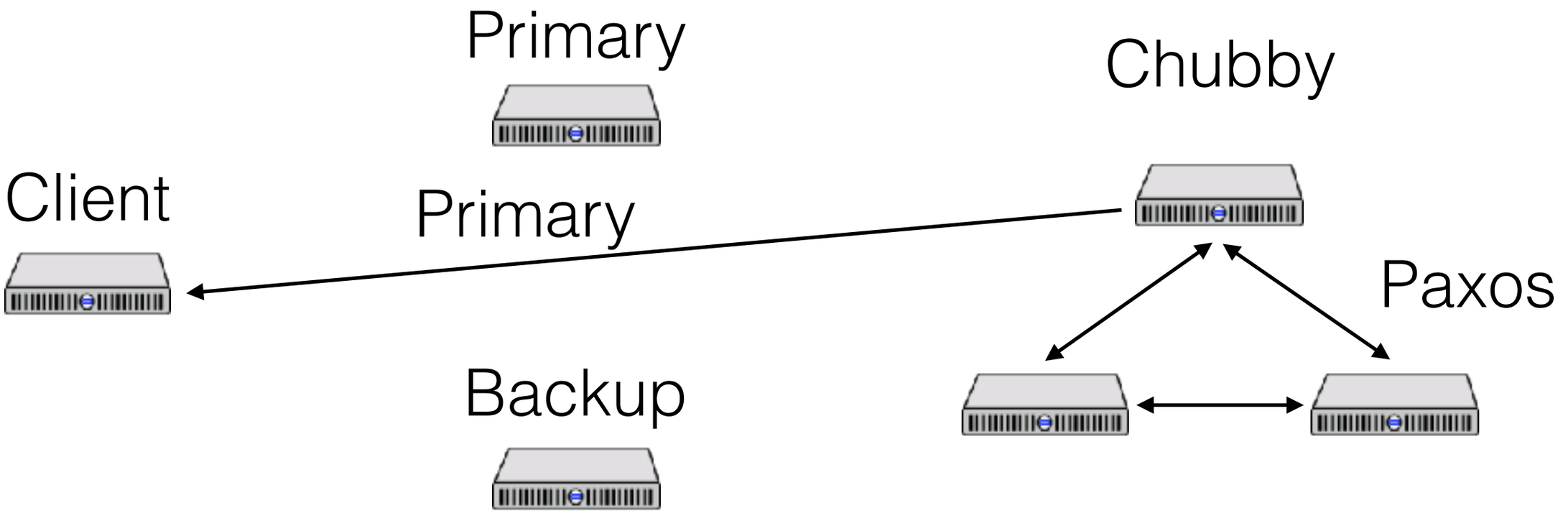
Example



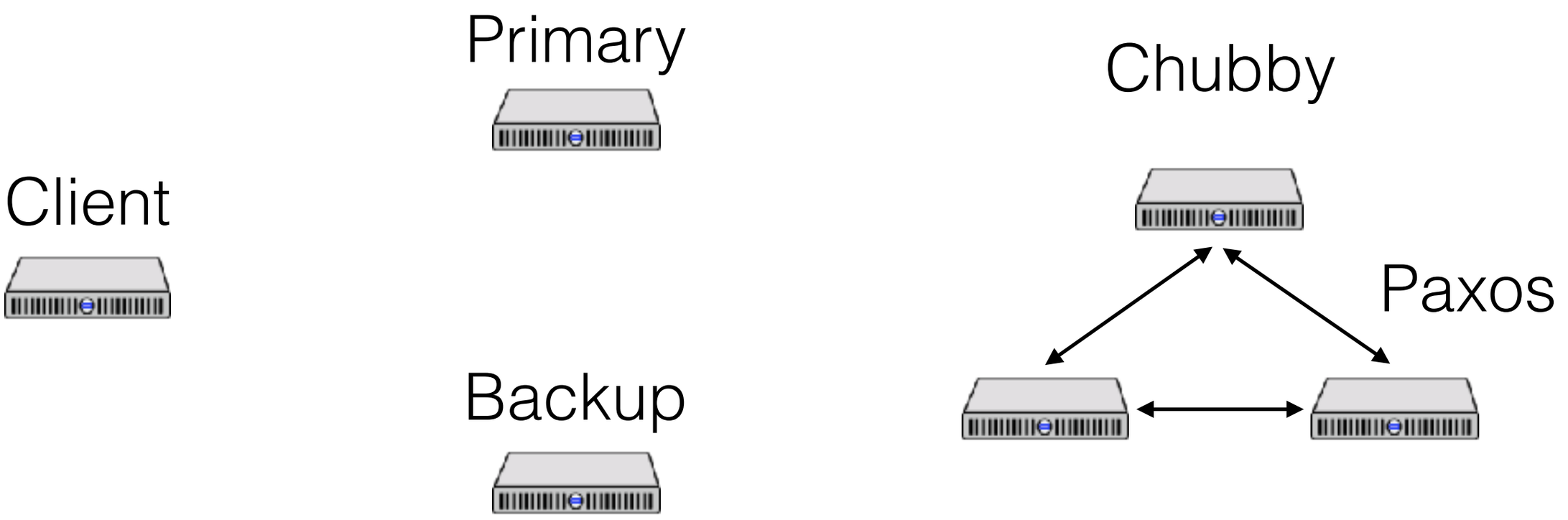
Example



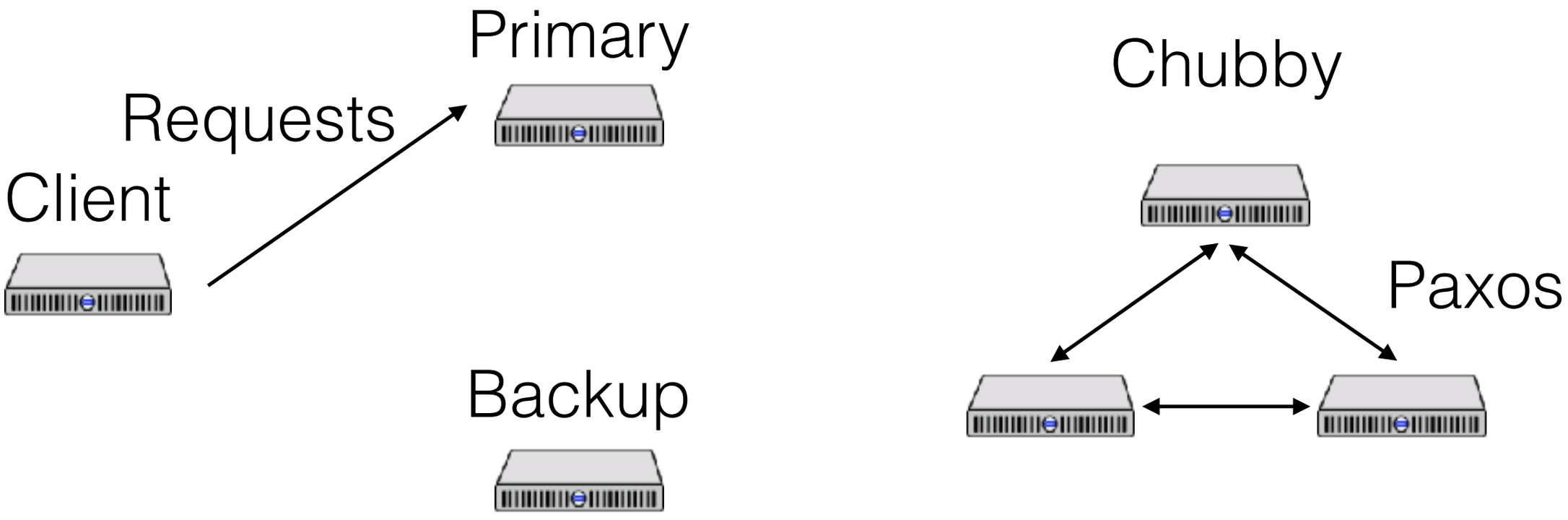
Example



Example



Example



Why a lock service?

One option: a Paxos library (these exist)

Why a service:

- Easier to add to existing systems

- Want to store small amounts of data, e.g. names, externally (for clients)

- Developers don’t understand Paxos!

  - As it turns out, they don’t understand locks either

- Can have fewer app servers

Performance

Not highly optimized!

Later (and last Thursday): how to do Paxos, fast

Paxos implementation: ~1000 ops/s

Initially, needed to handle ~5000 ops/s

How to scale?

- Adding nodes to Paxos group?

Performance

Batching

Partitioning

Leases

(Consistent) Caching

Proxies

Batching

Master accumulates requests from many clients

Does one round of Paxos to commit all to log

Big throughput gains at expense of latency

- Classic systems trick (e.g. disks)

- Ubiquitous in systems w/o latency requirements

Partitioning

Run multiple Paxos groups, each responsible for different keys

Different replicas master in some, replica in others

Common in practice

- Alternative: Egalitarian Paxos

Leases

Most requests are reads

Want to avoid communication on reads

- Communication not needed for durability

- Just need to ensure master hasn't changed

Optimization: master gets lease, renewed while up

- Chubby: ~10s

- Master can process reads alone if holding lease

- If master fails, need to wait 10s before new master can respond to requests (why?)

Caching

Chubby uses client caching heavily

- file data
- file metadata (incl. non-existence!)

Writ-through, strong leases (+ invalidations)

- Master tracks which clients might have file cached
- Sends invalidations on update
- Caches expire automatically after 12s

Proxies

KeepAlives and invalidations are a huge % of load

Use proxies to track state for groups of clients

- To master, proxies act exactly like clients

- To clients, proxies act exactly like master

Client

Proxy

Master

Handling failure

Replica failure: no problem

Master failure

Client failure

Master failure

Client stops hearing from master

- Notifies application (stop sending new requests!)
- Clears cache
- “grace period” begins (wait for election before giving up on Chubby entirely)
- If new master found, continue
- Otherwise, throw an error to the application

Master failure

Meanwhile, in the Chubby cell…

If master has failed:

- Do leader election (PMMC)
- Rebuild state from other replicas + clients
- Wait for old lease to expire!

Performance

~50k clients per cell

~20k files

- Majority are open at any given time
- Most < 1k
- All < 256k (hard limit—why?)

2k RPCs/s

- 93% KeepAlives!

All of these numbers probably bigger now!

Name service

Surprising dominant use case: name servers!

Problems with DNS

- Designed for web, where slow propagation OK

- Weak leases

- Performance bad (see Ousterhout!) if TTLs are low

Chubby: decent performance, strongly consistent

Why not use Chubby on the web?

Discussion

Most errors in failover code

- Netflix: Chaos Monkey

Chubby metadata stored in Chubby itself

Advisory vs. Mandatory locks

Importance of programmer convenience

- Locks—familiar, but programmers get it wrong!

How much are clients trusted?

Note: interesting paper called “Paxos made live”

- Making Paxos work within Chubby