ZooKeeper: Wait-free coordination for Internet-scale systems

Patrick Hunt, Mahadev Konar, Flavio P. Junqueira, Benjamin Reed

Presented by Justin Beemer

What is Coordination?

- Configuration Management
- Leader Election
- Group Membership
- Locks & Synchronization Primitives

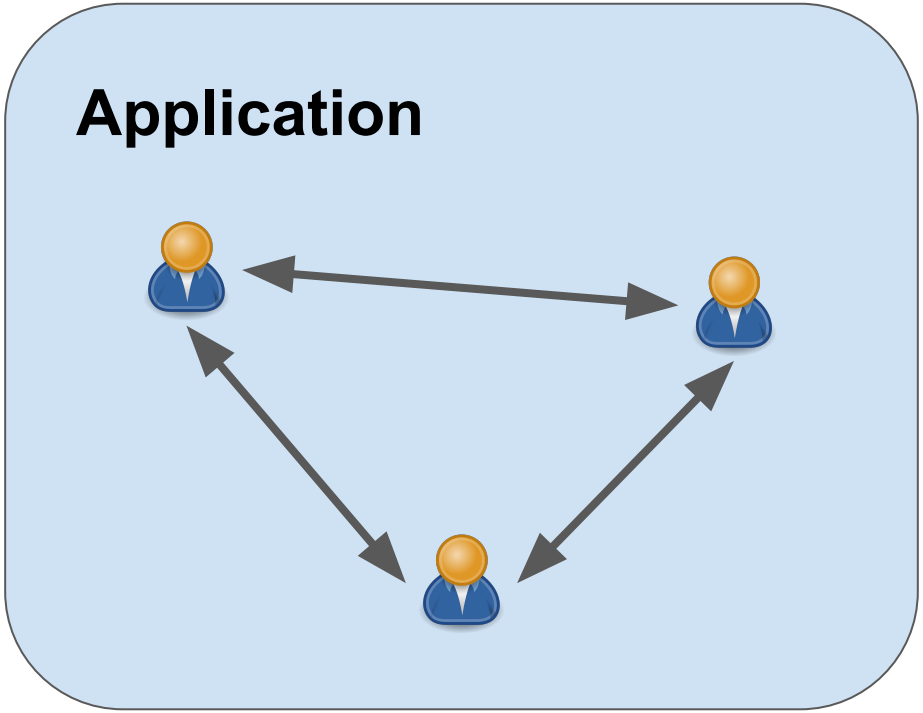
Approach #1: Implement Coordination in the Application

Pros

- Can design and optimize for specific application needs

Cons

- Not reusable
- Really hard to implement correctly



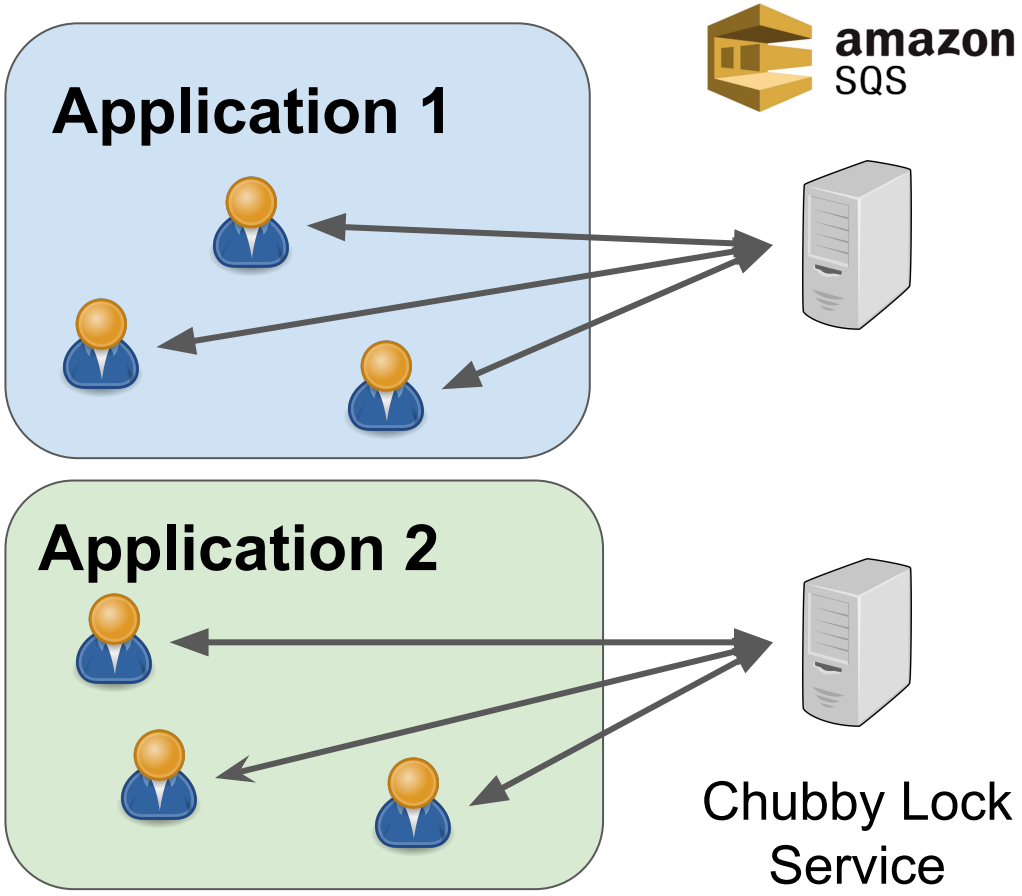
Approach #2: Implement Coordination Services

Pros

- Easier for the client to implement
- Services usable by many applications

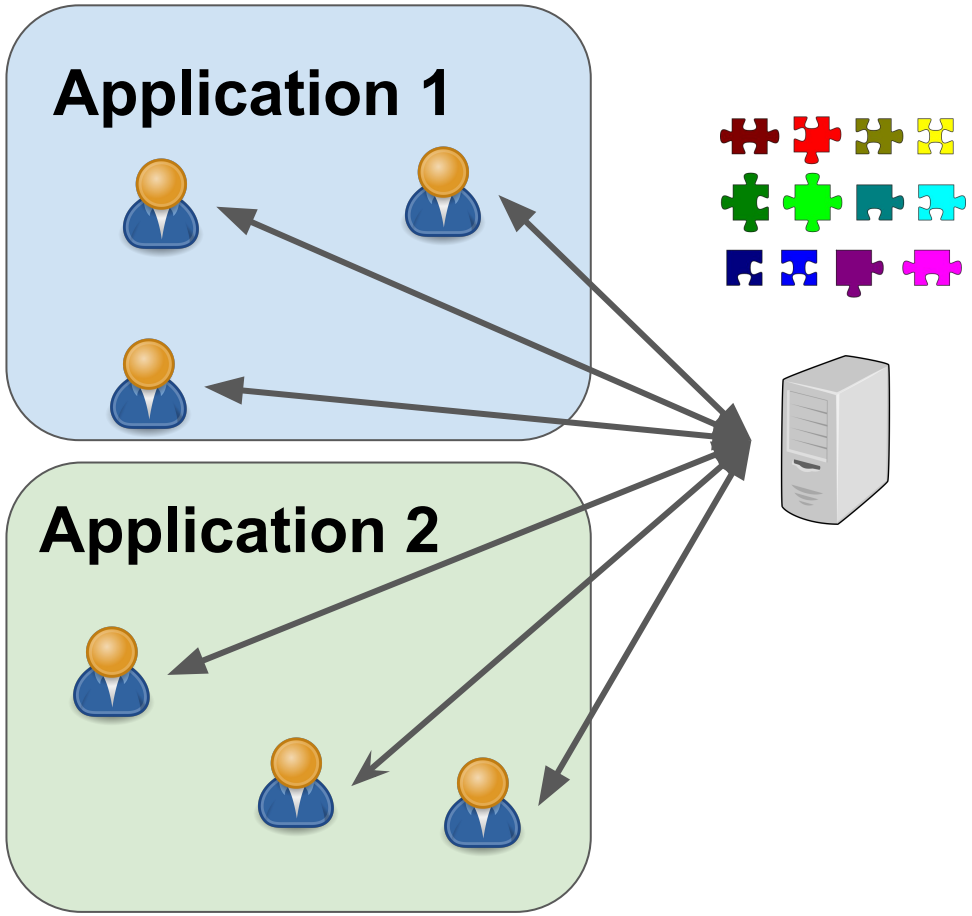
Cons

- May require multiple services
- Flexibility



Best of Both Worlds: Give Clients the Tools

- Single service
- Suitable for many different applications
- Simple for clients to implement
- Can tailor coordination to specific use-cases



What Kind of Tools Are Useful?

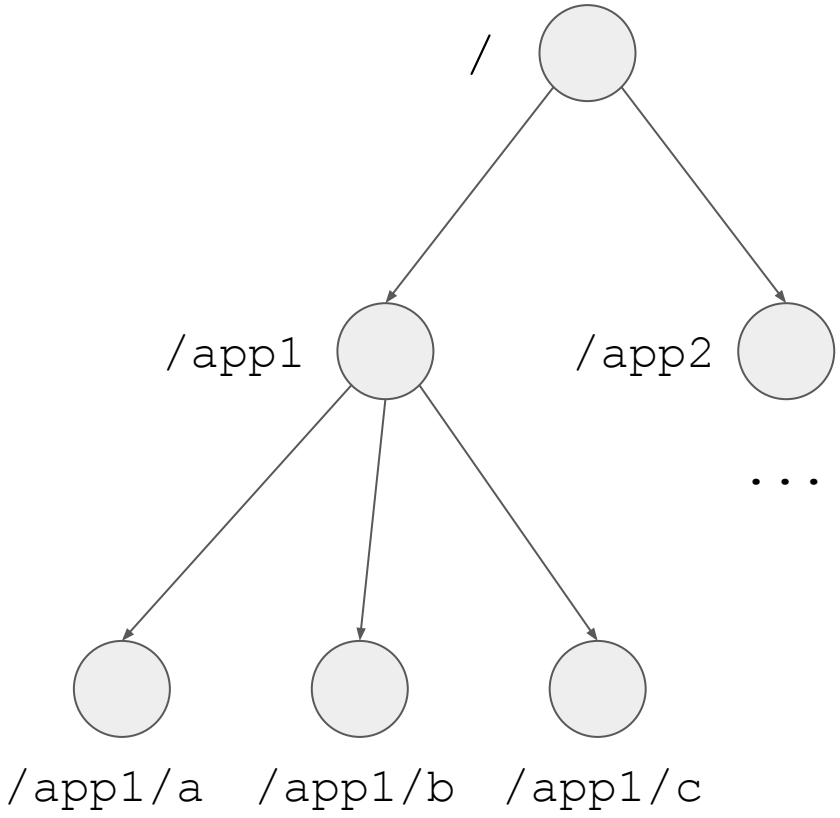
- Simple but Powerful
- High Performance
- Fault Tolerant
- Consistent

ZooKeeper

ZooKeeper gives clients the abstraction of a set of data nodes arranged in a hierarchical namespace.

ZooKeeper

ZooKeeper gives clients the abstraction of a set of data nodes arranged in a hierarchical namespace.

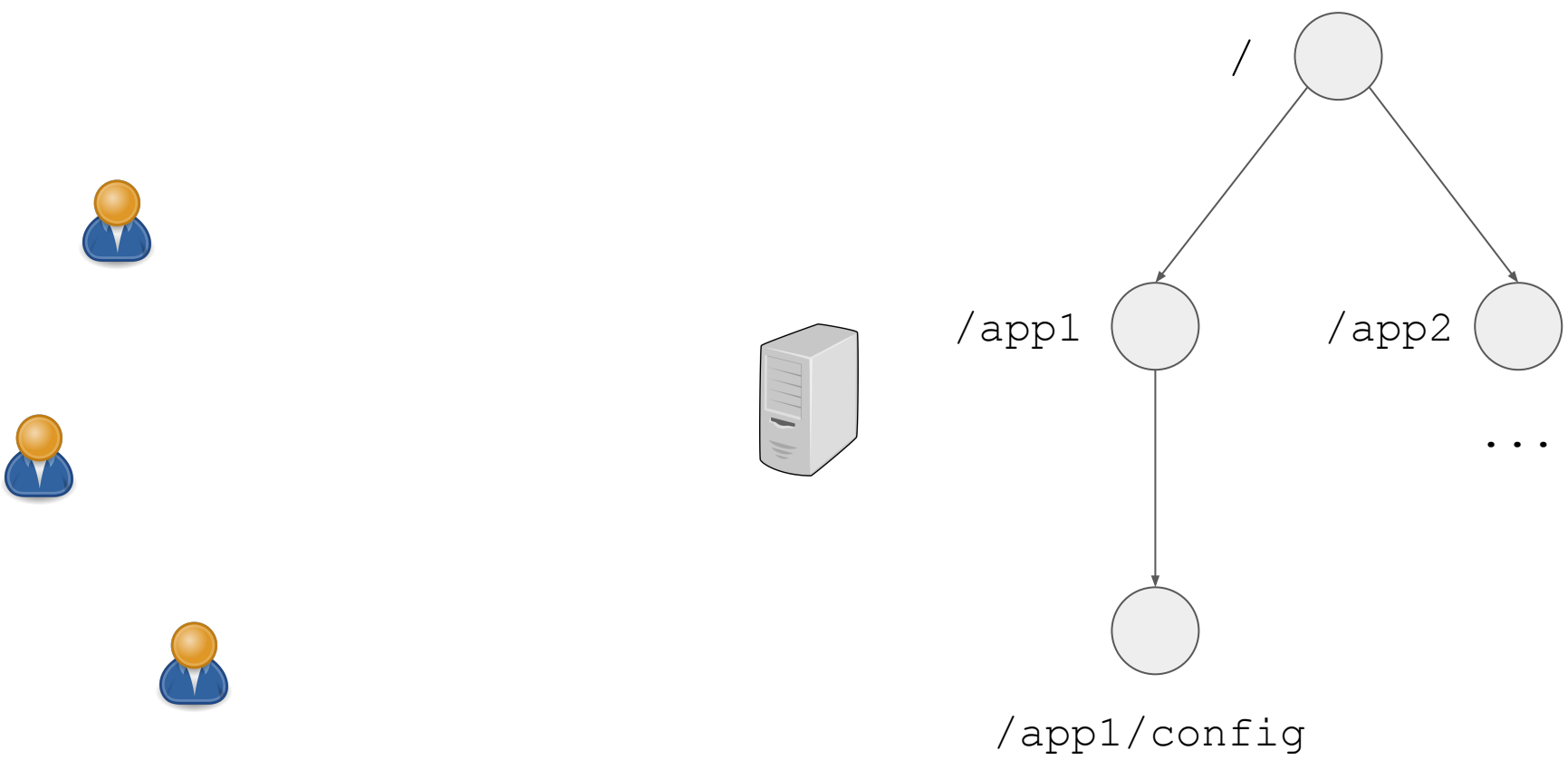


ZooKeeper

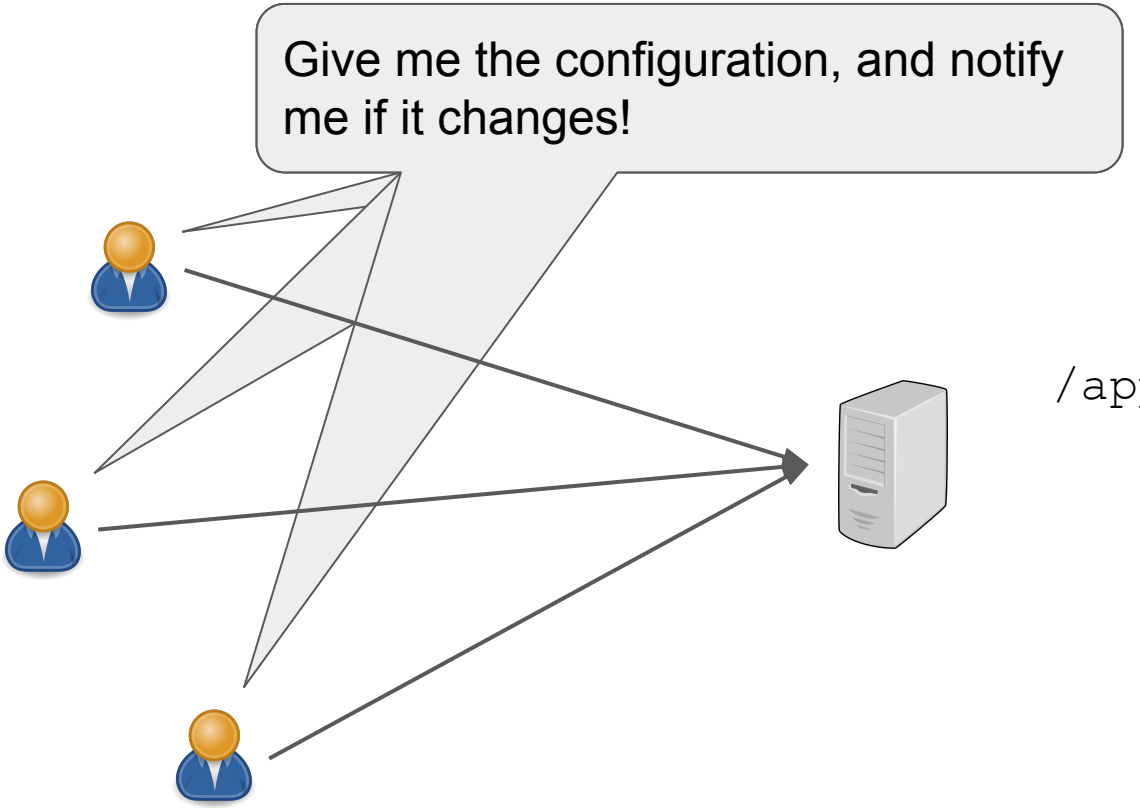
ZooKeeper gives clients the abstraction of a set of data nodes arranged in a hierarchical namespace.

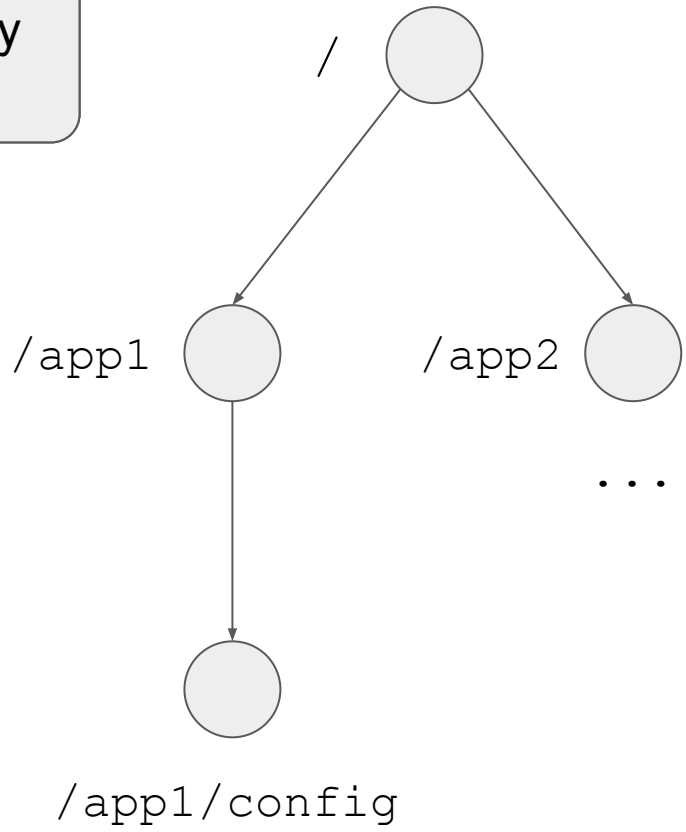
- **Watches** notify clients when nodes change.
- **Performance**
  - Wait-free
  - Optimized for reads
- **Consistency**
  - Linearizable writes
  - FIFO ordering for a given client

Configuration Management

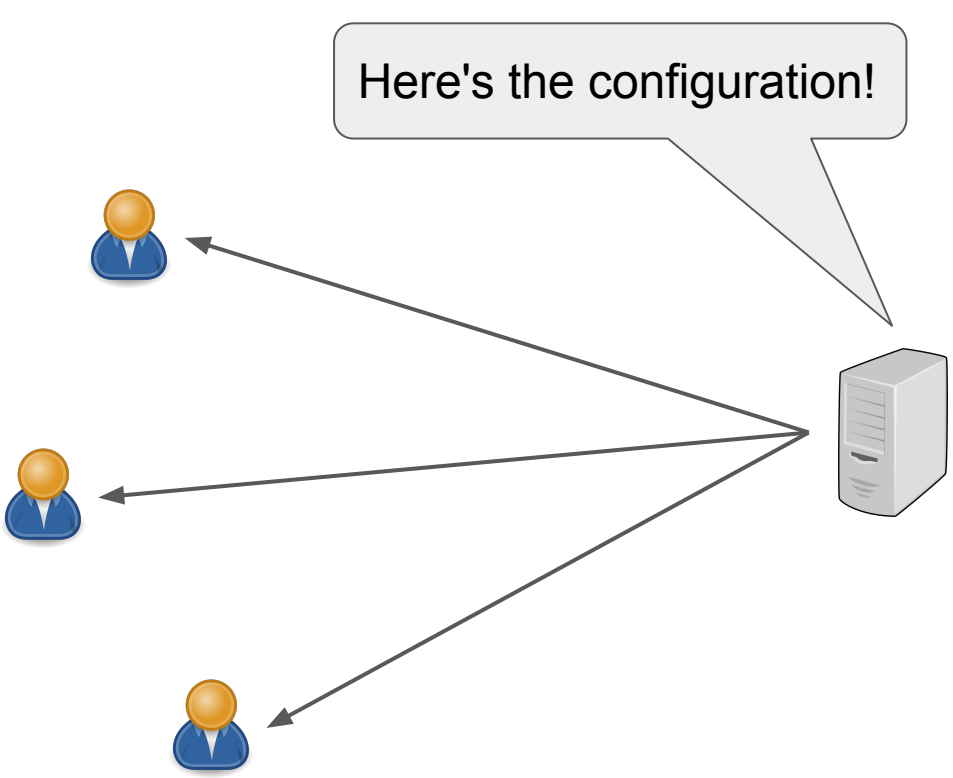


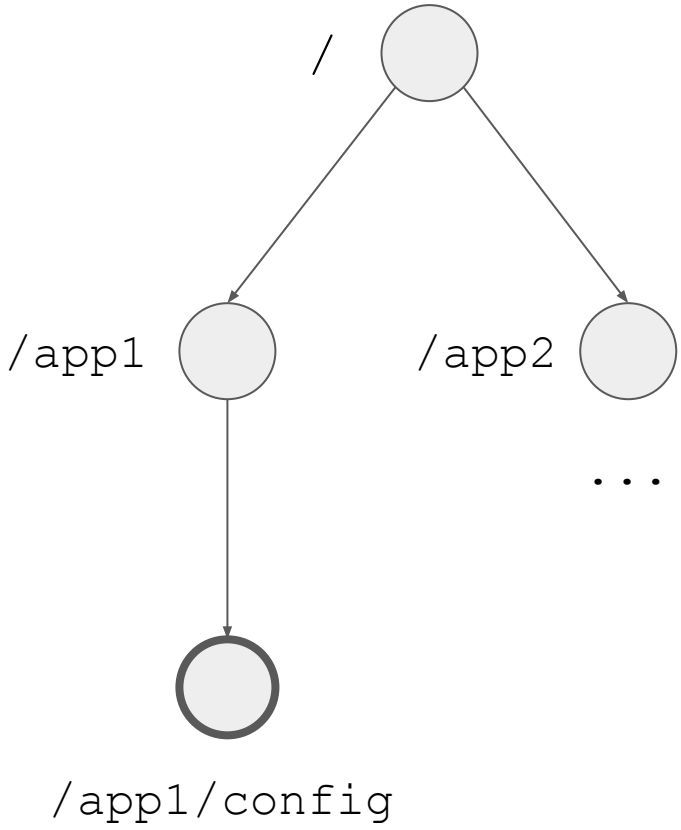
Configuration Management



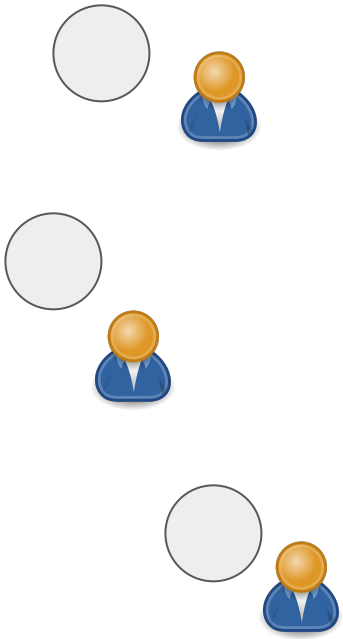


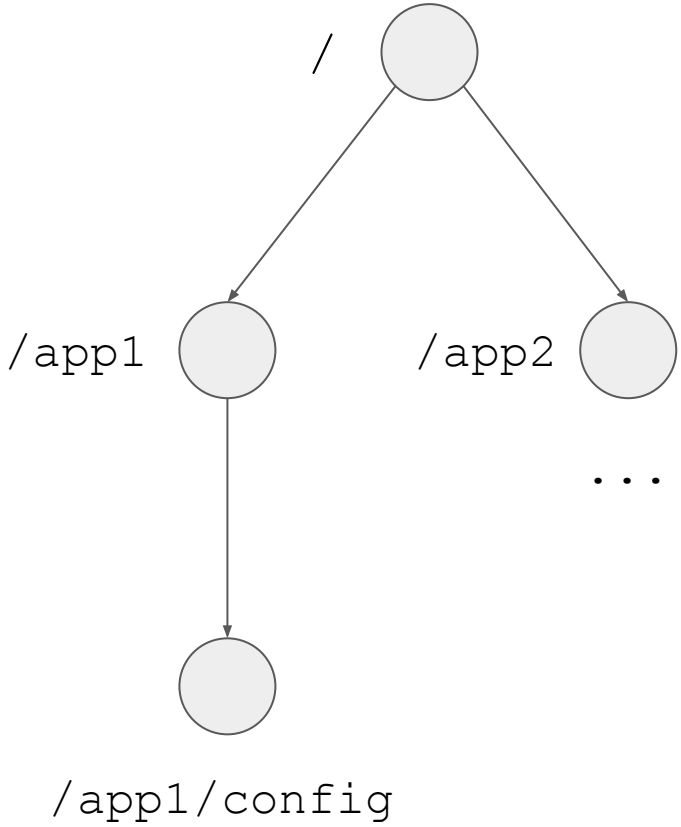
Configuration Management



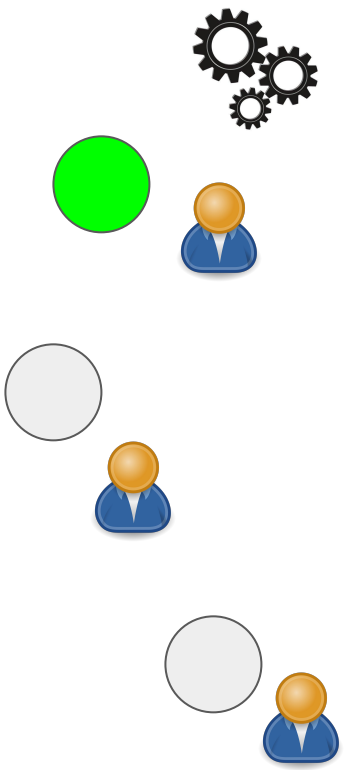


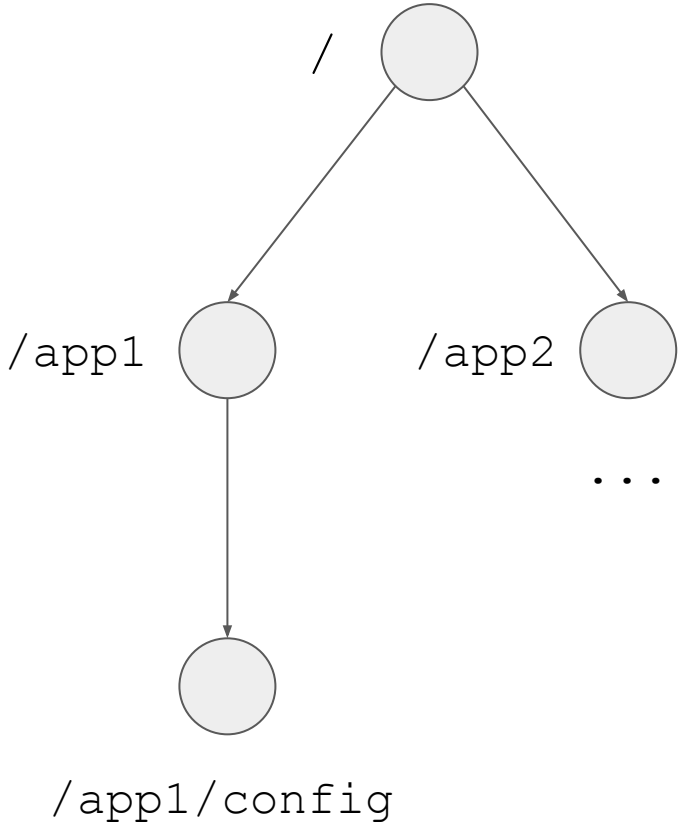
Configuration Management



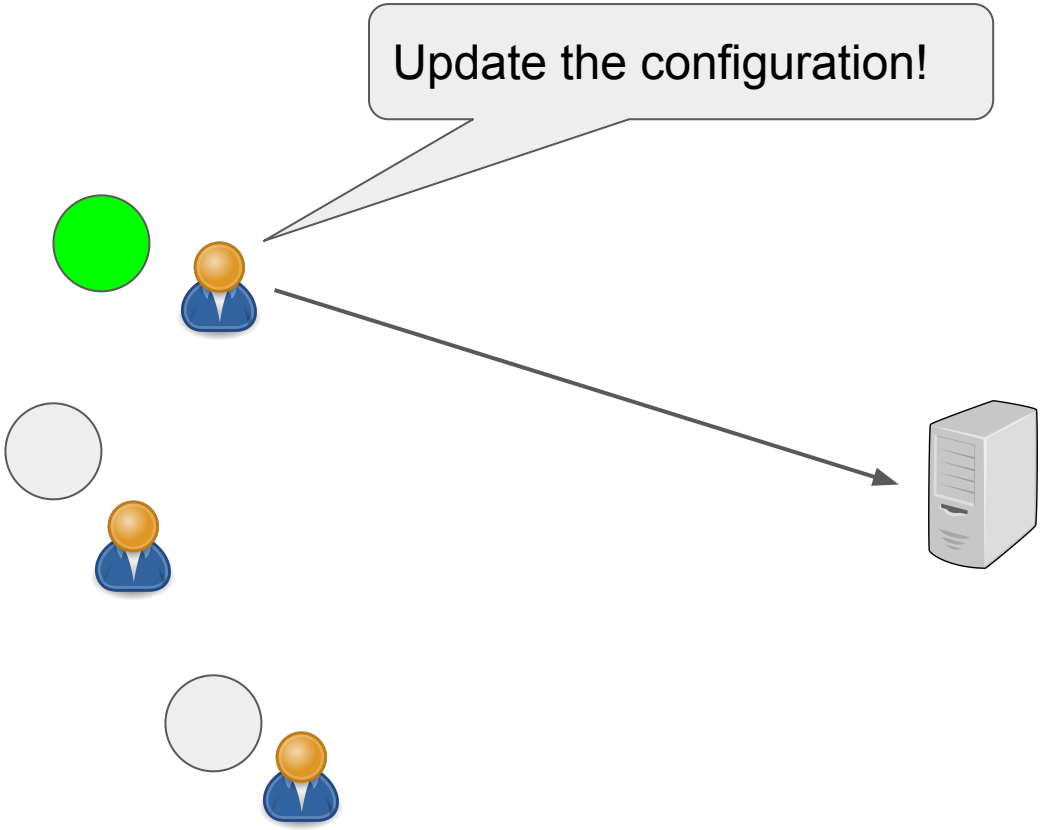


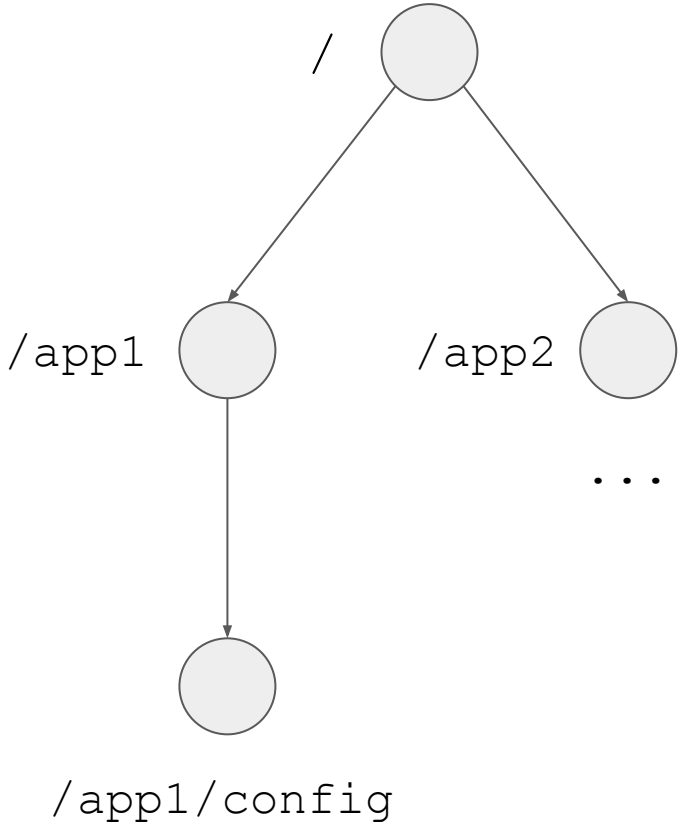
Configuration Management



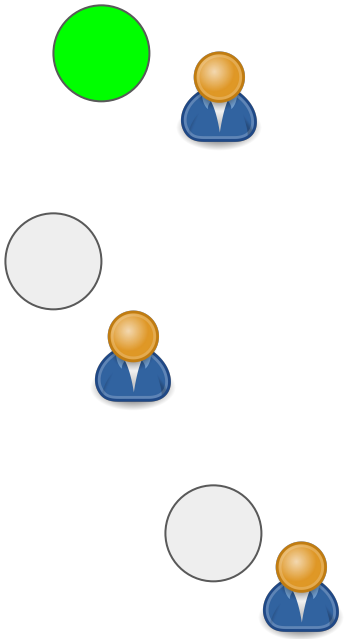


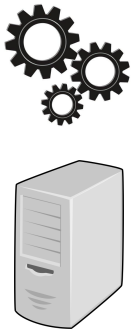
Configuration Management

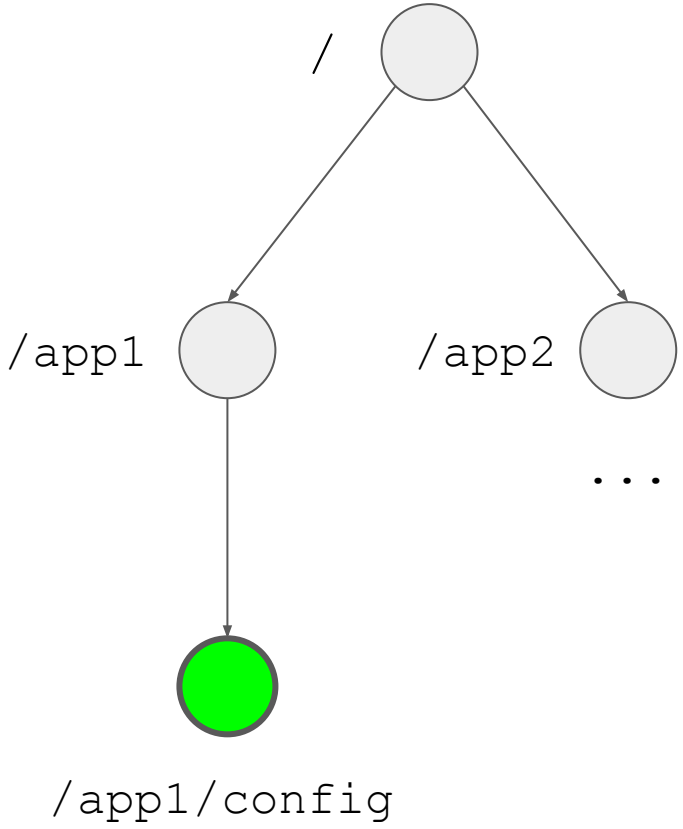




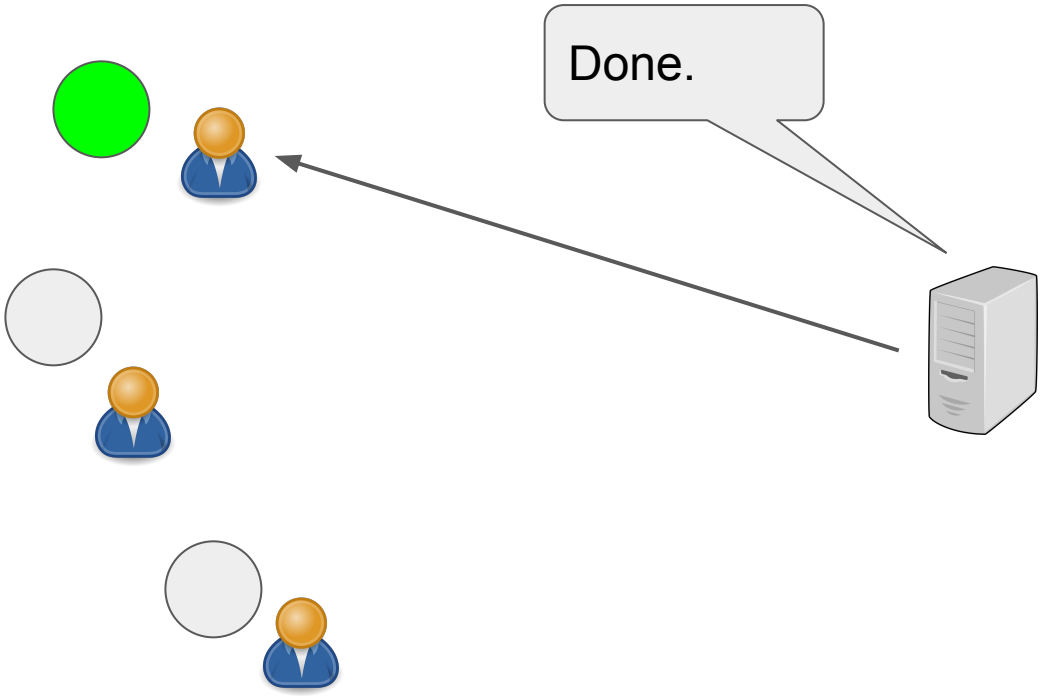
Configuration Management

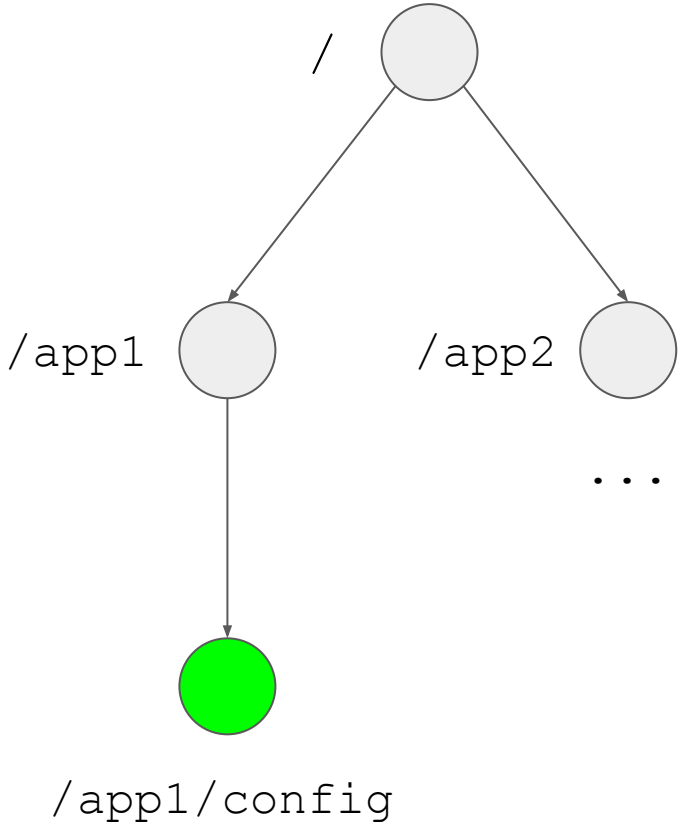




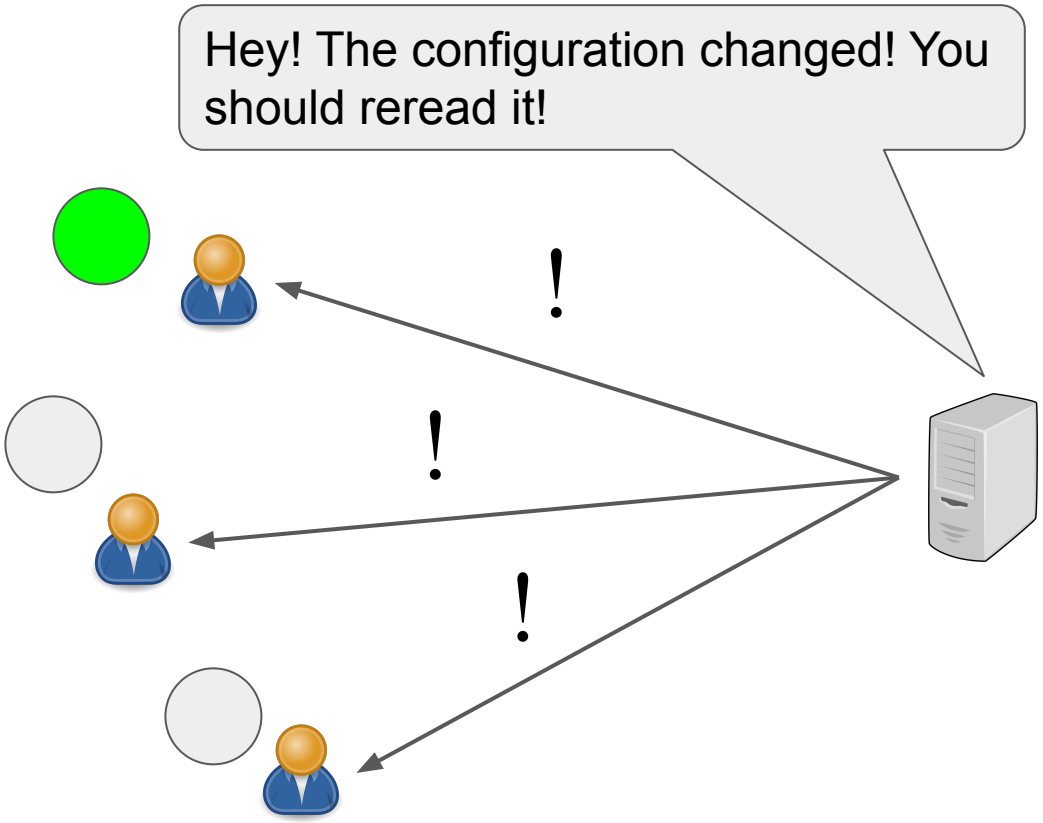


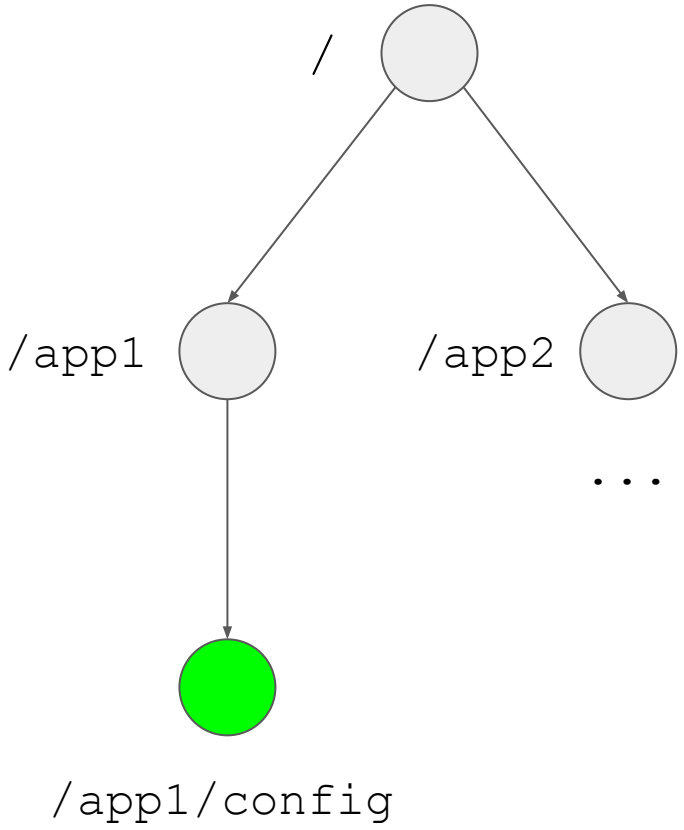
Configuration Management



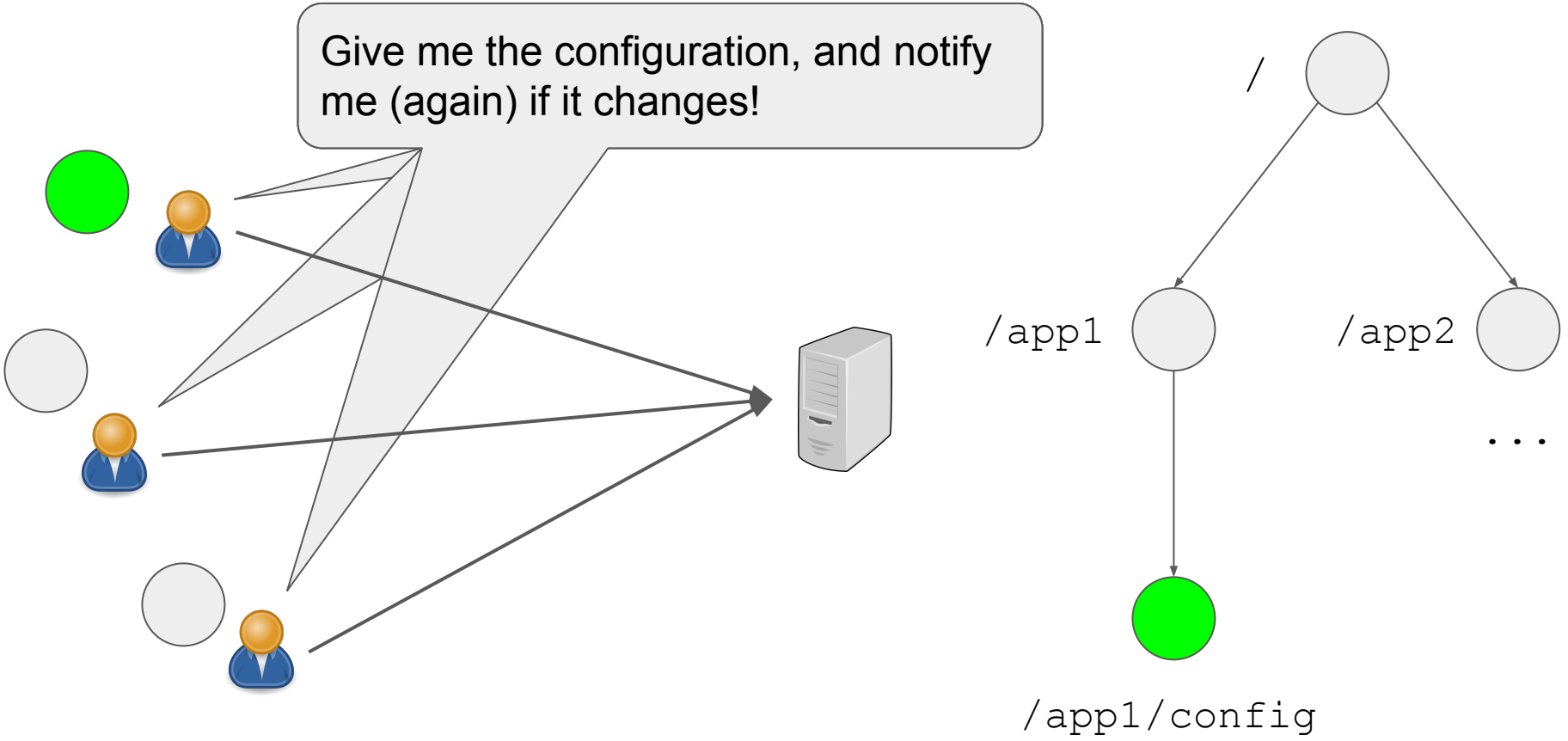


Configuration Management

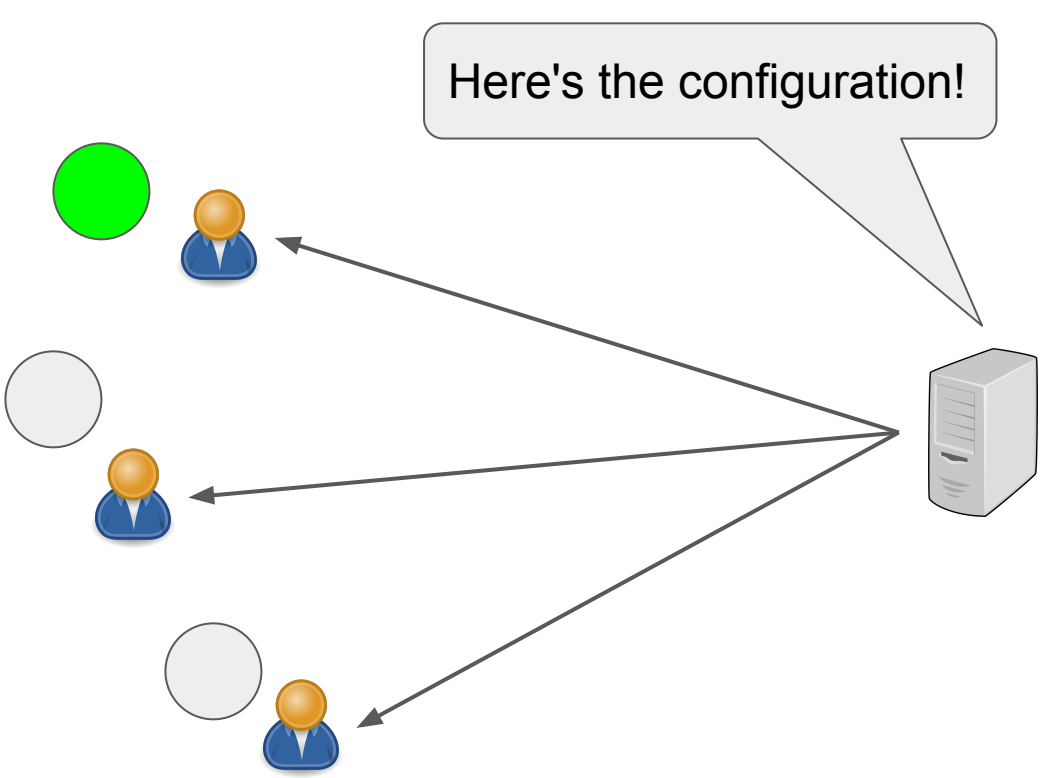


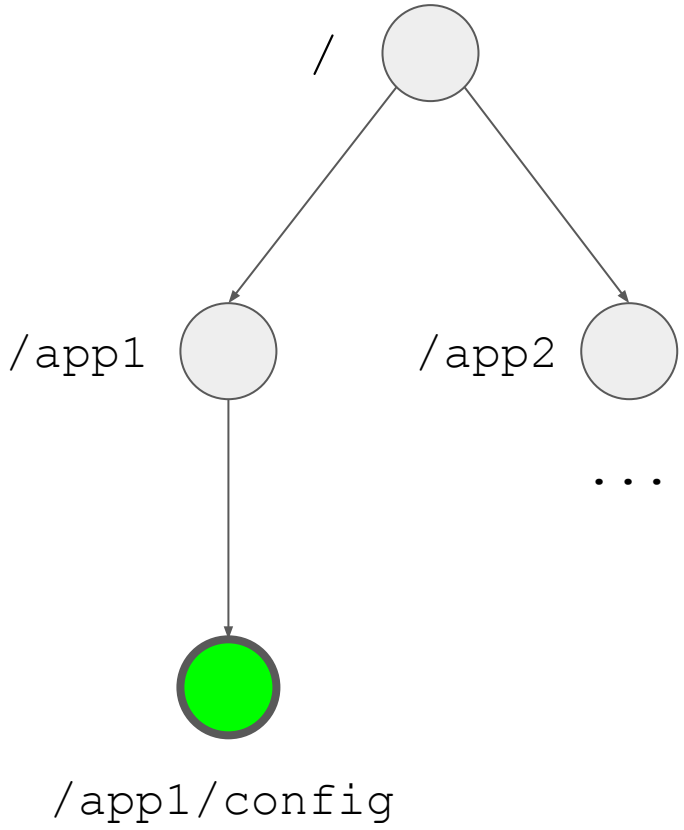


Configuration Management

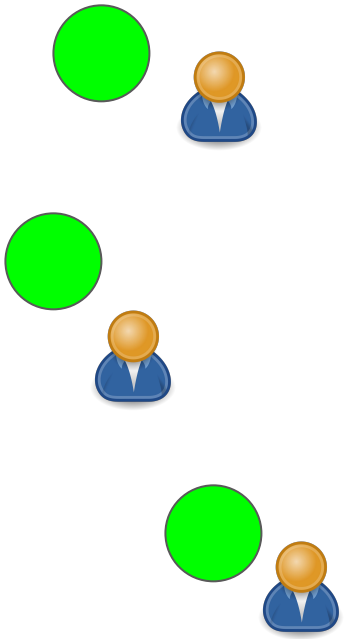


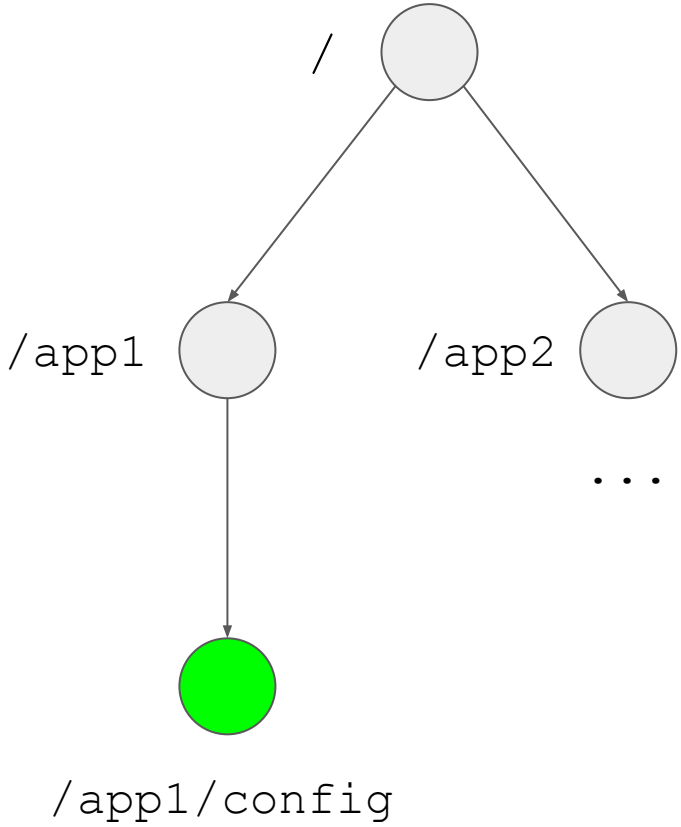
Configuration Management



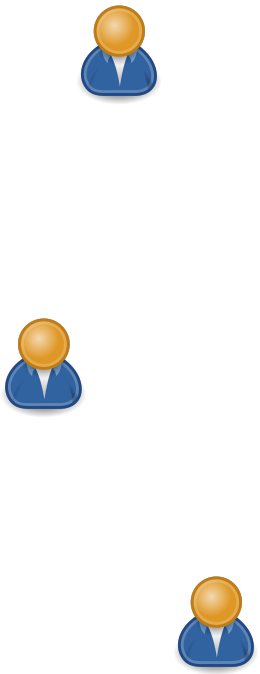


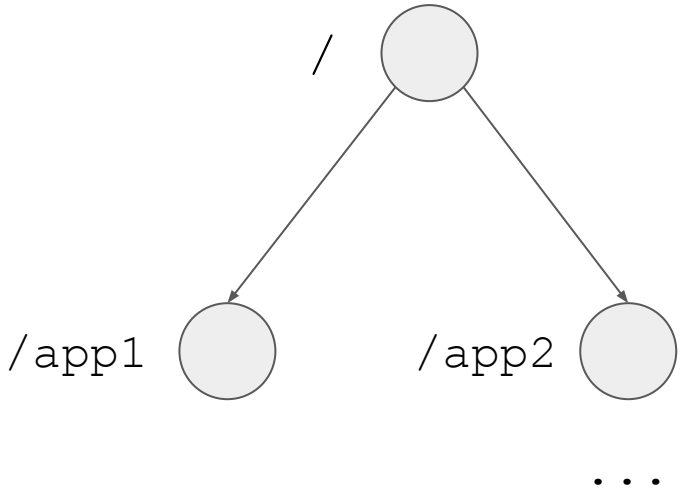
Configuration Management



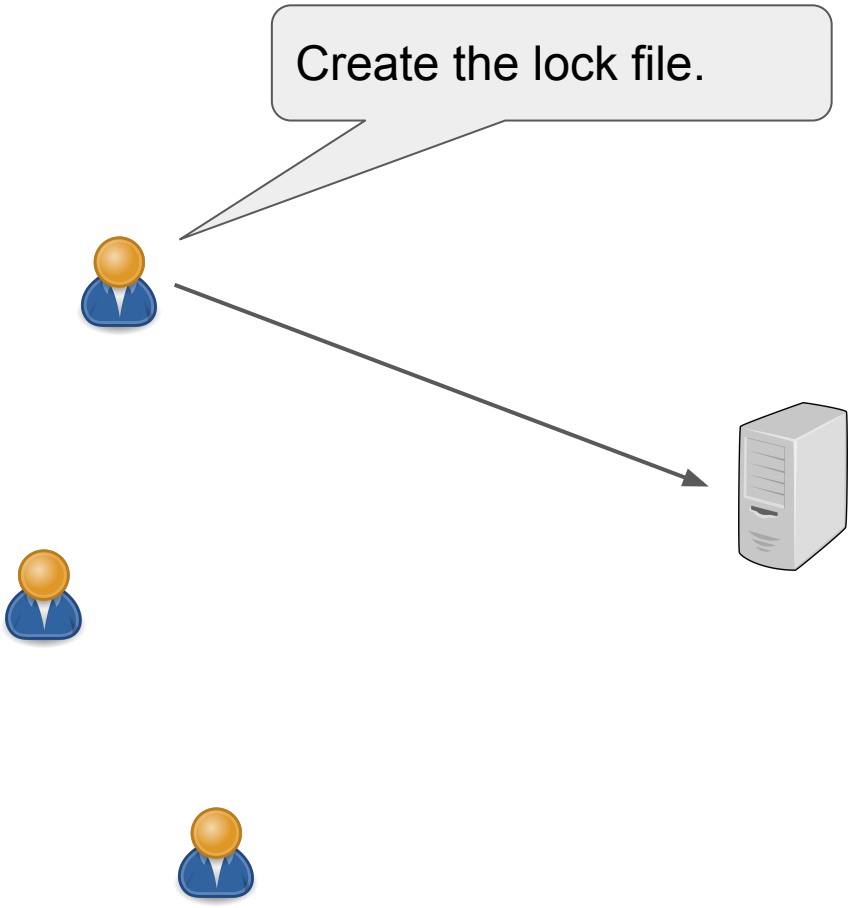


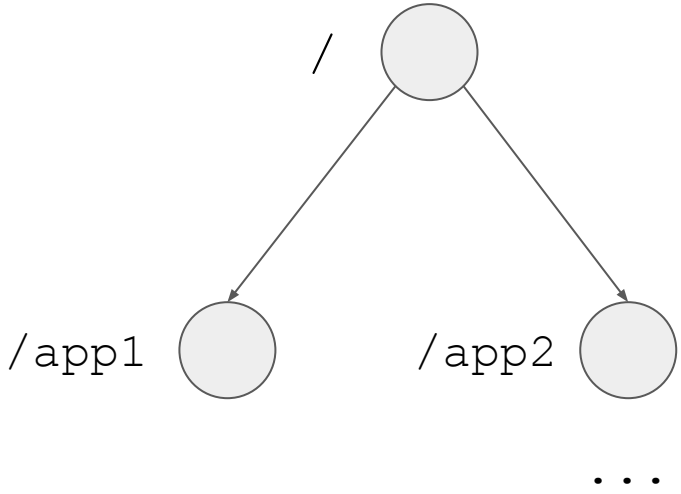
Locks



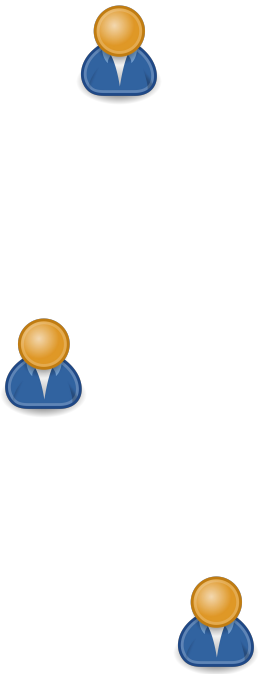


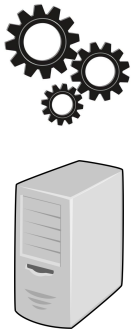
Locks

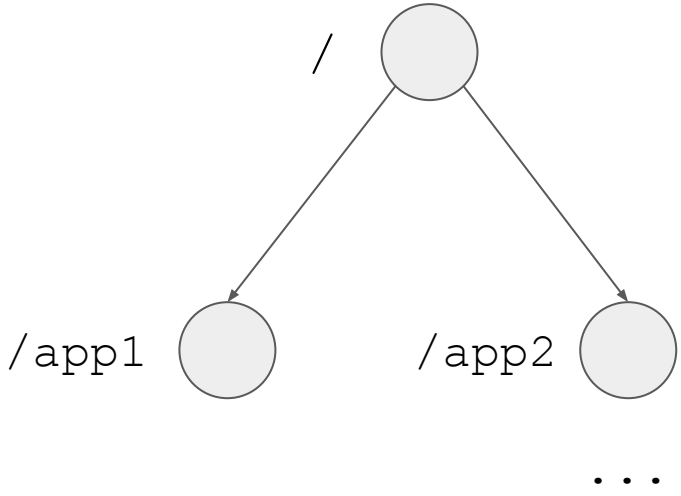




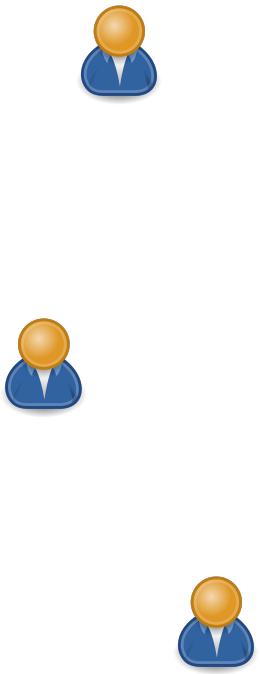
Locks

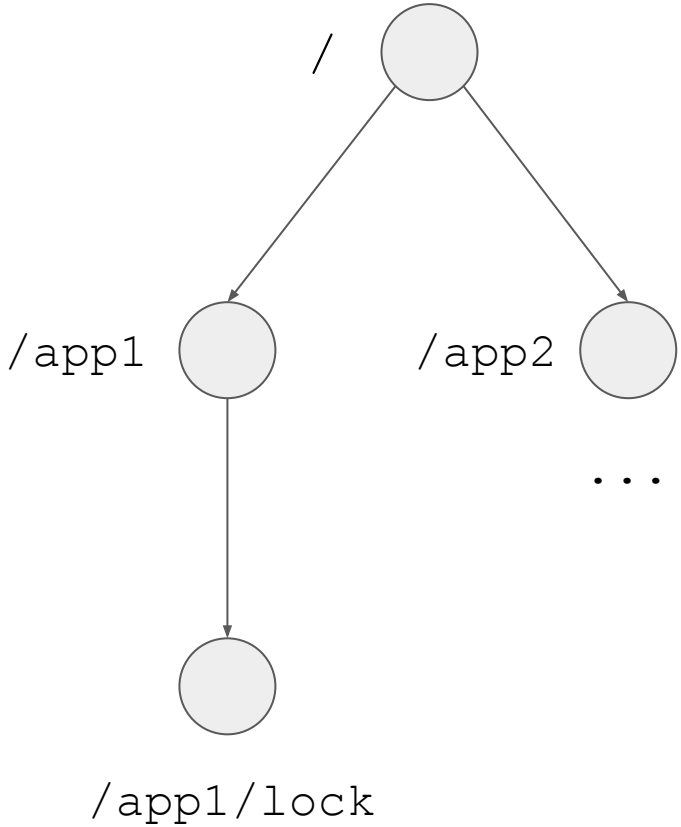




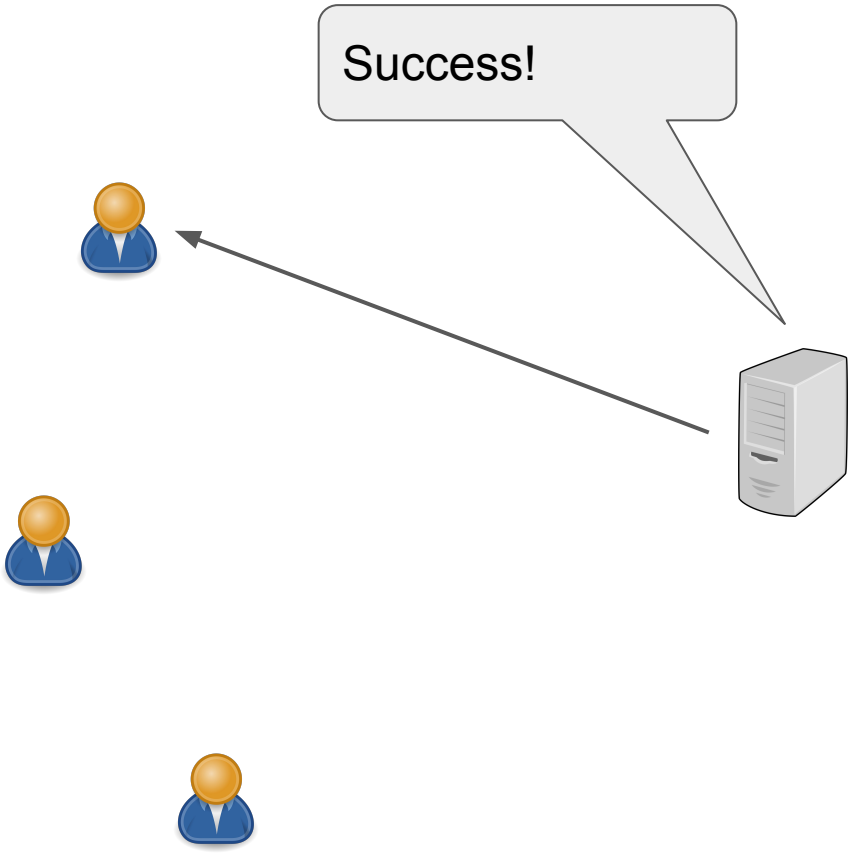


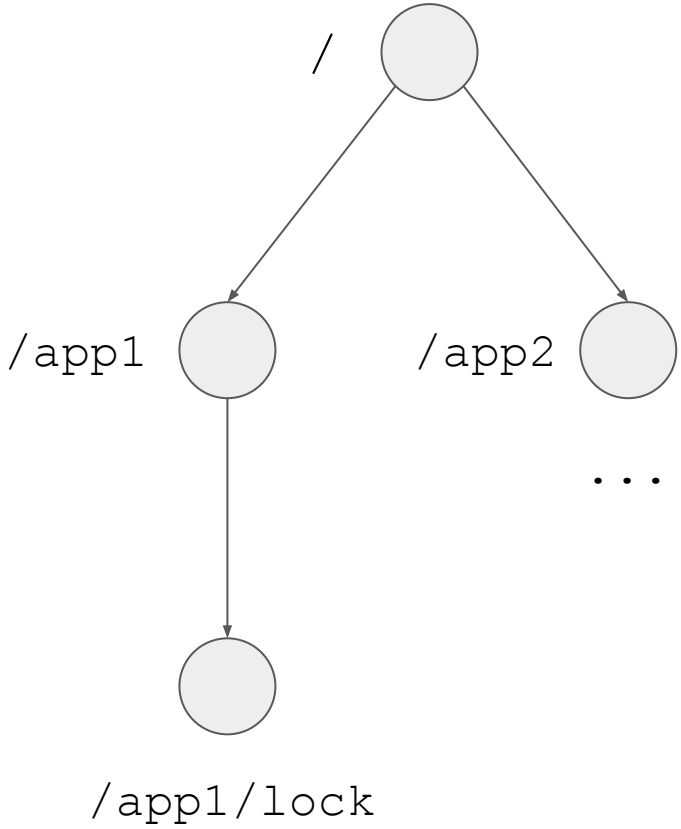
Locks



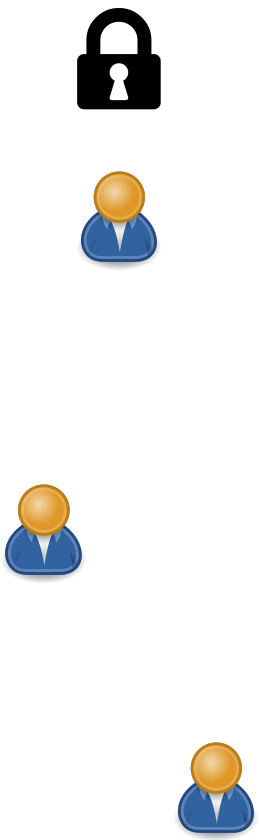


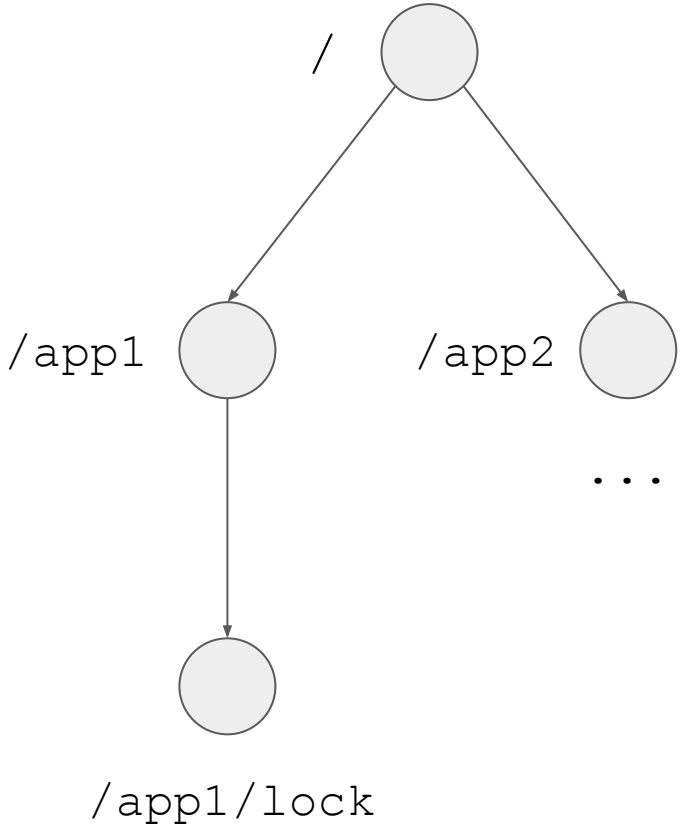
Locks



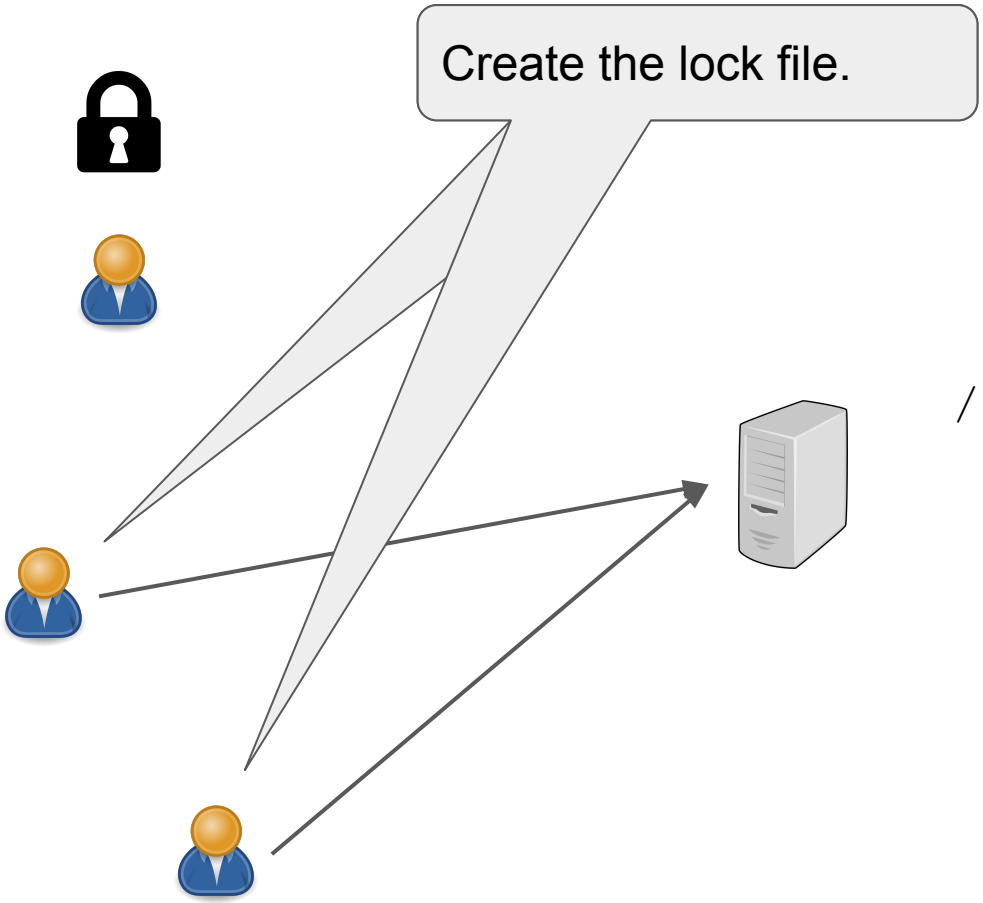


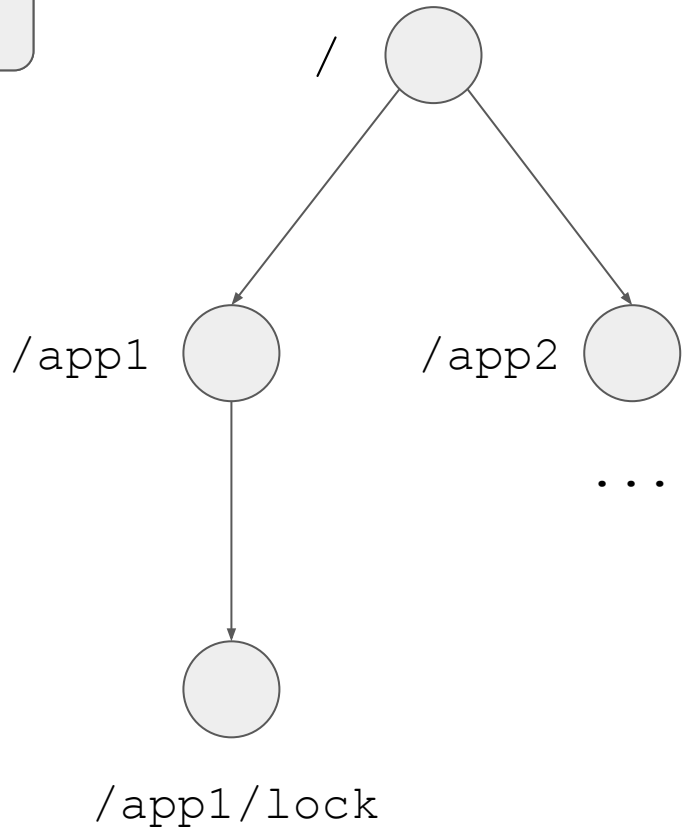
Locks



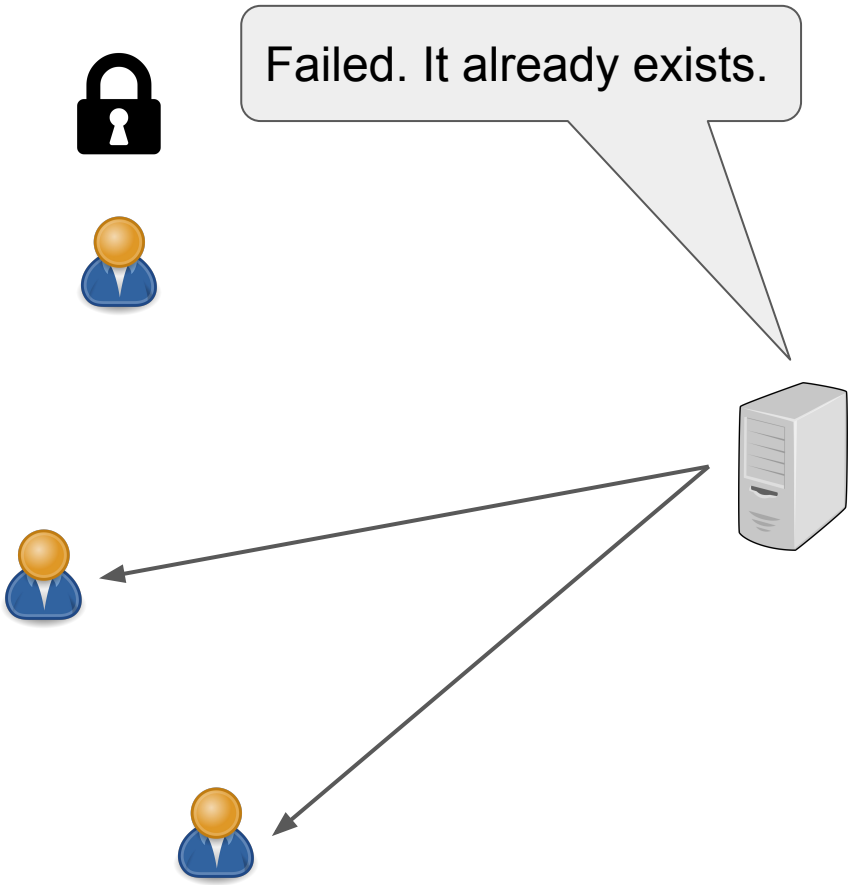


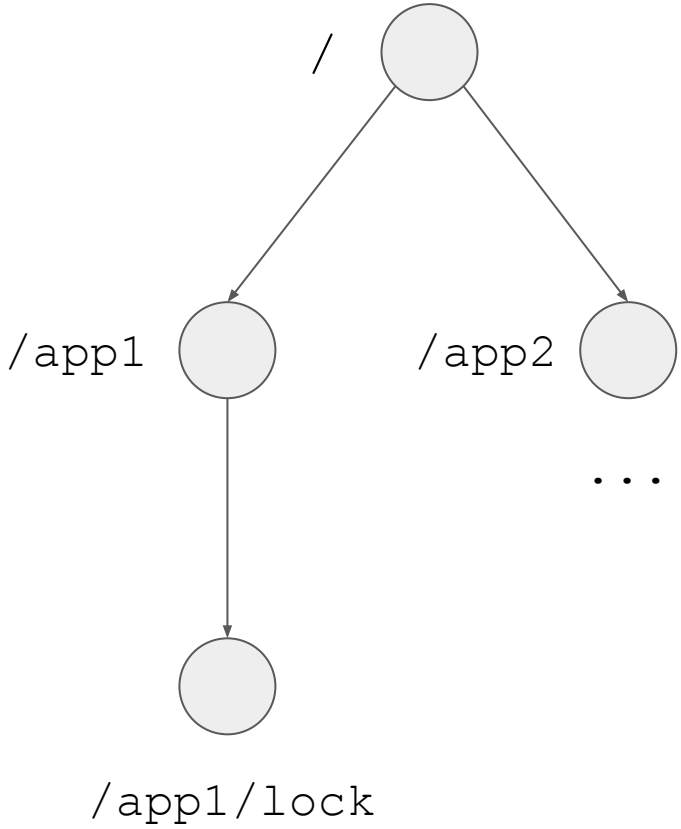
Locks



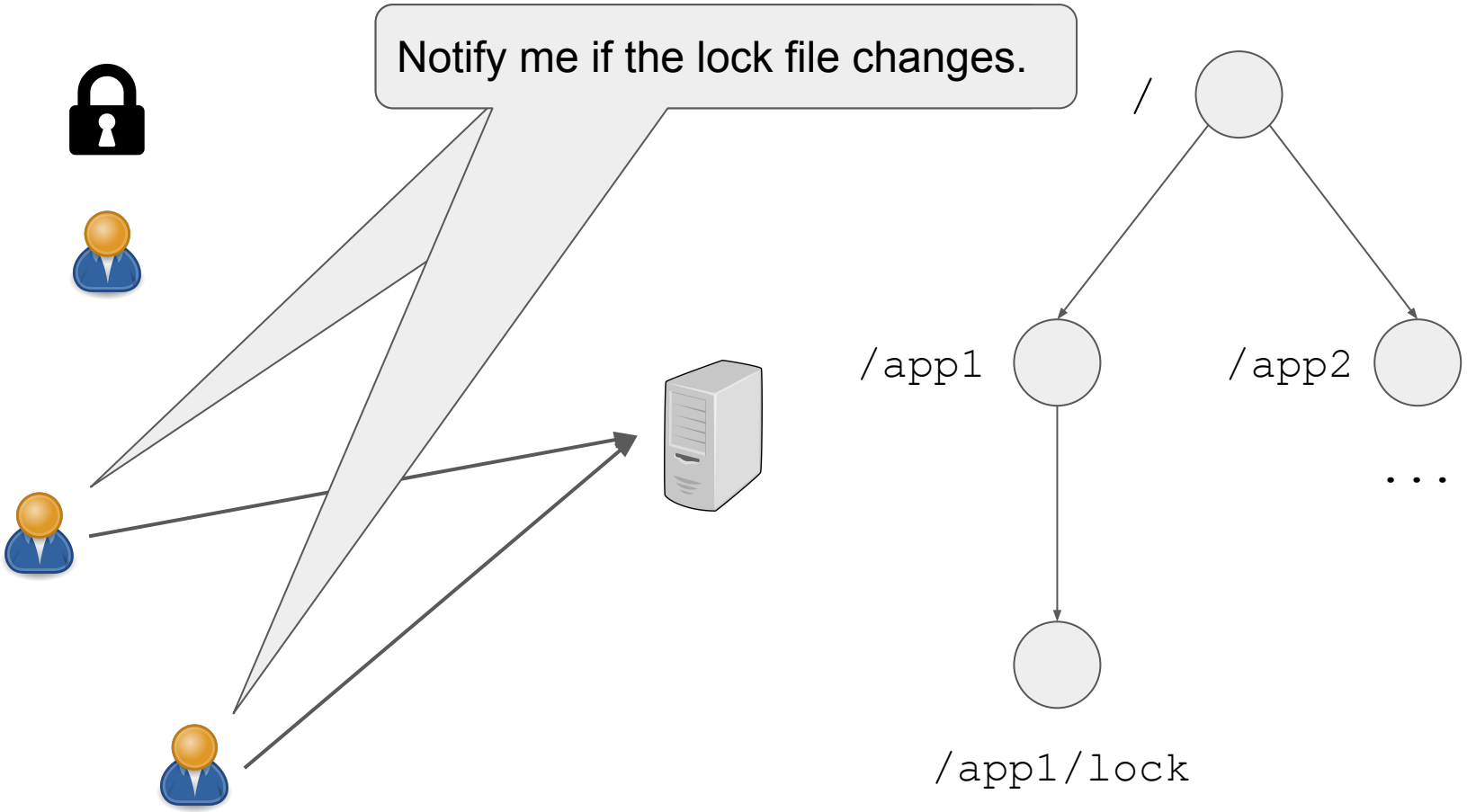


Locks

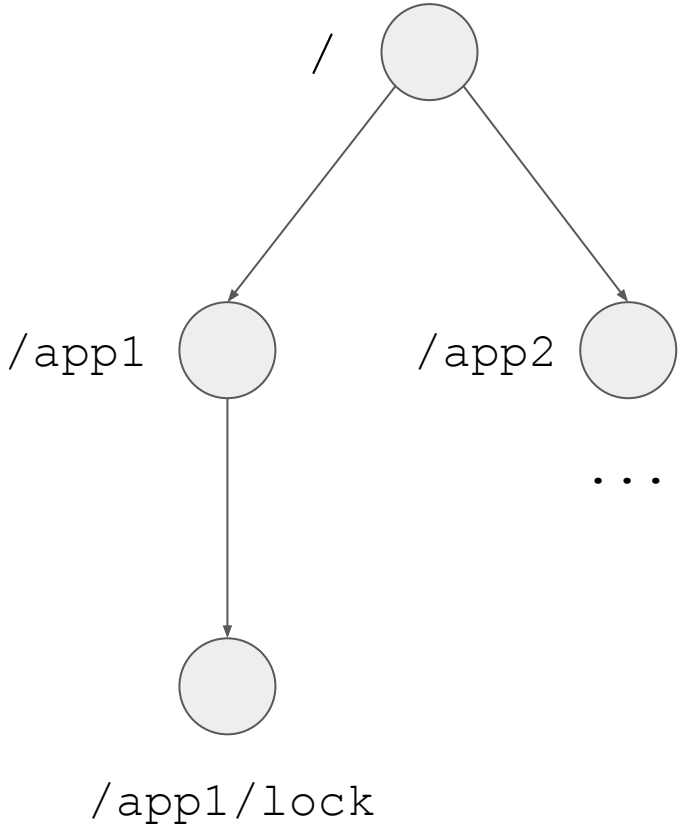




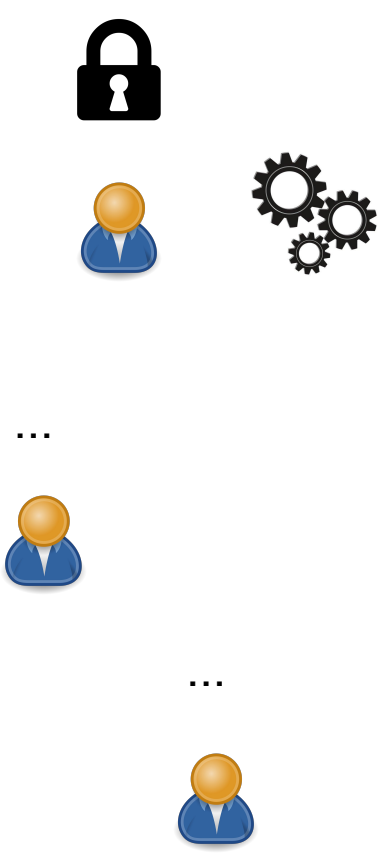
Locks

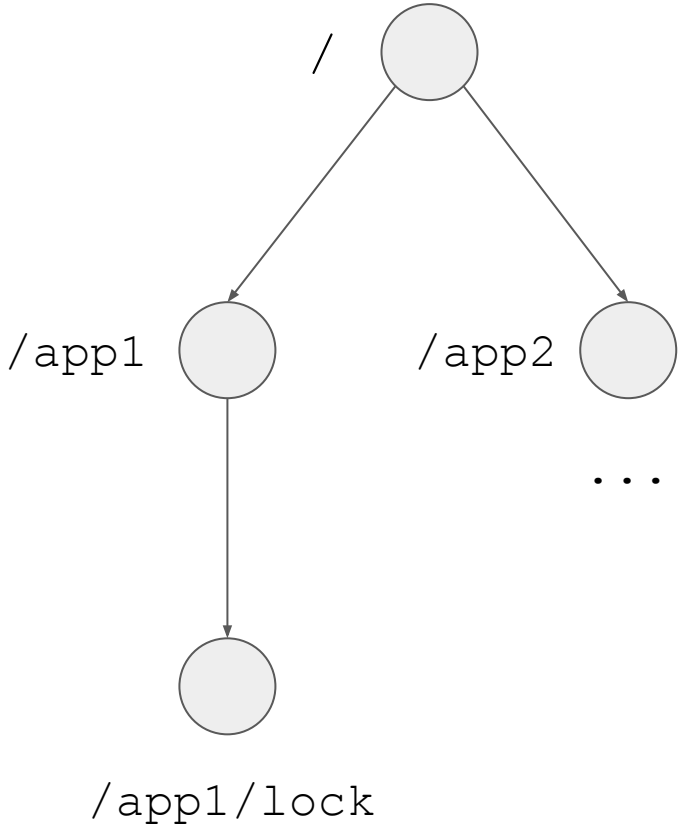


Locks

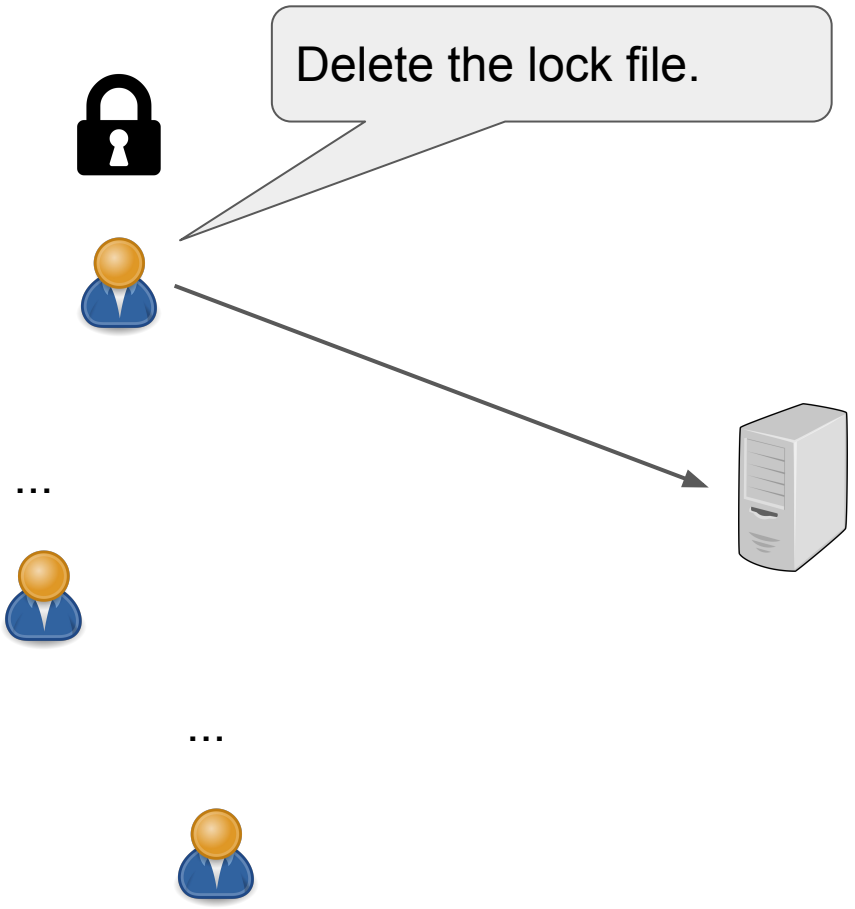


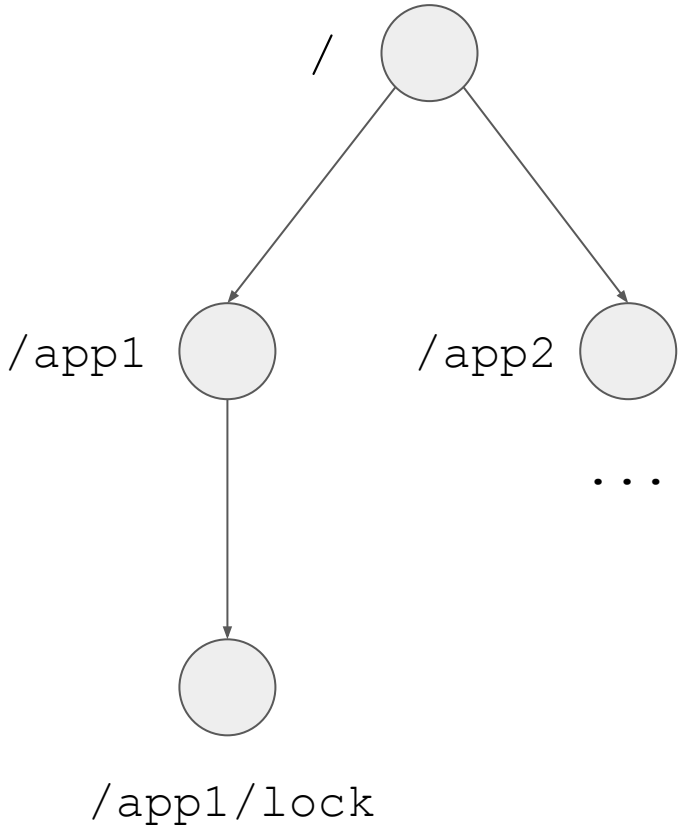
Locks



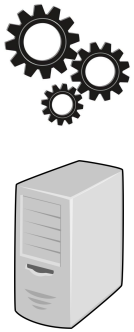


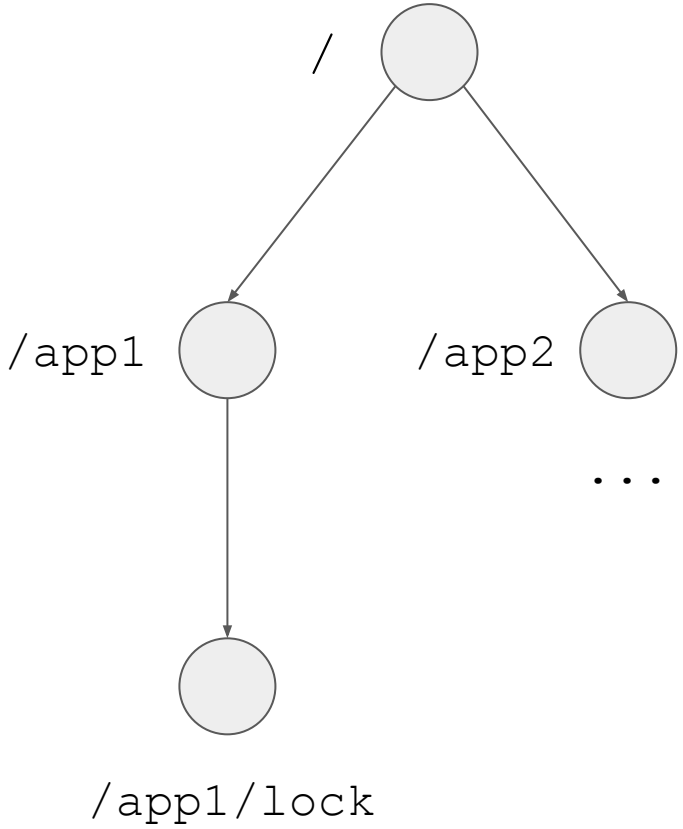
Locks



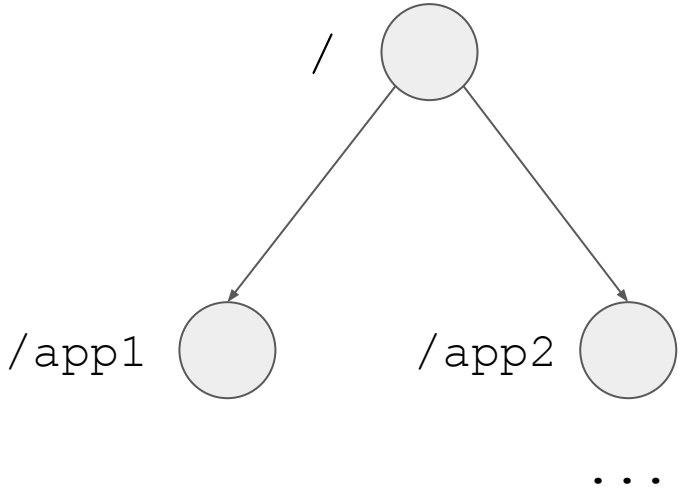


Locks

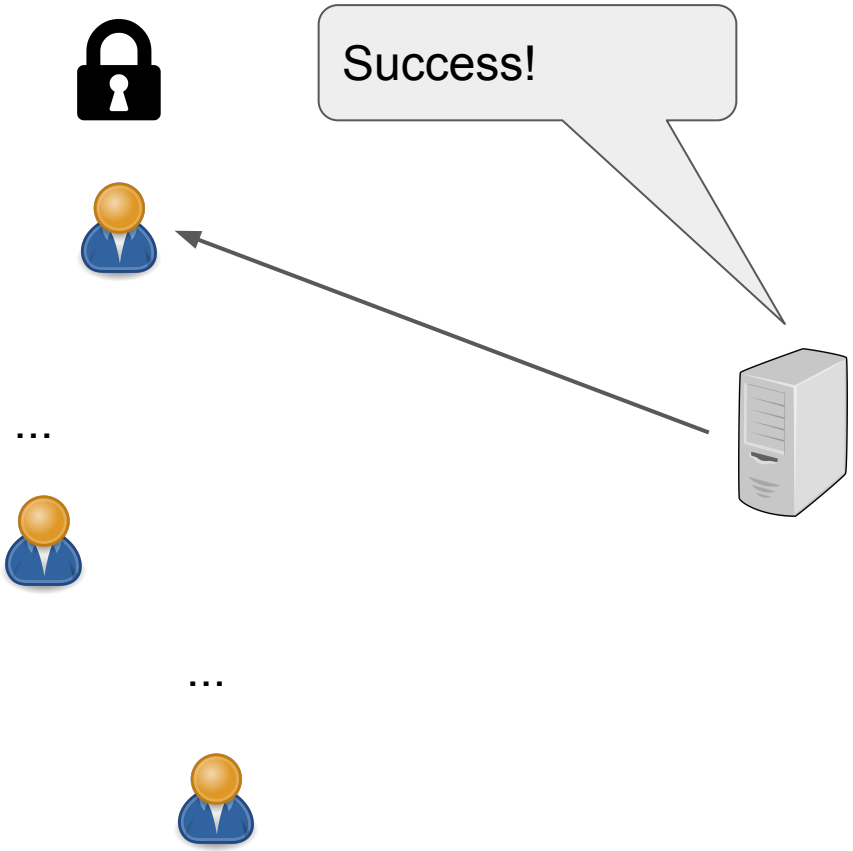


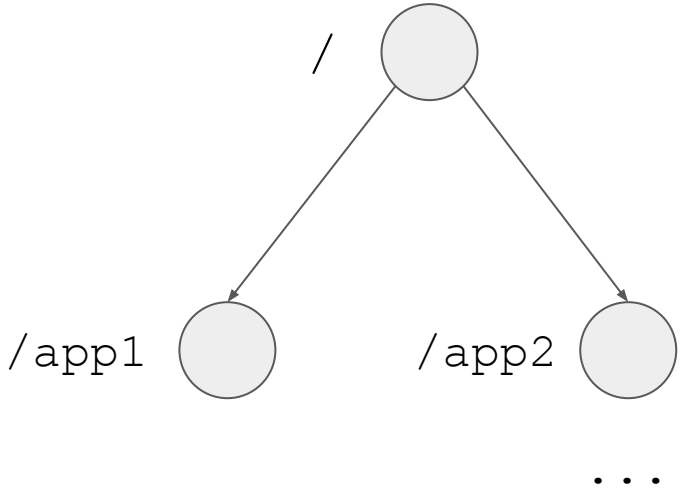


Locks

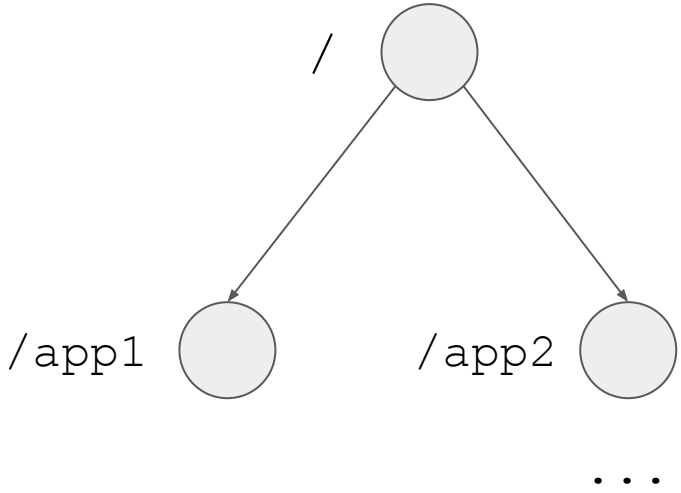


Locks

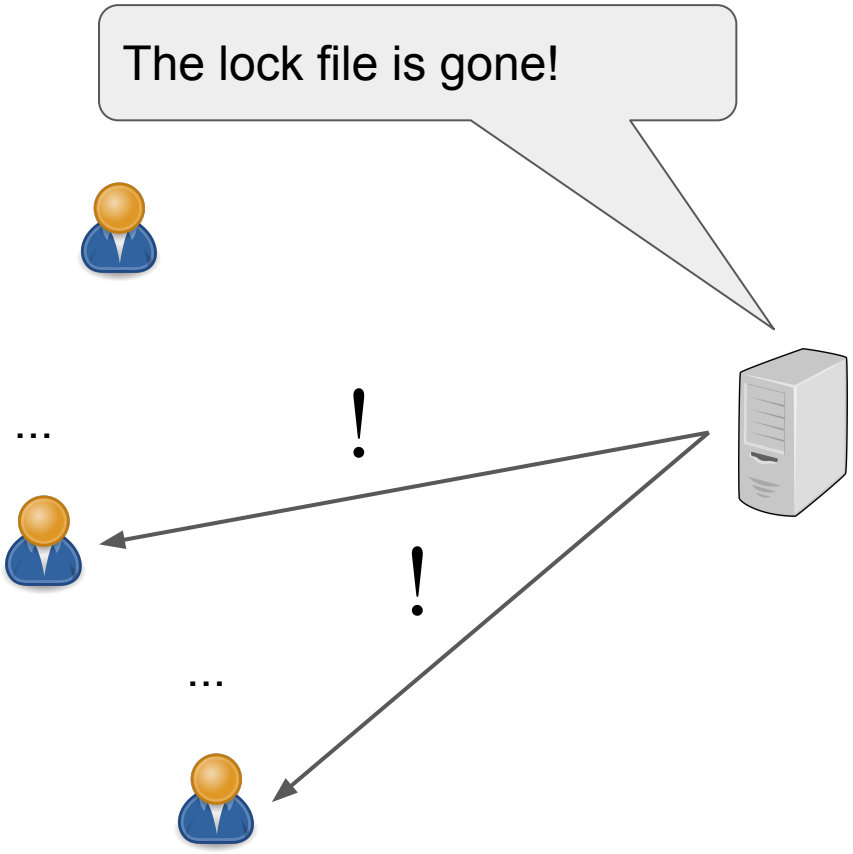


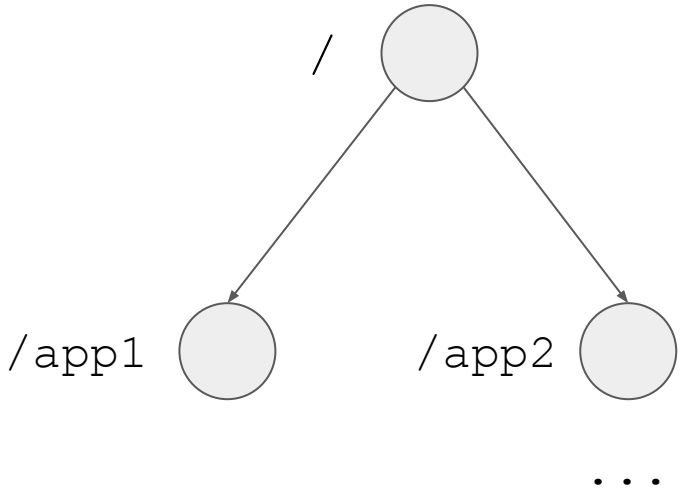


Locks

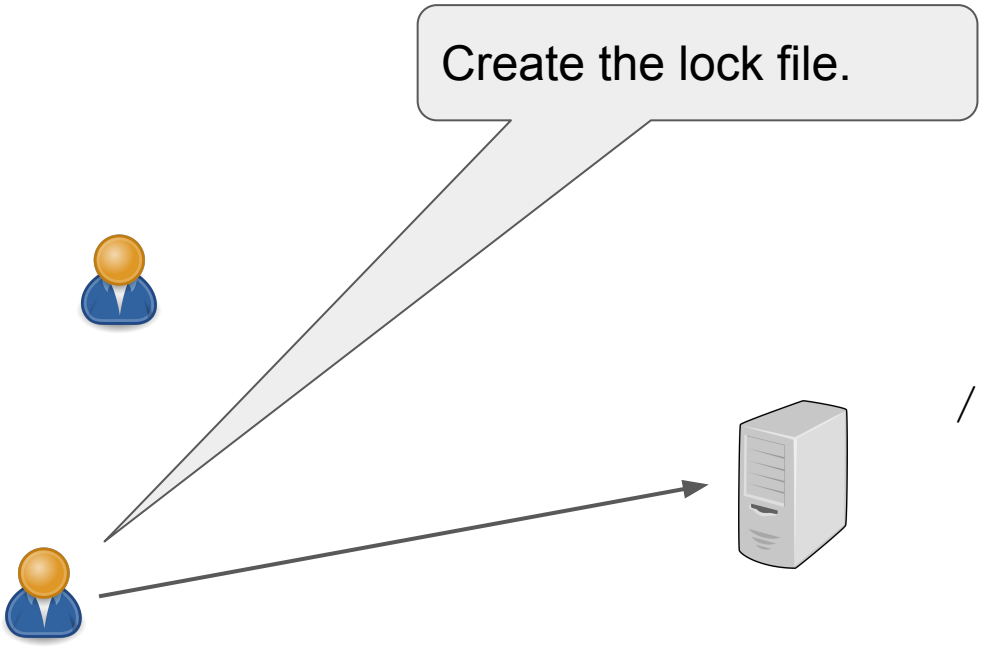


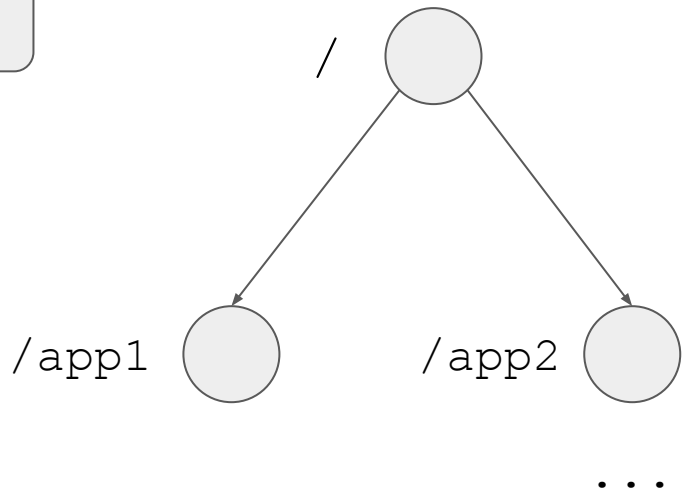
Locks



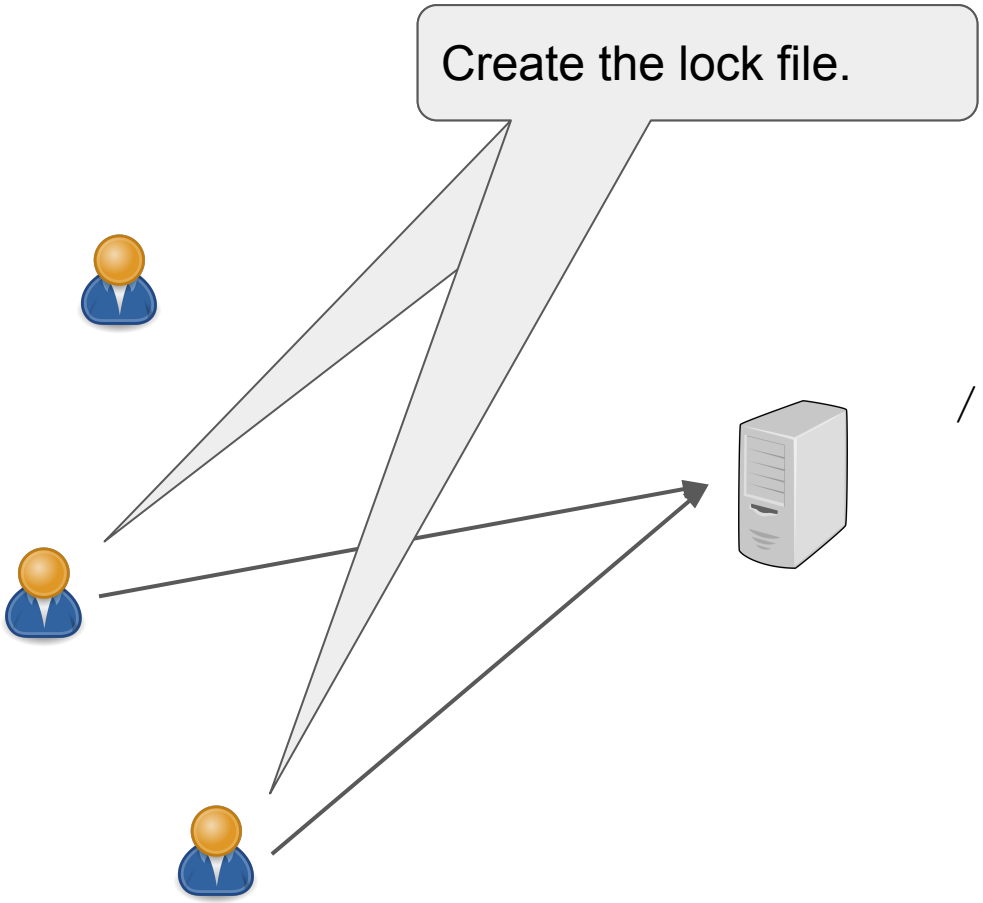


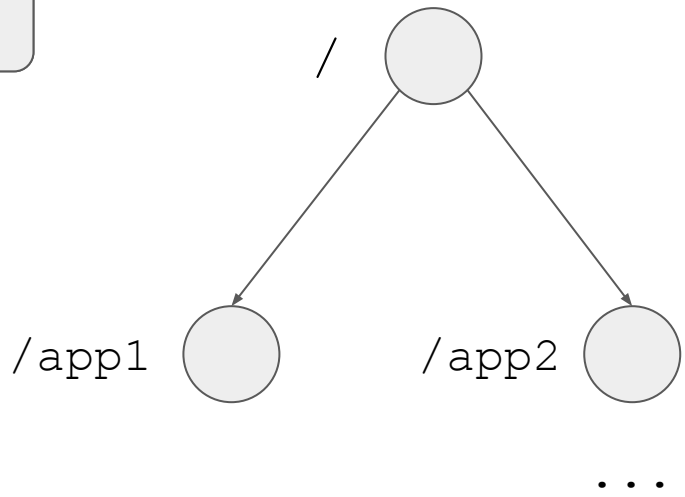
Locks



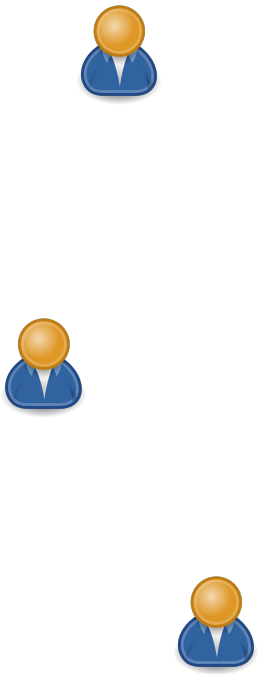


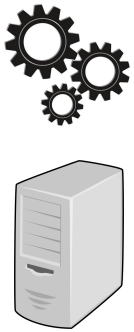
Locks

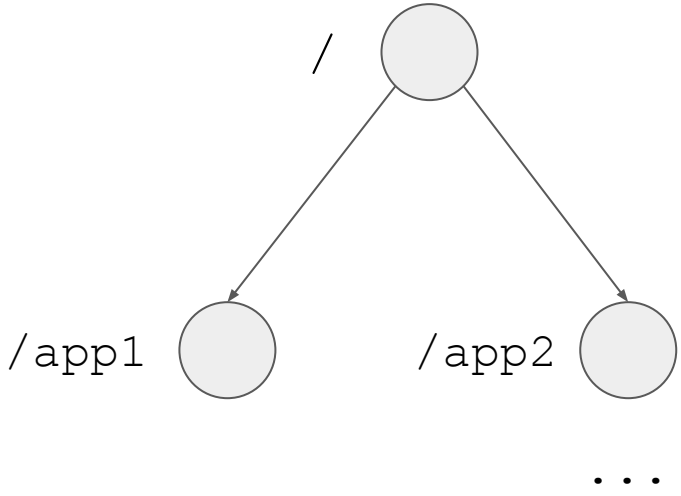




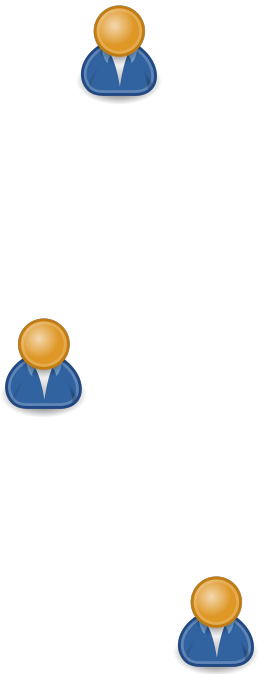
Locks

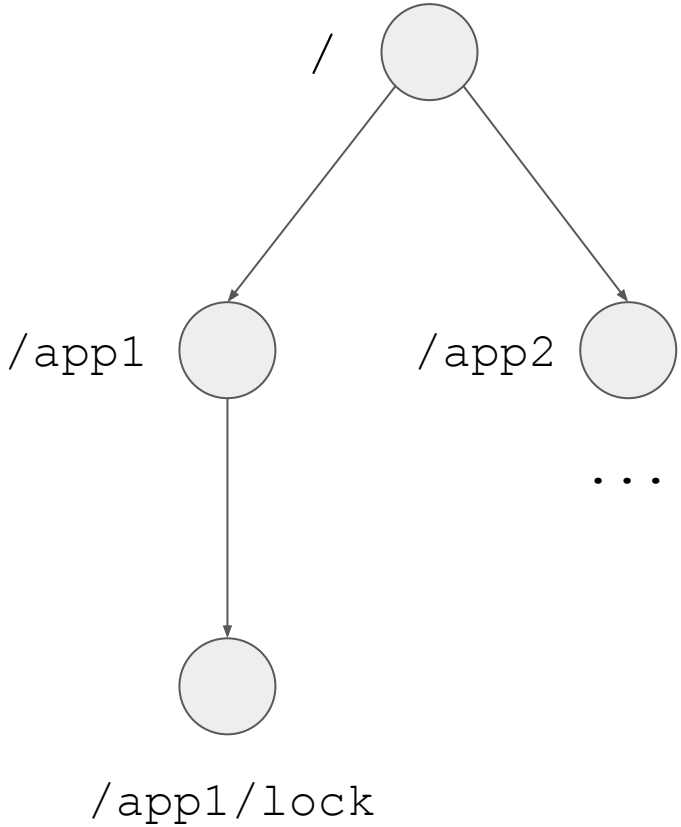




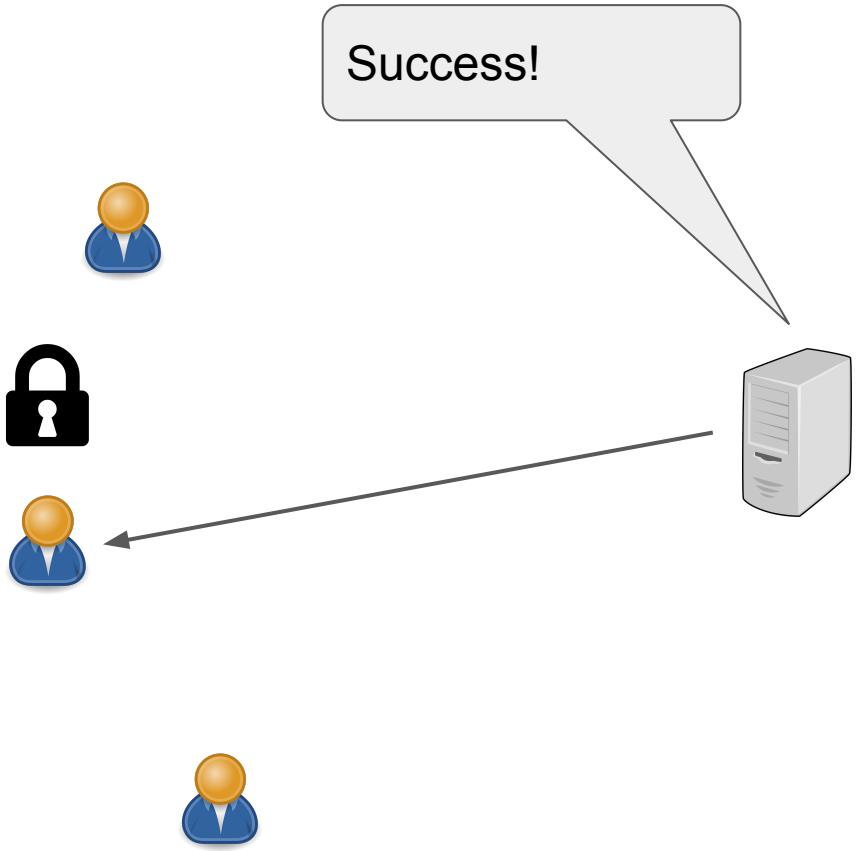


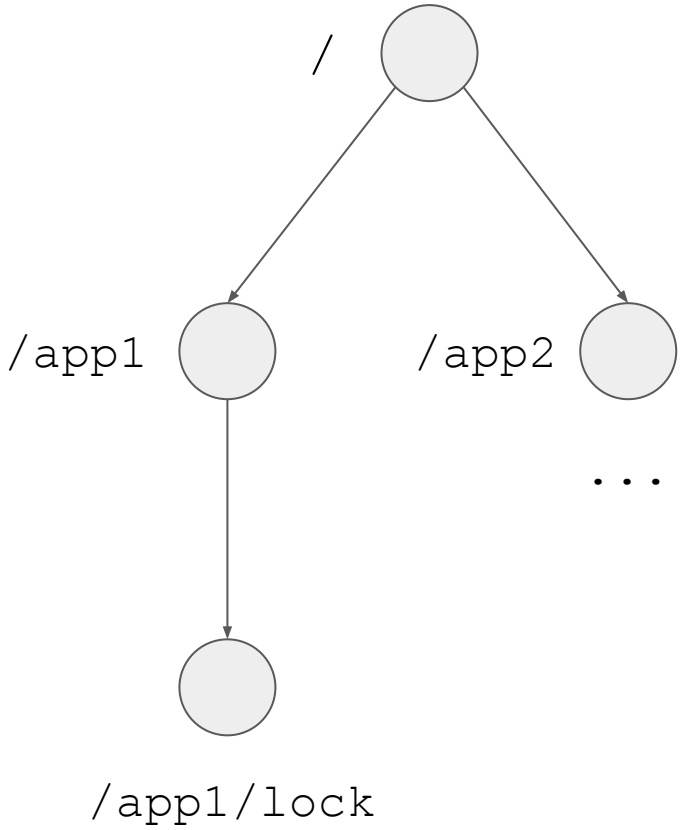
Locks



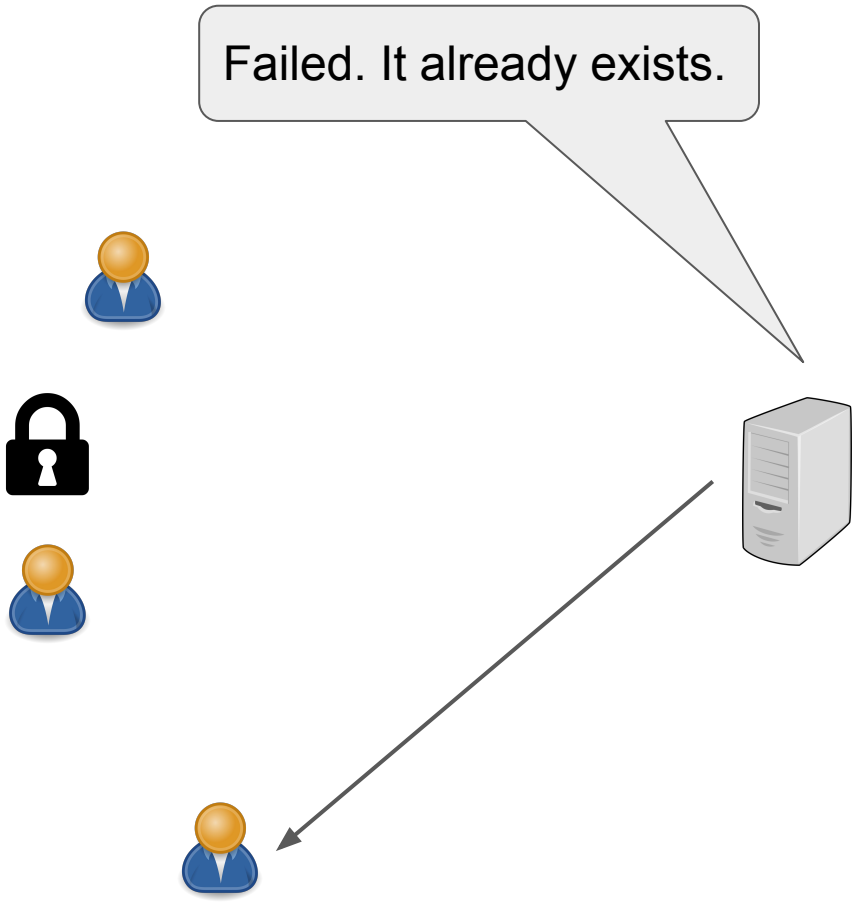


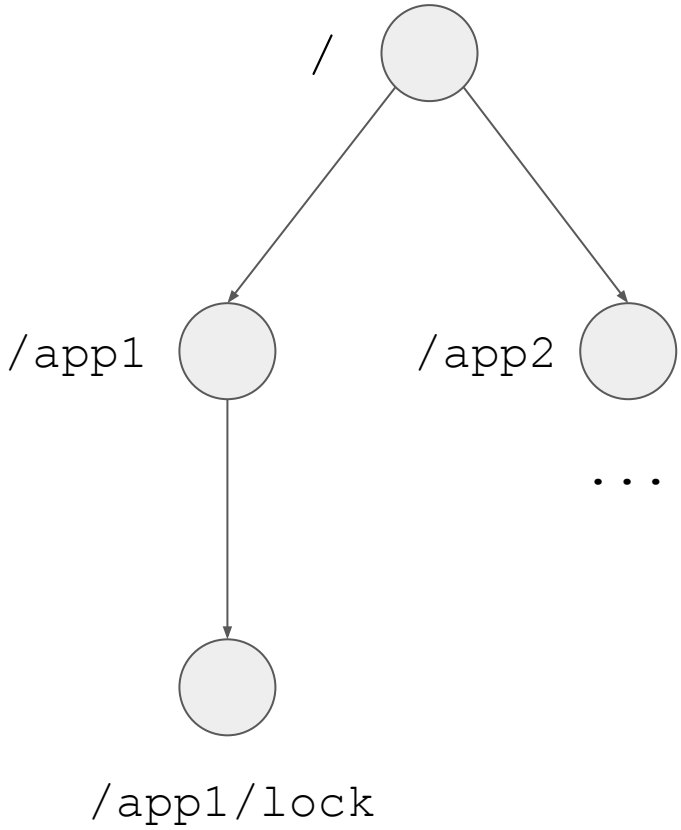
Locks



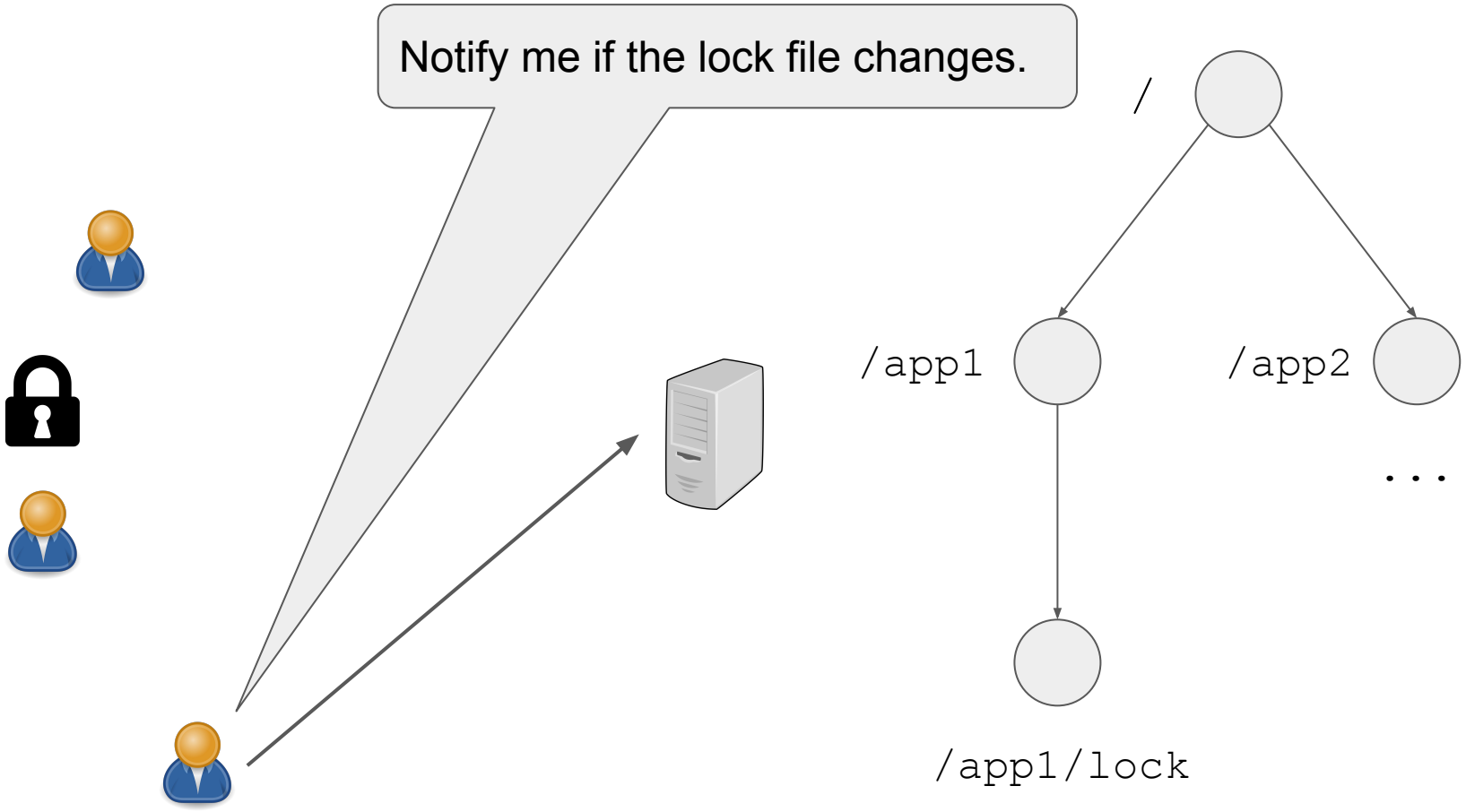


Locks

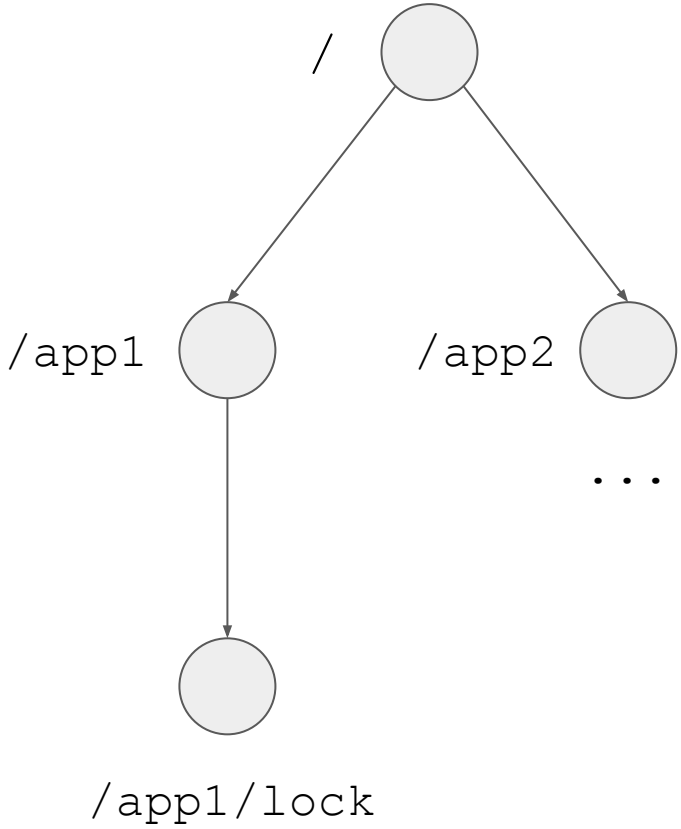




Locks



Locks

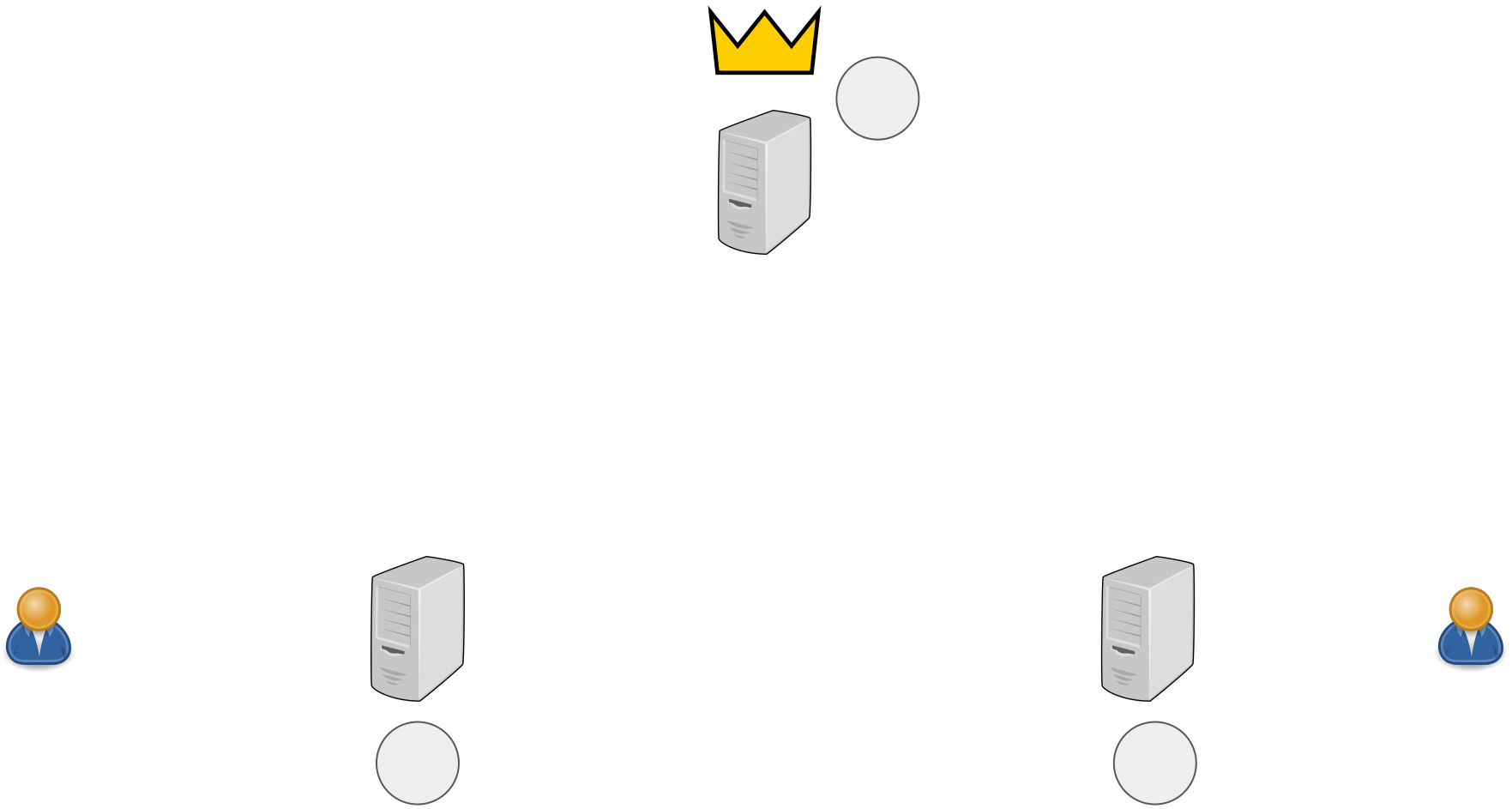


How does ZooKeeper achieve consistency?

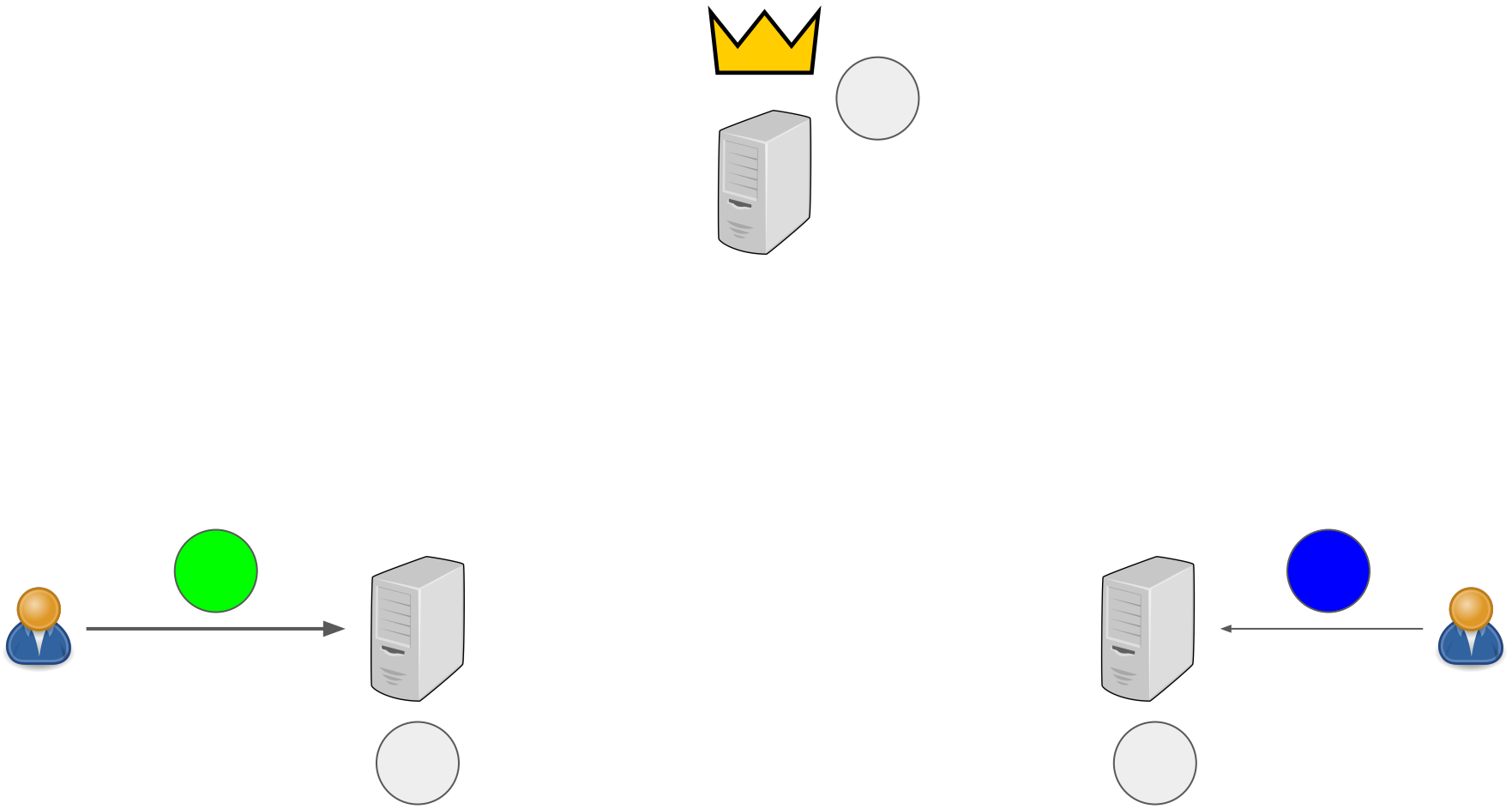
Write requests are funneled through a single leader process, which determines consistent total order via an **atomic broadcast** protocol.

Request processing is **pipelined**, naturally providing FIFO ordering for each client.

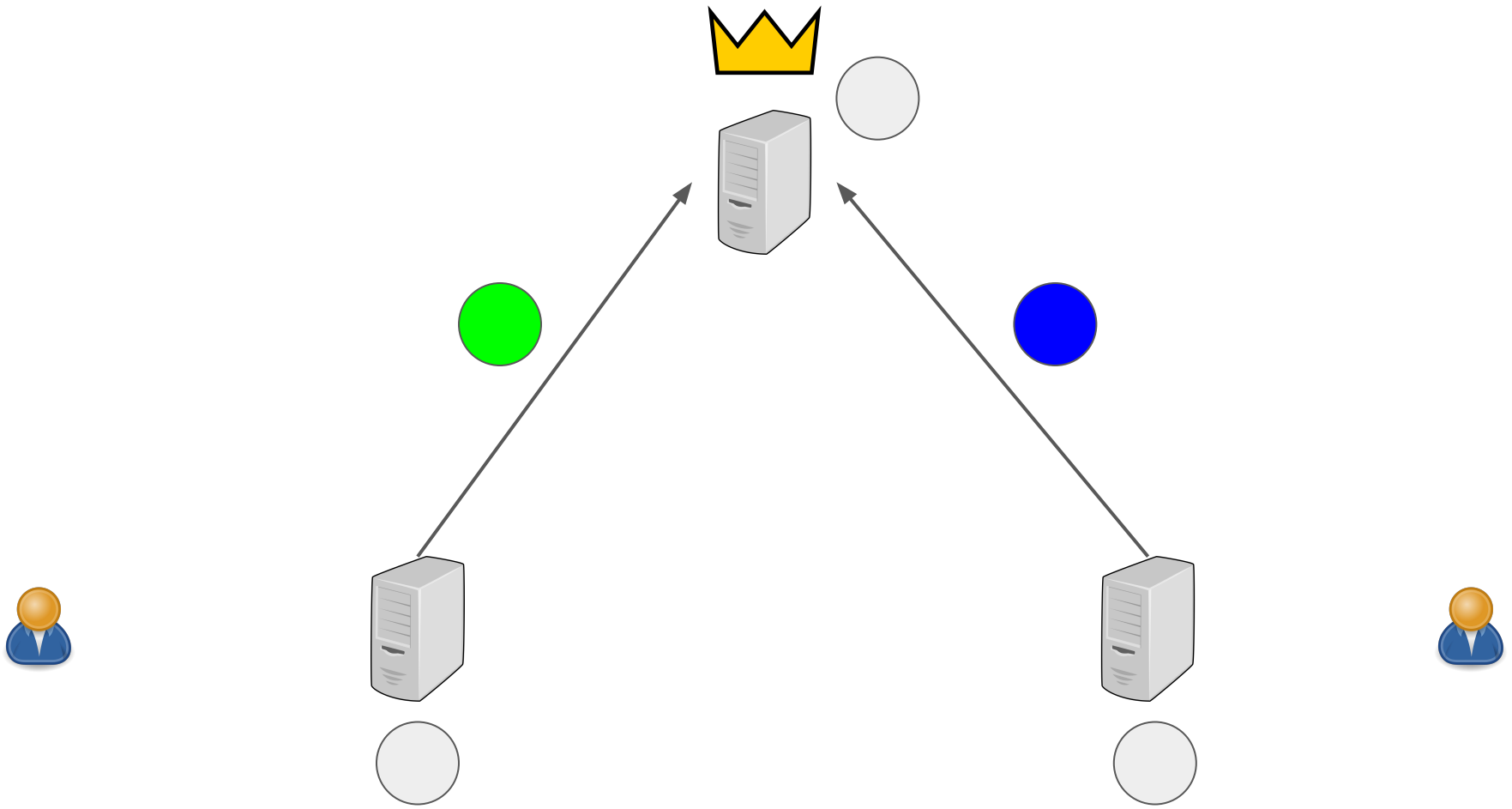
Linearizable Writes



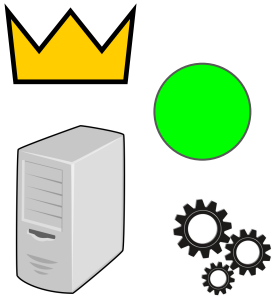
Linearizable Writes



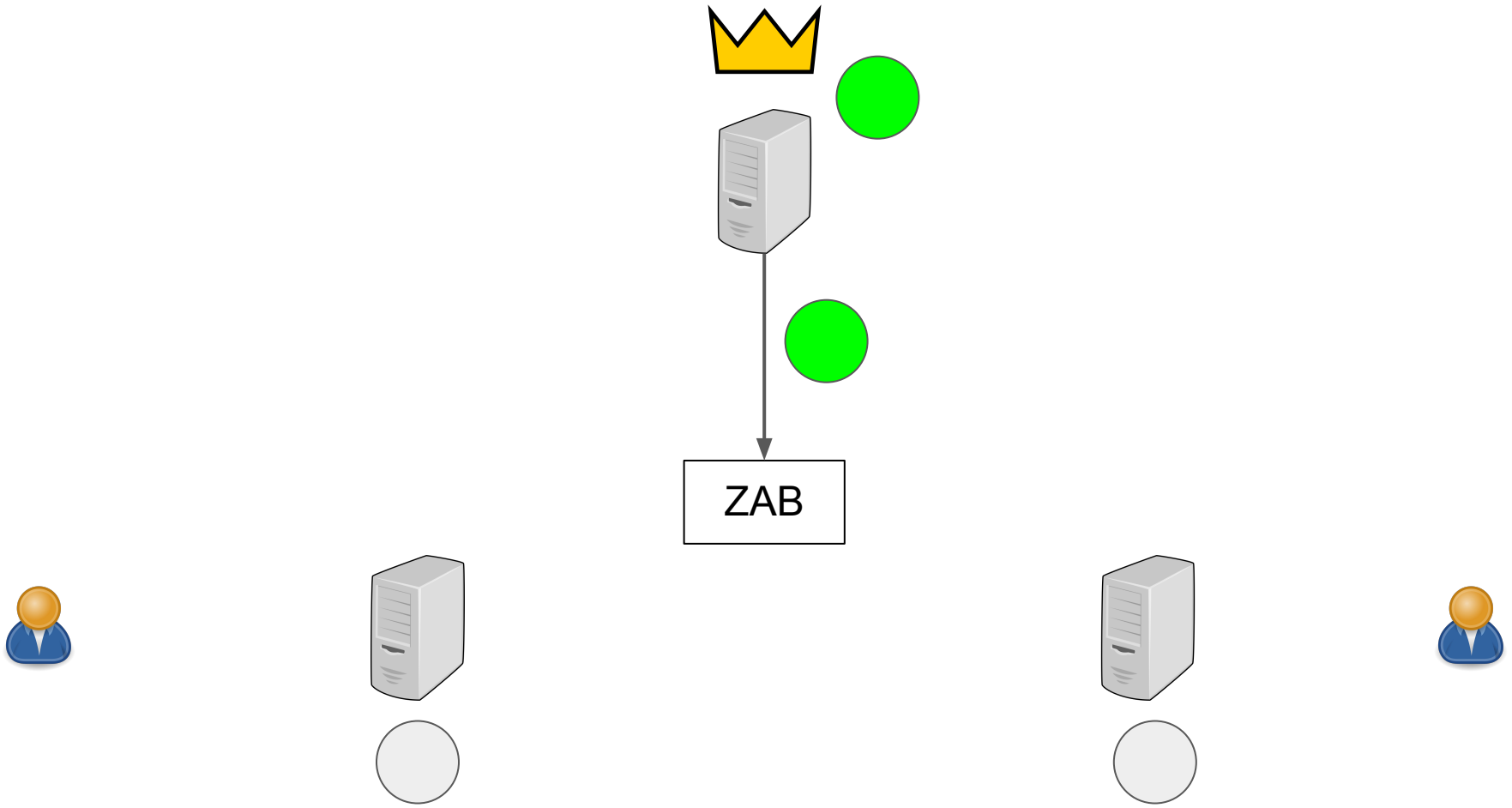
Linearizable Writes



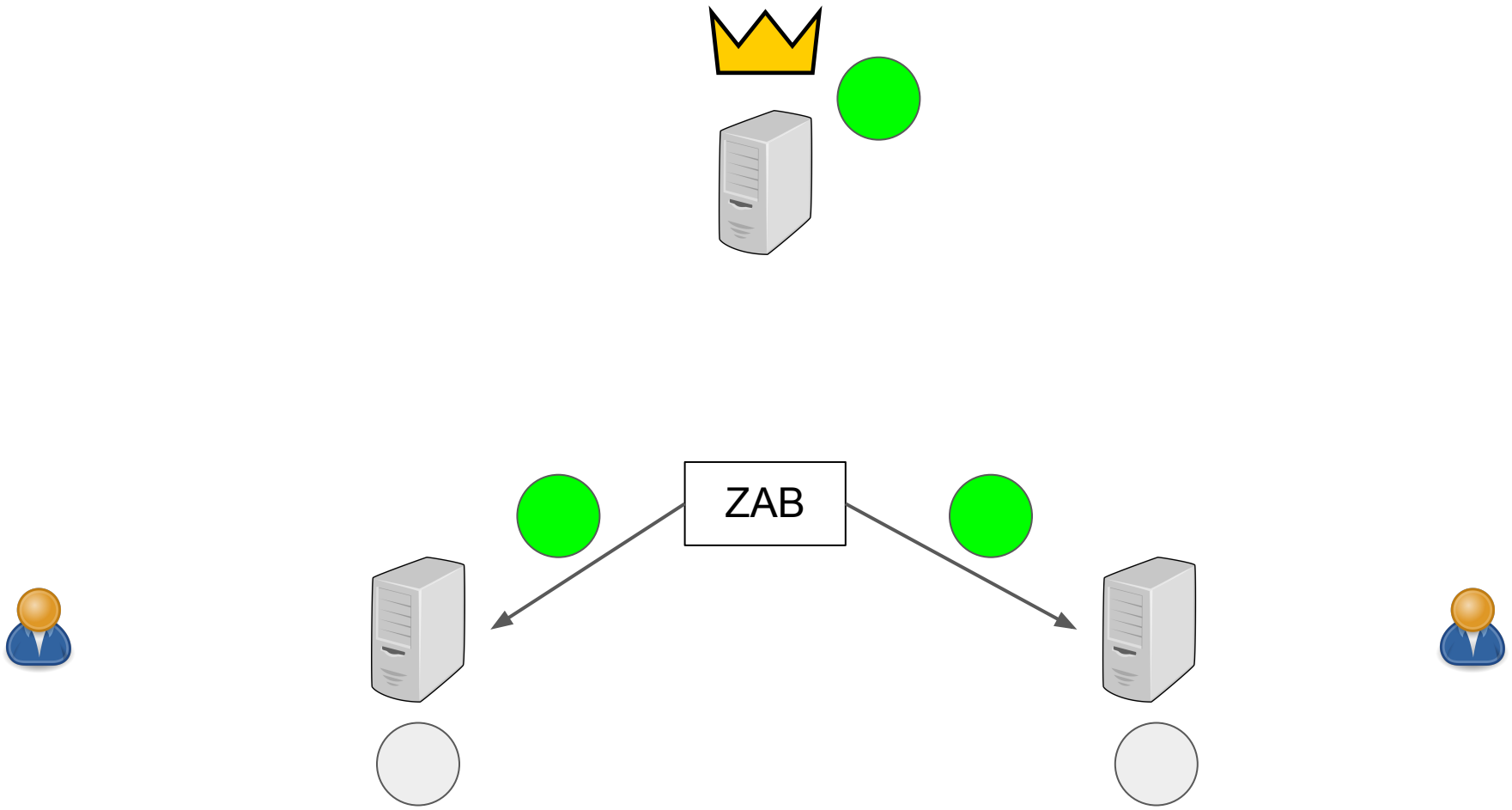
Linearizable Writes



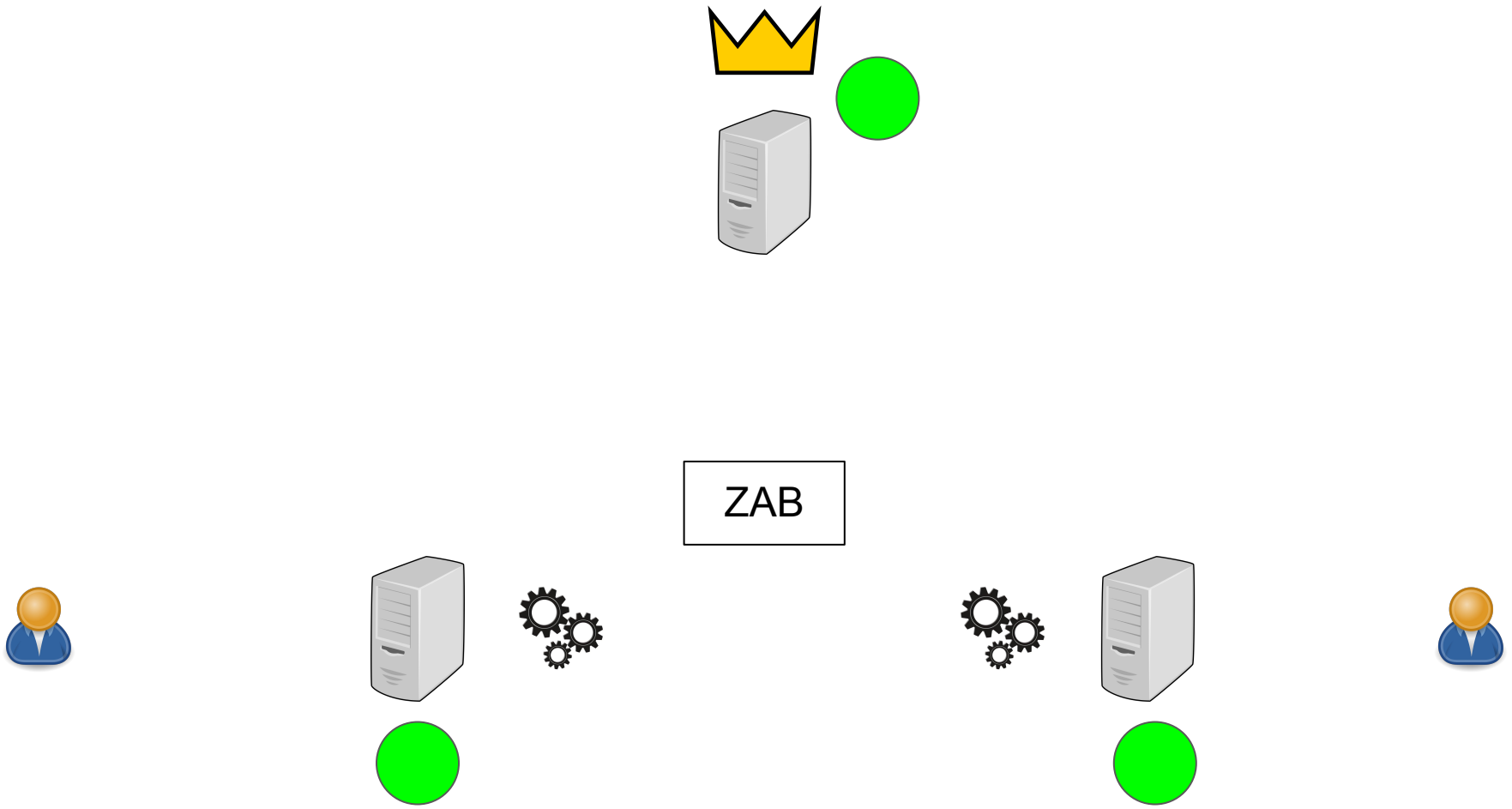
Linearizable Writes



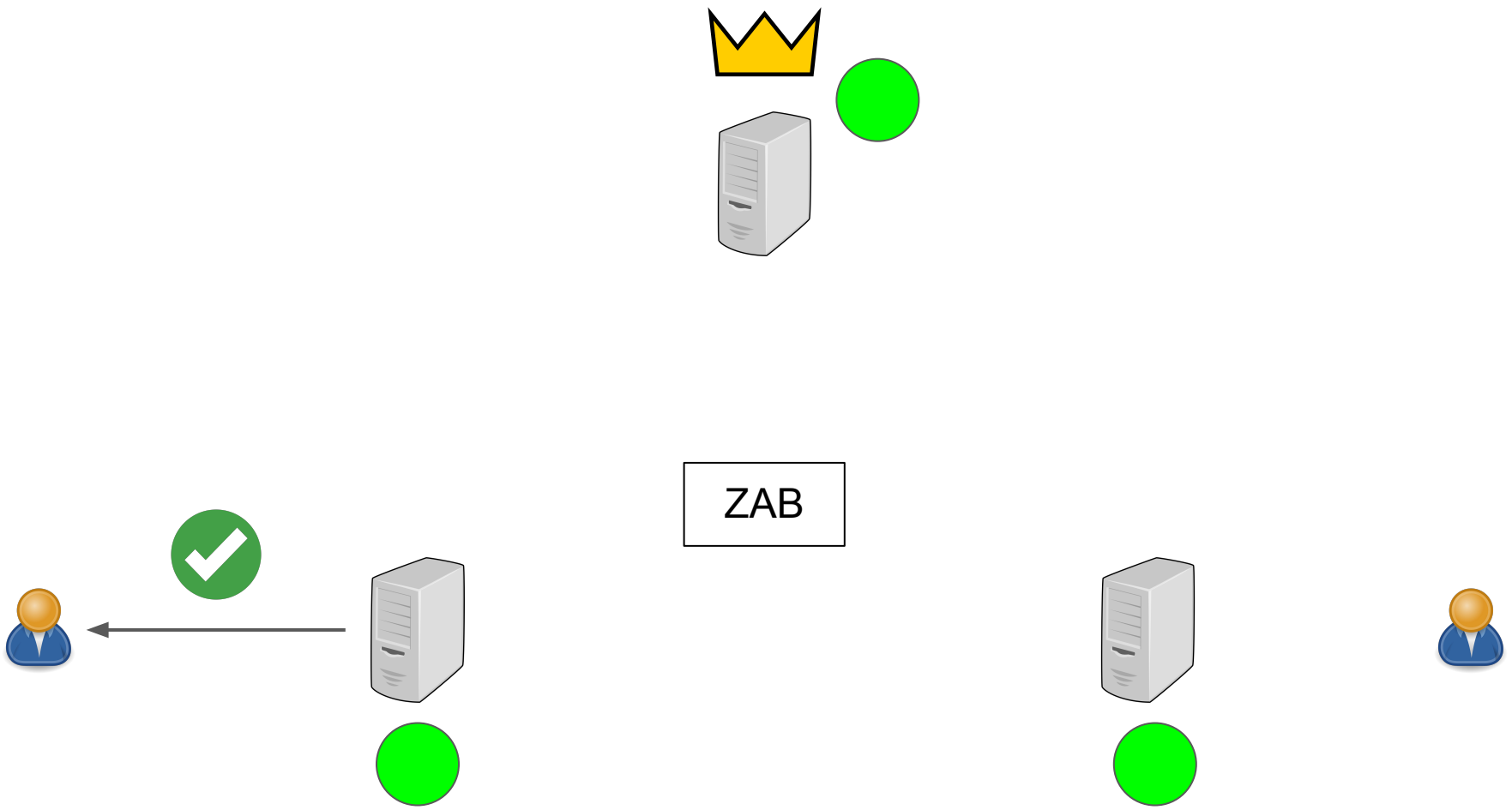
Linearizable Writes



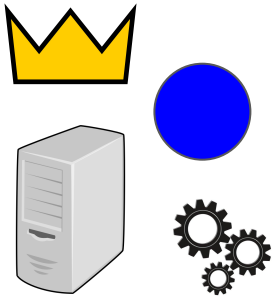
Linearizable Writes



Linearizable Writes

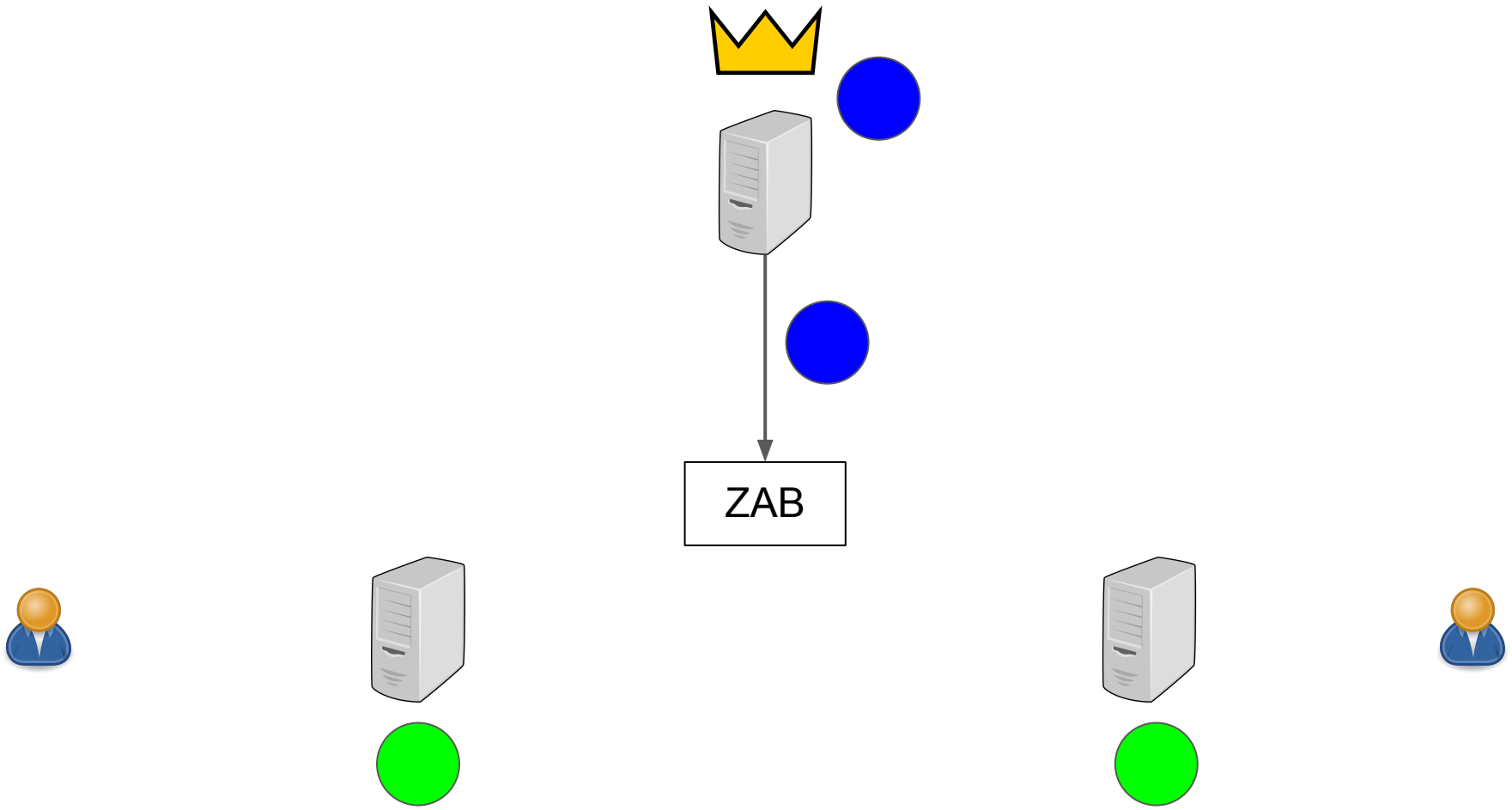


Linearizable Writes

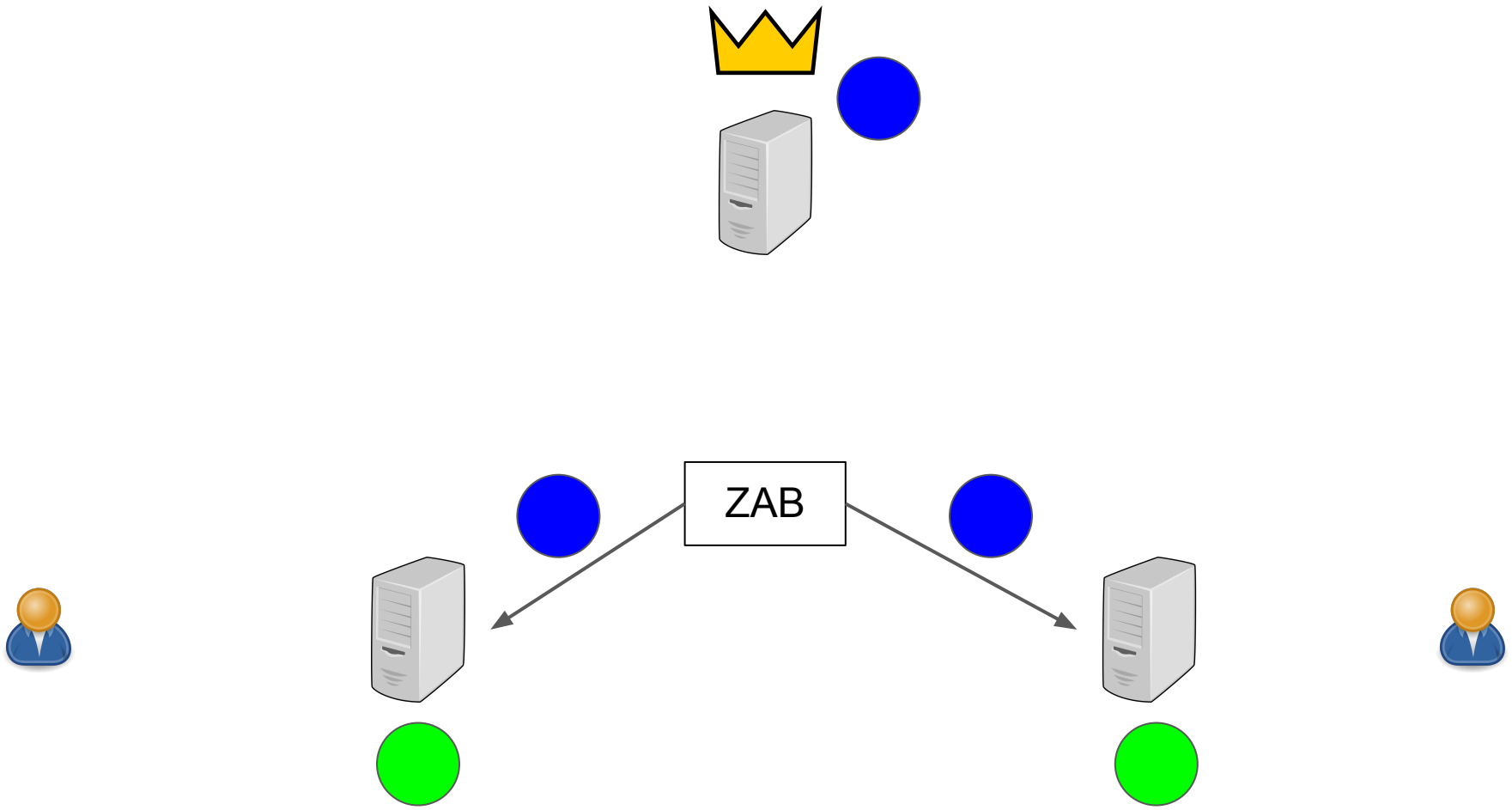


ZAB

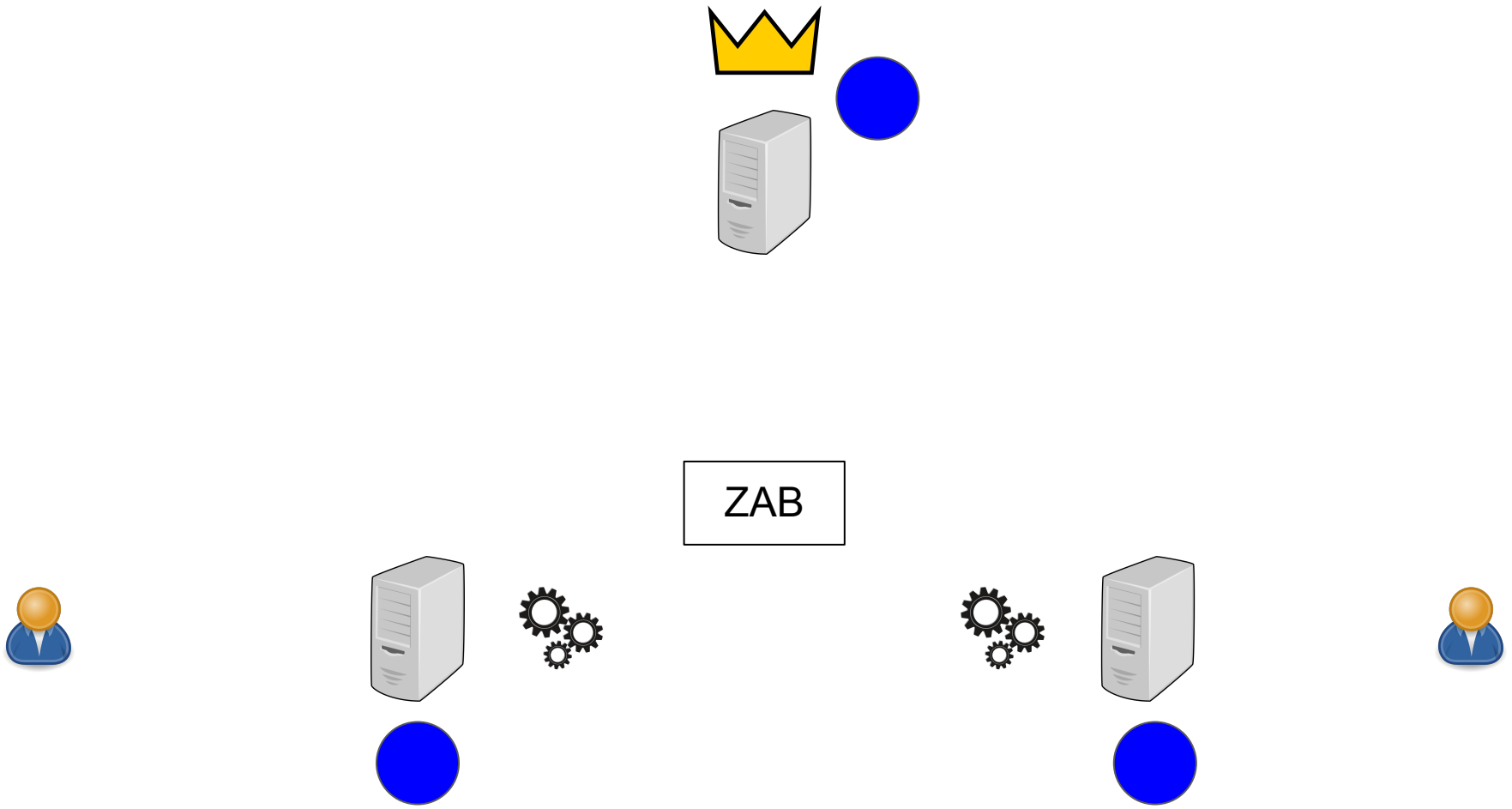
Linearizable Writes



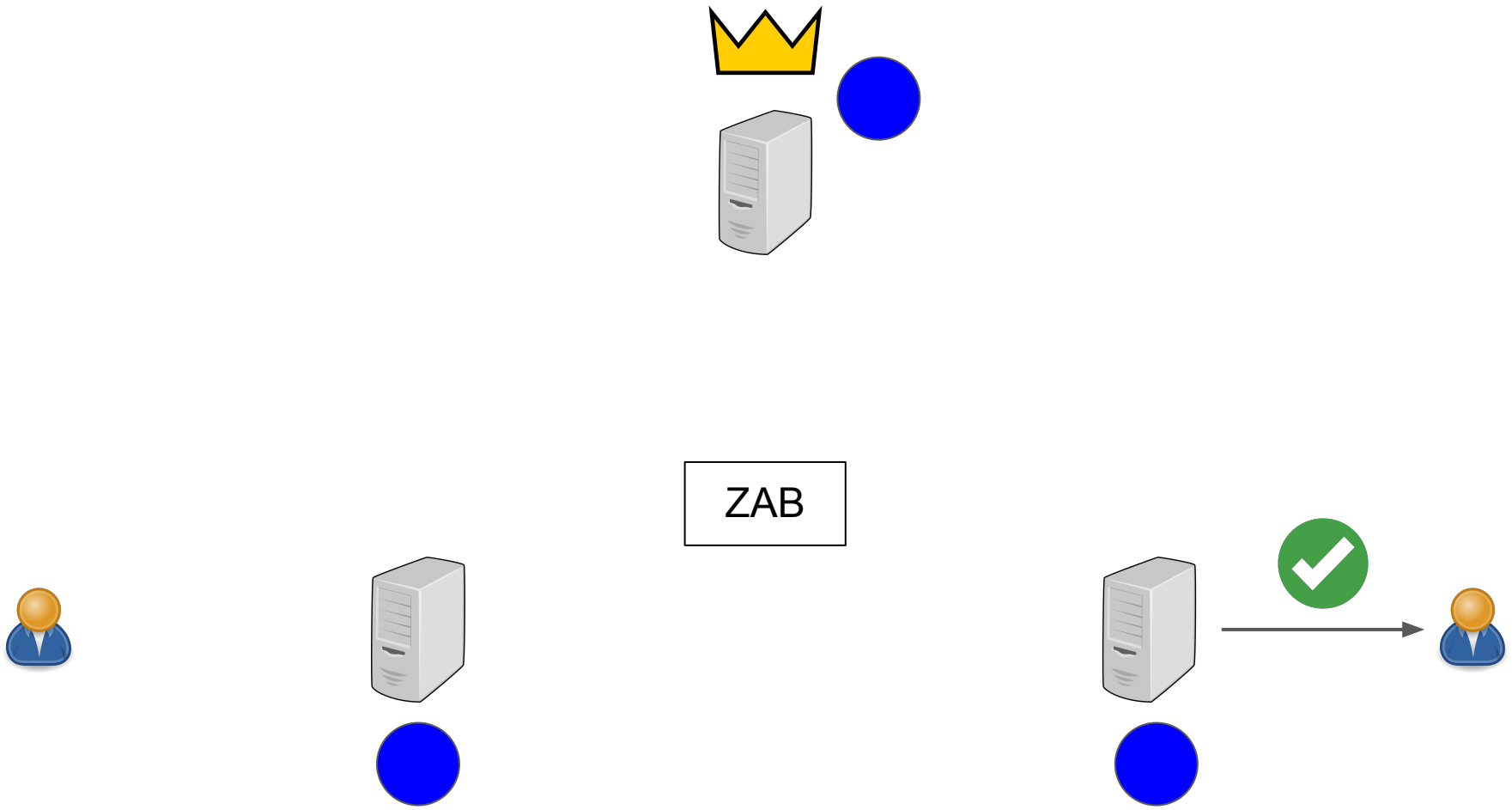
Linearizable Writes



Linearizable Writes

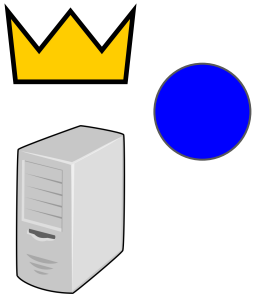


Linearizable Writes

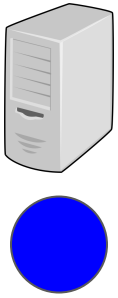


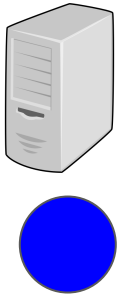
Linearizable Writes

Atomic Broadcast guarantees that writes are applied in the same total order at all replicas



ZAB



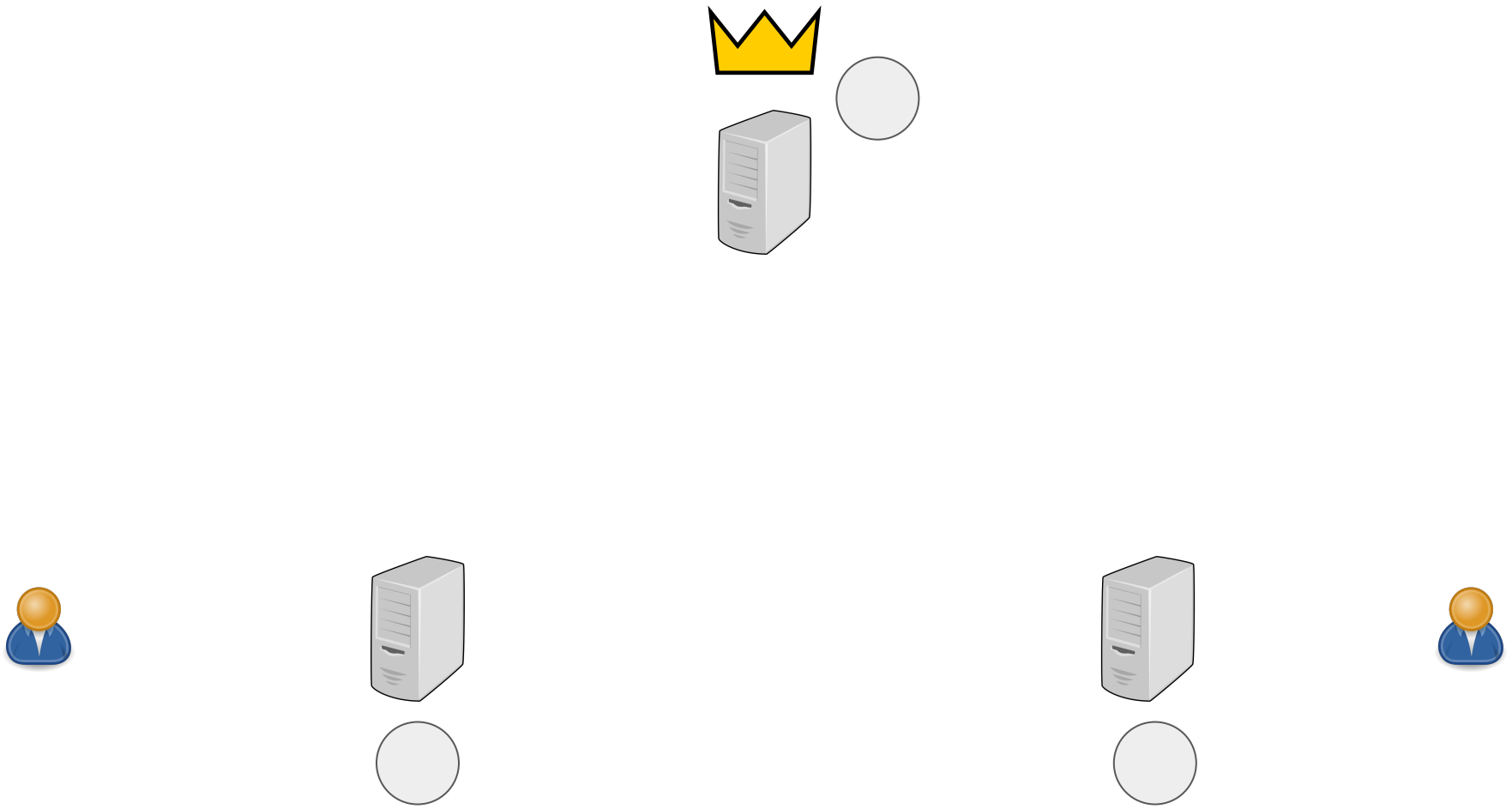


How does ZooKeeper achieve performance?

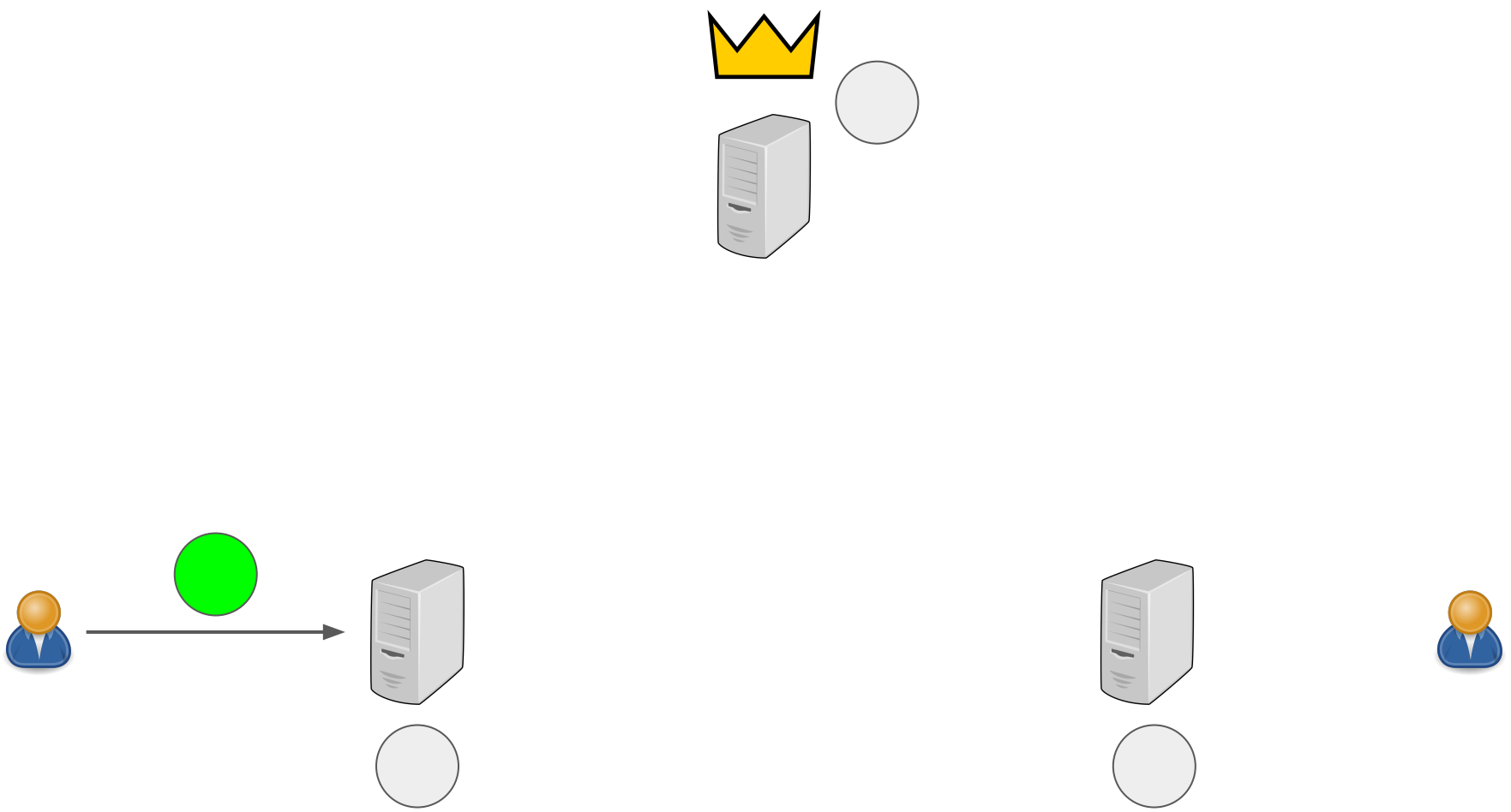
All ZooKeeper operations are **wait-free**.

Read requests are processed **locally**, by the individual replica that communicates with the client.

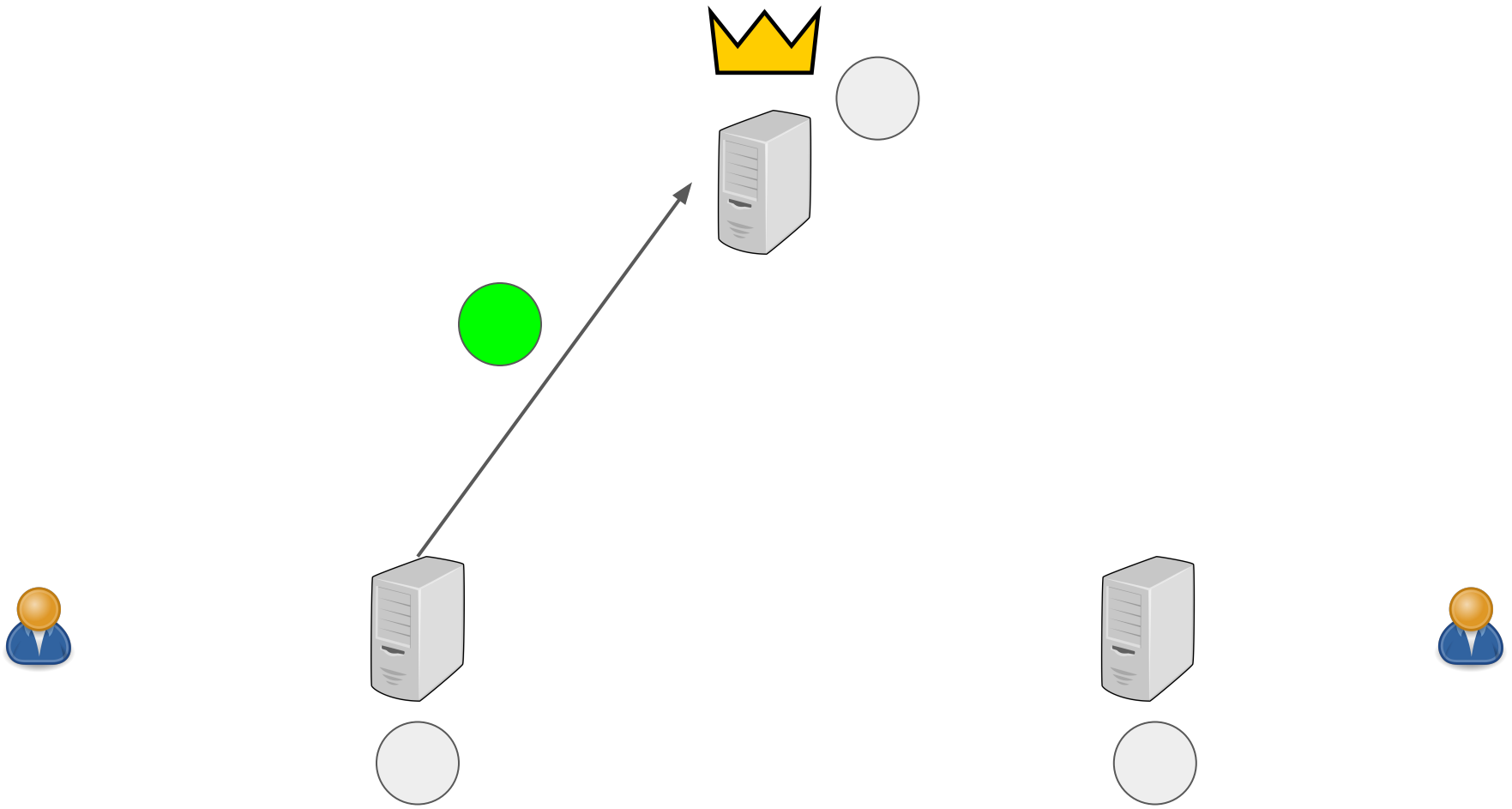
Fast Reads



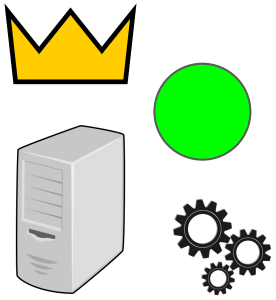
Fast Reads



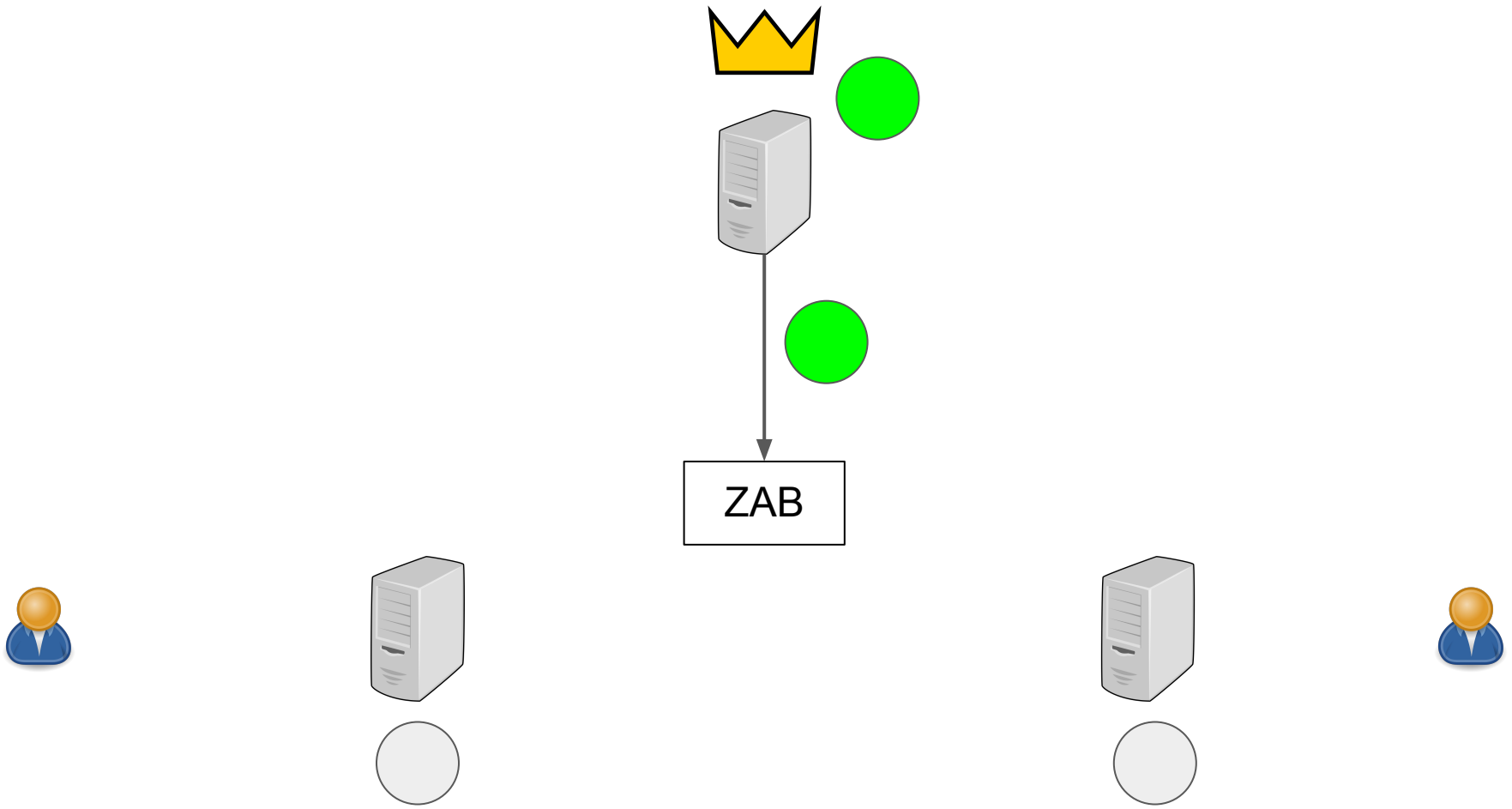
Fast Reads



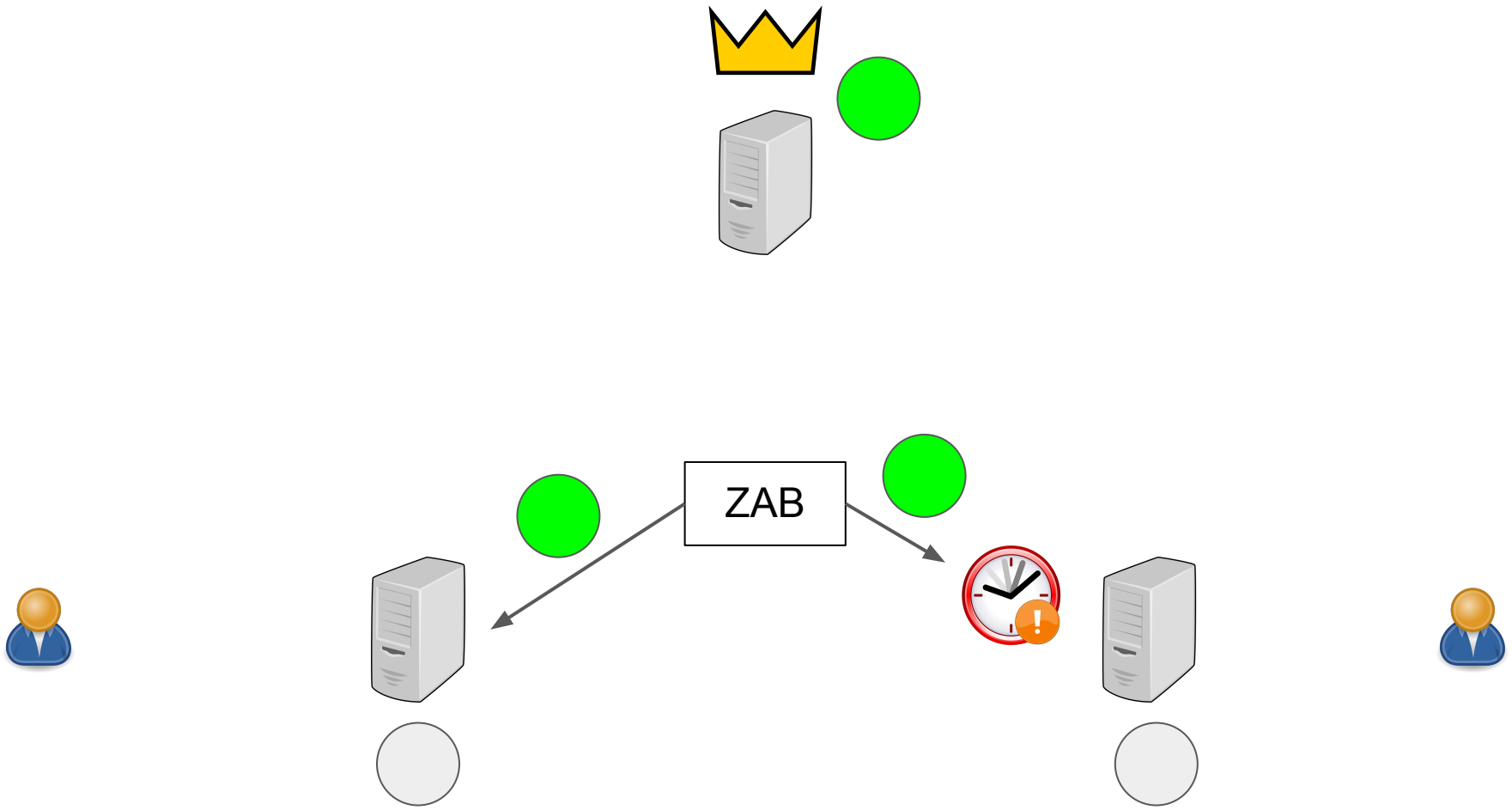
Fast Reads



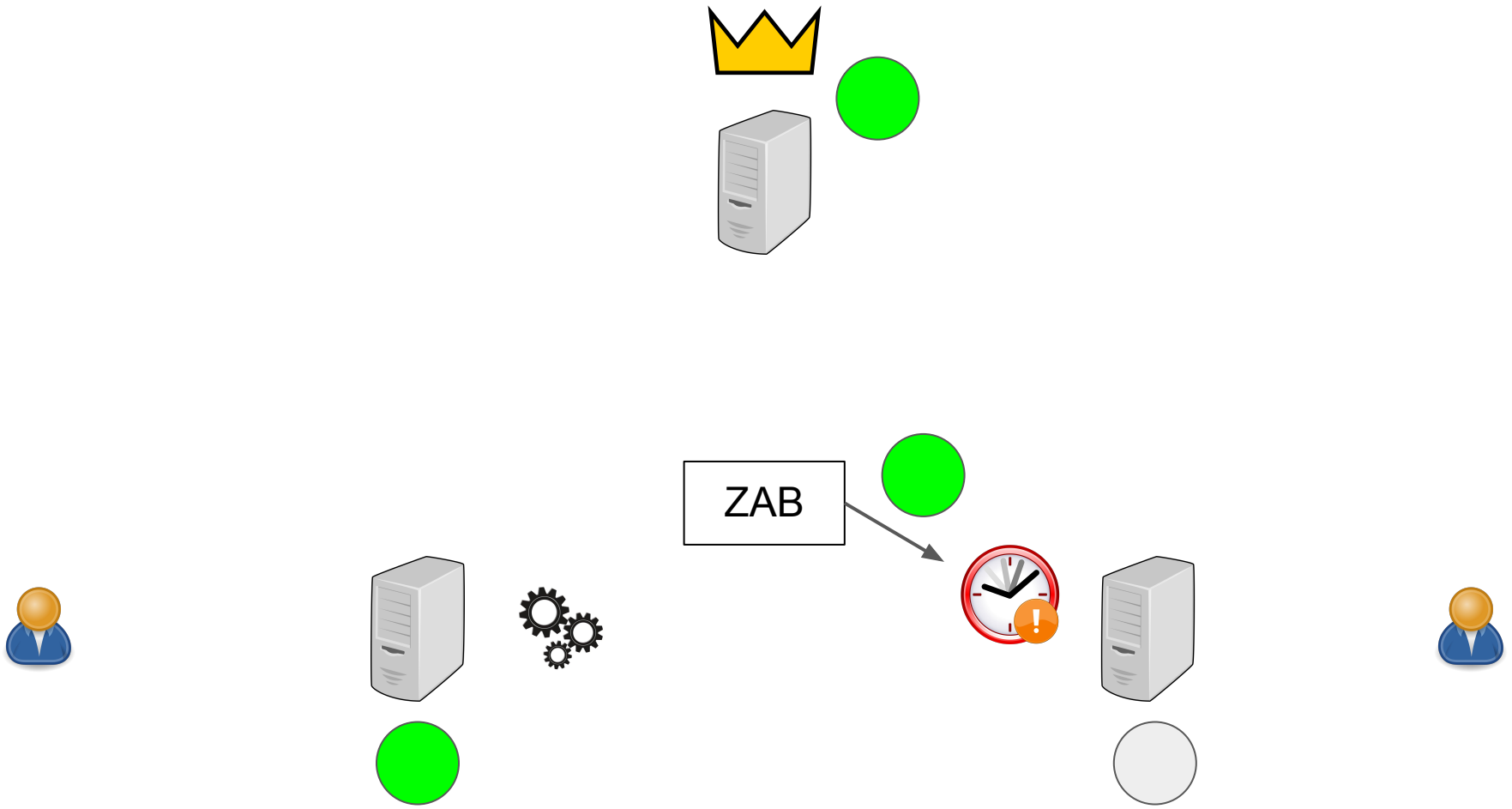
Fast Reads



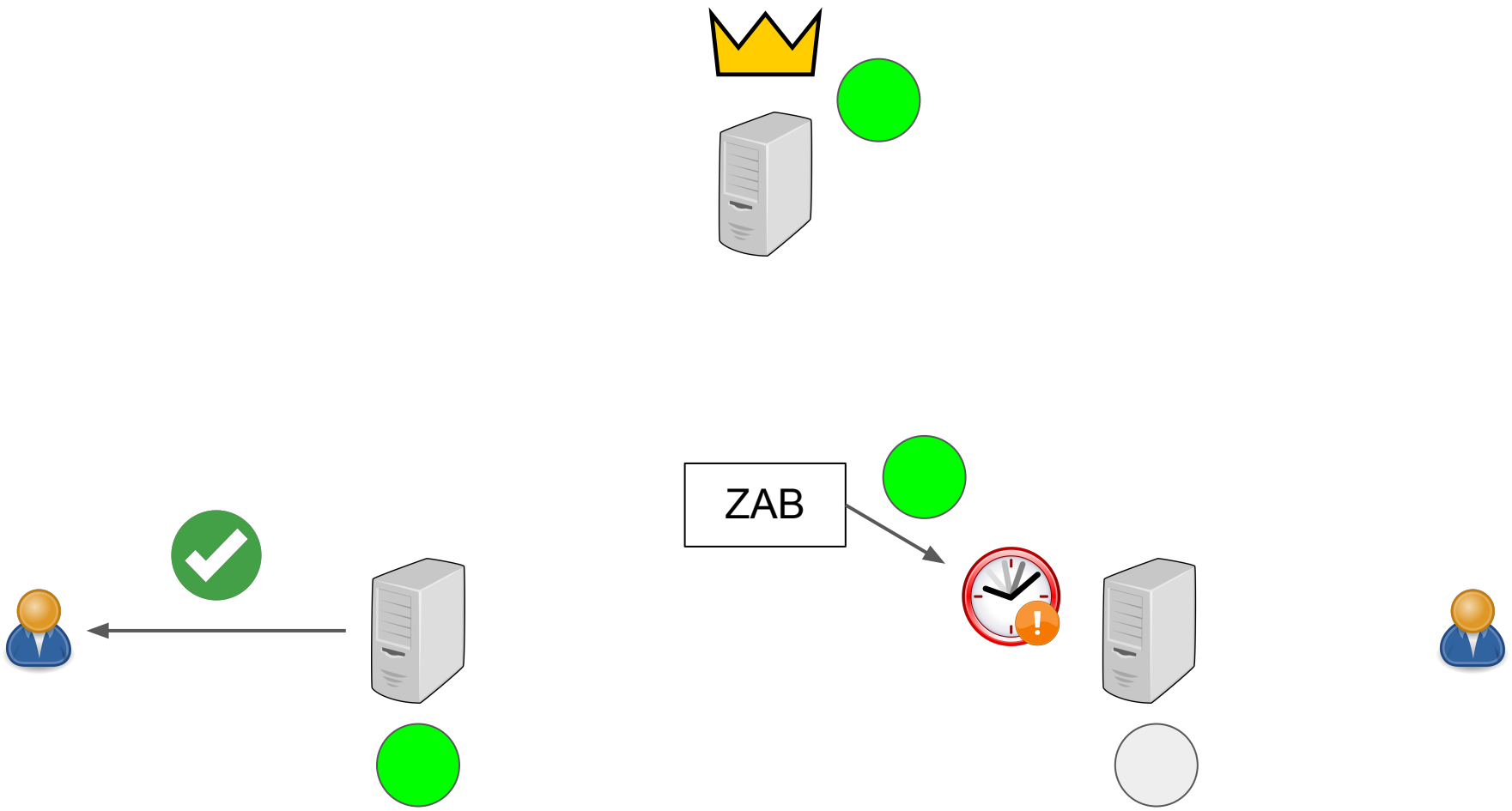
Fast Reads



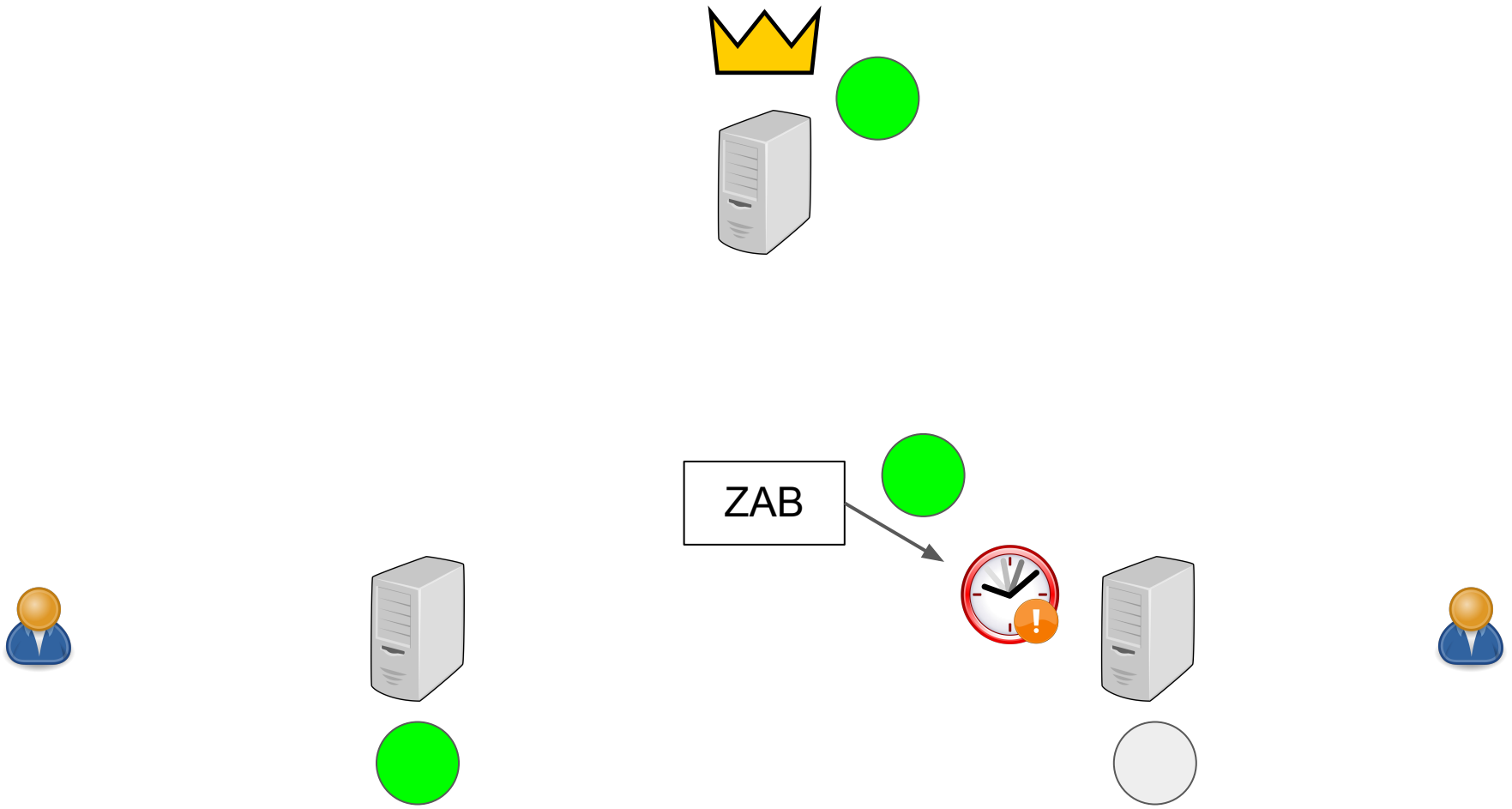
Fast Reads



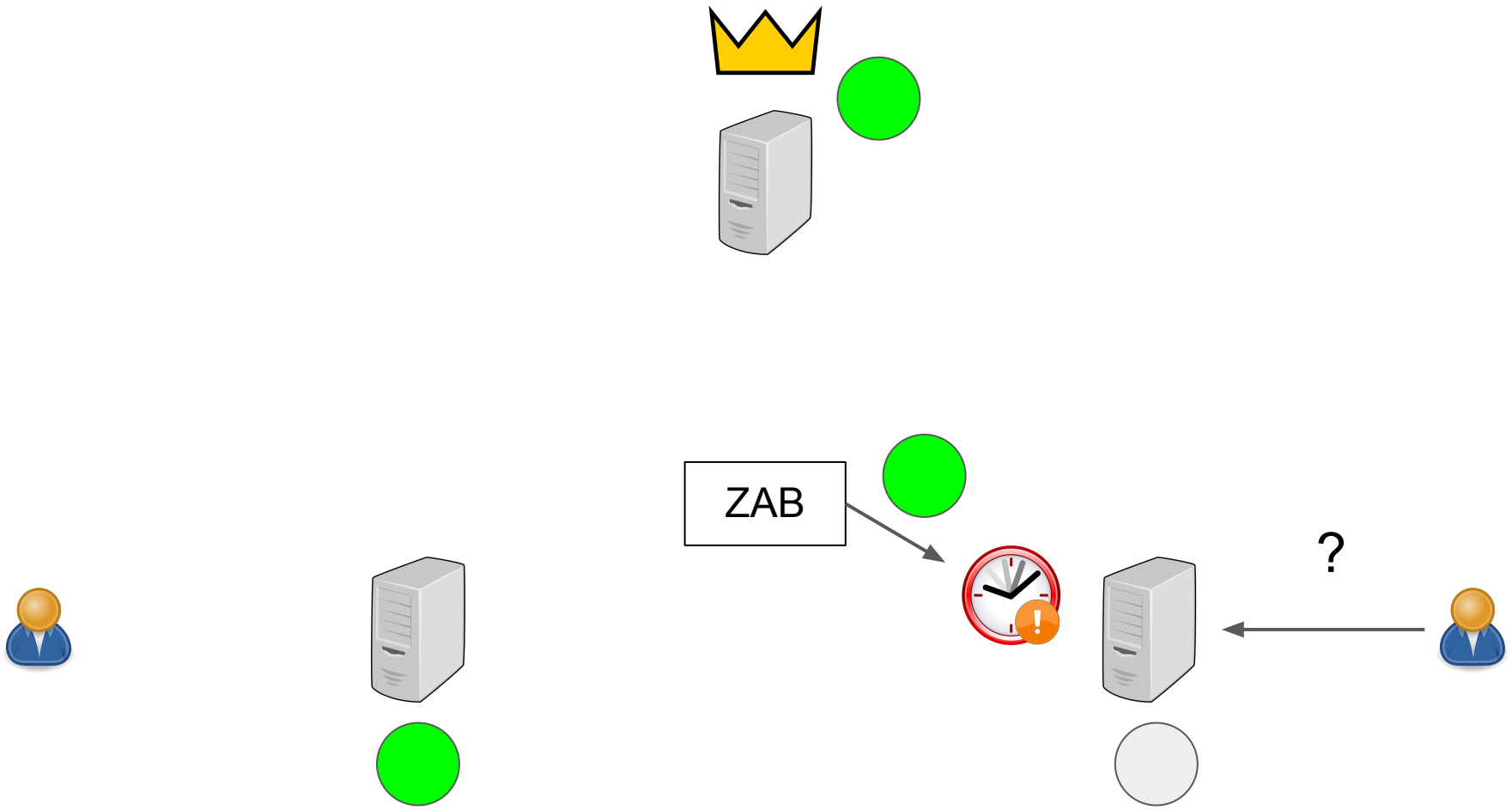
Fast Reads



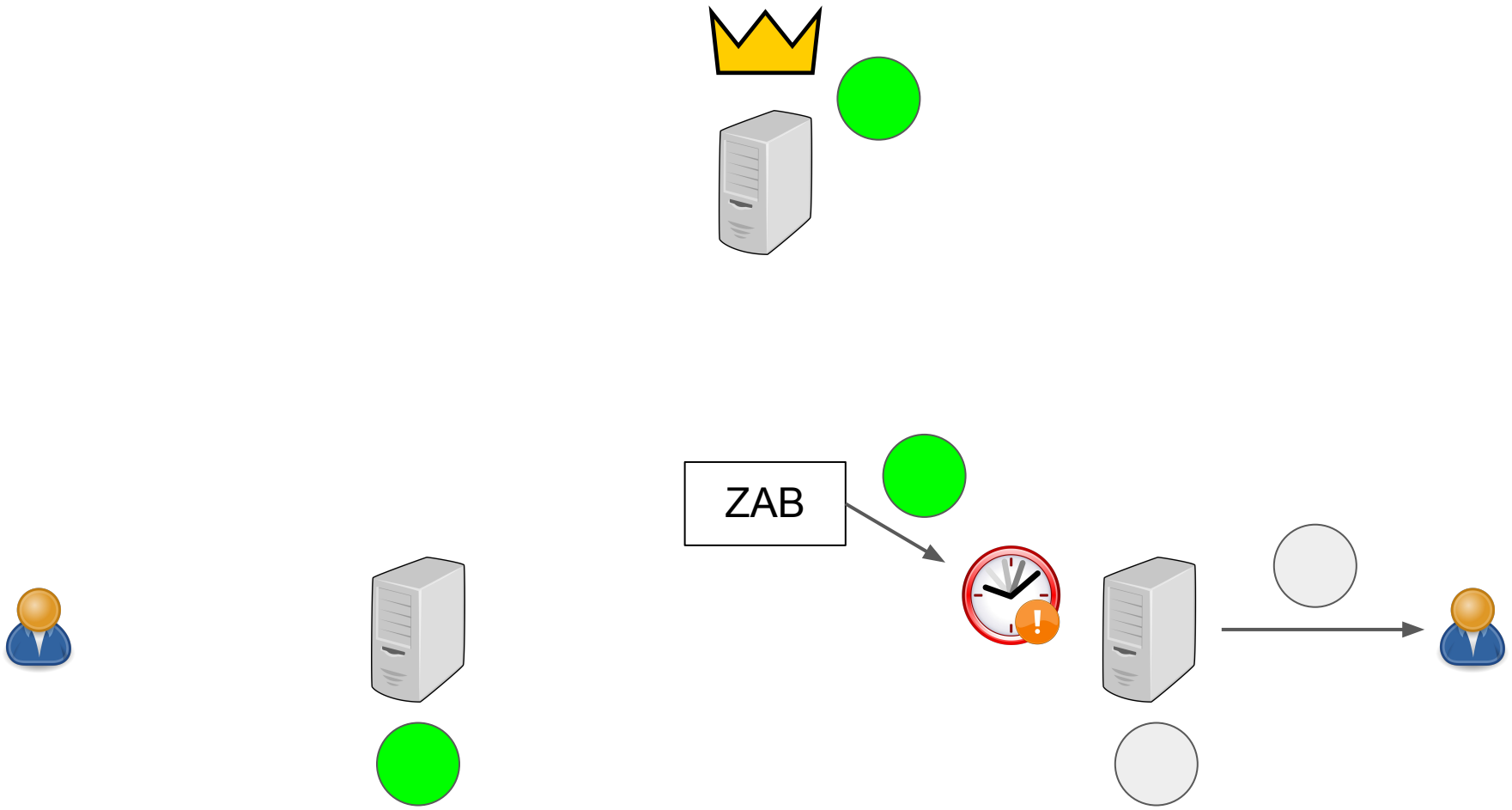
Fast Reads



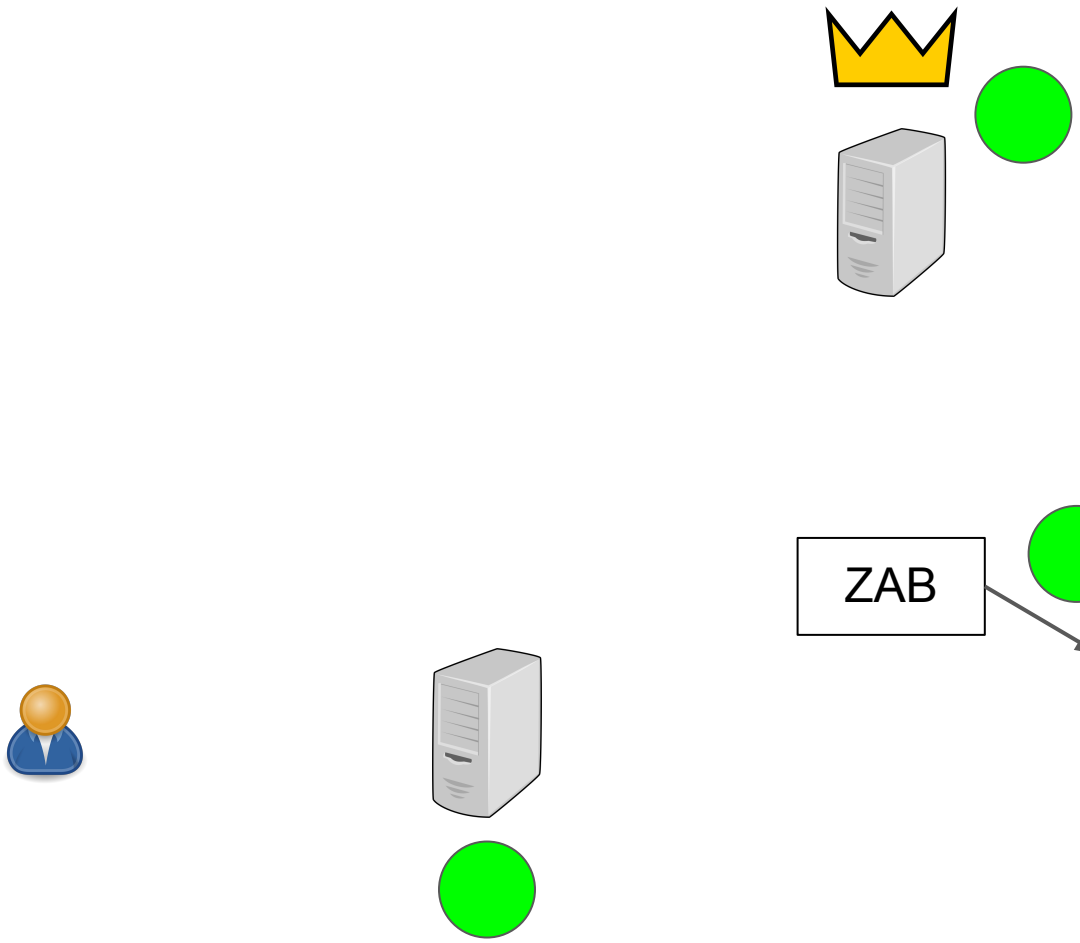
Fast Reads



Fast Reads



Fast Reads



A read may return a stale value, even though a write request has completed!

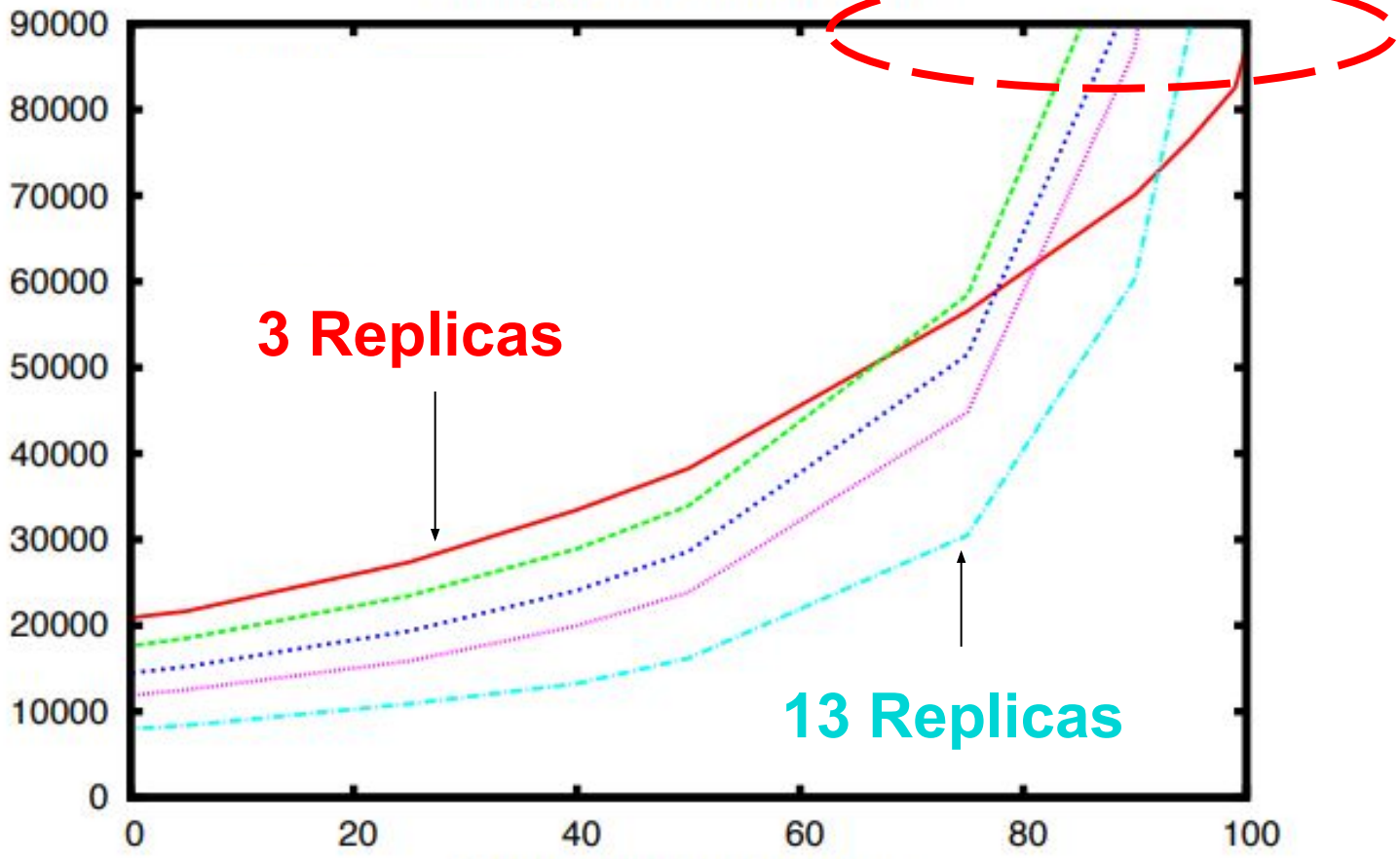
How does it Perform?

To evaluate ZooKeeper's performance, the throughput was measured for a fully-saturated system.

- 250 clients
- 1K Data for read/write requests
- Read:Write ratio varied from 0 to 1
- Experiment repeated for different numbers of ZooKeeper replicas

Throughput

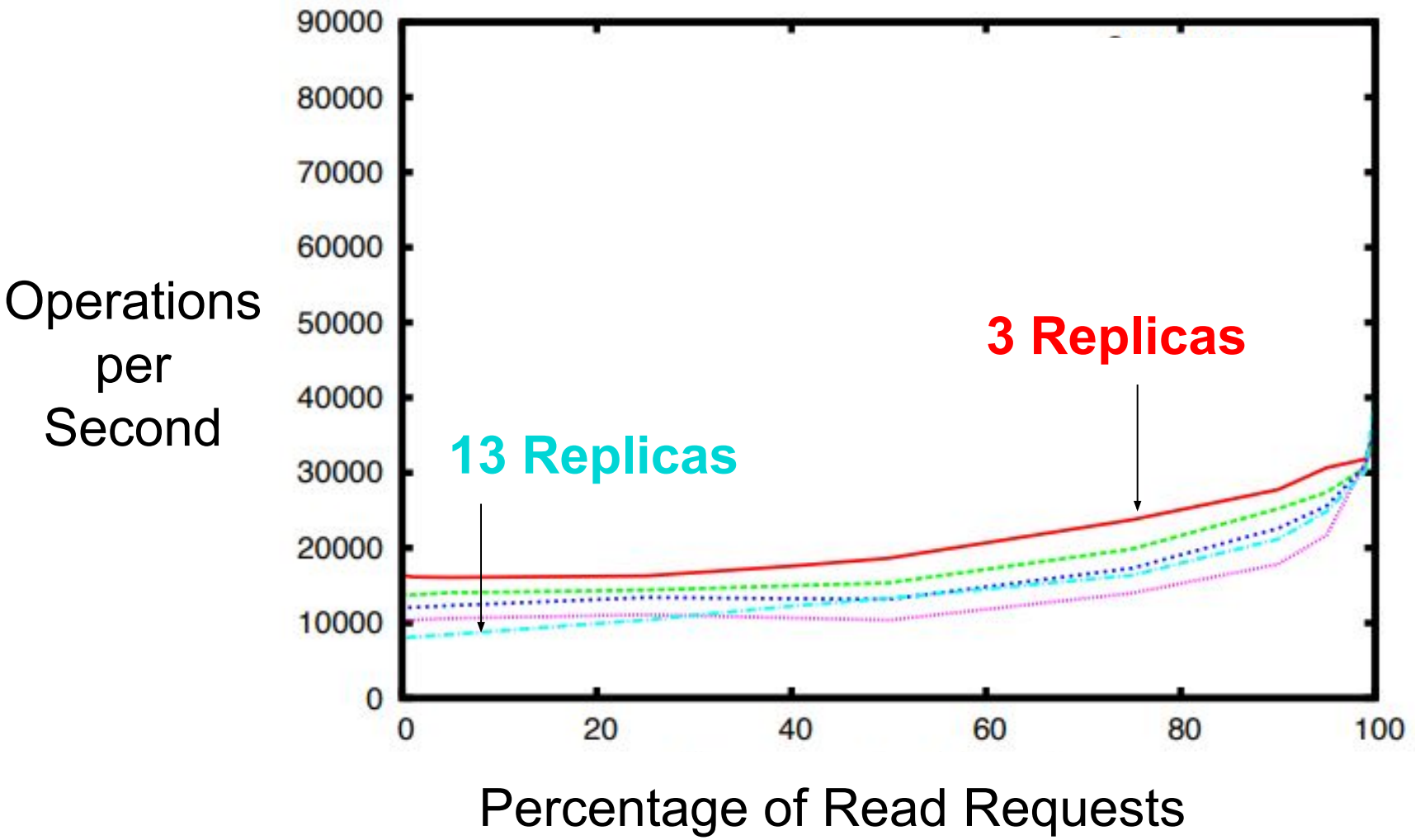
Throughput for 13 replicas, 100% reads: 460k



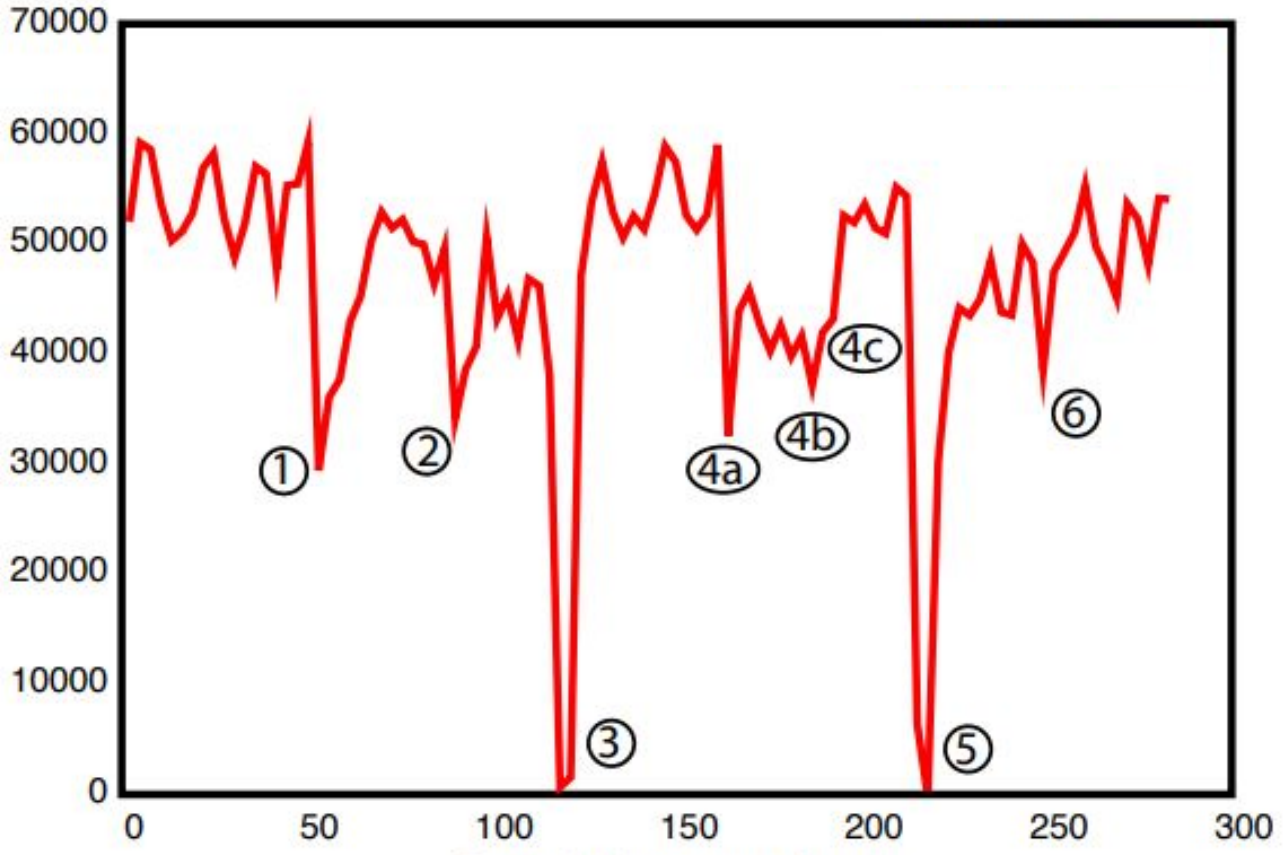
Operations per Second

Percentage of Read Requests

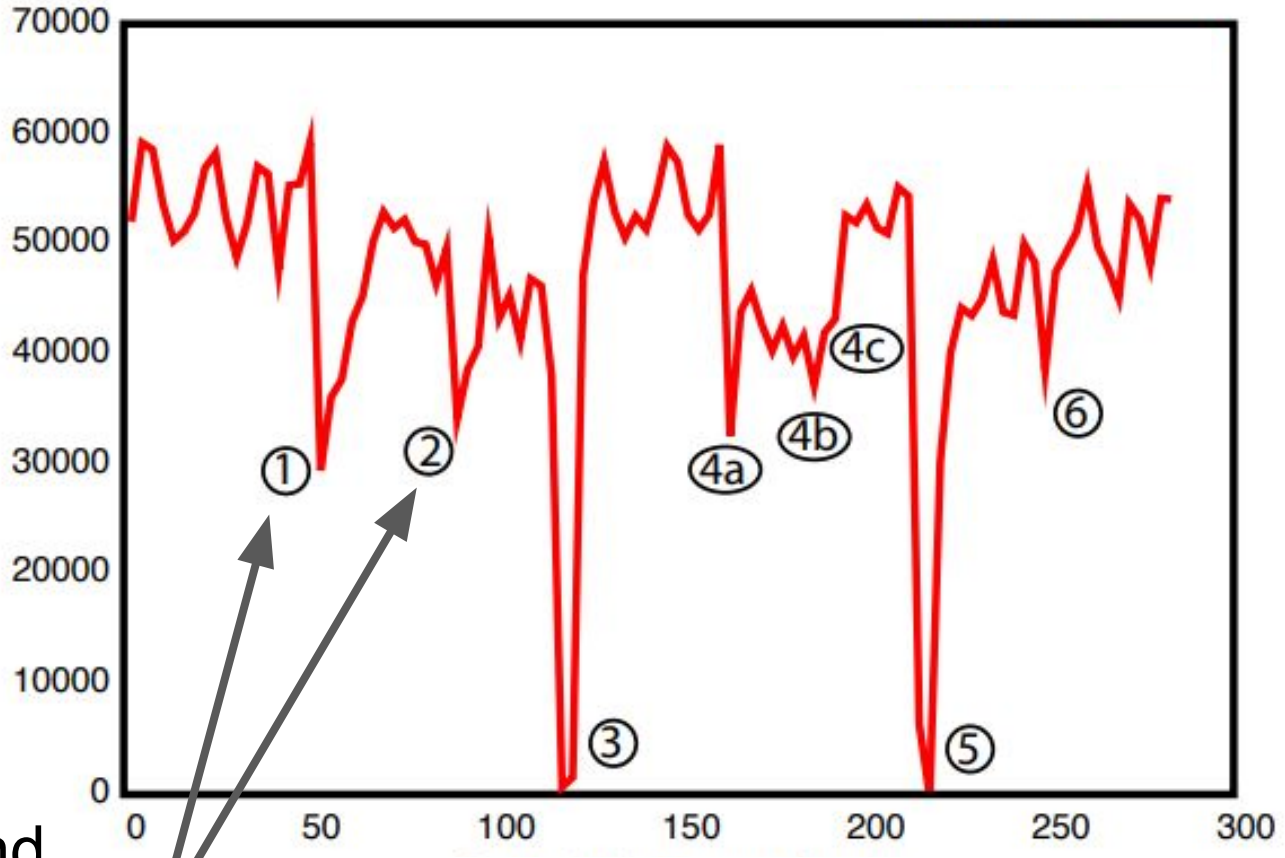
Throughput (All Requests to Leader)



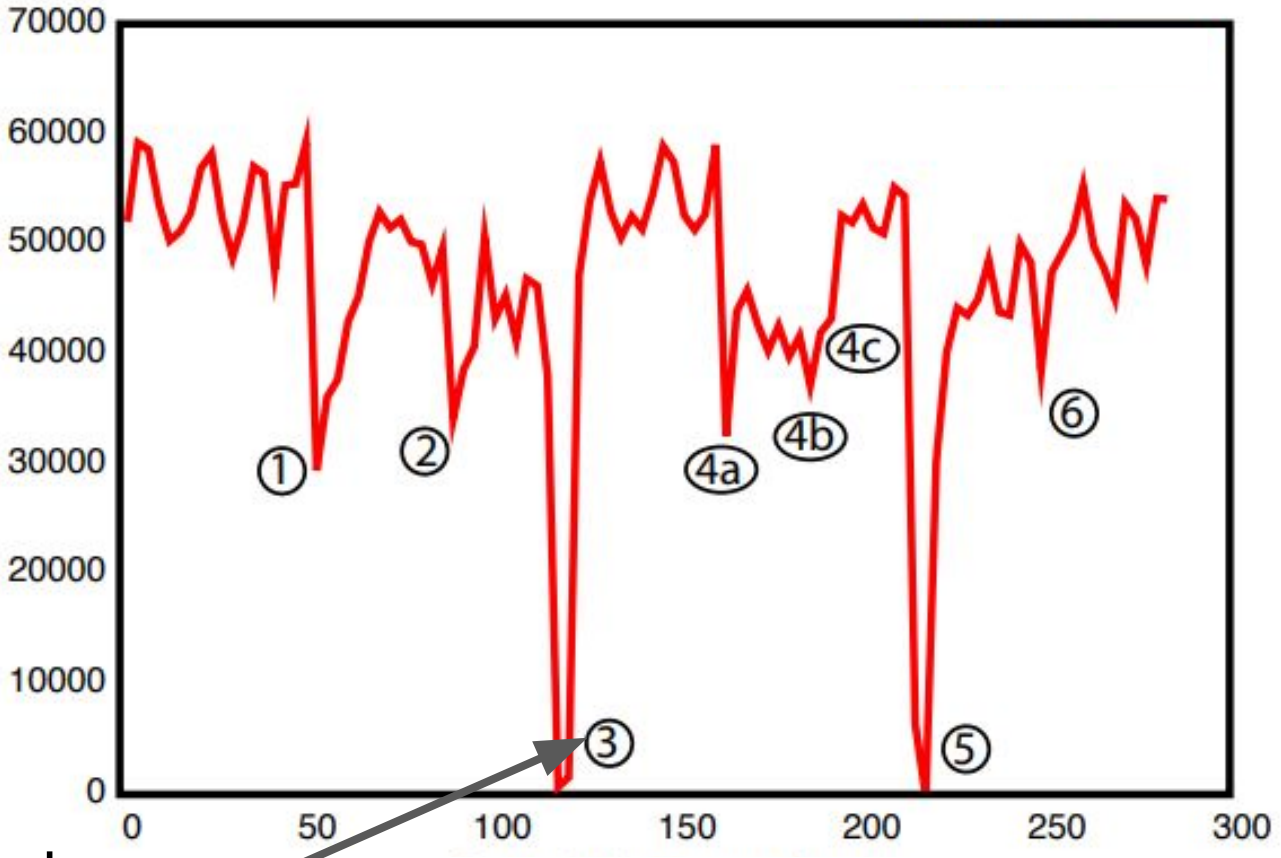
Throughput with Failures



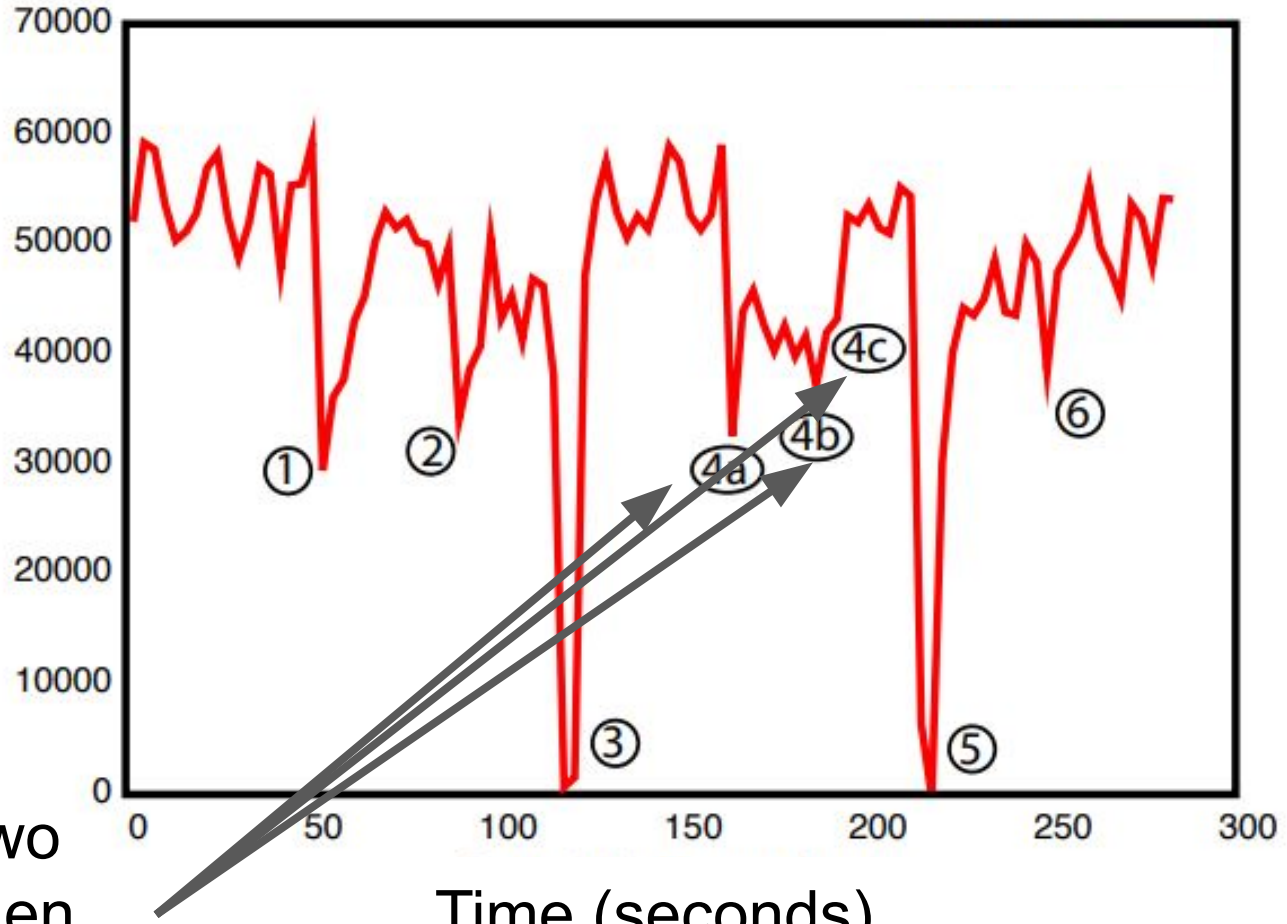
Throughput with Failures



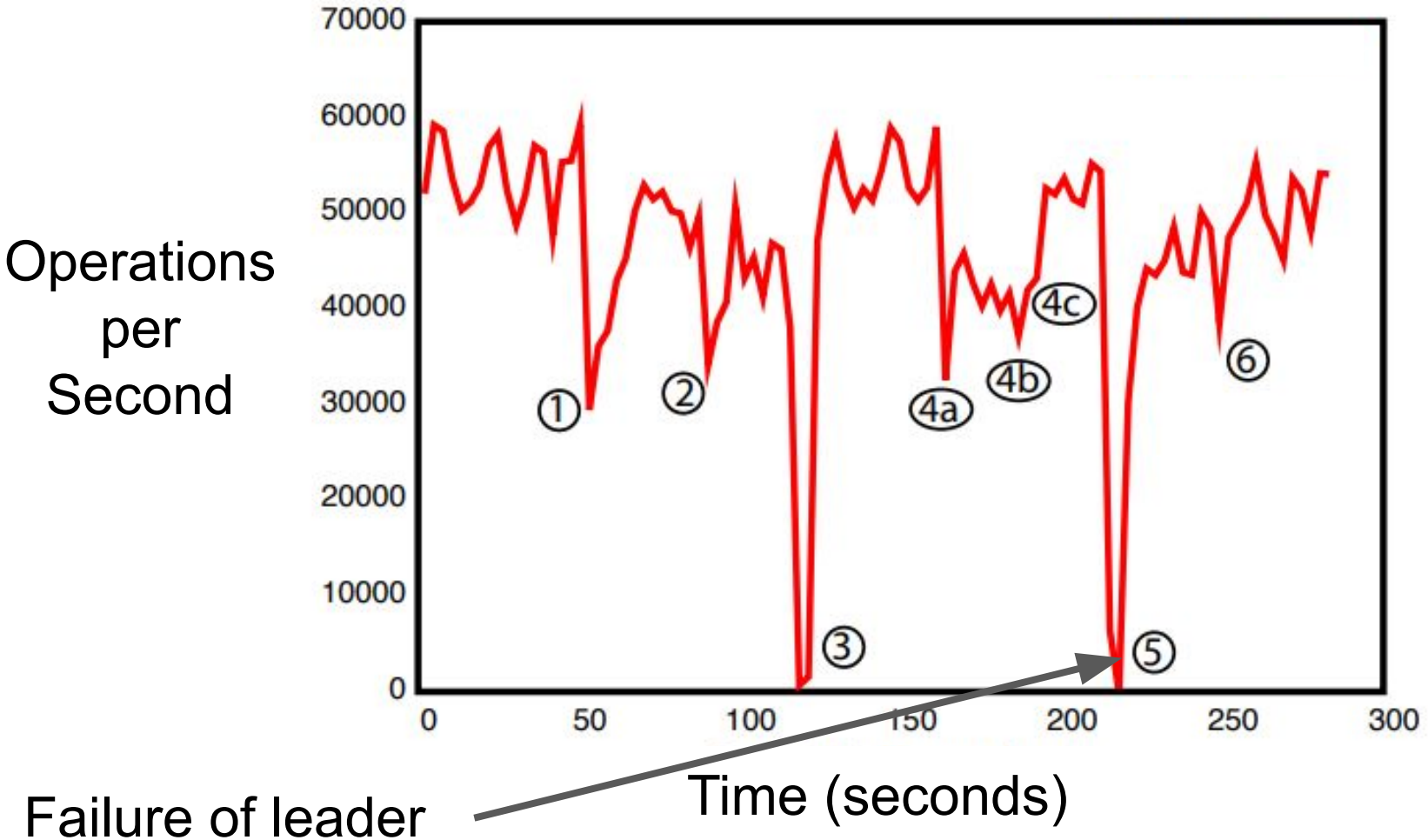
Throughput with Failures



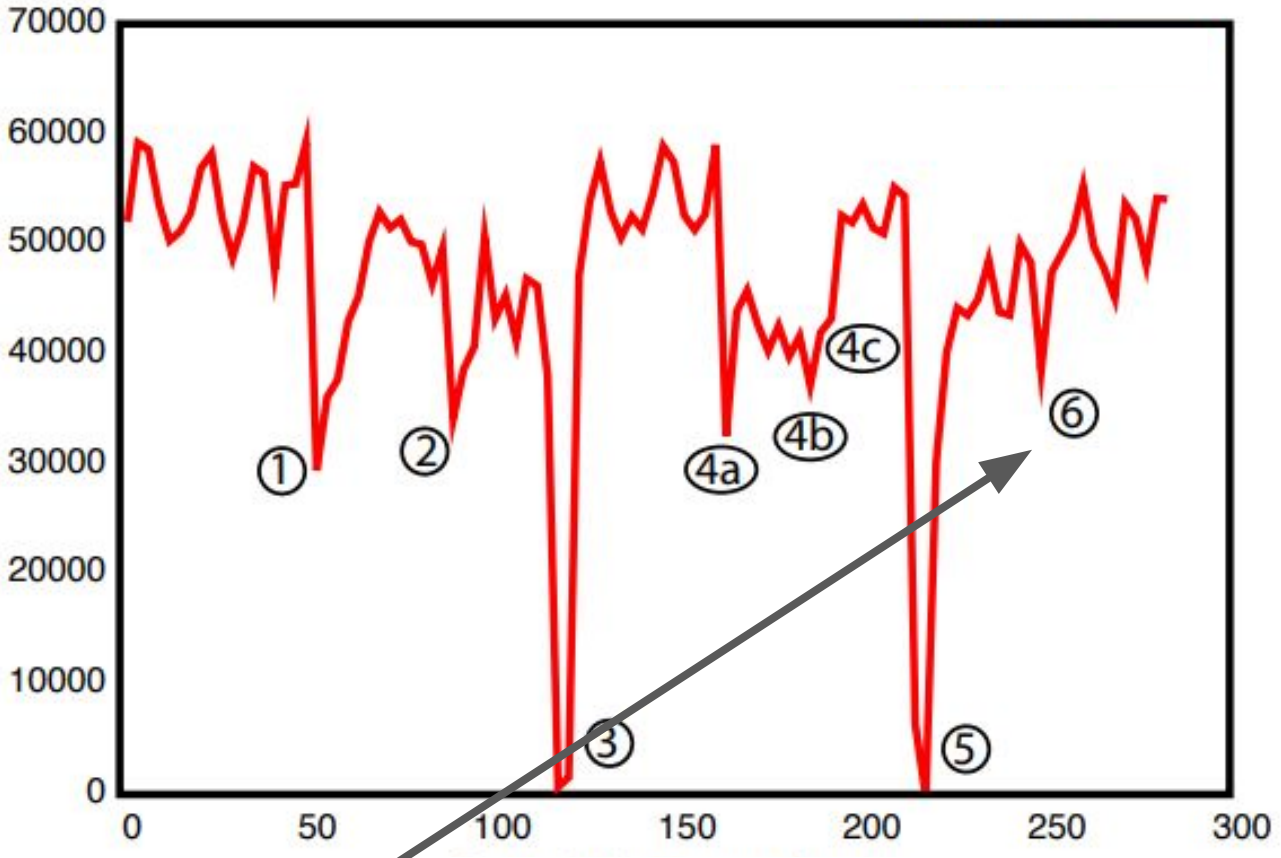
Throughput with Failures



Throughput with Failures



Throughput with Failures



Latency

	Number of servers			
Workers	3	5	7	9
1	776	748	758	711
10	2074	1832	1572	1540
20	2740	2336	1934	1890

Table 2: Create requests processed per second.

Thank you! Questions?