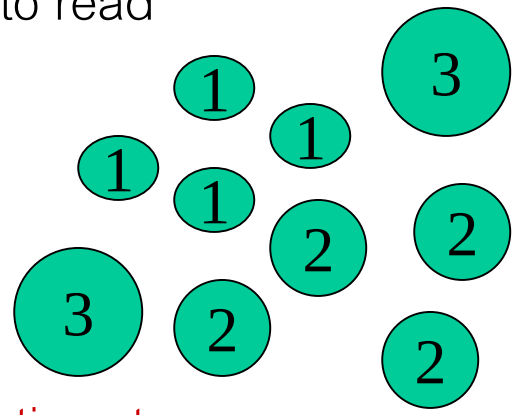
Lecture 10b:

Zookeeper

Raft

Voting Revisited

- Let V be the number of votes in the system
- Let W be the number of votes required to write
- Let R be the number of votes required to read
- Overlap Constraint (Requirement):
  - 1. V < R + W
- Recommended:
  - 2. V < 2 * W

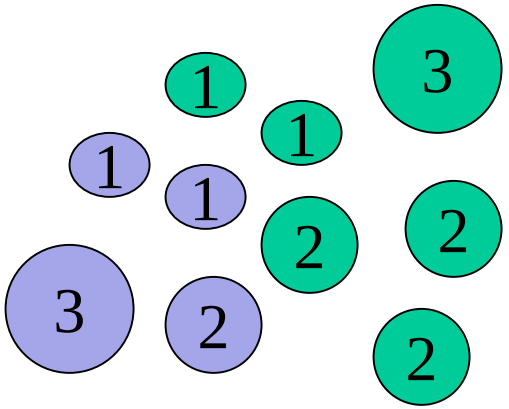


- Data must contain a version number or timestamp
- If version numbers used => 2. becomes a requirement!
- If constraints are met, then data will remain consistent.
- Note that votes can be arbitrarily assigned to servers in the system (i.e. weights can be assigned to servers)

D. Gifford, "Weighted Voting for Replicated Data" SOSP 1979

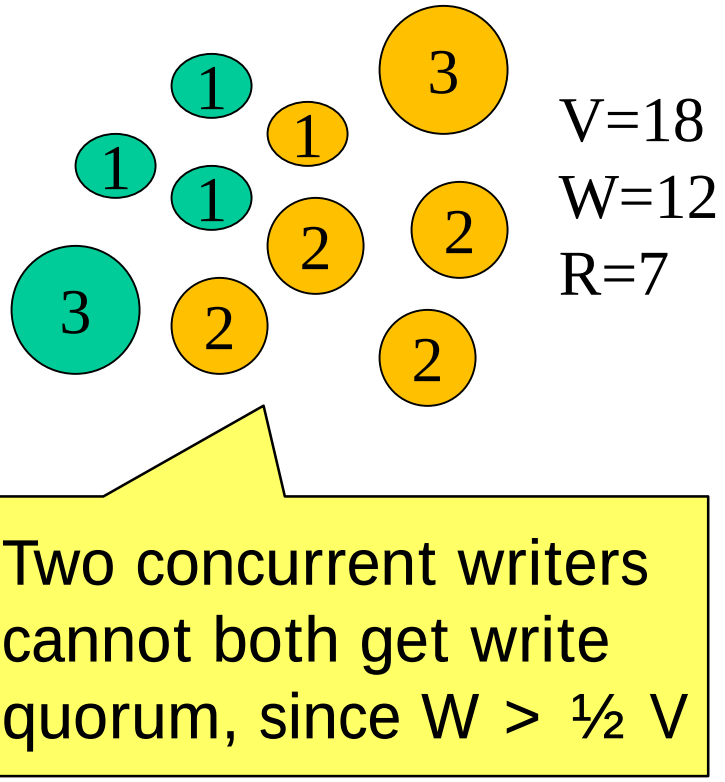
Example with version numbers

- R+W > V and W > V/2
  - Read and Write quorums overlap
  - Write more than ½.
- To read:
  - Request version numbers until R votes are collected
  - Overlap constraint means at least 1 voter has latest version. Read it.



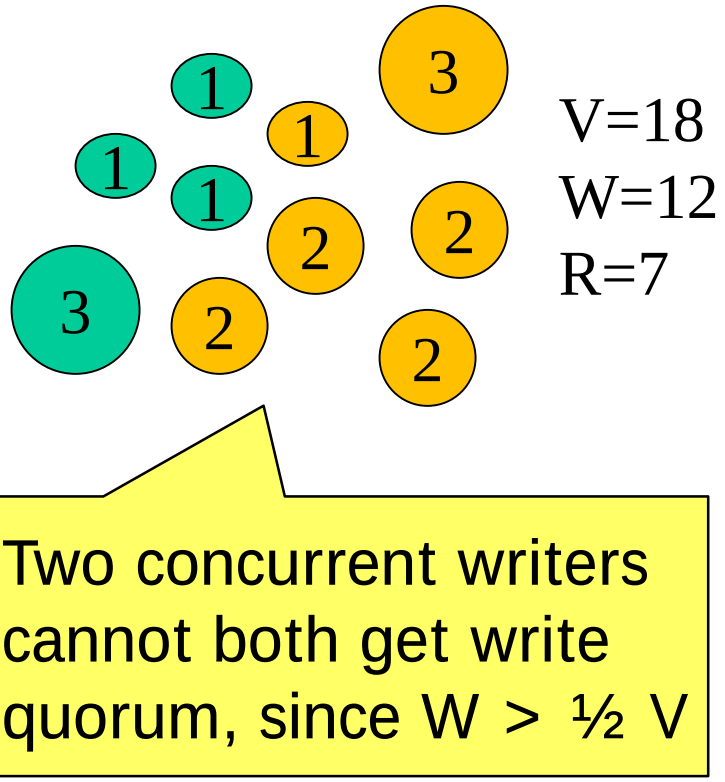
Example with version numbers

- To write:
  - Request version numbers until W votes are collected from servers with up-to-date copies
    - May require latest write to be propagated to more servers
    - Any set of voters with W votes must include at least 1 with latest version. Writer can detect if it needs more votes.
  - Apply update to servers in write quorum



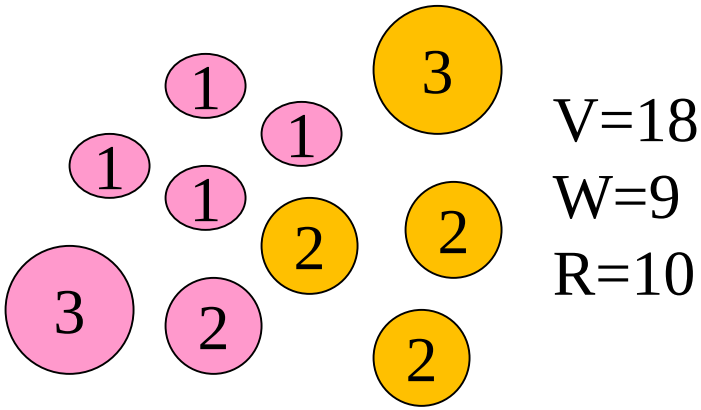
Example with version numbers

- To write:
  - Request version numbers until W votes are collected from servers with up-to-date copies
    - May require latest write to be propagated to more servers
    - Any set of voters with W votes must include at least 1 with latest version. Writer can detect if it needs more votes.
  - Apply update to servers in write quorum



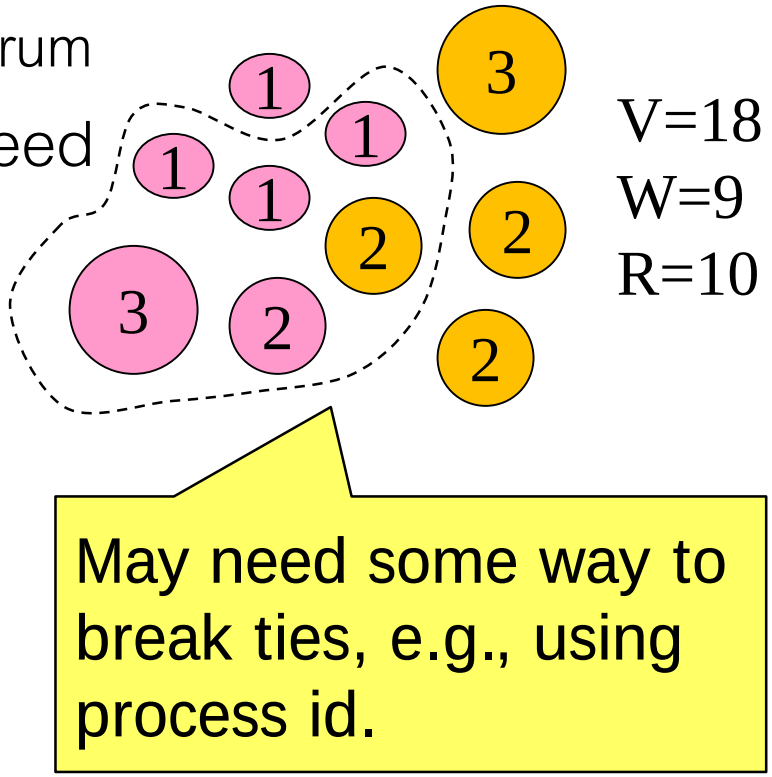
Version numbers with W <= V/2

- To write:
  - Request version numbers until W votes are collected from servers with up-to-date copies
  - Apply update to servers in write quorum
- 2 concurrent writers can both succeed in getting a write quorum
  - Suppose current version is “2”
  - P1: write(value=x, version=3)
  - P2: write(value=y, version=3)
- Reader can’t distinguish latest write



Timestamps with W <= V/2

- To write:
  - Collect W votes from servers to build write quorum (need not be up-to-date)
  - Apply update to servers in write quorum
- 2 concurrent writers can both succeed in getting a write quorum
  - P1: write(value=x, timestamp=T1)
  - P2: write(value=y, timestamp=T2)
- Read quorum must include at least one server with later timestamp
  - Last write wins



ZooKeeper

"ZooKeeper: Wait-free Coordination for Internet-Scale Systems"

- Patrick Hunt and Mahadev (Yahoo! Grid), Flavio Junqueira and Benjamin Reed (Yahoo! Research)
- USENIX Annual Technical Conference, 2010

- (can watch Ben Reed's USENIX presentation video: https://www.usenix.org/conference/usenix-atc-10/zookeeper-wait-free-coordination-internet-scale-systems)

What is ZooKeeper?

- *"…a service for coordinating processes of distributed applications."*

What kinds of coordination?

- *"…enables a high-performance service implementation."*

- from the paper

Coordination Services

- Leader election and group membership
  - Know which processes are alive and what those processes are in charge of
- Dynamic configuration changes
- Status monitoring
- Locks
  - provide mutually exclusive access to critical resource

What is ZooKeeper?

- *"…a service for coordinating processes of distributed applications."*

- *"…enables a high-performance service implementation."*

- from the paper

What aspects of performance are important?

Performance Considerations

- Large scalability
  - Easy to increase number of servers
  - Performance grows proportionally with server growth
- High throughput, low latency read requests
  - Support many users simultaneously
  - Expect reads more common than writes
- Wait-free
  - Slow or faulty clients can't hurt performance of faster clients
- Coordination kernel exposes API to give developers flexibility
  - Adapt coordination to application needs (end-to-end arg?)

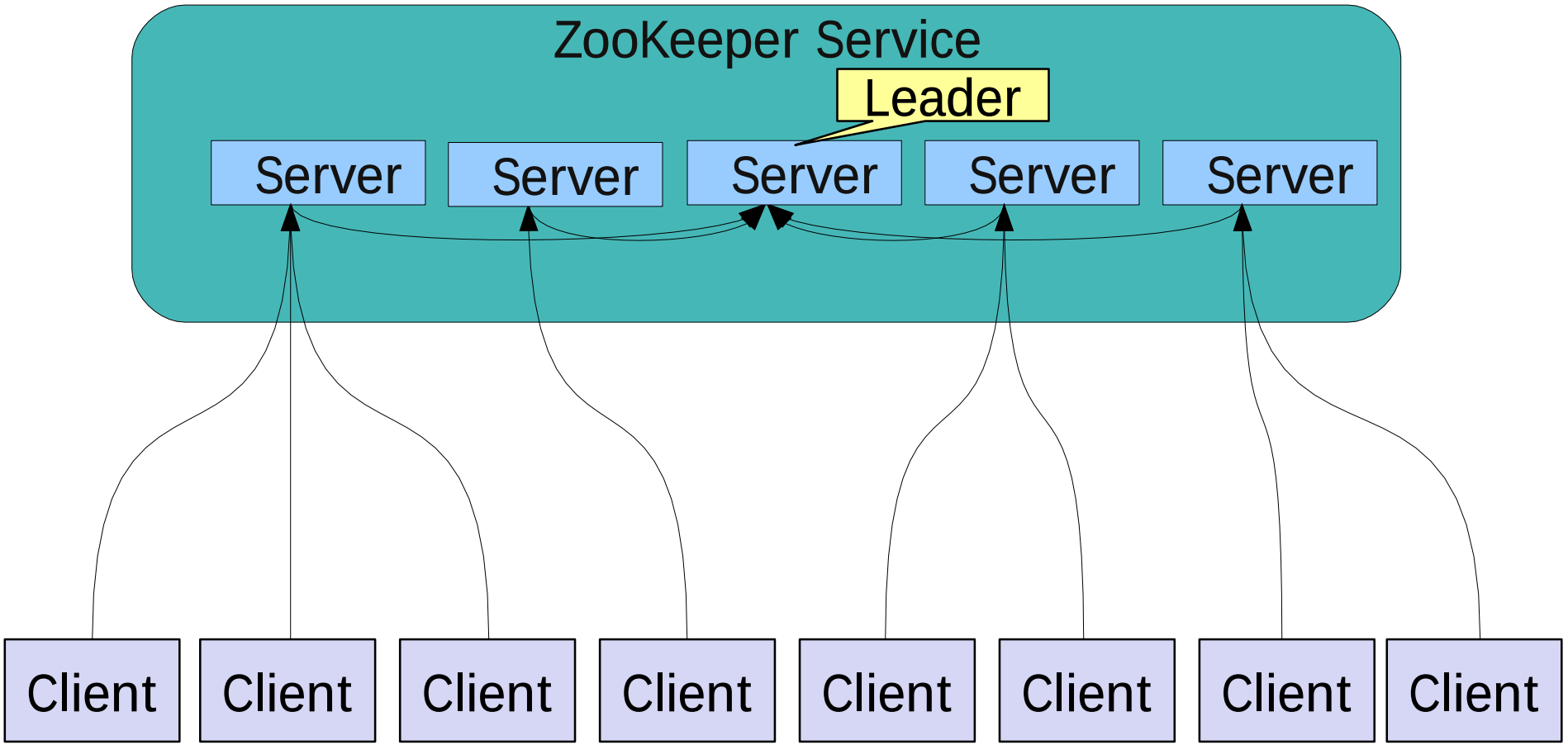
Terminology

- *Servers:* the set of processes that cooperate to provide the ZooKeeper service.

- *Clients:* users of the ZooKeeper service

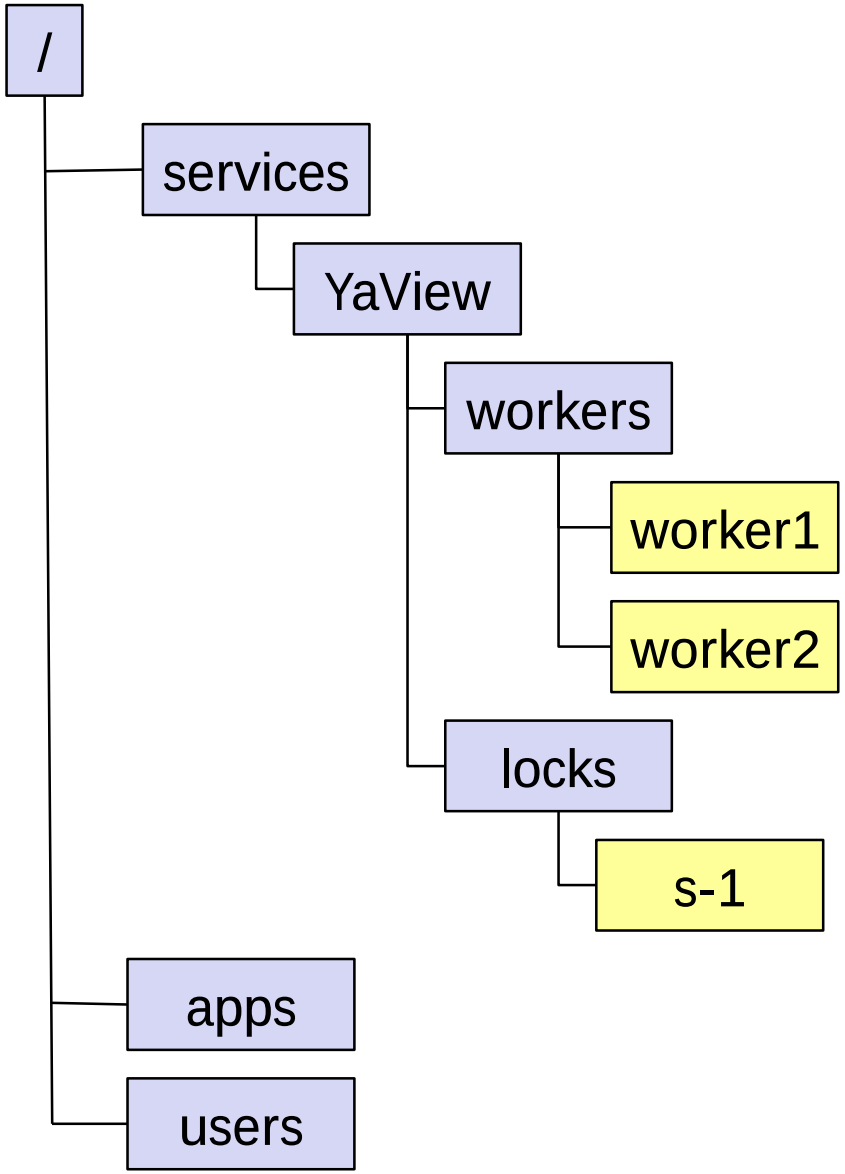
- Typically, a set of clients are themselves providing some distributed service.
  - E.g., Yahoo! Message Broker – distributed pub/sub service

Overall System Diagram



Data Model

- Hierarchical namespace
  - Like file system
  - Items in tree are called *znodes*
- Each znode has data and possibly children
- Data is always fully read or written
  - Metadata store, small

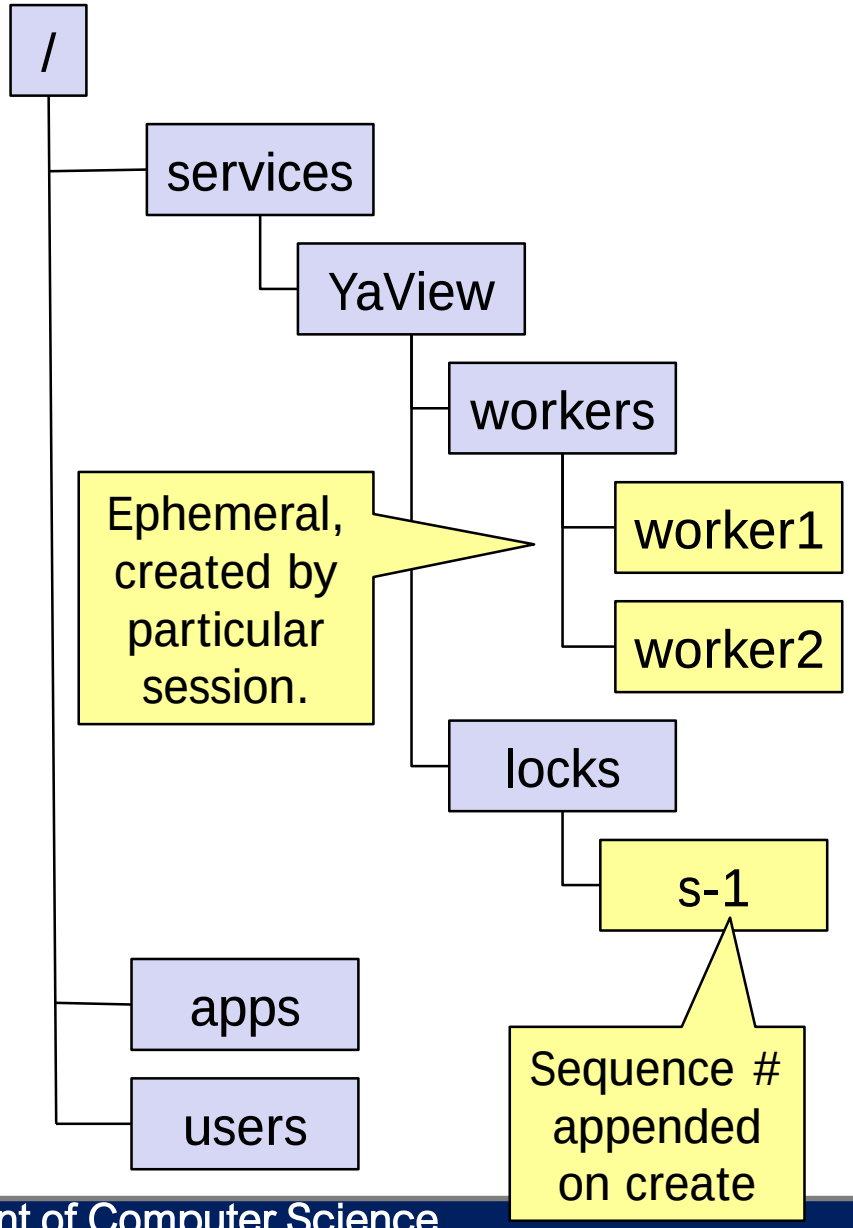


ZooKeeper API (simplified)

- String create(path, data, flags)
- void delete(path, expectedVersion)
- Stat setData(path, data, expectedVersion)
- (data, Stat) getData(path, watch)
- Stat exists(path, watch)
- String[] getChildren(path, watch)
- void sync()

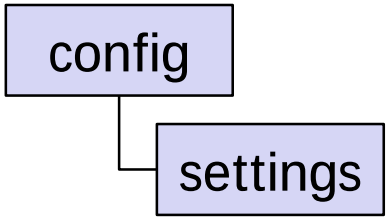
Create flags

- *Ephemeral*: znode deleted when creator fails or explicitly deleted
- *Sequence:* append a monotonically increasing counter



Change Notifications

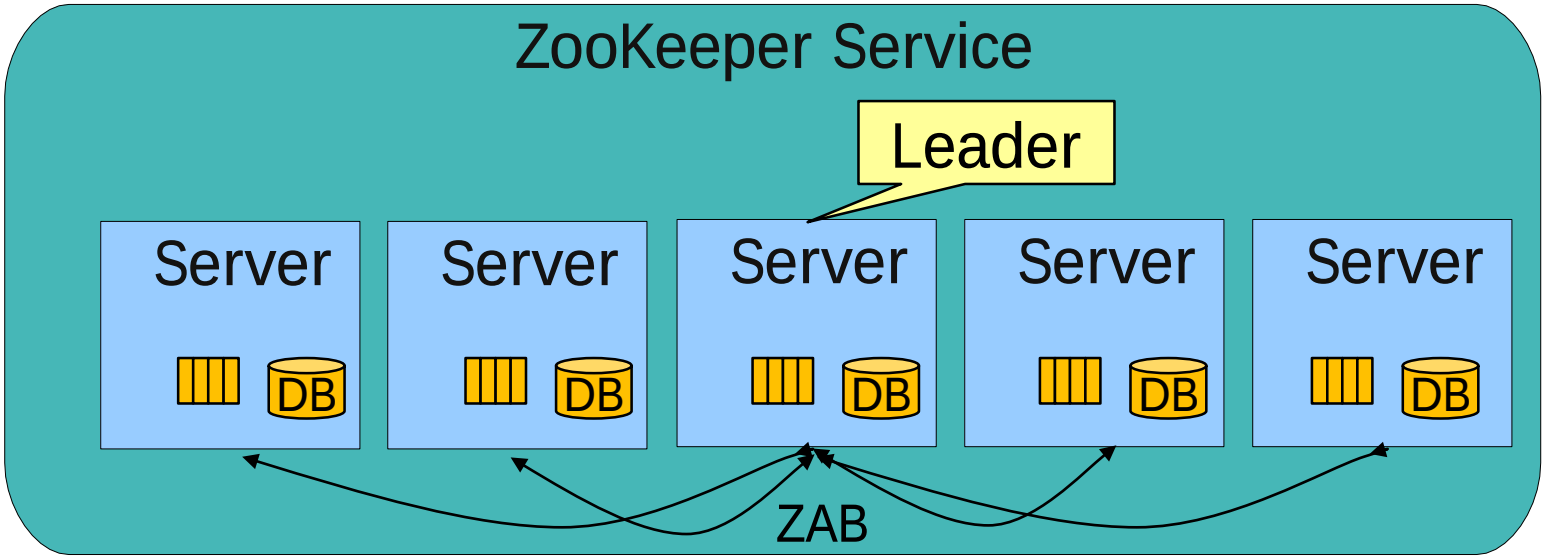
- Clients can cache results to improve performance
  - But don't want service to manage client caches
  - Don't block writes while invalidating cached data
- Solution: Clients can request notification of changes
- Example:
  - Workers get configuration
    - getData(".../config/settings", true)
  - Administrators change the configuration
    - setData(".../config/settings", newConf, -1)
  - Workers notified of change and get the new settings
    - getData(".../config/settings", true)



Ordering Guarantee

- Clients may have multiple outstanding requests
- Updates are *linearizable*
- Each clients' requests are handled in FIFO order

Server Internals



- Replicated in-memory database + persistent log
  - Provides highly-available service
  - Clients connect to and read from any server
- Linearizable updates
  - Handled by leader
  - Uses atomic broadcast protocol called ZAB

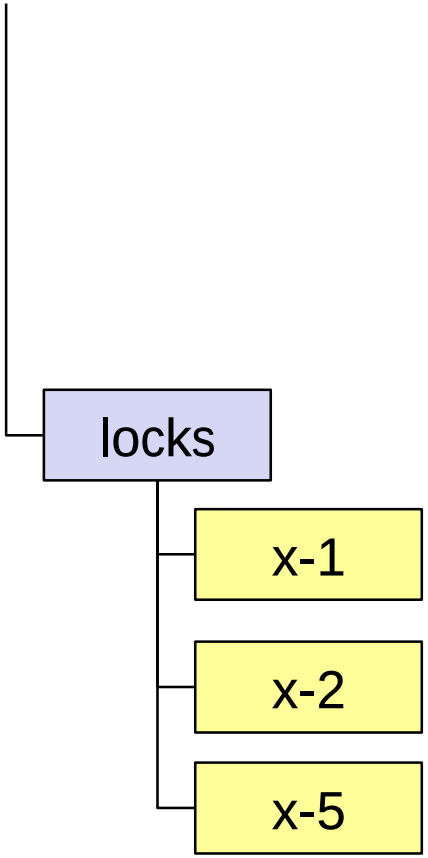
Example: Lock Service on ZooKeeper

Lock(path):

1. id = create(“.../locks/x-”, SEQUENCE|EPHEMERAL)
2. getChildren(“.../locks”/, false)
3. If id is the 1st child, exit
4. exists(name of last child before id, true)
5. if does not exist, goto 2
6. wait for event notification
7. goto 2

Unlock(path):

1. delete(id,…)



Raft

- Stanford presentation from Diego Ongaro and John Ousterhout
  - Slides: https://ramcloud.stanford.edu/~ongaro/userstudy/raft.pptx
  - Listen to Ousterhout's presentation of these slides: https://www.youtube.com/watch?v=YbZ3zDzDnrw&feature=youtu.be
- Also, Paxos lecture, quiz questions, results, Raft paper, etc. at: https://ramcloud.stanford.edu/~ongaro/userstudy/