CS 138: ZooKeeper

ZooKeeper

- A "Coordination Kernel"
  - provides primitives making possible various sorts of distributed coordination
    - group membership
    - leader election
    - dynamic configuration
    - status monitoring
    - queueing
    - barriers
    - critical sections
- Similar to Google's Chubby, but more flexible

Model

- File system with simple API
  - whole file reads and writes
    - files are *znodes*
  - no open/close
- Reads are much more frequent than writes
- Writes are *linearizable*
  - sequentially consistent
  - totally ordered (determined by primary)
- FIFO client order
  - all client requests executed in order sent by client

Model (continued)

- Znodes
  - regular znodes
    - may have children
    - disappear when deleted
  - ephemeral znodes
    - may not have children
    - disappear when deleted or when creator terminates
  - sequential flag
    - property of regular znodes
    - children have strictly increasing integer appended to their names

Model (continued)

- Reads
  - serializable
  - may return stale data
- Watches
  - client issues read request with *watch* flag
    - retrieves current data
    - will be notified when znode changes
      - guaranteed notification will occur before it sees effects of change
- Pipelines
  - writes are asynchronous

Examples (1)

- Configuration management
  - configuration info stored in collection of znodes
  - *ready* znode indicates info is valid
  - new leader takes over
    - deletes *ready* znode
    - updates 5000 other znodes with new config info
    - recreates *ready* znode
    - other processes read new info
    - all updates are pipelined
      - latency is low

Examples (2)

- Rendezvous
  - client starts up master and worker processes asynchronously
    - master eventually establishes addresses and ports for use by clients to connect to it
    - client has gone on to other things
      - master must get this info to workers
  - client provides rendezvous znode z_r
    - server writes connection info to it
    - workers read z_r, with watch set to true
      - if server hasn't written yet, clients are notified when it does

Examples (3)

- Group membership
  - group is represented by znode z_g
  - members create ephemeral children of z_g with sequential flag
    - establish uniquely named children that disappear when the member disappears
  - group membership determined by listing children of z_g

Examples (4)

- Simple locks
  - use “lock” znodes
    - create with ephemeral flag
      - succeeds iff doesn’t already exist
      - disappears if creator fails
  - subject to “herd” problems
    - those waiting for lock all attempt to grab it at once when released

Examples (5)

- **Herd-free simple locks**

```
Lock
   1. n = create(z_lock + "/lock-", EPHEMERAL|
      SEQUENTIAL)
   2. C = getChildren(z_lock, 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)
```

Examples (6)

- Double barrier
  - No process enters until all are ready
  - No process exits until all are ready

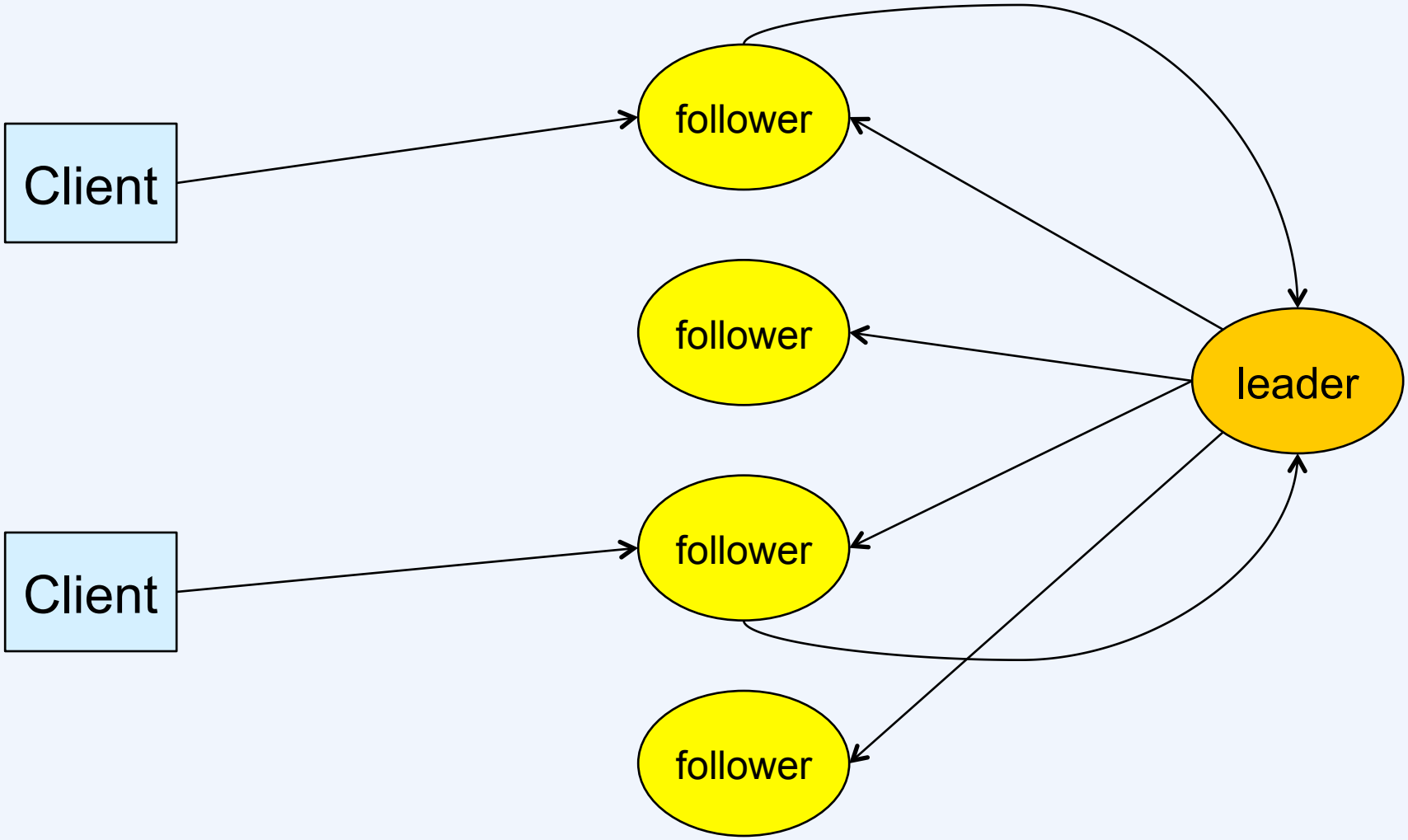
  - znode z_b represents barrier
    - process creates child when it wants to enter
      - may enter only when enough children
    - process deletes child when it wants to exit
      - may exit only when no more children

Examples (7)

- **Leader election**

```
1. getData("../workers/leader", true)
2. if successful follow the leader
   described in the data and exit
3. create("../workers/leader",
   hostname, EPHEMERAL)
4. if successful lead and exit
5. goto step 1
```

Implementation



The End …

- HW 4 out today, due May 5
- Raft returned this morning
- PuddleStore design proposals back
- PuddleStore due May 9
- Final Exam 2pm, May 18 in Smith Buonano 106
  - covers entire course
  - help session 5pm, May 16 in TBA