CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

LEADER ELECTION

Lecture C

ELECTION IN CHUBBY AND ZOOKEEPER

Can use Consensus to solve Election

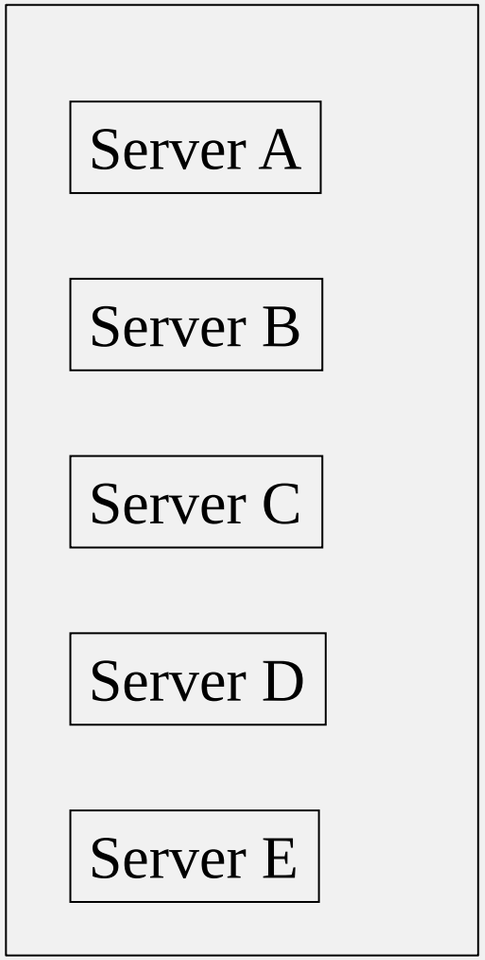
- One approach
  - Each process proposes a value
  - Everyone in group reaches consensus on some process P*i*'s value
  - That lucky P*i* is the new leader!

Election in Industry

- Several systems in industry use Paxos-like approaches for election
  - Paxos is a consensus protocol (safe, but eventually live): elsewhere in this course
- Google's Chubby system
- Apache Zookeeper

Election in Google Chubby

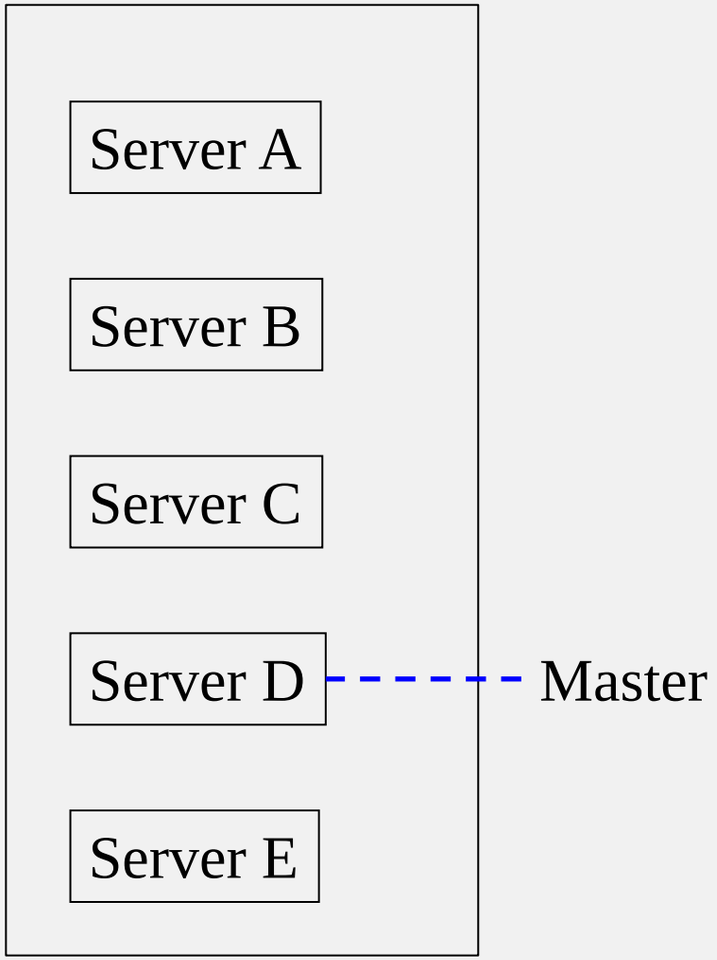
- A system for locking
- Essential part of Google's stack
  - Many of Google's internal systems rely on Chubby
  - BigTable, Megastore, etc.
- Group of replicas
  - Need to have a master server elected at all times



Reference: http://research.google.com/archive/chubby.html

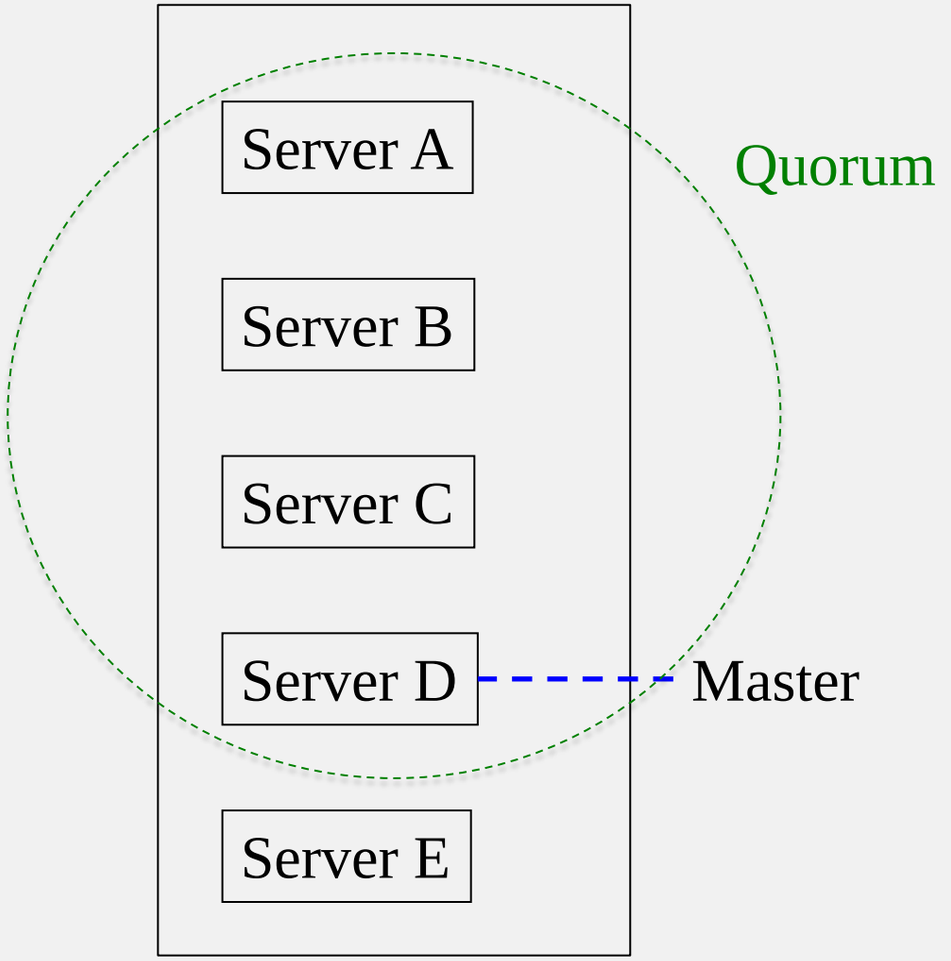
Election in Google Chubby (2)

- Group of replicas
  - Need to have a master (i.e., leader)
- Election protocol
  - Potential leader tries to get votes from other servers
  - Each server votes for at most one leader
  - Server with *majority* of votes becomes new leader, informs everyone



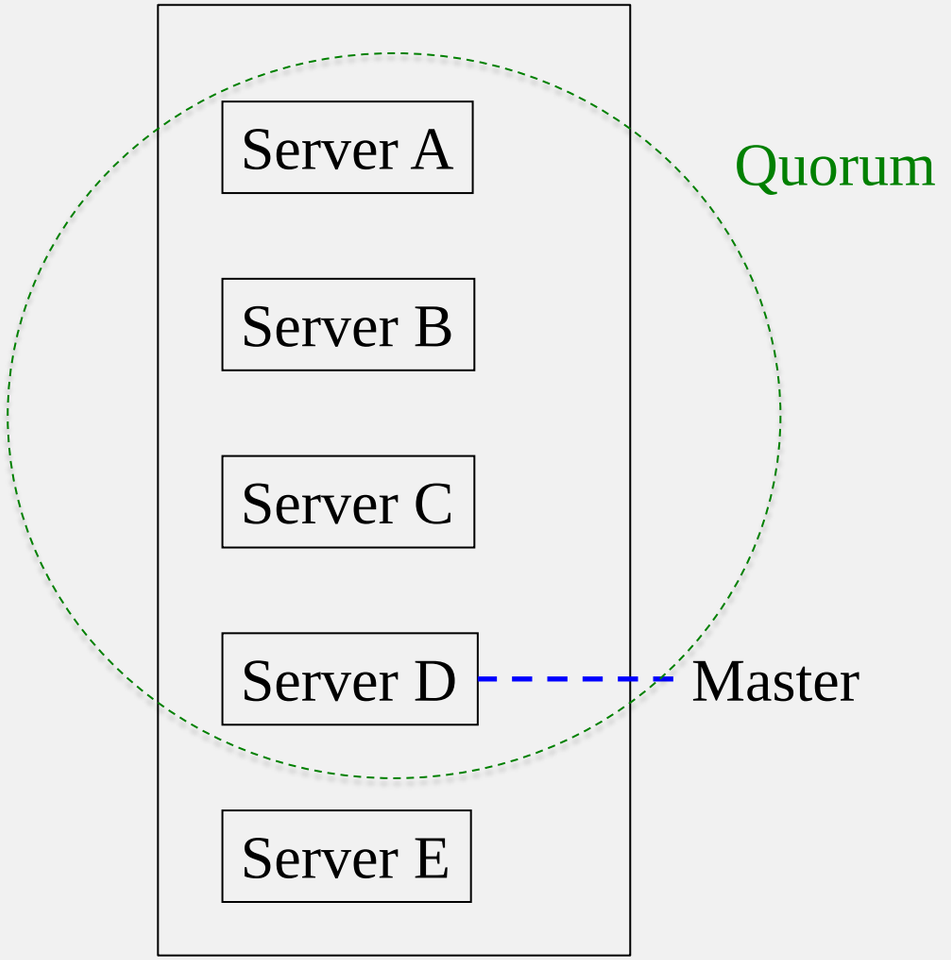
Election in Google Chubby (3)

- Why safe?
  - Essentially, each potential leader tries to reach a *quorum* (should sound familiar!)
  - Since any two quorums intersect, and each server votes at most once, cannot have two leaders elected simultaneously
- Why live?
  - Only eventually live! Failures may keep happening so that no leader is ever elected
  - In practice: elections take a few seconds. Worst-case noticed by Google: 30 s



Election in Google Chubby (4)

- After election finishes, other servers promise not to run election again for “a while”
  - “While” = time duration called “Master lease”
  - Set to a few seconds
- Master lease can be renewed by the master as long as it continues to win a majority each time
- Lease technique ensures automatic re-election on master failure



Election in Zookeeper

- Centralized service for maintaining configuration information
- Uses a variant of Paxos called Zab (Zookeeper Atomic Broadcast)
- Needs to keep a leader elected at all times

- http://zookeeper.apache.org/

Election in Zookeeper (2)

- Each server creates a new *sequence number* for itself
  - Let's say the sequence numbers are ids
  - Gets highest id so far (from ZK file system), creates next-higher id, writes it into ZK file system
- Elect the highest-id server as leader

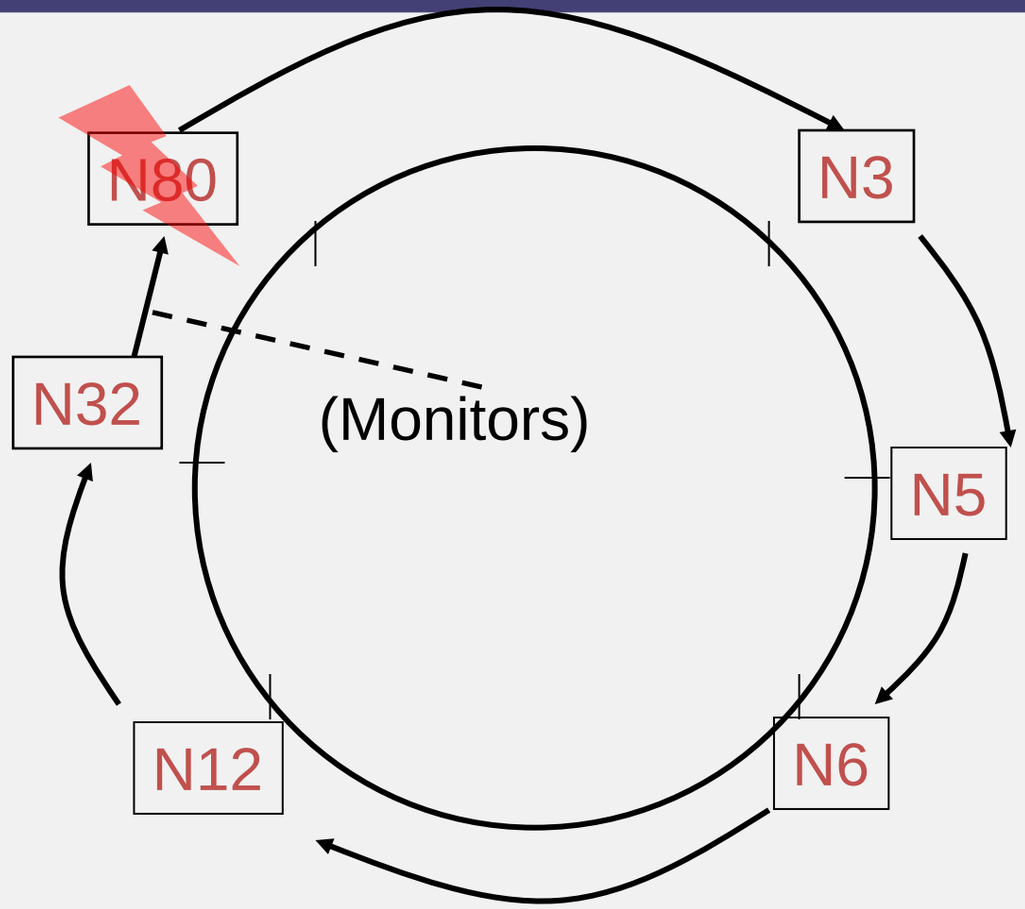
Election in Zookeeper (3)

Failures:

- One option: everyone monitors current master (directly or via a failure detector)
  - On failure, initiate election
  - Leads to a flood of elections
  - Too many messages

Election in Zookeeper (4)

- Second option (implemented in Zookeeper)
  - Each process monitors its next-higher id process
  - **if** that successor was the leader and it has failed
    - Become the new leader
  - **else**
    - wait for a timeout, and check your successor again



Election in Zookeeper (5)

- What about id conflicts? What if leader fails during election?
- To address this, Zookeeper uses a *two-phase commit* (run after the sequence/id) protocol to commit the leader
  - Leader sends NEW_LEADER message to all
  - Each process responds with ACK to at most one leader, i.e., one with highest process id
  - Leader waits for a majority of ACKs, and then sends COMMIT to all
  - On receiving COMMIT, process updates its leader variable
- Ensures that safety is still maintained

Next

- Another classical algorithm: Bully algorithm