

In search of an understandable consensus algorithm

Diego Ongaro and John Ousterhout
Stanford University
SEDCL Forum
January 24, 2013

How did we end up here?

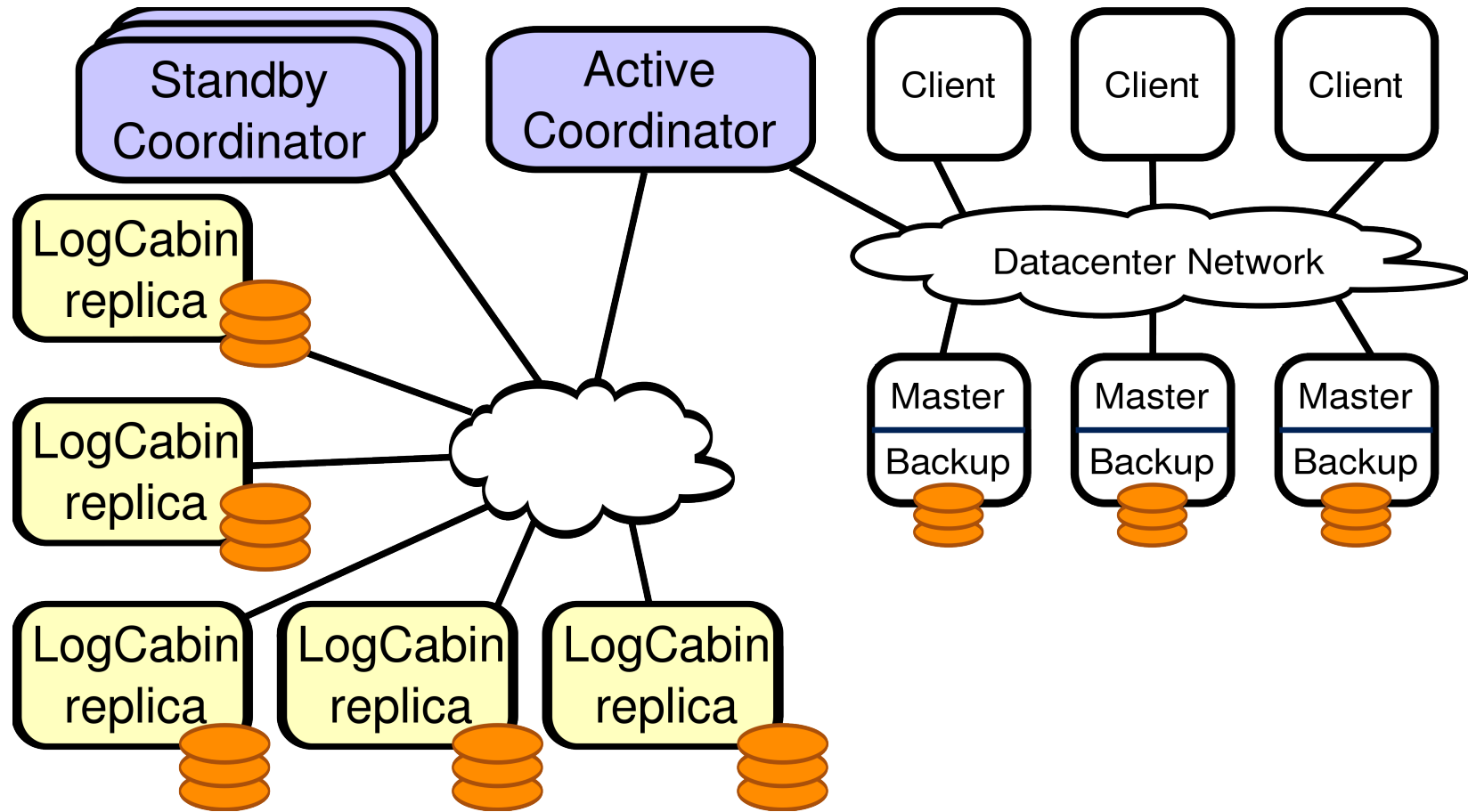
- RAMCloud relies on a single cluster coordinator
 - Need to elect a new one when it fails
 - Need a reliable place to store its state
- You told us to use ZooKeeper in April 2010
- But ZooKeeper is hard to use
 - so we started LogCabin
- And Paxos is hard to understand
 - so we started Raft



Outline

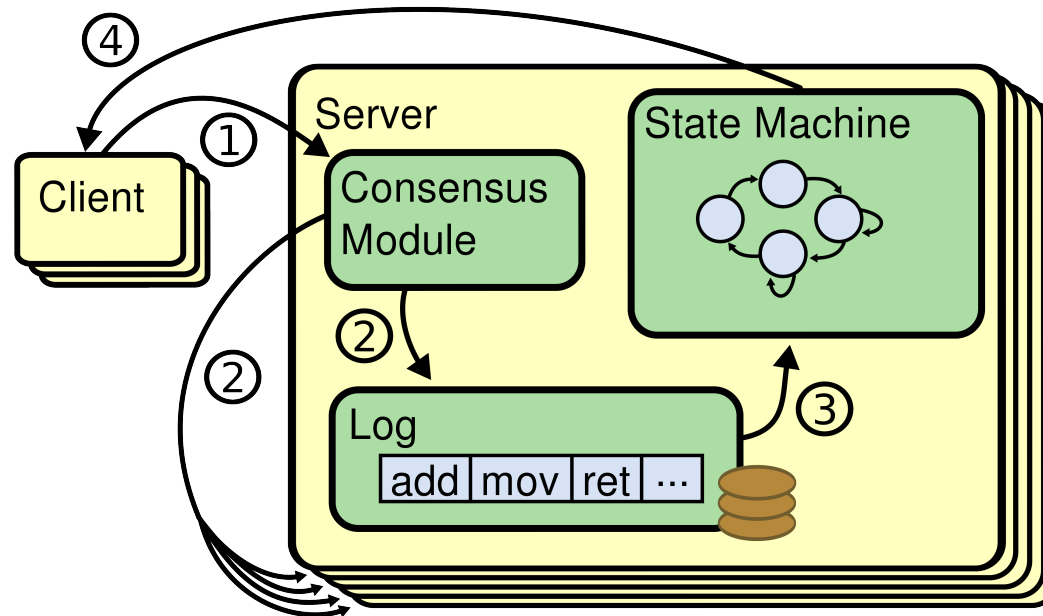
- Introduce the problem Paxos and Raft solve
- Discuss where we think Paxos went wrong and why Raft is easier to understand
- Give an overview of Raft
- Go into detail on Raft's leader election
- Project status

Birds-eye view



- Configuration service is available when a majority of replicas is available

Replicated state machines

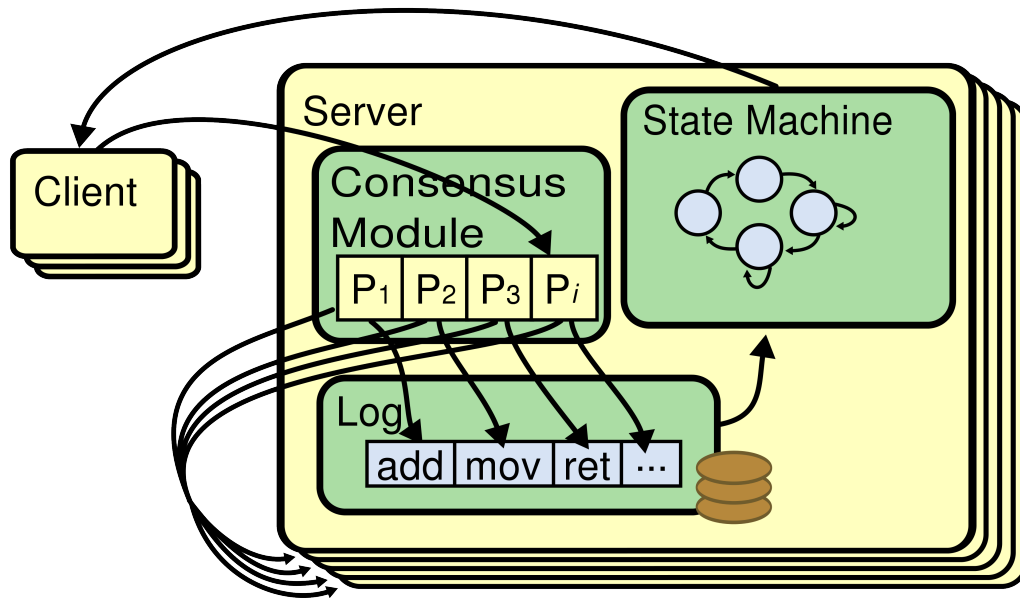


- State machine provides primitives for leader election, small amount of storage, etc
 - Easy to implement
 - Interface is application-specific
- Replicated log feeds commands to state machine
 - Same log → same sequence of states, outputs
- Raft and Paxos are two consensus algorithms to manage the replicated log

What's wrong with Paxos?

- Hard to understand
 - Not many computer scientists understand it
 - My attempt at teaching Paxos at last year's SEDCL retreat left *everyone* in the audience in fear
- Hard to implement
 - Requires complex “optimizations” to be practical
 - Leaves many “details” unspecified
 - “There are significant gaps between the description of the Paxos algorithm and the needs of a real-world system.”
 - Chandra, et al. *Paxos Made Live*

Paxos decomposition



- Basic Paxos (single-decree Paxos) solves a smaller problem: it manages a single replicated log entry
- Running an instance of the algorithm for each log entry results in a replicated log
- Optimizations that make this practical are called Multi-Paxos

Why is this decomposition bad?

- Basic Paxos
 - Suitable for theory, not great for practice
 - The problem of agreeing on a single value is hard to relate to (this is what theoreticians call *consensus*)
 - The two phases of the algorithm are hard to separate
- Multi-Paxos
 - Requires reasoning across instances of Basic Paxos
 - Fundamentally different behavior from Basic Paxos
 - Chooses a leader as an optimization, but does not use it to simplify the algorithm
 - No advantage to concurrent operation when the log is fundamentally sequential

Can we design a more understandable
consensus algorithm?

How is Raft more understandable?

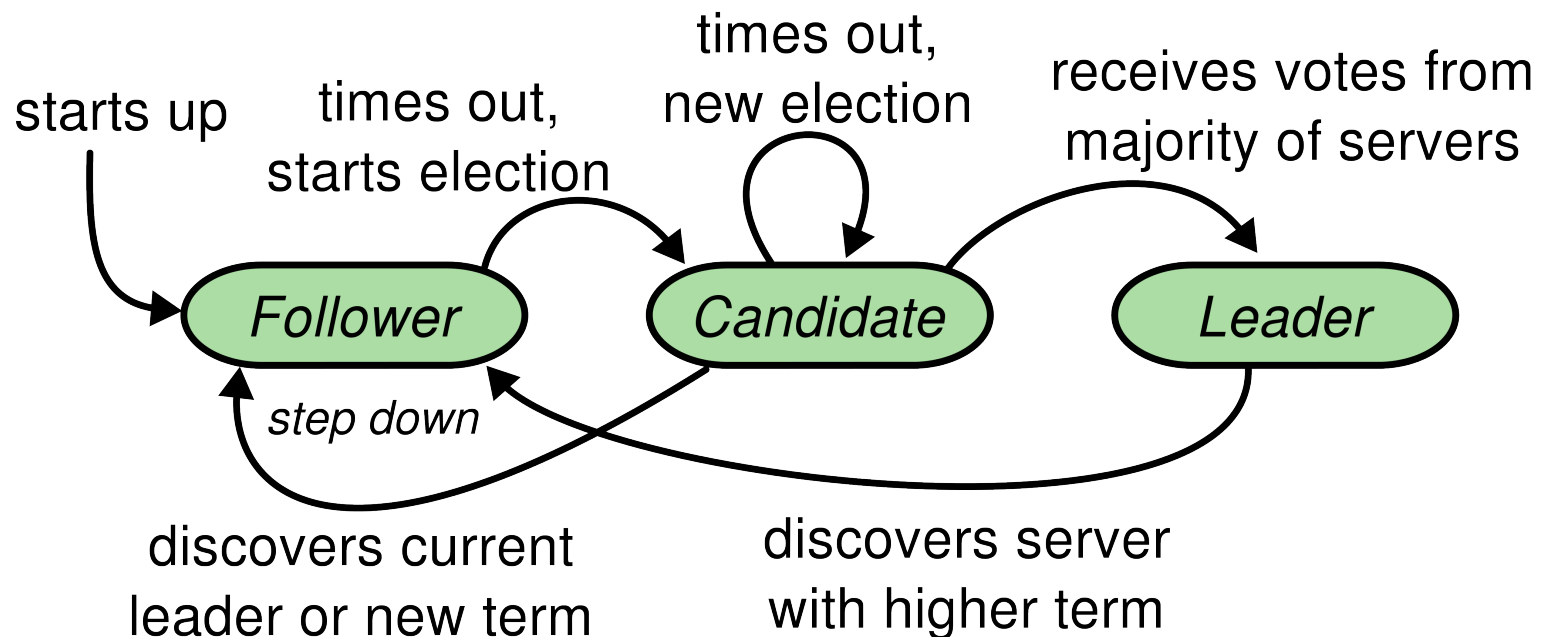
- Solves the real problem
 - Manages the replicated log directly
 - Uses sequential ordering
- Centralizes decisions
 - The leader manages all changes to the logs
 - Other servers are completely passive
- Decomposes into subproblems well
- Ready to be implemented (and actually implemented in C++)
 - RPCs are well-defined and small in size. There's just two of them.
 - Includes practical considerations

Raft overview

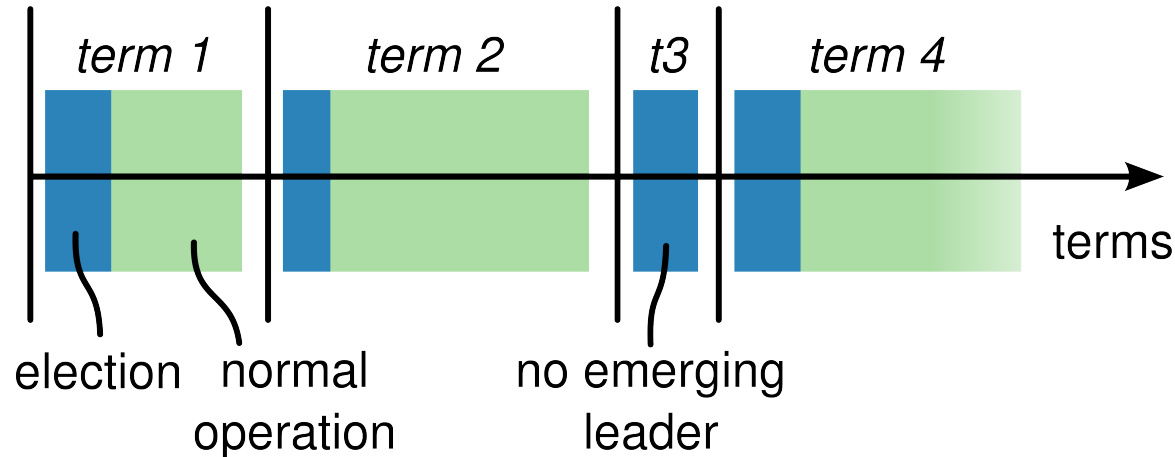
- Leader election:
 - elects a leader when the cluster doesn't have one
- Replication:
 - the leader orders client requests into the log and replicates them
- Restoring consistency after a crash:
 - a new leader cleans up temporary inconsistencies that arise when leaders crash
- Eliminating zombies:
 - a new leader prevents zombie leaders from modifying the replicated log

Server states

- Each server is either a follower, a candidate, or a leader
- In normal operation, there is exactly one leader and all other servers are followers
- Followers are passive



Terms



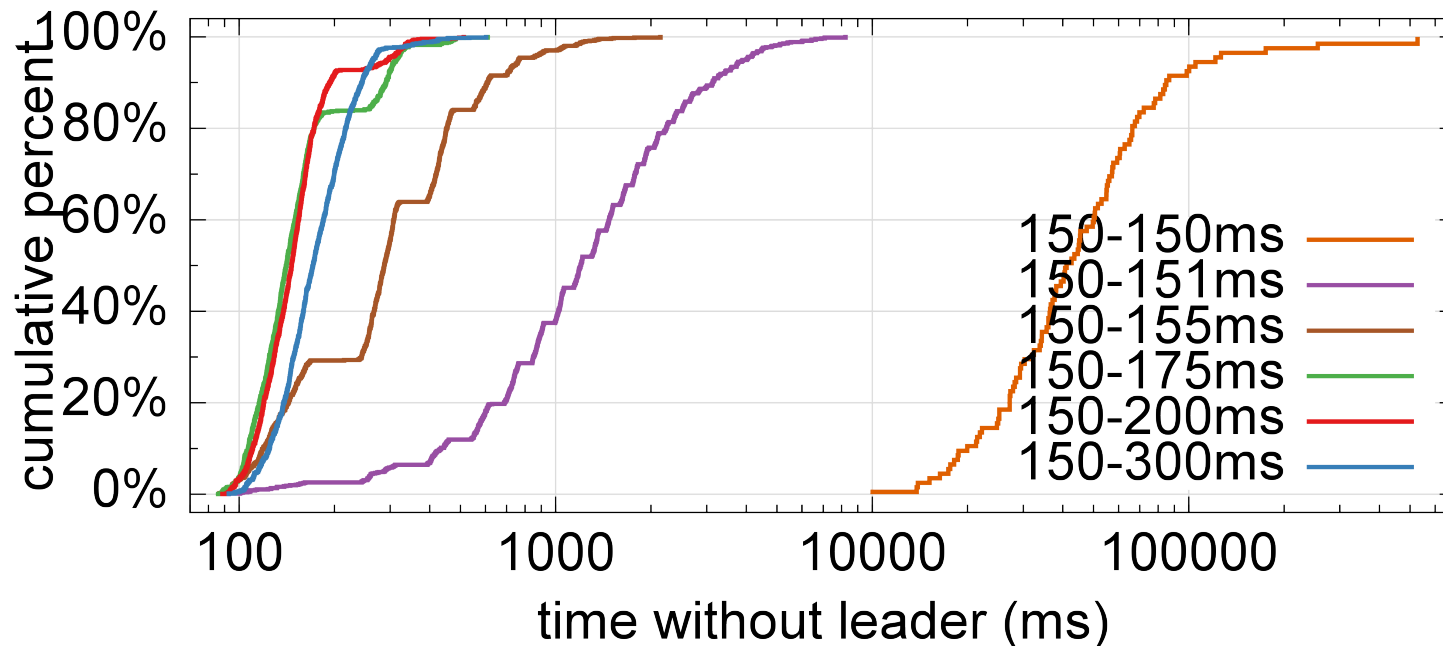
- Each term begins with an election
- Usually an election succeeds in choosing a leader for the rest of the term
- In case of a split vote, the term will end with no leader, and a new term with a new election starts shortly
- Leader election guarantees that there is at most one leader per term

Leader election

- Leaders send periodic heartbeats to all followers to maintain their authority
- After an *election timeout*, a follower begins an election
 - Increments its current term
 - Transitions to the candidate state
 - Issues RequestVote RPCs in parallel to the other servers
- Servers may only vote once per term, first-come-first-served
- Three possible outcomes:
 - It wins the election by receiving votes from a majority → becomes leader
 - Another server establishes itself as a leader → returns to follower
 - Another election timeout goes by (split vote) → new election

Randomized election timeouts

- Purpose: prevent split votes from occurring forever
- Election timeouts are chosen from a uniform range



- Previously considered more complex approaches
 - Server ranks – subtle bugs
 - Exponential random backoff – unnecessary

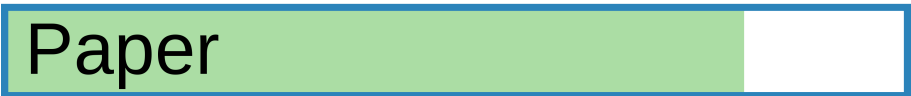
Is Raft easier to understand than Paxos?

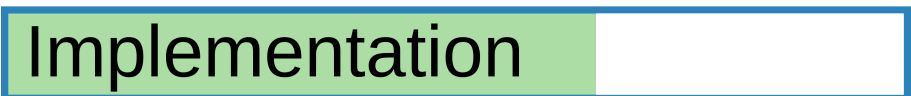
- NSDI PC doesn't think so, but they're Paxos experts!
- Running an experiment to find out – science!
- Participants are students of David Mazieres's Advanced OS class
- David will teach a lecture on Paxos, John will teach a lecture on Raft
- Students will be quizzed to determine which one they learn better
- Two groups allow us to factor out differences in individuals:
 - Raft video and quiz, then Paxos video and quiz
 - Paxos video and quiz, then Raft video and quiz

Project status

- Raft is implemented in LogCabin (~1500 lines of C++)
- Ankita is using it for RAMCloud's coordinator
- Code and paper draft available on RAMCloud wiki

Raft algorithm 

Paper 

Implementation 

User study 

Correctness proof 

Conclusions

- We think Raft is more understandable than Paxos
 - Solves the real problem
 - Decomposes well
- Finding a simple and understandable solution is hard
 - Need to be open to changing your mind
- The end result is much more valuable
 - Easier to learn, discuss, implement, and extend