Raft Consensus

Consensus Problem

- A fundamental problem in distributed computing is to achieve overall system reliability in the presence of a number of faulty processes.
- Agree on some data value that is needed during computation
- Examples of applications of consensus include whether to commit a transaction to a database, agreeing on the identity of a leader, state machine replication, and atomic broadcasts.
- The consensus problem requires agreement among a number of processes for a single data value.
- Consensus protocols must be fault tolerant.
- Paxos, RAFT, ...

Paxos

- Born in 1989 by Lamport.
- Paxos is a family of protocols for solving consensus in asynchronous distributed systems.
- Consensus protocols are the basis for the state machine replication approach to distributed computing.
- Paxos describes the actions of the processes by their roles in the protocol: client, leader, acceptor, proposer, and learner.
- The protocol proceeds over several rounds (Prepare, Promise, Accepts).
- Quorums express the safety properties of Paxos by ensuring at least some surviving processor retains knowledge of the results.

Note

Originally, did not specify the actual requirements necessary to build a real system, including areas such as leader election, failure detection, and log management. Paxos Made Simple in 2001 useful for practitioners Still, is not as prescriptive as Raft is.

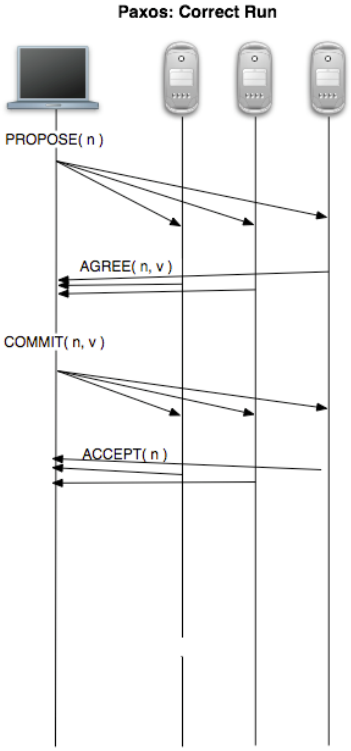
Paxos cont'd (Safety)

- Only a single value must be chosen
- A server never learns a value has chosen unless it really has been

Paxos cont'd (Liveness)

- Some proposed value is eventually chosen.
- If a value is chosen, servers eventually learn about it.

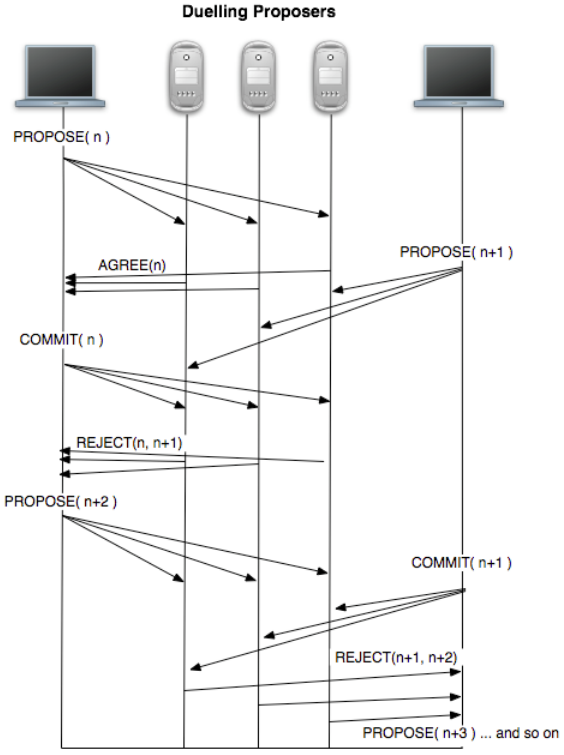
Paxos (Normal operation)



Note

- The first is ordering the proposals so that it may be determined which of two proposals should be accepted.
- The second improvement is to consider a proposal accepted when a majority of acceptors have indicated that they have decided upon it.

Paxos (Duelling proposers)



Paxos (Drawbacks)

- Paxos is exceptionally difficult to understand. As a result, there have been several attempts to explain Paxos in simpler term.
- Paxos is difficult implement in practical systems.
- "Paxos Made Simple" paper in 2001 is still not prescriptive as Raft paper.
- There are several implementations of Paxos - {Fast, Multi, Cheap}-Paxos

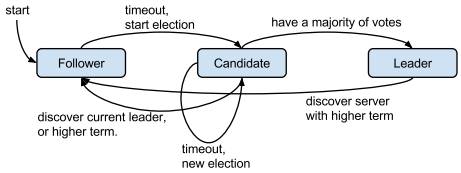
Raft

- Raft is a consensus algorithm designed as an alternative to Paxos.
- Designed to be more understandable.
- Less complexity and mechanism.
- Uses a Leader and Followers.
- Mechanisms for coordinating changes to cluster membership.
- It can achieve joint consensus using two overlapping majorities.

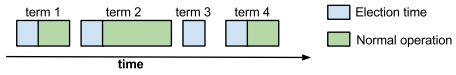
Raft (Server states)

At any given time, each server is either:

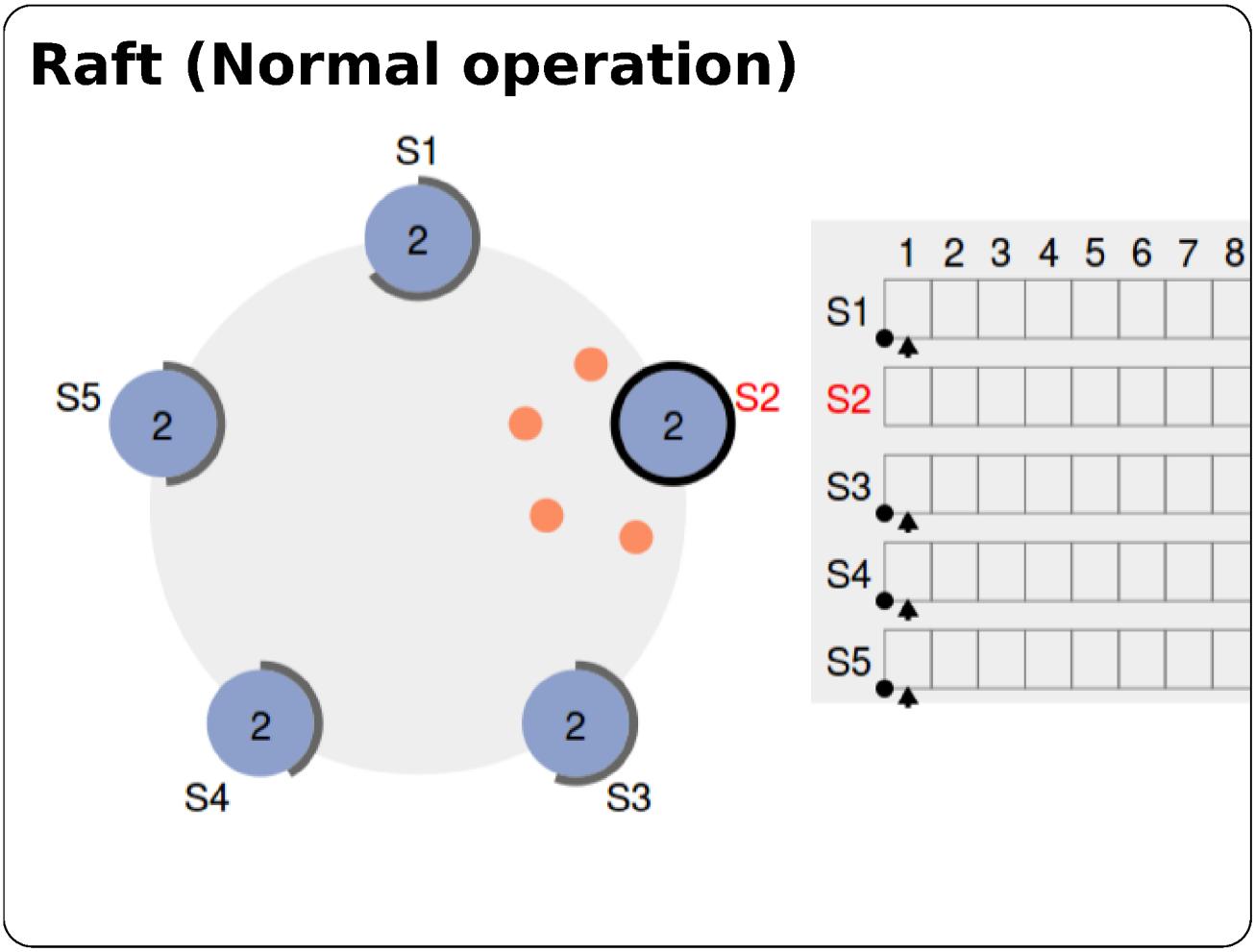
- Leader: handles all client interactions, log replication.
- Follower: completely passive (issues no RPCs, responds to incoming RPCs)
- Candidate: used to elect a new leader



Raft (Terms)



- Time divided into terms
- At most 1 leader per term
- Some terms have no leader (failed election)
- Each server maintains current term value
- Key role of terms: identify obsolete information


Raft (Normal operation)
S1
2
S5
2
S2
2
2
S4
2
S3
1 2 3 4 5 6 7 8
S1
S2
S3
S4
S5

Raft (Leader election)

Leader election is much more complicated than in Paxos

- Safety: Allow at most one leader per term.
- Liveness: some candidate must eventually win.

Raft vs Paxos

Paxos:

- A process can have the role of Proposer, Acceptor, and Learner.
- Use of optional timeouts to guarantee progress.
- Lots of messages exchanged in a correct run.
- Simple leader change.
- Sequence numbers to guarantee order of messages.
- The leader has the biggest proposal number.

Raft:

- A process can be candidate, follower, or leader.
- Requires timeouts for progress.
- The leader has the most recent log.
- Messages just go from the leader to the followers.
- Time is divided in terms.
- Use of heartbeats.
- Easier to implement.

That's all folks!