Consensus
An Introduction to Raft
con·sen·sus
/kən'sensəs/

Agreeing upon state across distributed processes even in the presence of failures.
Problem

- Distributed System
- Consistency
- Partition tolerance
Solution

- Quorum
- Replicated State Machines
Consensus Data
We are sacrificing Availability
Why not Paxos?

- Difficult to understand
- Not practical enough to implement
Raft
A Practical Paxos
Components

• Consensus Module
• State Machine
• Log
Consensus Module

- Roles: Leader, Follower, and Candidate
- Time is divided into Terms
- Commands: RequestVote and AppendEntries
Leader
Accept commands from clients, commit entries, and send heartbeats

Follower
Replicate state from leaders and vote for candidates

Candidate
Start and handle leader elections
Follower

- Times out, starts election

Candidate

- Times out, restarts election
- Discovers new leader, steps down
- Discovers current leader or new leader, steps down

Leader

- Wins election
- Discovers new leader, steps down
Term

Higher numbers are used to determine leaders and check log entries. The term is incremented each time an election is started. Any command with an old term is ignored.
Example
Happy Log Entry
Role: **Leader**
Term: 1
Commit Index: 0
Log: []

Role: **Follower**
Term: 1
Commit Index: 0
Log: []

Role: **Follower**
Term: 1
Commit Index: 0
Log: []
Role: Leader
Term: 1
Commit Index: 0
Log: [★]

Role: Follower
Term: 1
Commit Index: 0
Log: []

Role: Follower
Term: 1
Commit Index: 0
Log: []
Leader sends log entries to followers
Majority of followers respond with success
Leader sends commit index to followers and responds to client
Example
Sad Log Entry
Role: Leader
Term: 1
Commit Index: 0
Log: []

Role: Follower
Term: 1
Commit Index: 0
Log: []

Role: Follower
Term: 1
Commit Index: 0
Log: []
Leader receives command

Role: Leader
Term: 1
Commit Index: 0
Log: [[★]]

Role: Follower
Term: 1
Commit Index: 0
Log: []
Leader sends log entries to followers

Role: Leader
Term: 1
Commit Index: 0
Log: [★]

Role: Follower
Term: 1
Commit Index: 0
Log: [★]

Role: Follower
Term: 1
Commit Index: 0
Log: []
Majority of followers do not respond

Role: **Leader**
Term: 1
Commit Index: 0
Log: [ ★ ]

Role: **Follower**
Term: 1
Commit Index: 0
Log: [★]

Role: **Follower**
Term: 1
Commit Index: 0
Log: []
Leader continues to retry log entry

Role: **Leader**
Term: 1
Commit Index: 0
Log: [★]

Role: **Follower**
Term: 1
Commit Index: 0
Log: [★]

Role: **Follower**
Term: 1
Commit Index: 0
Log: [★]
Example
Leader Failure
Followers do not receive heartbeat

Role: Leader
Term: 1
Commit Index: 0
Log: []

Role: Follower
Term: 1
Commit Index: 0
Log: []

Role: Follower
Term: 1
Commit Index: 0
Log: []

Role: Follower
Term: 1
Commit Index: 0
Log: []
First follower to timeout becomes candidate

Role: **Leader**
Term: 1
Commit Index: 0
Log: []

Role: **Candidate**
Term: 2
Commit Index: 0
Log: []

Role: **Follower**
Term: 1
Commit Index: 0
Log: []

Role: **Follower**
Term: 1
Commit Index: 0
Log: []
Candidate starts election and requests votes

Role: **Leader**
Term: 1
Commit Index: 0
Log: []

Role: **Candidate**
Term: 2
Commit Index: 0
Log: []

Role: **Follower**
Term: 2
Commit Index: 0
Log: []
Followers respond with votes
Extras

• Log safety and compaction
• Cluster changes
Real-life Application

- Distributed lock server
- Configuration management
- Background job storage
Smart People

- **Raft Paper** by Diego Ongaro and John Ousterhout
- **Raft Implementation**
- **ThinkDistributed**