Recitation 19: Raft II
Plan

* Recitation Qs
* Recap: Big picture
* Normal operation
* Operation under failure
* Scenarios

Logistics

* DB hands-on due tomorrow May 5
* Design project due May 11
* Rest of class is about security...
1. The authors of Raft were looking for an "understandable" consensus alg. What does this mean?

2. How understandable is Raft?

3. Why is understandability important? Is it?
The Big Picture

Goal: Implement a never-fail server using a cluster of sometimes-fail servers.
Normal operation in Raft

1. Client talks to leader.
2. Leader pushes update out to all servers.
3. When a majority of servers reply to leader:
   * leader applies change to "state machine"
   * leader replies to client.
Why a majority of servers?

Imagine a partitioned network

If leader waited for < ½ of nodes to store entry, then you could have a “split brain” situation

=> Two inconsistent DBs?

A big mess?

Imagine if this happened with CorioPass... very bad.
The only tricky detail:

What happens when the leader fails?

(Problem with all benevolent dictatorships.)

Raft does **NOT** protect against "Byzantine" failures.

Raft does **not** prevent against "Fail Stop" failures, as long as not "too many" nodes fail.

How many?
How does cluster recover from failure?

- Leader election.

* When follower doesn't hear from leader after a while → run election
* Node with most up-to-date log wins
  - Later "term" or longest log (if last term equal)

Important: Followers may delete log entries...

The only log entries that are committed are ones that LEADER has decided are committed

→ after hearing back from maj of nodes.
Example of how can't tell commitment from logs alone.

Servus keep extra state to track which entries the cluster has committed.

CRASH

See Fig 2 "Commit Index" "Last Applied"
When a new leader takes over...

→ Its log is authoritative.
→ Committed entries stay committed.
→ Non-leader logs may change.

Followers' logs may be inconsistent with leader's log. Leader takes precedence.

Rules to remember

* In election, server with most up-to-date log wins
* Terms increase with each election
* Need a majority of votes to win!
Example Scenarios (2019 Final)

One node crashes

- S1 crashes & restarts, S2, S3 crashes & restarts
- Network partition
- S3, S4, S5 start making progress
Example scenarios (2019 final)

* S1, S2 crash

* S3 is new leader, adds new entry, crashes

* S1 new leader, adds new entry to S2, crashes with S2

* S3 back up, new leader, replicates log to S4, S5
Back to the big picture

The illusion of talking to a never-fail server, but constructed from many sometimes-fail servers.

Very powerful idea?