Improving RAFT When There Are Failures

Christian Fluri, Darya Melnyk, Roger Wattenhofer
Majority decisions in Paxos are...
Majority decisions in Paxos are leader decisions in RAFT.
RAFT Protocol: four sub-problems

- Leader Election
- Terms
- Log Replication
- Consistency
Leader Election

Leader

Follower

discovers server with higher term

Candidate

discovers leader

> n/2 votes
Terms

- Time from a leader election until the next leader election takes place

- A node increases its term when
  - it times out
  - it receives a message with a higher term
Log Replication

Client → Servers

forward „execute c“

Leader
Log Replication

Client

Servers

Leader

append "execute c"

append "execute c"
Log Replication

“execute c” appended

Leader

“execute c” appended

Servers

Client
Log Replication

Client

Leader

commit c

Servers
Consistency

- followers only vote for candidates that are consistent with all their committed log entries

- only candidates with all committed log entries have a chance to win an election
BUILDING (A RAFT) WITH

...NO SPECS
...NO FUNDING

...NO QA
...NO MARKET RESEARCH

...NO DEADLINE
...NO DEMAND

MONKEYUSER.COM
We did (almost) all of this...

- We followed the instructions from Diego Ongaro and John Ousterhout, “In Search of an Understandable Consensus Algorithm”

- all server processes are independent threads and let them

- Communication runs via sockets

- For each socket listener we generated a new thread that constantly performs a blocking socket-read

- Implemented in Python 3.6, since it provides a threading library with a fair distributed scheduling in terms of CPU allocation

- ZeroMQ as library for asynchronous messaging
What about failures?
Link Failures
Link Failures: Policies

- send the RequestVote and the corresponding reply messages several times
- number of times a message is sent is equal to number of terms since the last leader was active
Link Failures: Evaluation

![Graph showing the relationship between terms for election and the probability of a link failure. The graph compares the scenarios with and without ReplicaVR.](image)
Isolation
Isolation: Policies

- Isolated server is a **leader**
  - **Commit Timeout**: timer for the leader when no more log entries have been committed within a certain time interval.

- Isolated server is a **candidate**
  - Each RequestVote has to contain the LastLeaderTerm
  - The server checks if its own LastLeaderTerm is higher
  - If this is true, the follower proceeds with the RequestVote as normal
Partition
Partition: Timeout Length

![Graph showing the relationship between terms for election and the number of servers cut off for different timeout lengths.](image-url)
Partition: Timeout Policies

- `increaseTimeoutLinear`: Increase the timeout linearly, the more split votes happen

- `increaseCandidateTimeout`: Adjust the timeout according to the ratio between positive and negative votes
Partition: Comparison
Conclusion

• Link failures, Isolation, Partition

• Additional timers

• Small number of simulated servers

• Different interval policies may become relevant
Thank You!