Assignment 2- Tobias Axelsson

911217-4116
tobaxe-2@student.ltu.se

1. How Raft works

Leader election

Initially every node starts out as a ”follower” node. Every node also has a election time-out procedure which will trigger an election in which a single leader will be elected. A leader is elected when it has the majority of the votes from the other participating nodes. Every election also has a time-out in case it takes too long, that is if a leader was not able to elected, a new election is held. Not reaching a majority might be due to failed nodes. If a leader “dies”, i.e. a server goes down, a new election is held.

Each node has its own election timer to when a new election is to start. The election timer could be incremented in the nodes from a leader that sends out information, meaning the leader is still “alive”.

Exchanging information

Information is stored in logs. A log entry on one server is a triple containing:

(index, term, data)

Where index is the “list index” of the log, term is the time-period the data was stored (each term is a unit of time separated by elections), and the data is the actual information that is to be stored.

A collection of logs among servers may look something like this:

	Index 0	Index 1	Index 2
Server 0 (leader)	Log entry 1	Log entry 2	Log entry 3
Server 1	Log entry 1	Log entry 2	
Server 2	Log entry 1	Log entry 2	

The colored area represents the committed entries, since Log entry 1 and Log entry 2 are represented among a majority of the active servers, they are committed.

Server 0 might commit Log entry 3 by sending a message to the other active nodes that Log entry 3 should be at index 2 and it should be come after Log entry 2 at index 1, this to ensure that the logs are replicated among the servers.

Electing “the right” leader

Raft works to make sure that the leader is a node with as many log entries as possible. If an election would be held by Server 2 in the state shown above, Server 0 would not vote for Server 2 as Server 0 has more entries.

2. Compared with Zookeeper

- Raft spends more resources in having “the right leader” node through elections and that the leader should have many log entries, whereas in Zookeeper, the leader is selected from a given “queue” of who should become the leader.
- Information is updated in raft through the leader replicating entries. If a z-node in Zookeeper is updated, all of its underlying z-nodes (data) is updated.
- Raft is more simple than Zookeeper.

3. Major takeaways

- I’ve learned about some of the functionality and importance of coordination services in distributed systems.
- It’s hard to make sure that all of the nodes in a network agree on the same information.
- Studying Paxos, Zookeeper and Raft I have learned that it’s good to think through the “worst-case scenarios” in which nodes are failing unexpectedly etc and this is part of the reason that distributed systems are complex.