Multi-Decree Paxos vs Raft

Miguel Alves, Nº 49828

1 Introduction

As opposed to **Single-Decree Paxos**, **Multi-Decree Paxos** protocol is used in the context of log-replication, ensuring consensus on the order in which the log entries will be applied.

The goal of the Rust implementation of **Multi-Decree Paxos** was to reuse as much code as possible from the implementation of the **Raft** protocol.

There are multiple definitions of **Multi-Decree Paxos**, but the description of the protocol found in "Paxos vs Raft: Have we reached consensus on distributed consensus?"[1] was chosen as a guideline for the implementation, since it is presented using the same nomenclature used in the original **Raft** paper, making it easier to compare the algorithms and find reusable code.

2 Differences Between the Protocols

The following summarizes the differences between the protocols, as highlighted in the paper[1]:

1. How does the protocol ensure that each term has at most one leader?
   (a) `Raft` → a follower can become a candidate in any term. Each follower will only vote for a single candidate per term, therefore only one candidate will get a majority of votes and become the leader.
   (b) `Multi-Decree Paxos` → a peer **p** can only be a candidate in a term **t** if **t % n = p**. This means there will only be one candidate per term, therefore there will can only be one leader per term.
2. How does the protocol ensure that a new leader's log contains all committed log entries?
   (a) `Raft` → a vote is granted only if the candidate's log is at least as up-to-date as the follower's log.
   (b) `Multi-Decree Paxos` → each VoteResponse includes the log entries of the follower. Upon receiving a majority of votes, the candidate will merge the entries with the highest term to its own log.
3. How does the protocol ensure that leader's safely commit log entries from previous terms?
   (a) `Raft` → the leader replicates the log entires to the other servers without changing them, but they will only be committed when the leader commits one entry from his current term.
   (b) `Multi-Decree Paxos` → uncommitted log entries from previous terms are added to the leader's log with the leader's term. The leader then replicates these entries as if they were from his term.

3 Code Reused in the Rust Implementation

The code for the implementations can be found here.

Originally the Raft implementation had 660 lines of code.

About 400 lines of code were reusable, and the Raft implementation ended up with 280 non-reusable lines of code, and the Multi-Decree Paxos implementation with 240 non-reusable lines of code.

These non-reusable lines reflect the portion of code that corresponds to leader election and to committing log entries.

4 Sources

[1] *Paxos vs Raft: Have we reached consensus on distributed consensus?*, Heidi Howard & Richard Mortier, https://arxiv.org/abs/2004.05074