Implementing Raft in Python

Xuangui Huang *and David Stalfa

June 8, 2022

Title. Implementing Raft in Python

Team. Xuangui Huang and David Stalfa

Goals and Objectives. The primary objective of our project is to implement and better understand the Raft consensus algorithm. Our first objective will be to implement the basic functionality of Raft, consisting in the leader election and log replication protocols. Once this is complete, we will turn to a second objective of implementing membership changes, which Raft implements as a joint consensus protocol. We prioritize the membership change protocol because, along with leader election, it seems like one of the more distinctive elements of the Raft algorithm. We believe that implementing this aspect of Raft will provide a different perspective on consensus algorithms than we have seen with Paxos and other algorithms in class.

Proposed Approach. We describe our approach from several angles: programming language, server environment, underlying state machine, and testing. We have chosen to use Python 3x as our programming language due to the simplicity of the code. Since we are emphasizing understanding over performance, we are not over concerned with the run time of our code and so a higher level language like Python is more suited to our purpose. For our server environment we will use Docker, since this is the environment we have used so far in class. Our underlying state machine will be a replicated key-value store. The client will request that a certain key-value pair be put into the store and the servers will come to a consensus on what the store contains, where each replica contains a copy of the same store. The client will have two types of commands to the serves: a $\mathsf{put}(key, value)$ command which will request a store and a $\mathsf{get}(key)$ command which will request the integer $value$ associated with string key. Our testing will use this underlying state machine to retrieve values by key. We will test the `get` and `put` functions before/after successfully placing key-value pairs in the store, before/after leader elections, and before/after membership changes. This will provide evidence that the each of the underlying Raft protocols are being implemented correctly.

Individual Tasks. We divide the initial tasks as follows. The log replication protocol (and the underlying key-value store) as well as the client side protocols will be implemented by David. The leader election protocol and networking functionality will be implemented by Xuangui. The membership change protocol will be handled after these are complete and will be worked on together.

*Email: