Design Exercise 2
Engineering Notebook

Adarsh Hiremath and Andrew Palacci

March 8, 2023

I. Introduction

In our design exercise, we implemented a model of a small, asynchronous distributed system. In this system, each of the three simulated machines run at different clock rates assigned randomly at initialization. Each machine listens on a socket for incoming messages from other machines and updates the corresponding logical clock and message queue for each machine. In this engineering notebook, we intend to describe our design decisions and the results from the following experiments:

- 5 iterations of the scale model for exactly one minute.
- A single run of the scale model for two minutes.
- Modifying the the probabilities that machines send messages to other machines.

The source code for our chat application can be found here.

II. Model Distributed System Design

Building our scale model for the simulated machines required implementing a protocol for the three machines to communicate. Otherwise, it would be impossible to receive logical clock information, send messages, or update the message queue. Thus, the first question Andrew and I sought to answer in the design process was what protocol we wanted to implement for our scale model.

Initially, Andrew and I were inclined to implement our communication protocol using a simple peer-to-peer (P2P) networking library such as Python's P2P. The idea of P2P networking was appealing since there would be no single point of failure; each machine could act as both a client and server, allowing for a more dynamic and flexible system. Additionally, with P2P, we thought that we could cut the number of connections in half because every connection would be bidirectional.

However, we decided to use web sockets instead for a couple of reasons:

- By using web sockets, we could partially reuse implementations from the first design project instead of learning P2P networking for the first time for this project.
- The problem specification recommended that each machine in the simulation send and receive information using web sockets.
- Completely decentralizing communication is not realistic, especially at scale. It is far easier to accomodate more machines in a distributed system without dramatically revamping the code base when every other machine can ping a server without much additional information.

Instead of using a P2P system, we decided to implement a server and a client on each machine. In our code, the server exclusively listens for messages from other machines and the client sends messages to other machines in the model. Since we implemented the previous design project in Python, we did the same for this design project as well.