Final Project Suggestions

The Course Staff

1 Introduction

This document lists a few suggestions for final projects. Many of the suggestions are open-ended, and allow you to go in several directions. All of them require writing code: while we suggest doing so in Elixir (since it allows you to reuse some of the emulation setup and the familiarity you have hopefully developed), you don't necessarily have to use Elixir. In either case you must show that whatever you do faithfully uses or emulates isolated processes, i.e., we are looking for distributed computing not concurrency. The projects are all at different levels of complexity.

A consequence of the projects being open ended is that you might find it hard to actually complete everything you propose. While we would prefer complete projects, if it turns out that an unforeseen technical challenge prevents completion then that is fine: in your final write-up you should carefully describe the challenge and explain how you think one should approach the problem of solving it. In this case, we will grade based on both how much effort you put in, and on your description of the challenge and why that prevented progress. We of course recommend discussing concerns with the course staff during office hours.

2 Using this document

Read through the proposals below and select one that catches your fancy. Each of the proposals leaves several unanswered questions. Additionally, all of them leave unspecified:

- Implementation group and division of labor: Who do you plan to work with? How do you plan to split your work? **Note** You are **strongly** encouraged to work in groups of two. If you are picking one of these projects and **not** working in a group you must get explicit permission from Panda.

- Implementation strategy: how do you plan to implement the project? What environment are you planning on using for your implementation.
- Testing strategy: how do you plan to test your project?

In your project proposal (due on October 14th), you must answer the above three questions and any questions unique to the proposal you pick.

3 Proposal 1: Implement Practical Byzantine Fault Tolerance

Practical Byzantine Fault Tolerance (PBFT) is a paper by Castro and Liskov that we are going to read as a class on November 4. This paper considers the question of how to reach agreement in an environment where nodes can exhibit Byzantine failures, i.e., act in arbitrary ways. The paper we read builds on digital signatures (a common cryptographic primitive). An accompanying Tech Report showed how one can use HMACs, a faster primitive for the same purpose. Later papers, for instance SBFT have looked at how to achieve additional scalability.

For this class you can stick to digital signatures, but you should choose and use a reasonable signature scheme for the moment. You might find Cloak and ExCrypto useful in doing this. You are however encouraged to try one of the other approaches if you desire.

Testing is going to be one of the main challenges in this project: you must demonstrate that your implementation does in fact behave correctly in the presence of Byzantine nodes, and also in their absence. To test this you could use various mechanisms, including modifying and adding to the the fuzzers provided by the classes emulation layer.

3.1 Questions for this proposal

- What cryptographic primitive are you going to use?
- What application are you going to use to build on top of the PBFT protocol? You should motivate the desire for Byzantine fault tolerance in this case.
- While included in the common list, this is a reminder that testing is particularly tricky in this case, and you must discuss how you plan to test your work.

4 Proposal 2: Implement a system that detects bugs in distributed systems

We talked about Vector Clocks in Lecture 2, and about how they are now widely used in order to record and analyze the behavior of distributed system. Pip by Reynolds et al was an early system that presented a system that showed how this can be done in practice. In this project you will implement something similar to PIP, that is you will implement a system that accepts a set of expectations about how programs should behave, and then monitor them to determine violations. In adopting Pip, you do not need to consider threads or futures, and figuring out what is in scope is an important question that your proposal needs to address.

In building this proposal you are encouraged (though not **required**) to use the vector clock implementation you worked on for Lab 2. You **may not** use existing distributed logging infrastructure.

4.1 Questions for this proposal

- What types of programs do you plan to consider? Specifically what limits are you imposing on the user of your system.
- What types of expectations do you plan to support? How are they specified? We strongly recommend choosing a representation that makes everything else easier.
- What applications and expectations do you plan to start with? Having an initial set is probably helpful in getting you started.

5 Proposal 3: Implement a weakly consistent key-value store and measure the impact of weak consistency

In lecture 4 we discuss linearizability and the CAP theorem, and end with the observation that many weaker forms of consistency are now used in practice. One example is Dynamo a key-value store from Amazon, though others exist. This project requires that you do two things:

- First, implement a version of one of these data stores. It is perfectly fine to implement Dynamo, but you can implement any of them. If

you are implementing Dynamo you must implement the anti-entropy mechanism and gossip portions.

- Second, analyze the behavior of your implementation, specifically quantifying how often you read stale data, or see inconsistencies. Specifically, your measurement must consider the effect of different failure rates, message latencies and message drop rates. You might find the techniques and ideas presented in PBS useful in doing this.

5.1 Questions for this proposal

- What key-value store do you plan to implement and analyze?
- What metrics do you plan to measure in your analysis?
- What is your hypothesis about what your measurements will show?
- For the testing strategy: how do you plan to validate your results?