TCSS 558 HW5 Design Document

Daniel McDonald, Jesse Carrigan and Sven Berger

December 8, 2013

Introduction

This project is meant to fulfill the basic requirements of the Chord protocol as outlined in the third and fourth homework literature. This document will explain the basic design of the entire project, and detailed design of those components necessary for this portion of the assignment.

Design Overview

Current Requirements

In addition to the requirements outlined in HW 4, in this assignment our chord protocol implementation must be extended to:

- Allow nodes to recover when one other node unexpectedly leaves the network.
- Any data stored on the leaving node must be recoverable within the network.

The architectural changes made in HW4 have proven adequate, and a major rewrite such as the one required at that time are unnecessary for this assignment. Additions had to be made to accomodate data redundancy, but nodes do a good job detecting neighbor/finger failure and fixing themselves with the method employed in HW 4. To facilitate data redundancy, we have chosen to make each node store the data of it's predecessor. It is important to note that our solution is only guaranteed to work when only one node fails at a time. If two neighbor nodes fail, data will be lost.

Network Recovery

When a node is dropped from the network, it is the job of several periodic tasks to detect, and repair the network. The stabilize function resets the predecessor and successor, if they are found to be missing. FixFinger will take a finger at random and verify it exists.

Data Recovery

forwardValuesForBackup - push our data to successor, and the backup values to predecessor.

The forwardValuesForBackup function serves to push our locally owned values to our successor for backup, ands sends the values we have stored to our predecessor. By doing this it populates the values a new node needs, or a node taking over the role of a lost member.

Limitations

Fresh data can be replaced by backup data in some uncommon circumstances. If a node sets data to it's range, and then the successor backs up what it believes to be the predecessor's correct data, the newly set data will be overwritten. In our tests this never occurred, but it is theoretically possible.

As with the assignments leading up to this, multiple node failures are not tolerated.