Santiago Roig
Milestone 2 Report

Introduction

In milestone 2 the objective was to create a P2P Gossip Server. This server had to accept new peers, send gossip messages, and respond to requests for a list of peers. In order to accomplish this task I wrote all of my code in Java. I found that leveraging Java's object oriented approach made each of these tasks more manageable.

Technologies Used

In order to create this project I used a few technologies. The code itself was written in Java. I also leveraged a few libraries to help handle the server responsibilities. For databases I used SQLite JDBC and for handling command line options I used Apache Commons CLI. I wrote most of the code in the Eclipse IDE, but also used Visual Studio Code at some points. In order to compile, run, and test my code I wrote bash scripts.

Architecture

The architecture has a flow down as following: start server → kick off UDP and TCP iterative threads → receive message on thread → identify message using Message class → handle the message appropriately using the MessageHandler class → wait for new message. The MessageHandler class took care of much of the heavy lifting. The database is set up as a singleton so it can be accessed universally throughout the program. I initially packaged all of the java classes into packages, but ran into issues creating a script to run all the packages. Normally I would have used ant or maven, but since we were constrained to make and .sh scripts I just removed the packages and put all source in the default package.

User Manual

In order to run GOSSIP Server use the following commands:

./run.sh [port number]
nc localhost [port number]

In nc type one of the following three example commands:

-GOSSIP:mBHL7IKilvdcOFKR03ASvBNX//ypQkTRUvilYmB1/OY=:2017-01-09-16-18-20-001Z:Tom eats Jerry% ← Sends gossip message to server in format.

-"PEER:John:PORT=2356:IP=163.118.239.68% ← Tells server about new peer.

-PEERS?\n ← Request list of known peers from the server.

Conclusion

This milestone took a lot of time and planning. Debugging network programs can be frustrating especially when multithreading is involved. The code base is clean and does a good job of leveraging Java's OOP. The behavior and description of each of my functions, input, and parameters can be found inside of DOCUMENTATION.pdf, which is a javadoc for my source code.

References

https://commons.apache.org/proper/commons-cli/
https://www.sqlite.org/
https://www.java.com/en/
http://code.visualstudio.com/
https://eclipse.org/