Santiago Roig
Xiang Li
Milestone 3 Report

Introduction

In milestone 3 the objectives were to create a concurrent server, create a client to connects to the server from milestone 2. This client must send sha 256 gossip messages and new peer messages to the server. The server must synchronize each thread so that database access is blocked when threads attempt to access it at the same time.

Technologies Used

Code: Java with a few libraries to help handle the server responsibilities
Database: SQLite JDBC
Command line options: Apache Commons CLI.
Client: Android
Bash Script: Compile, run, and test
Development tools: Eclipse IDE, Visual Studio Code, Android Studio

Architecture

Server:
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.

Client:
The client is an Android app that has the following flow down:
Start client → Enter server IP Address and Port → Select TCP or UDP connection type → Send GOSSIP or PEER message → enter corresponding message → send → validate server response. The client is fairly straight forward architecture. The main activity handles the connection information to the server. There are two other activities, one for sending a GOSSIP message and another for sending a PEER message.

User Manual

SERVER:

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.

CLIENT:

- Start app
- Enter IP Address and port number
- Select UDP or TCP
- Select GOSSIP or PEER message
- Enter in required fields
- Click send button in bottom right.

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 our 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/