Project Evaluation Report

TauNet System v. 1.0

RJ Russell

08 December 2015

This project is composed of five major components: RC4 encryption, server, client, client interface, and the address book.

I started this project by investigating what a simple client/server pairing looks like. I also bounced ideas off of three of my peers, Rachael Johnson, Jacob Martin and Andy Wood to gain a better understanding of how sockets work and what the RC4 encryption should look like. Using the python docs, the BYU website and peers, I constructed a simple design and implemented it. I first implemented a simple server, and then once that was tested I implemented the simple client. I tested each piece thoroughly by making each file its own main and running individual unit tests on each file. I chose to keep the server as a standalone program so that I did not have to manage an interface that accommodated both the server and client. I kept both the server and client as simple as possible and only added in the necessary try/catch blocks as well as the calls to encrypt/decrypt messages.

I then tackled the task of implementing the RC4 encryption. By following the algorithm online, and looking up some python syntax to make sure my knowledge of how to implement the algorithm using python was correct, I was able to piece together the encrypt/decrypt methods. I tested this first with hard coded messages, and encrypting then decrypting and displaying to ensure I got the same message back. I also used the test files to make sure the decryption worked, and finally tested it with other users.

The client interface is where all of the logic lives for the client side of the messaging program. I started by reading in the address book, displaying it and then moved on to being able to choose recipients from the list and have the correct address returned. I added features such as a help menu and a display address book, and also an exit keyword to quit the program. This piece of the program is the hub for the client side, and I ensured that all the loops, data members, and logic was running smoothly.

When I was finished with coding and unit testing, I tested the program numerous times by just simply using it over and over. I was able to successfully send messages to 19 people in class, and received replies back from them. A few of those 19 other users are people that I have sent numerous messages back and forth to. I also used the echo server to validate the header formatting and passed the echo server test. Overall I think the current implementation works quite well. It is simple to use and does what it is supposed to.

I learned quite a bit about networking, sockets, and the encryption algorithm during this project. I learned enough to know that dealing with networking issues can be tricky as the network can be finicky. I also learned enough to know that debugging random pieces of raw data can be frustrating. The hardest part of this project has been the documentation. Ensuring that the documents are accurate and organized is quite a bit of work.

There are a few features in the current version of this project that I am going to look into adding. The project currently does not support threading and a unified client interface for the server and client. I had to learn sockets from the get go and it took a lot of reading to understand what the small amount of code that runs the server and client are actually doing under the hood. Threading seemed to be a bit complicated for me at this time, but I intend on researching various

approaches and implementing a multi-threaded system at some point. By implementing threading I can unify the client and server into one interface, so that each piece is running simultaneously in the background on separate threads. I think this would streamline the project and give the interface a much better appearance. I would also like to find a method for creating containers for each piece much like IRC does. Where the users online are shown on the screen in a separate buffer, and private messages to each user can be accomplished with a key binding. Once I build that, I can incorporate a buffer where the messages sent in are displayed and a buffer where input is taken so that the whole program can run at once in parallel.

Overall, this was a good program to learn the basics of encryption/decryption, networking and building a simple project. It was also a good basis for understanding documentation, copyrights, and other various documents that need to be included in any project.