Project Evaluation v1.0

This project was extremely interesting and fun to get working. I decided to use this opportunity to learn python. It was an interesting challenge, but what I think is the most important thing I learned during this process was that although languages have different nuances, as far as syntax and operations are concerned, they are all pretty similar and learning new ones over a couple of days isn't as daunting as it seems.

It was also nice putting the software design process to use. Although I wish we would have started earlier and had some of the documents required before coding. I followed the general process but it was mostly in my head. I knew how I wanted to break the project up in to smaller pieces, and the order in which to get them working. Also I would have made an effort to save my pseudo code, instead of pseudo coding in my file with comments and then deleting the comments before submit. In the future I plan to create pseudo files for each subsystem, where I will store the pseudo code for all functions that elicit pseudo code.

In my project I feel like I was using Agile more than Waterfall methodologies. Where once I was done with a system I would find myself going back and making improvements to systems that I had previously 'finished'. The only system I didn't really touch once I got it working was the cipher saber encryption. The hardest part of that system was learning how python handles for loops and how python handles characters differently then C++.

Overall I think the whole project worked out very well and I am left with a product that I am very pleased with and excited to show my future employers. I do wish that I could have started working on it earlier in the term however. Who knows where I would have been able to take the project.

If I were to do this project again I would take more care in writing down and following a development plan. There were some cases when I got the bug to work on a particular piece of the project earlier than I should have. This resulted in me needing to recode some functions that I otherwise would not have needed to.

I was also not able to fully test the ability of my program to message across the internet. I live behind the Campus firewall and it blocks incoming connections. I contacted OIT, but they had no solutions for me. Instead I installed Linux on another system of mine and used that to send and receive messages with my Pi over my LAN. This worked well, but connecting to the echo server was impossible, and once it was revealed that the echo server was even going to exist it was too late in the term to again reach out the OIT for solutions to this problem.

Things that I would still like to do on this project are add the following additional commands and features:

#user_name	Display messages from user_name
~user_name	Display user_name's info. This would remove the need to display a user's IP address when displaying all users.
Onion Routing	This is much easier than I think the class realized, but would require a new protocol format and shorter message lengths.
Better interface	I would have liked to have an interface that was split in to two windows where the messages would be displayed above and the user inputs below (much like how modern text messaging programs work).
Update user request	I would have liked to add the ability for the user to requret IP address updates to all other nodes in the network, or request that another user add them to their user

	list. This could open up an easy way to preform man in the middle attacks however.
Group Mesasging @user1 user2...	I would have also like to crate a way to send a message to multiple users at the same times, all users or a specified group of users.
Sent Messages in Message History	This would be easy to implement, but I just ran out of time, and needed to call it quits. If I would have needed another week to finish all the features that have been floating around in my head.
Online users	With the most recent update to the protocol it is now possible to check and see if users are online by sending them empty message. These same empty messages could be used to to determine when another user comes online, or disconnects, once TauNet is already running.