Andrew Beshay, z5019760
COMP3331

Assignment for Term 1, 2019

For this project I completed it in C and have included a makefile. To make the program simply type “make” into the terminal and run the program from the script provided.

Demo: https://youtu.be/_Se2uGb9YmQ

First Step:

The first design choice I had would be storing all the info to do with the peer in a global struct. It is not the safest choice unless I could use a const for the ID, but I couldn’t manage to figure out getting arguments from command line into the global space in preprocessor method. I used pthread library for the multithreading and this also required me to keep my steps in chunks which offered cleaner code. I made 2 functions (also separate threads for each function) for a UDP server and a pinging client. The pinging client would maintain pinging at a certain interval while the server would be constantly listening for requests and responses and handle them accordingly. I used ANSI escape codes to colour code messages as it made for easier visualization for responses and requests. The interval I chose between pings was honestly just a random guess, something that’s not too fast or not too slow. I also decided to implement the input function here as well to constantly be listening the stdin for command requests and such. I left it pretty bareboned originally, just taking in commands and printing them to make sure they were working.

Second Step:

I then when next onto the gracefully quitting a peer step and decided to leave requesting a file last. I felt as if I had implementing requesting a file before the quitting and detecting a lost peer, I would introduce more bugs implementing them, instead of dealing with all the peers perfectly beforehand. For peer departure, at this point when reading on the functionality of what was required, I decided to increase my struct to also keep track of peer predecessors as well as successors. I then had to find a method of quickly detecting new previous peers after detection, so I had to vary my ping request time based on that, as a one much slower could lead to invalid predecessors being displayed. I also implemented a TCP server like the UDP one in step one, for maintaining all TCP requests but for the messaging predecessors I made another function specifically for that command called notify departure. I had a bug originally which accidently dealt with peers departing like peer 3, mistaking identifying peer predecessor orders. When 3 would quit, peer 4 would make 15 its first pred and 1 second instead of other way around.

Third step:

For implementing the file transfer, I started off one step at a time, first was getting the correct hash and location of the peers and just getting the right response messages across

the networks. I again extended my struct for the peer info to include file information and other such. Next was beginning to do the file transfer. I started with just simply transferring the file across with no drops or errors. This worked fine and perfectly. I had trouble implementing the stop and wait method, but finally got it working after about 6 hours. It was basic things or errors that were not noticeable until you rest for a bit and come back that slowed down a lot of my progress.

Improvements

Improvements I would make are including some of the functions in separate c files to really keep the code clean, cause right now its in one giant CDHT. Other improvements are in the way which things are implement, are sort of messy as it was implemented in a linear method I didn't return to things. I'm sure if I could rewrite the program there would be parts that would be shorter and maybe not as complicated when coming back with a clearer head. Also I might have implemented the detect a peer death (not quit) better using just timers instead of a ping tracker like specified in the assignment.