Reliable File Transfer Assignment # 2

By Jose Eduardo Soto SID#...5673

Introduction

This report is dedicated to informing the reader on the strategy taken to solve the assignment. It will contain evidence, samples, and usage.

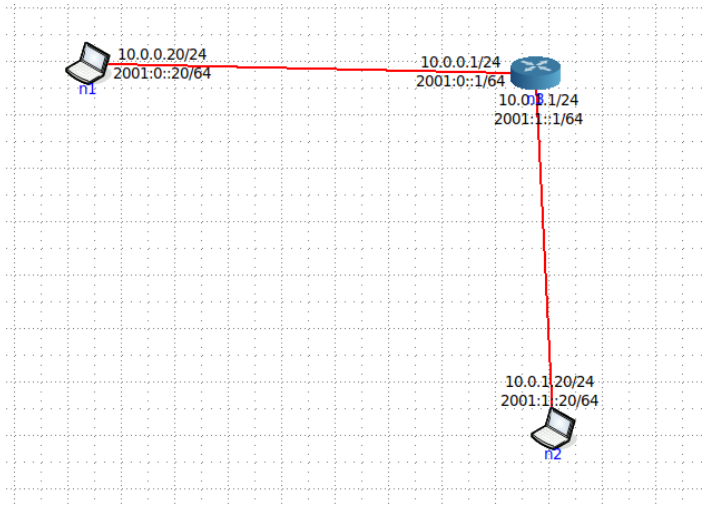
Report

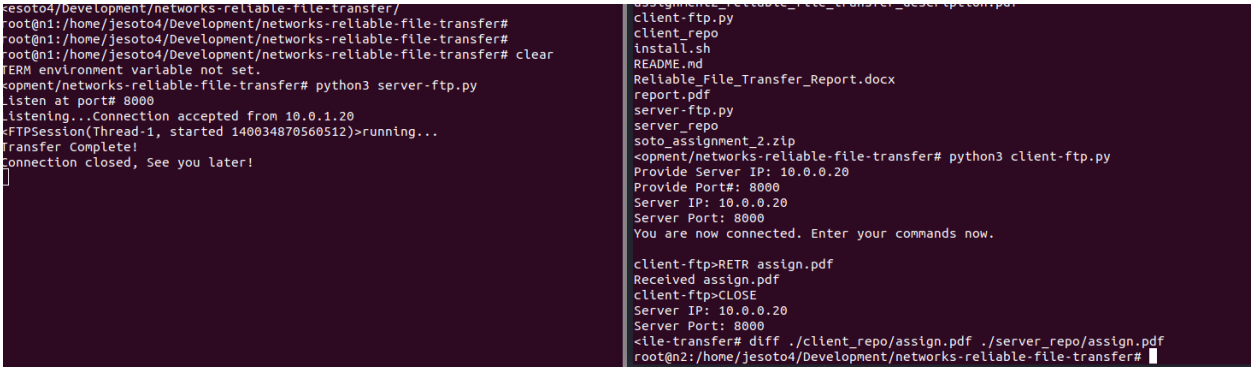
The product of the assignment was a success! The software artifacts produced is able to successfully transfer a file of any type. As of now, the program may transfer up to 9,999,999,999 bytes of data. That's just under 10GB of data.

The strategy was to first do the client. A simple class is constructed to organize the session. It continuously prompts the user for IP and Port and uses regex to validate the input. Next it connects to the server and has two commands: RETR <file name> and CLOSE. Clearly the RETR command retrieves the file from the server if it exists and CLOSE ends the session with the server and closes the client. The server accepts multiple concurrent connections and has a class to encapsulate the sessions like my previous project. The server listens for connections then creates an FTPSession object when a connection comes in. It waits for input and upon receiving it, then responds correspondingly to the client. All data transfers are done by reading and writing bytes. This mechanism settles the differences between different file types.

Perhaps the most challenging aspect of this product was developing an algorithm that enabled required behavior. In particular, the client knowing when the transmission of data was completed. This was the main concern. The reason why was because of the assumption of the reliable connection. The TCP protocol allowed the developer to now worry about corruption, out of order transmission, or dropped packets. To accomplish the end-of-transmission behavior was simple (after lots of failure). First the server sends to the client what the size of the file is in bytes. The client stores the size of the file. The server will then send over the data of the file. As it writes the data to the client-side, the client program tracks the number of bytes it has received. Now one or both conditions might be met that captures the end-of-transmission behavior: The number of bytes received is equal to the stored size of the file or the received data was smaller than the 1000 byte chunk expected. This event triggers the program to close the file and that session.

Deploying the service on CORE nodes raised an issue. The design of the server doesn't require to specify the IP but if the program is deployed to a CORE node, then the program must specify the IP it was assigned. Once that issue is resolved then the program works in CORE.





The usage is simple. Start the server first with “python3 server-ftp.py” you will be prompted. Input a valid port number and it will wait for connections. Next the client will run with “python3 client-ftp.py” and will prompt for the IP address and port. You will receive a prompt where you may put commands. Easy as that.

References

[1] Regular Expressions Cookbook, Chapter 7, section 16. O’Reilly Media. Inc https://learning.oreilly.com/library/view/regular-expressions-cookbook/9780596802837/ch07s16.html#uripath-ip4-problem