Phase 3 Readme Document

Title and Authors

J.M. Gallagher, James Whitney, Boban Pallathucharry
March-18-2020

Environment

OS Used: Windows 10 or Ubuntu Linux
Programming Language: Python 3

Submitted Files
Phase3Readme.pdf
Phase3DesignDocument.pdf
RDTClient_Phase3.py, implements the client portion of the program
RDTServer_Phase3.py, implements the server portion of the program
image1.bmp, sample image
PerformancePlot.PDF, shows the transmit time of the program in different scenarios

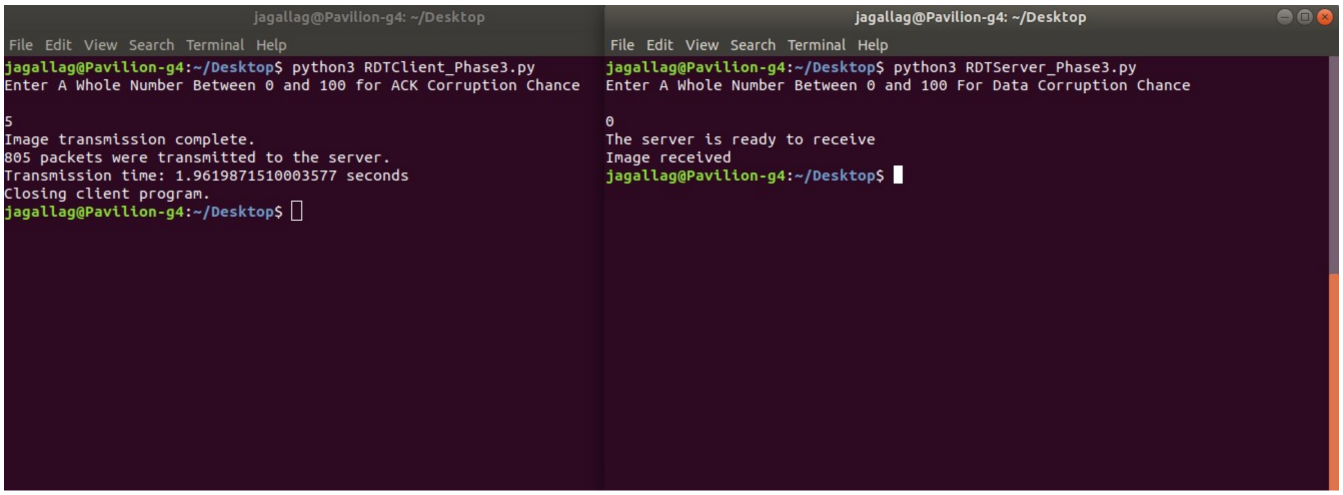
Instructions

1. The code is meant to be run from Windows 10 or Ubuntu Linux. Ubuntu Linux already has Python 3 installed and ready to be called from the terminal.
2. Folders can be stored in the location of choice of the user so long as the terminal directory is set to that location. The design document shows how to run the files.
3. **Important:** The code must directly reference the file to be transferred. The submitted code references the "image1.bmp" test file for demonstration purposes. There is no prompt for user input to change the test file. It must be changed directly in the source code.
4. The code requires use of the Python module "bitstring" that needs to be installed with the **pip3** installer. For a Linux machine, pip3 can be installed with the the first two commands showing here. The third command installs the bitstring module.

   ```
   sudo apt update
   sudo apt install python3-pip
   sudo pip3 install bitstring
   ```

   On a Windows machine the procedure is different but can easily be found on the internet.
5. The server file must be run first in order for the code to work. The server file runs idle until it is contacted by a client. The server will prompt the user for input to determine the level of corruption introduced into the data packets it receives from the client.

6. Once the server code is running, open another terminal and run the client code file. The client code prompts the user to determine the level of corruption introduced into the ACK packets it receives from the server.
7. **Option 1, No Corruption** may be executed by entering “0” in both the client and server prompts for corruption.
8. **Option 2, ACK bit error** may be executed by entering a non-zero prompt in the client terminal and entering a zero in the server terminal.
9. **Option 3, Data packet error** may be executed by entering a “0” in the client terminal and a non-zero value at the prompt in the server terminal.
10. Be aware that the program can recover from errors when non-zero values are entered in both the client and the server, producing a fourth option for corruption configuration.
11. Figure 1 shows an example case of running Option 2. The server code run on the right hand side is set to 0% corruption chance and the client code is set to 5% corruption chance.


jagallag@Pavilion-g4: ~/Desktop
File Edit View Search Terminal Help
jagallag@Pavilion-g4: ~/Desktop
File Edit View Search Terminal Help

```
jagallag@Pavilion-g4:~/Desktop$ python3 RDTClient_Phase3.py
Enter A Whole Number Between 0 and 100 for ACK Corruption Chance

5
Image transmission complete.
805 packets were transmitted to the server.
Transmission time: 1.9619871510003577 seconds
Closing client program.
jagallag@Pavilion-g4:~/Desktop$
```

```
jagallag@Pavilion-g4:~/Desktop$ python3 RDTServer_Phase3.py
Enter A Whole Number Between 0 and 100 For Data Corruption Chance

0
The server is ready to receive
Image received
jagallag@Pavilion-g4:~/Desktop$
```

Figure 1: Running Option 2 from Linux terminals