Joel Aguirre
Josue Soto
Computer Networks

Programming Assignment 2b

1. Problem Strategy

a. Go Back N

Taking the segmentation of the data into packets and the client/server setup from assignment 2a, what was left was to write the Go Back N implementation for unreliable networks. The animation that was listed on the bottom of the assignment sheet was extremely helpful and almost entirely was the source of all of the implementation. Now to the implementation:

In the server/sender:
First we have to define our window, its start and end point as well as the expectedAck. All of these variables are iterated on successful packet sends. Next, going over server setup, we start our timer with a time set of 5 seconds, and send our packets (after making them with the packets module) within our initial window. Once we do, we listen for an ack from the client that aligns with our expected ack. In that process, if we don't receive an ack, a timeout exception is raised and the window is resent and the timer is reset. However, if the packet is acknowledged we check it's validity (inside our window), if it is, then we iterate the variables stated before and send the next window's packets that have not already been sent. If not, then we ignore the ack and resend the window's packets that have not been already sent. Looping this we will receive all acknowledgments and mark the file transfer as complete!

In the client/receiver:
Only a few changes had to be made, which was to first extract the packet into a sequence number and the actual data. Then we check if that sequence num was as expected and if it is, then we write the data into the file and send an acknowledgment number equal to the received sequence number.

b. Stop and Wait

In the server/sender:

As in the GBN, there is a set of variables to track the packets that is updated on a successful packet send. Once an ACK is received, the server sends the next packet. If a timeout occurs, the packet is resent. If sending an ACK or EOF, the server sends the packet without expecting an ACK back, as this would lead to a loop of confirming ACKs.

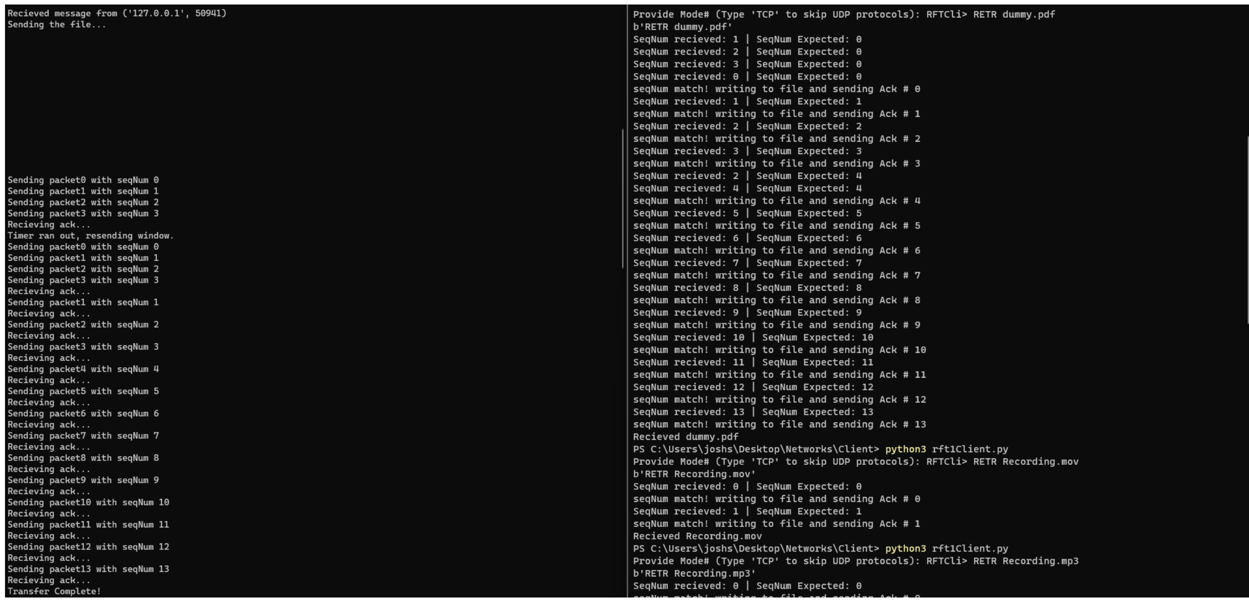
In the client/receiver:

The client operates on a similar logic, constantly receiving packets and then sending ACKs for said packets. An issue that I ran into was the server sending duplicate packets sent due to server timeout, which were then being written to the file, making it innacurate. I ended up resolving this by having the client temporarily save packet data, then whenever it received a new packet, it would compare it to the previous packet's data to check for duplicates. If it was a duplicate, it simply would not write the data to the file, but would still send an ACK in case the server was losing the packet during transit. However this clearly does introduce an error in the case that multiple copies of the exact same data are sent in succession.

2. Execution Samples

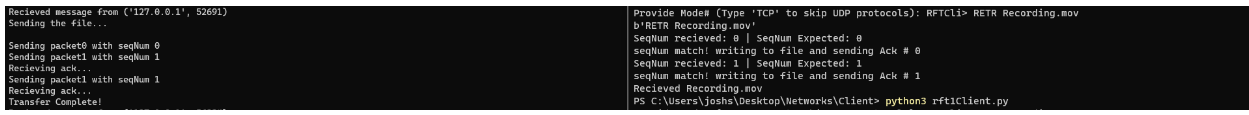
a. Go Back N

(Port #, Host and Mode inputs are pre-defined for ease of execution in execution samples)



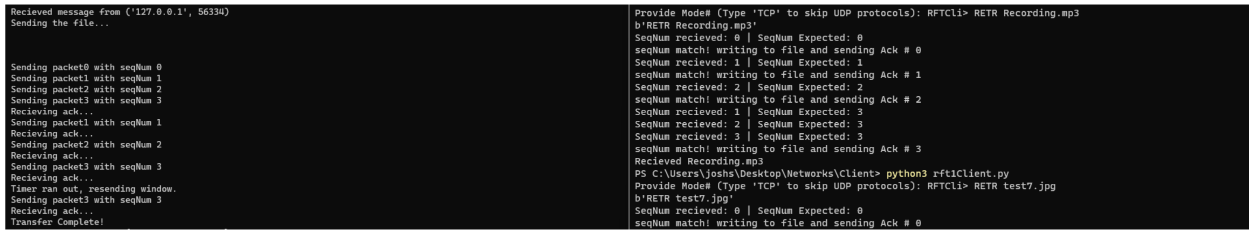
```
Recieved message from ('127.0.0.1', 50941)
Sending the file...









Sending packet0 with seqNum 0
Sending packet1 with seqNum 1
Sending packet2 with seqNum 2
Sending packet3 with seqNum 3
Recieving ack...
Timer ran out, resending window.
Sending packet0 with seqNum 0
Sending packet1 with seqNum 1
Sending packet2 with seqNum 2
Sending packet3 with seqNum 3
Recieving ack...
Sending packet1 with seqNum 1
Recieving ack...
Sending packet2 with seqNum 2
Recieving ack...
Sending packet3 with seqNum 3
Recieving ack...
Sending packet4 with seqNum 4
Recieving ack...
Sending packet5 with seqNum 5
Recieving ack...
Sending packet6 with seqNum 6
Recieving ack...
Sending packet7 with seqNum 7
Recieving ack...
Sending packet8 with seqNum 8
Recieving ack...
Sending packet9 with seqNum 9
Recieving ack...
Sending packet10 with seqNum 10
Recieving ack...
Sending packet11 with seqNum 11
Recieving ack...
Sending packet12 with seqNum 12
Recieving ack...
Sending packet13 with seqNum 13
Recieving ack...
Transfer Complete!
```

```
Provide Mode# (Type 'TCP' to skip UDP protocols): RFTCli> RETR dummy.pdf
b'RETR dummy.pdf'
SeqNum recieved: 1 | SeqNum Expected: 0
SeqNum recieved: 2 | SeqNum Expected: 0
SeqNum recieved: 3 | SeqNum Expected: 0
SeqNum recieved: 0 | SeqNum Expected: 0
seqNum match! writing to file and sending Ack # 0
SeqNum recieved: 1 | SeqNum Expected: 1
seqNum match! writing to file and sending Ack # 1
SeqNum recieved: 2 | SeqNum Expected: 2
seqNum match! writing to file and sending Ack # 2
SeqNum recieved: 3 | SeqNum Expected: 3
seqNum match! writing to file and sending Ack # 3
SeqNum recieved: 2 | SeqNum Expected: 4
SeqNum recieved: 4 | SeqNum Expected: 4
seqNum match! writing to file and sending Ack # 4
SeqNum recieved: 5 | SeqNum Expected: 5
seqNum match! writing to file and sending Ack # 5
SeqNum recieved: 6 | SeqNum Expected: 6
seqNum match! writing to file and sending Ack # 6
SeqNum recieved: 7 | SeqNum Expected: 7
seqNum match! writing to file and sending Ack # 7
SeqNum recieved: 8 | SeqNum Expected: 8
seqNum match! writing to file and sending Ack # 8
SeqNum recieved: 9 | SeqNum Expected: 9
seqNum match! writing to file and sending Ack # 9
SeqNum recieved: 10 | SeqNum Expected: 10
seqNum match! writing to file and sending Ack # 10
SeqNum recieved: 11 | SeqNum Expected: 11
seqNum match! writing to file and sending Ack # 11
SeqNum recieved: 12 | SeqNum Expected: 12
seqNum match! writing to file and sending Ack # 12
SeqNum recieved: 13 | SeqNum Expected: 13
seqNum match! writing to file and sending Ack # 13
Recieved dummy.pdf
PS C:\Users\joshs\Desktop\Networks\Client> python3 rft1Client.py
Provide Mode# (Type 'TCP' to skip UDP protocols): RFTCli> RETR Recording.mov
b'RETR Recording.mov'
SeqNum recieved: 0 | SeqNum Expected: 0
seqNum match! writing to file and sending Ack # 0
SeqNum recieved: 1 | SeqNum Expected: 1
seqNum match! writing to file and sending Ack # 1
Recieved Recording.mov
PS C:\Users\joshs\Desktop\Networks\Client> python3 rft1Client.py
Provide Mode# (Type 'TCP' to skip UDP protocols): RFTCli> RETR Recording.mp3
b'RETR Recording.mp3'
SeqNum recieved: 0 | SeqNum Expected: 0
```



```
Recieved message from ('127.0.0.1', 52691)
Sending the file...

Sending packet0 with seqNum 0
Sending packet1 with seqNum 1
Recieving ack...
Sending packet1 with seqNum 1
Recieving ack...
Transfer Complete!
```

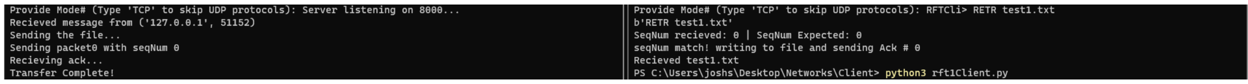
```
Provide Mode# (Type 'TCP' to skip UDP protocols): RFTCli> RETR Recording.mov
b'RETR Recording.mov'
SeqNum recieved: 0 | SeqNum Expected: 0
seqNum match! writing to file and sending Ack # 0
SeqNum recieved: 1 | SeqNum Expected: 1
seqNum match! writing to file and sending Ack # 1
Recieved Recording.mov
PS C:\Users\joshs\Desktop\Networks\Client> python3 rft1Client.py
```



```
Recieved message from ('127.0.0.1', 56334)
Sending the file...



Sending packet0 with seqNum 0
Sending packet1 with seqNum 1
Sending packet2 with seqNum 2
Sending packet3 with seqNum 3
Recieving ack...
Sending packet1 with seqNum 1
Recieving ack...
Sending packet2 with seqNum 2
Recieving ack...
Sending packet3 with seqNum 3
Recieving ack...
Timer ran out, resending window.
Sending packet3 with seqNum 3
Recieving ack...
Transfer Complete!
```

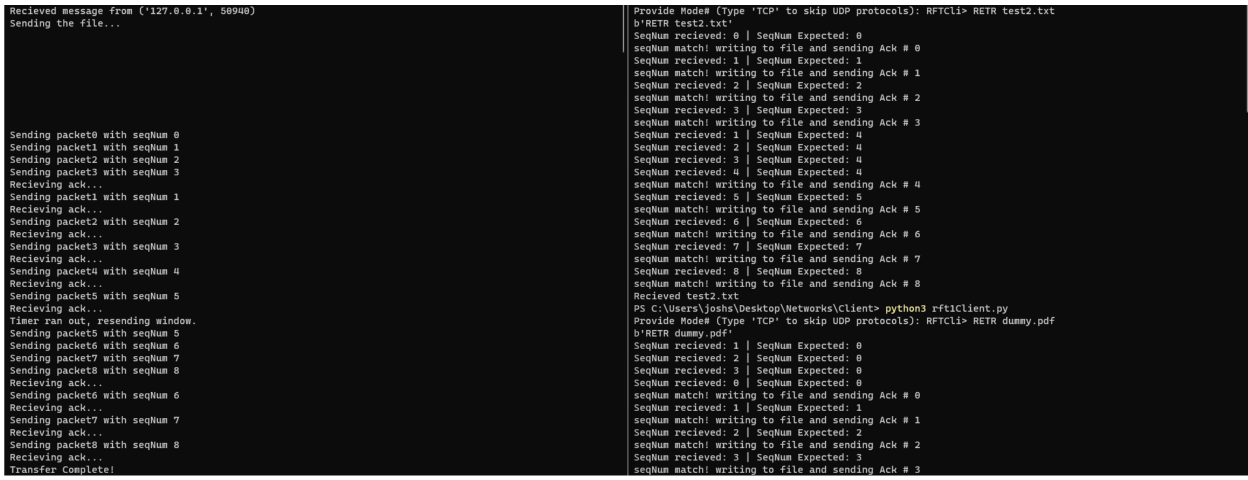
```
Provide Mode# (Type 'TCP' to skip UDP protocols): RFTCli> RETR Recording.mp3
b'RETR Recording.mp3'
SeqNum recieved: 0 | SeqNum Expected: 0
seqNum match! writing to file and sending Ack # 0
SeqNum recieved: 1 | SeqNum Expected: 1
seqNum match! writing to file and sending Ack # 1
SeqNum recieved: 2 | SeqNum Expected: 2
seqNum match! writing to file and sending Ack # 2
SeqNum recieved: 1 | SeqNum Expected: 3
SeqNum recieved: 2 | SeqNum Expected: 3
SeqNum recieved: 3 | SeqNum Expected: 3
seqNum match! writing to file and sending Ack # 3
Recieved Recording.mp3
PS C:\Users\joshs\Desktop\Networks\Client> python3 rft1Client.py
Provide Mode# (Type 'TCP' to skip UDP protocols): RFTCli> RETR test7.jpg
b'RETR test7.jpg'
SeqNum recieved: 0 | SeqNum Expected: 0
seqNum match! writing to file and sending Ack # 0
```

Joel Aguirre
Josue Soto
Computer Networks



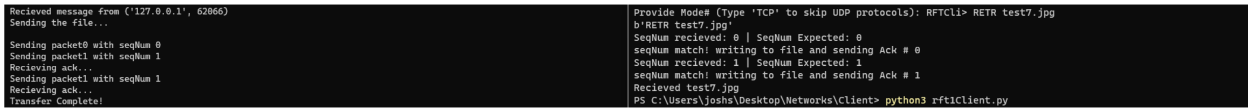
```
Provide Mode# (Type 'TCP' to skip UDP protocols): Server listening on 8000...
Recieved message from ('127.0.0.1', 51152)
Sending the file...
Sending packet0 with seqNum 0
Recieving ack...
Transfer Complete!
```

```
Provide Mode# (Type 'TCP' to skip UDP protocols): RFTCli> RETR test1.txt
b'RETR test1.txt'
SeqNum recieved: 0 | SeqNum Expected: 0
seqNum match! writing to file and sending Ack # 0
Recieved test1.txt
PS C:\Users\joshs\Desktop\Networks\Client> python3 rft1Client.py
```



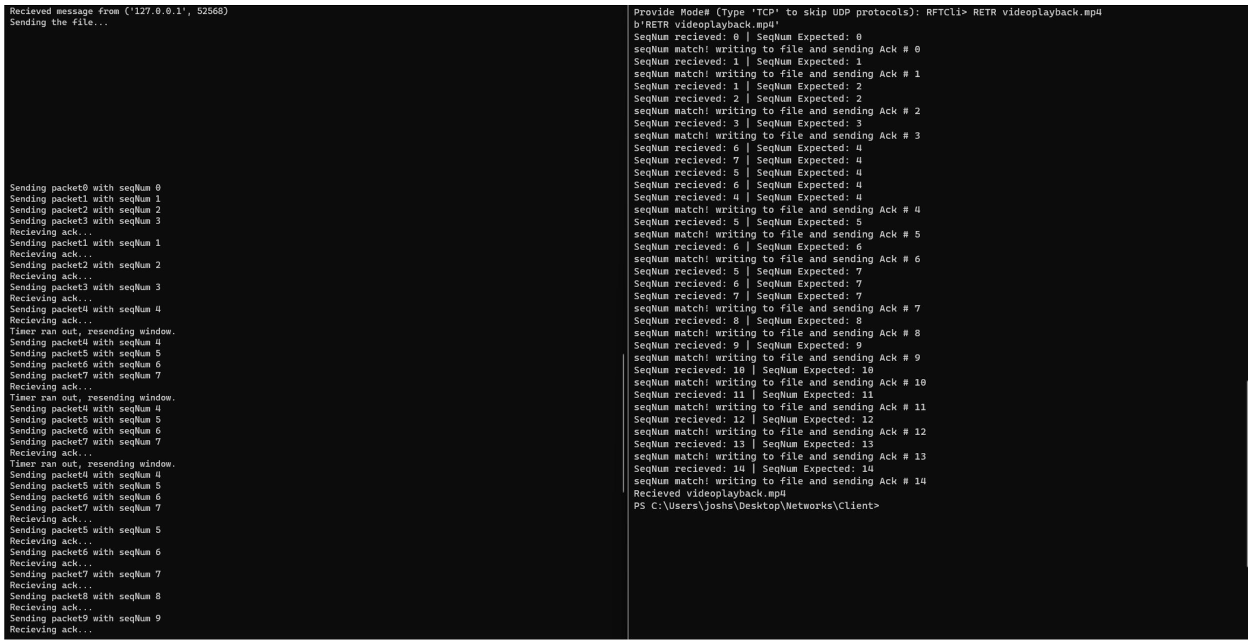
```
Recieved message from ('127.0.0.1', 50940)
Sending the file...









Sending packet0 with seqNum 0
Sending packet1 with seqNum 1
Sending packet2 with seqNum 2
Sending packet3 with seqNum 3
Recieving ack...
Sending packet1 with seqNum 1
Recieving ack...
Sending packet2 with seqNum 2
Recieving ack...
Sending packet3 with seqNum 3
Recieving ack...
Sending packet4 with seqNum 4
Recieving ack...
Sending packet5 with seqNum 5
Recieving ack...
Timer ran out, resending window.
Sending packet5 with seqNum 5
Sending packet6 with seqNum 6
Sending packet7 with seqNum 7
Sending packet8 with seqNum 8
Recieving ack...
Sending packet6 with seqNum 6
Recieving ack...
Sending packet7 with seqNum 7
Recieving ack...
Sending packet8 with seqNum 8
Recieving ack...
Transfer Complete!
```

```
Provide Mode# (Type 'TCP' to skip UDP protocols): RFTCli> RETR test2.txt
b'RETR test2.txt'
SeqNum recieved: 0 | SeqNum Expected: 0
seqNum match! writing to file and sending Ack # 0
SeqNum recieved: 1 | SeqNum Expected: 1
seqNum match! writing to file and sending Ack # 1
SeqNum recieved: 2 | SeqNum Expected: 2
seqNum match! writing to file and sending Ack # 2
SeqNum recieved: 3 | SeqNum Expected: 3
seqNum match! writing to file and sending Ack # 3
SeqNum recieved: 1 | SeqNum Expected: 4
SeqNum recieved: 2 | SeqNum Expected: 4
SeqNum recieved: 3 | SeqNum Expected: 4
SeqNum recieved: 4 | SeqNum Expected: 4
seqNum match! writing to file and sending Ack # 4
SeqNum recieved: 5 | SeqNum Expected: 5
seqNum match! writing to file and sending Ack # 5
SeqNum recieved: 6 | SeqNum Expected: 6
seqNum match! writing to file and sending Ack # 6
SeqNum recieved: 7 | SeqNum Expected: 7
seqNum match! writing to file and sending Ack # 7
SeqNum recieved: 8 | SeqNum Expected: 8
seqNum match! writing to file and sending Ack # 8
Recieved test2.txt
PS C:\Users\joshs\Desktop\Networks\Client> python3 rft1Client.py
Provide Mode# (Type 'TCP' to skip UDP protocols): RFTCli> RETR dummy.pdf
b'RETR dummy.pdf'
SeqNum recieved: 1 | SeqNum Expected: 0
SeqNum recieved: 2 | SeqNum Expected: 0
SeqNum recieved: 3 | SeqNum Expected: 0
SeqNum recieved: 0 | SeqNum Expected: 0
seqNum match! writing to file and sending Ack # 0
SeqNum recieved: 1 | SeqNum Expected: 1
seqNum match! writing to file and sending Ack # 1
SeqNum recieved: 2 | SeqNum Expected: 2
seqNum match! writing to file and sending Ack # 2
SeqNum recieved: 3 | SeqNum Expected: 3
seqNum match! writing to file and sending Ack # 3
```



```
Recieved message from ('127.0.0.1', 62066)
Sending the file...

Sending packet0 with seqNum 0
Sending packet1 with seqNum 1
Recieving ack...
Sending packet1 with seqNum 1
Recieving ack...
Transfer Complete!
```

```
Provide Mode# (Type 'TCP' to skip UDP protocols): RFTCli> RETR test7.jpg
b'RETR test7.jpg'
SeqNum recieved: 0 | SeqNum Expected: 0
seqNum match! writing to file and sending Ack # 0
SeqNum recieved: 1 | SeqNum Expected: 1
seqNum match! writing to file and sending Ack # 1
Recieved test7.jpg
PS C:\Users\joshs\Desktop\Networks\Client> python3 rft1Client.py
```

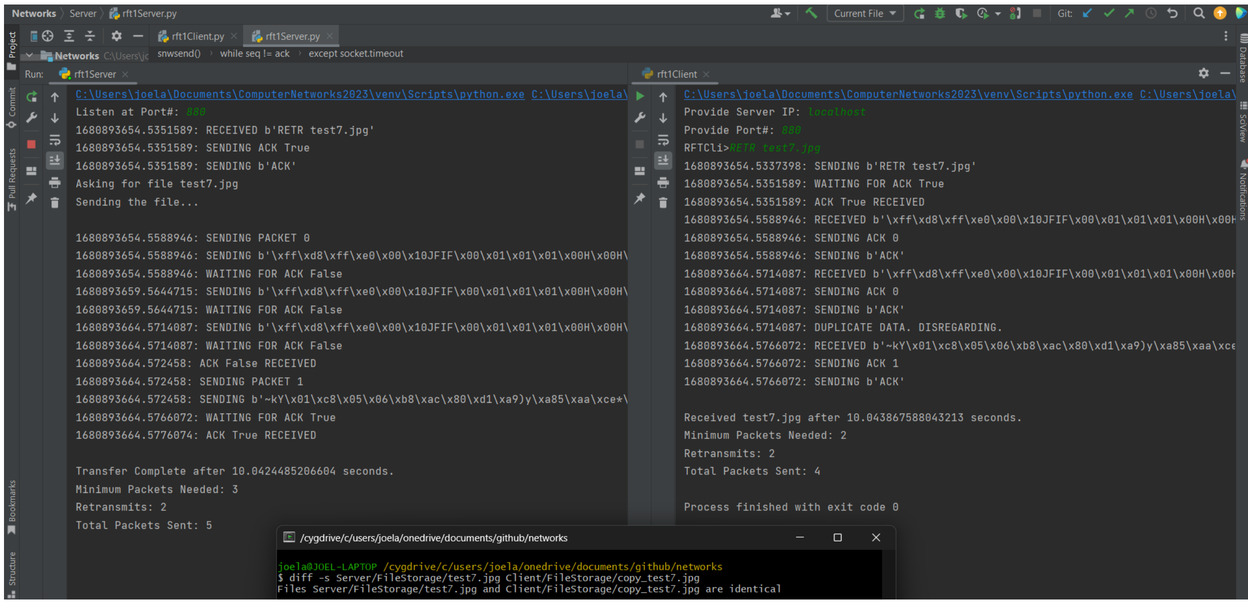


```
Recieved message from ('127.0.0.1', 52568)
Sending the file...














Sending packet0 with seqNum 0
Sending packet1 with seqNum 1
Sending packet2 with seqNum 2
Sending packet3 with seqNum 3
Recieving ack...
Sending packet1 with seqNum 1
Recieving ack...
Sending packet2 with seqNum 2
Recieving ack...
Sending packet3 with seqNum 3
Recieving ack...
Sending packet4 with seqNum 4
Recieving ack...
Timer ran out, resending window.
Sending packet4 with seqNum 4
Sending packet5 with seqNum 5
Sending packet6 with seqNum 6
Sending packet7 with seqNum 7
Recieving ack...
Timer ran out, resending window.
Sending packet4 with seqNum 4
Sending packet5 with seqNum 5
Sending packet6 with seqNum 6
Sending packet7 with seqNum 7
Recieving ack...
Timer ran out, resending window.
Sending packet4 with seqNum 4
Sending packet5 with seqNum 5
Sending packet6 with seqNum 6
Sending packet7 with seqNum 7
Recieving ack...
Sending packet5 with seqNum 5
Recieving ack...
Sending packet6 with seqNum 6
Recieving ack...
Sending packet7 with seqNum 7
Recieving ack...
Sending packet8 with seqNum 8
Recieving ack...
Sending packet9 with seqNum 9
Recieving ack...
```

```
Provide Mode# (Type 'TCP' to skip UDP protocols): RFTCli> RETR videoplayback.mp4
b'RETR videoplayback.mp4'
SeqNum recieved: 0 | SeqNum Expected: 0
seqNum match! writing to file and sending Ack # 0
SeqNum recieved: 1 | SeqNum Expected: 1
seqNum match! writing to file and sending Ack # 1
SeqNum recieved: 1 | SeqNum Expected: 2
SeqNum recieved: 2 | SeqNum Expected: 2
seqNum match! writing to file and sending Ack # 2
SeqNum recieved: 3 | SeqNum Expected: 3
seqNum match! writing to file and sending Ack # 3
SeqNum recieved: 6 | SeqNum Expected: 4
SeqNum recieved: 7 | SeqNum Expected: 4
SeqNum recieved: 5 | SeqNum Expected: 4
SeqNum recieved: 6 | SeqNum Expected: 4
SeqNum recieved: 4 | SeqNum Expected: 4
seqNum match! writing to file and sending Ack # 4
SeqNum recieved: 5 | SeqNum Expected: 5
seqNum match! writing to file and sending Ack # 5
SeqNum recieved: 6 | SeqNum Expected: 6
seqNum match! writing to file and sending Ack # 6
SeqNum recieved: 5 | SeqNum Expected: 7
SeqNum recieved: 6 | SeqNum Expected: 7
SeqNum recieved: 7 | SeqNum Expected: 7
seqNum match! writing to file and sending Ack # 7
SeqNum recieved: 8 | SeqNum Expected: 8
seqNum match! writing to file and sending Ack # 8
SeqNum recieved: 9 | SeqNum Expected: 9
seqNum match! writing to file and sending Ack # 9
SeqNum recieved: 10 | SeqNum Expected: 10
seqNum match! writing to file and sending Ack # 10
SeqNum recieved: 11 | SeqNum Expected: 11
seqNum match! writing to file and sending Ack # 11
SeqNum recieved: 12 | SeqNum Expected: 12
seqNum match! writing to file and sending Ack # 12
SeqNum recieved: 13 | SeqNum Expected: 13
seqNum match! writing to file and sending Ack # 13
SeqNum recieved: 14 | SeqNum Expected: 14
seqNum match! writing to file and sending Ack # 14
Recieved videoplayback.mp4
PS C:\Users\joshs\Desktop\Networks\Client>
```

b. Stop and Wait

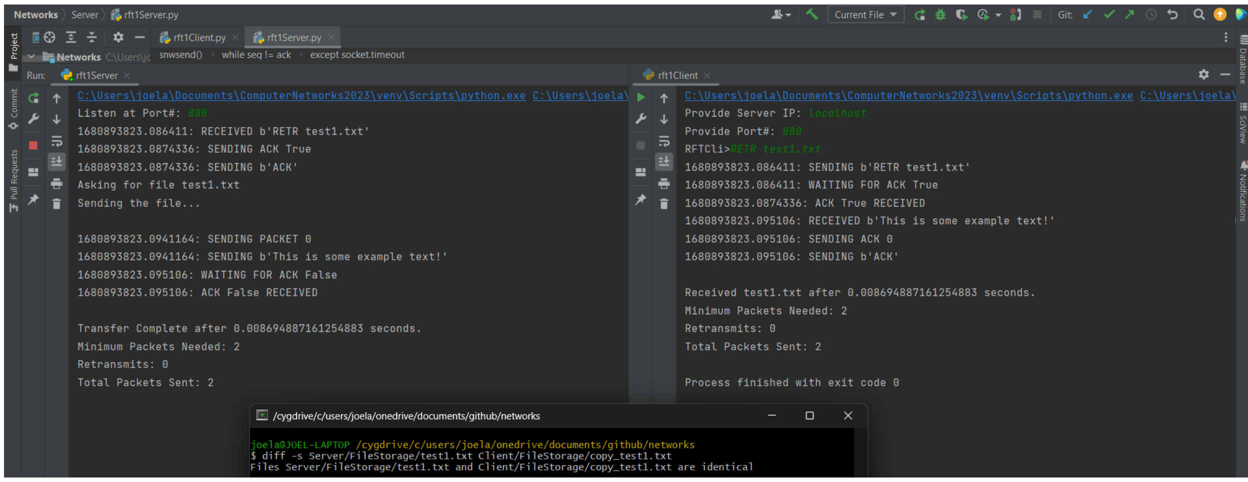
Joel Aguirre
Josue Soto
Computer Networks



```
C:\Users\joela\Documents\ComputerNetworks2023\venv\Scripts\python.exe C:\Users\joela\
Listen at Port#: 880
1680893654.5351589: RECEIVED b'RETR test7.jpg'
1680893654.5351589: SENDING ACK True
1680893654.5351589: SENDING b'ACK'
Asking for file test7.jpg
Sending the file...

1680893654.5588946: SENDING PACKET 0
1680893654.5588946: SENDING b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x01\x00H\x00H\
1680893654.5588946: WAITING FOR ACK False
1680893659.5644715: SENDING b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x01\x00H\x00H\
1680893659.5644715: WAITING FOR ACK False
1680893664.5714087: SENDING b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x01\x00H\x00H\
1680893664.5714087: WAITING FOR ACK False
1680893664.572458: ACK False RECEIVED
1680893664.572458: SENDING PACKET 1
1680893664.572458: SENDING b'~kY\x01\xc8\x05\x06\xb8\xac\x80\xd1\xa9)y\xa85\xaa\xce*\
1680893664.5766072: WAITING FOR ACK True
1680893664.5776074: ACK True RECEIVED

Transfer Complete after 10.0424485206604 seconds.
Minimum Packets Needed: 3
Retransmits: 2
Total Packets Sent: 5
```

```
C:\Users\joela\Documents\ComputerNetworks2023\venv\Scripts\python.exe C:\Users\joela\
Provide Server IP: localhost
Provide Port#: 880
RFTCli>RETR test7.jpg
1680893654.5337398: SENDING b'RETR test7.jpg'
1680893654.5351589: WAITING FOR ACK True
1680893654.5351589: ACK True RECEIVED
1680893654.5588946: RECEIVED b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x01\x00H\x00H
1680893654.5588946: SENDING ACK 0
1680893654.5588946: SENDING b'ACK'
1680893664.5714087: RECEIVED b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x01\x00H\x00H
1680893664.5714087: SENDING ACK 0
1680893664.5714087: SENDING b'ACK'
1680893664.5714087: DUPLICATE DATA. DISREGARDING.
1680893664.5766072: RECEIVED b'~kY\x01\xc8\x05\x06\xb8\xac\x80\xd1\xa9)y\xa85\xaa\xce
1680893664.5766072: SENDING ACK 1
1680893664.5766072: SENDING b'ACK'

Received test7.jpg after 10.043867588043213 seconds.
Minimum Packets Needed: 2
Retransmits: 2
Total Packets Sent: 4

Process finished with exit code 0
```

```
joela@JOEL-LAPTOP /cygdrive/c/users/joela/onedrive/documents/github/networks
$ diff -s Server/FileStorage/test7.jpg Client/FileStorage/copy_test7.jpg
Files Server/FileStorage/test7.jpg and Client/FileStorage/copy_test7.jpg are identical
```



```
C:\Users\joela\Documents\ComputerNetworks2023\venv\Scripts\python.exe C:\Users\joela\
Listen at Port#: 880
1680893823.086411: RECEIVED b'RETR test1.txt'
1680893823.0874336: SENDING ACK True
1680893823.0874336: SENDING b'ACK'
Asking for file test1.txt
Sending the file...

1680893823.0941164: SENDING PACKET 0
1680893823.0941164: SENDING b'This is some example text!'
1680893823.095106: WAITING FOR ACK False
1680893823.095106: ACK False RECEIVED

Transfer Complete after 0.008694887161254883 seconds.
Minimum Packets Needed: 2
Retransmits: 0
Total Packets Sent: 2
```

```
C:\Users\joela\Documents\ComputerNetworks2023\venv\Scripts\python.exe C:\Users\joela\
Provide Server IP: localhost
Provide Port#: 880
RFTCli>RETR test1.txt
1680893823.086411: SENDING b'RETR test1.txt'
1680893823.086411: WAITING FOR ACK True
1680893823.0874336: ACK True RECEIVED
1680893823.095106: RECEIVED b'This is some example text!'
1680893823.095106: SENDING ACK 0
1680893823.095106: SENDING b'ACK'

Received test1.txt after 0.008694887161254883 seconds.
Minimum Packets Needed: 2
Retransmits: 0
Total Packets Sent: 2

Process finished with exit code 0
```

```
joela@JOEL-LAPTOP /cygdrive/c/users/joela/onedrive/documents/github/networks
$ diff -s Server/FileStorage/test1.txt Client/FileStorage/copy_test1.txt
Files Server/FileStorage/test1.txt and Client/FileStorage/copy_test1.txt are identical
```

3. How to use

a. Go Back N

- Switch to the branch you want to use (Go Back N branch to test Go Back n)
- Have two terminals available
- In each terminal navigate to the respective folder, server and client.

(From local directory)

a. cd ./Server

b. cd ./Client

- For the server, type python3 rft1Server.py and type in the needed port #
- For the client, type python3 rft1Client.py and type in the needed port # and host

(Recognized commands work as RETR examplename.exampleformat to transfer a file)

(CLOSE will close the server)

- Type in TCP for 2a submission
- Type in GBN for Go Back N implementation in a D_GRAM socket

b. SnW

i. Same as above, however there is not toggle option, and it runs in SnW as default

4. References

Starter echo server and client functionality.

https://realpython.com/python-sockets/#echo-client-and-server

https://www2.tkn.tu-berlin.de/teaching/rn/animations/gbn_sr/

Work Attribution

Task List	Joel Aguirre	Josue Soto
Implement SnW	Did full implementation	
Implement GBN		Did full implementation