Samuel Ramos Sanchez
April 4, 2023
Computer Networks (CS-5313-001 CRN:25992) – Spring 2023
Instructor: Dr. Deepak K. Tosh
Programming Assignment 2b: Reliable File Transfer

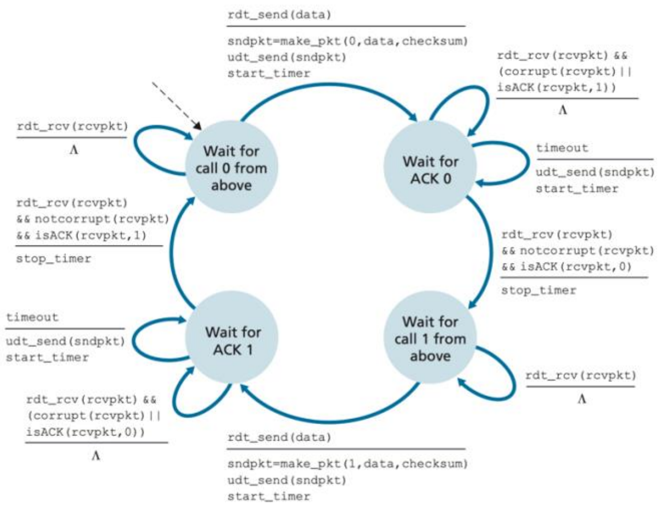
1. Program Explanation

This is a file transfer application that allows users to transfer any kind of file between a server and a client throughout an unreliable channel. The application is based on two Python socket programs, a server program, and a client program.
In this assignment the goal was implement transport layer programs that implement Stop-and-Wait (SnW) and Go-Back-N (GBN) protocols to upload files reliably over unreliable medium.
To run the programs is necessary to copy the folder "assignment_2b" to other respective folder and run client or server as "python3 rtf2Server.py" or "python3 rtf2Client.py" and type information asked.

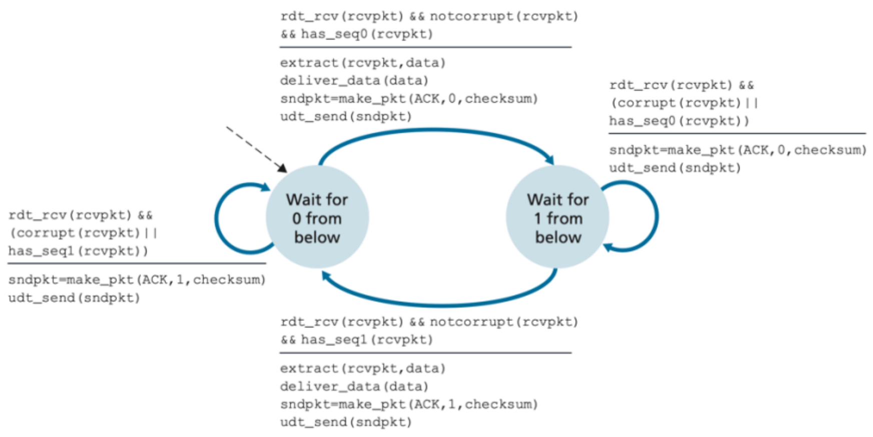
2. Solution Design

Stop-and-Wait (SnW)

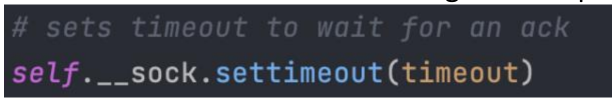
The sender is defined with next FSM:


rdt_send(data)
sndpkt=make_pkt(0,data,checksum)
udt_send(sndpkt)
start_timer
rdt_rcv(rcvpkt) && (corrupt(rcvpkt)|| isACK(rcvpkt,1))
Λ
rdt_rcv(rcvpkt)
Λ
Wait for call 0 from above
Wait for ACK 0
timeout
udt_send(sndpkt)
start_timer
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(rcvpkt,1)
stop_timer
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(rcvpkt,0)
stop_timer
timeout
udt_send(sndpkt)
start_timer
Wait for ACK 1
Wait for call 1 from above
rdt_rcv(rcvpkt)
Λ
rdt_rcv(rcvpkt) && (corrupt(rcvpkt)|| isACK(rcvpkt,0))
Λ
rdt_send(data)
sndpkt=make_pkt(1,data,checksum)
udt_send(sndpkt)
start_timer

The receiver is defined with next FSM:

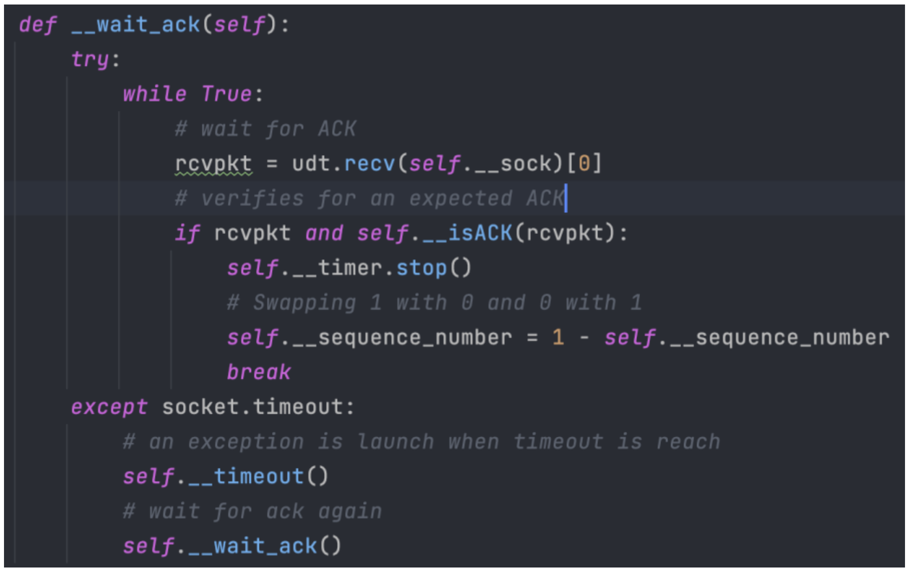

rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq0(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pkt(ACK,0,checksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && (corrupt(rcvpkt)|| has_seq0(rcvpkt))
sndpkt=make_pkt(ACK,0,checksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && (corrupt(rcvpkt)|| has_seq1(rcvpkt))
sndpkt=make_pkt(ACK,1,checksum)
udt_send(sndpkt)
Wait for 0 from below
Wait for 1 from below
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq1(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pkt(ACK,1,checksum)
udt_send(sndpkt)

Timeout for Stop-and-Wait (SnW) can be solved with professor timer module or setting a timeout to the socket and handling the exception received. I used both for double verification.



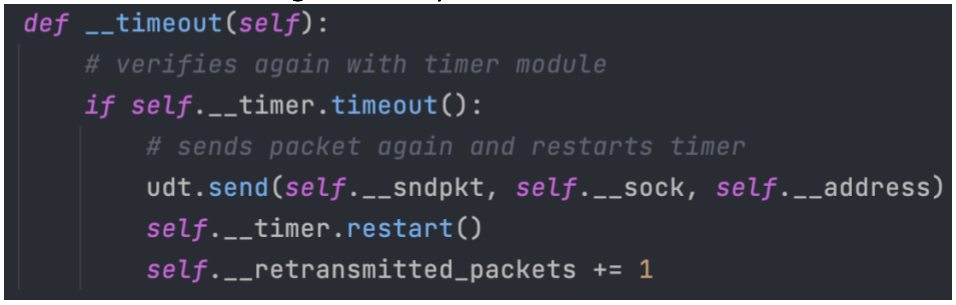
```python
# sets timeout to wait for an ack
self.__sock.settimeout(timeout)
```

Waits for an “ACK”.



```python
def __wait_ack(self):
    try:
        while True:
            # wait for ACK
            rcvpkt = udt.recv(self.__sock)[0]
            # verifies for an expected ACK
            if rcvpkt and self.__isACK(rcvpkt):
                self.__timer.stop()
                # Swapping 1 with 0 and 0 with 1
                self.__sequence_number = 1 - self.__sequence_number
                break
    except socket.timeout:
        # an exception is launch when timeout is reach
        self.__timeout()
        # wait for ack again
        self.__wait_ack()
```

This method verifies again if really it was a timeout.

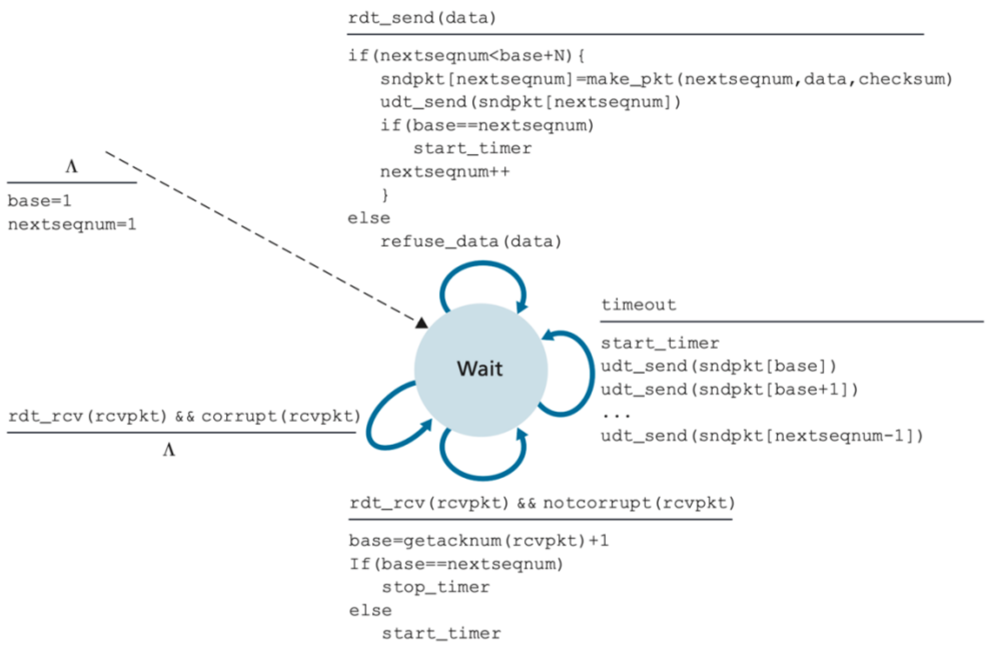


```python
def __timeout(self):
    # verifies again with timer module
    if self.__timer.timeout():
        # sends packet again and restarts timer
        udt.send(self.__sndpkt, self.__sock, self.__address)
        self.__timer.restart()
        self.__retransmitted_packets += 1
```

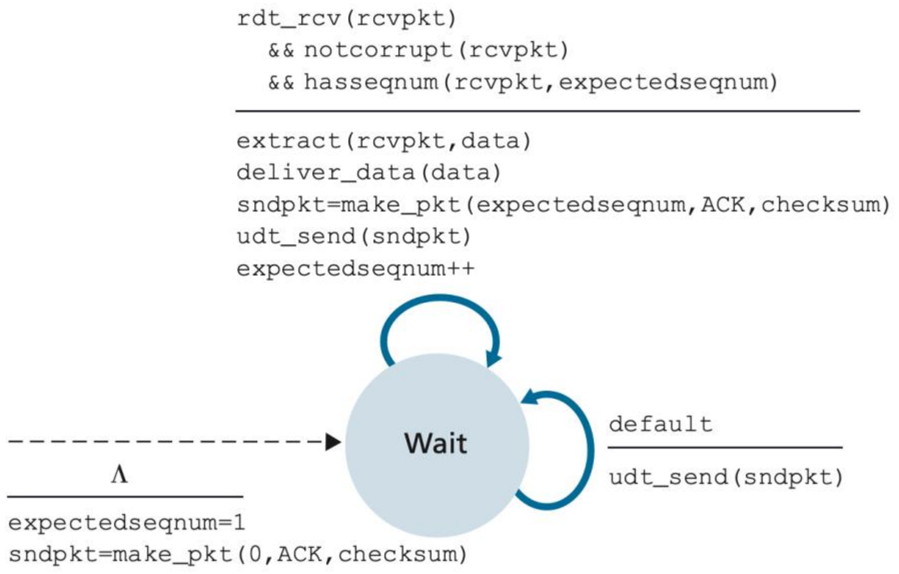
Go-Back-N (GBN)

In the Go-Back-N (GBN) protocol to handle socket read/write events asynchronously, I used “selectors” module that provides a platform-independent abstraction layer on top of the platform-specific I/O monitoring functions in select.

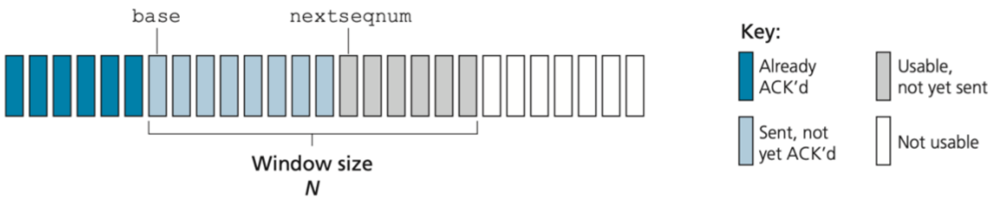
The sender for GBN is defined with next FSM:



The receiver for GBN is defined with next FSM:



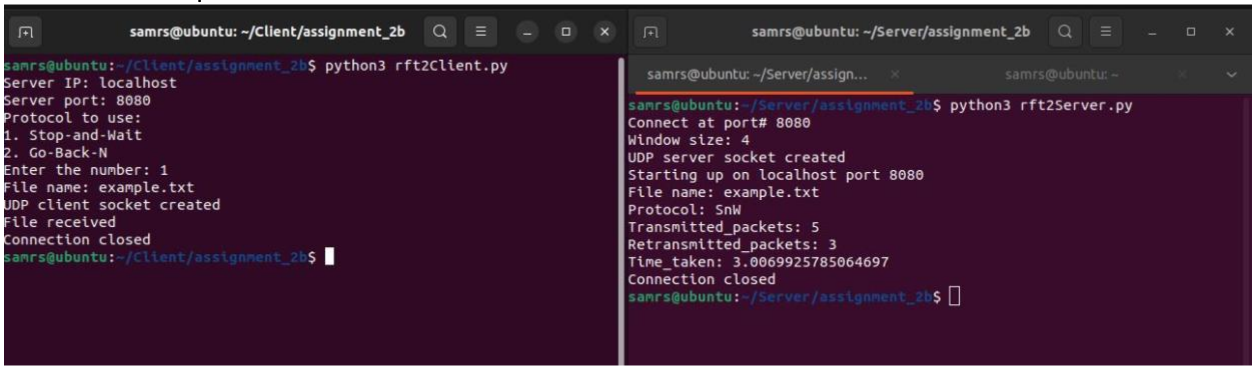
Window functionality works like next image:



3. Testing

Stop-and-Wait (SnW)

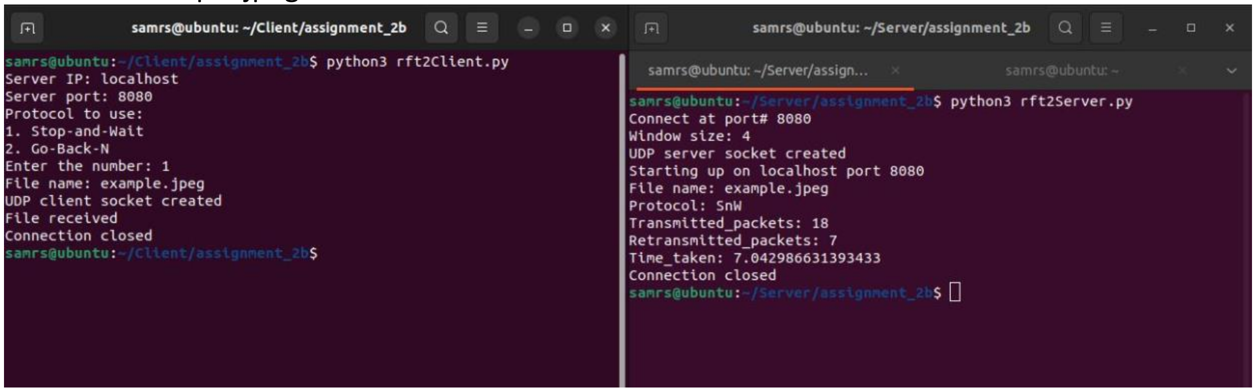
It sends “example.txt”.


samrs@ubuntu: ~/Client/assignment_2b
samrs@ubuntu: ~/Server/assignment_2b
samrs@ubuntu: ~/Server/assign...
samrs@ubuntu: ~

```
samrs@ubuntu:~/Client/assignment_2b$ python3 rft2Client.py
Server IP: localhost
Server port: 8080
Protocol to use:
1. Stop-and-Wait
2. Go-Back-N
Enter the number: 1
File name: example.txt
UDP client socket created
File received
Connection closed
samrs@ubuntu:~/Client/assignment_2b$
```

```
samrs@ubuntu:~/Server/assignment_2b$ python3 rft2Server.py
Connect at port# 8080
Window size: 4
UDP server socket created
Starting up on localhost port 8080
File name: example.txt
Protocol: SnW
Transmitted_packets: 5
Retransmitted_packets: 3
Time_taken: 3.0069925785064697
Connection closed
samrs@ubuntu:~/Server/assignment_2b$
```

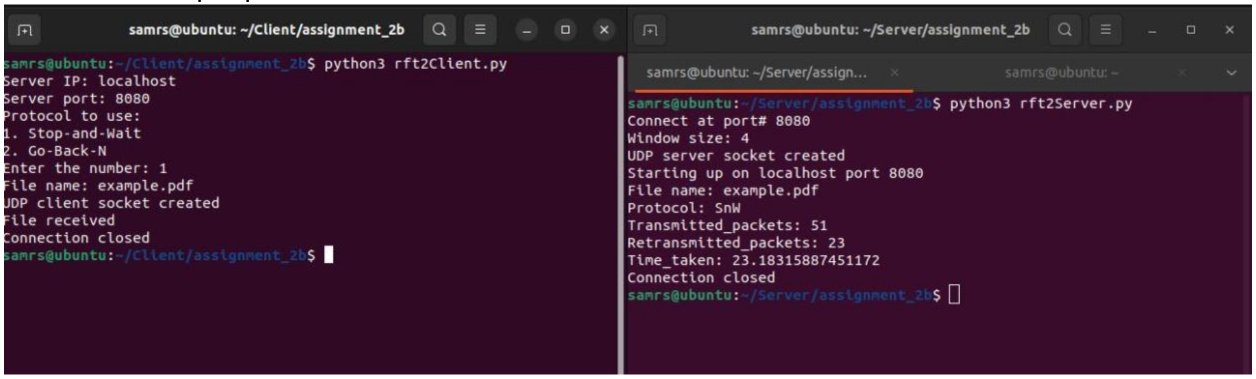
It sends “example.jpeg”.


samrs@ubuntu: ~/Client/assignment_2b
samrs@ubuntu: ~/Server/assignment_2b
samrs@ubuntu: ~/Server/assign...
samrs@ubuntu: ~

```
samrs@ubuntu:~/Client/assignment_2b$ python3 rft2Client.py
Server IP: localhost
Server port: 8080
Protocol to use:
1. Stop-and-Wait
2. Go-Back-N
Enter the number: 1
File name: example.jpeg
UDP client socket created
File received
Connection closed
samrs@ubuntu:~/Client/assignment_2b$
```

```
samrs@ubuntu:~/Server/assignment_2b$ python3 rft2Server.py
Connect at port# 8080
Window size: 4
UDP server socket created
Starting up on localhost port 8080
File name: example.jpeg
Protocol: SnW
Transmitted_packets: 18
Retransmitted_packets: 7
Time_taken: 7.042986631393433
Connection closed
samrs@ubuntu:~/Server/assignment_2b$
```

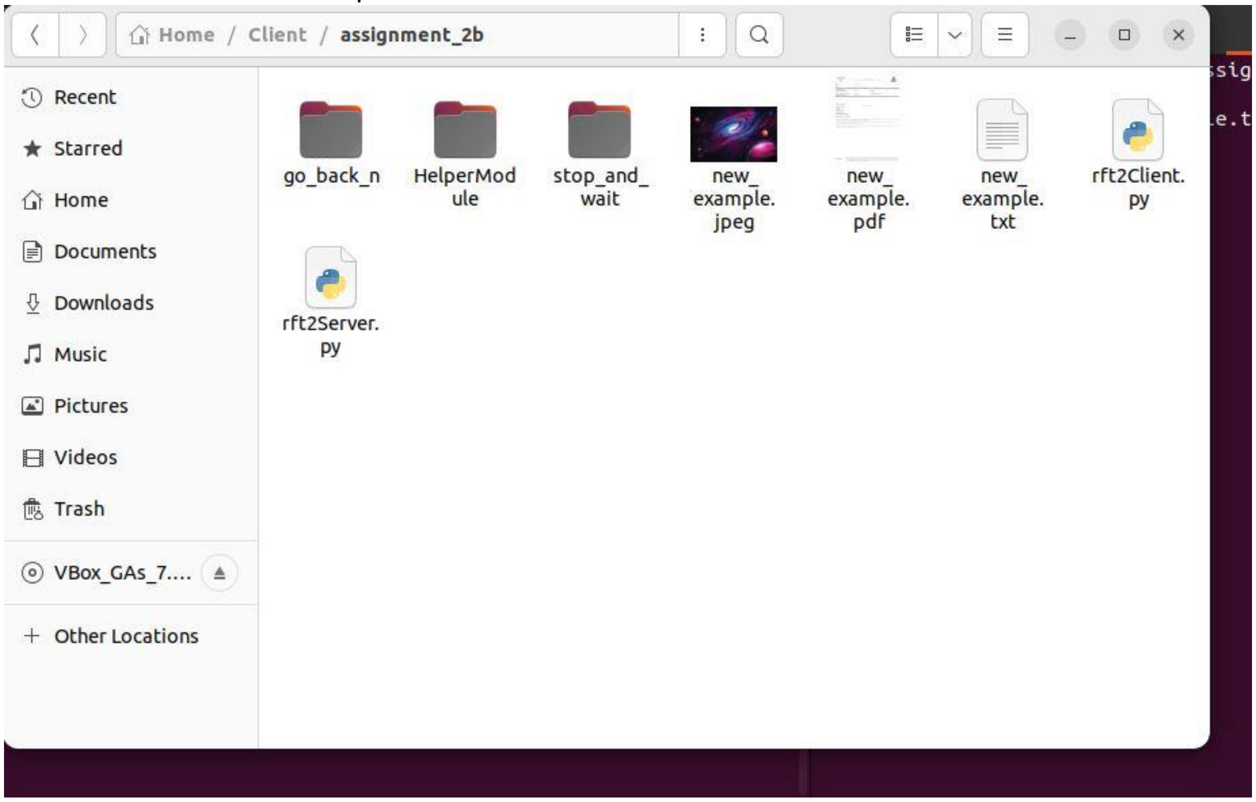
It sends “example.pdf”.


samrs@ubuntu: ~/Client/assignment_2b
samrs@ubuntu: ~/Server/assignment_2b
samrs@ubuntu: ~/Server/assign...
samrs@ubuntu: ~

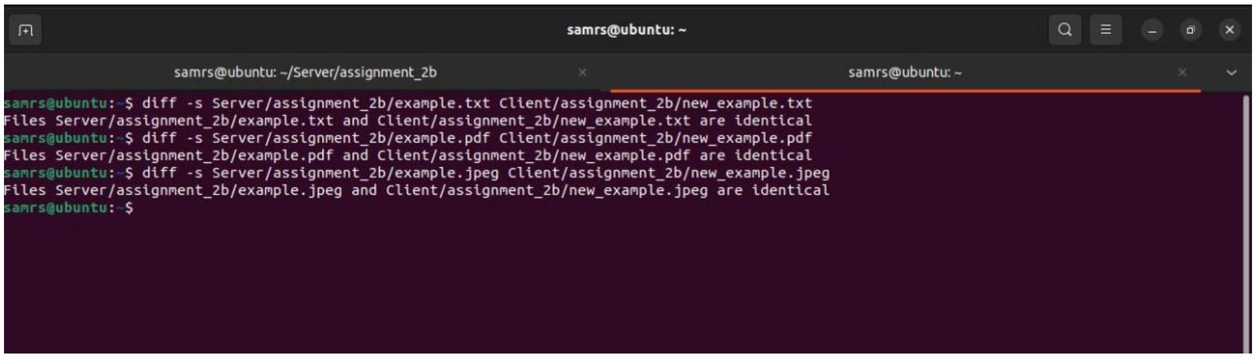
```
samrs@ubuntu:~/Client/assignment_2b$ python3 rft2Client.py
Server IP: localhost
Server port: 8080
Protocol to use:
1. Stop-and-Wait
2. Go-Back-N
Enter the number: 1
File name: example.pdf
UDP client socket created
File received
Connection closed
samrs@ubuntu:~/Client/assignment_2b$
```

```
samrs@ubuntu:~/Server/assignment_2b$ python3 rft2Server.py
Connect at port# 8080
Window size: 4
UDP server socket created
Starting up on localhost port 8080
File name: example.pdf
Protocol: SnW
Transmitted_packets: 51
Retransmitted_packets: 23
Time_taken: 23.18315887451172
Connection closed
samrs@ubuntu:~/Server/assignment_2b$
```

New files inside window explorer:



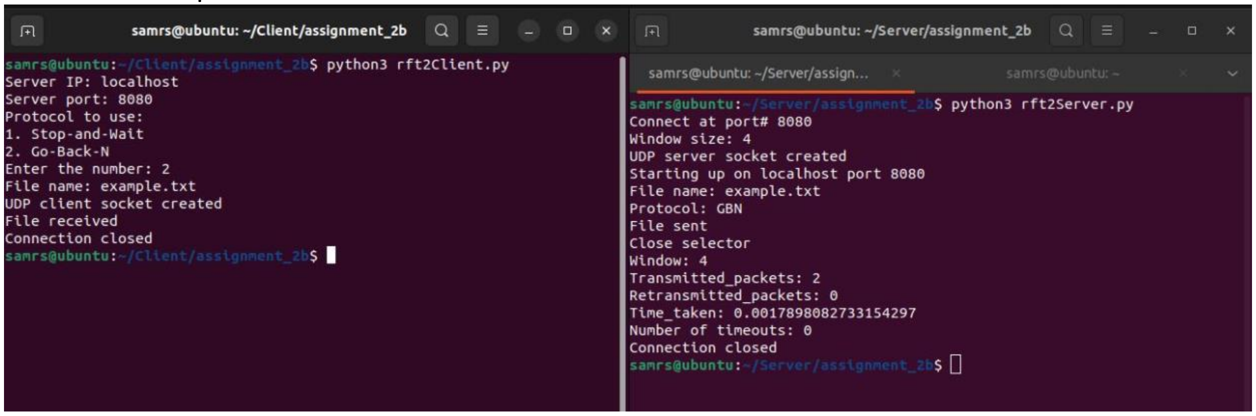
A DIFF command, “diff -s recvdFile sentFile” was used to test all files sent.
All files are identical.



```
samrs@ubuntu:~$ diff -s Server/assignment_2b/example.txt Client/assignment_2b/new_example.txt
Files Server/assignment_2b/example.txt and Client/assignment_2b/new_example.txt are identical
samrs@ubuntu:~$ diff -s Server/assignment_2b/example.pdf Client/assignment_2b/new_example.pdf
Files Server/assignment_2b/example.pdf and Client/assignment_2b/new_example.pdf are identical
samrs@ubuntu:~$ diff -s Server/assignment_2b/example.jpeg Client/assignment_2b/new_example.jpeg
Files Server/assignment_2b/example.jpeg and Client/assignment_2b/new_example.jpeg are identical
samrs@ubuntu:~$
```

Stop-and-Wait (SnW)
The timeout for all executions was 1 second.

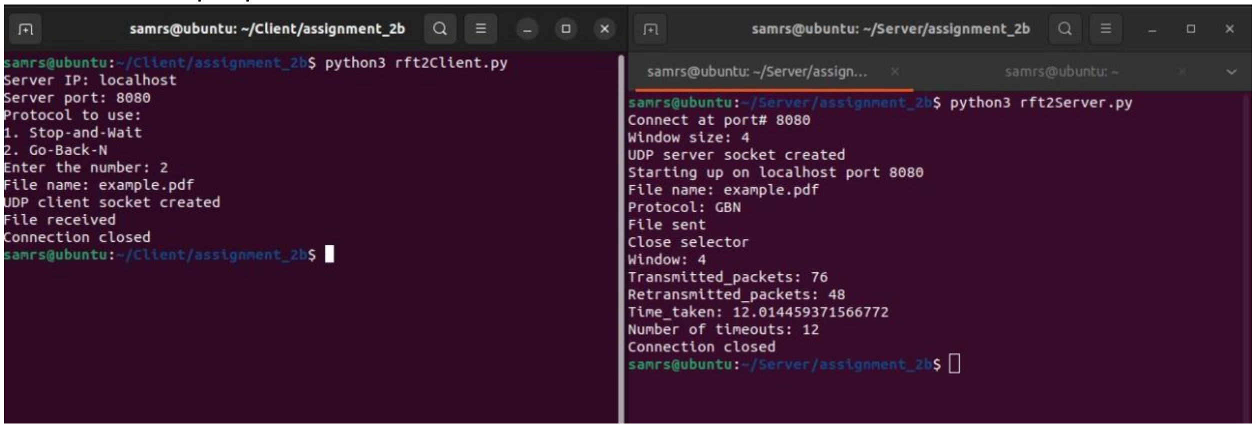
It sends "example.txt".



```
samrs@ubuntu:~/Client/assignment_2b$ python3 rft2Client.py
Server IP: localhost
Server port: 8080
Protocol to use:
1. Stop-and-Wait
2. Go-Back-N
Enter the number: 2
File name: example.txt
UDP client socket created
File received
Connection closed
samrs@ubuntu:~/Client/assignment_2b$
```

```
samrs@ubuntu:~/Server/assignment_2b$ python3 rft2Server.py
Connect at port# 8080
Window size: 4
UDP server socket created
Starting up on localhost port 8080
File name: example.txt
Protocol: GBN
File sent
Close selector
Window: 4
Transmitted_packets: 2
Retransmitted_packets: 0
Time_taken: 0.0017898082733154297
Number of timeouts: 0
Connection closed
samrs@ubuntu:~/Server/assignment_2b$
```

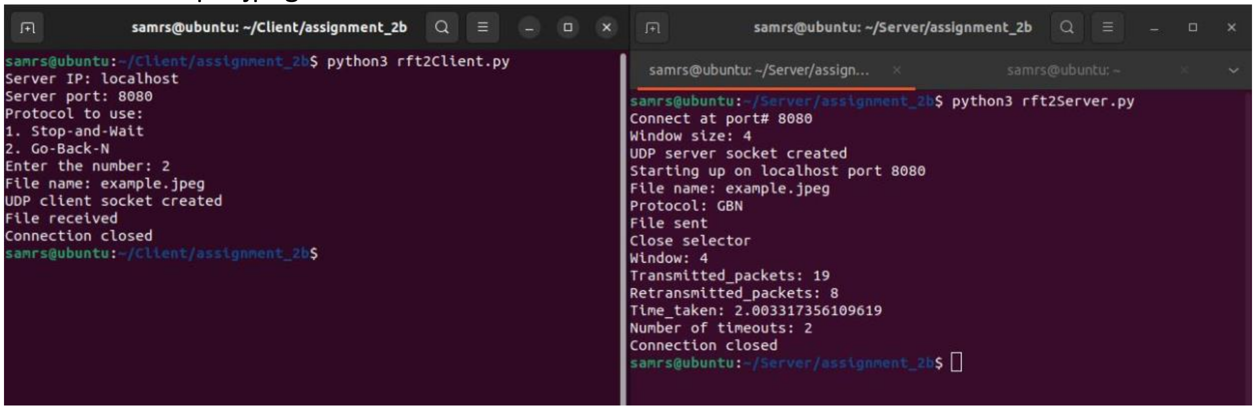
It sends "example.pdf".



```
samrs@ubuntu:~/Client/assignment_2b$ python3 rft2Client.py
Server IP: localhost
Server port: 8080
Protocol to use:
1. Stop-and-Wait
2. Go-Back-N
Enter the number: 2
File name: example.pdf
UDP client socket created
File received
Connection closed
samrs@ubuntu:~/Client/assignment_2b$
```

```
samrs@ubuntu:~/Server/assignment_2b$ python3 rft2Server.py
Connect at port# 8080
Window size: 4
UDP server socket created
Starting up on localhost port 8080
File name: example.pdf
Protocol: GBN
File sent
Close selector
Window: 4
Transmitted_packets: 76
Retransmitted_packets: 48
Time_taken: 12.014459371566772
Number of timeouts: 12
Connection closed
samrs@ubuntu:~/Server/assignment_2b$
```

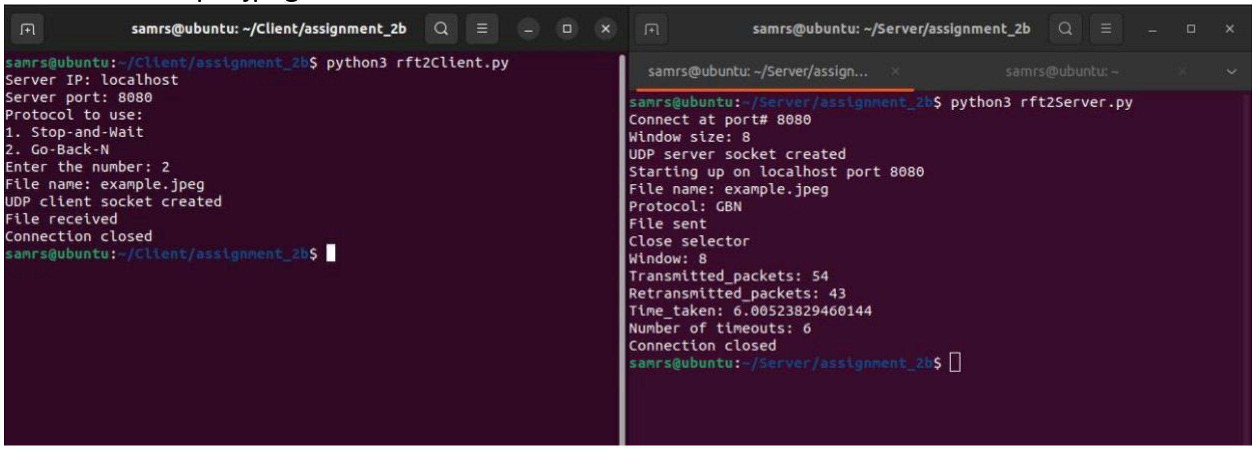
It sends "example.jpeg" with Window size of 4.



```
samrs@ubuntu:~/Client/assignment_2b$ python3 rft2Client.py
Server IP: localhost
Server port: 8080
Protocol to use:
1. Stop-and-Wait
2. Go-Back-N
Enter the number: 2
File name: example.jpeg
UDP client socket created
File received
Connection closed
samrs@ubuntu:~/Client/assignment_2b$
```

```
samrs@ubuntu:~/Server/assignment_2b$ python3 rft2Server.py
Connect at port# 8080
Window size: 4
UDP server socket created
Starting up on localhost port 8080
File name: example.jpeg
Protocol: GBN
File sent
Close selector
Window: 4
Transmitted_packets: 19
Retransmitted_packets: 8
Time_taken: 2.003317356109619
Number of timeouts: 2
Connection closed
samrs@ubuntu:~/Server/assignment_2b$
```

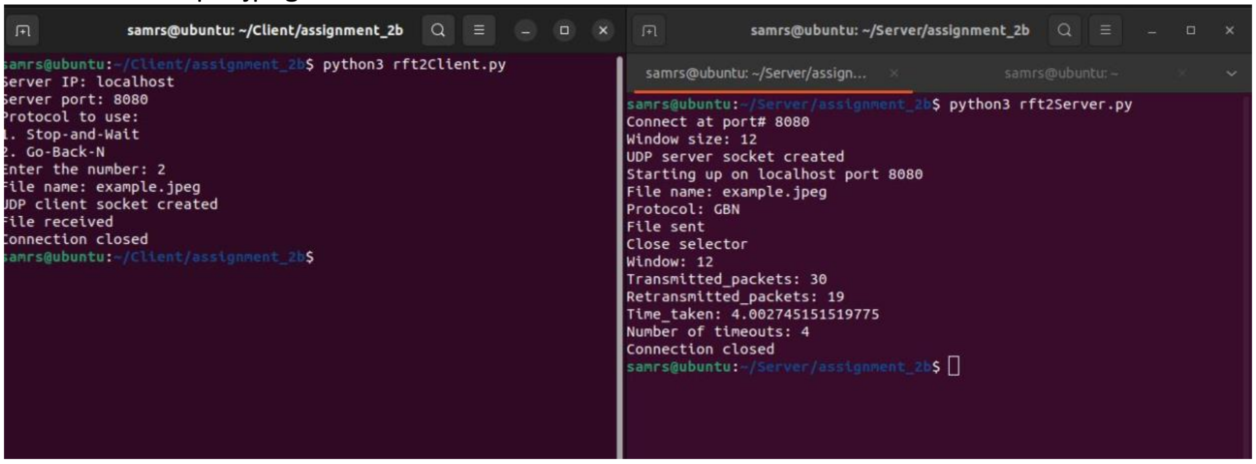
It sends “example.jpeg” with Window size of 8.


samrs@ubuntu: ~/Client/assignment_2b
samrs@ubuntu: ~/Server/assignment_2b
samrs@ubuntu: ~/Server/assign...
samrs@ubuntu: ~

```
samrs@ubuntu:~/Client/assignment_2b$ python3 rft2Client.py
Server IP: localhost
Server port: 8080
Protocol to use:
1. Stop-and-Wait
2. Go-Back-N
Enter the number: 2
File name: example.jpeg
UDP client socket created
File received
Connection closed
samrs@ubuntu:~/Client/assignment_2b$
```

```
samrs@ubuntu:~/Server/assignment_2b$ python3 rft2Server.py
Connect at port# 8080
Window size: 8
UDP server socket created
Starting up on localhost port 8080
File name: example.jpeg
Protocol: GBN
File sent
Close selector
Window: 8
Transmitted_packets: 54
Retransmitted_packets: 43
Time_taken: 6.00523829460144
Number of timeouts: 6
Connection closed
samrs@ubuntu:~/Server/assignment_2b$
```

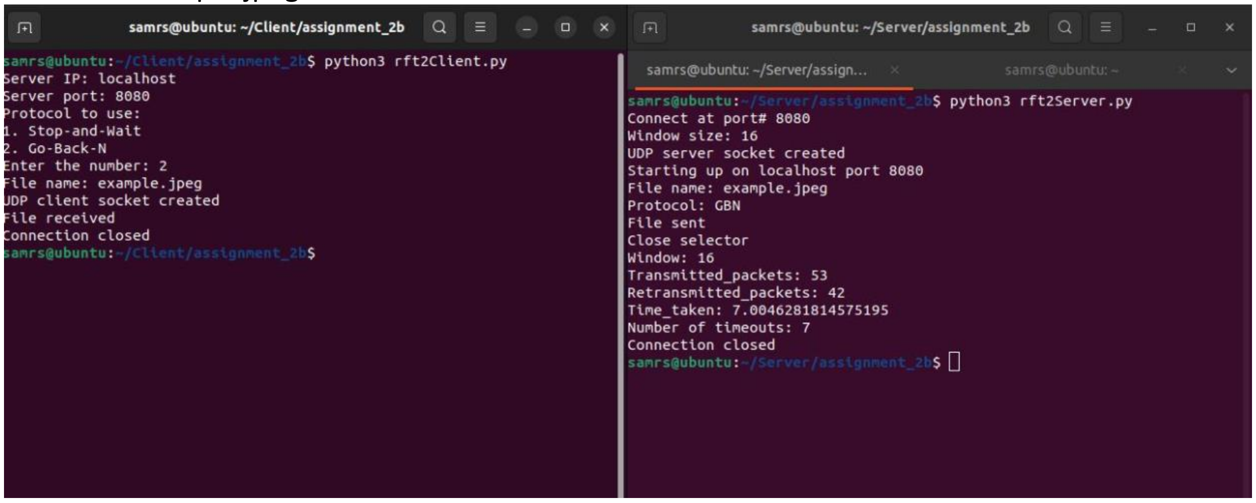
It sends “example.jpeg” with Window size of 12.


samrs@ubuntu: ~/Client/assignment_2b
samrs@ubuntu: ~/Server/assignment_2b
samrs@ubuntu: ~/Server/assign...
samrs@ubuntu: ~

```
amrs@ubuntu:~/Client/assignment_2b$ python3 rft2Client.py
erver IP: localhost
erver port: 8080
rotocol to use:
. Stop-and-Wait
. Go-Back-N
nter the number: 2
ile name: example.jpeg
DP client socket created
ile received
onnection closed
amrs@ubuntu:~/Client/assignment_2b$
```

```
samrs@ubuntu:~/Server/assignment_2b$ python3 rft2Server.py
Connect at port# 8080
Window size: 12
UDP server socket created
Starting up on localhost port 8080
File name: example.jpeg
Protocol: GBN
File sent
Close selector
Window: 12
Transmitted_packets: 30
Retransmitted_packets: 19
Time_taken: 4.002745151519775
Number of timeouts: 4
Connection closed
samrs@ubuntu:~/Server/assignment_2b$
```

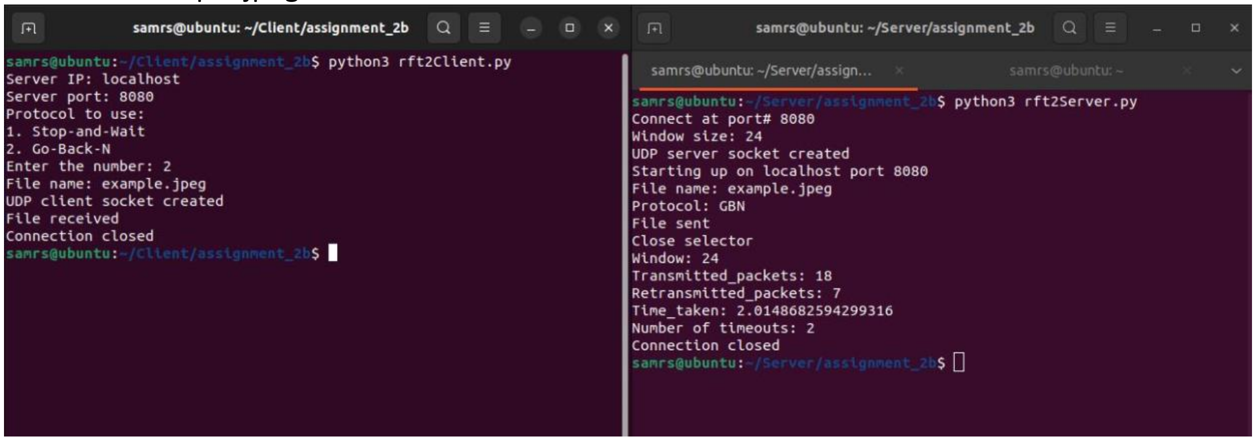
It sends “example.jpeg” with Window size of 16.


samrs@ubuntu: ~/Client/assignment_2b
samrs@ubuntu: ~/Server/assignment_2b
samrs@ubuntu: ~/Server/assign...
samrs@ubuntu: ~

```
amrs@ubuntu:~/Client/assignment_2b$ python3 rft2Client.py
erver IP: localhost
erver port: 8080
rotocol to use:
. Stop-and-Wait
. Go-Back-N
nter the number: 2
ile name: example.jpeg
DP client socket created
ile received
onnection closed
amrs@ubuntu:~/Client/assignment_2b$
```

```
samrs@ubuntu:~/Server/assignment_2b$ python3 rft2Server.py
Connect at port# 8080
Window size: 16
UDP server socket created
Starting up on localhost port 8080
File name: example.jpeg
Protocol: GBN
File sent
Close selector
Window: 16
Transmitted_packets: 53
Retransmitted_packets: 42
Time_taken: 7.0046281814575195
Number of timeouts: 7
Connection closed
samrs@ubuntu:~/Server/assignment_2b$
```

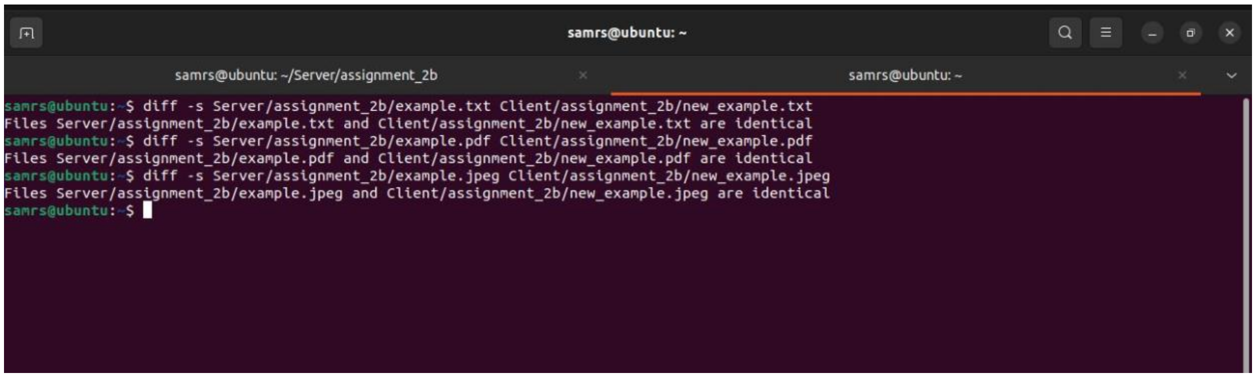
It sends “example.jpeg” with Window size of 24.


samrs@ubuntu: ~/Client/assignment_2b
samrs@ubuntu: ~/Server/assignment_2b
samrs@ubuntu: ~/Server/assign...
samrs@ubuntu: ~

```
samrs@ubuntu:~/Client/assignment_2b$ python3 rft2Client.py
Server IP: localhost
Server port: 8080
Protocol to use:
1. Stop-and-Wait
2. Go-Back-N
Enter the number: 2
File name: example.jpeg
UDP client socket created
File received
Connection closed
samrs@ubuntu:~/Client/assignment_2b$
```

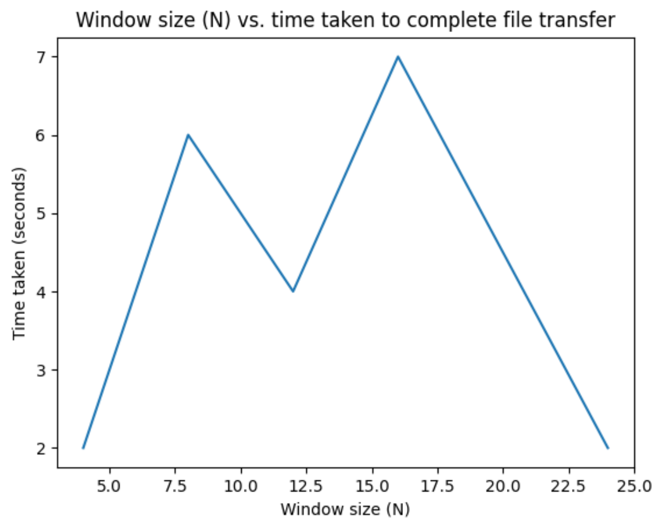
```
samrs@ubuntu:~/Server/assignment_2b$ python3 rft2Server.py
Connect at port# 8080
Window size: 24
UDP server socket created
Starting up on localhost port 8080
File name: example.jpeg
Protocol: GBN
File sent
Close selector
Window: 24
Transmitted_packets: 18
Retransmitted_packets: 7
Time_taken: 2.0148682594299316
Number of timeouts: 2
Connection closed
samrs@ubuntu:~/Server/assignment_2b$
```

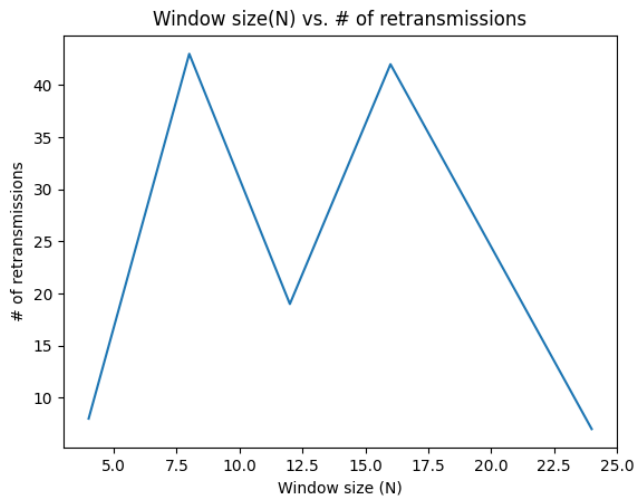
A DIFF command, “diff -s recvdFile sentFile” was used to test all files sent.
All files are identical.

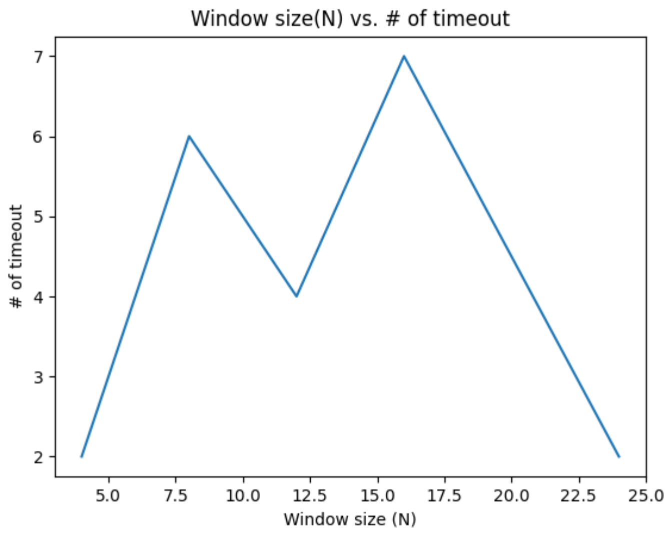


```
samrs@ubuntu:~$ diff -s Server/assignment_2b/example.txt Client/assignment_2b/new_example.txt
Files Server/assignment_2b/example.txt and Client/assignment_2b/new_example.txt are identical
samrs@ubuntu:~$ diff -s Server/assignment_2b/example.pdf Client/assignment_2b/new_example.pdf
Files Server/assignment_2b/example.pdf and Client/assignment_2b/new_example.pdf are identical
samrs@ubuntu:~$ diff -s Server/assignment_2b/example.jpeg Client/assignment_2b/new_example.jpeg
Files Server/assignment_2b/example.jpeg and Client/assignment_2b/new_example.jpeg are identical
samrs@ubuntu:~$
```

Figures:







4. Conclusion

All the files were successfully transmitted by the programs without any issues, and upon verification, it was found that the transmitted files were identical to the original ones. In the case of the Go-Back-N protocol, it was observed that the best window sizes were 4 and 24, but the time taken for transmission can be affected by the timeout value.
During the transmission, one of the challenges faced was the transfer of additional information such as the file name and the protocol from the client to the server, as this information could also be lost in the process. Another challenge was encountered when sending the "end of file" to the client, where the client is notified when the end of the file is reached. However, it is possible to lose the last ACK signal to the server, indicating the successful receipt of the file.

References:

[1] https://realpython.com/python-sockets/#multi-connection-client-and-server
[2] https://pymotw.com/3/selectors/
[3] https://docs.python.org/3/library/selectors.html