CS5313: Computer Network

Fall 2020

Instructor: Dr. Deepak Tosh

**Assignment 2**

**Erik Macik, M.S.**

**Aaron Espinosa, M.S**

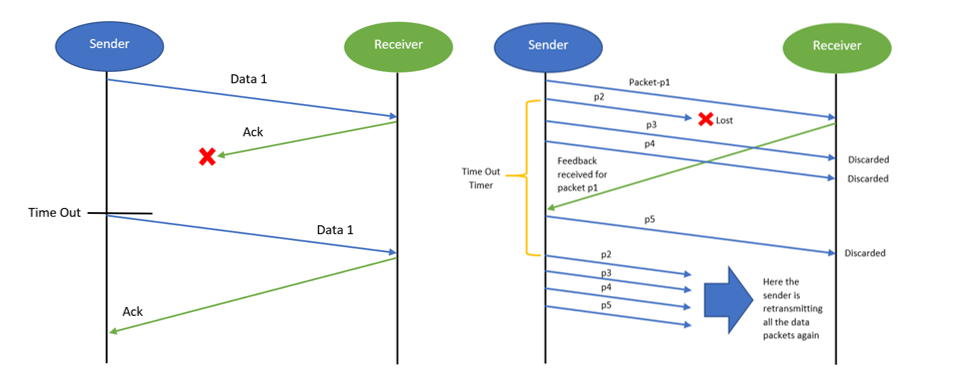
**Introduction**

It is important to understand how reliable data transfer works. Since reliable data transfer in a general context is considered a problem, as it does not only occur at the transport layer, it also occurs at the link and application layers. The general problem is thus of central importance to networking. [1]

Within the reliable data transfer protocol, a simple alternating bit protocol can be designed. This protocol is also known as a stop-and-wait protocol: after sending each packet the sender stops and waits for feedback from the receiver indicating that the packet has been received. [2]

The stop-and-wait protocol offers good control over the flow of errors, but it can cause significant performance problems, as the sender always waits for confirmation, even if they have the next packet ready to send. The sliding window protocol handles this efficiency problem by sending more than one packet at a time with larger sequence number. Practically it is implemented in two protocols namely: Go back N (GBN) and Selective Repeat (SR) [3]

In this report, we will explain how we implement both versions of the reliable data transfer protocol: Stop-and-Wait protocol and Go-Back-N (GBN) protocol using python programming language. Figure 1.



Go-Back-N

Stop-and-Wait

Figure 1 Stop-and-Wait and Go-Back-N protocols.

**Method**

First the receiver and sender files were run on two different consoles to see how it worked. Figure 2.

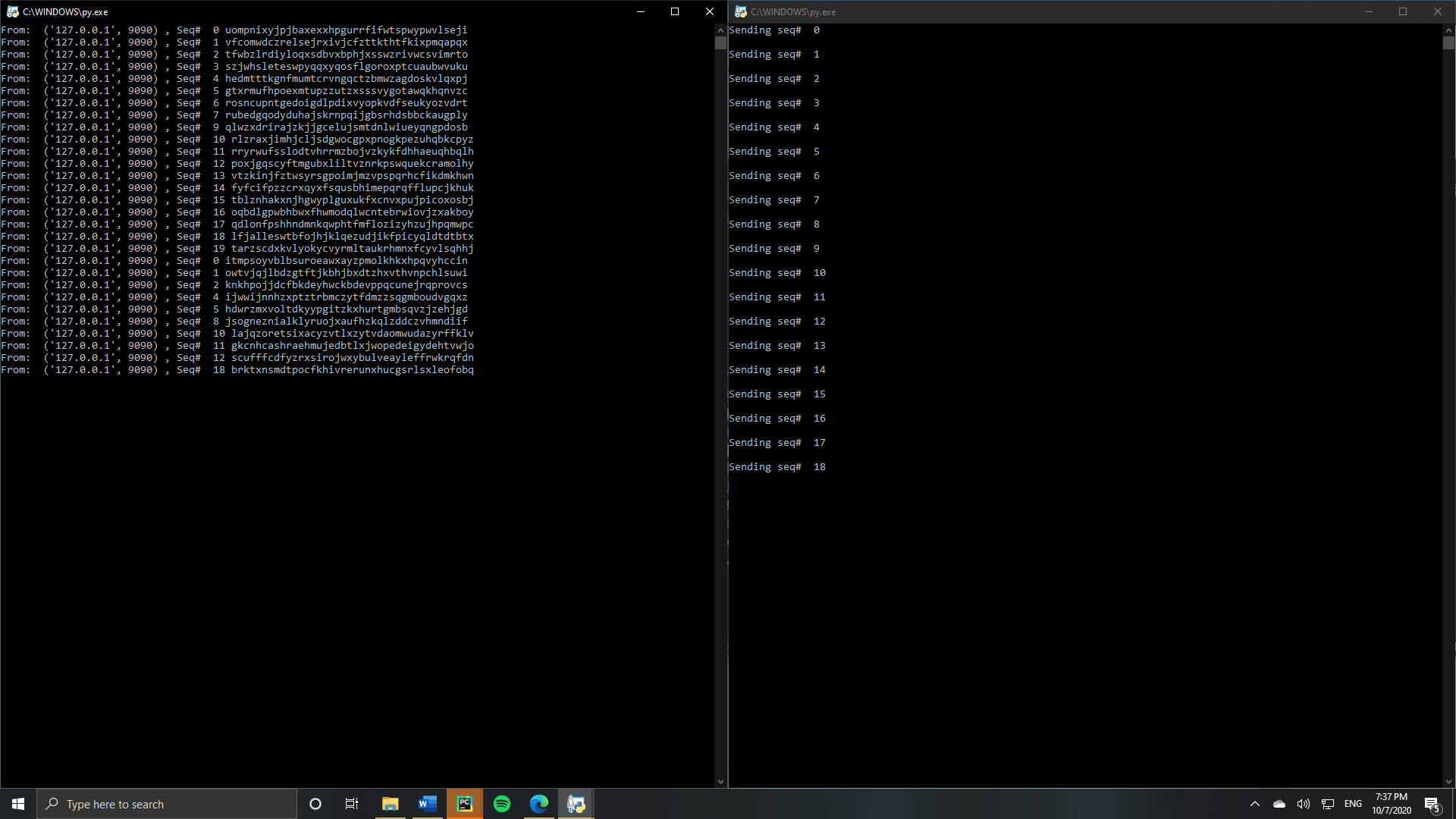


Figure 2 Receiver and Sender work.

After reviewing the code, we focus on the sender. We add the libraries that would be needed and uncomment \_thread. Figure 3.

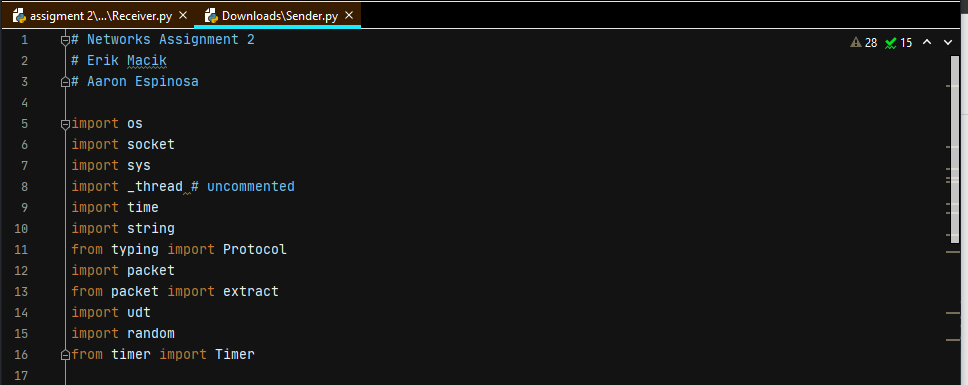


Figure 3 Import modules.

In this part the lines of code that are seen in the following image (figure 4) were uncommented and a new variable declared as *end*.

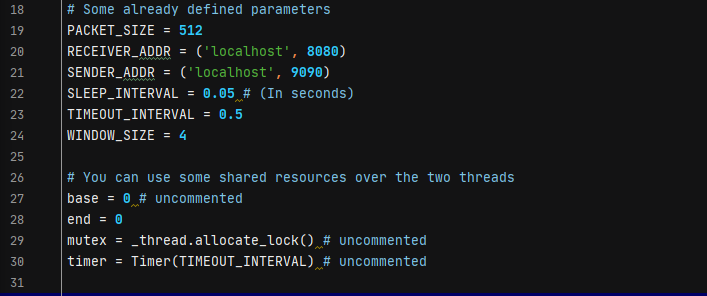


Figure 4 Adding and uncomment variables.

Then, we specify the number of bytes of our file data to send assigning to our function payload the file and the size of the package. Later, all packets are getting from a file along with an ending packet. Finally, a list of packets is created of size of *packet\_size* from the file. (Figure 5)

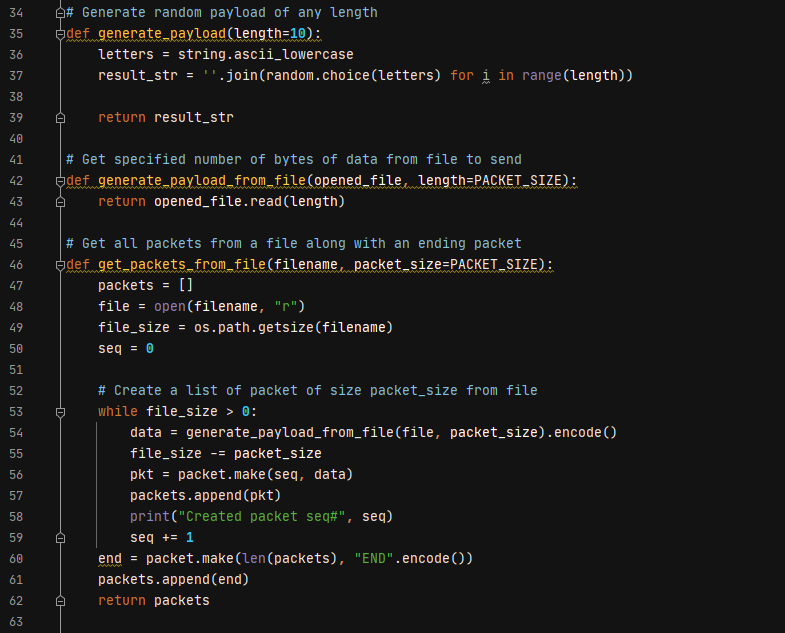


Figure 5 Creating and assigning the package size.

And the stop-and-wait protocol code was left the same. Figure 6.

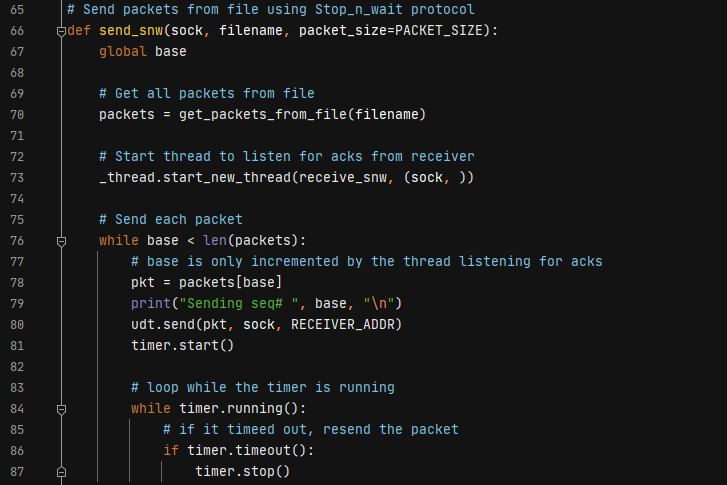


Figure 6 Stop-and-Wait protocol.

Once having the sender ready we move on to the receiver code. Here, we incorporate the missing part of the Stop-and-Wait code adding the file name and the variables that we used. Then, we added a while loop to continually receive the packets until we get the last one. Inside of the same loop, we used a condition statements if to tell the program if we already have the packet send a repeat ack, and write the data to a file if it does not end then send an ack of that packet to the sender. Figure 7.

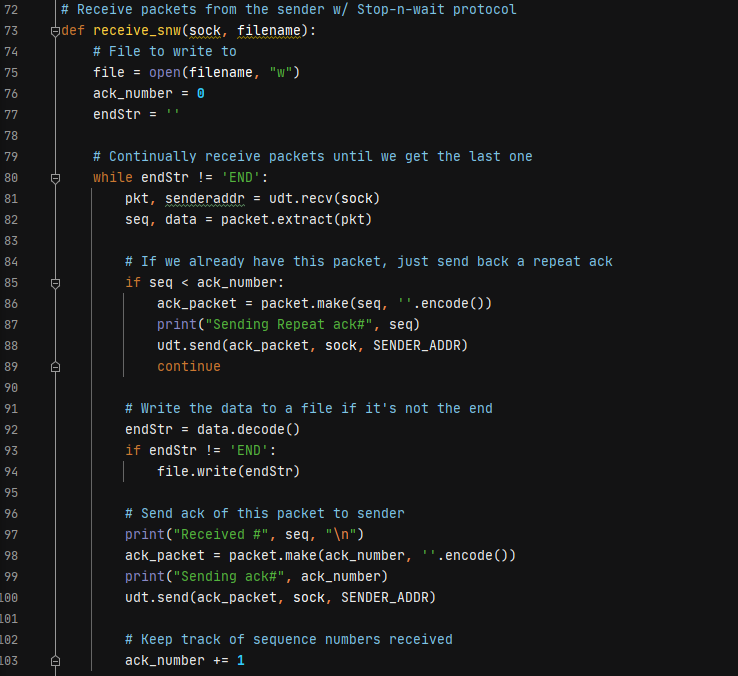


Figure 7 Stop-and-Wait Receiver code.

Once having the sender ready we move on to the receive code. Here, we incorporate the missing part of the GBN code adding the file name and the variables that we used. Then, we added a while loop to continually receive the packets until we get the last one. Figure 7.

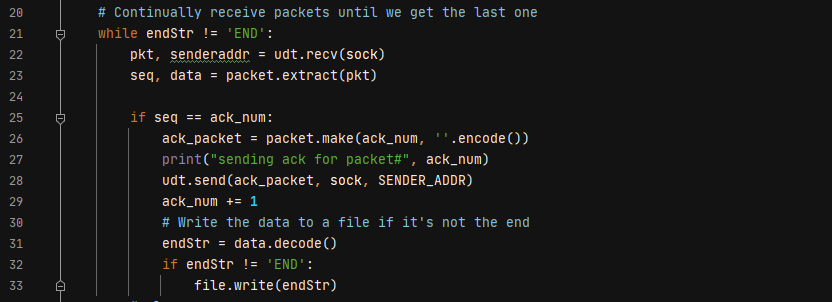


Figure 8 Receive packets from the sender Stop-and-Wait.

Finally, for the GBN protocol we incorporate the missing part code adding the file name and the variables that we used as well. Then, we added a while loop to continually receive the packets until we get the last one. And again, we used condition statements if to write the data to a file if it is not the end. Figure 9.

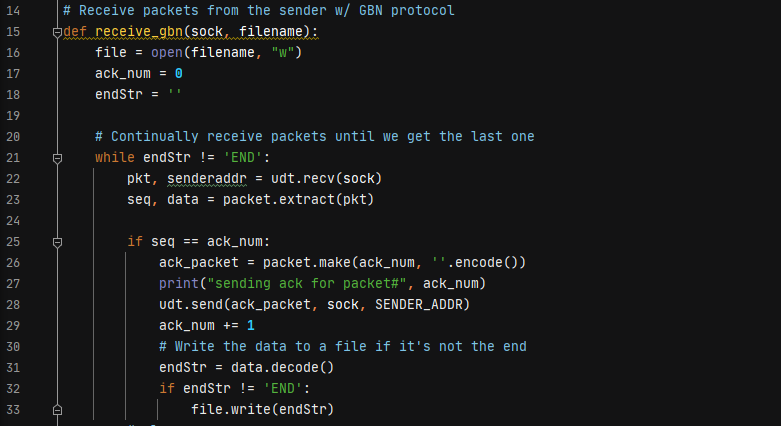
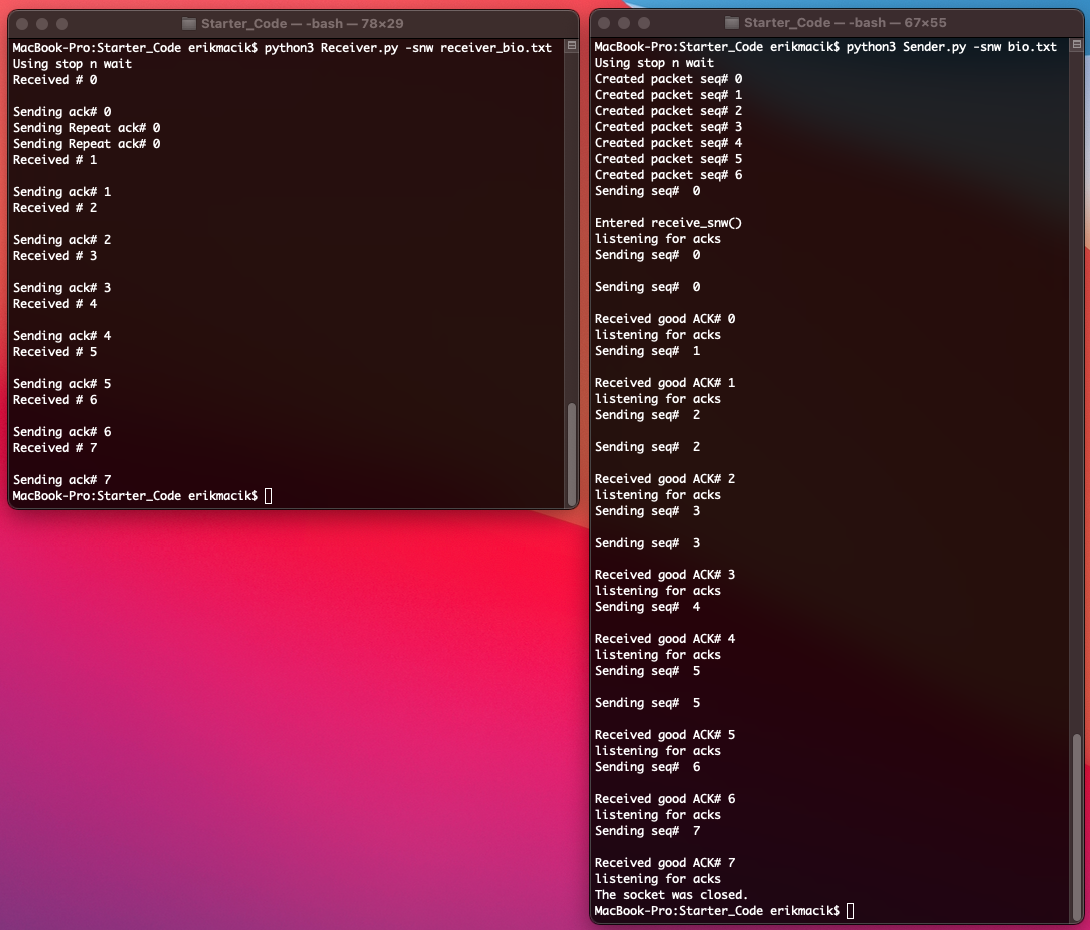


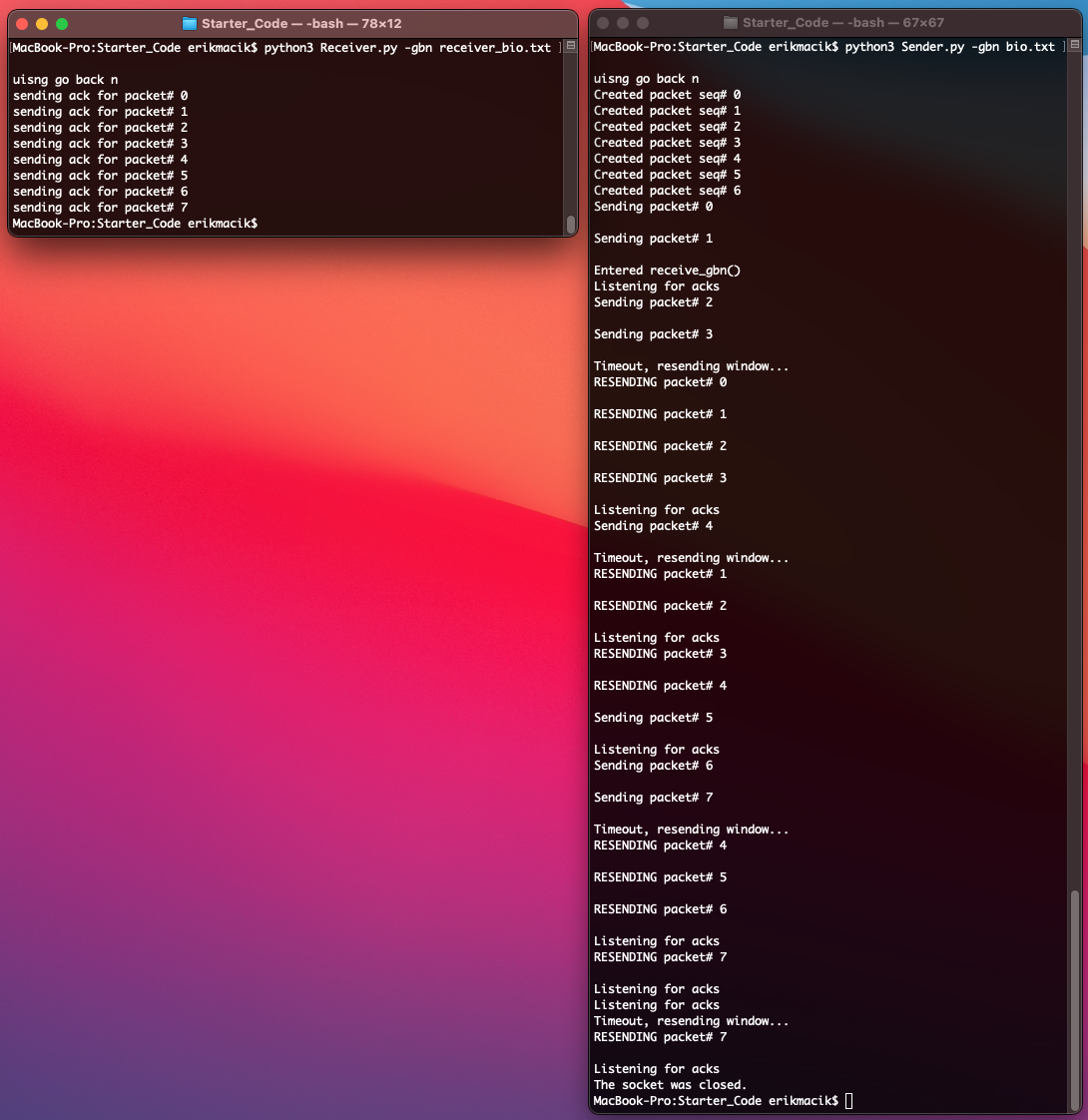
Figure 9 Receive packets from the sender GBN.

**Conclusions**

Stop-and-Wait Protocol



Go-Back-N Protocol



**Students contribution**

|  |  |  |
| --- | --- | --- |
| Task list | Erik Macik | Aaron Espinosa |
| Code | ✓ | ✓ |
| Report | ✓ | ✓ |

**References**

|  |  |
| --- | --- |
| [1] | James F. Kurose, Keith W. Ross, «3.4 Principles of Reliable Data Transfer,» de *Computer Networking A Top-Down Approach*, New Jersey, Pearson, 2017. |
| [2] | G. Shute, «Reliable Data Transfer,» University Of Minnesota Duluth, [En línea]. Available: https://www.d.umn.edu/~gshute/net/reliable-data-transfer.xhtml#:~:text=For%20connection%2Doriented%20service%20provided,designed%20using%20some%20basic%20tools.. [Último acceso: 5 Octubre 2020]. |
| [3] | James F. Kurose, Keith W. Ross, «Computer Networking A Top-Down Approach,» de *3.4.3 Go-Back-N (GBN)*, New Jersey, Pearson, 2017. |