**Final Project:**

**Implementing an efficient many-to-many communication protocol over UDP**

**In order to run the program we first compile all files except [events, filepeer, sender, testClient, chatwindow] which means we should compile**

*javac*

*Address.java*

*FileReconstructor.java*

*Client2.java*

*FilePeer\_finished.java*

*Logging.java*

*FileBreaker.java*

*packets/ackPacket.java*

*packets/dataPacket.java*

*packets/infoPacket.java*

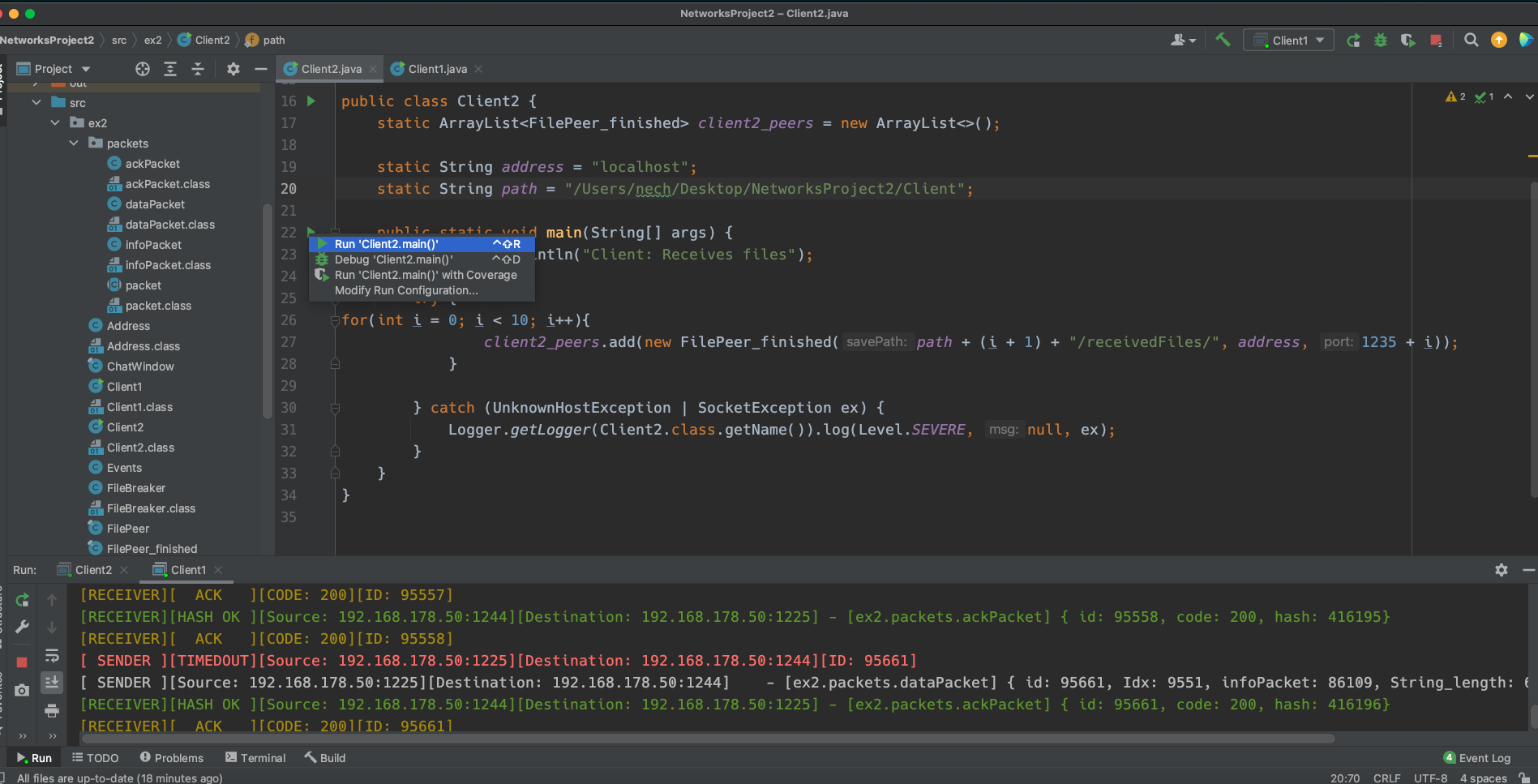
*packets/packet.java*

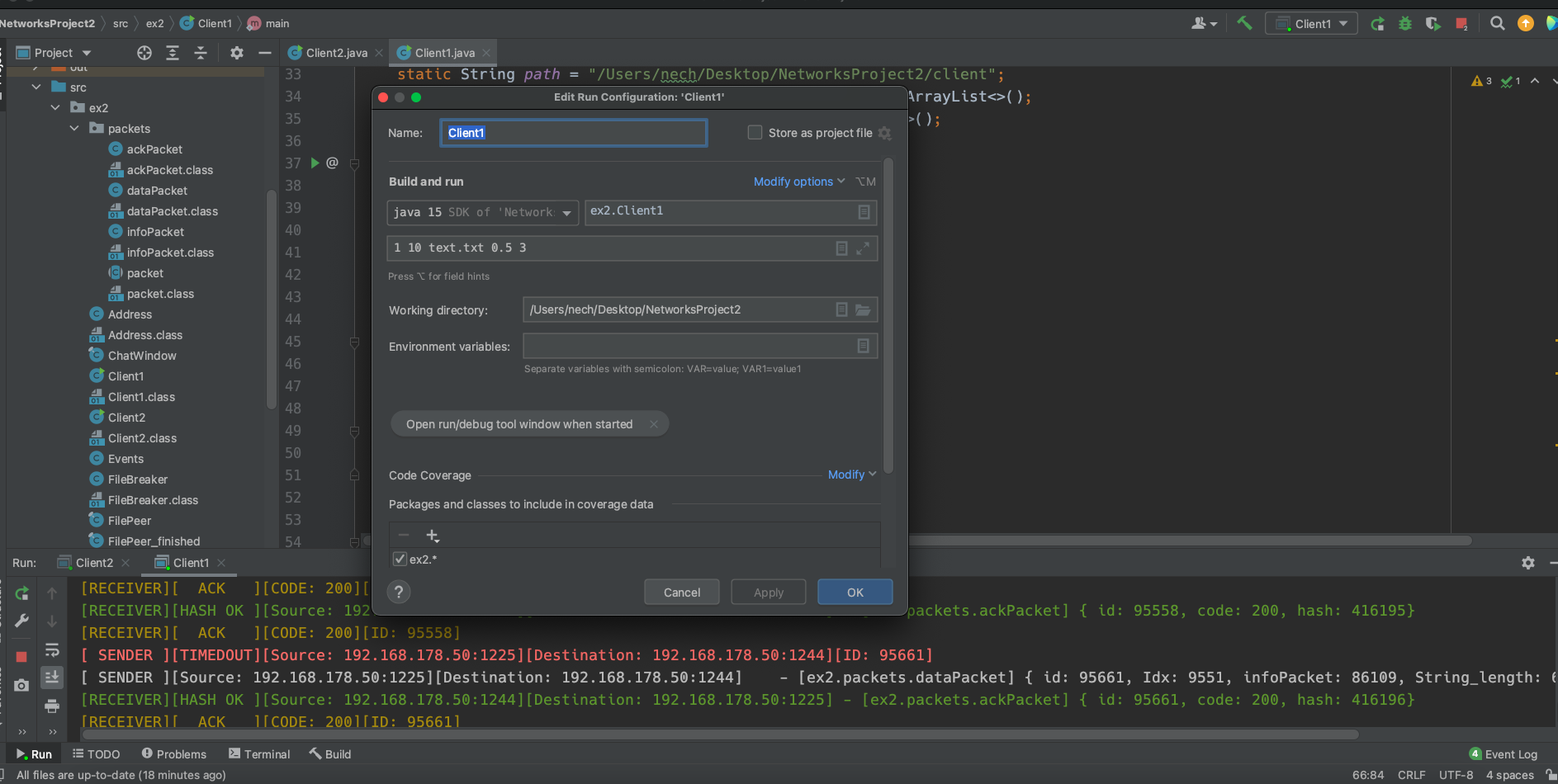
**after compile, we run client2 which represents the server the client1 which represents the client side where we should provide the following arguments [id\_process number\_of\_processes filename probability window]**

*java client2*

*java client1 1 10 1GB.bin 0.5 5*

The goal of this project is to implement an efficient protocol allowing the communication among multiple participants. You will have to implement and write a report and submit the report + code before the deadline:





The tool will be started as follows from the command line:

*toolname id\_process number\_of\_processes filename probability protocol window*

Where:

* • \_*Id\_process:* number of the current process (for instance 2)
* • \_*number\_of\_processes*: total number of processes that will join this communication (for instance 10)
* • \_*filename*: name of file to be sent
* • \_*probability*: probability of an UDP transmission not to be successful
* • \_*protocol*: Go-back-N or Selective Repeat
* • \_*window*: parameter for Go-back-N or Selective Repeat

Once started, the process will wait until all the processes have joined the session and then send its local file (provided by the filename) and receive the files sent by the other participants. You can use only UDP for your program. You will need to provide a wrapper function to the UDP send() function that ensures that the success of really sending data is (1-probability). In case failures occur, you will need to ensure that retransmissions are handled correctly. After execution, your program should have correctly received the transmitted files from all the processes and report on the display the total number of bytes/packets received, total number of bytes/packets sent and how many retransmissions (received/sent) were done.

**Your report will need to address the following topics:**

1) General architecture covering

a. Bootstrapping: this means sending/receiving starts only when all processes have joined and somehow the IP addresses of them are known to the other processes

Bootstrap Protocol (BOOTP) is a basic protocol that automatically provides each participant in a network connection with a unique IP address for identification and authentication as soon as it connects to the network.

Each network participant does not have an IP address. The network administrator then provides each host on the network with a unique IP address using the IPv4 protocol.

b. Details on how you simulated the loss and the protocols (Go-back-N and Selective Repeat)

Go-Back-N: the sending process continues to send the number of frames specified by a window size without receiving an ACK packet from the receiver. The receiver process keeps track of the sequence number of the next frame it expects to receive and sends that number with every ACK it sends. If a frame from the sender does not reach the receiver, the receiver will stop acknowledging received frames. Once the sender has sent all of the frames in its window, it will detect that all of the frames since the first lost frame are outstanding, and will go back to the sequence number of the last ACK is received from the receiver process and fill its window starting with that frame and continue the process over again.

Selective Repeat: Here, only the erroneous or lost frames are retransmitted, while the good frames are received and buffered. It uses two windows of equal size: a sending window that stores the frames to be sent and a receiving window that stores the frames received by the receiver. The size is half the maximum sequence number of the frame. For example, if the sequence number is from 0 – 15, the window size will be 8.

c. Details on how to verify that the received file has no corrupted content

To check if a PDF file is valid to load and read, we use ***com.itextpdf.text.pdf.PdfReader.***

If the file is corrupted, an exception like ***com.itextpdf.text.exceptions.InvalidPdfException,*** is thrown.

d. Details on how you can achieve savings in bandwidth and better delivery times

Latency is the time (delay) it takes for a package of data to travel from its source to its destination. It depends on the travel distance and the speed of traveling.

Bandwidth is the number of bytes that can be transferred in a unit of time, usually measured in terms of kbps (kilobits per second).

e. Details on parameters choice for Go-back-N and Selective Repeat protocols (timeouts, window size, etc.)

When we run client1file it asks to provide parameters, and one of the parameters is to choose the protocol used in the program

The main difference between these two protocols is that after finding the suspect or damage in sent frames, go-back-n protocol re-transmits all the frames whereas selective repeat protocol re-transmits only that frame that is damaged.

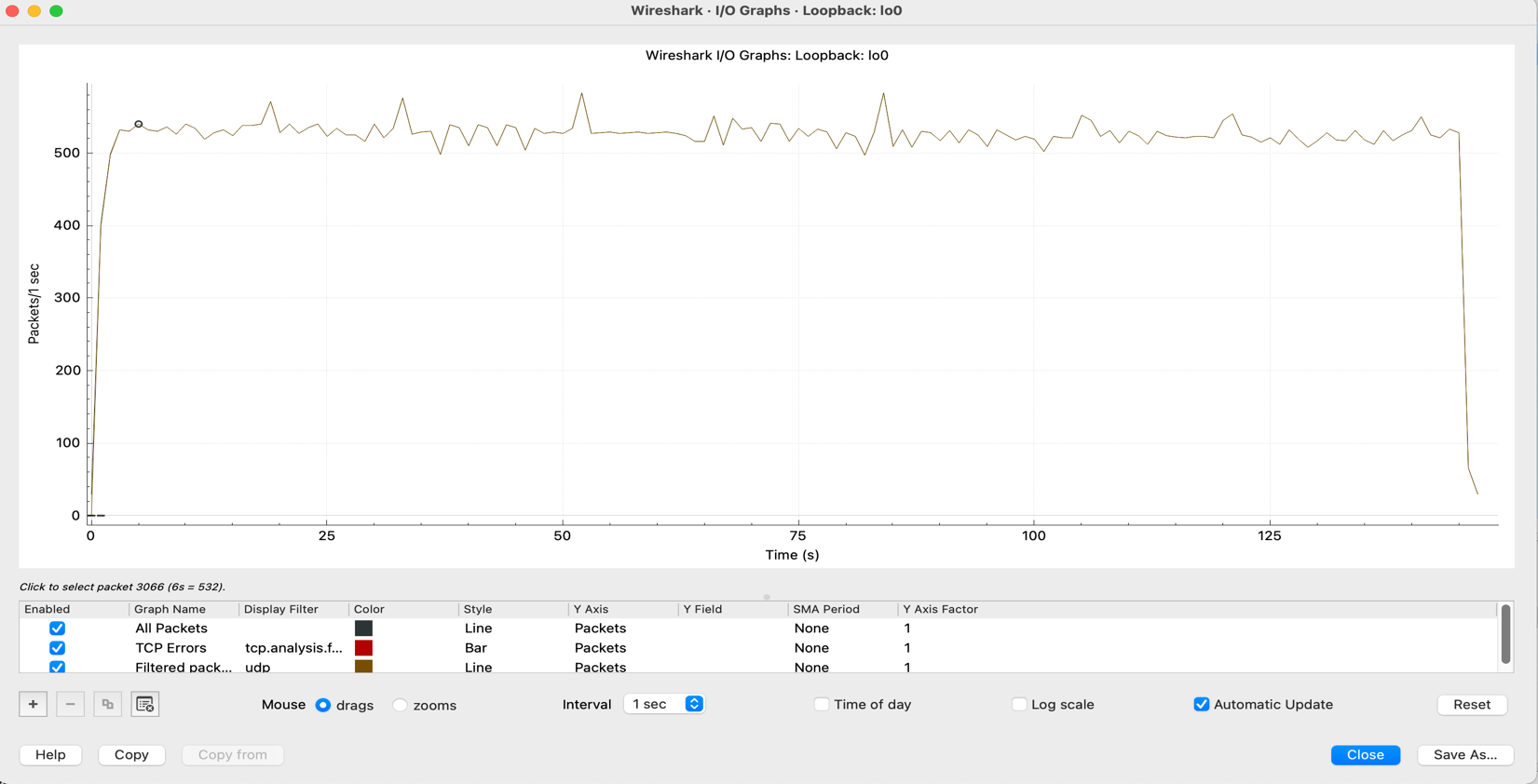
2) Performance evaluation

a. Graphs/tables showing the impact of different probabilities (0.05, 0.10, 0.15, 0.20, 0.30 and 0.40) and protocol (timeout, window size) on the total time (delay) and bandwidth.

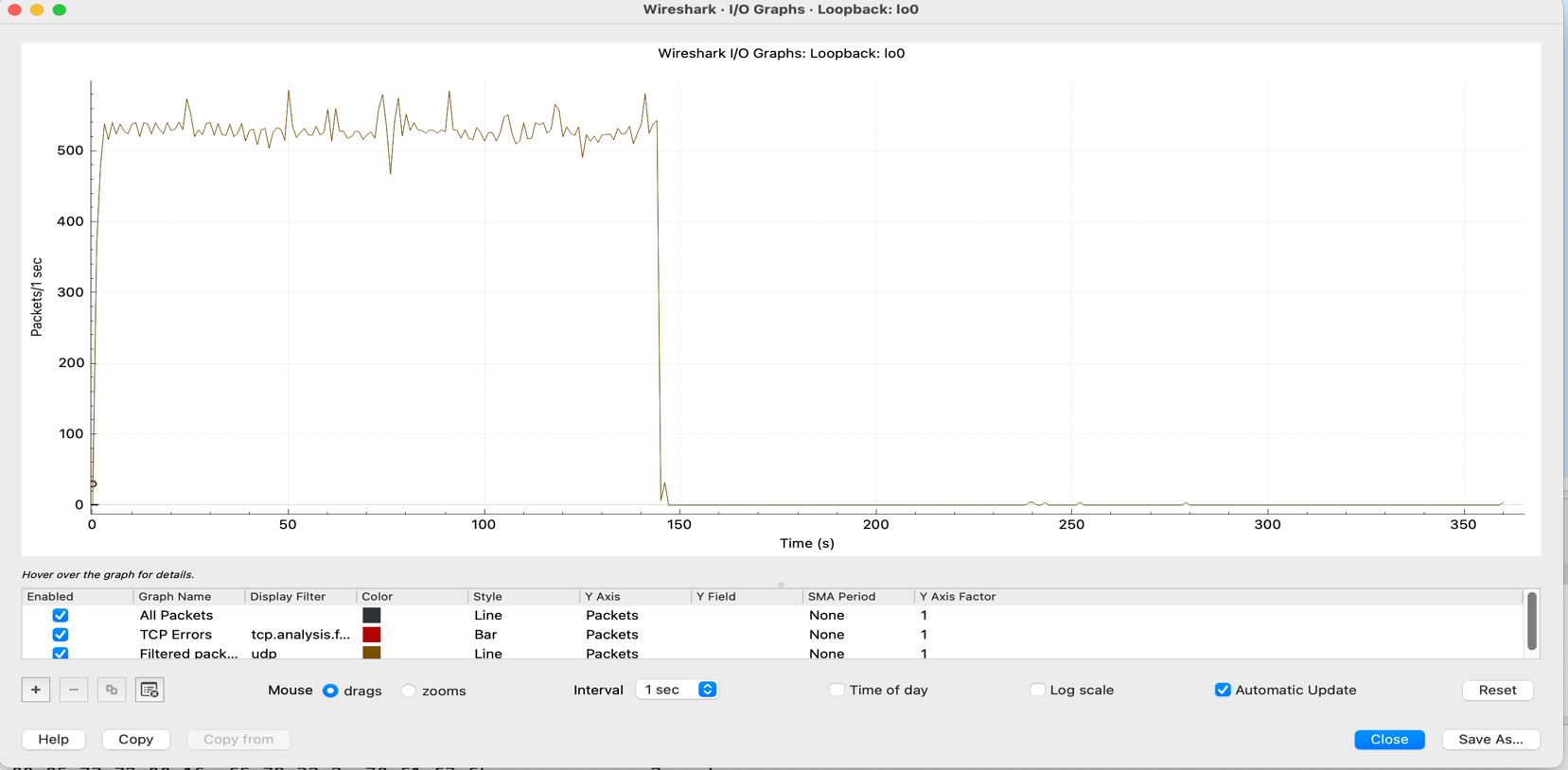
b. These graphs should cover at least 2 scenarios:

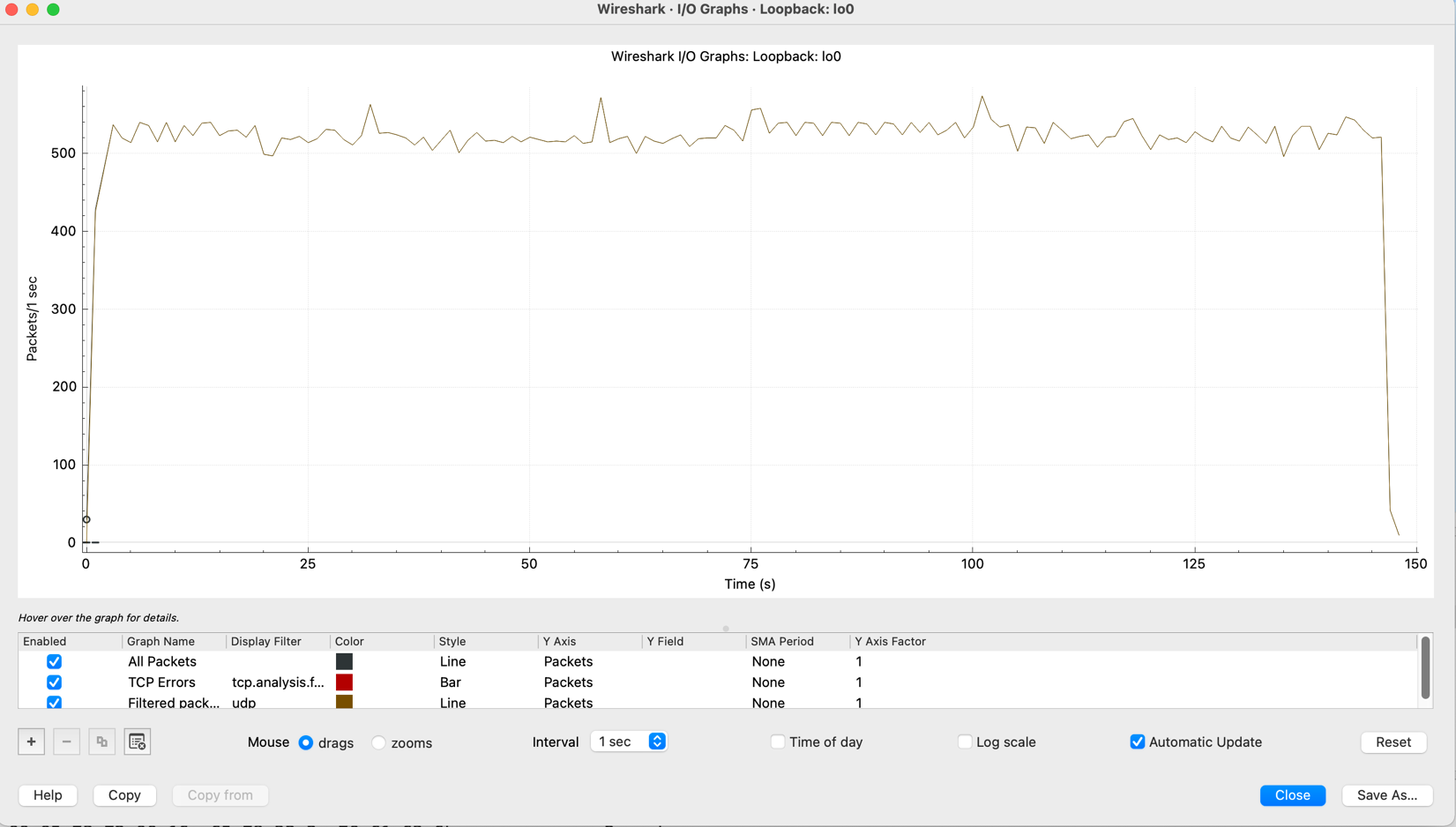
* • \_Scenario 1: two processes exchanging 1 GB files

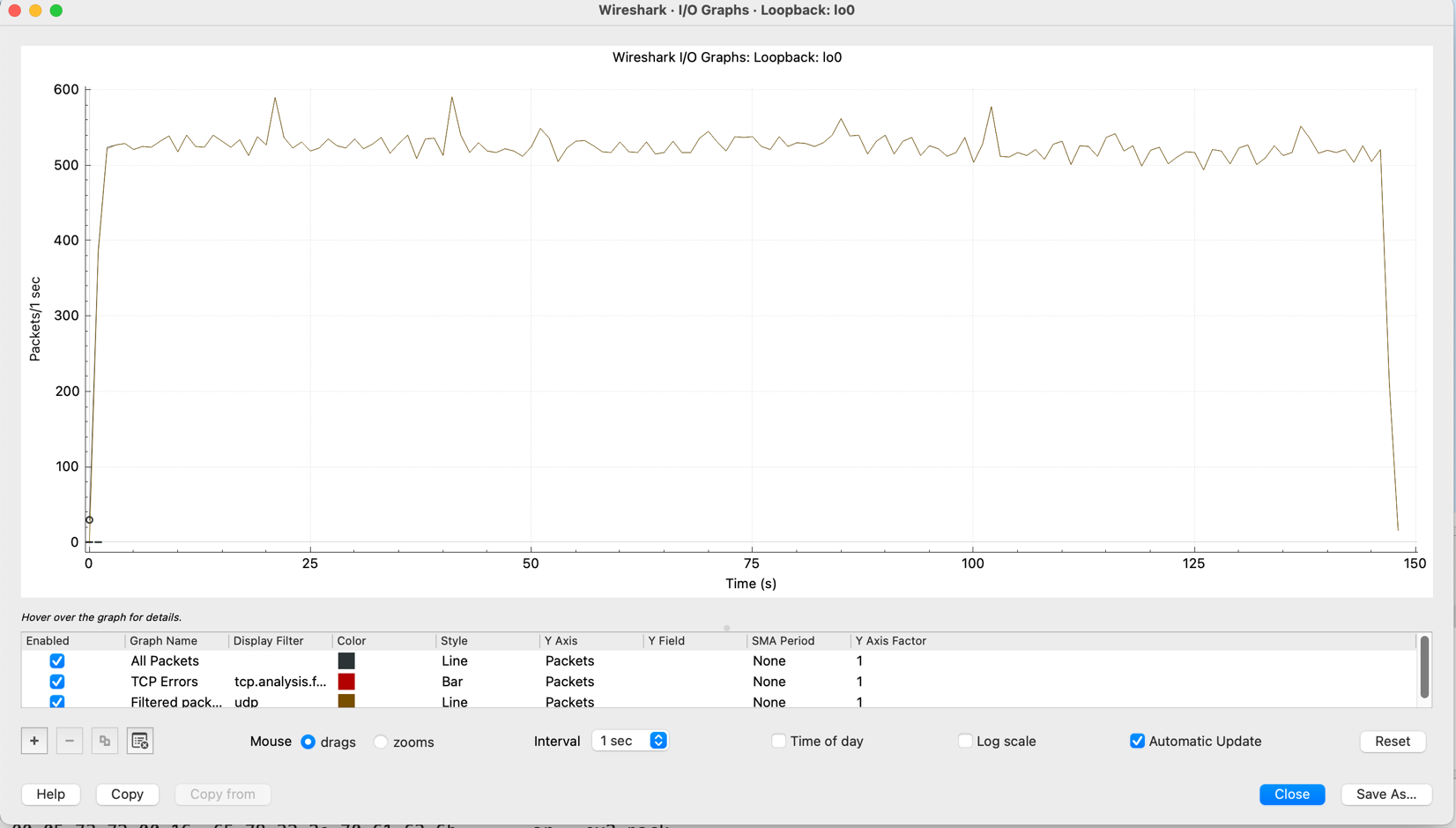
Probability = 0.05 two processes exchanging 1 GB files



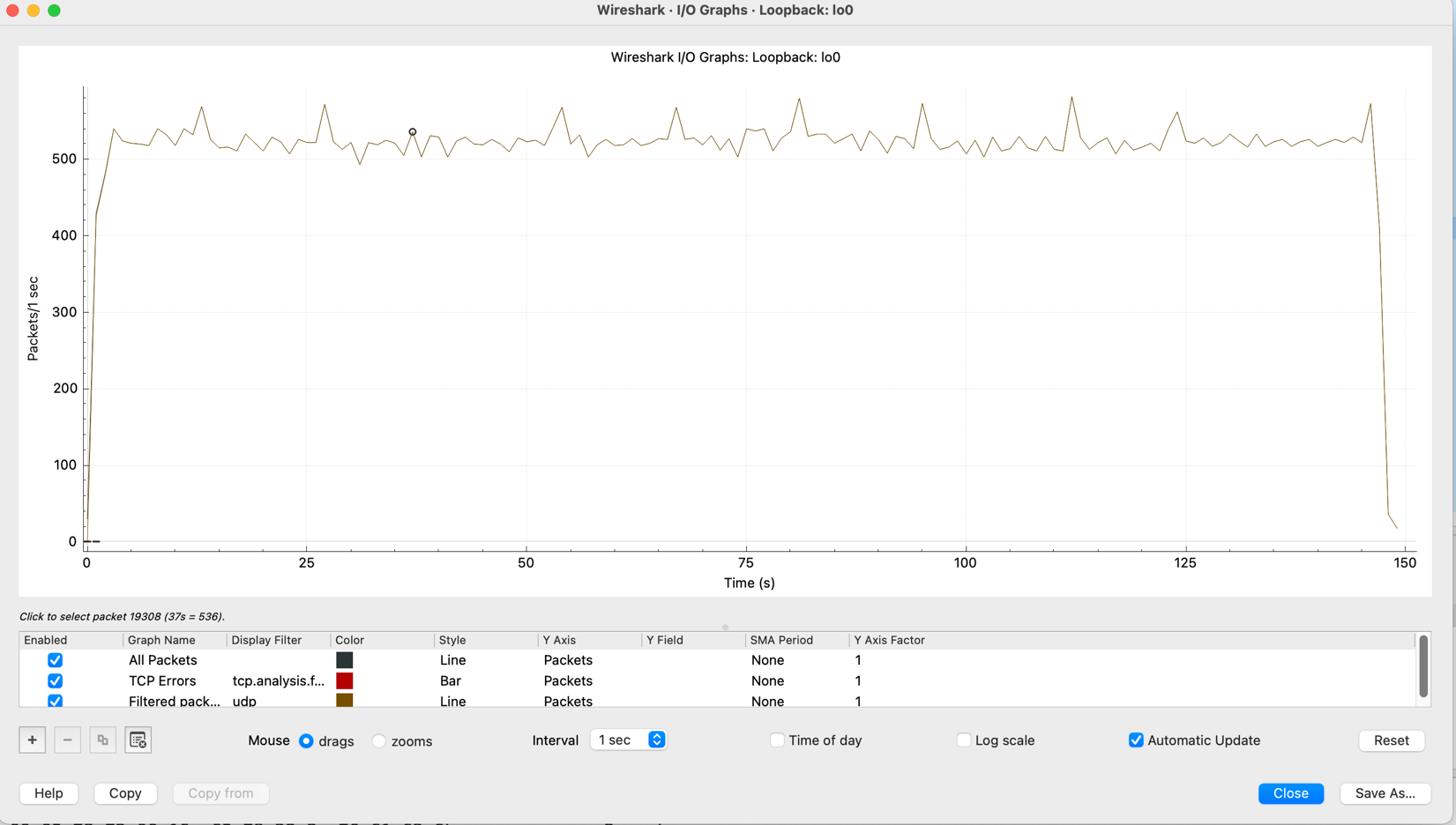
Probability = 0.10 two processes exchanging 1 GB files



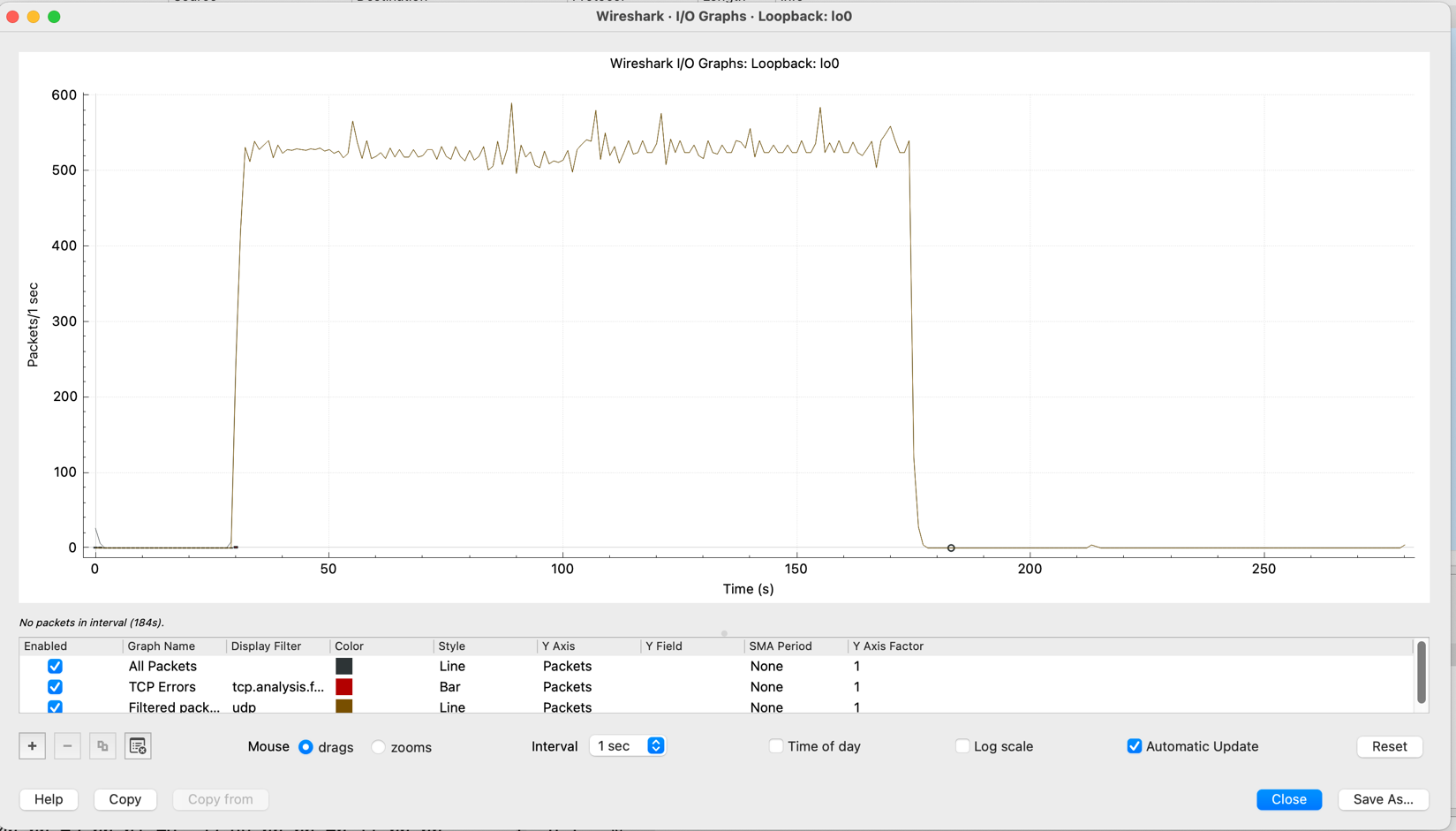
Probability = 0.15 two processes exchanging 1 GB files 

Probability = 0.20 two processes exchanging 1 GB files 

Probability = 0.30 two processes exchanging 1 GB files

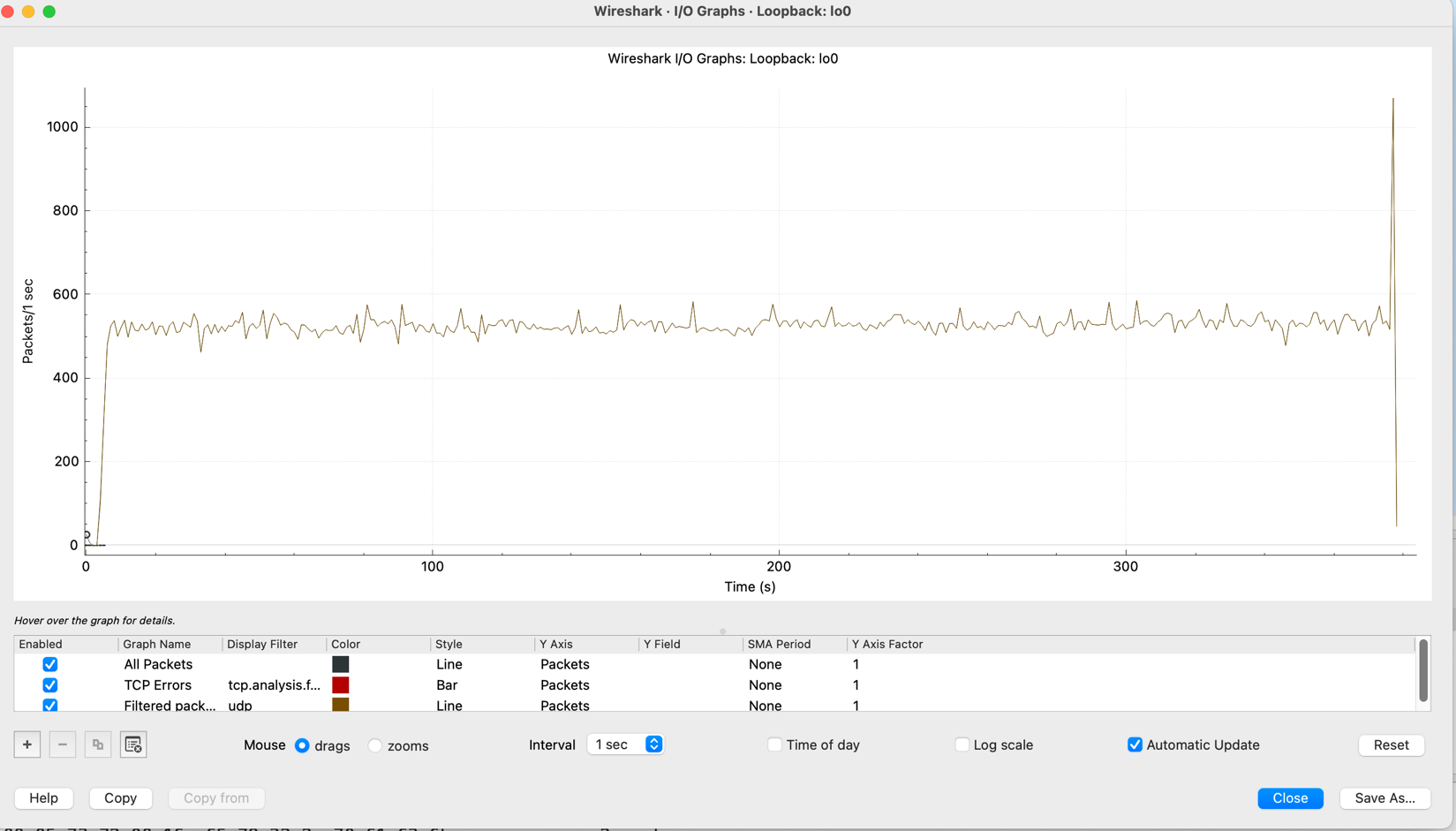


Probability = 0.40 two processes exchanging 1 GB files



* • \_Scenario 2: 10 processes exchanging 10 files, et least 500MB per file.

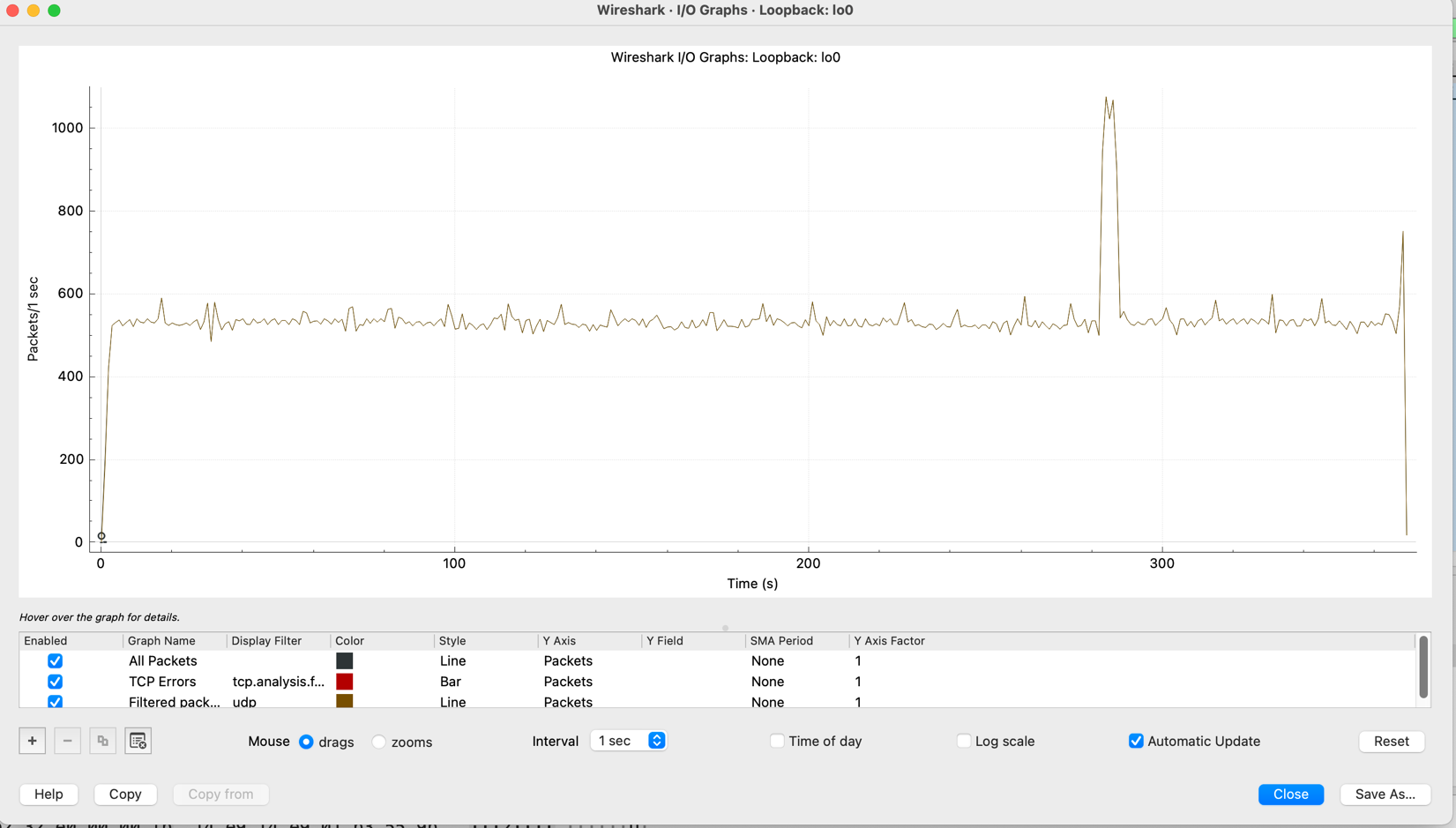
10 processes exchanging 10 files 587MB with Probability 0.05



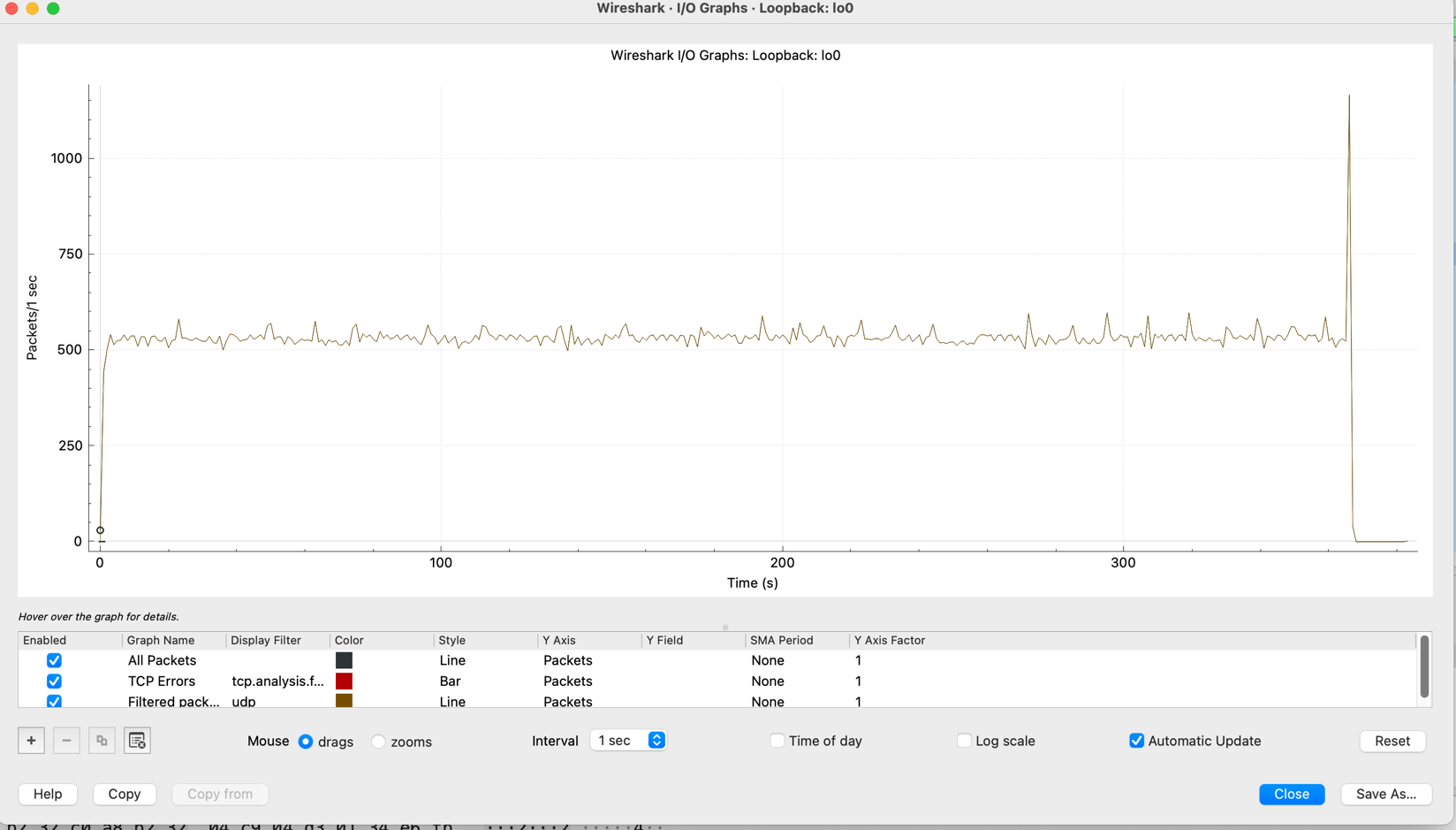
10 processes exchanging 10 files 587MB with Probability 0.10,



10 processes exchanging 10 files 587MB with Probability 0.15,



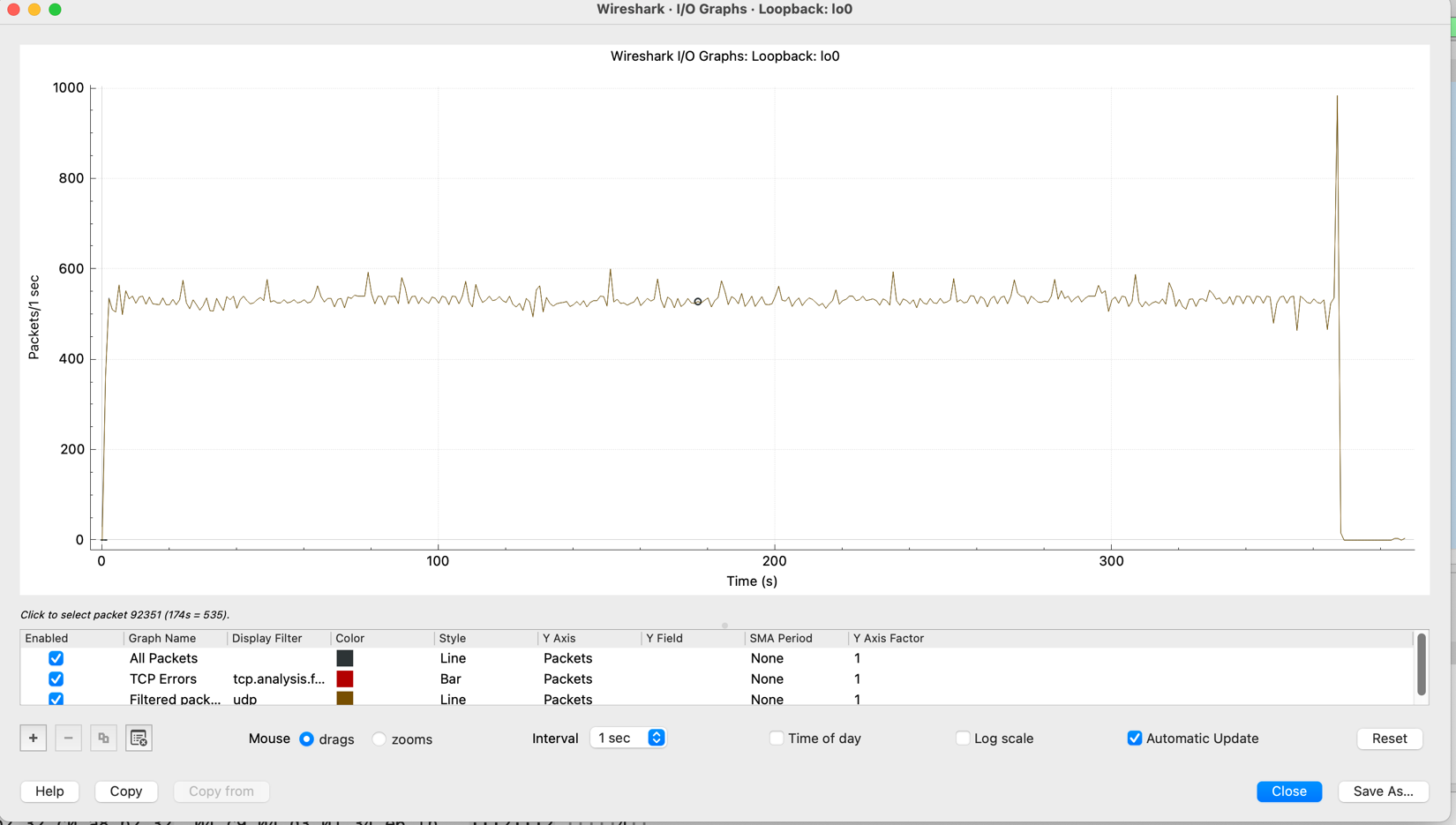
10 processes exchanging 10 files 587MB with Probability 0.20,



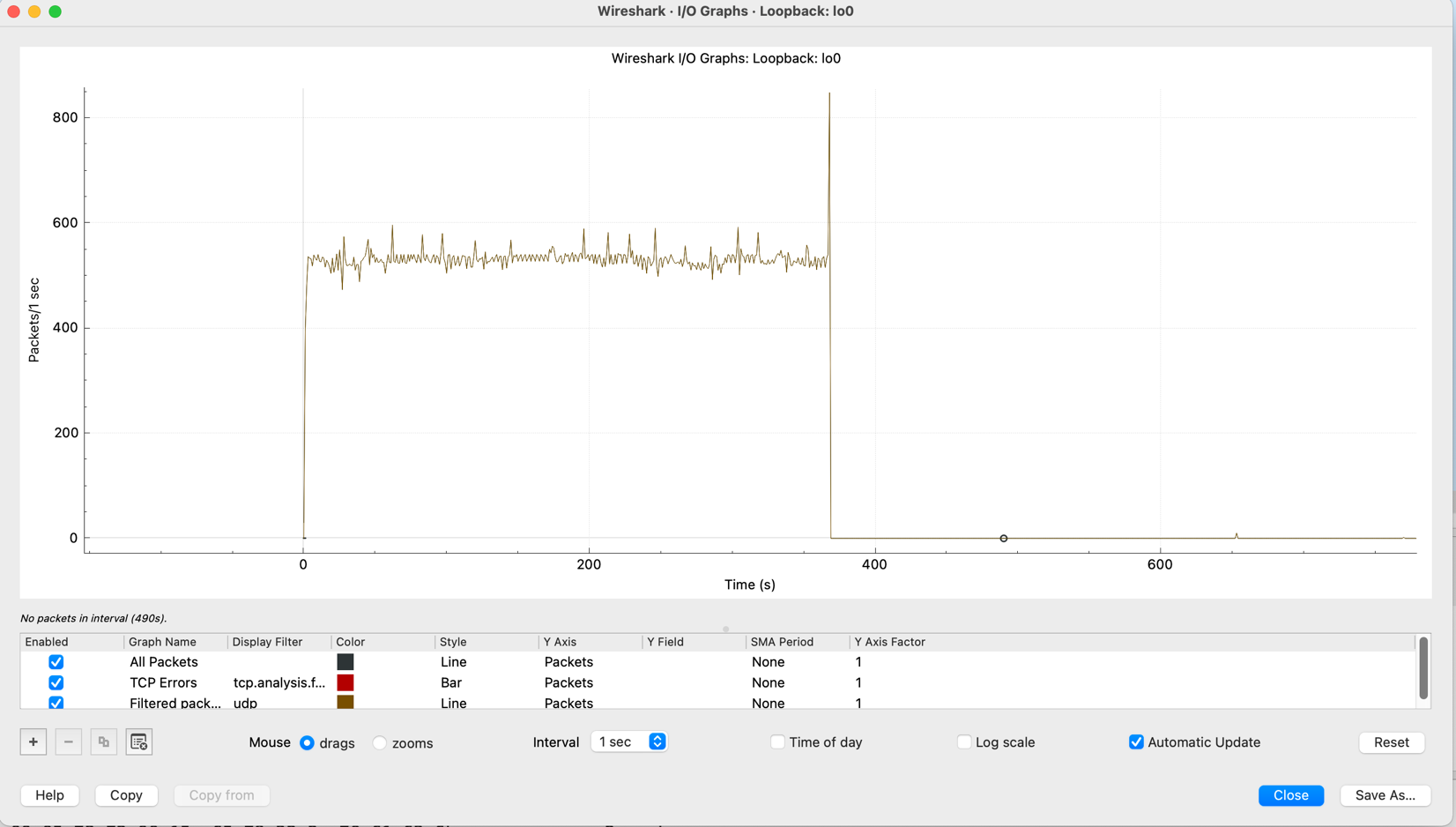
10 processes exchanging 10 files 587MB with Probability 0.30



10 processes exchanging 10 files 587MB with Probability 0.40



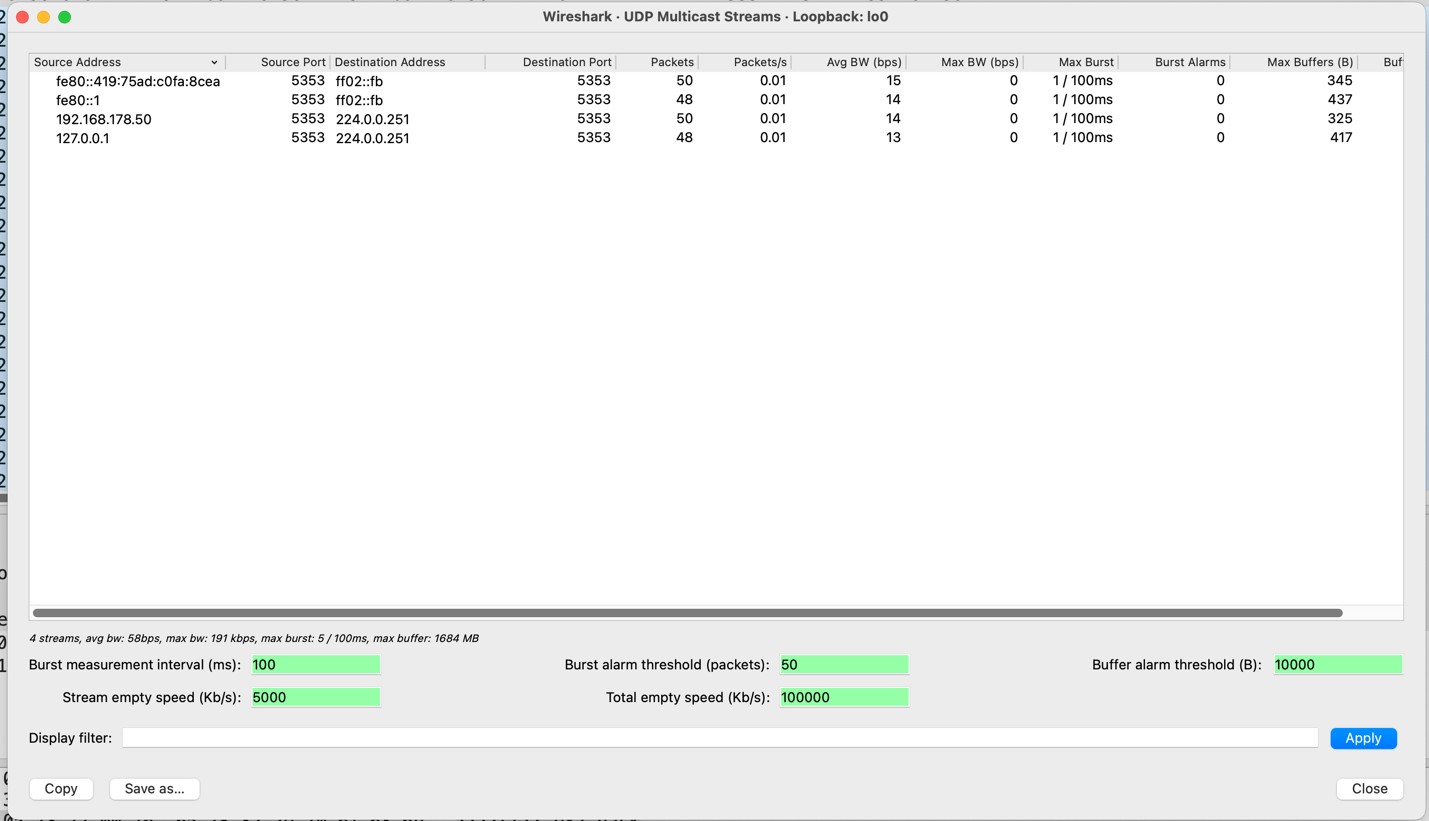
10 processes exchanging 10 files 587MB with Probability 0.90,

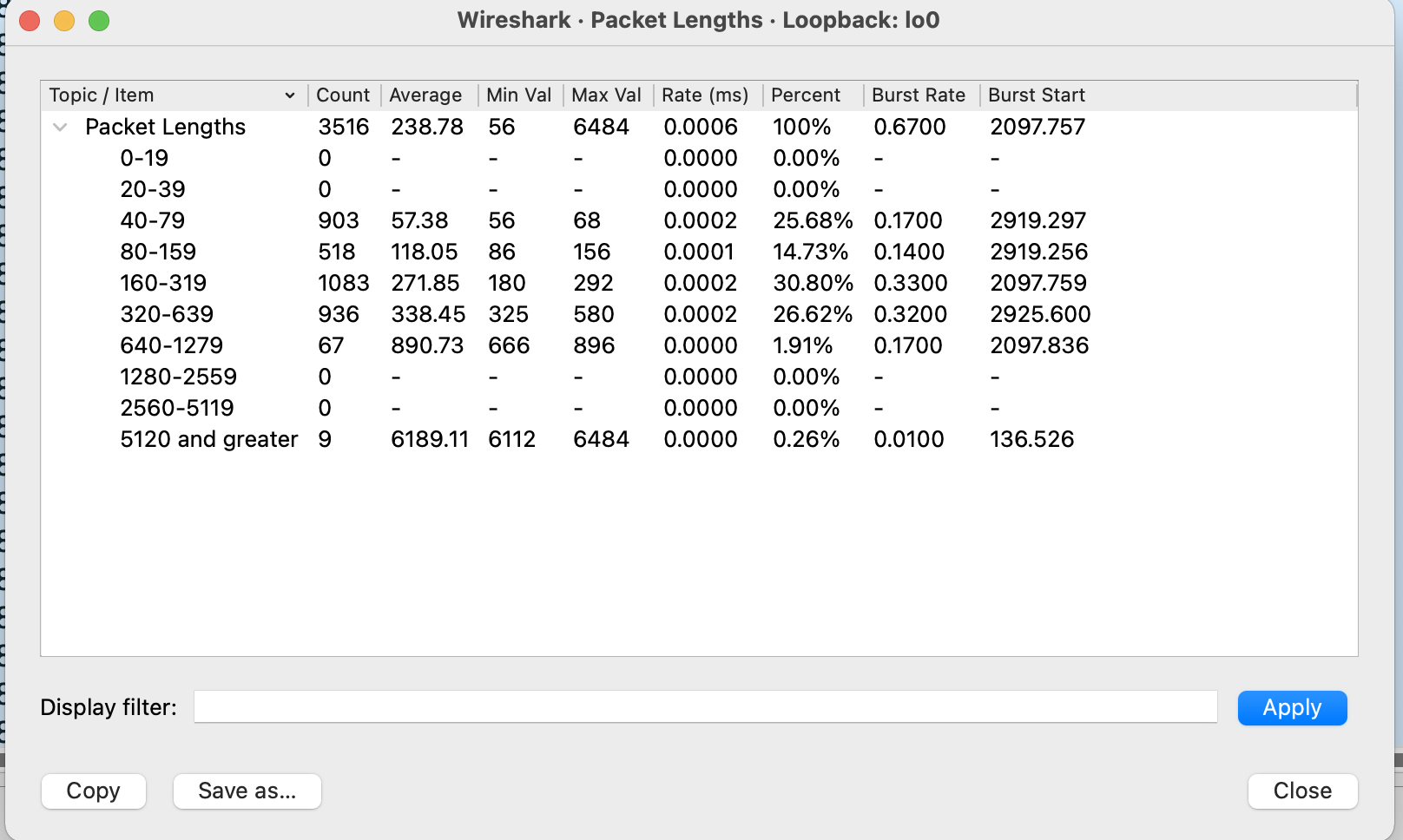


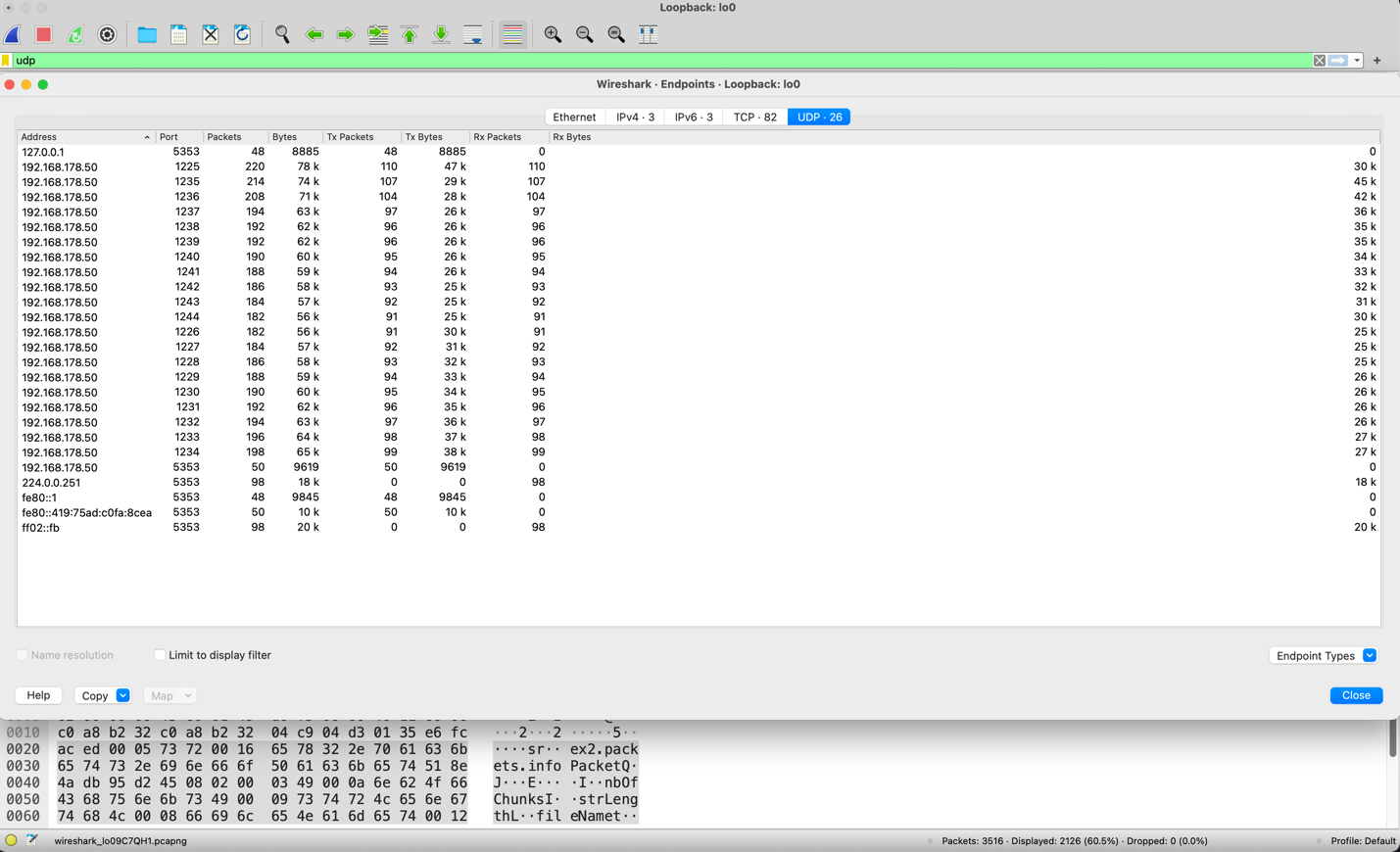
3) Wireshark screenshots

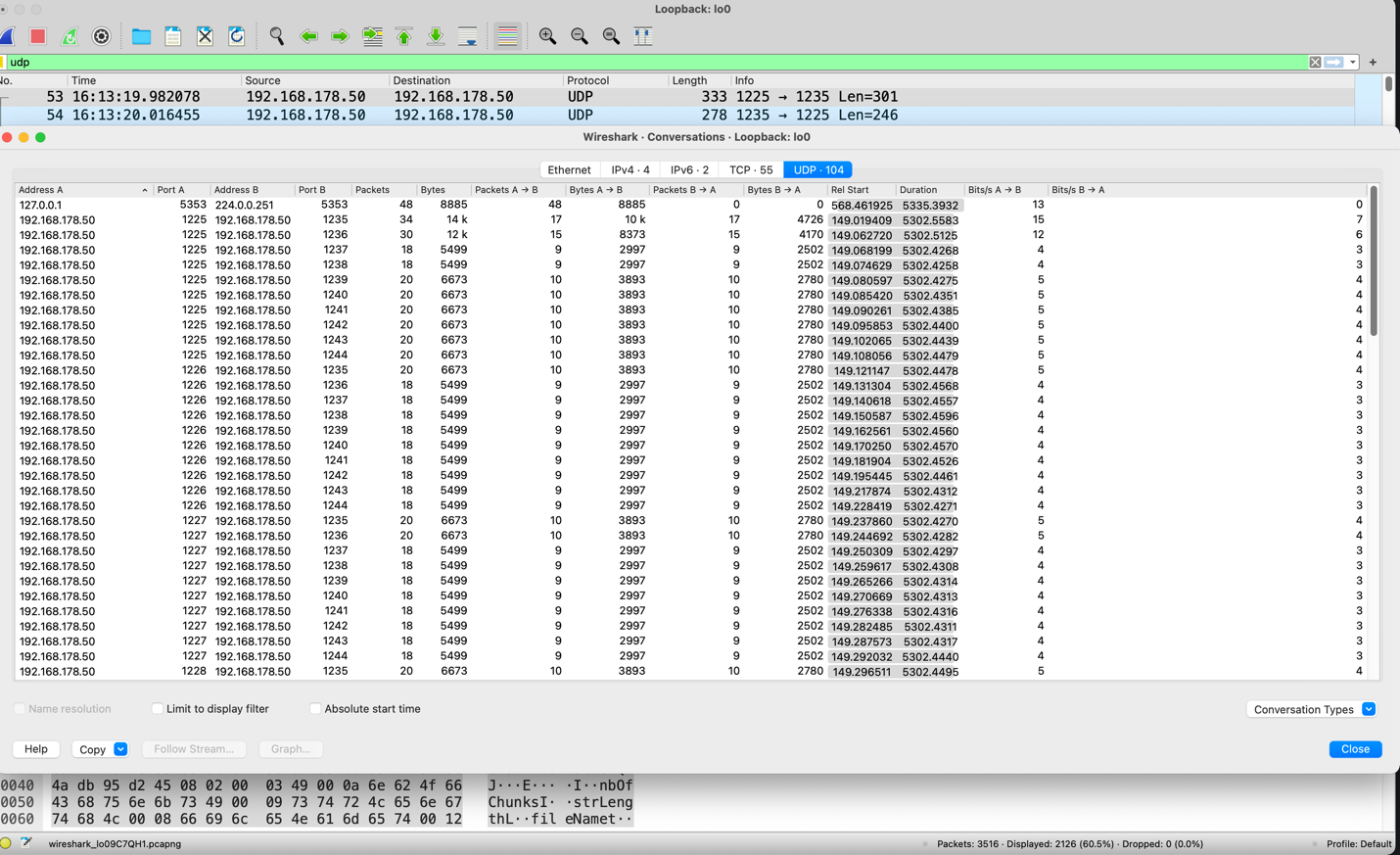
a. Show statistics, connection/other display options only for the selected communication. This should include statistics about #bytes/#packets, endpoints, etc.

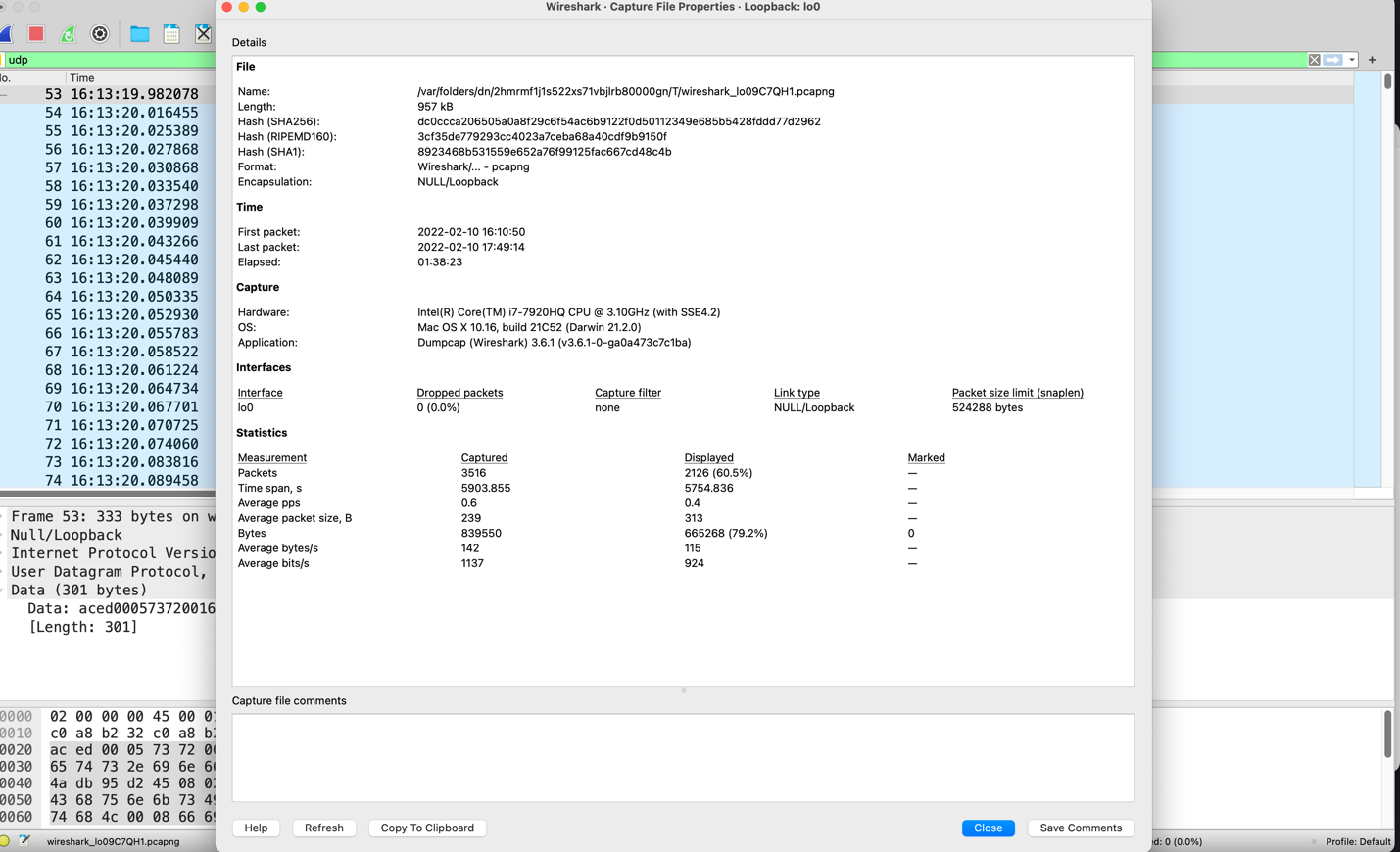
Here we have 10 nodes, each node broadcasts its local file to all other nodes (each file around 563 bytes)

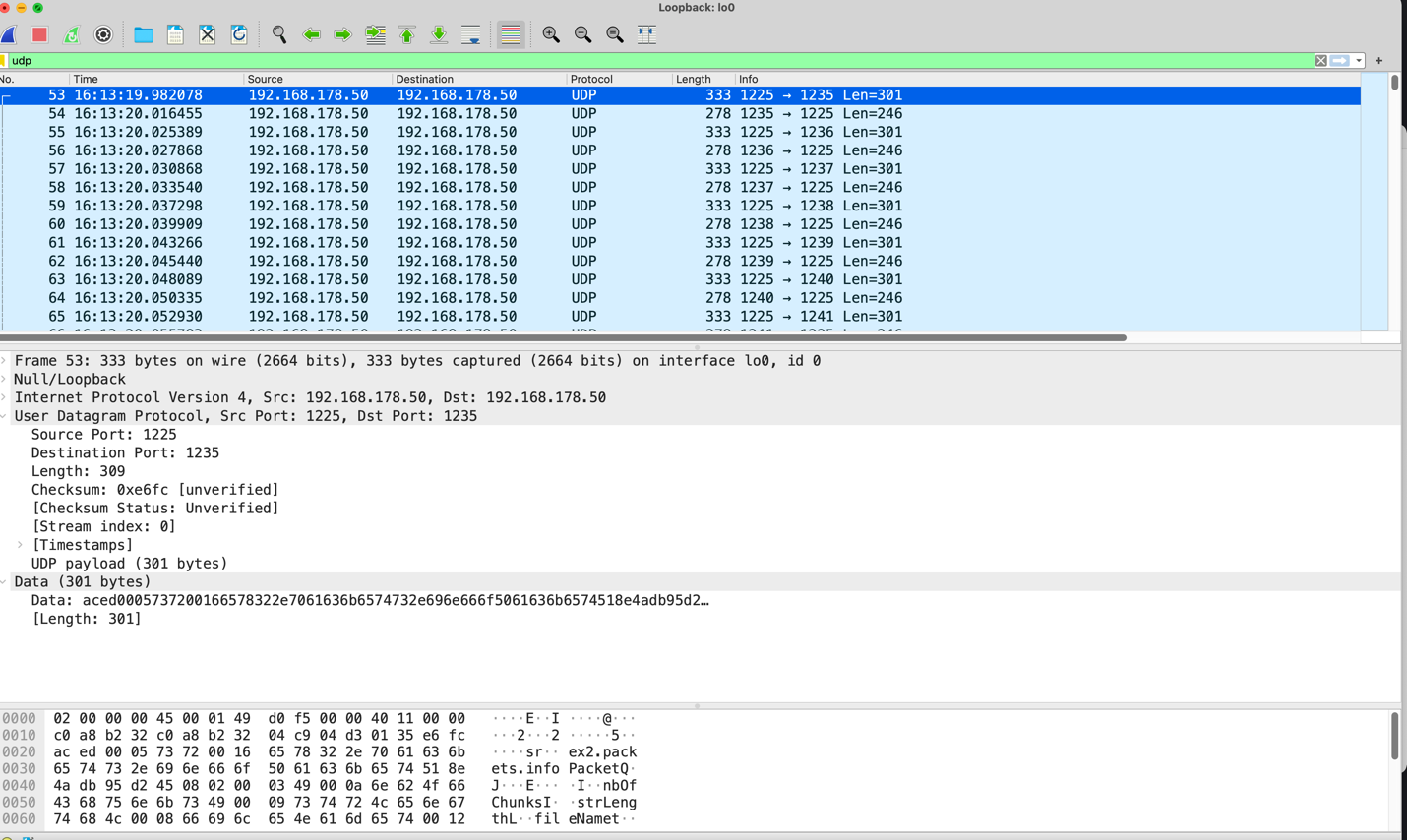


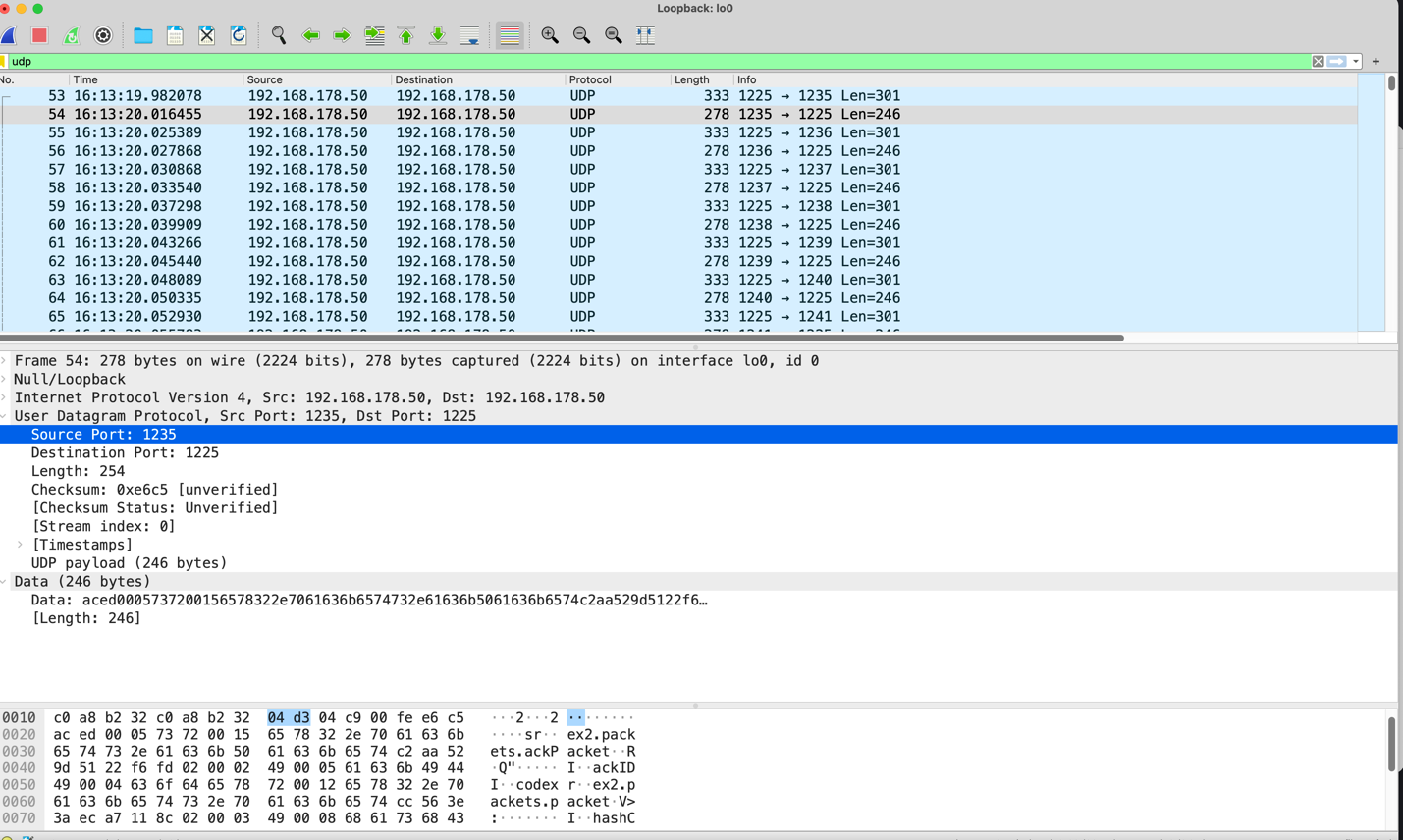


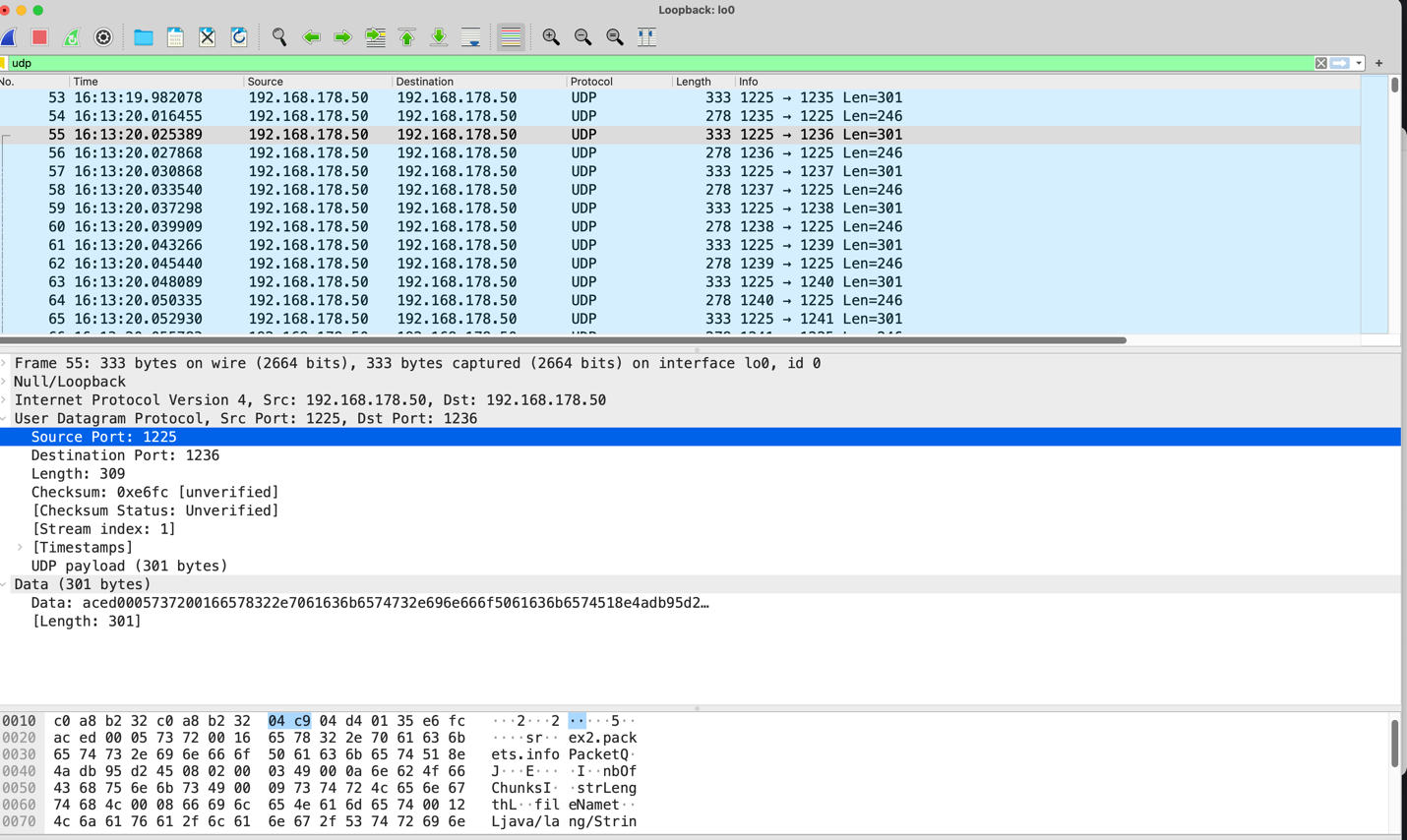


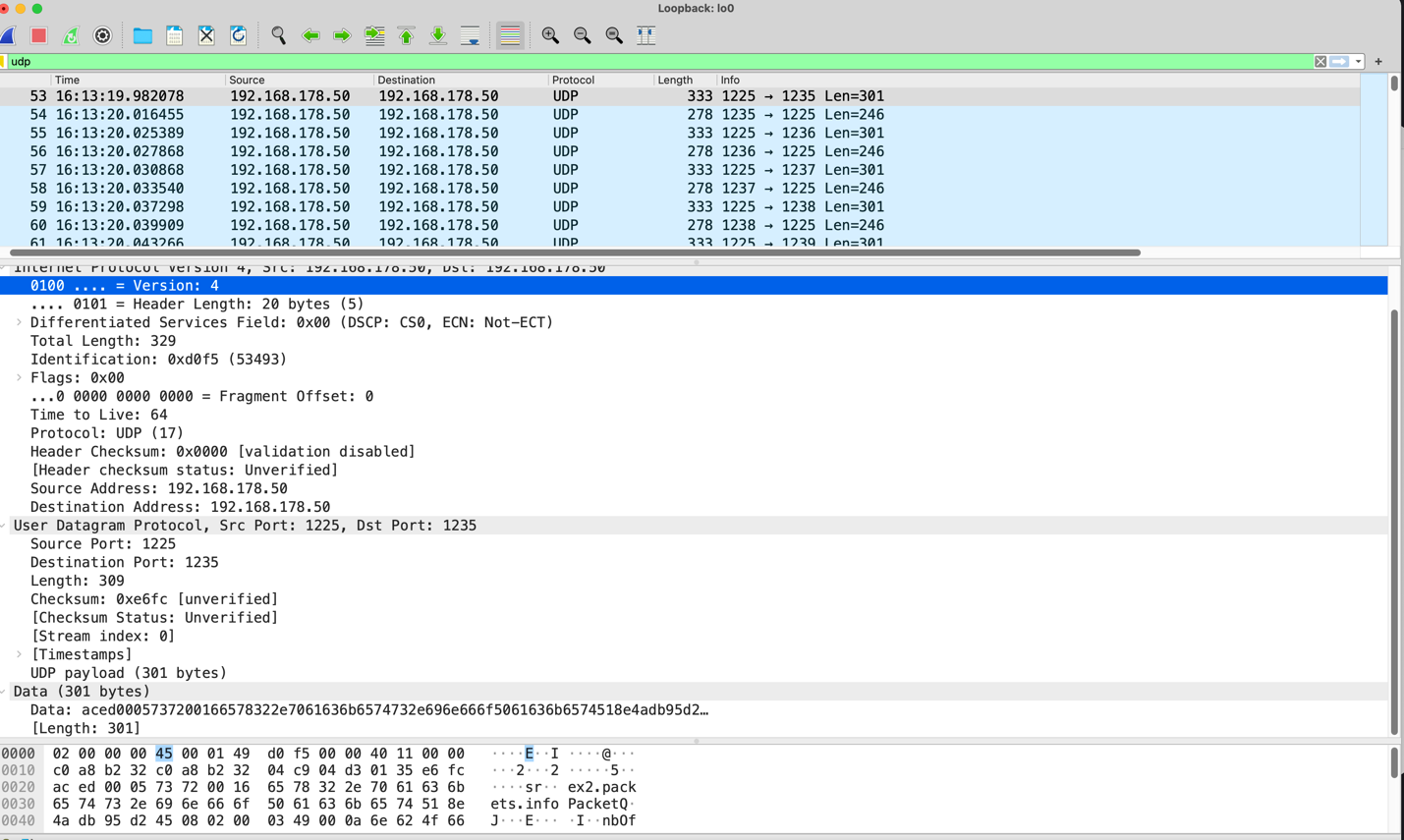












Important remark: All the material that you use and was not developed by you (code, text, etc) needs to be quoted and cited correctly in your report.

RESOURCES:

Bootstrapping: <https://www.investopedia.com/terms/b/bootstrapping.asp>

GoBackN: <https://www.geeksforgeeks.org/bootstrap-protocol-bootp/>

Selective Repeat: <https://www.geeksforgeeks.org/sliding-window-protocol-set-3-selective-repeat/>

Book: Learning Network Programming with Java by Richard M Reese