|  |
| --- |
| Report |

Assignment 1

*1DV701*

|  |
| --- |
| *Author:* LEONARDO PEDRO  *Semester:* Spring 2020  *Email lr222qp@student.lnu.se* |

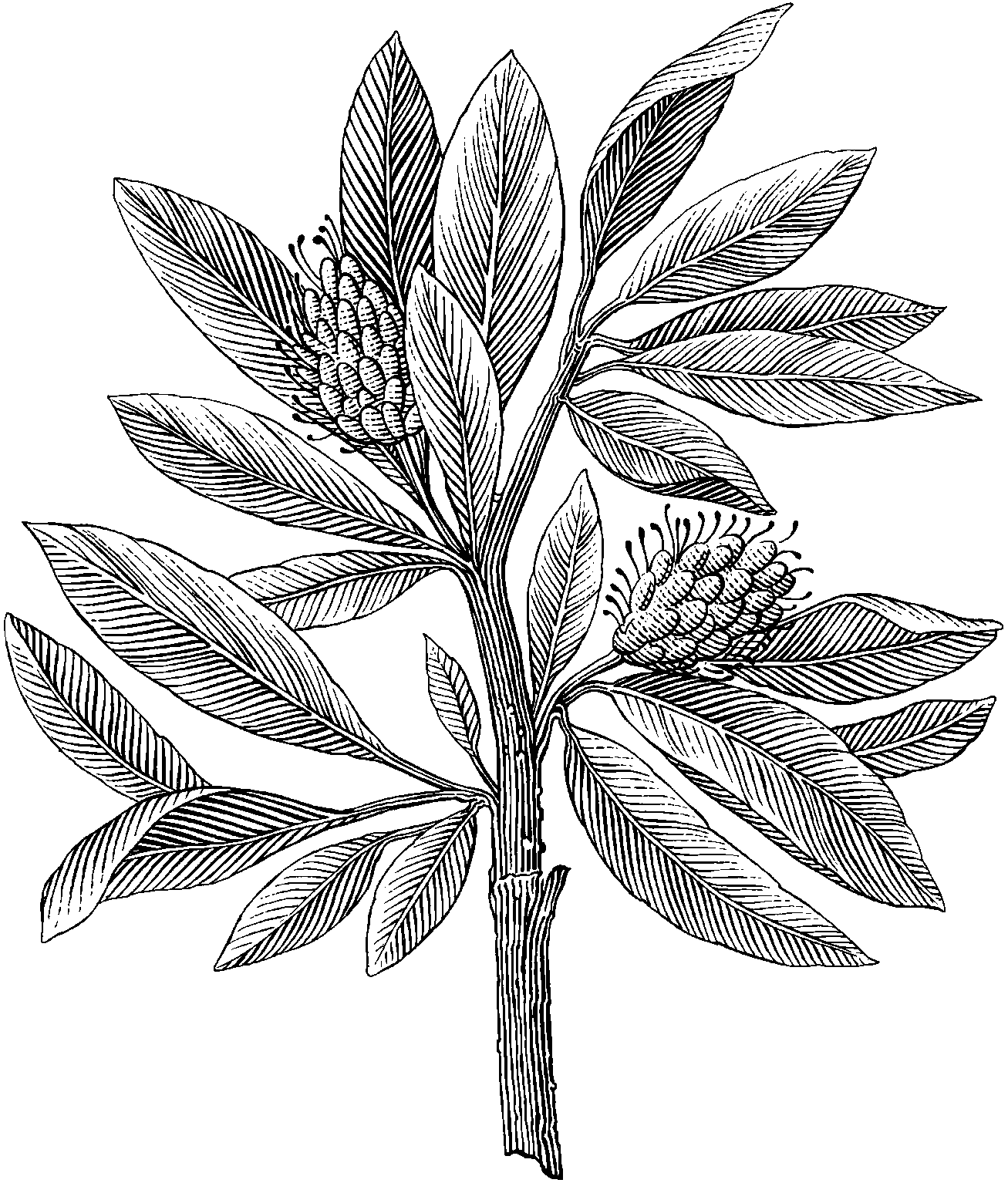


Table of Contents

**[Problem 1](#_3znysh7) 2**

[Discussion](#_2et92p0) 2

**[Problem 2](#_tyjcwt) 2**

[Discussion](#_3dy6vkm) 2

*2*

**[Problem 3](#_3rdcrjn) 2**

[Discussion](#_26in1rg) 2

**[Problem 4](#_lnxbz9) 2**

[Discussion](#_mofdz88n8hh8) 2

**[Problem 5](#_c9l9ef6unch1) 2**

[Discussion](#_wnl2oddmgft6) 2

# Problem 1

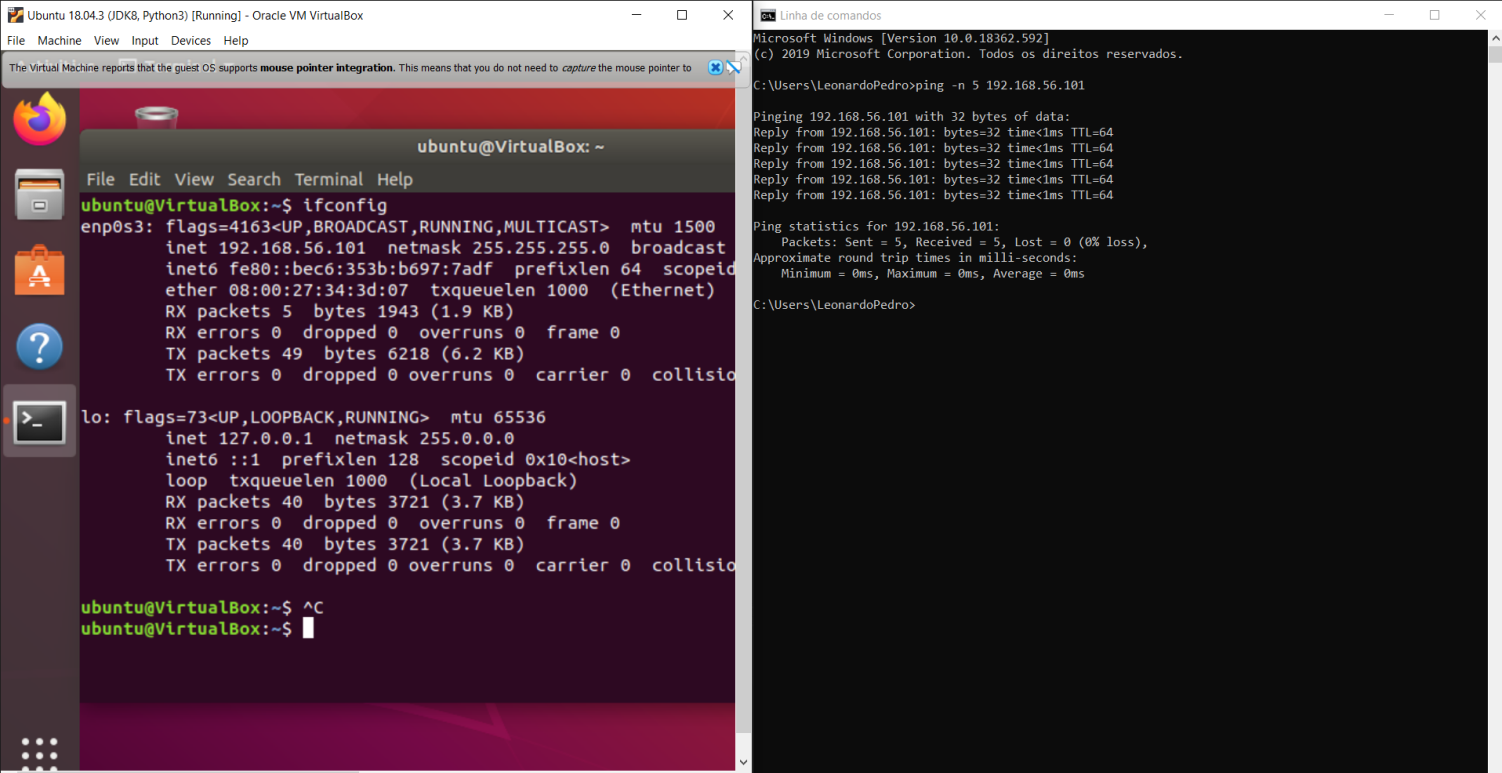


Figure 1. Virtual Machine Terminal and Windows cmd

## Discussion

The screenshot shows us that my network works with a ping of 5 from the host to the virtual machine.

# UDP_screenshotProblem 2

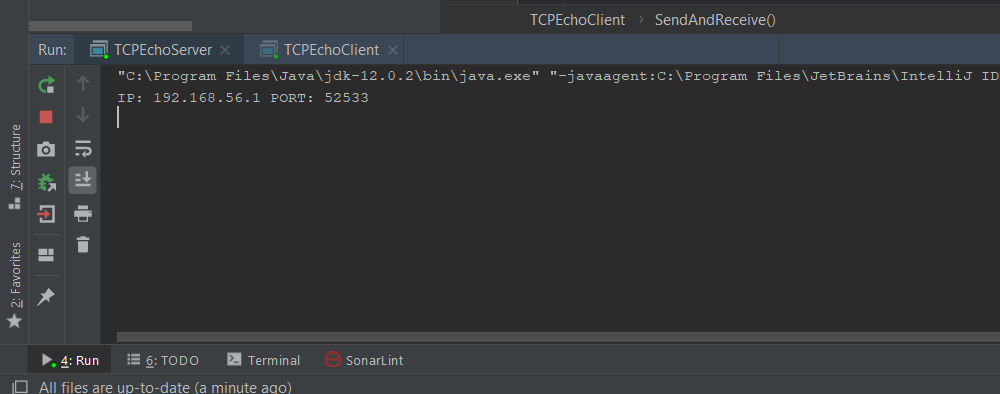
Figure 2. UDP Server (VM) and UDP Client (Host)

## Discussion

The screenshot above shows how much information was sent and received between the UDP server being run in the virtual machine and the Client hosted on the local device. To test the connection the input format is IP, PORT, BUFFERSIZE, MESSAGERATE, where the message rate is how many times it should exchange information between them. To make sure the program runs properly when trying it please take in mind the input specifications and all the following handled exceptions/errors :

1. When the number of arguments is not equal to 4;
2. When the IP is uncompleted;
3. When PORT number is out of range (0 - 255);
4. When the Buffer Size is out of range (0-1024);
5. When any of the parameters are out of order from the desired format.

# Problem 3



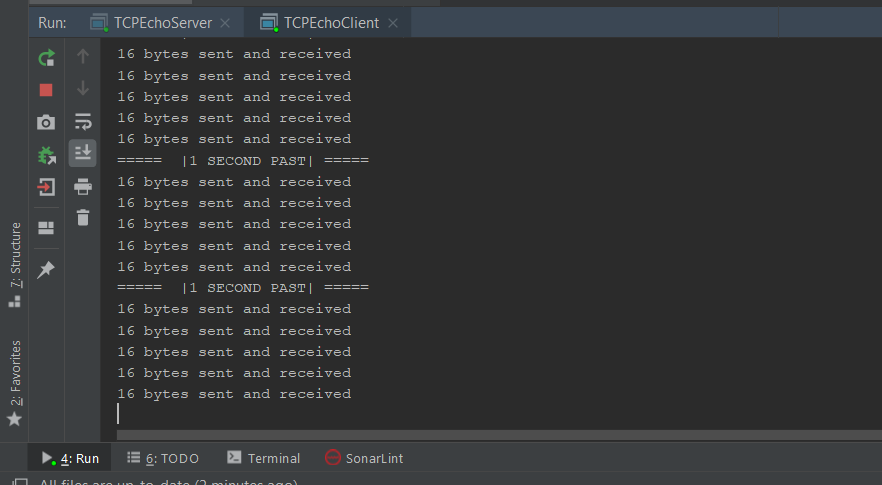


Figure 3. TCP Server and TCP Client

## TCP_2Clients

Figure 3. TCP Server and 2 TCP Clients

## Discussion

On the first two figures we can see the Client request to the TCP Server and the amount of data that is being transferred between server and client.

In figure 3 we use two Clients and as it shows both of them sent a request to the server and were connected successfully.

# Problem 4

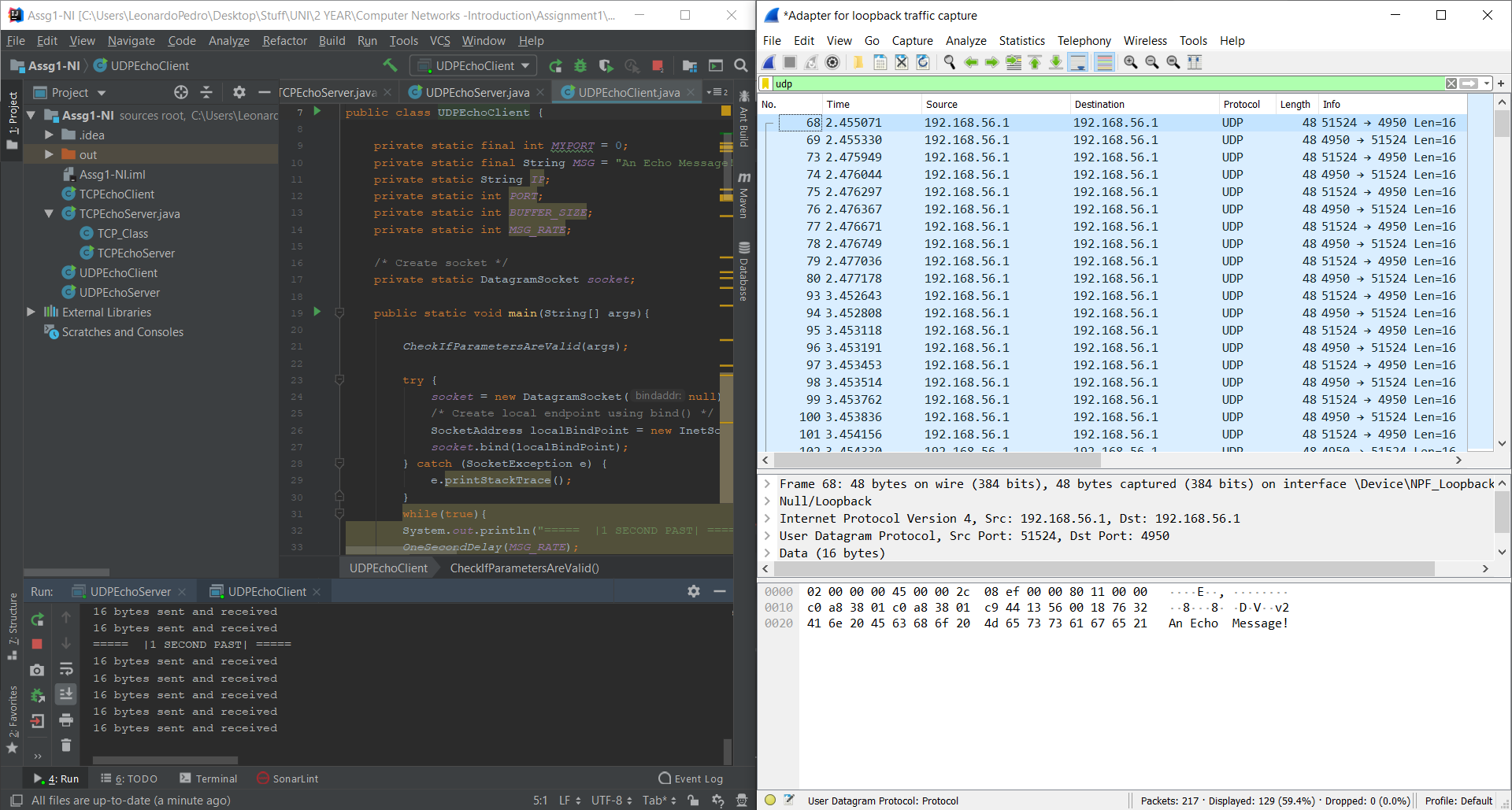
****

Figure 4. UDP traffic using Wire Shark

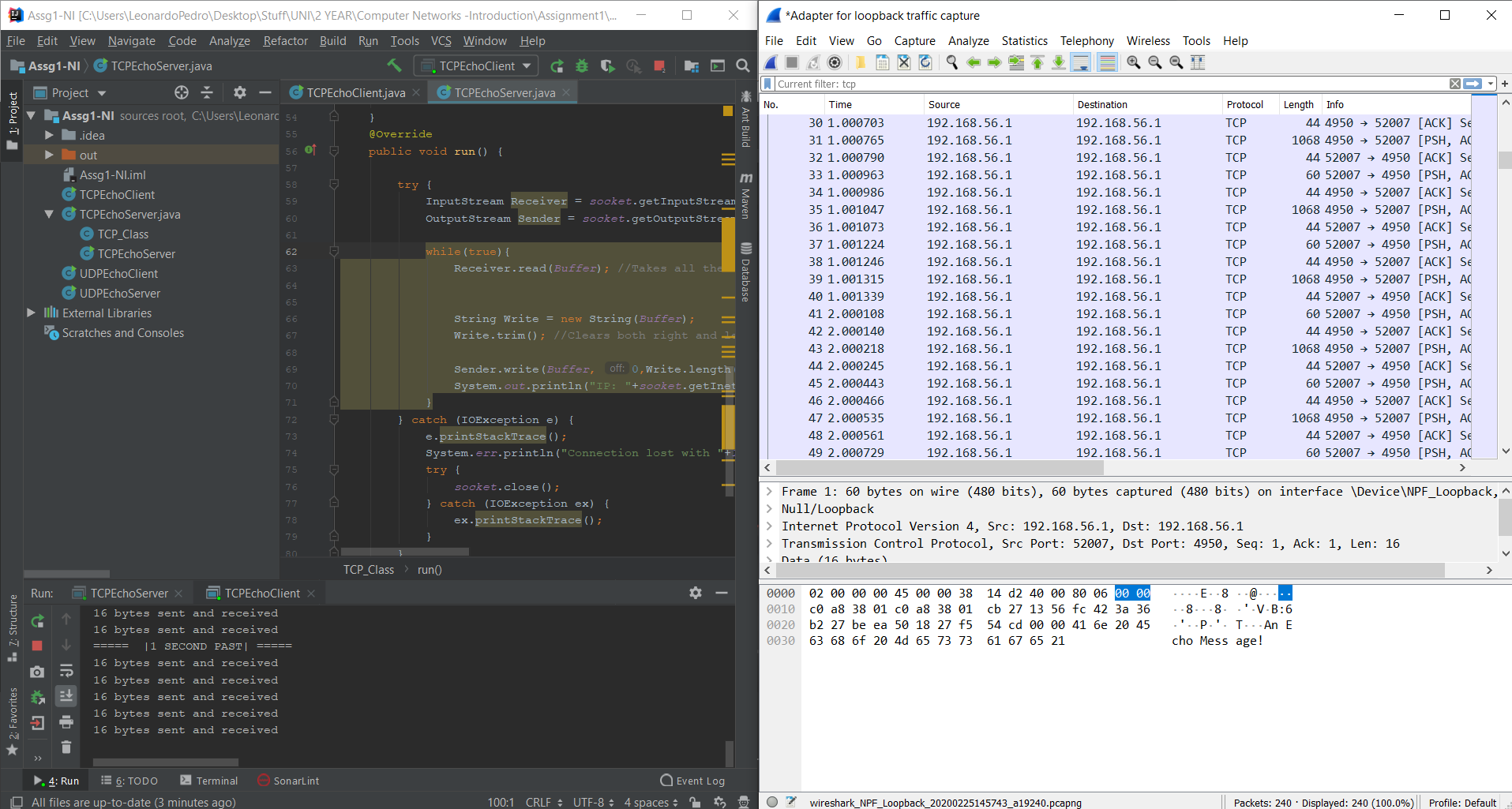
****

Figure 5. TCP traffic using Wire Shark

## Discussion

**SYN -** The SYN flag, which stands for “Synchronize”, its used to establish communication between two systems over the TCP/IP protocol, it is the first step in a 3-way handshake between two hosts where the first packet from both the sender and receiver should have this flag set and when the server receives a SYN request, it responds with SYN-ACK message.

**ACK -** The ACK flag, which stands for “Acknowledgment”, is used to acknowledge the successful receipt of a packet.The receiver sends an ACK as well as a SYN in the second step of the 3-way handshake process to tell the sender that it received its initial packet.

In TCP once the connection is established, all packets will contain an ACK to make sure all the sent data from the other machine is acknowledged or receipt of message as part of the communication protocol.

**PSH -**  The PSH flag, which stands for “Push”, tells the receiver to process the packets as they are received instead of buffering them. PSH flag is an option provided TCP that allows the sending application to start sending data even when the buffer is not full and when its received with PSH set, it will immediately transfer the received data to the application.

**UDP**

**Wire Shark Analysis**

1. The UDP Echo Client sends a message to the UDP Server with 16 length and shows the received 48 bytes that corresponds to 32 bytes from header and the rest 16 bytes contains the message.
2. UDP Server sends the Echo message back to the client.
3. UDP Client sends the same message back to the server.

**TCP**

**Wire Shark Analysis**

1. Despite of not being in the image, the first thing wire shark captures when analyzing the traffic between TCP server and client is a Synchronize message from the client to the server with a sequence number of 0.
2. After the first synchronization message the server replies with a synchronize/acknowledgment (SYN/ACK) message with the value of Seq= 0 and ACK = 1.
3. The client replies with an acknowledgment (ACK) message with ACK = 1 and Seq= 1 establishing the connection between them. (Server - Client)
4. The client receives the message and sends back the ACK
5. The client sends to the server a message with PSH flag and ACK
6. The server sends to the client with PSH flag and ACK set

7. The client receives the message and sends back the same ACK

# 

# Problem 5

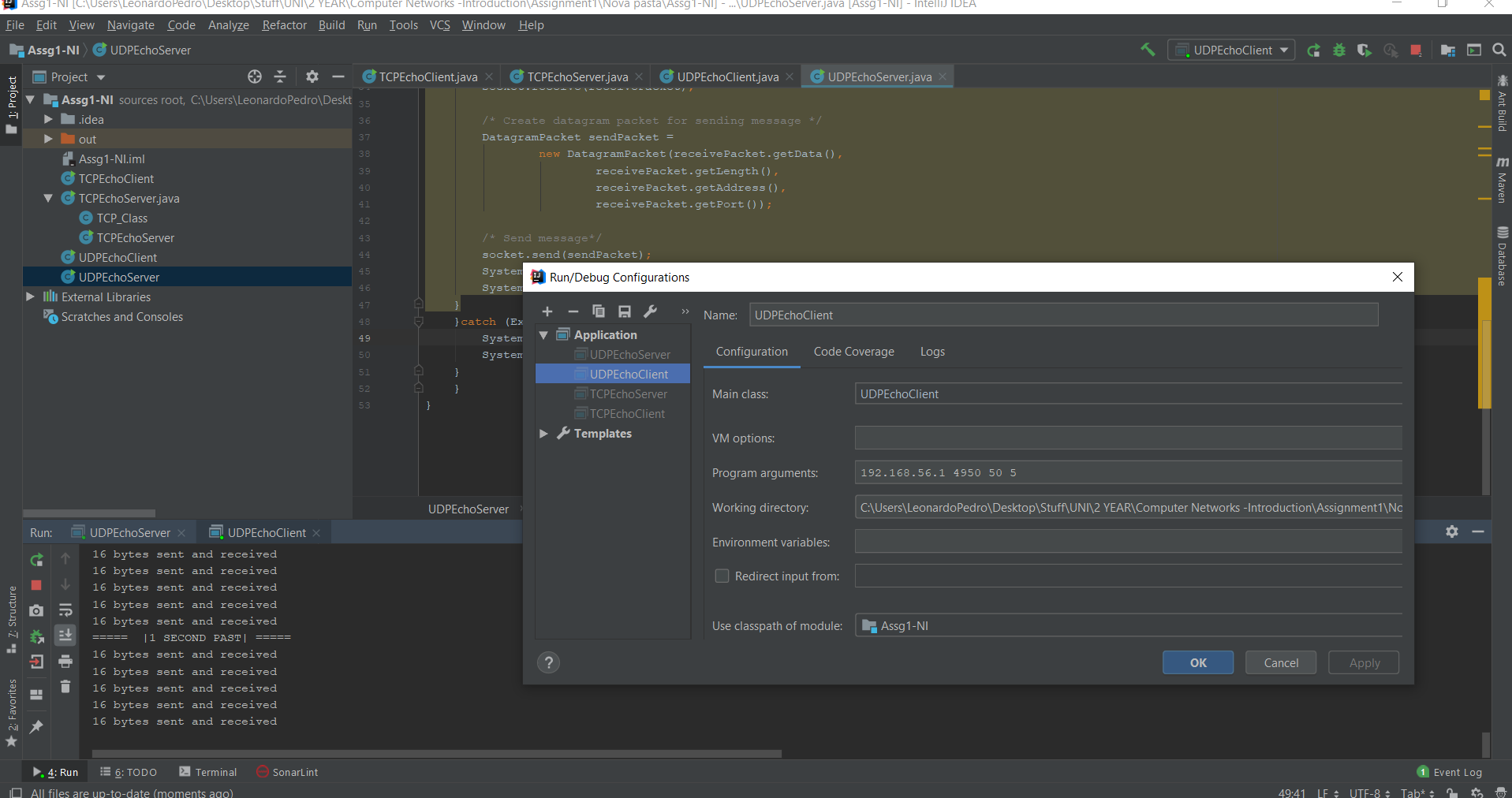


Figure 6.UDP - Buffer Size of 50

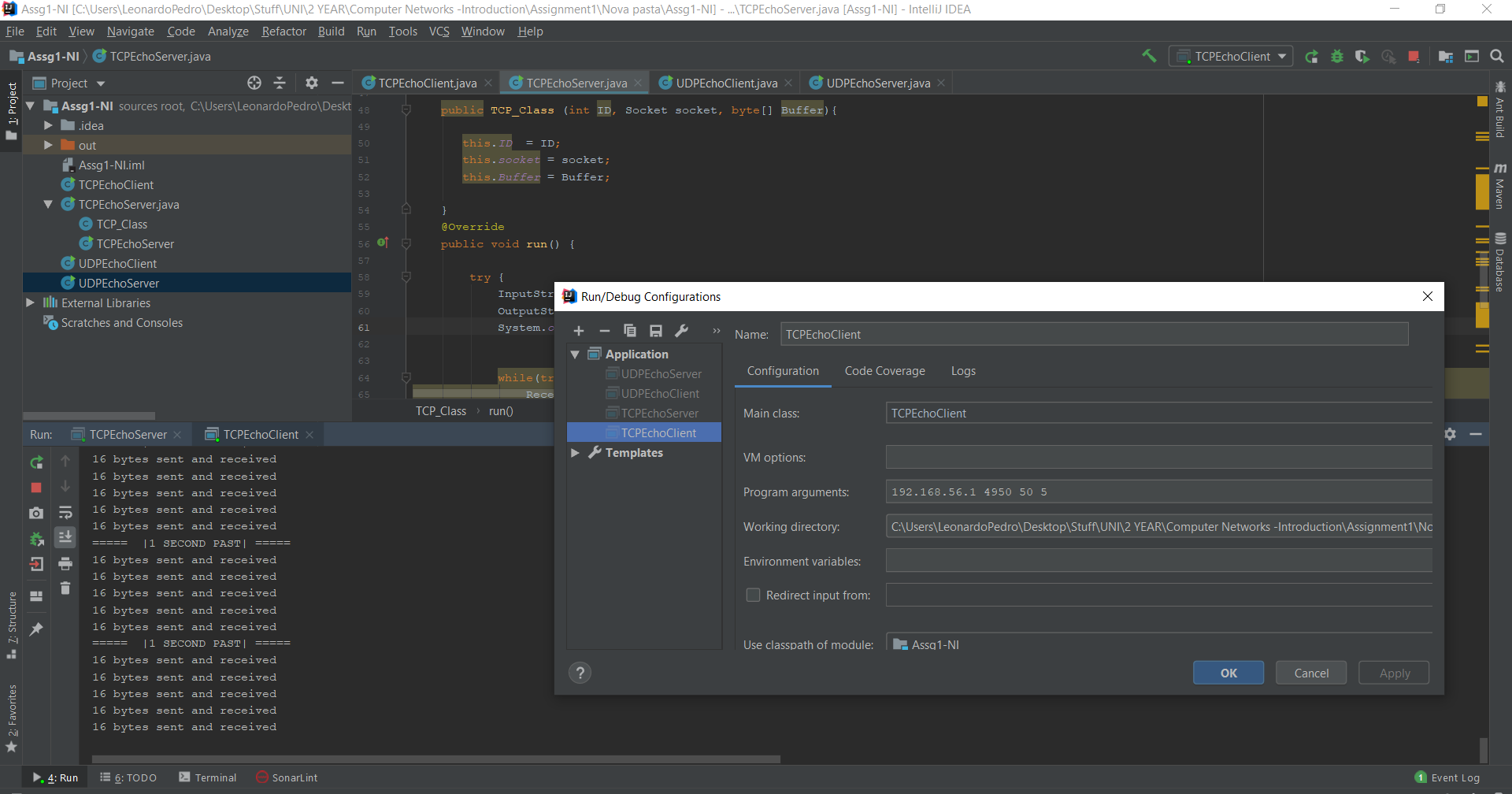


Figure 7. TCP - Buffer Size of 50

## 

## Discussion

Main Differences between **UDP** and **TCP**

|  |  |
| --- | --- |
| UDP | TCP |
| Datagram oriented | Connection oriented |
| Data delivery is not reliable (no orders) | Reliable data delivery |
| Basic error checking mechanism (checksum) | Extensive error checking mechanism |
| Fast Transmission | Slow Transmission |
| Less overhead (8 Bytes) | More overhead (20- 80 Bytes) |
| Supports broadcasting | Doesn't support broadcasting |
| No flow control | Flow Control |