Linnéuniversitetet
Kalmar Växjö

Report

Assignment 1

1DV701

Author: Sirwan Rasoul
Semester: Spring 2020
Email sr222qn

Linnéuniversitetet
Kalmar Växjö

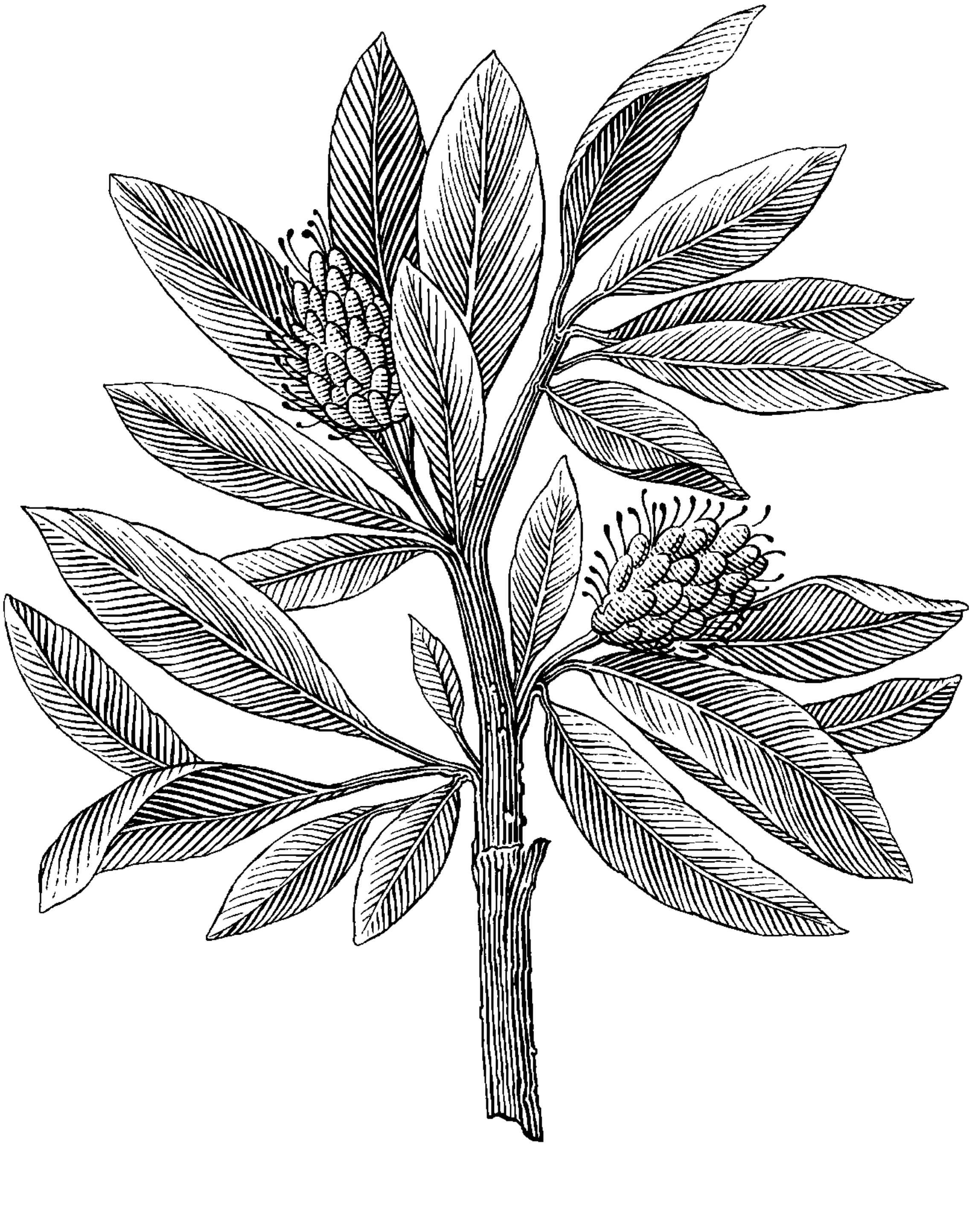
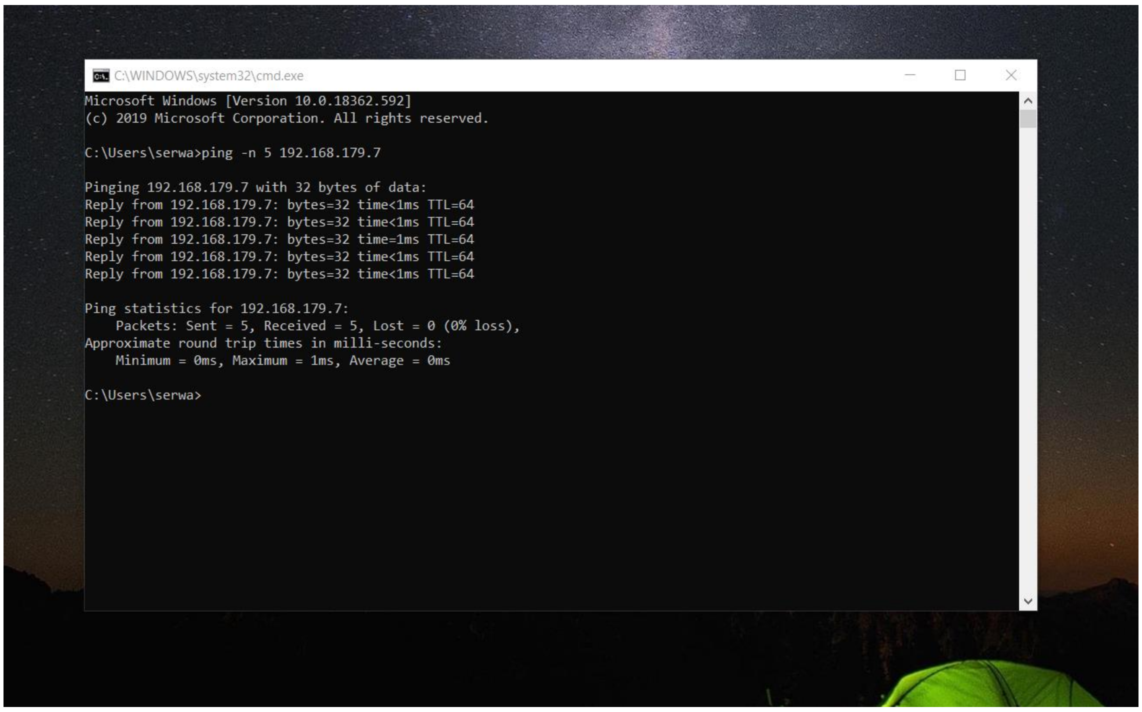


Table of Contents

1 Problem 1



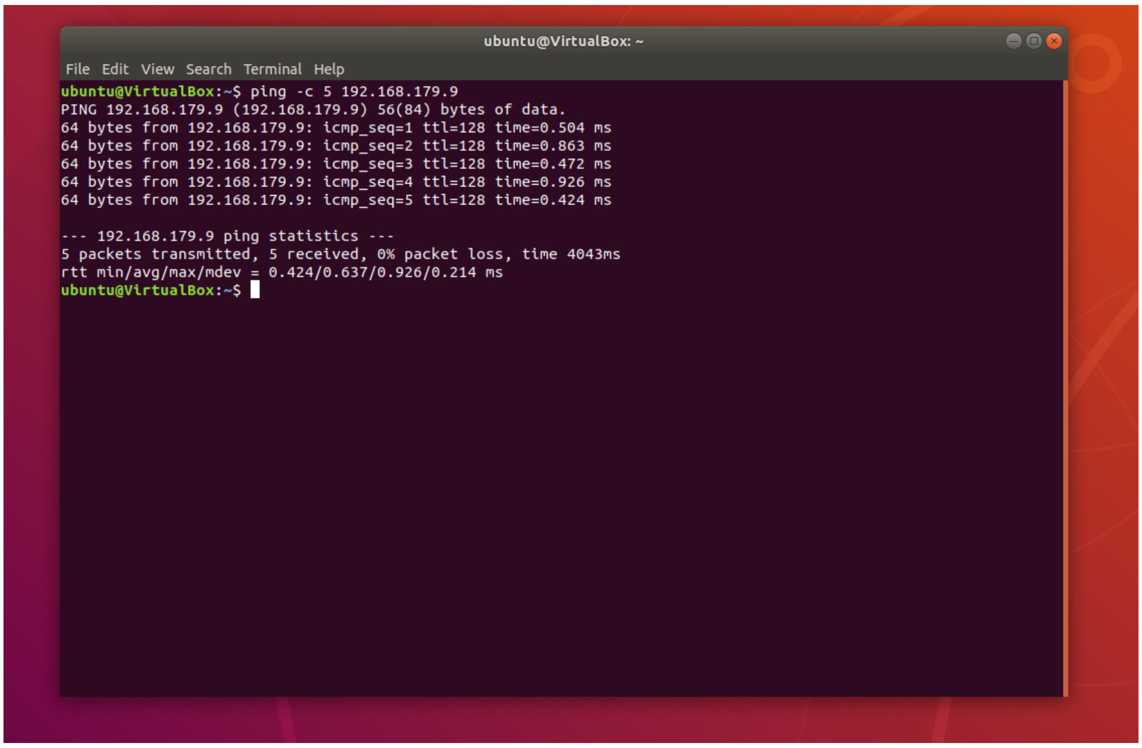
```
Microsoft Windows [Version 10.0.18362.592]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\serwa>ping -n 5 192.168.179.7

Pinging 192.168.179.7 with 32 bytes of data:
Reply from 192.168.179.7: bytes=32 time<1ms TTL=64
Reply from 192.168.179.7: bytes=32 time<1ms TTL=64
Reply from 192.168.179.7: bytes=32 time=1ms TTL=64
Reply from 192.168.179.7: bytes=32 time<1ms TTL=64
Reply from 192.168.179.7: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.179.7:
    Packets: Sent = 5, Received = 5, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\serwa>
```



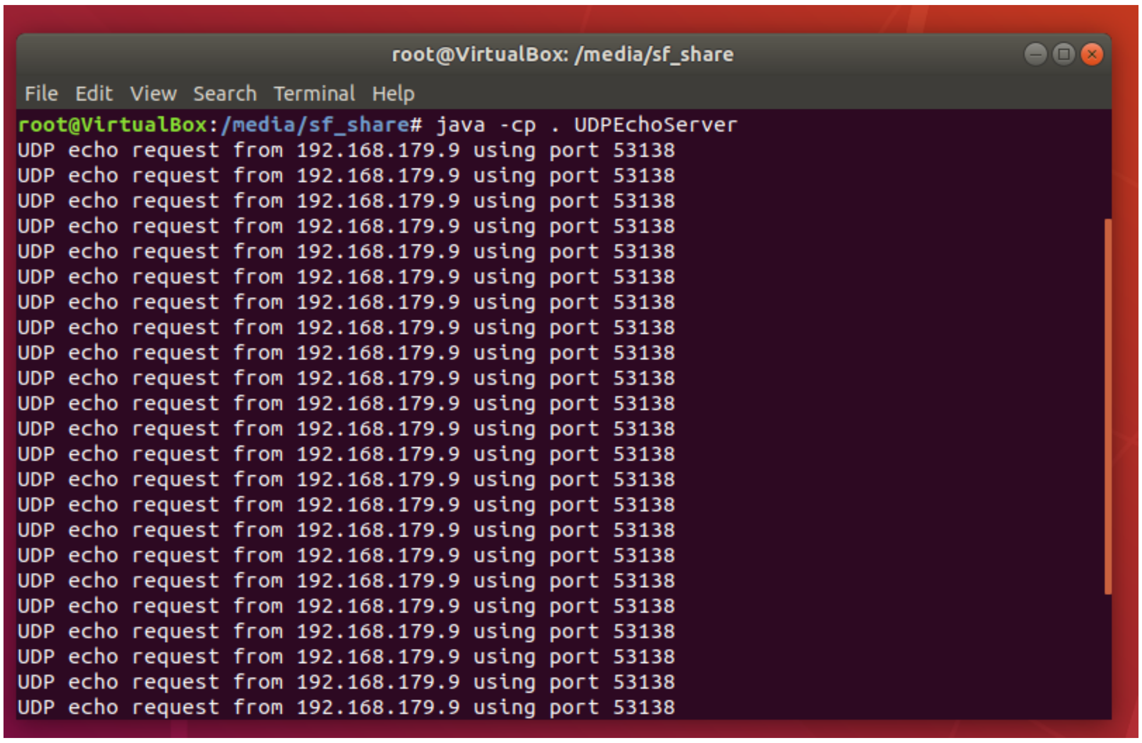
```
ubuntu@VirtualBox:~$ ping -c 5 192.168.179.9
PING 192.168.179.9 (192.168.179.9) 56(84) bytes of data.
64 bytes from 192.168.179.9: icmp_seq=1 ttl=128 time=0.504 ms
64 bytes from 192.168.179.9: icmp_seq=2 ttl=128 time=0.863 ms
64 bytes from 192.168.179.9: icmp_seq=3 ttl=128 time=0.472 ms
64 bytes from 192.168.179.9: icmp_seq=4 ttl=128 time=0.926 ms
64 bytes from 192.168.179.9: icmp_seq=5 ttl=128 time=0.424 ms

--- 192.168.179.9 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4043ms
rtt min/avg/max/mdev = 0.424/0.637/0.926/0.214 ms
ubuntu@VirtualBox:~$ 
```

1.1 Discussion

In the above screenshots, we can see ping from one to another

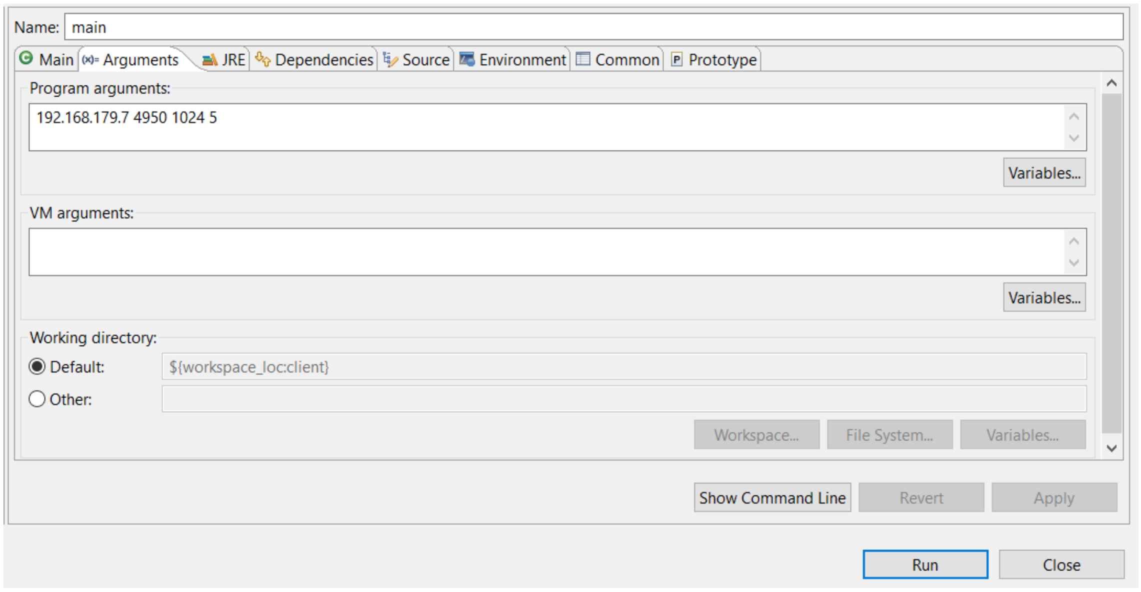
2 Problem 2



```
root@VirtualBox:/media/sf_share# java -cp . UDPEchoServer
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
UDP echo request from 192.168.179.9 using port 53138
```

root@VirtualBox: /media/sf_share
File Edit View Search Terminal Help

UDPEchoServer



Name: main

Program arguments: 192.168.179.7 4950 1024 5

VM arguments:

Working directory:
Default: ${workspace_loc:client}
Other:

UDPEchoClient

2.1 Discussion

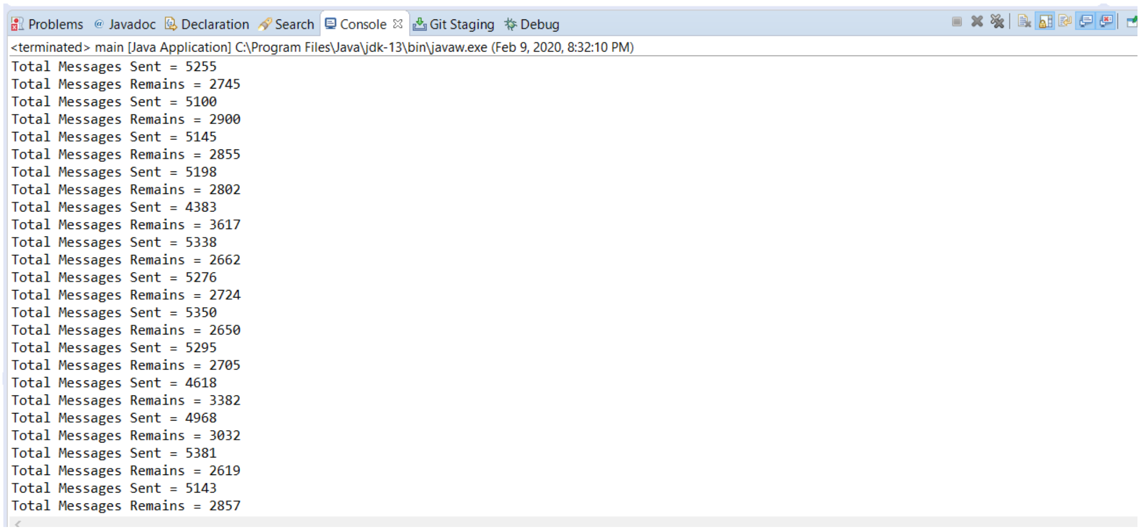
Here the first screenshot is for the UDP server and it shows the transfer rate of 5 per second, and I run it for 5 seconds.

The next screenshot shows that I am providing the IP address, port number, buffer size and transfer rate as arguments on the client-side.

List Off Exceptions:
Most of the cases I created my own classes to handle the different exception and in each class, you can provide a new error message to the constructor or use the default one that I provided earlier.

1- The IP address: I have provided a piece of code that handles the verification of the IP address for example if IP is empty or null if IP is ending with a (.) if each byte of the IP is less than 0 or greater than 255 and throws an exception if that is true.
2- The Port Number: I have provided a piece of code that handles the verification of the port number, and it checks if the port number is less than 0 or greater than 65535 and throws an exception if that is true.
3- The Buffer Size: Here I check for the case of UDP if the buffer size is less than the message size and I throw an exception in this case.
4- Arguments: If the user provides a wrong number of arguments he will get an exception.
5- Transfer Rate: If the transfer rate is less than 0 or is not an integer the user will get an exception.
6- Socket Exception: if the socket is closed or not connected the user will get an exception.
7- InterruptedException: for the interruption of the thread.
8- IO Exception: for the output stream and input stream.

2.2 VG 1



```
<terminated> main [Java Application] C:\Program Files\Java\jdk-13\bin\javaw.exe (Feb 9, 2020, 8:32:10 PM)
Total Messages Sent = 5255
Total Messages Remains = 2745
Total Messages Sent = 5100
Total Messages Remains = 2900
Total Messages Sent = 5145
Total Messages Remains = 2855
Total Messages Sent = 5198
Total Messages Remains = 2802
Total Messages Sent = 4383
Total Messages Remains = 3617
Total Messages Sent = 5338
Total Messages Remains = 2662
Total Messages Sent = 5276
Total Messages Remains = 2724
Total Messages Sent = 5350
Total Messages Remains = 2650
Total Messages Sent = 5295
Total Messages Remains = 2705
Total Messages Sent = 4618
Total Messages Remains = 3382
Total Messages Sent = 4968
Total Messages Remains = 3032
Total Messages Sent = 5381
Total Messages Remains = 2619
Total Messages Sent = 5143
Total Messages Remains = 2857
```

UDPEchoClient with 8000 transfer rate

2.2.1 Discussion

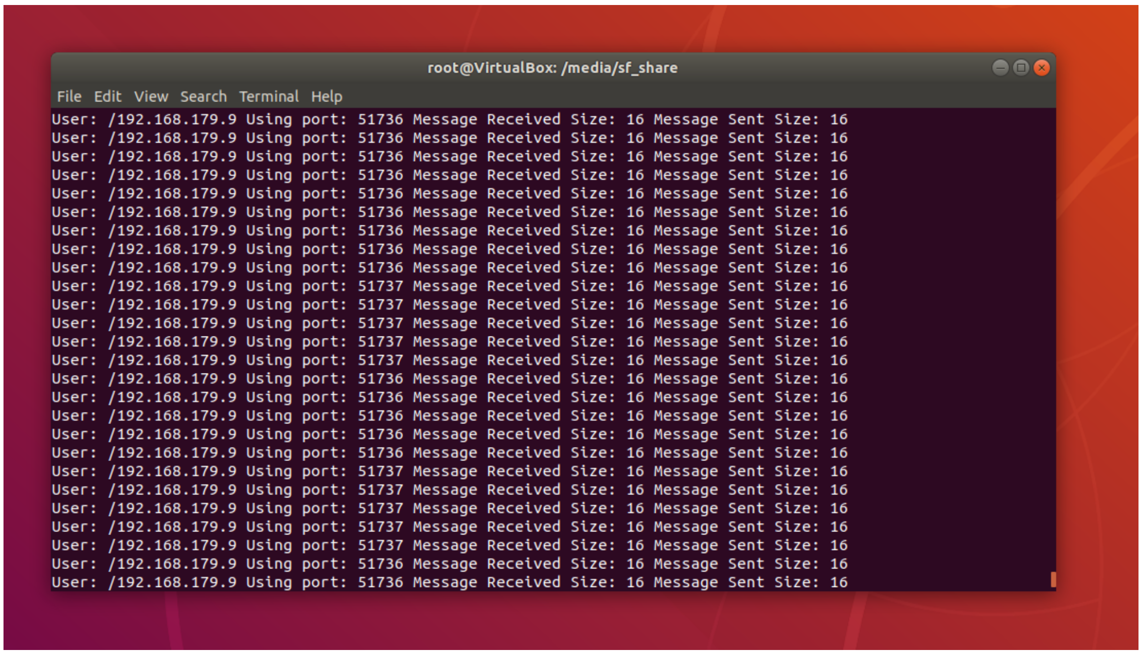
When I tried a higher transfer rate which was 8000 per second. I found that only nearly 5000 messages went through, but I also noticed when I run it for 30 seconds that the number is increasing by the time so I think that the connection is becoming faster after running for a while.

2.3 VG 2

2.3.1 Discussion

I created an abstract class called NetworkingLayer. First I had all my code in the main for the UDP client but then I realized there is a common code between both UDP client and TCP client. I have in the abstract class public method called argumentsAreValid which takes the program arguments and validates them, and also the public method isInteger to check if an argument is an integer or not, for example, the port number. The public method sentAndReceivedMessagesVerification to check if the sent and received messages are the same. The public method printConnectionDetalis, and finally an abstract method called run to be implemented by UDP or TCP client.

3 Problem 3


root@VirtualBox: /media/sf_share
File Edit View Search Terminal Help

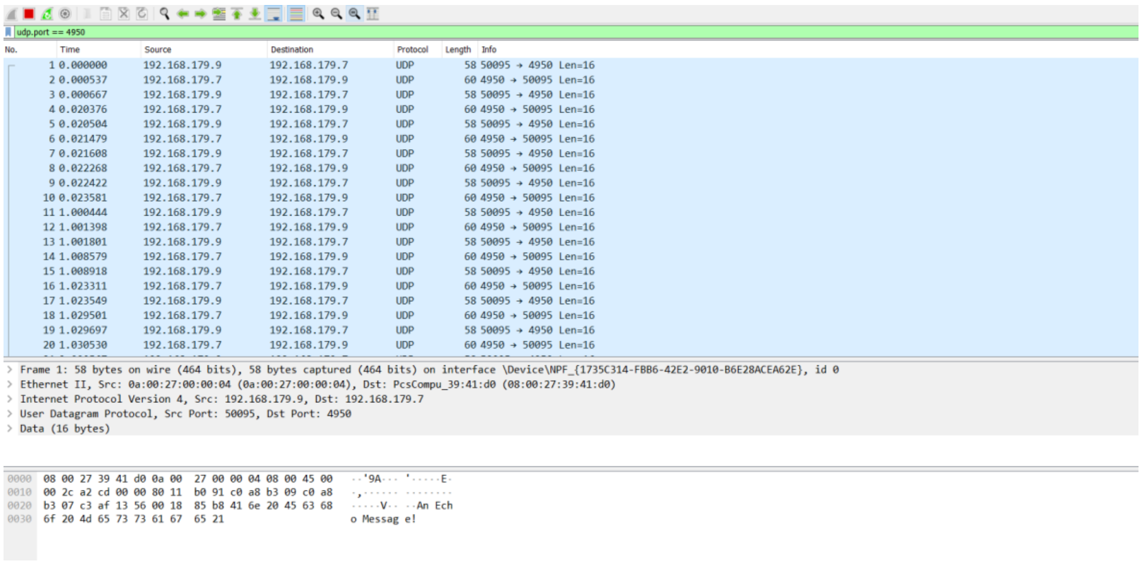
```
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51737 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51737 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51737 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51737 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51737 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51737 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51737 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51737 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51737 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51737 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
User: /192.168.179.9 Using port: 51736 Message Received Size: 16 Message Sent Size: 16
```

Multiple TCPEchoClients connection to a TCPEchoServer

3.1 Discussion

In the above screenshot, we can see that two different TCP clients are connected to the TCP server

4 Problem 4

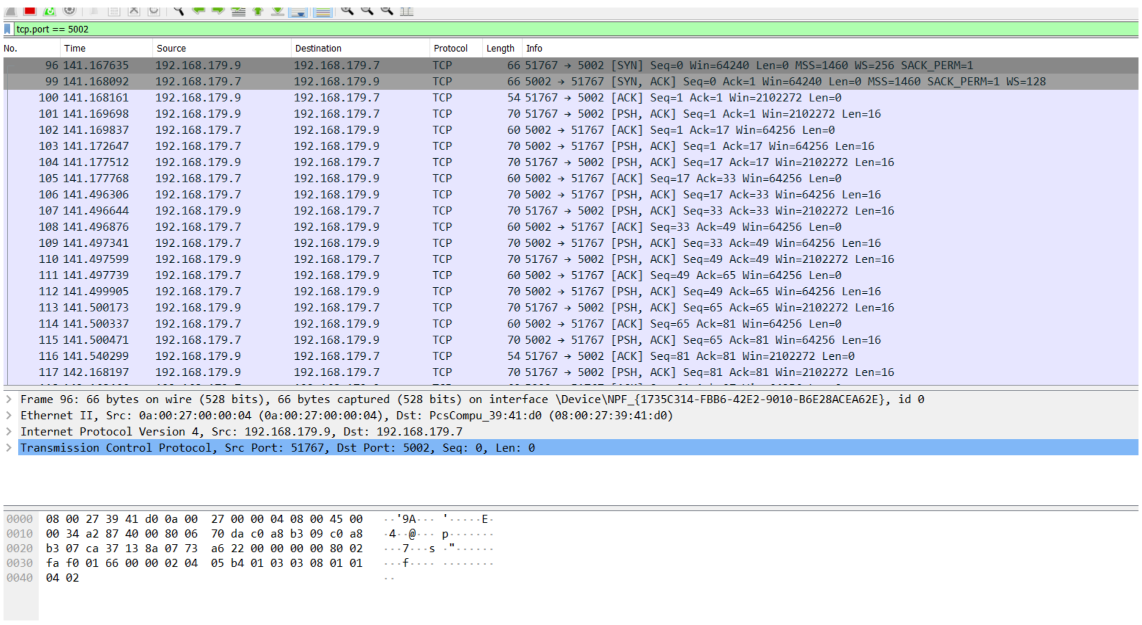


```
udp.port == 4950
No.  Time       Source          Destination     Protocol Length Info
   1 0.000000   192.168.179.9   192.168.179.7   UDP      58 50095 → 4950 Len=16
   2 0.000537   192.168.179.7   192.168.179.9   UDP      60 4950 → 50095 Len=16
   3 0.000667   192.168.179.9   192.168.179.7   UDP      58 50095 → 4950 Len=16
   4 0.020376   192.168.179.7   192.168.179.9   UDP      60 4950 → 50095 Len=16
   5 0.020504   192.168.179.9   192.168.179.7   UDP      58 50095 → 4950 Len=16
   6 0.021479   192.168.179.7   192.168.179.9   UDP      60 4950 → 50095 Len=16
   7 0.021608   192.168.179.9   192.168.179.7   UDP      58 50095 → 4950 Len=16
   8 0.022268   192.168.179.7   192.168.179.9   UDP      60 4950 → 50095 Len=16
   9 0.022422   192.168.179.9   192.168.179.7   UDP      58 50095 → 4950 Len=16
  10 0.023581   192.168.179.7   192.168.179.9   UDP      60 4950 → 50095 Len=16
  11 1.000444   192.168.179.9   192.168.179.7   UDP      58 50095 → 4950 Len=16
  12 1.001398   192.168.179.7   192.168.179.9   UDP      60 4950 → 50095 Len=16
  13 1.001801   192.168.179.9   192.168.179.7   UDP      58 50095 → 4950 Len=16
  14 1.008579   192.168.179.7   192.168.179.9   UDP      60 4950 → 50095 Len=16
  15 1.008918   192.168.179.9   192.168.179.7   UDP      58 50095 → 4950 Len=16
  16 1.023311   192.168.179.7   192.168.179.9   UDP      60 4950 → 50095 Len=16
  17 1.023549   192.168.179.9   192.168.179.7   UDP      58 50095 → 4950 Len=16
  18 1.029501   192.168.179.7   192.168.179.9   UDP      60 4950 → 50095 Len=16
  19 1.029697   192.168.179.9   192.168.179.7   UDP      58 50095 → 4950 Len=16
  20 1.030530   192.168.179.7   192.168.179.9   UDP      60 4950 → 50095 Len=16

> Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface \Device\NPF_{1735C314-FBB6-42E2-9010-B6E28ACEA62E}, id 0
> Ethernet II, Src: 0a:00:27:00:00:04 (0a:00:27:00:00:04), Dst: PcsCompu_39:41:d0 (08:00:27:39:41:d0)
> Internet Protocol Version 4, Src: 192.168.179.9, Dst: 192.168.179.7
> User Datagram Protocol, Src Port: 50095, Dst Port: 4950
> Data (16 bytes)

0000  08 00 27 39 41 d0 0a 00  27 00 00 04 08 00 45 00   ··'9A··· '·····E·
0010  00 2c a2 cd 00 00 80 11  b0 91 c0 a8 b3 09 c0 a8   ·,······ ········
0020  b3 07 c3 af 13 56 00 18  85 b8 41 6e 20 45 63 68   ·····V·· ··An Ech
0030  6f 20 4d 65 73 73 61 67  65 21                     o Messag e!
```

UDPEchoClient connection to UDPEchoServer



```
tcp.port == 5002
No.  Time         Source          Destination     Protocol Length Info
  96 141.167635   192.168.179.9   192.168.179.7   TCP      66 51767 → 5002 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  99 141.168092   192.168.179.7   192.168.179.9   TCP      66 5002 → 51767 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
 100 141.168161   192.168.179.9   192.168.179.7   TCP      54 51767 → 5002 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
 101 141.169698   192.168.179.9   192.168.179.7   TCP      70 51767 → 5002 [PSH, ACK] Seq=1 Ack=1 Win=2102272 Len=16
 102 141.169837   192.168.179.7   192.168.179.9   TCP      60 5002 → 51767 [ACK] Seq=1 Ack=17 Win=64256 Len=0
 103 141.172647   192.168.179.7   192.168.179.9   TCP      70 5002 → 51767 [PSH, ACK] Seq=1 Ack=17 Win=64256 Len=16
 104 141.177512   192.168.179.9   192.168.179.7   TCP      70 51767 → 5002 [PSH, ACK] Seq=17 Ack=17 Win=2102272 Len=16
 105 141.177768   192.168.179.7   192.168.179.9   TCP      60 5002 → 51767 [ACK] Seq=17 Ack=33 Win=64256 Len=0
 106 141.496306   192.168.179.7   192.168.179.9   TCP      70 5002 → 51767 [PSH, ACK] Seq=17 Ack=33 Win=64256 Len=16
 107 141.496644   192.168.179.9   192.168.179.7   TCP      70 51767 → 5002 [PSH, ACK] Seq=33 Ack=33 Win=2102272 Len=16
 108 141.496876   192.168.179.7   192.168.179.9   TCP      60 5002 → 51767 [ACK] Seq=33 Ack=49 Win=64256 Len=0
 109 141.497341   192.168.179.7   192.168.179.9   TCP      70 5002 → 51767 [PSH, ACK] Seq=33 Ack=49 Win=64256 Len=16
 110 141.497599   192.168.179.9   192.168.179.7   TCP      70 51767 → 5002 [PSH, ACK] Seq=49 Ack=49 Win=2102272 Len=16
 111 141.497739   192.168.179.7   192.168.179.9   TCP      60 5002 → 51767 [ACK] Seq=49 Ack=65 Win=64256 Len=0
 112 141.499905   192.168.179.7   192.168.179.9   TCP      70 5002 → 51767 [PSH, ACK] Seq=49 Ack=65 Win=64256 Len=16
 113 141.500173   192.168.179.9   192.168.179.7   TCP      70 51767 → 5002 [PSH, ACK] Seq=65 Ack=65 Win=2102272 Len=16
 114 141.500337   192.168.179.7   192.168.179.9   TCP      60 5002 → 51767 [ACK] Seq=65 Ack=81 Win=64256 Len=0
 115 141.500471   192.168.179.7   192.168.179.9   TCP      70 5002 → 51767 [PSH, ACK] Seq=65 Ack=81 Win=64256 Len=16
 116 141.540299   192.168.179.9   192.168.179.7   TCP      54 51767 → 5002 [ACK] Seq=81 Ack=81 Win=2102272 Len=0
 117 142.168197   192.168.179.9   192.168.179.7   TCP      70 51767 → 5002 [PSH, ACK] Seq=81 Ack=81 Win=2102272 Len=16

> Frame 96: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{1735C314-FBB6-42E2-9010-B6E28ACEA62E}, id 0
> Ethernet II, Src: 0a:00:27:00:00:04 (0a:00:27:00:00:04), Dst: PcsCompu_39:41:d0 (08:00:27:39:41:d0)
> Internet Protocol Version 4, Src: 192.168.179.9, Dst: 192.168.179.7
> Transmission Control Protocol, Src Port: 51767, Dst Port: 5002, Seq: 0, Len: 0

0000  08 00 27 39 41 d0 0a 00  27 00 00 04 08 00 45 00   ··'9A··· '·····E·
0010  00 34 a2 87 40 00 80 06  70 da c0 a8 b3 09 c0 a8   ·4··@··· p·······
0020  b3 07 ca 37 13 8a 07 73  a6 22 00 00 00 00 80 02   ···7···s ·"······
0030  fa f0 01 66 00 00 02 04  05 b4 01 03 03 08 01 01   ···f···· ········
0040  04 02                                              ··
```

TCPEchoClient connection to TCPEchoServer

4.1 Discussion

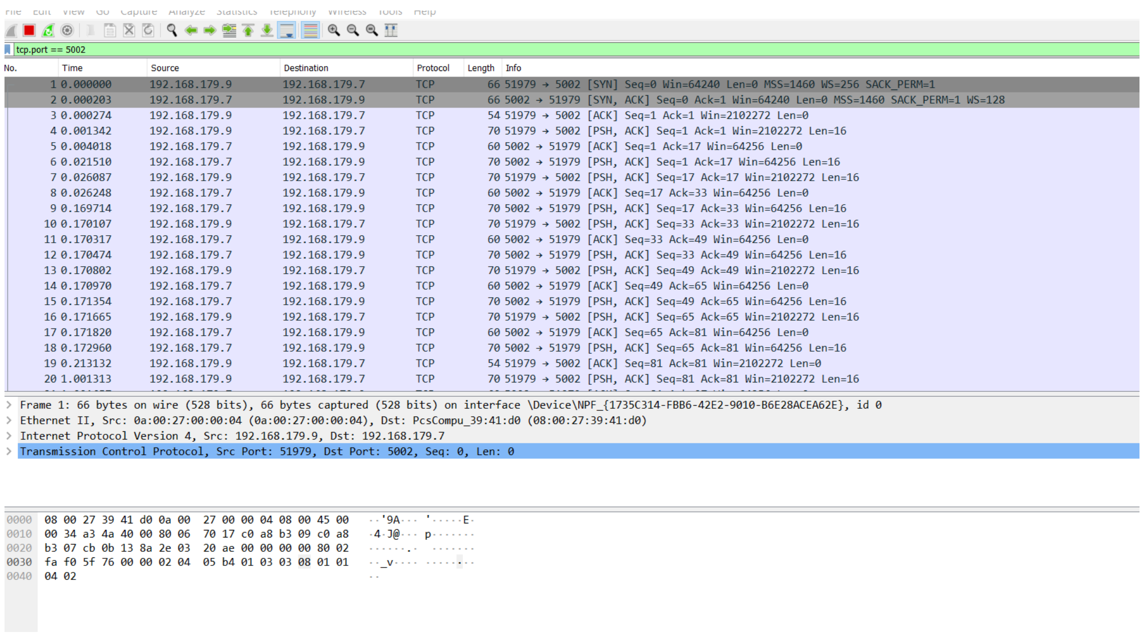
The first screenshot shows a UDP traffic UDP client and UDP server.
Here I see the following steps:
1. The client with IP address 192.168.179.9 sent a message to a server using UDP protocol the actual message size is 16 and the rest is for the header.
2. The server sent back the same message with different header size as we see.
And so on.
In the middle of the image, we see the different connection layers.
In the bottom, we see the actual data represented by hexadecimal.

The second screenshot is for the TCP traffic TCP client and TCP server.
Here I see the following:
1. The client with IP address 192.168.192.9 sent a synchronize (SYN) message to the server using TCP and the length here is 66.
2. The server sent back a synchronize-acknowledgment (SYN-ACK) message to the client and the length here is 66.
3. The client sent back an acknowledgment message.
4. In this step, the client sent a message to the server, and here we see the PSH flag is. The message size is 16 and the rest is for the header.
5. The server after receiving the message sent back an acknowledgment message.
6. The server sent the actual message back to the client.
7. The client got the message back and sent again a new message and an acknowledgment message for the previous message to the server.
And so on.
In the middle of the image, we see the different connection layers.
In the bottom, we see the actual data represented by hexadecimal.

SYN: is a packet sent to another computer to demand a connection to it.
ACK: is an acknowledgment message used to verify that the packet sent is received.
PSH: is a flag in TCP that makes the receiver to send data even though the buffer is not filled yet

What is the difference between TCP and UDP?
The obvious difference is that the TCP is a connection-based protocol and the UDP is a connectionless protocol, and we can see that clearly in the Wireshark screenshots.
We do not see too much stuff in UDP, but in TCP more stuff are happening during the connection.
In TCP the message is not lost when the buffer size is different the stram continuous to send and receive until the message is empty
In UDP the message will be sent out one time and if the buffer size is smaller the rest of the message will be lost.

5 Problem 5

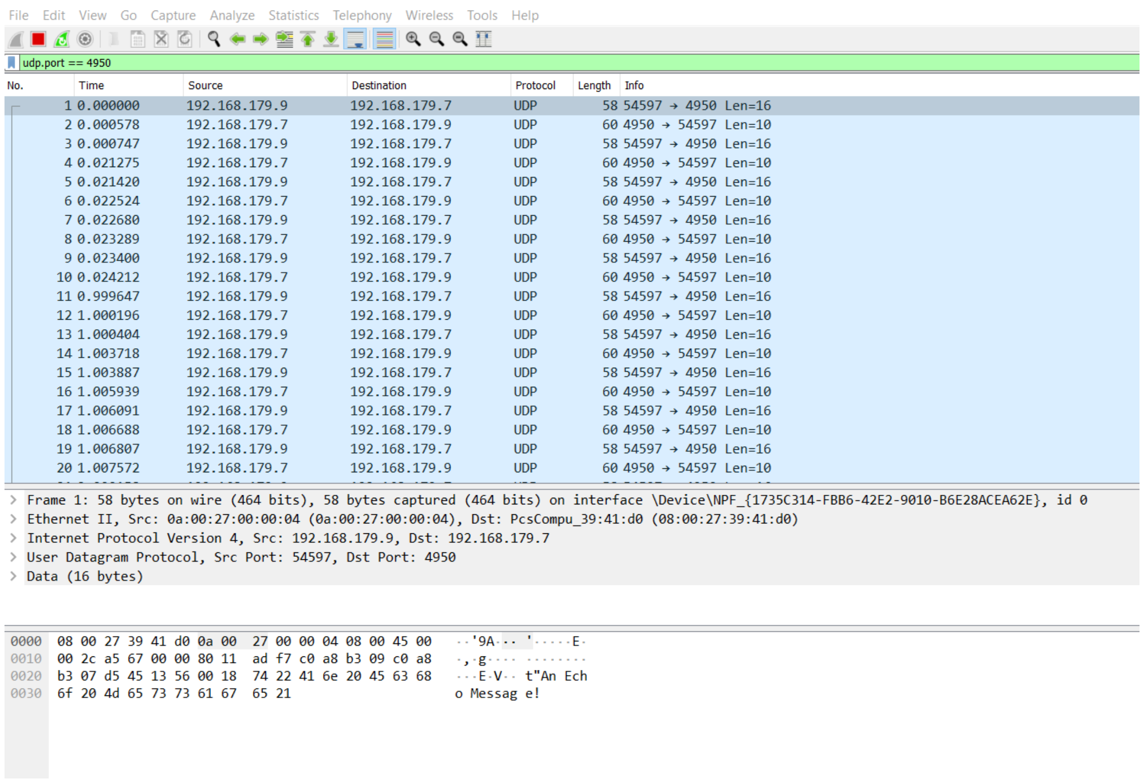

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
tcp.port == 5002
No.  Time      Source         Destination    Protocol Length Info
   1 0.000000  192.168.179.9  192.168.179.7  TCP      66 51979 → 5002 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
   2 0.000203  192.168.179.7  192.168.179.9  TCP      66 5002 → 51979 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
   3 0.000274  192.168.179.9  192.168.179.7  TCP      54 51979 → 5002 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
   4 0.001342  192.168.179.9  192.168.179.7  TCP      70 51979 → 5002 [PSH, ACK] Seq=1 Ack=1 Win=2102272 Len=16
   5 0.004018  192.168.179.7  192.168.179.9  TCP      60 5002 → 51979 [ACK] Seq=1 Ack=17 Win=64256 Len=0
   6 0.021510  192.168.179.7  192.168.179.9  TCP      70 5002 → 51979 [PSH, ACK] Seq=1 Ack=17 Win=64256 Len=16
   7 0.026087  192.168.179.9  192.168.179.7  TCP      70 51979 → 5002 [PSH, ACK] Seq=17 Ack=17 Win=2102272 Len=16
   8 0.026248  192.168.179.7  192.168.179.9  TCP      60 5002 → 51979 [ACK] Seq=17 Ack=33 Win=64256 Len=0
   9 0.169714  192.168.179.7  192.168.179.9  TCP      70 5002 → 51979 [PSH, ACK] Seq=17 Ack=33 Win=64256 Len=16
  10 0.170107  192.168.179.9  192.168.179.7  TCP      70 51979 → 5002 [PSH, ACK] Seq=33 Ack=33 Win=2102272 Len=16
  11 0.170317  192.168.179.7  192.168.179.9  TCP      60 5002 → 51979 [ACK] Seq=33 Ack=49 Win=64256 Len=0
  12 0.170474  192.168.179.7  192.168.179.9  TCP      70 5002 → 51979 [PSH, ACK] Seq=33 Ack=49 Win=64256 Len=16
  13 0.170802  192.168.179.9  192.168.179.7  TCP      70 51979 → 5002 [PSH, ACK] Seq=49 Ack=49 Win=2102272 Len=16
  14 0.170970  192.168.179.7  192.168.179.9  TCP      60 5002 → 51979 [ACK] Seq=49 Ack=65 Win=64256 Len=0
  15 0.171354  192.168.179.7  192.168.179.9  TCP      70 5002 → 51979 [PSH, ACK] Seq=49 Ack=65 Win=64256 Len=16
  16 0.171665  192.168.179.9  192.168.179.7  TCP      70 51979 → 5002 [PSH, ACK] Seq=65 Ack=65 Win=2102272 Len=16
  17 0.171820  192.168.179.7  192.168.179.9  TCP      60 5002 → 51979 [ACK] Seq=65 Ack=81 Win=64256 Len=0
  18 0.172960  192.168.179.7  192.168.179.9  TCP      70 5002 → 51979 [PSH, ACK] Seq=65 Ack=81 Win=64256 Len=16
  19 0.213132  192.168.179.9  192.168.179.7  TCP      54 51979 → 5002 [ACK] Seq=81 Ack=81 Win=2102272 Len=0
  20 1.001313  192.168.179.9  192.168.179.7  TCP      70 51979 → 5002 [PSH, ACK] Seq=81 Ack=81 Win=2102272 Len=16

> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{1735C314-FBB6-42E2-9010-B6E28ACEA62E}, id 0
> Ethernet II, Src: 0a:00:27:00:00:04 (0a:00:27:00:00:04), Dst: PcsCompu_39:41:d0 (08:00:27:39:41:d0)
> Internet Protocol Version 4, Src: 192.168.179.9, Dst: 192.168.179.7
> Transmission Control Protocol, Src Port: 51979, Dst Port: 5002, Seq: 0, Len: 0

0000  08 00 27 39 41 d0 0a 00  27 00 00 04 08 00 45 00   ··'9A··· '·····E·
0010  00 34 a3 4a 40 00 80 06  70 17 c0 a8 b3 09 c0 a8   ·4·J@··· p·······
0020  b3 07 cb 0b 13 8a 2e 03  20 ae 00 00 00 00 80 02   ······.·  ·······
0030  fa f0 5f 76 00 00 02 04  05 b4 01 03 03 08 01 01   ··_v···· ·····.··
0040  04 02                                              ··
```

TCP client sends a message with 16 bytes and buffer size of 10 bytes.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
udp.port == 4950
No.  Time      Source         Destination    Protocol Length Info
   1 0.000000  192.168.179.9  192.168.179.7  UDP      58 54597 → 4950 Len=16
   2 0.000578  192.168.179.7  192.168.179.9  UDP      60 4950 → 54597 Len=10
   3 0.000747  192.168.179.9  192.168.179.7  UDP      58 54597 → 4950 Len=16
   4 0.021275  192.168.179.7  192.168.179.9  UDP      60 4950 → 54597 Len=10
   5 0.021420  192.168.179.9  192.168.179.7  UDP      58 54597 → 4950 Len=16
   6 0.022524  192.168.179.7  192.168.179.9  UDP      60 4950 → 54597 Len=10
   7 0.022680  192.168.179.9  192.168.179.7  UDP      58 54597 → 4950 Len=16
   8 0.023289  192.168.179.7  192.168.179.9  UDP      60 4950 → 54597 Len=10
   9 0.023400  192.168.179.9  192.168.179.7  UDP      58 54597 → 4950 Len=16
  10 0.024212  192.168.179.7  192.168.179.9  UDP      60 4950 → 54597 Len=10
  11 0.999647  192.168.179.9  192.168.179.7  UDP      58 54597 → 4950 Len=16
  12 1.000196  192.168.179.7  192.168.179.9  UDP      60 4950 → 54597 Len=10
  13 1.000404  192.168.179.9  192.168.179.7  UDP      58 54597 → 4950 Len=16
  14 1.003718  192.168.179.7  192.168.179.9  UDP      60 4950 → 54597 Len=10
  15 1.003887  192.168.179.9  192.168.179.7  UDP      58 54597 → 4950 Len=16
  16 1.005939  192.168.179.7  192.168.179.9  UDP      60 4950 → 54597 Len=10
  17 1.006091  192.168.179.9  192.168.179.7  UDP      58 54597 → 4950 Len=16
  18 1.006688  192.168.179.7  192.168.179.9  UDP      60 4950 → 54597 Len=10
  19 1.006807  192.168.179.9  192.168.179.7  UDP      58 54597 → 4950 Len=16
  20 1.007572  192.168.179.7  192.168.179.9  UDP      60 4950 → 54597 Len=10

> Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface \Device\NPF_{1735C314-FBB6-42E2-9010-B6E28ACEA62E}, id 0
> Ethernet II, Src: 0a:00:27:00:00:04 (0a:00:27:00:00:04), Dst: PcsCompu_39:41:d0 (08:00:27:39:41:d0)
> Internet Protocol Version 4, Src: 192.168.179.9, Dst: 192.168.179.7
> User Datagram Protocol, Src Port: 54597, Dst Port: 4950
> Data (16 bytes)

0000  08 00 27 39 41 d0 0a 00  27 00 00 04 08 00 45 00   ··'9A··· '·····E·
0010  00 2c a5 67 00 00 80 11  ad f7 c0 a8 b3 09 c0 a8   ·,·g···· ········
0020  b3 07 d5 45 13 56 00 18  74 22 41 6e 20 45 63 68   ···E·V·· t"An Ech
0030  6f 20 4d 65 73 73 61 67  65 21                     o Messag e!
```

UDP client sends a message with 16 bytes and buffer size of 10 in the server

5.1 Discussion

In the TCP case, the client received the message even though he had a lower buffer size, and we can say the message will be delivered to the client regardless of the buffer size.

In the UDP case, we see the client did not receive what he sent to the server, and he only received 10 bytes and the other bytes are lost due to the UDP protocol.