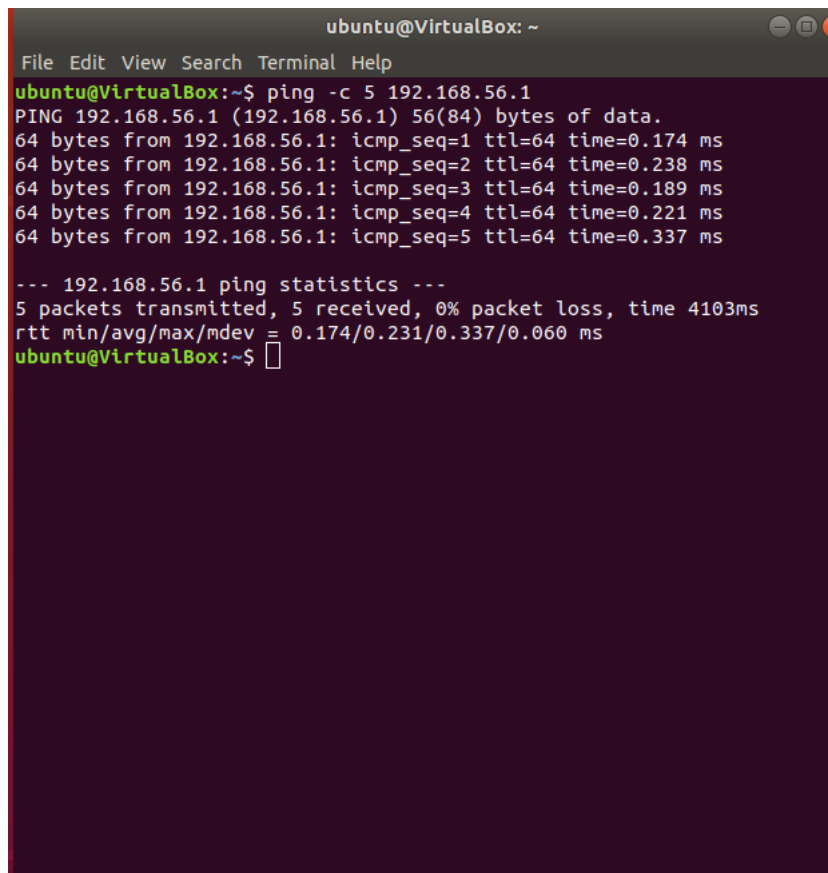


Assignment 1 1DV701 Spring 2020

Andrei Neagu (an223kj)

February 2020

1 Problem 1

A screenshot of a terminal window titled 'ubuntu@VirtualBox: ~'. The terminal shows the execution of the command 'ping -c 5 192.168.56.1'. The output displays five successful ping responses with their respective times and sequence numbers. Below the individual responses, a summary line shows '5 packets transmitted, 5 received, 0% packet loss, time 4103ms'. The final line shows the round-trip time statistics: 'rtt min/avg/max/mdev = 0.174/0.231/0.337/0.060 ms'. The prompt 'ubuntu@VirtualBox:~\$' is visible at the bottom.

```
ubuntu@VirtualBox: ~  
File Edit View Search Terminal Help  
ubuntu@VirtualBox:~$ ping -c 5 192.168.56.1  
PING 192.168.56.1 (192.168.56.1) 56(84) bytes of data:  
64 bytes from 192.168.56.1: icmp_seq=1 ttl=64 time=0.174 ms  
64 bytes from 192.168.56.1: icmp_seq=2 ttl=64 time=0.238 ms  
64 bytes from 192.168.56.1: icmp_seq=3 ttl=64 time=0.189 ms  
64 bytes from 192.168.56.1: icmp_seq=4 ttl=64 time=0.221 ms  
64 bytes from 192.168.56.1: icmp_seq=5 ttl=64 time=0.337 ms  
  
--- 192.168.56.1 ping statistics ---  
5 packets transmitted, 5 received, 0% packet loss, time 4103ms  
rtt min/avg/max/mdev = 0.174/0.231/0.337/0.060 ms  
ubuntu@VirtualBox:~$
```

I pinged my main machine from the virtual machine.

2 Problem 2

The UDP server and it's activity

```
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
UDP echo request from 127.0.0.1 using port 46152
```

The UDP client and it's activity. The arguments here were a buffer of 1024 bytes, 5 messages per second and the messages were 10 bytes long

```
Number of messages sent in a second: 5 out of 5 || Number of messages left: 0
Number of messages sent in a second: 5 out of 5 || Number of messages left: 0
Number of messages sent in a second: 5 out of 5 || Number of messages left: 0
Number of messages sent in a second: 5 out of 5 || Number of messages left: 0
Number of messages sent in a second: 5 out of 5 || Number of messages left: 0
Number of messages sent in a second: 5 out of 5 || Number of messages left: 0
Number of messages sent in a second: 5 out of 5 || Number of messages left: 0
Number of messages sent in a second: 5 out of 5 || Number of messages left: 0
```

Exceptions covered:

- Transmission rate must be positive.
- Port less than 65535 and greater than 0.
- The IP address that are allowed. Comments in the code.
- The buffer size allowed. Comments in the code.
- I allowed the messages to be sent if the are larger than the buffer but I print errors in the console
- I set the timeout to 2 seconds and print errors if the server is not served

I made this implementation this way because I also did VG task 1

2.1 VG task 1

The main logic is in UDPEchoClient.sendAndReceiveOneSec().

It turned out to be fairly complicated because I found some last minute bugs and I added some if clauses to fix them.

```
Number of messages sent in a second: 9065 out of 30000 || Number of messages left: 20935
Number of messages sent in a second: 14769 out of 30000 || Number of messages left: 15231
Number of messages sent in a second: 28314 out of 30000 || Number of messages left: 1686
Number of messages sent in a second: 29375 out of 30000 || Number of messages left: 625
Number of messages sent in a second: 29625 out of 30000 || Number of messages left: 375
Number of messages sent in a second: 28305 out of 30000 || Number of messages left: 1695
Number of messages sent in a second: 30000 out of 30000 || Number of messages left: 0
Number of messages sent in a second: 29448 out of 30000 || Number of messages left: 552
Number of messages sent in a second: 28873 out of 30000 || Number of messages left: 1127
Number of messages sent in a second: 28671 out of 30000 || Number of messages left: 1329
Number of messages sent in a second: 29405 out of 30000 || Number of messages left: 595
Number of messages sent in a second: 27796 out of 30000 || Number of messages left: 2204
Number of messages sent in a second: 29017 out of 30000 || Number of messages left: 983
Number of messages sent in a second: 29603 out of 30000 || Number of messages left: 397
Number of messages sent in a second: 28863 out of 30000 || Number of messages left: 1137
Number of messages sent in a second: 29183 out of 30000 || Number of messages left: 817
Number of messages sent in a second: 29184 out of 30000 || Number of messages left: 816
Number of messages sent in a second: 27450 out of 30000 || Number of messages left: 2550
Number of messages sent in a second: 26450 out of 30000 || Number of messages left: 3550
Number of messages sent in a second: 27784 out of 30000 || Number of messages left: 2216
Number of messages sent in a second: 28343 out of 30000 || Number of messages left: 1657
Number of messages sent in a second: 27839 out of 30000 || Number of messages left: 2161
```

The behaviour is the same like for every java program it speeds up due to the JVM optimizing the code. On my machine the limit is about 30000 messages depending on how many apps I have open. If I make room for the processing power it goes up. Resource dependant. This is a LAN network. It also stabilizes in the upper range.

2.2 VG task 2

I created a NetworkLayer abstract class for this task. This class deals with argument validation and it holds the fields that classes have in common. I did so because it was the easiest for me. All the other classes extend the NetworkLayer. The Network layer has various constructors and private methods for validation. I specify in the comments which constructors are used by which classes. Also it looks better because in this way the servers and the clients can isolate their particular implementation. In the Network layer there is some code duplication that I kept for readability of the TA.

3 Task 3

This task was wrong. With a smaller buffer than the message the transmission was broken. I fixed this by understanding TCP better. Now the transmission works with a smaller buffer size too. The string transmitted was message.

Here the client has a buffer of 1 and the server 10.

| | | | | | |
|----|----------|----------------|----------------|-----|--|
| 1 | 0.00009. | 192.168.56.1 | 192.168.56.101 | TCP | 54574 → 4850 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2288813299 TSecr=0 WS=1 |
| 2 | 0.00024. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54574 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3831821244 TSecr=3831821244 |
| 3 | 0.00028. | 192.168.56.1 | 192.168.56.101 | TCP | 54574 → 4850 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2288813299 TSecr=3831821244 |
| 4 | 0.00092. | 192.168.56.1 | 192.168.56.101 | TCP | 54574 → 4850 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=7 TSval=2288813300 TSecr=3831821244 |
| 5 | 0.00110. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54574 [ACK] Seq=1 Ack=8 Win=65280 Len=0 TSval=3831821245 TSecr=2288813300 |
| 6 | 0.00254. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54574 [PSH, ACK] Seq=1 Ack=8 Win=65280 Len=7 TSval=3831821246 TSecr=2288813300 |
| 7 | 0.00257. | 192.168.56.1 | 192.168.56.101 | TCP | 54574 → 4850 [ACK] Seq=8 Ack=8 Win=64256 Len=0 TSval=2288813301 TSecr=3831821246 |
| 8 | 1.00447. | 192.168.56.1 | 192.168.56.101 | TCP | 54574 → 4850 [PSH, ACK] Seq=8 Ack=8 Win=64256 Len=7 TSval=2288814303 TSecr=3831821246 |
| 9 | 1.00471. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54574 [ACK] Seq=8 Ack=15 Win=65280 Len=0 TSval=3831822249 TSecr=2288814303 |
| 10 | 1.00498. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54574 [PSH, ACK] Seq=8 Ack=15 Win=65280 Len=7 TSval=3831822249 TSecr=2288814303 |
| 11 | 1.00508. | 192.168.56.1 | 192.168.56.101 | TCP | 54574 → 4850 [ACK] Seq=15 Ack=15 Win=64256 Len=0 TSval=2288814304 TSecr=3831822249 |
| 12 | 1.90990. | 192.168.56.1 | 192.168.56.101 | TCP | 54574 → 4850 [FIN, ACK] Seq=15 Ack=15 Win=64256 Len=0 TSval=2288815209 TSecr=3831822249 |
| 13 | 1.95165. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54574 [ACK] Seq=15 Ack=16 Win=65280 Len=0 TSval=3831823196 TSecr=2288815209 |

```
Frame 4: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
Ethernet II, Src: 0a:00:27:00:00:00 (0a:00:27:00:00:00), Dst: PcsCompu.77:5c:90 (08:00:27:77:5c:90)
Internet Protocol Version 4, Src: 192.168.56.1, Dst: 192.168.56.101
Transmission Control Protocol, Src Port: 54574, Dst Port: 4850, Seq: 1, Ack: 1, Len: 7
Data (7 bytes)
Data: 6d657373616765
[Length: 7]
```

```
30 08 00 27 77 5c 90 0a 00 27 00 00 00 08 00 45 00 --'\w...E...
10 00 3b 9c a6 40 00 40 06 ac 5f c0 a8 38 01 c0 a8 --;@:..8...
20 38 65 d5 2e 12 f2 f9 0c 37 6b b8 97 24 38 80 18 8e.....7k.$8...
30 01 f6 00 d5 00 00 01 01 08 0a 88 6c 84 f4 e4 64 --.....1...d
40 f3 bc 6d 65 73 73 61 67 65 --message
```

Here the server has a buffer of 1 and the client 10.

| | | | | | |
|----|----------|----------------|----------------|-----|--|
| 1 | 0.00000. | 192.168.56.1 | 192.168.56.101 | TCP | 54552 → 4850 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2288486758 TSecr=0 WS=1 |
| 2 | 0.00022. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54552 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3831494542 TSecr=3831494542 |
| 3 | 0.00026. | 192.168.56.1 | 192.168.56.101 | TCP | 54552 → 4850 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2288486758 TSecr=3831494542 |
| 4 | 0.00070. | 192.168.56.1 | 192.168.56.101 | TCP | 54552 → 4850 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=7 TSval=2288486759 TSecr=3831494542 |
| 5 | 0.00084. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54552 [ACK] Seq=1 Ack=8 Win=65280 Len=0 TSval=3831494542 TSecr=2288486759 |
| 6 | 0.00227. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54552 [PSH, ACK] Seq=1 Ack=8 Win=65280 Len=7 TSval=3831494544 TSecr=2288486759 |
| 7 | 0.00230. | 192.168.56.1 | 192.168.56.101 | TCP | 54552 → 4850 [ACK] Seq=8 Ack=8 Win=64256 Len=0 TSval=2288486760 TSecr=3831494544 |
| 8 | 1.00361. | 192.168.56.1 | 192.168.56.101 | TCP | 54552 → 4850 [PSH, ACK] Seq=8 Ack=8 Win=64256 Len=7 TSval=2288487762 TSecr=3831494544 |
| 9 | 1.00381. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54552 [ACK] Seq=8 Ack=15 Win=65280 Len=0 TSval=3831495546 TSecr=2288487762 |
| 10 | 1.00413. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54552 [PSH, ACK] Seq=8 Ack=15 Win=65280 Len=7 TSval=3831495546 TSecr=2288487762 |
| 11 | 1.00415. | 192.168.56.1 | 192.168.56.101 | TCP | 54552 → 4850 [ACK] Seq=15 Ack=15 Win=64256 Len=0 TSval=2288487762 TSecr=3831495546 |
| 12 | 2.00500. | 192.168.56.1 | 192.168.56.101 | TCP | 54552 → 4850 [PSH, ACK] Seq=15 Ack=15 Win=64256 Len=7 TSval=2288488763 TSecr=3831495546 |
| 13 | 2.00534. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54552 [ACK] Seq=15 Ack=22 Win=65280 Len=0 TSval=3831496548 TSecr=2288488763 |
| 14 | 2.00580. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54552 [PSH, ACK] Seq=15 Ack=22 Win=65280 Len=7 TSval=3831496548 TSecr=2288488763 |
| 15 | 2.00583. | 192.168.56.1 | 192.168.56.101 | TCP | 54552 → 4850 [ACK] Seq=22 Ack=22 Win=64256 Len=0 TSval=2288488764 TSecr=3831496548 |
| 16 | 2.00896. | 192.168.56.1 | 192.168.56.101 | TCP | 54552 → 4850 [FIN, ACK] Seq=22 Ack=22 Win=64256 Len=0 TSval=2288489947 TSecr=3831496548 |
| 17 | 2.92973. | 192.168.56.101 | 192.168.56.1 | TCP | 4850 → 54552 [ACK] Seq=22 Ack=23 Win=65280 Len=0 TSval=3831497472 TSecr=2288489947 |

```
Frame 6: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
Ethernet II, Src: PcsCompu.77:5c:90 (08:00:27:77:5c:90), Dst: 0a:00:27:00:00:00 (0a:00:27:00:00:00)
Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.1
Transmission Control Protocol, Src Port: 4850, Dst Port: 54552, Seq: 1, Ack: 8, Len: 7
Data (7 bytes)
Data: 6d657373616765
[Length: 7]
```

```
00 0a 00 27 00 00 00 00 00 27 77 5c 90 08 00 45 00 --'\w...E...
10 00 3b a6 39 40 00 40 06 9a cc c0 a8 38 65 c0 a8 --;@:..8e...
20 38 01 12 f2 d5 18 6e ad 96 79 9b 3f 34 8f 80 18 8.....y:74...
30 01 fe 30 f8 00 00 01 01 08 0a e4 5f f7 90 88 67 --@.....g...
40 89 67 6d 65 73 73 61 67 65 --message
```

4 Task 4

| | | | | | |
|----|----------|----------------|----------------|-----|----------------------------|
| 1 | 0.000000 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |
| 2 | 0.000120 | 192.168.56.101 | 192.168.56.1 | UDP | 1042 4950 → 45130 Len=1000 |
| 3 | 0.000300 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |
| 4 | 0.000540 | 192.168.56.101 | 192.168.56.1 | UDP | 1042 4950 → 45130 Len=1000 |
| 5 | 0.000644 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |
| 6 | 0.000859 | 192.168.56.101 | 192.168.56.1 | UDP | 1042 4950 → 45130 Len=1000 |
| 7 | 0.000975 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |
| 8 | 0.001190 | 192.168.56.101 | 192.168.56.1 | UDP | 1042 4950 → 45130 Len=1000 |
| 9 | 0.001351 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |
| 10 | 0.001581 | 192.168.56.101 | 192.168.56.1 | UDP | 1042 4950 → 45130 Len=1000 |
| 11 | 0.001691 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |
| 12 | 0.001891 | 192.168.56.101 | 192.168.56.1 | UDP | 1042 4950 → 45130 Len=1000 |
| 13 | 0.002001 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |
| 14 | 0.002198 | 192.168.56.101 | 192.168.56.1 | UDP | 1042 4950 → 45130 Len=1000 |
| 15 | 0.002299 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |
| 16 | 0.002505 | 192.168.56.101 | 192.168.56.1 | UDP | 1042 4950 → 45130 Len=1000 |
| 17 | 0.002589 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |
| 18 | 0.002794 | 192.168.56.101 | 192.168.56.1 | UDP | 1042 4950 → 45130 Len=1000 |
| 19 | 0.002918 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |
| 20 | 0.003117 | 192.168.56.101 | 192.168.56.1 | UDP | 1042 4950 → 45130 Len=1000 |
| 21 | 0.003203 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |
| 22 | 0.003406 | 192.168.56.101 | 192.168.56.1 | UDP | 1042 4950 → 45130 Len=1000 |
| 23 | 0.003493 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |
| 24 | 0.003696 | 192.168.56.101 | 192.168.56.1 | UDP | 1042 4950 → 45130 Len=1000 |
| 25 | 0.003779 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |
| 26 | 0.003983 | 192.168.56.101 | 192.168.56.1 | UDP | 1042 4950 → 45130 Len=1000 |
| 27 | 0.010851 | 192.168.56.1 | 192.168.56.101 | UDP | 1042 45130 → 4950 Len=1000 |

Here you can see clearly that UDP is much simpler compared to TCP. No flags no startup packages. The datagrams here are 1042 because I mis-typed the buffer size in the arguments. The message length was 1000.

| | | | | | | | | | | | | | | | | | | |
|----|----------|----------------|----------------|-----|----|-------|---|-------|------------|--------|-----------|-----------|----------|------------------|------------------|-----------------|------------------|--------|
| 1 | 0.000000 | 192.168.56.1 | 192.168.56.101 | TCP | 74 | 46242 | → | 4950 | [SYN] | Seq=0 | Win=64240 | Len=0 | MSS=1460 | SACK_PERM=1 | TSval=2752588752 | TSecr=0 | WS=128 | |
| 2 | 0.000039 | 192.168.56.101 | 192.168.56.1 | TCP | 74 | 4950 | → | 46242 | [SYN, ACK] | Seq=0 | Ack=1 | Win=65160 | Len=0 | MSS=1460 | SACK_PERM=1 | TSval=224731642 | TSecr=2752588752 | WS=128 |
| 3 | 0.000099 | 192.168.56.1 | 192.168.56.101 | TCP | 66 | 46242 | → | 4950 | [ACK] | Seq=1 | Ack=1 | Win=64256 | Len=0 | TSval=2752588753 | TSecr=224731642 | | | |
| 4 | 0.000865 | 192.168.56.1 | 192.168.56.101 | TCP | 71 | 46242 | → | 4950 | [PSH, ACK] | Seq=1 | Ack=1 | Win=64256 | Len=5 | TSval=2752588762 | TSecr=224731642 | | | |
| 5 | 0.009712 | 192.168.56.101 | 192.168.56.1 | TCP | 66 | 4950 | → | 46242 | [ACK] | Seq=1 | Ack=6 | Win=65280 | Len=0 | TSval=224731652 | TSecr=2752588762 | | | |
| 6 | 0.035164 | 192.168.56.101 | 192.168.56.1 | TCP | 71 | 4950 | → | 46242 | [PSH, ACK] | Seq=1 | Ack=6 | Win=65280 | Len=5 | TSval=224731677 | TSecr=2752588762 | | | |
| 7 | 0.035303 | 192.168.56.1 | 192.168.56.101 | TCP | 66 | 46242 | → | 4950 | [ACK] | Seq=6 | Ack=6 | Win=64256 | Len=0 | TSval=2752588788 | TSecr=224731677 | | | |
| 8 | 0.036233 | 192.168.56.1 | 192.168.56.101 | TCP | 71 | 46242 | → | 4950 | [PSH, ACK] | Seq=6 | Ack=6 | Win=64256 | Len=5 | TSval=2752588789 | TSecr=224731677 | | | |
| 9 | 0.036255 | 192.168.56.101 | 192.168.56.1 | TCP | 66 | 4950 | → | 46242 | [ACK] | Seq=6 | Ack=11 | Win=65280 | Len=0 | TSval=224731678 | TSecr=2752588789 | | | |
| 10 | 0.036613 | 192.168.56.101 | 192.168.56.1 | TCP | 71 | 4950 | → | 46242 | [PSH, ACK] | Seq=6 | Ack=11 | Win=65280 | Len=5 | TSval=224731679 | TSecr=2752588789 | | | |
| 11 | 0.036738 | 192.168.56.1 | 192.168.56.101 | TCP | 66 | 46242 | → | 4950 | [ACK] | Seq=11 | Ack=11 | Win=64256 | Len=0 | TSval=2752588789 | TSecr=224731679 | | | |
| 12 | 0.036857 | 192.168.56.1 | 192.168.56.101 | TCP | 71 | 46242 | → | 4950 | [PSH, ACK] | Seq=11 | Ack=11 | Win=64256 | Len=5 | TSval=2752588790 | TSecr=224731679 | | | |
| 13 | 0.036864 | 192.168.56.101 | 192.168.56.1 | TCP | 66 | 4950 | → | 46242 | [ACK] | Seq=11 | Ack=16 | Win=65280 | Len=0 | TSval=224731679 | TSecr=2752588790 | | | |
| 14 | 0.037126 | 192.168.56.101 | 192.168.56.1 | TCP | 71 | 4950 | → | 46242 | [PSH, ACK] | Seq=11 | Ack=16 | Win=65280 | Len=5 | TSval=224731679 | TSecr=2752588790 | | | |
| 15 | 0.037194 | 192.168.56.1 | 192.168.56.101 | TCP | 66 | 46242 | → | 4950 | [ACK] | Seq=16 | Ack=16 | Win=64256 | Len=0 | TSval=2752588790 | TSecr=224731679 | | | |
| 16 | 0.037358 | 192.168.56.1 | 192.168.56.101 | TCP | 71 | 46242 | → | 4950 | [PSH, ACK] | Seq=16 | Ack=16 | Win=64256 | Len=5 | TSval=2752588790 | TSecr=224731679 | | | |
| 17 | 0.037364 | 192.168.56.101 | 192.168.56.1 | TCP | 66 | 4950 | → | 46242 | [ACK] | Seq=16 | Ack=21 | Win=65280 | Len=0 | TSval=224731679 | TSecr=2752588790 | | | |
| 18 | 0.037575 | 192.168.56.101 | 192.168.56.1 | TCP | 71 | 4950 | → | 46242 | [PSH, ACK] | Seq=16 | Ack=21 | Win=65280 | Len=5 | TSval=224731680 | TSecr=2752588790 | | | |
| 19 | 0.037641 | 192.168.56.1 | 192.168.56.101 | TCP | 66 | 46242 | → | 4950 | [ACK] | Seq=21 | Ack=21 | Win=64256 | Len=0 | TSval=2752588790 | TSecr=224731680 | | | |
| 20 | 0.037743 | 192.168.56.1 | 192.168.56.101 | TCP | 71 | 46242 | → | 4950 | [PSH, ACK] | Seq=21 | Ack=21 | Win=64256 | Len=5 | TSval=2752588790 | TSecr=224731680 | | | |
| 21 | 0.037749 | 192.168.56.101 | 192.168.56.1 | TCP | 66 | 4950 | → | 46242 | [ACK] | Seq=21 | Ack=26 | Win=65280 | Len=0 | TSval=224731680 | TSecr=2752588790 | | | |
| 22 | 0.037961 | 192.168.56.101 | 192.168.56.1 | TCP | 71 | 4950 | → | 46242 | [PSH, ACK] | Seq=21 | Ack=26 | Win=65280 | Len=5 | TSval=224731680 | TSecr=2752588790 | | | |
| 23 | 0.038115 | 192.168.56.1 | 192.168.56.101 | TCP | 66 | 46242 | → | 4950 | [ACK] | Seq=26 | Ack=26 | Win=64256 | Len=0 | TSval=2752588791 | TSecr=224731680 | | | |
| 24 | 0.038121 | 192.168.56.1 | 192.168.56.101 | TCP | 71 | 46242 | → | 4950 | [PSH, ACK] | Seq=26 | Ack=26 | Win=64256 | Len=5 | TSval=2752588791 | TSecr=224731680 | | | |
| 25 | 0.038125 | 192.168.56.101 | 192.168.56.1 | TCP | 66 | 4950 | → | 46242 | [ACK] | Seq=26 | Ack=31 | Win=65280 | Len=0 | TSval=224731680 | TSecr=2752588791 | | | |
| 26 | 0.038306 | 192.168.56.101 | 192.168.56.1 | TCP | 71 | 4950 | → | 46242 | [PSH, ACK] | Seq=26 | Ack=31 | Win=65280 | Len=5 | TSval=224731680 | TSecr=2752588791 | | | |
| 27 | 0.038366 | 192.168.56.1 | 192.168.56.101 | TCP | 66 | 46242 | → | 4950 | [ACK] | Seq=31 | Ack=31 | Win=64256 | Len=0 | TSval=2752588791 | TSecr=224731680 | | | |

TCP is much more complicated. The difference is connection oriented vs connectionless. Stream based vs datagram based. PSH is a Push flag, ACK is acknowledge the transmission of a packet. SYN is a TCP packet sent to another computer requesting that a connection be established between them. If the SYN is received by the second machine, an SYN/ACK is sent back to the address requested by the SYN.

5 Task 5

UDP traffic

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|----------|----------------|----------------|----------|--------|---------------------|
| 1 | 0.000000 | 192.168.56.1 | 192.168.56.101 | UDP | 18 | 42034 → 4950 Len=10 |
| 2 | 0.000407 | 192.168.56.101 | 192.168.56.1 | UDP | 9 | 4950 → 42034 Len=1 |
| 3 | 0.000848 | 192.168.56.1 | 192.168.56.101 | UDP | 18 | 42034 → 4950 Len=10 |
| 4 | 0.036613 | 192.168.56.101 | 192.168.56.1 | UDP | 9 | 4950 → 42034 Len=1 |
| 5 | 0.037065 | 192.168.56.1 | 192.168.56.101 | UDP | 18 | 42034 → 4950 Len=10 |
| 6 | 0.037828 | 192.168.56.101 | 192.168.56.1 | UDP | 9 | 4950 → 42034 Len=1 |
| 7 | 0.038173 | 192.168.56.1 | 192.168.56.101 | UDP | 18 | 42034 → 4950 Len=10 |
| 8 | 0.038886 | 192.168.56.101 | 192.168.56.1 | UDP | 9 | 4950 → 42034 Len=1 |
| 9 | 0.039185 | 192.168.56.1 | 192.168.56.101 | UDP | 18 | 42034 → 4950 Len=10 |
| 10 | 0.040567 | 192.168.56.101 | 192.168.56.1 | UDP | 9 | 4950 → 42034 Len=1 |

> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)

> Ethernet II, Src: 0a:00:27:00:00:00 (0a:00:27:00:00:00), Dst: PcsCompu_77:5c:90 (08:00:27:77:5c:90)

> Internet Protocol Version 4, Src: 192.168.56.1, Dst: 192.168.56.101

✓ User Datagram Protocol, Src Port: 42034, Dst Port: 4950

- Source Port: 42034
- Destination Port: 4950
- Length: 18
- Checksum: 0x6fa3 [unverified]
- [Checksum Status: Unverified]
- [Stream index: 0]

✓ Data (10 bytes)

- Data: 61616161616161616161
- [Length: 10]

1. Client - 192.168.56.1
2. Server - 192.168.56.101

Packet number 1 is the packet that is sent by the client to the server. It's total length is 18 bytes. 8 bytes for the UDP header and 10 bytes for the payload. The payload is represented in HEX, in ASCII it is a string of 10 'a' chars. You can see that the port of the server is 4950 and the port of the client is 42034, an automatically generated one. For this task I used a buffer for the server of 1 byte. The server receives the message, discards the rest of the bytes, and sends the first byte back. This can be seen in packet number 2.

TCP traffic

1. Client - 192.168.56.1
2. Server - 192.168.56.101

| No. | Time | Source | Destination | Protocol | Info |
|-----|----------|----------------|----------------|----------|---|
| 1 | 0.000000 | 192.168.56.1 | 192.168.56.101 | TCP | 47368 → 4950 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2760570283 TSecr=0 WS |
| 2 | 0.000060 | 192.168.56.101 | 192.168.56.1 | TCP | 4950 → 47368 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2945979817 |
| 3 | 0.000176 | 192.168.56.1 | 192.168.56.101 | TCP | 47368 → 4950 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2760570283 TSecr=2945979817 |
| 4 | 0.001128 | 192.168.56.1 | 192.168.56.101 | TCP | 47368 → 4950 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=10 TSval=2760570284 TSecr=2945979817 |
| 5 | 0.001153 | 192.168.56.101 | 192.168.56.1 | TCP | 4950 → 47368 [ACK] Seq=1 Ack=11 Win=65152 Len=0 TSval=2945979818 TSecr=2760570284 |
| 6 | 0.037513 | 192.168.56.101 | 192.168.56.1 | TCP | 4950 → 47368 [PSH, ACK] Seq=1 Ack=11 Win=65152 Len=1 TSval=2945979854 TSecr=2760570284 |
| 7 | 0.037671 | 192.168.56.1 | 192.168.56.101 | TCP | 47368 → 4950 [ACK] Seq=11 Ack=2 Win=64256 Len=0 TSval=2760570320 TSecr=2945979854 |
| 8 | 0.037968 | 192.168.56.1 | 192.168.56.101 | TCP | 47368 → 4950 [PSH, ACK] Seq=11 Ack=2 Win=64256 Len=10 TSval=2760570321 TSecr=2945979854 |
| 9 | 0.037981 | 192.168.56.101 | 192.168.56.1 | TCP | 4950 → 47368 [ACK] Seq=2 Ack=21 Win=65152 Len=0 TSval=2945979855 TSecr=2760570321 |
| 10 | 0.038241 | 192.168.56.101 | 192.168.56.1 | TCP | 4950 → 47368 [PSH, ACK] Seq=2 Ack=21 Win=65152 Len=1 TSval=2945979855 TSecr=2760570321 |
| 11 | 0.038349 | 192.168.56.1 | 192.168.56.101 | TCP | 47368 → 4950 [ACK] Seq=21 Ack=3 Win=64256 Len=0 TSval=2760570321 TSecr=2945979855 |
| 12 | 0.038479 | 192.168.56.1 | 192.168.56.101 | TCP | 47368 → 4950 [PSH, ACK] Seq=21 Ack=3 Win=64256 Len=10 TSval=2760570321 TSecr=2945979855 |
| 13 | 0.038488 | 192.168.56.101 | 192.168.56.1 | TCP | 4950 → 47368 [ACK] Seq=3 Ack=31 Win=65152 Len=0 TSval=2945979855 TSecr=2760570321 |
| 14 | 0.038856 | 192.168.56.101 | 192.168.56.1 | TCP | 4950 → 47368 [PSH, ACK] Seq=3 Ack=31 Win=65152 Len=1 TSval=2945979856 TSecr=2760570321 |
| 15 | 0.038948 | 192.168.56.1 | 192.168.56.101 | TCP | 47368 → 4950 [ACK] Seq=31 Ack=4 Win=64256 Len=0 TSval=2760570322 TSecr=2945979856 |
| 16 | 0.039072 | 192.168.56.1 | 192.168.56.101 | TCP | 47368 → 4950 [PSH, ACK] Seq=31 Ack=4 Win=64256 Len=10 TSval=2760570322 TSecr=2945979856 |

```

> Frame 4: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)
> Ethernet II, Src: 0a:00:27:00:00:00 (0a:00:27:00:00:00), Dst: PcsCompu_77:5c:90 (08:00:27:77:5c:90)
> Internet Protocol Version 4, Src: 192.168.56.1, Dst: 192.168.56.101
> Transmission Control Protocol, Src Port: 47368, Dst Port: 4950, Seq: 1, Ack: 1, Len: 10
  - Source Port: 47368
  - Destination Port: 4950
  - [Stream index: 0]
  - [TCP Segment Len: 10]
  - Sequence number: 1 (relative sequence number)
  - [Next sequence number: 11 (relative sequence number)]
  - Acknowledgment number: 1 (relative ack number)
  - 1000 .... = Header Length: 32 bytes (8)
  - Flags: 0x018 (PSH, ACK)
  - Window size value: 502
  - [Calculated window size: 64256]
  - [Window size scaling factor: 128]
  - Checksum: 0x9eae [unverified]
  - [Checksum Status: Unverified]
  - Urgent pointer: 0
  - Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  - [SEQ/ACK analysis]
  - [Timestamps]
  - TCP payload (10 bytes)
  - Data (10 bytes)
    - Data: 61616161616161616161
    - [Length: 10]

```

This is much more complicated. The first 3 packets represent the 3 way handshake. In the packet number 4, the client sends a message with a length of 10 bytes. Here you can see that the PSH flag is enabled. In the packet number 5, the server replies with a message with an ACK. In the packet number 6 the server sends the client a message with a length of 1 bytes. This is for the same reason as before, the buffer is 1, the server discards every byte that does not fit in the browser. In the picture there is no FIN flag but the FIN flag indicates the end of data transmission to finish a TCP connection.

5.1 Difference between UDP and TCP

The difference between TCP and UDP is, connection oriented vs connection-less. TCP is slower, UDP is faster. TCP guarantees the order of the packets, UDP does not. TCP is reliable, UDP is not. TCP has a bigger header size, UDP smaller. TCP checks for errors, UDP does not.