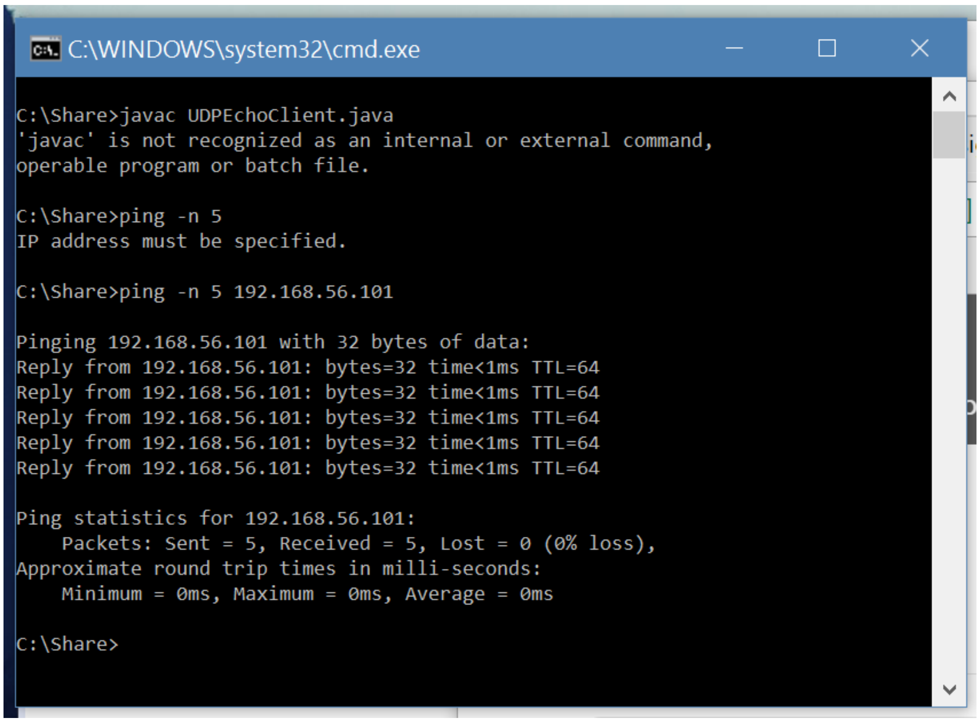
1dv701

Assignment 1

Henry Pap

hp222fq@student.lnu.se

Problem 1



```
C:\Share>javac UDPEchoClient.java
'javac' is not recognized as an internal or external command,
operable program or batch file.

C:\Share>ping -n 5
IP address must be specified.

C:\Share>ping -n 5 192.168.56.101

Pinging 192.168.56.101 with 32 bytes of data:
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.56.101:
    Packets: Sent = 5, Received = 5, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Share>
```

Shows that the setup works.

Problem 2

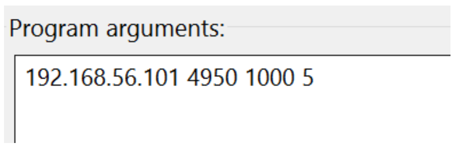
Since I started with an abstract class, all the error handling plus additional information (e.g. rate, bufsize etc.) are within the class “Transport”. This would make it easy for the next part (TCP), also I like structure (why I did abstract first).

The abstract class consist of the constructor (that handle the arguments with all of its errors), getters (the argument values), IP validator and a Verify method to verify the message with the received message. There is a third method “printErr”, but it is not very important.

Send with the transfer rate of 5:

```
201ms 16 bytes sent and received
201ms 16 bytes sent and received
199ms 16 bytes sent and received
202ms 16 bytes sent and received
198ms 16 bytes sent and received
The operation took: 1032ms, +32ms
```

Arguments: (always the same)


Program arguments:

192.168.56.101 4950 1000 5

The argument errors:

1. valid IP
2. rate is > 0, if 0 then 1 and also an integer
3. buffsize, min 0 max 1450 (books says 1450 is a good) and also an integer
4. port number is >= 0 && <= 65535, also that it is an integer

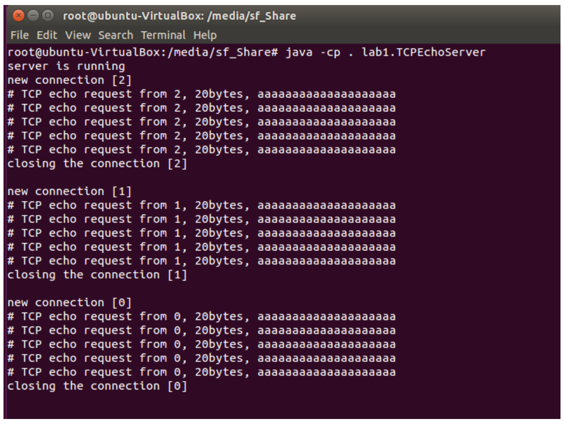
VG task 1 is ignored for now, just calculate the average value to send and take that into account, and on the last packet, check how much time left minus the average. This should work, but as loops and other codes takes time, there will never be exact. I never had a chance to ask the teacher on the class, so I decided not to further investigate in it (a student took really long time).

Problem 3

TCP with multiple connections.

In the report I will use 3 connections (in order to fit into a small image).

Server:


root@ubuntu-VirtualBox: /media/sf_Share
File Edit View Search Terminal Help

```
root@ubuntu-VirtualBox:/media/sf_Share# java -cp . lab1.TCPEchoServer
server is running
new connection [2]
# TCP echo request from 2, 20bytes, aaaaaaaaaaaaaaaaaaaa
# TCP echo request from 2, 20bytes, aaaaaaaaaaaaaaaaaaaa
# TCP echo request from 2, 20bytes, aaaaaaaaaaaaaaaaaaaa
# TCP echo request from 2, 20bytes, aaaaaaaaaaaaaaaaaaaa
# TCP echo request from 2, 20bytes, aaaaaaaaaaaaaaaaaaaa
closing the connection [2]

new connection [1]
# TCP echo request from 1, 20bytes, aaaaaaaaaaaaaaaaaaaa
# TCP echo request from 1, 20bytes, aaaaaaaaaaaaaaaaaaaa
# TCP echo request from 1, 20bytes, aaaaaaaaaaaaaaaaaaaa
# TCP echo request from 1, 20bytes, aaaaaaaaaaaaaaaaaaaa
# TCP echo request from 1, 20bytes, aaaaaaaaaaaaaaaaaaaa
closing the connection [1]

new connection [0]
# TCP echo request from 0, 20bytes, aaaaaaaaaaaaaaaaaaaa
# TCP echo request from 0, 20bytes, aaaaaaaaaaaaaaaaaaaa
# TCP echo request from 0, 20bytes, aaaaaaaaaaaaaaaaaaaa
# TCP echo request from 0, 20bytes, aaaaaaaaaaaaaaaaaaaa
# TCP echo request from 0, 20bytes, aaaaaaaaaaaaaaaaaaaa
closing the connection [0]
```

Client:

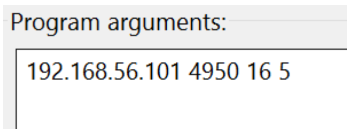
```
start
[1] : connected
20bytes sent and received msg are equal
20bytes sent and received msg are equal
20bytes sent and received msg are equal
20bytes sent and received msg are equal
20bytes sent and received msg are equal
[1] : closed

[0] : connected
20bytes sent and received msg are equal
20bytes sent and received msg are equal
20bytes sent and received msg are equal
20bytes sent and received msg are equal
20bytes sent and received msg are equal
[0] : closed

[2] : connected
20bytes sent and received msg are equal
20bytes sent and received msg are equal
20bytes sent and received msg are equal
20bytes sent and received msg are equal
20bytes sent and received msg are equal
[2] : closed
```

Sending a message that is larger than the buffer size

New arguments (buffer size: 16)



Program arguments:

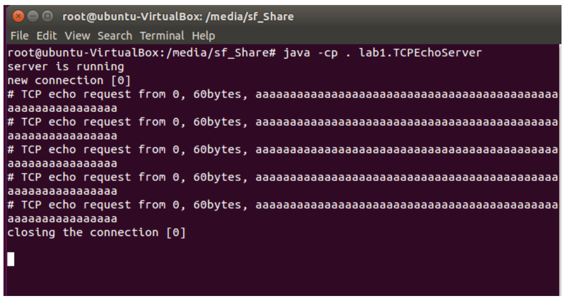
```
192.168.56.101 4950 16 5
```

And a text message set to 60bytes

```
private String ip, MSG = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"; // the message to be sent
```

For TCP (only 1 client)

Server:



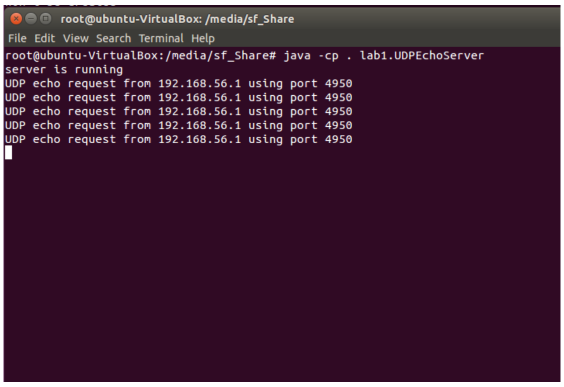
```
root@ubuntu-VirtualBox:/media/sf_Share# java -cp . lab1.TCPEchoServer
server is running
new connection [0]
# TCP echo request from 0, 60bytes, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa
# TCP echo request from 0, 60bytes, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa
# TCP echo request from 0, 60bytes, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa
# TCP echo request from 0, 60bytes, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa
# TCP echo request from 0, 60bytes, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa
closing the connection [0]
```

Client:

```
start
[0] : connected
60bytes sent and received msg are equal
60bytes sent and received msg are equal
60bytes sent and received msg are equal
60bytes sent and received msg are equal
60bytes sent and received msg are equal
[0] : closed
```

For UDP

Server:



```
root@ubuntu-VirtualBox:/media/sf_Share# java -cp . lab1.UDPEchoServer
server is running
UDP echo request from 192.168.56.1 using port 4950
UDP echo request from 192.168.56.1 using port 4950
UDP echo request from 192.168.56.1 using port 4950
UDP echo request from 192.168.56.1 using port 4950
UDP echo request from 192.168.56.1 using port 4950
```

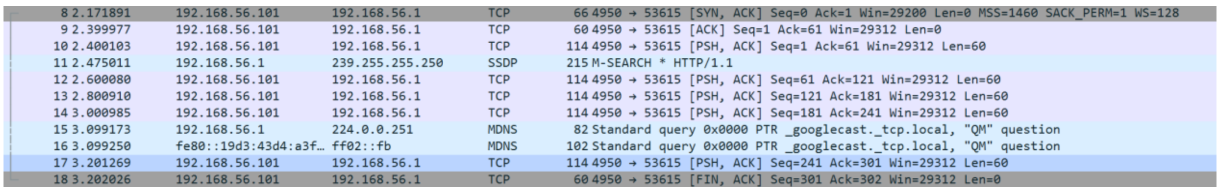
Client

```
200ms Sent and received msg not equal
200ms Sent and received msg not equal
200ms Sent and received msg not equal
201ms Sent and received msg not equal
200ms Sent and received msg not equal
The operation took: 1027ms, +27ms
```

There’s a major difference in UDP and TCP, while UDP fails to receive the message, TCP manages to receive the full packet thanks to its underlying window frame.

Problem 4

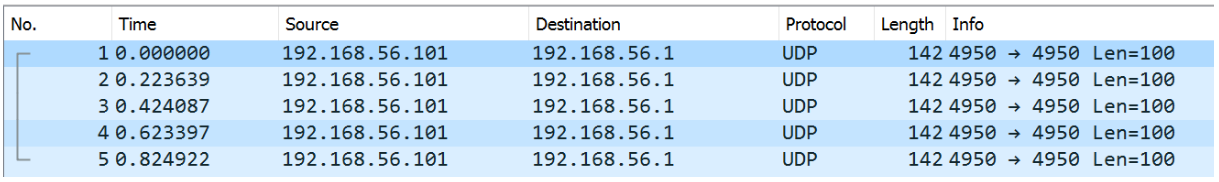
TCP network capture using Wireshark (1 client, 5 message sent)



8	2.171891	192.168.56.101	192.168.56.1	TCP	66	4950 → 53615 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
9	2.399977	192.168.56.101	192.168.56.1	TCP	60	4950 → 53615 [ACK] Seq=1 Ack=61 Win=29312 Len=0
10	2.400103	192.168.56.101	192.168.56.1	TCP	114	4950 → 53615 [PSH, ACK] Seq=1 Ack=61 Win=29312 Len=60
11	2.475011	192.168.56.1	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
12	2.600080	192.168.56.101	192.168.56.1	TCP	114	4950 → 53615 [PSH, ACK] Seq=61 Ack=121 Win=29312 Len=60
13	2.800910	192.168.56.101	192.168.56.1	TCP	114	4950 → 53615 [PSH, ACK] Seq=121 Ack=181 Win=29312 Len=60
14	3.000985	192.168.56.101	192.168.56.1	TCP	114	4950 → 53615 [PSH, ACK] Seq=181 Ack=241 Win=29312 Len=60
15	3.099173	192.168.56.1	224.0.0.251	MDNS	82	Standard query 0x0000 PTR _googlecast._tcp.local, "QM" question
16	3.099250	fe80::19d3:43d4:a3f…	ff02::fb	MDNS	102	Standard query 0x0000 PTR _googlecast._tcp.local, "QM" question
17	3.201269	192.168.56.101	192.168.56.1	TCP	114	4950 → 53615 [PSH, ACK] Seq=241 Ack=301 Win=29312 Len=60
18	3.202026	192.168.56.101	192.168.56.1	TCP	60	4950 → 53615 [FIN, ACK] Seq=301 Ack=302 Win=29312 Len=0

Can’t see the three way handshake, this because I use windows, or the last ACK is embedded with a PSH.

UDP:

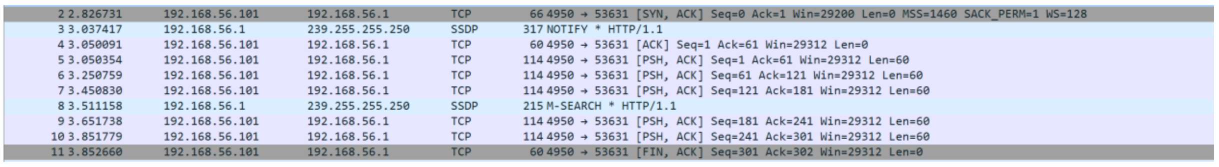


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.56.101	192.168.56.1	UDP	142	4950 → 4950 Len=100
2	0.223639	192.168.56.101	192.168.56.1	UDP	142	4950 → 4950 Len=100
3	0.424087	192.168.56.101	192.168.56.1	UDP	142	4950 → 4950 Len=100
4	0.623397	192.168.56.101	192.168.56.1	UDP	142	4950 → 4950 Len=100
5	0.824922	192.168.56.101	192.168.56.1	UDP	142	4950 → 4950 Len=100

All five messages sent is clearly shown.

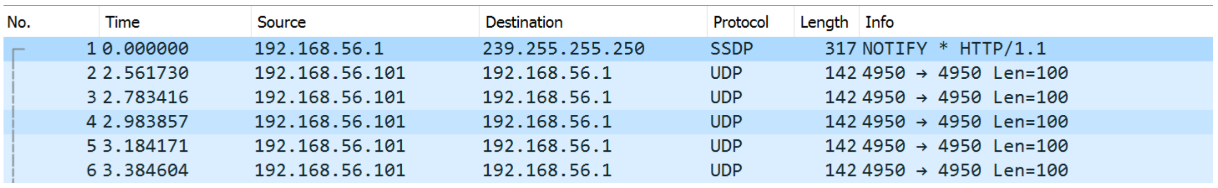
With lower client buffer size then message:

TCP



2	2.826731	192.168.56.101	192.168.56.1	TCP	66	4950 → 53631 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
3	3.037417	192.168.56.1	239.255.255.250	SSDP	317	NOTIFY * HTTP/1.1
4	3.050091	192.168.56.101	192.168.56.1	TCP	60	4950 → 53631 [ACK] Seq=1 Ack=61 Win=29312 Len=0
5	3.050354	192.168.56.101	192.168.56.1	TCP	114	4950 → 53631 [PSH, ACK] Seq=1 Ack=61 Win=29312 Len=60
6	3.250759	192.168.56.101	192.168.56.1	TCP	114	4950 → 53631 [PSH, ACK] Seq=61 Ack=121 Win=29312 Len=60
7	3.450830	192.168.56.101	192.168.56.1	TCP	114	4950 → 53631 [PSH, ACK] Seq=121 Ack=181 Win=29312 Len=60
8	3.511158	192.168.56.1	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
9	3.651738	192.168.56.101	192.168.56.1	TCP	114	4950 → 53631 [PSH, ACK] Seq=181 Ack=241 Win=29312 Len=60
10	3.851779	192.168.56.101	192.168.56.1	TCP	114	4950 → 53631 [PSH, ACK] Seq=241 Ack=301 Win=29312 Len=60
11	3.852660	192.168.56.101	192.168.56.1	TCP	60	4950 → 53631 [FIN, ACK] Seq=301 Ack=302 Win=29312 Len=0

UDP



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.56.1	239.255.255.250	SSDP	317	NOTIFY * HTTP/1.1
2	2.561730	192.168.56.101	192.168.56.1	UDP	142	4950 → 4950 Len=100
3	2.783416	192.168.56.101	192.168.56.1	UDP	142	4950 → 4950 Len=100
4	2.983857	192.168.56.101	192.168.56.1	UDP	142	4950 → 4950 Len=100
5	3.184171	192.168.56.101	192.168.56.1	UDP	142	4950 → 4950 Len=100
6	3.384604	192.168.56.101	192.168.56.1	UDP	142	4950 → 4950 Len=100

There is no differ. Although since I’m using windows 10, I can’t see the client to server side, only server to client side. I guess that the sender (me) would see many ACK.