1dv701

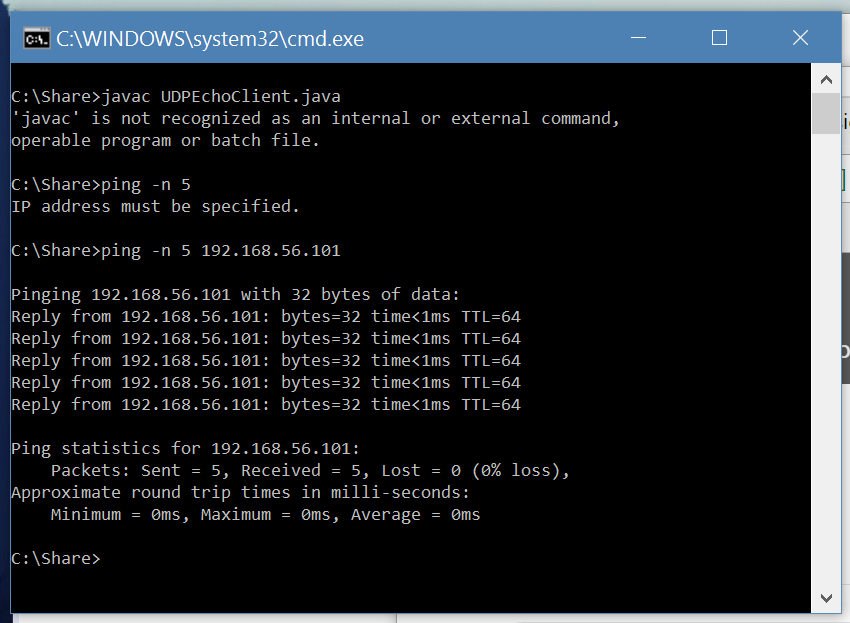
Assignment 1

Henry Pap ­

[hp222fq@student.lnu.se](mailto:hp222fq@student.lnu.se)

­­­

Problem 1



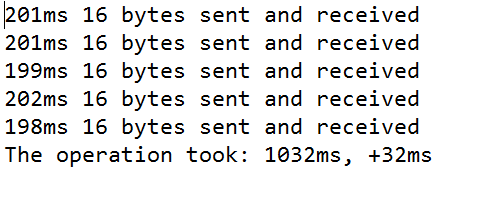
Shows that the setup works.

Problem 2

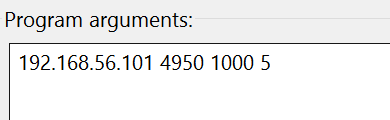
Since I started with an abstract class, all the error handling plus additional information (e.g. rate, bufsize etc.) are within the class “Transport”. This would make it easy for the next part (TCP), also I like structure (why I did abstract first).

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:



Arguments: (always the same)



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

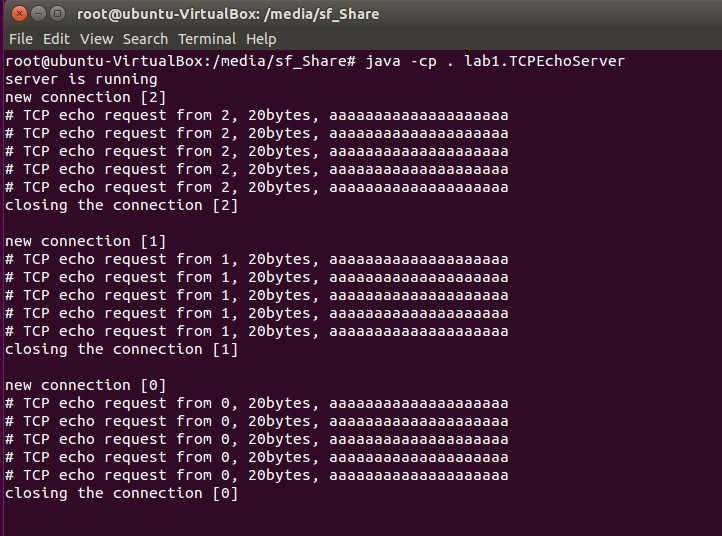
VG task 1 is ignored for now, just calculate the average value to send and take that into account, and on the last packet, check how much time left minus the average. This should work, but as loops and other codes takes time, there will never be exact. I never had a chance to ask the teacher on the class,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

Problem 3

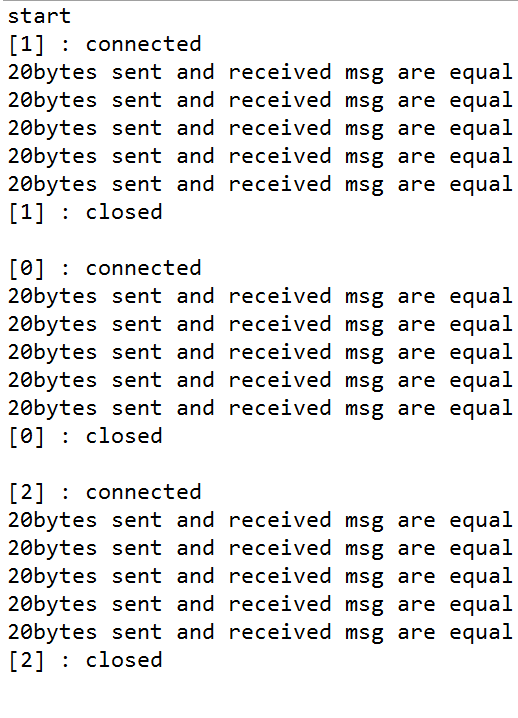
TCP with multiple connections.

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

Server:

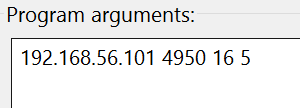


Client:



Sending a message that is larger than the buffer size

New arguments (buffer size: 16)

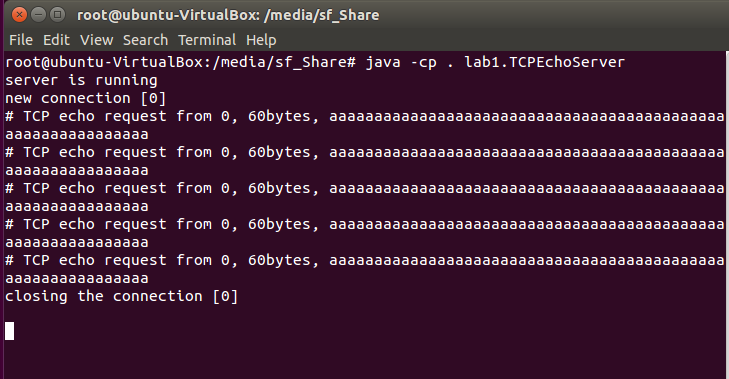


And a text message set to 60bytes

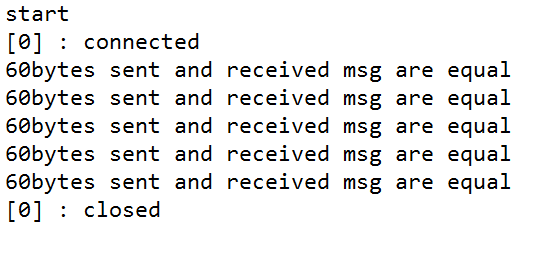


For TCP (only 1 client)

Server:

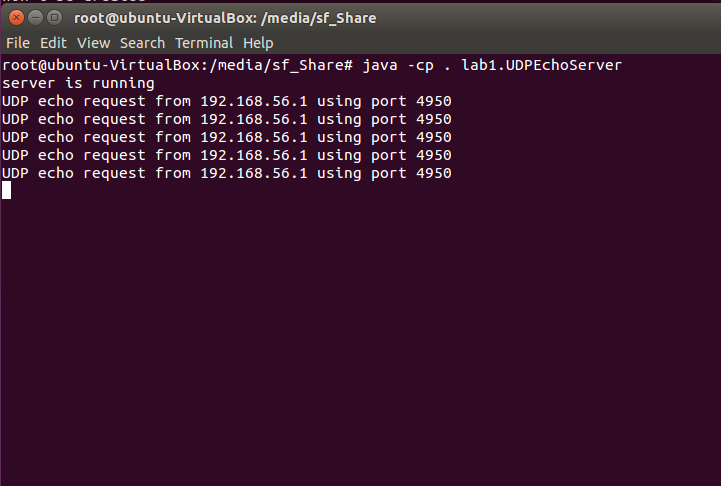


Client:

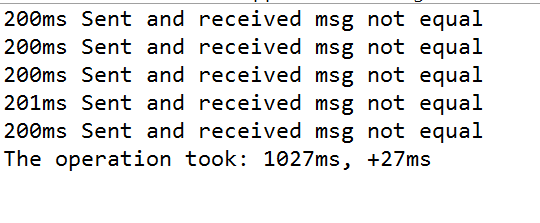


For UDP

Server:



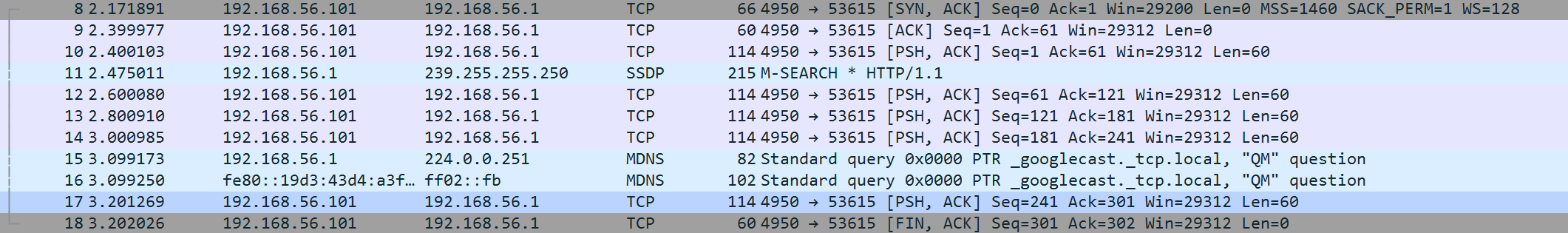
Client



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.

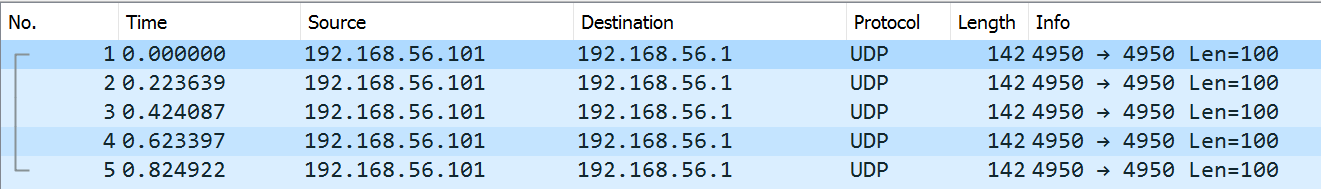
Problem 4

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



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

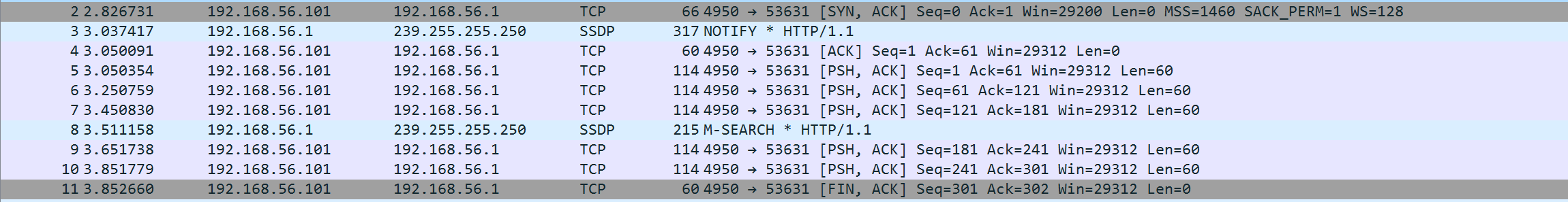
UDP:



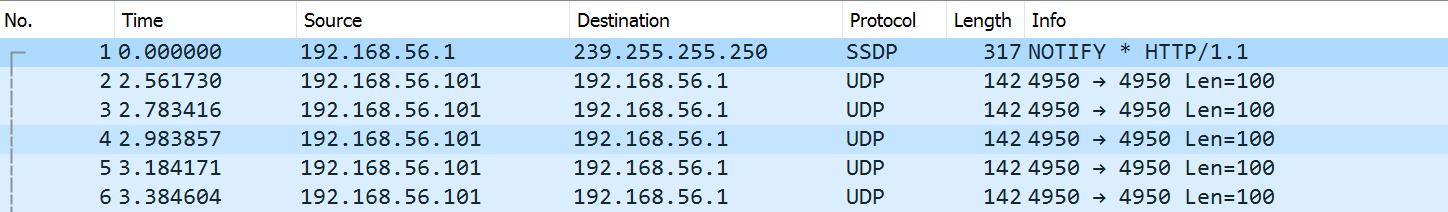
All five messages sent is clearly shown.

With lower client buffer size then message:

TCP



UDP



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.