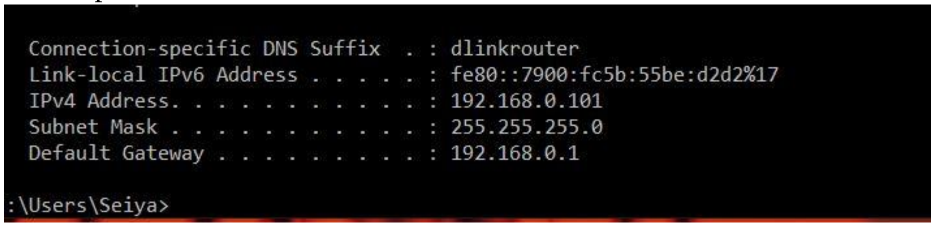
# Assignment one

## Setup:

I had error with Linux as It could not identify the VM Ip address and instead worked with the same Ip as the host machine, so I switched to a second separate windows machines.

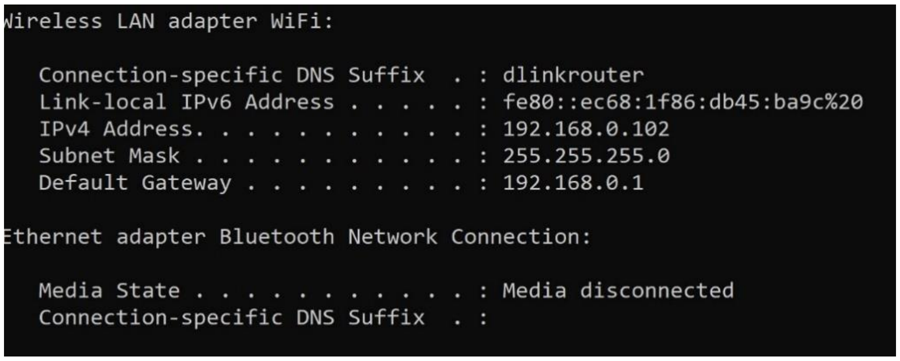
Two windows machines:

- Desktop with the IP address of 192.168.0.101



```
  Connection-specific DNS Suffix  . : dlinkrouter
  Link-local IPv6 Address . . . . . : fe80::7900:fc5b:55be:d2d2%17
  IPv4 Address. . . . . . . . . . . : 192.168.0.101
  Subnet Mask . . . . . . . . . . . : 255.255.255.0
  Default Gateway . . . . . . . . . : 192.168.0.1

:\Users\Seiya>
```

- Laptop with the IP address of 192.168.0.102

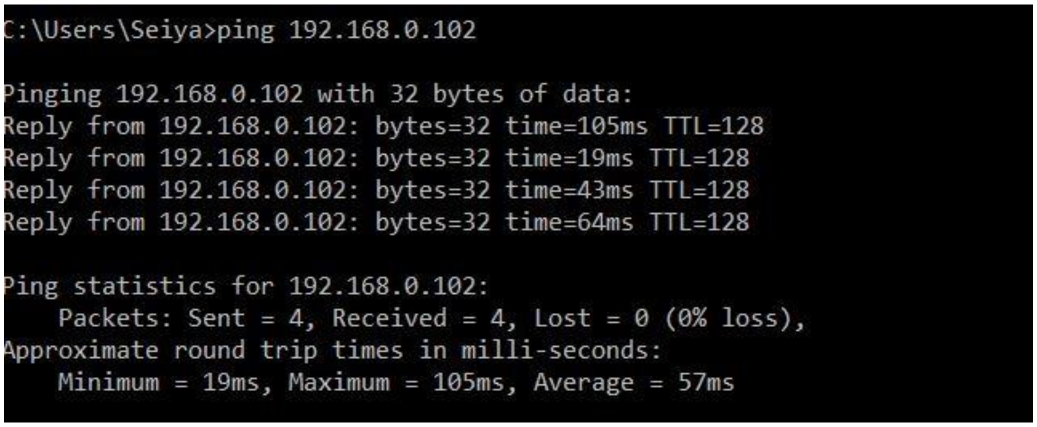


```
ireless LAN adapter WiFi:

   Connection-specific DNS Suffix  . : dlinkrouter
   Link-local IPv6 Address . . . . . : fe80::ec68:1f86:db45:ba9c%20
   IPv4 Address. . . . . . . . . . . : 192.168.0.102
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.0.1

thernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

Arguments inserted in order are:

1. Ip
2. port
3. Buffer size
4. transfer rate

## Problem one:

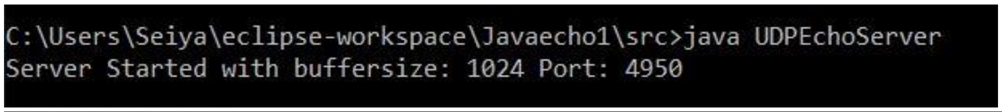


```
C:\Users\Seiya>ping 192.168.0.102

Pinging 192.168.0.102 with 32 bytes of data:
Reply from 192.168.0.102: bytes=32 time=105ms TTL=128
Reply from 192.168.0.102: bytes=32 time=19ms TTL=128
Reply from 192.168.0.102: bytes=32 time=43ms TTL=128
Reply from 192.168.0.102: bytes=32 time=64ms TTL=128

Ping statistics for 192.168.0.102:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 19ms, Maximum = 105ms, Average = 57ms
```

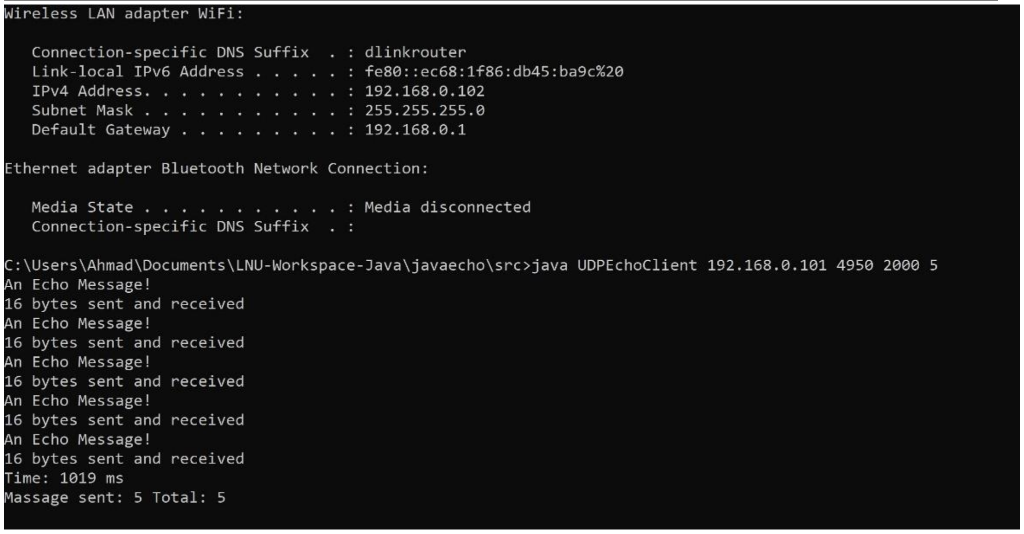
*Figure 1 Ping output*

The figure above shows the ping results pinging from my desktop to my laptop.

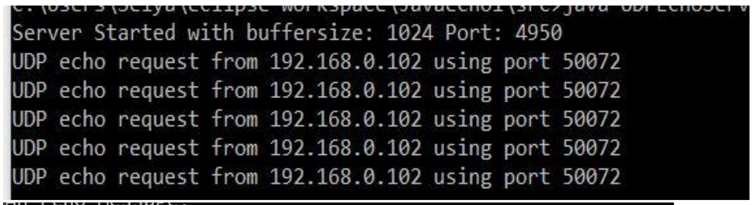
## Problem two:



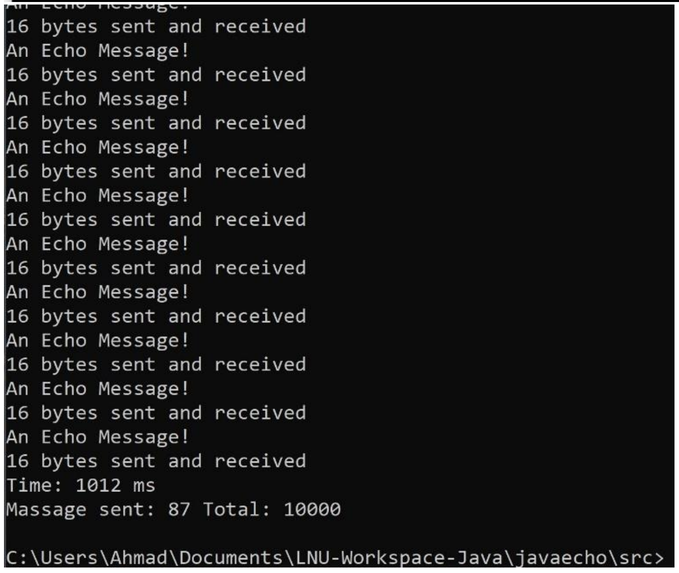
```
C:\Users\Seiya\eclipse-workspace\Javaecho1\src>java UDPEchoServer
Server Started with buffersize: 1024 Port: 4950
```



```
Wireless LAN adapter WiFi:

   Connection-specific DNS Suffix  . : dlinkrouter
   Link-local IPv6 Address . . . . . : fe80::ec68:1f86:db45:ba9c%20
   IPv4 Address. . . . . . . . . . . : 192.168.0.102
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.0.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\Ahmad\Documents\LNU-Workspace-Java\javaecho\src>java UDPEchoClient 192.168.0.101 4950 2000 5
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
Time: 1019 ms
Massage sent: 5 Total: 5
```



```
Server Started with buffersize: 1024 Port: 4950
UDP echo request from 192.168.0.102 using port 50072
UDP echo request from 192.168.0.102 using port 50072
UDP echo request from 192.168.0.102 using port 50072
UDP echo request from 192.168.0.102 using port 50072
UDP echo request from 192.168.0.102 using port 50072
```



```
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
Time: 1012 ms
Massage sent: 87 Total: 10000

C:\Users\Ahmad\Documents\LNU-Workspace-Java\javaecho\src>
```

The first 3 screen shots are when running the Client and sending 5 massages and with the server receiving 5 massages also.

The last screen shot is when I tried sent 10000 messages in one second.

Server: IP: 192.168.0.101 initialized buffer size: 1024 port 4950

Client arguments: IP: 192.168.0.102 port:4950 buffer:2000 massage rate: 5

Client arguments: IP: 192.168.0.102 port:4950 buffer:2000 massage rate: 10000

Handled exceptions and errors:

- IOException
- SocketException
- NumberFormatException
- InterruptedException
- Ip Validation
- Port Validation
- Buffer limitation
- Message is empty
- Message rate validation
- UDP message limited to 63999 in length
- Exception if message is bigger than buffer size

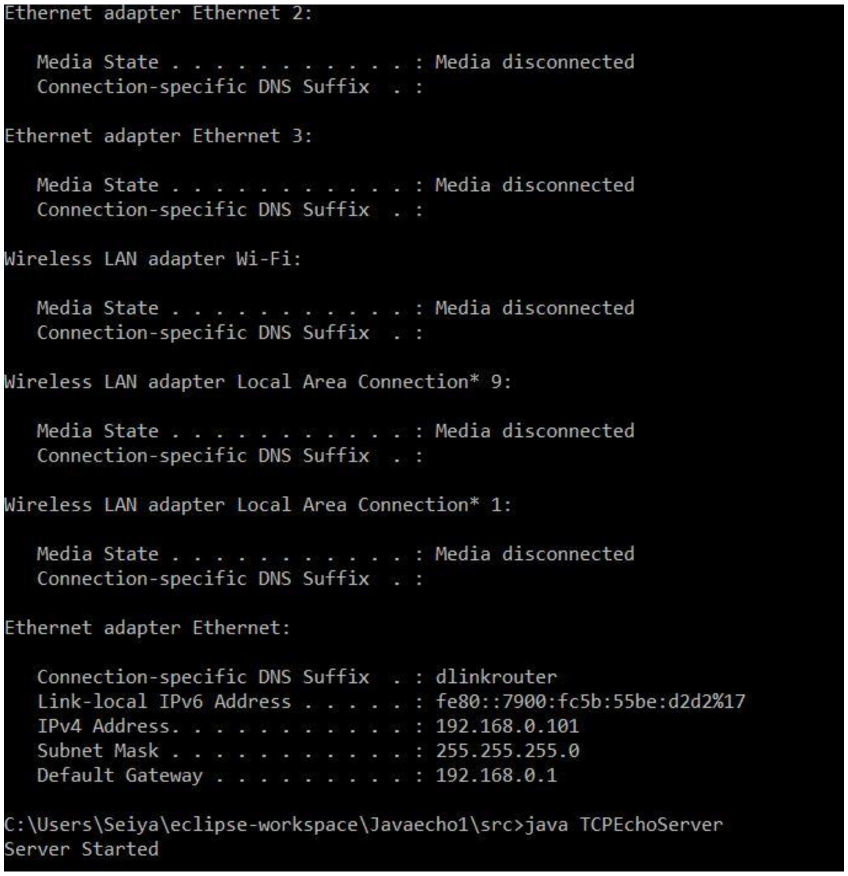
## VG 1:

I spread the messages sent in one second under 45 messages in one second, so when sending for example: 5 messages in one second it spreads the 5 messages within this second instead of sending them all at once in less than once second. when the transfer rate is set above the 45 it will send as much messages as possible then it stopes after one second or if the messages are all sent within this second.

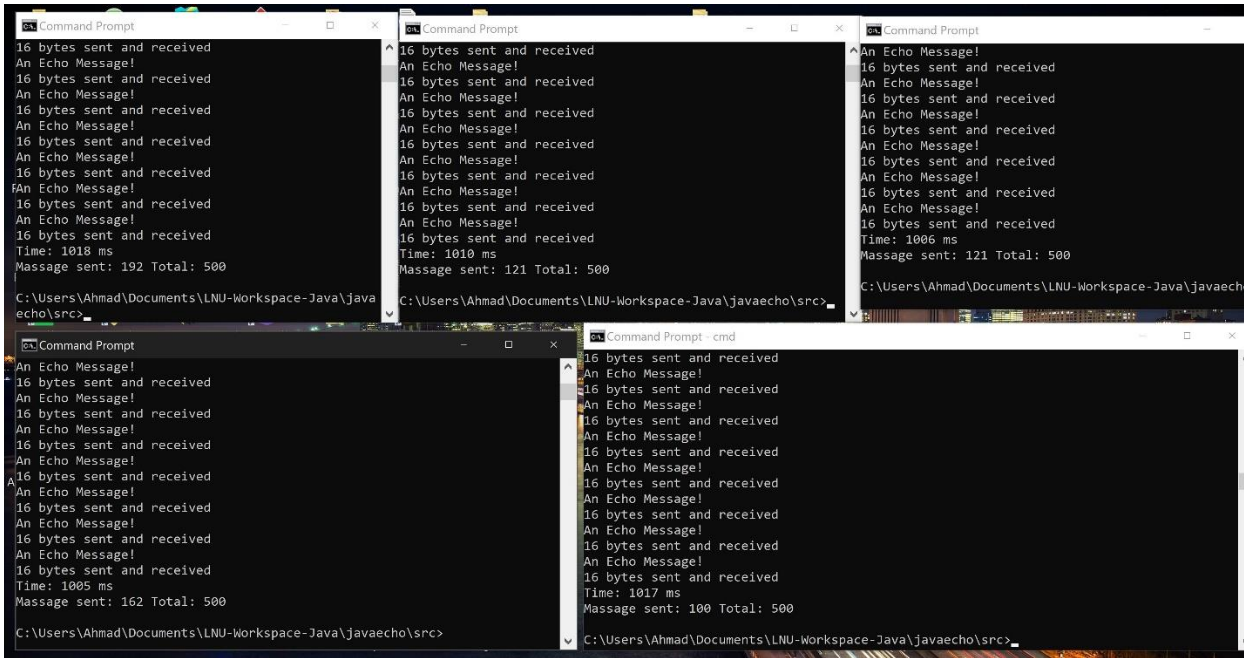
## VG 2:

For the abstract class I created a setters and getters, so I could use them in both TCP and UDP clients and validate the inputs/arguments there an creating an object of UDP or TCP.

## Problem Three:



```
Ethernet adapter Ethernet 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 9:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : dlinkrouter
   Link-local IPv6 Address . . . . . : fe80::7900:fc5b:55be:d2d2%17
   IPv4 Address. . . . . . . . . . . : 192.168.0.101
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.0.1

C:\Users\Seiya\eclipse-workspace\Javaecho1\src>java TCPEchoServer
Server Started
```



Command Prompt

```
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
Time: 1018 ms
Massage sent: 192 Total: 500

C:\Users\Ahmad\Documents\LNU-Workspace-Java\java
echo\src>
```

Command Prompt

```
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
Time: 1010 ms
Massage sent: 121 Total: 500

C:\Users\Ahmad\Documents\LNU-Workspace-Java\javaecho\src>
```

Command Prompt

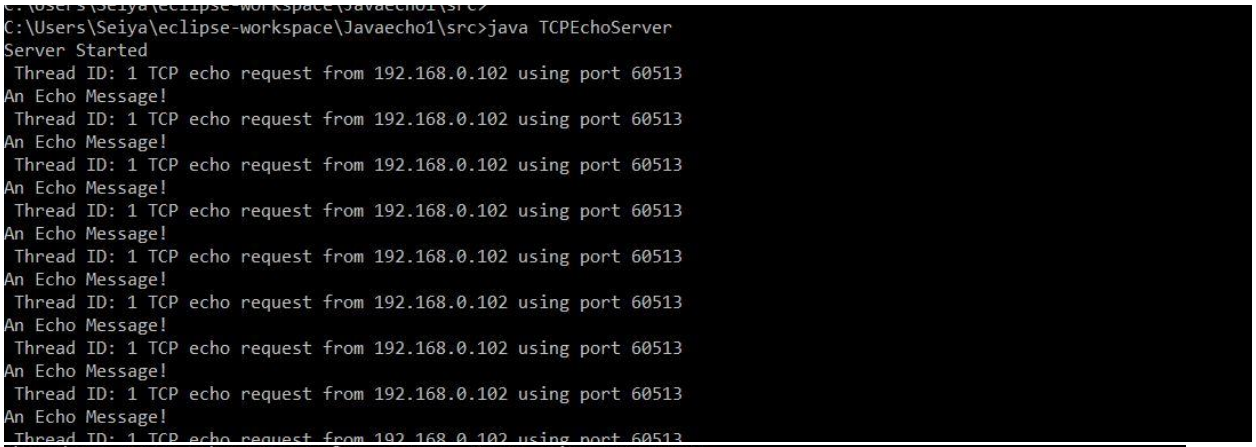
```
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
Time: 1006 ms
Massage sent: 121 Total: 500

C:\Users\Ahmad\Documents\LNU-Workspace-Java\javaech
```

Command Prompt

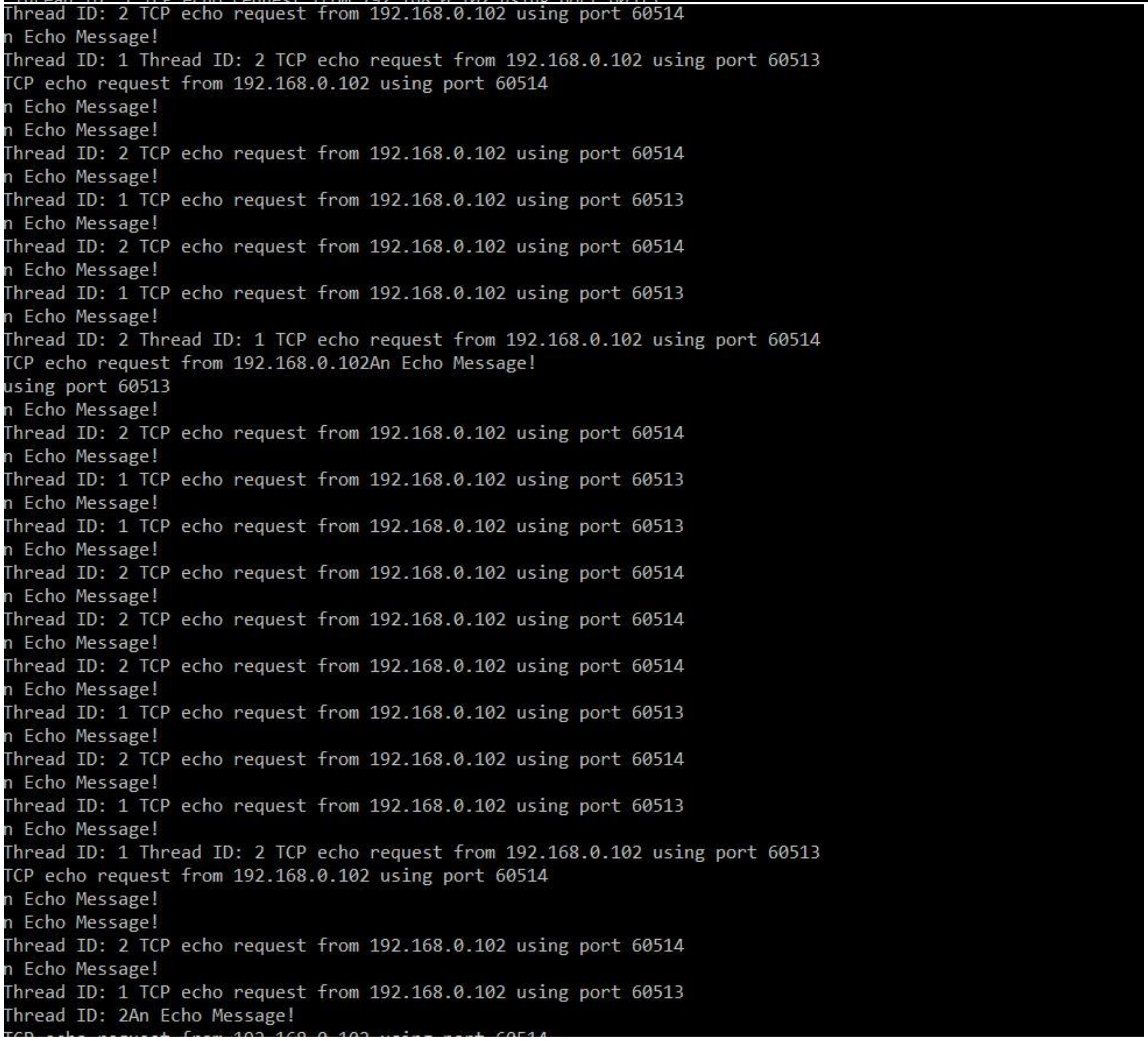
```
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
Time: 1005 ms
Massage sent: 162 Total: 500

C:\Users\Ahmad\Documents\LNU-Workspace-Java\javaecho\src>
```

Command Prompt - cmd

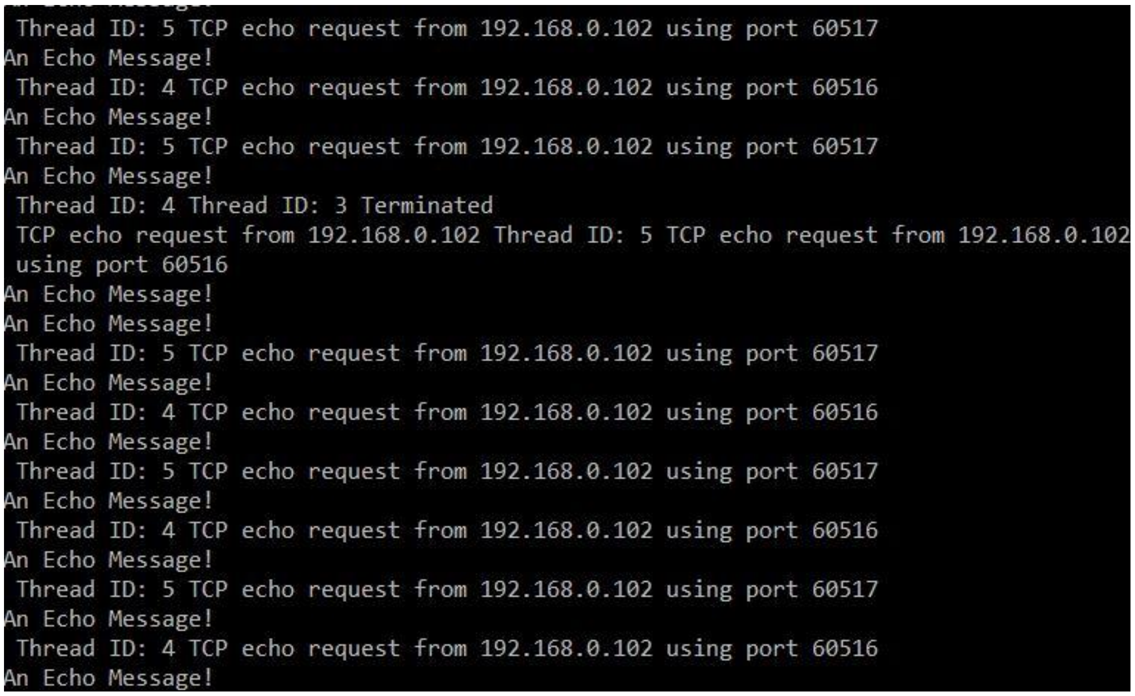
```
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
An Echo Message!
16 bytes sent and received
Time: 1017 ms
Massage sent: 100 Total: 500

C:\Users\Ahmad\Documents\LNU-Workspace-Java\javaecho\src>
```



```
C:\Users\Seiya\eclipse-workspace\Javaecho1\src>
C:\Users\Seiya\eclipse-workspace\Javaecho1\src>java TCPEchoServer
Server Started
 Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
An Echo Message!
 Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
An Echo Message!
 Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
An Echo Message!
 Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
An Echo Message!
 Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
An Echo Message!
 Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
An Echo Message!
 Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
An Echo Message!
 Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
An Echo Message!
 Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
```



```
Thread ID: 2 TCP echo request from 192.168.0.102 using port 60514
n Echo Message!
Thread ID: 1 Thread ID: 2 TCP echo request from 192.168.0.102 using port 60513
TCP echo request from 192.168.0.102 using port 60514
n Echo Message!
n Echo Message!
Thread ID: 2 TCP echo request from 192.168.0.102 using port 60514
n Echo Message!
Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
n Echo Message!
Thread ID: 2 TCP echo request from 192.168.0.102 using port 60514
n Echo Message!
Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
n Echo Message!
Thread ID: 2 Thread ID: 1 TCP echo request from 192.168.0.102 using port 60514
TCP echo request from 192.168.0.102An Echo Message!
using port 60513
n Echo Message!
Thread ID: 2 TCP echo request from 192.168.0.102 using port 60514
n Echo Message!
Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
n Echo Message!
Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
n Echo Message!
Thread ID: 2 TCP echo request from 192.168.0.102 using port 60514
n Echo Message!
Thread ID: 2 TCP echo request from 192.168.0.102 using port 60514
n Echo Message!
Thread ID: 2 TCP echo request from 192.168.0.102 using port 60514
n Echo Message!
Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
n Echo Message!
Thread ID: 2 TCP echo request from 192.168.0.102 using port 60514
n Echo Message!
Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
n Echo Message!
Thread ID: 1 Thread ID: 2 TCP echo request from 192.168.0.102 using port 60513
TCP echo request from 192.168.0.102 using port 60514
n Echo Message!
n Echo Message!
Thread ID: 2 TCP echo request from 192.168.0.102 using port 60514
n Echo Message!
Thread ID: 1 TCP echo request from 192.168.0.102 using port 60513
Thread ID: 2An Echo Message!
TCP echo request from 192.168.0.102 using port 60514
```



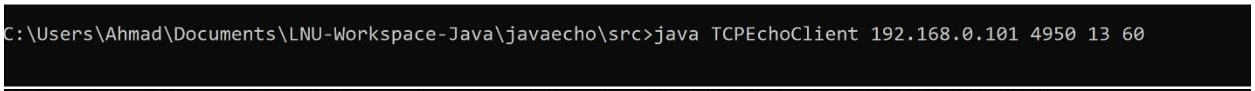
```
 Thread ID: 5 TCP echo request from 192.168.0.102 using port 60517
An Echo Message!
 Thread ID: 4 TCP echo request from 192.168.0.102 using port 60516
An Echo Message!
 Thread ID: 5 TCP echo request from 192.168.0.102 using port 60517
An Echo Message!
 Thread ID: 4 Thread ID: 3 Terminated
 TCP echo request from 192.168.0.102 Thread ID: 5 TCP echo request from 192.168.0.102
 using port 60516
An Echo Message!
An Echo Message!
 Thread ID: 5 TCP echo request from 192.168.0.102 using port 60517
An Echo Message!
 Thread ID: 4 TCP echo request from 192.168.0.102 using port 60516
An Echo Message!
 Thread ID: 5 TCP echo request from 192.168.0.102 using port 60517
An Echo Message!
 Thread ID: 4 TCP echo request from 192.168.0.102 using port 60516
An Echo Message!
 Thread ID: 5 TCP echo request from 192.168.0.102 using port 60517
An Echo Message!
 Thread ID: 4 TCP echo request from 192.168.0.102 using port 60516
An Echo Message!
```

Server: IP: 192.168.0.101 start buffer size: 1024 port 4950

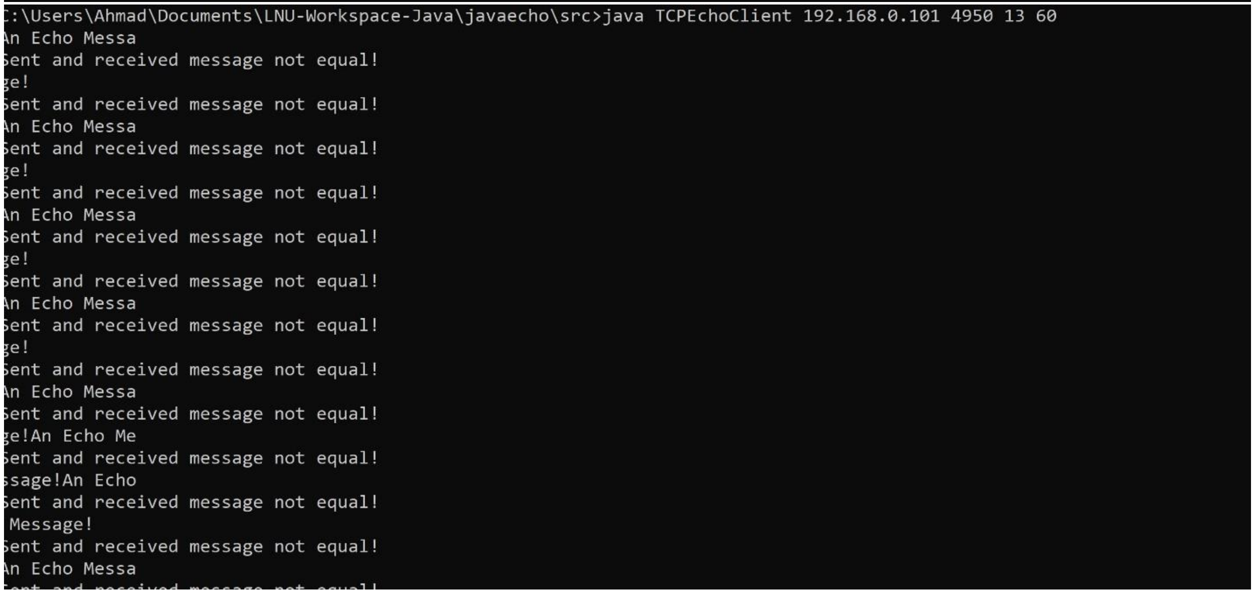
5x Client arguments: IP: 192.168.0.102 port:4950 buffer:2000 massage rate: 500

In the TCP server I implemented a runnable class called RequestHandler which has a main thread that accepts new connections and the background threads handles each client separately.

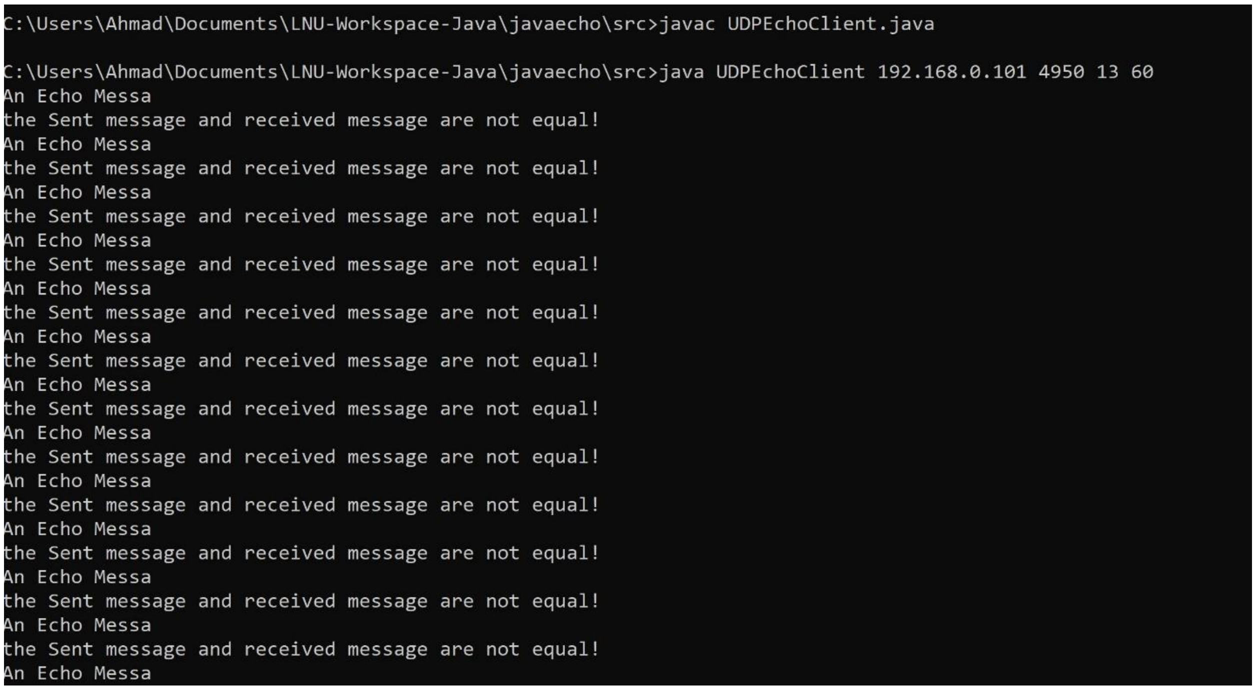
Buffer lower than message size:



```
C:\Users\Ahmad\Documents\LNU-Workspace-Java\javaecho\src>java TCPEchoClient 192.168.0.101 4950 13 60
```



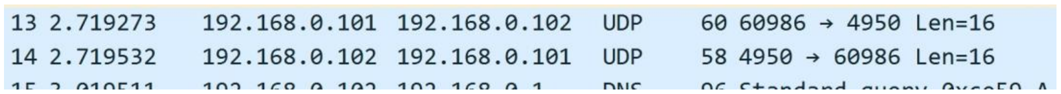
```
C:\Users\Ahmad\Documents\LNU-Workspace-Java\javaecho\src>java TCPEchoClient 192.168.0.101 4950 13 60
An Echo Messa
Sent and received message not equal!
ge!
Sent and received message not equal!
An Echo Messa
Sent and received message not equal!
ge!
Sent and received message not equal!
An Echo Messa
Sent and received message not equal!
ge!
Sent and received message not equal!
An Echo Messa
Sent and received message not equal!
ge!
Sent and received message not equal!
An Echo Messa
Sent and received message not equal!
ge!An Echo Me
Sent and received message not equal!
ssage!An Echo
Sent and received message not equal!
 Message!
Sent and received message not equal!
An Echo Messa
Sent and received message not equal!
```



```
C:\Users\Ahmad\Documents\LNU-Workspace-Java\javaecho\src>javac UDPEchoClient.java

C:\Users\Ahmad\Documents\LNU-Workspace-Java\javaecho\src>java UDPEchoClient 192.168.0.101 4950 13 60
An Echo Messa
the Sent message and received message are not equal!
An Echo Messa
the Sent message and received message are not equal!
An Echo Messa
the Sent message and received message are not equal!
An Echo Messa
the Sent message and received message are not equal!
An Echo Messa
the Sent message and received message are not equal!
An Echo Messa
the Sent message and received message are not equal!
An Echo Messa
the Sent message and received message are not equal!
An Echo Messa
the Sent message and received message are not equal!
An Echo Messa
the Sent message and received message are not equal!
An Echo Messa
the Sent message and received message are not equal!
An Echo Messa
the Sent message and received message are not equal!
An Echo Messa
the Sent message and received message are not equal!
An Echo Messa
```

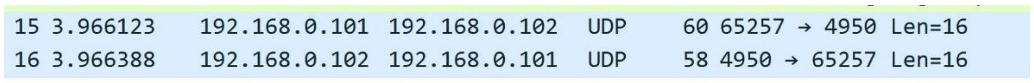
The UDP only receives the amount that fits the buffer then it drops the rest. On the other hand, the TCP receives the amount that fits the buffer, but the rest is left in the socket for the next read, this resulting in erratic messages. First message is incomplete, second message is part of the first and part of itself, etc.

**Problem 4:** only one message was sent.



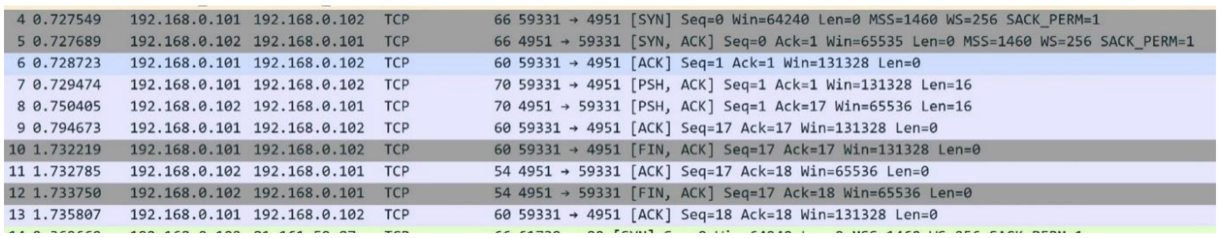
```
13 2.719273    192.168.0.101 192.168.0.102  UDP     60 60986 → 4950 Len=16
14 2.719532    192.168.0.102 192.168.0.101  UDP     58 4950 → 60986 Len=16
```

*Figure 2 UDP normal buffer size*



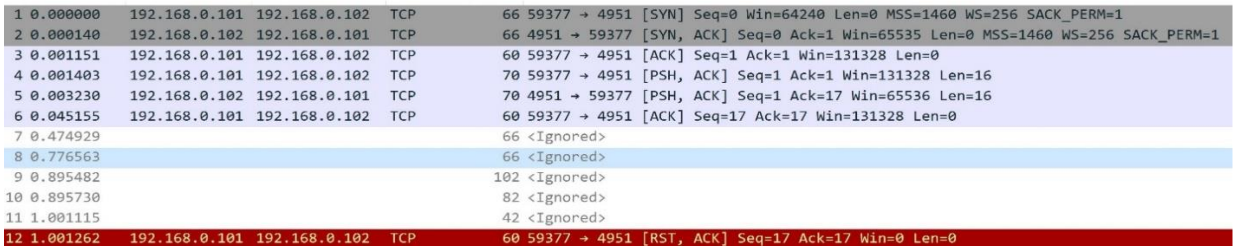
```
15 3.966123    192.168.0.101 192.168.0.102  UDP     60 65257 → 4950 Len=16
16 3.966388    192.168.0.102 192.168.0.101  UDP     58 4950 → 65257 Len=16
```

*Figure 3 UDP small buffer size*



```
 4 0.727549   192.168.0.101 192.168.0.102  TCP        66 59331 → 4951 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 5 0.727689   192.168.0.102 192.168.0.101  TCP        66 4951 → 59331 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
 6 0.728723   192.168.0.101 192.168.0.102  TCP        60 59331 → 4951 [ACK] Seq=1 Ack=1 Win=131328 Len=0
 7 0.729474   192.168.0.101 192.168.0.102  TCP        70 59331 → 4951 [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=16
 8 0.750405   192.168.0.102 192.168.0.101  TCP        70 4951 → 59331 [PSH, ACK] Seq=1 Ack=17 Win=65536 Len=16
 9 0.794673   192.168.0.101 192.168.0.102  TCP        60 59331 → 4951 [ACK] Seq=17 Ack=17 Win=131328 Len=0
10 1.732219   192.168.0.101 192.168.0.102  TCP        60 59331 → 4951 [FIN, ACK] Seq=17 Ack=17 Win=131328 Len=0
11 1.732785   192.168.0.102 192.168.0.101  TCP        54 4951 → 59331 [ACK] Seq=17 Ack=18 Win=65536 Len=0
12 1.733750   192.168.0.102 192.168.0.101  TCP        54 4951 → 59331 [FIN, ACK] Seq=17 Ack=18 Win=65536 Len=0
13 1.735807   192.168.0.101 192.168.0.102  TCP        60 59331 → 4951 [ACK] Seq=18 Ack=18 Win=131328 Len=0
```

*Figure 4 TCP normal buffer size*



```
 1 0.000000   192.168.0.101 192.168.0.102  TCP        66 59377 → 4951 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 2 0.000140   192.168.0.102 192.168.0.101  TCP        66 4951 → 59377 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
 3 0.001151   192.168.0.101 192.168.0.102  TCP        60 59377 → 4951 [ACK] Seq=1 Ack=1 Win=131328 Len=0
 4 0.001403   192.168.0.101 192.168.0.102  TCP        70 59377 → 4951 [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=16
 5 0.003230   192.168.0.102 192.168.0.101  TCP        70 4951 → 59377 [PSH, ACK] Seq=1 Ack=17 Win=65536 Len=16
 6 0.045155   192.168.0.101 192.168.0.102  TCP        60 59377 → 4951 [ACK] Seq=17 Ack=17 Win=131328 Len=0
 7 0.474929                                           66 <Ignored>
 8 0.776563                                           66 <Ignored>
 9 0.895482                                          102 <Ignored>
10 0.895730                                           82 <Ignored>
11 1.001115                                           42 <Ignored>
12 1.001262   192.168.0.101 192.168.0.102  TCP        60 59377 → 4951 [RST, ACK] Seq=17 Ack=17 Win=0 Len=0
```

*Figure 5 TCP small buffer size*

As it can be seen from figure 1 and 2 UDP acts the same in both scenarios.

(For TCP the small buffer is explained)

TCP on the other hand is more complicated from the start. In the first 3 messages we can see the three-way handshake. The client sends a SYN (synchronization) message to the server, the server acknowledges (ACK) and send its own SYN, to which the client ACK. Then the client pushes (PSH) the data to the server, which is acknowledged and pushed back by the server. The client acknowledges the receival. Then the client resets the connection to the server, I believe because there is still other unread data on the channel.

The Differences between TCP and UDP:

TCP is connection oriented where UDP is connectionless. TCP is reliable where UDP is unreliable. TCP date arrive in order where UDP data that arrives are unordered. Since TCP uses Stream it can send and receive several packets. UDP on the other hand can only send and receive on packet in one go. TCP is used by most application such as emails and UDP is used for multimedia applications.

## Change Log:

Changed the checkIp method from regex to StringTokenizer.

Created two mains one for the TCP client and the other for UDP client.

Fixed the buffer management in both TCP client and server.

Added more error handling and management.

Fixed the transfer rate method to sleep for the rest of the one second if transfer rate reached max.