Ruth Dirnfeld (rd222dv)

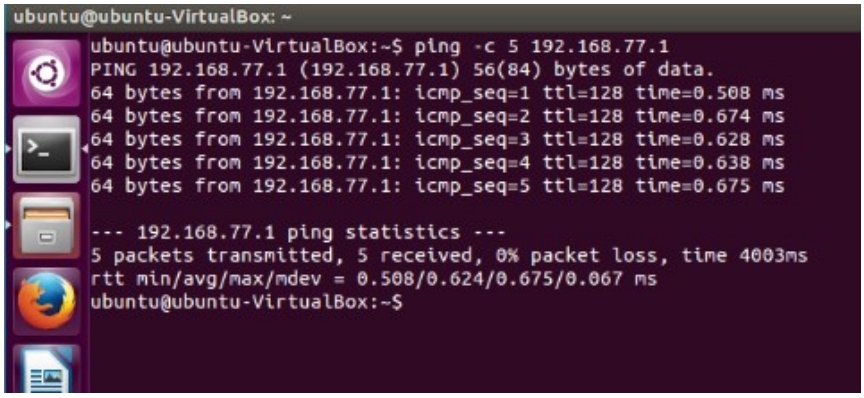
Problem 1 – Setup works

```
C:\WINDOWS\system32>ping 192.168.77.3

Pinging 192.168.77.3 with 32 bytes of data:
Reply from 192.168.77.3: bytes=32 time<1ms TTL=64
Reply from 192.168.77.3: bytes=32 time<1ms TTL=64
Reply from 192.168.77.3: bytes=32 time<1ms TTL=64
Reply from 192.168.77.3: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.77.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\WINDOWS\system32>
```



Ruth Dirnfeld (rd222dv)

Problem 2

The next feature to implement is configuration of client buffer size (in bytes) and messages transfer rate (in messages per second). On the included screenshot we can see the 5 messages / second.

```
C:\Users\Ruthi\Desktop>java -cp . lab1/UDPEchoClient 192.168.77.3 4950 1024 5
16 bytes sent and received || Buffer Size: 1024 ||
16 bytes sent and received || Buffer Size: 1024 ||
16 bytes sent and received || Buffer Size: 1024 ||
16 bytes sent and received || Buffer Size: 1024 ||
16 bytes sent and received || Buffer Size: 1024 ||
Runtime: 1000 ms
```

Added handled **exceptions/errors**:

1. If number of arguments is not 4.

2. If IP address that doesn't match the created pattern:

    - IP pattern that is allowed: 192.168.77.3

    - "10.10.10" or "10.10" or "10" – must have 4 "."

    - "a.a.a.a" or "10.0.0.a" – only digits are allowed

    - "10.10.10.256" or "222.222.2.999" or "2222.22.22.22" – digit must between [0-255]

3. If Port is smaller or equal to 0.

4. If Port is bigger than 65535.

5. If Buffer size is smaller than 1 (too small).

6. If Transfer rate is smaller than 0 (too small).

7. If Transfer rate is equal to 0, then send at least one message.

Ruth Dirnfeld (rd222dv)

VG-task 1

Let's assume messages transfer rate >= 100/sec. It is necessary to send as many messages as possible during a second and notify user about the amount of remaining messages.

Sent message with small transfer rate vs. with bigger transfer rate:

```
16 bytes sent and received || Buffer Size: 1024 ||
16 bytes sent and received || Buffer Size: 1024 ||
16 bytes sent and received || Buffer Size: 1024 ||
Time: 1000 ms. Not sent: 0
```

```
16 bytes sent and received || Buffer Size: 1024 ||
16 bytes sent and received || Buffer Size: 1024 ||
16 bytes sent and received || Buffer Size: 1024 ||
Time: 1002 ms. Not sent: 12
```

Problem 3

3.1.1 Multiple Clients:

The server must support multiple client connections. After sending response (echo), the thread execution stops. The main server thread runs in a loop until manual termination. Included resulting screenshot with multiple clients:

```
root@ubuntu-VirtualBox:/media/sf_linux/1DV701/src# java -cp . lab1/TCPEchoServer***running***
TCP echo request from Client 0 || IP: /192.168.77.1 || Port: 60635 || Received 16 bytes || Sent 16 bytes || Buffer size = 1024

TCP echo request from Client 0 || IP: /192.168.77.1 || Port: 60635 || Received 16 bytes || Sent 16 bytes || Buffer size = 1024

--Connection for Client 0 was closed--

TCP echo request from Client 1 || IP: /192.168.77.1 || Port: 60636 || Received 16 bytes || Sent 16 bytes || Buffer size = 1024

TCP echo request from Client 1 || IP: /192.168.77.1 || Port: 60636 || Received 16 bytes || Sent 16 bytes || Buffer size = 1024

TCP echo request from Client 2 || IP: /192.168.77.1 || Port: 60637 || Received 16 bytes || Sent 16 bytes || Buffer size = 1024

--Connection for Client 1 was closed--

TCP echo request from Client 2 || IP: /192.168.77.1 || Port: 60637 || Received 16 bytes || Sent 16 bytes || Buffer size = 1024

TCP echo request from Client 3 || IP: /192.168.77.1 || Port: 60638 || Received 16 bytes || Sent 16 bytes || Buffer size = 1024

--Connection for Client 2 was closed--

TCP echo request from Client 3 || IP: /192.168.77.1 || Port: 60638 || Received 16 bytes || Sent 16 bytes || Buffer size = 1024

--Connection for Client 3 was closed--

TCP echo request from Client 4 || IP: /192.168.77.1 || Port: 60639 || Received 16 bytes || Sent 16 bytes || Buffer size = 1024

TCP echo request from Client 4 || IP: /192.168.77.1 || Port: 60639 || Received 16 bytes || Sent 16 bytes || Buffer size = 1024
```

3.1.2 Multiple Clients from Client side:

The main server thread runs in a loop until manual termination

```
16 bytes sent and received || Buffer Size: 1024 ||
16 bytes sent and received || Buffer Size: 1024 ||
16 bytes sent and received || Buffer Size: 1024 ||
16 bytes sent and received || Buffer Size: 1024 ||
Terminate batch job (Y/N)? y
```

Ruth Dirnfeld (rd222dv)

3.2.1 UDP with small buffer size:

Setting client buffer size to the small value: 64 and running programs with message of bigger size: 151 bytes. Including the resulting UDP screenshots:

```
root@ubuntu-VirtualBox: /media/sf_linux/1DV701/src
UDP echo request from IP: 192.168.77.1 Port: 62159 || Received: 151 bytes || Sent: 151 bytes || Buffer size: 1024
UDP echo request from IP: 192.168.77.1 Port: 62159 || Received: 151 bytes || Sent: 151 bytes || Buffer size: 1024
UDP echo request from IP: 192.168.77.1 Port: 62159 || Received: 151 bytes || Sent: 151 bytes || Buffer size: 1024
UDP echo request from IP: 192.168.77.1 Port: 62159 || Received: 151 bytes || Sent: 151 bytes || Buffer size: 1024
UDP echo request from IP: 192.168.77.1 Port: 62159 || Received: 151 bytes || Sent: 151 bytes || Buffer size: 1024
UDP echo request from IP: 192.168.77.1 Port: 62159 || Received: 151 bytes || Sent: 151 bytes || Buffer size: 1024
UDP echo request from IP: 192.168.77.1 Port: 62159 || Received: 151 bytes || Sent: 151 bytes || Buffer size: 1024
UDP echo request from IP: 192.168.77.1 Port: 62159 || Received: 151 bytes || Sent: 151 bytes || Buffer size: 1024
```

```
C:\Users\Ruthi\Desktop>java -cp . lab1/UDPEchoClient 192.168.77.3 4950 64 50
Sent bytes: 151 Received bytes: 64 msg not equal! || Buffer Size: 64 ||
Sent bytes: 151 Received bytes: 64 msg not equal! || Buffer Size: 64 ||
Sent bytes: 151 Received bytes: 64 msg not equal! || Buffer Size: 64 ||
Sent bytes: 151 Received bytes: 64 msg not equal! || Buffer Size: 64 ||
Sent bytes: 151 Received bytes: 64 msg not equal! || Buffer Size: 64 ||
Sent bytes: 151 Received bytes: 64 msg not equal! || Buffer Size: 64 ||
Sent bytes: 151 Received bytes: 64 msg not equal! || Buffer Size: 64 ||
Sent bytes: 151 Received bytes: 64 msg not equal! || Buffer Size: 64 ||
Sent bytes: 151 Received bytes: 64 msg not equal! || Buffer Size: 64 ||
Sent bytes: 151 Received bytes: 64 msg not equal! || Buffer Size: 64 ||
```

3.2.2 TCP with small buffer size:

Setting client buffer size to the small value: 64 and running programs with message of bigger size: 151 bytes. Including the resulting TCP screenshots:

```
root@ubuntu-VirtualBox:/media/sf_linux/1DV701/src# java -cp . lab1/TCPEchoServer***running***
TCP echo request from Client 0 || IP: /192.168.77.1 || Port: 61114 || Received 151 bytes || Sent 151 bytes || Buffer size = 1024

TCP echo request from Client 0 || IP: /192.168.77.1 || Port: 61114 || Received 151 bytes || Sent 151 bytes || Buffer size = 1024

TCP echo request from Client 0 || IP: /192.168.77.1 || Port: 61114 || Received 151 bytes || Sent 151 bytes || Buffer size = 1024

TCP echo request from Client 0 || IP: /192.168.77.1 || Port: 61114 || Received 151 bytes || Sent 151 bytes || Buffer size = 1024

TCP echo request from Client 0 || IP: /192.168.77.1 || Port: 61114 || Received 151 bytes || Sent 151 bytes || Buffer size = 1024

TCP echo request from Client 0 || IP: /192.168.77.1 || Port: 61114 || Received 151 bytes || Sent 151 bytes || Buffer size = 1024
```

```
C:\Users\Ruthi\Desktop>java -cp . lab1/TCPEchoClient2 192.168.77.3 4950 64 50
151 bytes sent and received || Buffer Size: 64 ||
151 bytes sent and received || Buffer Size: 64 ||
151 bytes sent and received || Buffer Size: 64 ||
151 bytes sent and received || Buffer Size: 64 ||
151 bytes sent and received || Buffer Size: 64 ||
151 bytes sent and received || Buffer Size: 64 ||
151 bytes sent and received || Buffer Size: 64 ||
```

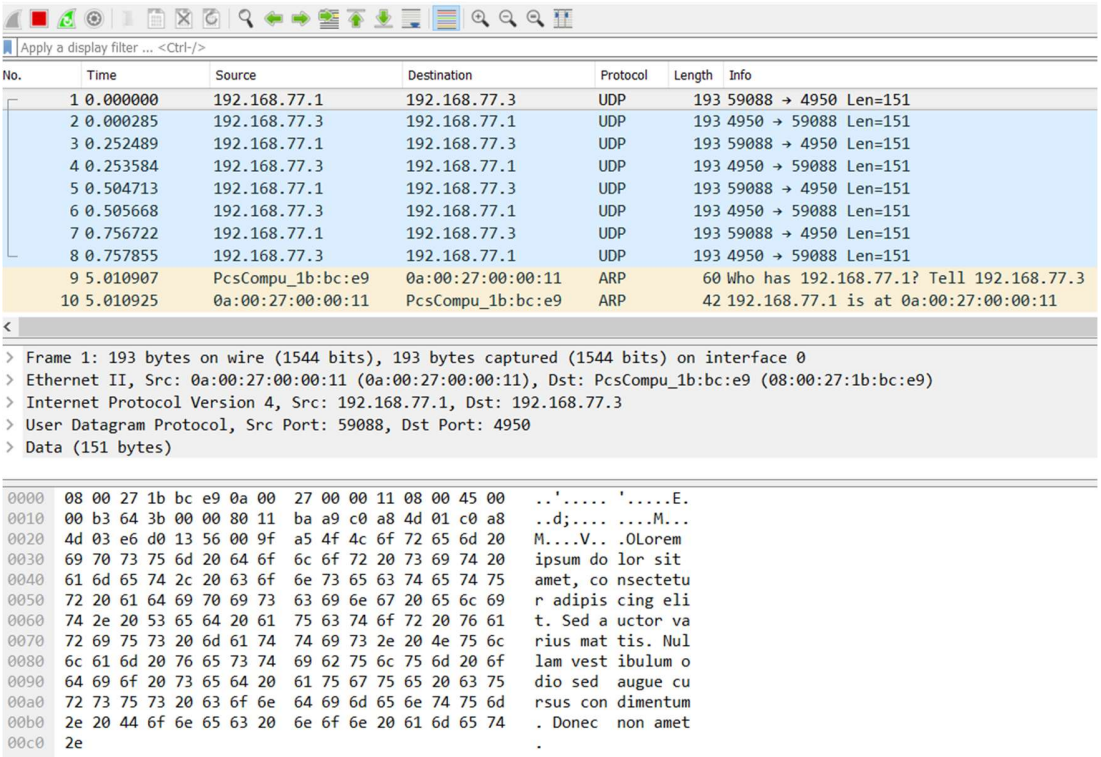
3.3 What is the difference and why?

There is a major difference between UDP and TCP. In the included screenshots we can see that UDP sends 151 bytes, but receives only 64 bytes, so the message is not equal. That is, because UDP uses Datagram packets, which use fixed specified sizes and all overdue bytes will be lost. Where, on the other hand we can see that TCP sends and receives the same message of 151 bytes. That is, because TCP gives us the ability to send or receive a byte stream. Furthermore, TCP manages message acknowledgment and retransmissions in case of lost parts. That means that there is no missing data. UDP is not reliable, since it does not use any concepts of acknowledgment, time out and retransmission. UDP does not ensure that communication has reached the receiver.

Problem 4

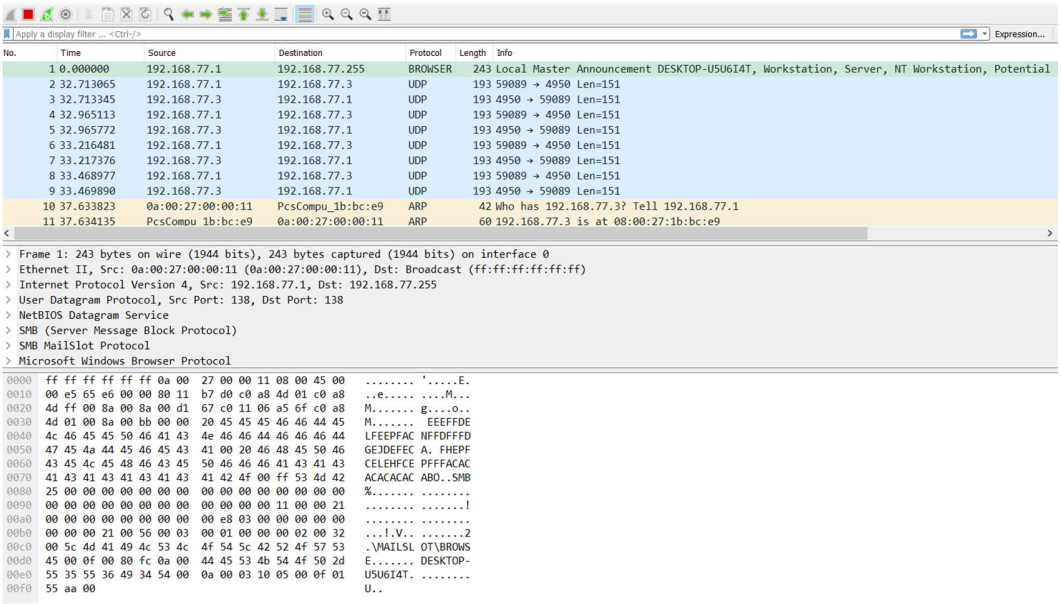
The Wireshark experiment has been performed with a transfer rate of 4 and once a big buffer size and once with a small buffer size for both, UDP and TCP.

UDP big buffer size



Ruth Dirnfeld (rd222dv)

UDP small buffer size



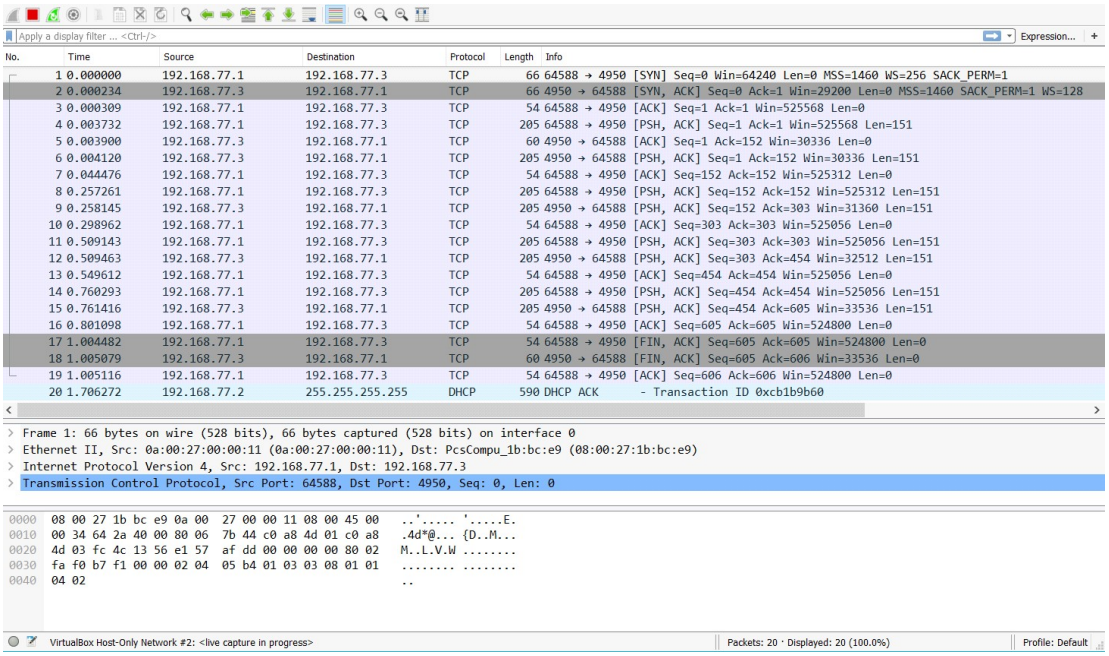
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.77.1	192.168.77.255	BROWSER	243	Local Master Announcement DESKTOP-U5U6I4T, Workstation, Server, NT Workstation, Potential
2	32.713065	192.168.77.1	192.168.77.3	UDP	193	59089 → 4950 Len=151
3	32.713345	192.168.77.3	192.168.77.1	UDP	193	4950 → 59089 Len=151
4	32.965113	192.168.77.1	192.168.77.3	UDP	193	59089 → 4950 Len=151
5	32.965772	192.168.77.3	192.168.77.1	UDP	193	4950 → 59089 Len=151
6	33.216481	192.168.77.1	192.168.77.3	UDP	193	59089 → 4950 Len=151
7	33.217376	192.168.77.3	192.168.77.1	UDP	193	4950 → 59089 Len=151
8	33.468977	192.168.77.1	192.168.77.3	UDP	193	59089 → 4950 Len=151
9	33.469890	192.168.77.3	192.168.77.1	UDP	193	4950 → 59089 Len=151
10	37.633823	0a:00:27:00:00:11	PcsCompu_1b:bc:e9	ARP	42	Who has 192.168.77.3? Tell 192.168.77.1
11	37.634135	PcsCompu_1b:bc:e9	0a:00:27:00:00:11	ARP	60	192.168.77.3 is at 08:00:27:1b:bc:e9

```
> Frame 1: 243 bytes on wire (1944 bits), 243 bytes captured (1944 bits) on interface 0
> Ethernet II, Src: 0a:00:27:00:00:11 (0a:00:27:00:00:11), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 192.168.77.1, Dst: 192.168.77.255
> User Datagram Protocol, Src Port: 138, Dst Port: 138
> NetBIOS Datagram Service
> SMB (Server Message Block Protocol)
> SMB MailSlot Protocol
> Microsoft Windows Browser Protocol
```

```
0000  ff ff ff ff ff ff 0a 00  27 00 00 11 08 00 45 00   ........ '.....E.
0010  00 e5 65 e6 00 00 80 11  b7 d0 c0 a8 4d 01 c0 a8   ..e..... ....M...
0020  4d ff 00 8a 00 8a 00 d1  67 c0 11 06 a5 6f c0 a8   M....... g....o..
0030  4d 01 00 8a 00 bb 00 00  20 45 45 45 46 46 44 45   M.......  EEEFFDE
0040  4c 46 45 45 50 46 41 43  4e 46 46 44 46 46 46 44   LFEEPFAC NFFDFFFD
0050  47 45 4a 44 45 46 45 43  41 00 20 46 48 45 50 46   GEJDEFEC A. FHEPF
0060  43 45 4c 45 48 46 43 45  50 46 46 46 41 43 41 43   CELEHFCE PFFFACAC
0070  41 43 41 43 41 43 41 43  41 42 4f 00 ff 53 4d 42   ACACACAC ABO..SMB
0080  25 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   %....... ........
0090  00 00 00 00 00 00 00 00  00 00 00 00 11 00 00 21   ........ .......!
00a0  00 00 00 00 00 00 00 00  00 e8 03 00 00 00 00 00   ........ ........
00b0  00 00 00 21 00 56 00 03  00 01 00 00 00 02 00 32   ...!.V.. .......2
00c0  00 5c 4d 41 49 4c 53 4c  4f 54 5c 42 52 4f 57 53   .\MAILSL OT\BROWS
00d0  45 00 0f 00 80 fc 0a 00  44 45 53 4b 54 4f 50 2d   E....... DESKTOP-
00e0  55 35 55 36 49 34 54 00  0a 00 03 10 05 00 0f 01   U5U6I4T. ........
00f0  55 aa 00                                          U..
```

In the two screenshots we can see that UDP connection follows request and response query from sender and receiver respectively. At the beginning the Client sent a message of size 151 bytes and the server received 193 bytes. (message: 151 bytes; header: 42 bytes). The difference is visible in the lower part of the screenshots where we can see the message because of the packet size which is dependent on the buffer size, which only in the first screenshot is bigger than the message size.

TCP big buffer size



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.77.1	192.168.77.3	TCP	66	64588 → 4950 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.000234	192.168.77.3	192.168.77.1	TCP	66	4950 → 64588 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
3	0.000309	192.168.77.1	192.168.77.3	TCP	54	64588 → 4950 [ACK] Seq=1 Ack=1 Win=525568 Len=0
4	0.003732	192.168.77.1	192.168.77.3	TCP	205	64588 → 4950 [PSH, ACK] Seq=1 Ack=1 Win=525568 Len=151
5	0.003900	192.168.77.3	192.168.77.1	TCP	60	4950 → 64588 [ACK] Seq=1 Ack=152 Win=30336 Len=0
6	0.004120	192.168.77.3	192.168.77.1	TCP	205	4950 → 64588 [PSH, ACK] Seq=1 Ack=152 Win=30336 Len=151
7	0.044476	192.168.77.1	192.168.77.3	TCP	54	64588 → 4950 [ACK] Seq=152 Ack=152 Win=525312 Len=0
8	0.257261	192.168.77.1	192.168.77.3	TCP	205	64588 → 4950 [PSH, ACK] Seq=152 Ack=152 Win=525312 Len=151
9	0.258145	192.168.77.3	192.168.77.1	TCP	205	4950 → 64588 [PSH, ACK] Seq=152 Ack=303 Win=31360 Len=151
10	0.298962	192.168.77.1	192.168.77.3	TCP	54	64588 → 4950 [ACK] Seq=303 Ack=303 Win=525056 Len=0
11	0.509143	192.168.77.1	192.168.77.3	TCP	205	64588 → 4950 [PSH, ACK] Seq=303 Ack=303 Win=525056 Len=151
12	0.509463	192.168.77.3	192.168.77.1	TCP	205	4950 → 64588 [PSH, ACK] Seq=303 Ack=454 Win=32512 Len=151
13	0.549612	192.168.77.1	192.168.77.3	TCP	54	64588 → 4950 [ACK] Seq=454 Ack=454 Win=525056 Len=0
14	0.760293	192.168.77.1	192.168.77.3	TCP	205	64588 → 4950 [PSH, ACK] Seq=454 Ack=454 Win=525056 Len=151
15	0.761416	192.168.77.3	192.168.77.1	TCP	205	4950 → 64588 [PSH, ACK] Seq=454 Ack=605 Win=33536 Len=151
16	0.801098	192.168.77.1	192.168.77.3	TCP	54	64588 → 4950 [ACK] Seq=605 Ack=605 Win=524800 Len=0
17	1.004482	192.168.77.1	192.168.77.3	TCP	54	64588 → 4950 [FIN, ACK] Seq=605 Ack=605 Win=524800 Len=0
18	1.005079	192.168.77.3	192.168.77.1	TCP	60	4950 → 64588 [FIN, ACK] Seq=605 Ack=606 Win=33536 Len=0
19	1.005116	192.168.77.1	192.168.77.3	TCP	54	64588 → 4950 [ACK] Seq=606 Ack=606 Win=524800 Len=0
20	1.706272	192.168.77.2	255.255.255.255	DHCP	590	DHCP ACK - Transaction ID 0xcb1b9b60

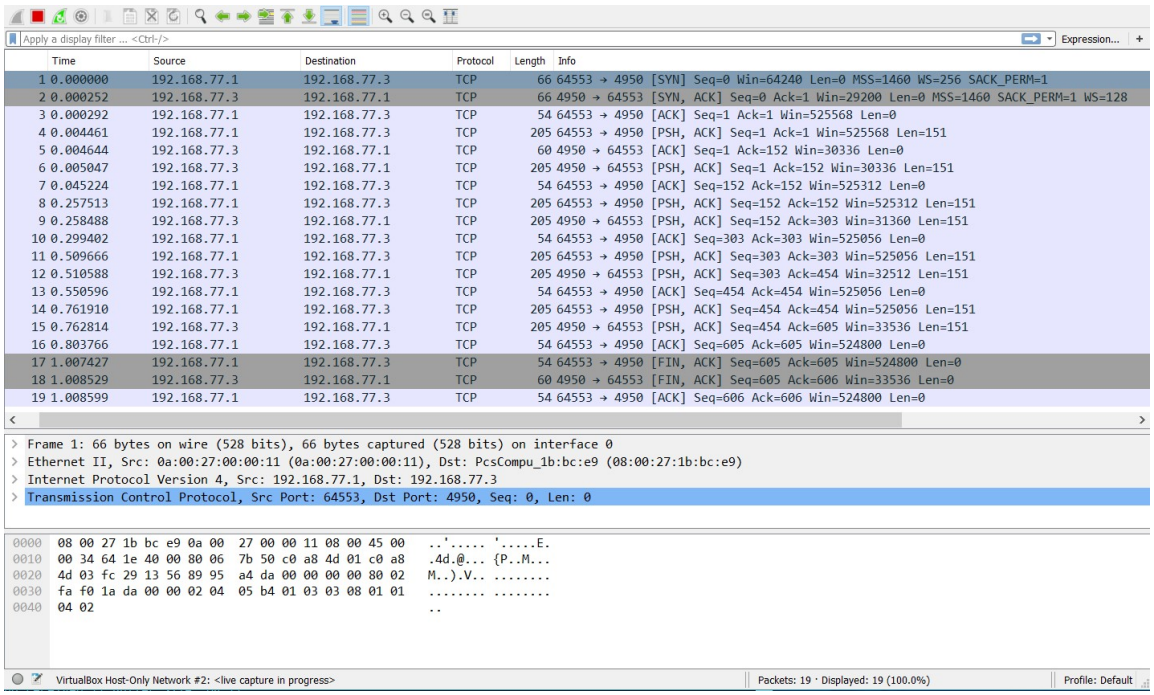
```
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: 0a:00:27:00:00:11 (0a:00:27:00:00:11), Dst: PcsCompu_1b:bc:e9 (08:00:27:1b:bc:e9)
> Internet Protocol Version 4, Src: 192.168.77.1, Dst: 192.168.77.3
> Transmission Control Protocol, Src Port: 64588, Dst Port: 4950, Seq: 0, Len: 0
```

```
0000  08 00 27 1b bc e9 0a 00  27 00 00 11 08 00 45 00   ..'..... '.....E.
0010  00 34 64 2a 40 00 80 06  7b 44 c0 a8 4d 01 c0 a8   .4d*@... {D..M...
0020  4d 03 fc 4c 13 56 e1 57  af dd 00 00 00 00 80 02   M..L.V.W ........
0030  fa f0 b7 f1 00 00 02 04  05 b4 01 03 03 08 01 01   ........ ........
0040  04 02                                             ..
```

VirtualBox Host-Only Network #2: <live capture in progress> | Packets: 20 · Displayed: 20 (100.0%) | Profile: Default

Ruth Dirnfeld (rd222dv)

TCP small buffer size



TCP connection is established using three steps:

1) [SYN] bit from Client to Server (shows the start of a TCP session).

2) [SYN, ACK] bit from Server to Client.

3) [ACK] bit from Client to Server (shows that the ACK number in the TCP header acknowledges data).

If any of this steps don't occur, it means that connection is not established between Client and Server.

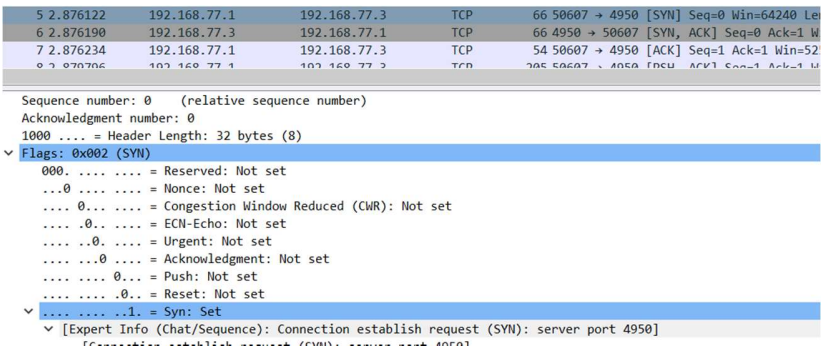
In my screenshots this connection is established and the Client replied with an ACK = 1. We can also see that the Client is sending a message of length = 151. The purpose of the PUSH flag here is to push data from the sending user to the receiving user and not to wait until the Server buffer size is full, but send immediately. The Server receives the message and sends ACK 152. The Client receives that message and sends a message to the Server again. The Server sends back a message with PSH and ACK = 303 (152 + 151). Sending and receiving continues until the Client answers with a [FIN, ACK]. A FIN indicates the termination of a TCP session. The Server answered with the same but increased the value of ACK by 1.

Ruth Dirnfeld (rd222dv)

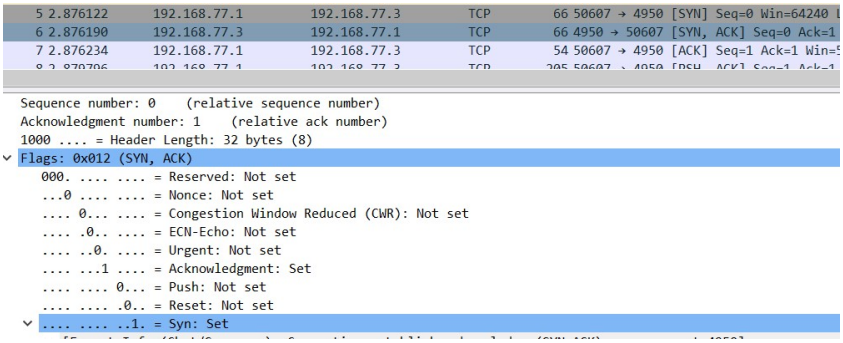
Three Way Handshake detailed:

By expanding the first three lines [SYN] then [SYN, ACK] and [ACK] we get the following:

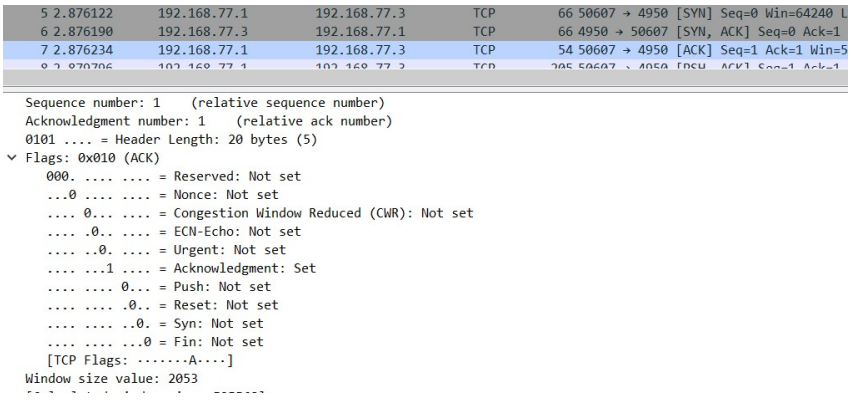
Screenshot #1: [SYN] is basically saying – Can I talk to you?



Screenshot #2: [SYN, ACK] is basically saying – [ACK] Yes, you can talk to me, and again [SYN] Can I talk to you?



Screenshot #3: Then the original machine sends back acknowledgment [ACK] saying – Yes, you can talk to me.



From all the resulting Screenshots we can see that the main differences between UDP and TCP are that TCP uses Segment sequencing and acknowledgments and is connection oriented, where UDP doesn't use any of those. All this makes TCP reliable and UDP unreliable.