

Part 2:

In this part we decided to run our server.py code twice on two separate terminals in Server's VM and client.py on terminal in Client's VM just like you guided us in the exercise.

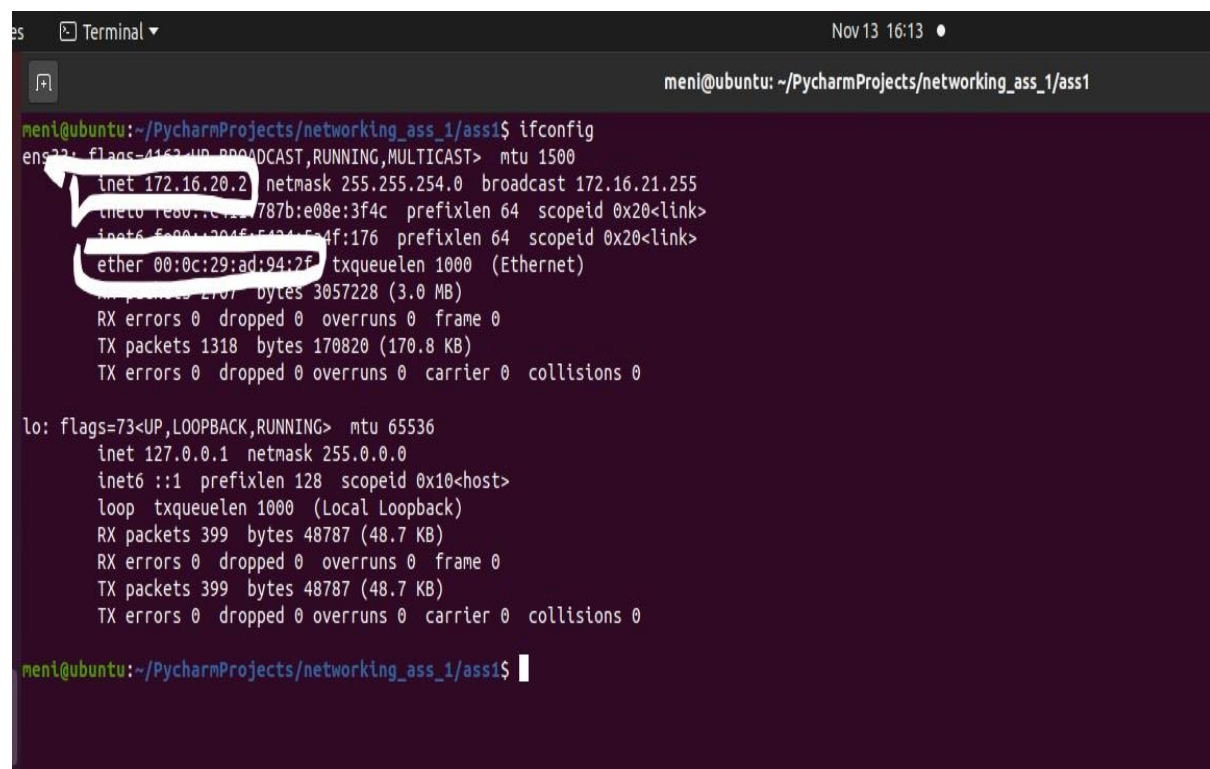
Now first let's determine the IP's and MAC's of the VM's:

Server VM:

Ip: 172.16.20.2

MAC : 00:0c:29:ad:94:2f

Print screen to show how I found these details:



```
meni@ubuntu: ~/PycharmProjects/networking_ass_1/ass1$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.16.20.2 netmask 255.255.254.0 broadcast 172.16.21.255
    inet6 fe80::c11:787b:e08e:3f4c prefixlen 64 scopeid 0x20<link>
    inet6 fe80::2015:5434:54f:176 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:ad:94:2f txqueuelen 1000 (Ethernet)
    RX packets 1707 bytes 3057228 (3.0 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1318 bytes 170820 (170.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 399 bytes 48787 (48.7 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 399 bytes 48787 (48.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

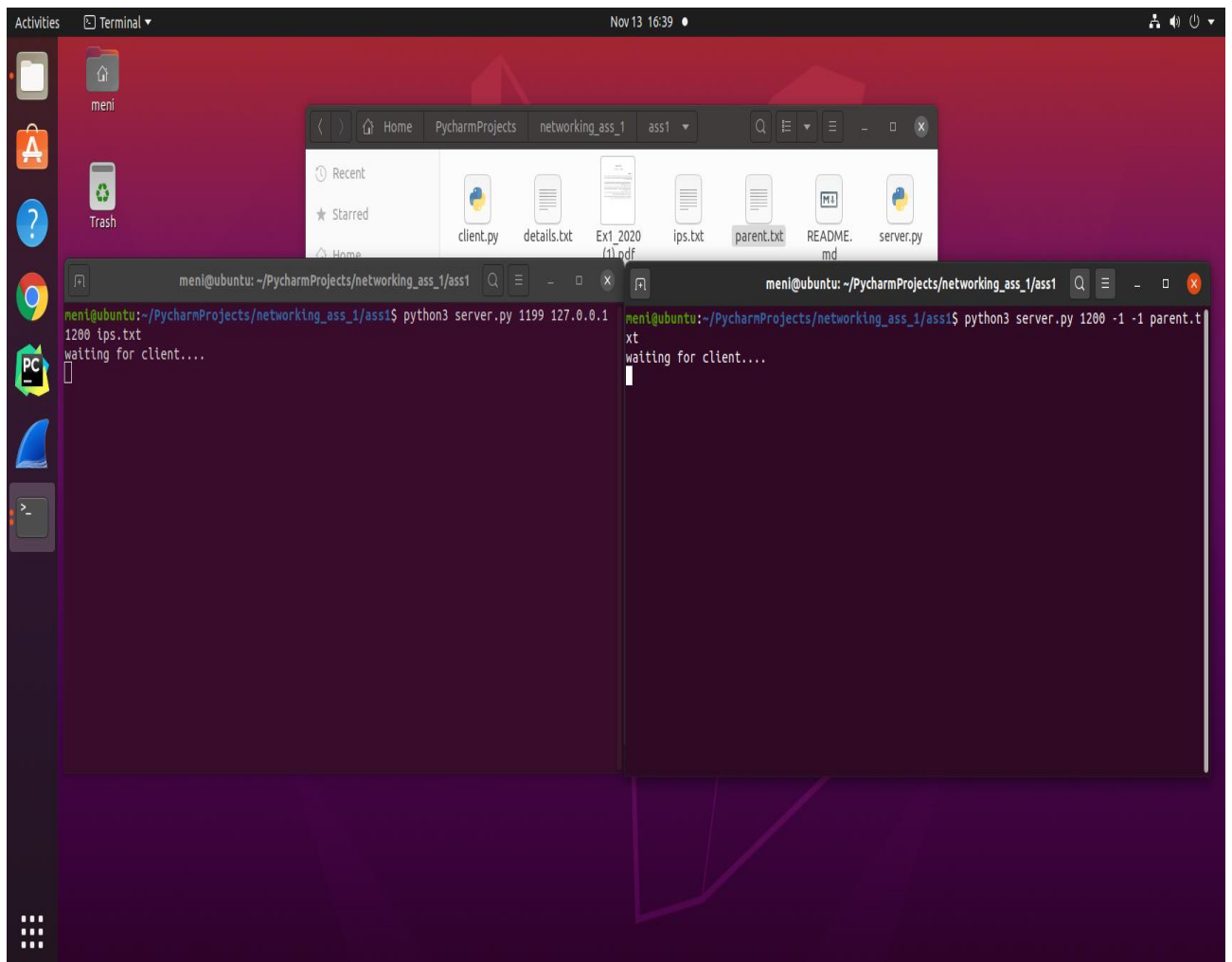
meni@ubuntu:~/PycharmProjects/networking_ass_1/ass1$
```

Let's run the server and its father on Server VM:

Command to run father server: `python3 server.py 1200 -1 -1 parent.txt`

Command to run original server: `python3 server.py 1199 127.0.0.1 1200 ips.txt`

Now they are working, screenshot:

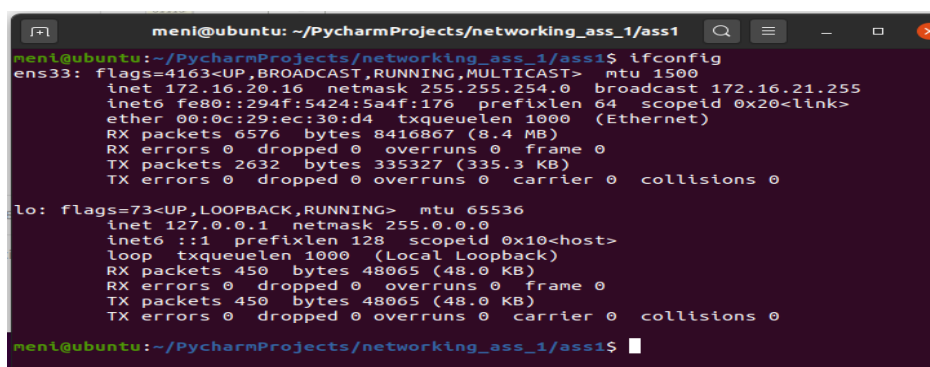


Client VM:

Ip: 172.16.20.16

MAC: 00:0c:29:ec:30:d4

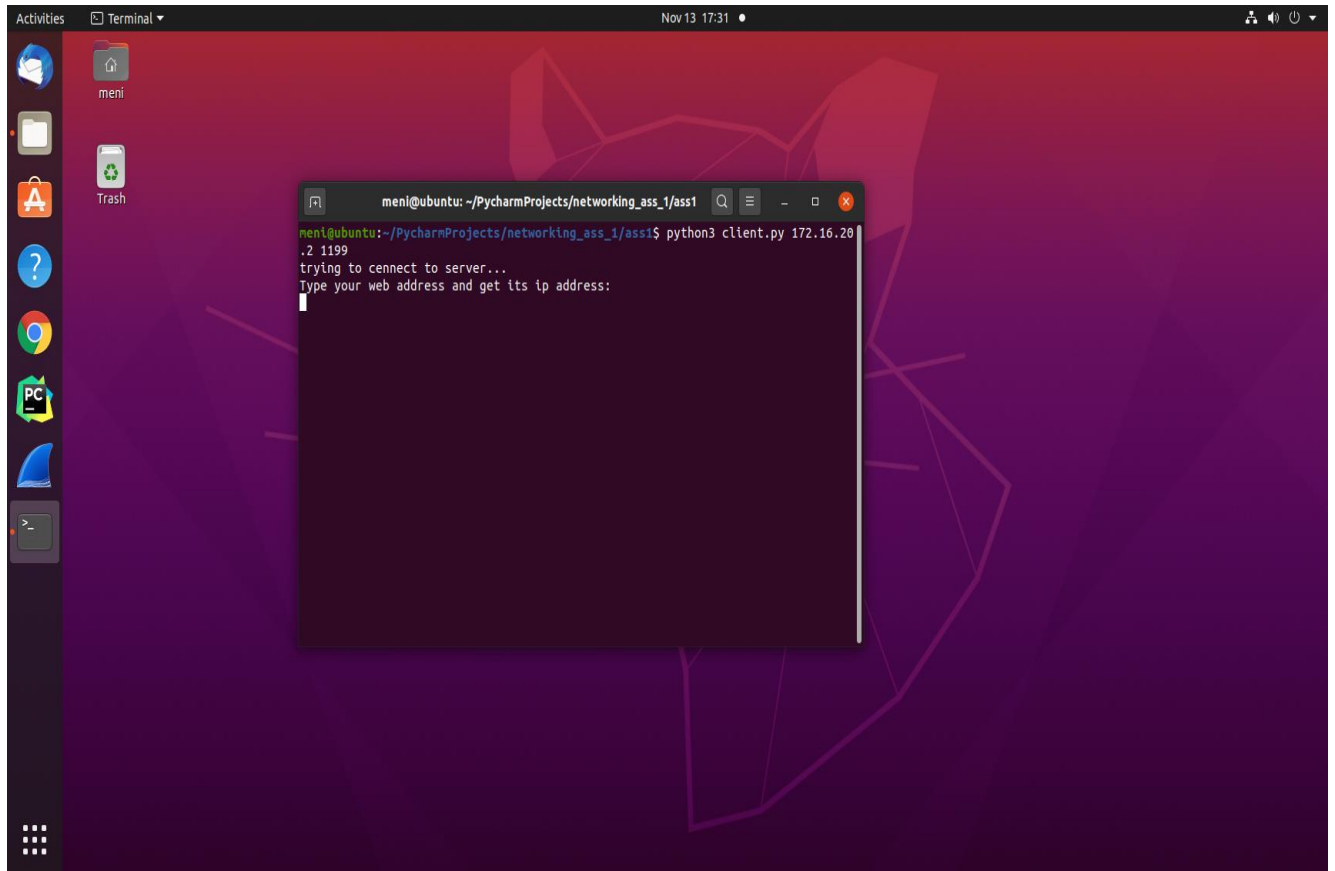
Print screen to show how I found these details:



Let's run the client on Client VM:

Command to run client: `python3 client.py 172.16.20.2 1199`

Screenshot:

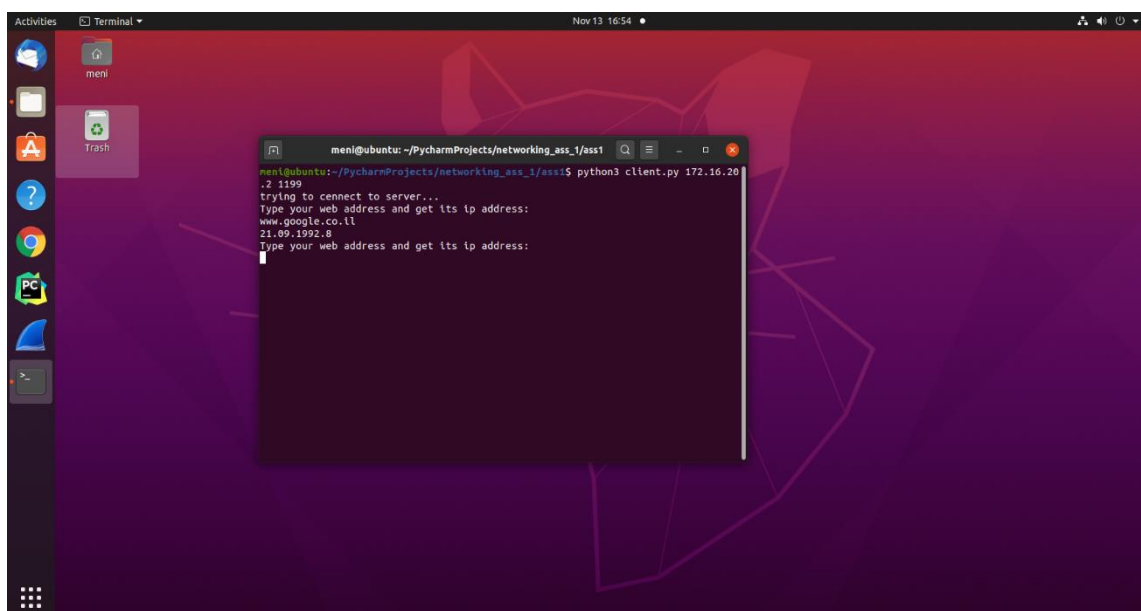


```
menil@ubuntu: ~/PycharmProjects/networking_ass_1/ass1
menil@ubuntu:~/PycharmProjects/networking_ass_1/ass1$ python3 client.py 172.16.20.2 1199
trying to connect to server...
Type your web address and get its ip address:

```

Now let's cut to the chase: try to send query to the server with an address that exists only in father server's file so the original server will have to ask for that info from the father server:

View from client:



```
menil@ubuntu: ~/PycharmProjects/networking_ass_1/ass1
menil@ubuntu:~/PycharmProjects/networking_ass_1/ass1$ python3 client.py 172.16.20.2 1199
trying to connect to server...
Type your web address and get its ip address:
www.google.co.il
21.09.1992.8
Type your web address and get its ip address:

```

View from servers machine :

The image displays a network analysis setup. At the top, a Wireshark packet capture window shows a filter for 'udp.port==1200 || udp.port==1199'. It contains four captured packets, all UDP, showing a sequence of data exchange between 172.16.20.16 and 172.16.20.2.

Below the Wireshark window are two terminal windows. The left terminal shows the output of a server script running on port 1199, which has successfully connected to a client on port 127.0.0.1. The right terminal shows a similar server script running on port 1200, also connected to a client on port 127.0.0.1. Both servers are waiting for a client to search for the IP of www.google.co.il.

Wireshark Filter: `udp.port==1200 || udp.port==1199`

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|--------------|--------------|----------|--------|---------------------|
| 9 | 17.630850987 | 172.16.20.16 | 172.16.20.2 | UDP | 62 | 54699 → 1199 Len=16 |
| 10 | 17.631600904 | 127.0.0.1 | 127.0.0.1 | UDP | 60 | 42460 → 1200 Len=16 |
| 11 | 17.632215529 | 127.0.0.1 | 127.0.0.1 | UDP | 112 | 1200 → 42460 Len=68 |
| 12 | 17.632561763 | 172.16.20.2 | 172.16.20.16 | UDP | 110 | 1199 → 54699 Len=66 |

Terminal 1 (Left):

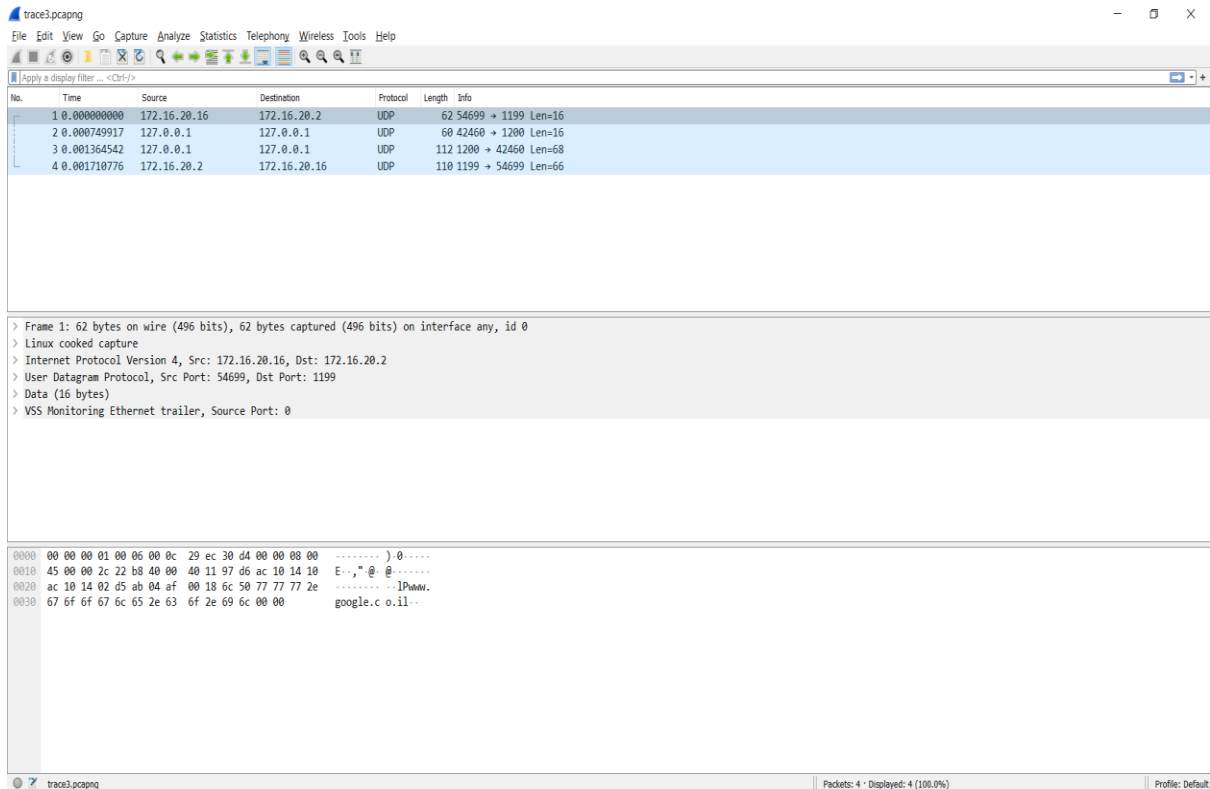
```
meni@ubuntu: ~/PycharmProjects/networking_ass_1/ass1$ python3 server.py 1199 127.0.0.1 1200 ips.txt
waiting for client....
connection established
client is searching for the ip of:www.google.co.il, no problemo, searching...
waiting for client....
```

Terminal 2 (Right):

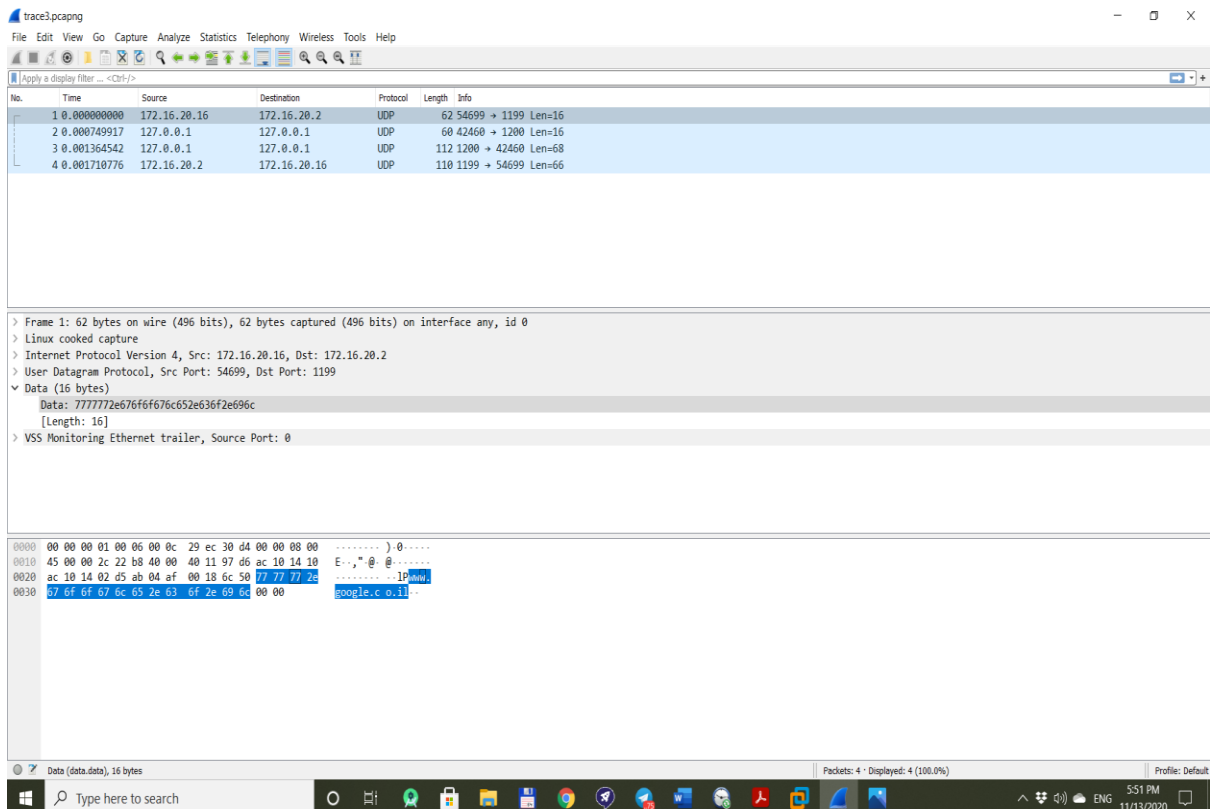
```
meni@ubuntu: ~/PycharmProjects/networking_ass_1/ass1$ python3 server.py 1200 -1 1 parent.txt
waiting for client....
connection established
client is searching for the ip of:www.google.co.il, no problemo, searching...
waiting for client....
```

To conclude: let's see the sniffing from Wireshark and analyze the data from it:

Open the trace2.pcap file:



First line indicates that socket opened between the original server and the client, at the Data tab we can see the string of the query



If you open the Datagram tab which is related to the transport layer you can see that the dest port is 1199 which is the port of our server program

The image shows the Wireshark network protocol analyzer interface. The top pane displays a list of four captured packets. The second packet, at time 0.000749917, is selected. The middle pane shows the details of this packet, expanded to the 'User Datagram Protocol' section. It indicates a source port of 54699 and a destination port of 1199. The data field shows a 16-byte payload with a hex dump and ASCII representation. The bottom status bar indicates the filter is 'Destination Port (udp.dstport), 2 bytes' and 4 packets are displayed.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------|--------------|--------------|----------|--------|---------------------|
| 1 | 0.000000000 | 172.16.20.16 | 172.16.20.2 | UDP | 62 | 54699 → 1199 Len=16 |
| 2 | 0.000749917 | 127.0.0.1 | 127.0.0.1 | UDP | 60 | 42460 → 1200 Len=16 |
| 3 | 0.001364542 | 127.0.0.1 | 127.0.0.1 | UDP | 112 | 1200 → 42460 Len=68 |
| 4 | 0.001710776 | 172.16.20.2 | 172.16.20.16 | UDP | 110 | 1199 → 54699 Len=66 |

Internet Protocol Version 4, Src: 172.16.20.16, Dst: 172.16.20.2

User Datagram Protocol, Src Port: 54699, Dst Port: 1199

Source Port: 54699

Destination Port: 1199

Length: 24

Checksum: 0x6c50 [unverified]

[Checksum Status: Unverified]

[Stream index: 0]

[Timestamps]

Data (16 bytes)

Data: 7777772e676f6676c652e636f2e696c

[Length: 16]

VSS Monitoring Ethernet trailer, Source Port: 0

0000 00 00 00 01 00 06 00 0c 29 ec 30 d4 00 00 08 00 } 0....

0010 45 00 00 2c 22 b8 40 00 40 11 97 d6 ac 10 14 10 E..*..@:.....

0020 ac 10 14 02 d5 ab 04 af 00 18 6c 50 77 77 7e :..IPwww.

0030 67 6f 6f 67 6c 65 2e 63 6f 2e 69 6c 00 00 google.c o.il..

Destination Port (udp.dstport), 2 bytes

Packets: 4 · Displayed: 4 (100.0%)

Profile: Default

If you open the Internet tab which is related to network layer you will see that the src ip is the ip of the client machine and the dest ip is the ip of the server machine

The image shows the Wireshark network protocol analyzer interface with the 'Internet' tab selected for the same packet. The details pane shows the 'Internet Protocol Version 4' section, displaying the source IP as 172.16.20.16 and the destination IP as 172.16.20.2. The status bar at the bottom shows the filter is 'Destination (ip.dst), 4 bytes'.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------|--------------|--------------|----------|--------|---------------------|
| 1 | 0.000000000 | 172.16.20.16 | 172.16.20.2 | UDP | 62 | 54699 → 1199 Len=16 |
| 2 | 0.000749917 | 127.0.0.1 | 127.0.0.1 | UDP | 60 | 42460 → 1200 Len=16 |
| 3 | 0.001364542 | 127.0.0.1 | 127.0.0.1 | UDP | 112 | 1200 → 42460 Len=68 |
| 4 | 0.001710776 | 172.16.20.2 | 172.16.20.16 | UDP | 110 | 1199 → 54699 Len=66 |

Total Length: 44

Identification: 0x22b8 (8888)

Flags: 0x0000, Don't fragment

Fragment offset: 0

Time to live: 64

Protocol: UDP (17)

Header checksum: 0x97d6 [validation disabled]

[Header checksum status: Unverified]

Source: 172.16.20.16

Destination: 172.16.20.2

User Datagram Protocol, Src Port: 54699, Dst Port: 1199

Data (16 bytes)

VSS Monitoring Ethernet trailer, Source Port: 0

0000 00 00 00 01 00 06 00 0c 29 ec 30 d4 00 00 08 00 } 0....

0010 45 00 00 2c 22 b8 40 00 40 11 97 d6 ac 10 14 10 E..*..@:.....

0020 ac 10 14 02 d5 ab 04 af 00 18 6c 50 77 77 7e :..IPwww.

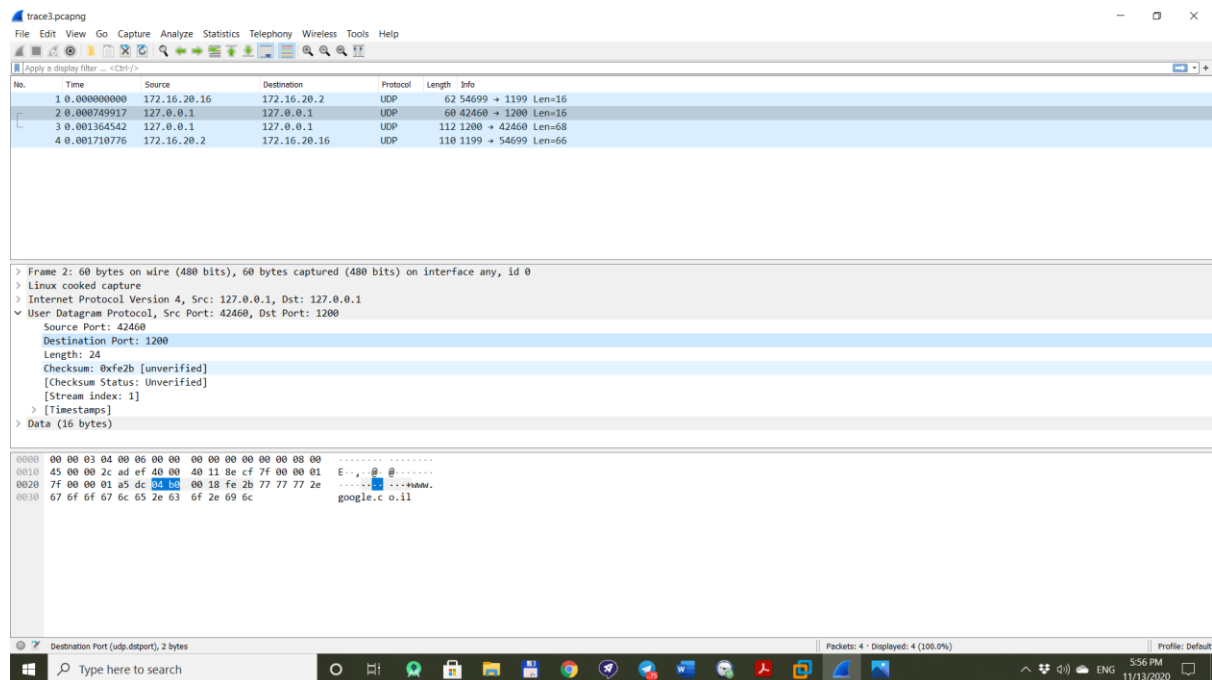
0030 67 6f 6f 67 6c 65 2e 63 6f 2e 69 6c 00 00 google.c o.il..

Destination (ip.dst), 4 bytes

Packets: 4 · Displayed: 4 (100.0%)

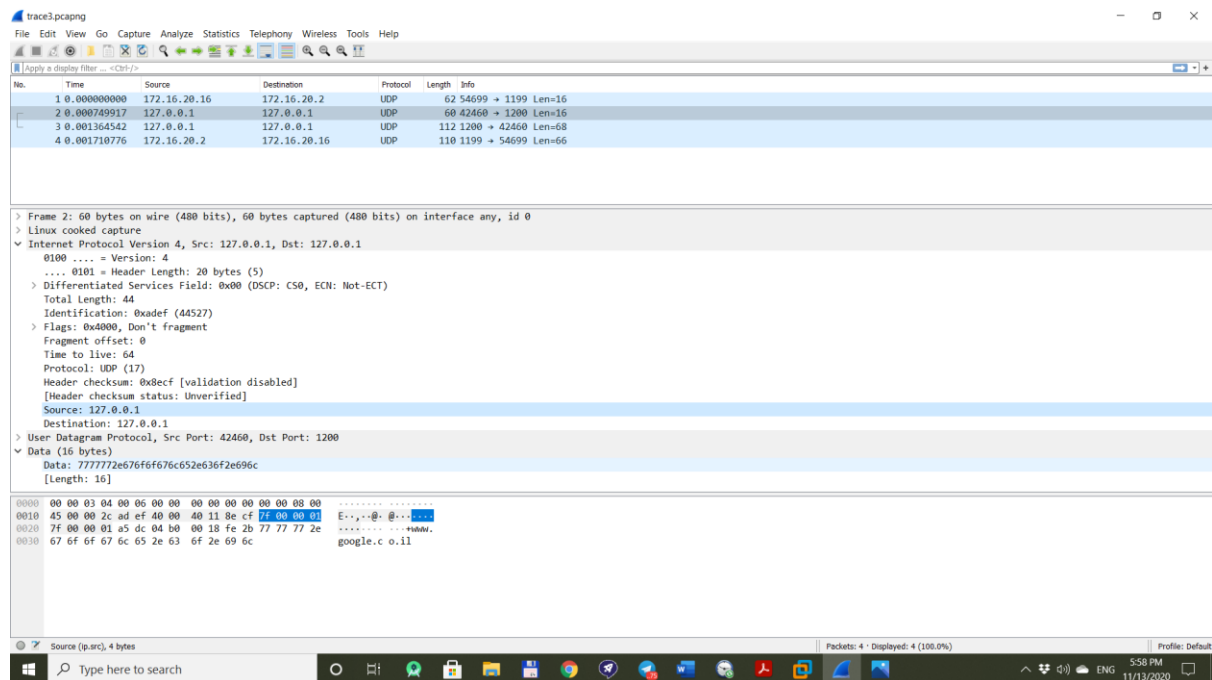
Profile: Default

Move on to the next line in wireshark sniff – we can see that the original server didn't know the ip address of the given query so he goes to his father in order to get more info from it so in the Datagram tab we can see that the dest port is the port of the father server:



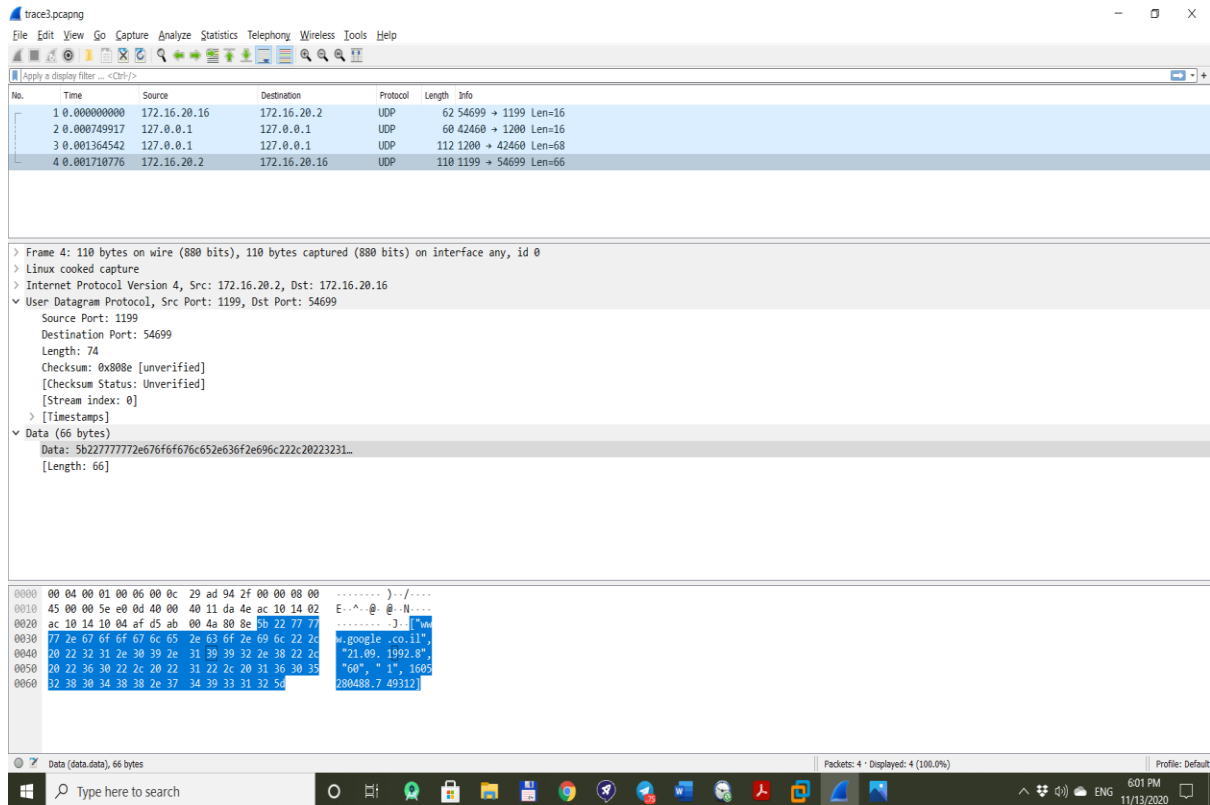
*the source port is some random that the OS on the Server VM gave the original server program after it asked to open new socket to the father server

In the Internet tab we can see that the src ip and des tip is the same: 127.0.0.1 – meaning that we work on the local machine :



Finally, let's take a look at the last line which is the data sent from the server to the client:

Screenshot from the last line in the Data tab:



Above we can see that the data sent back to the client was the full details about the address of www.google.co.il which is its web address, ip address and TTL.

We can conclude that the ip address of www.google.co.il is "21.09.1992.8" , and TTL is 60s.