

Info. To Networking – Ex. 1

Branch A:

Prerequisites : Two Virtual Machines connected to same NAT network just like Hemi showed us at the recitation – first virtual machine is with ip : 172.16.22.42 will serve as the "Client" and the second virtual machine is with ip : 172.16.22.60 will serve as the "Server"

I ran attached "Client.py" script on the "Client" machine and "Server.py" script on the "Server" machine and the results are as follows:

2. I've captured the packets via WireShark, which I ran on the "Server" machine, and filtered them with:

`"(udp.port==12345 && ip.src==172.16.22.42) || ip.dst ==172.16.22.42"`

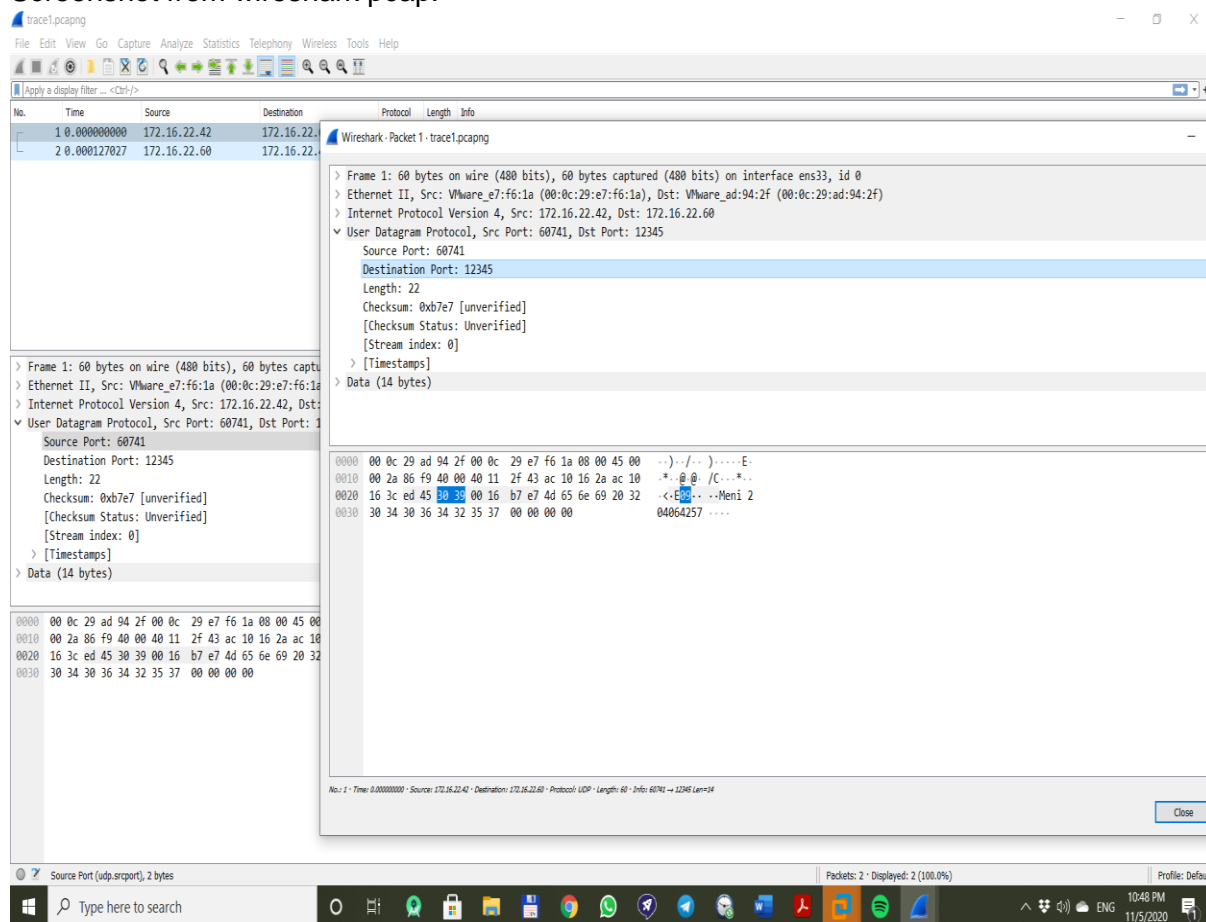
(See Attached **trace1.pcapng**)

3. The Ports are specified in the following:

client.py , Line 4

the client file specifies the destination port (the port to which the server listens), which is bound to 12345 In wireshark, we can notice that the message packet (sent from client) holds a destination field which holds port 12345, as well as the destination ip which is : 172.16.22.60

Screenshot from wireshark pcap:



In the screenshot you can see that the port specification is mentioned in the "Datagram" message which is part of the "Transport" responsibility.

the dst port is : "12345" just like we configured in the "Client.py" file and the source port is some random port that the OS chose for the client machine – like expected and just like what Hemi said in the recitation

Server.py, Line 4

Same as client.py, but the server listens to the port 12345 (source instead of destination).

The screenshot displays the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The toolbar contains icons for various functions like opening files, saving, and zooming. The main window is divided into three panes:

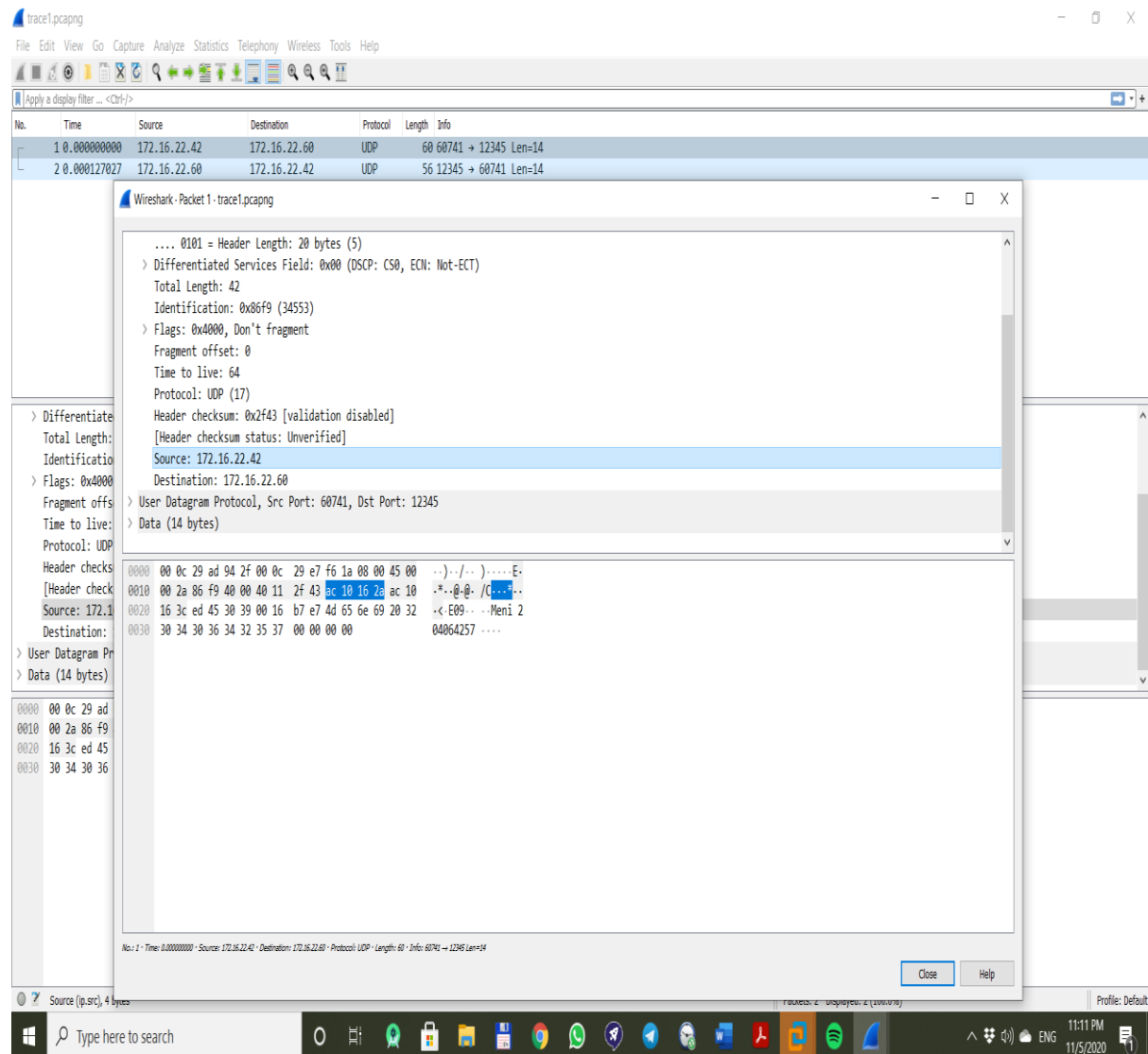
- Packet List:** Shows a list of captured packets. The second packet is selected, indicating it is the current packet being analyzed.
- Packet Details:** Provides a hierarchical view of the selected packet's structure. It shows the Ethernet II header, Internet Protocol Version 4 header, and the User Datagram Protocol (UDP) header. The UDP header details include Source Port (12345), Destination Port (60741), Length (22), and Checksum (0x84ae).
- Packet Bytes:** Displays the raw data of the selected packet in hexadecimal and ASCII format. The data is shown as a sequence of bytes, with the first few bytes being 0000, 000c, 29e7, f61a, 000c, 29ad, 942f, 0800, 4500, and so on.

The status bar at the bottom of the Wireshark window provides additional information about the selected packet, including its time (0.000127027), source (172.16.22.60), destination (172.16.22.42), protocol (UDP), length (56), info (12345 -> 60741 Len=14), and packet size (14 bytes).

the dst port is : "60741" which is the port that the Client used during the conversation with the server and sent the original message and the source port now is the port which the server is using to listen to intercept incoming clients sockets - just like expected and just like what Hemi said in the recitation

4. In this part I need to demonstrate how the ip address is used

o.k, lets see the relevant packet from the sniff of the conversation between the server and the client:



We can see that the information about the source and dst ip addresses are shown at the packet tab from the sniff : the ip address is used in the "Network" layer in order to guide the router to the recipient of the message which, in our case is the server. So we can see that the source ip addr is the ip addr of the client. Lets run "ifconfig" command in terminal of the "Client" machine and prove you that:

The screenshot shows a Linux desktop with a terminal window open. The terminal displays the output of the `ifconfig` command for the `ens33` interface, showing its IP address as `172.16.22.42`. Below this, the output of `python3 Client.py` is shown, displaying a Wireshark packet capture of a ping request from the client to the server. The packet is labeled `b'MENI 204064257' ('172.16.22.60', 12345)`. The desktop background is a red and purple gradient, and the terminal window has a dark background with green text.

```
meni@ubuntu: ~/Networking
python3 Client.py
^Z
[2]+  Stopped                  python3 Client.py
meni@ubuntu:~/Networking$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 172.16.22.42  netmask 255.255.254.0  broadcast 172.16.23.255
    inet6 fe80::99fb:3321:dd71:c03c  prefixlen 64  scopeid 0x20<link>
    ether 08:0c:29:e7:f6:1a  txqueuelen 1000  (Ethernet)
    RX packets 104370  bytes 146149349 (146.1 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 6321  bytes 554950 (554.9 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 472  bytes 48490 (48.4 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 472  bytes 48490 (48.4 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

meni@ubuntu:~/Networking$ python3 Client.py
b'MENI 204064257' ('172.16.22.60', 12345)
```

"Client.py" selected (191 bytes)

So we see that the NAT network name is "ens33" and the server machine also connected to that network.

Here you have the proof the client's ip addr is "172.16.22.42" which is shown in the wireshark sniff