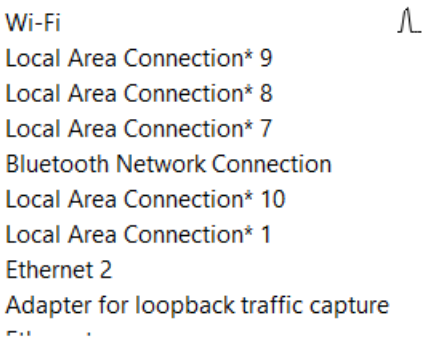
Assignment 1

1. **Do the following before you start attempting the exercise questions, here:**
   (a) Start up your web browser - not in the https mode.
   (b) Start up the Wireshark packet sniffer, but don't yet begin packet capture. Enter "http" (just the letters, not the quotation marks) in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window. (We're only interested in the HTTP protocol here, and don't want to see the clutter of all captured packets).
   (c) Wait a bit more than one minute (we'll see why shortly), and then begin Wireshark packet capture.
   (d) Enter the following to your browser http://gaia.cs.umass.edu/wireshark-labs/ HTTP-wireshark-file1.html Your browser should display the very simple, one-line HTML file.
   (e) Stop Wireshark packet capture.

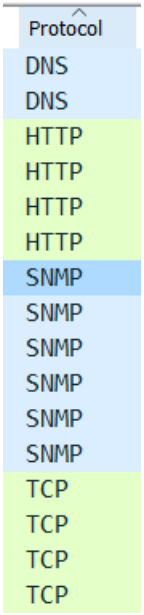
a) **What are the network interfaces available on your computer? Which network did you eventually select in your experiments.**

Available network interfaces:



For this experiment, I have selected Wi-Fi interface.

b) Which application layer protocol is used in this case?



Application layer protocol used: DNS, HTTP, SNMP

c) What are the other protocols used and displayed in the unfiltered packet listing window of Wireshark, besides the one that you answered in Q(b)?

Other protocols used: TCP

d) What is the IPA of your machine? What is the IPA of the destination machine? Is there any way by which you can ascertain that the IPA of the destination indeed is the same as that you observed in Wireshark? If so, how?

Source IPA: 192.168.1.102
Destination IPA: 128.119.245.12

```
C:\Users\nihar>ping gaia.cs.umass.edu

Pinging gaia.cs.umass.edu [128.119.245.12] with 32 bytes of data:
Reply from 128.119.245.12: bytes=32 time=277ms TTL=37
Reply from 128.119.245.12: bytes=32 time=275ms TTL=37
Reply from 128.119.245.12: bytes=32 time=279ms TTL=37
Reply from 128.119.245.12: bytes=32 time=286ms TTL=37

Ping statistics for 128.119.245.12:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 275ms, Maximum = 286ms, Average = 279ms

C:\Users\nihar>
```

Yes, we can ascertain by sending ping request on that domain and we can see the ip address of that domain in response.

e) What is the class of the IPA of the source machine? That of destination machine?

Range of 1st octet of class A = [1, 127]

Range of 1st octet of class B = [128, 191]

Range of 1st octet of class C = [192, 223]

Range of 1st octet of class D = [224, 239]

Range of 1st octet of class E = [240, 255]

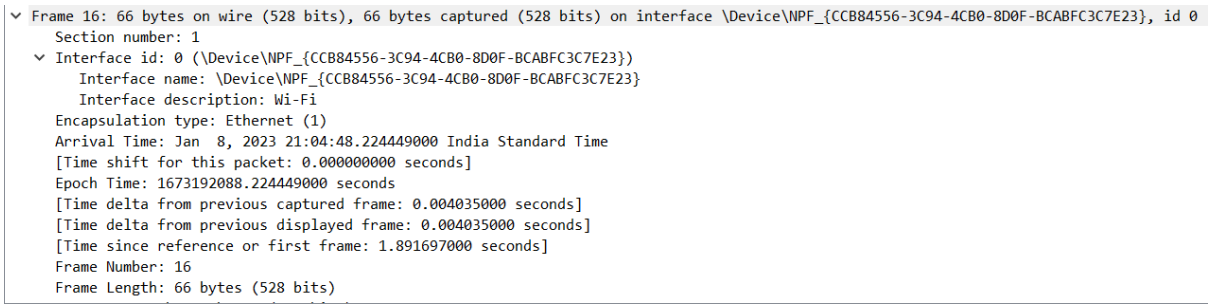
Class of Source machine is C.
Class of destination machine is B.

f) How many bits were captured in this packet? At what time was this packet captured?

```
Frame 10: 555 bytes on wire (4440 bits), 555 bytes captured (4440 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Sep 23, 2003 10:59:47.838388000 India Standard Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1064294987.838388000 seconds
    [Time delta from previous captured frame: 0.000392000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 4.694850000 seconds]
    Frame Number: 10
    Frame Length: 555 bytes (4440 bits)
    Capture Length: 555 bytes (4440 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp:http]
    [Coloring Rule Name: HTTP]
    [Coloring Rule String: http || tcp.port == 80 || http2]
```

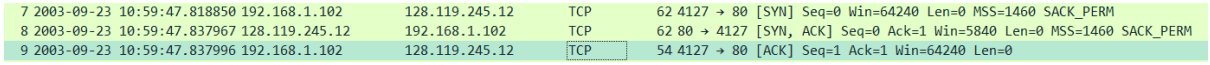
4440 bits were captured in this packet at Sep 23, 2003 10:59:47.838388000 India Standard Time.

g) What is the interface id used? What is the address of the interface?



Interface id: 0
Interface address: \Device\NPF_{CCB84556-3C94-4CB0-8D0F-BCABFC3C7E23}

h) Which packets are forming the TCP 3-way handshake for connection establishment? What are the SYN and ACK in each of the three packets?



Packet 1

SYN: 1

ACK: 0

Packet 2

SYN: 1

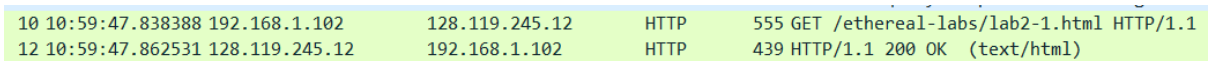
ACK: 1

Packet 3

SYN: 0

ACK: 1

i) How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received? (By default, the value of the Time column in the packet-listing window is the amount of time, in seconds, since Wireshark tracing began. To display the Time field in time-of-day format, select the Wireshark View pull down menu, then select Time Display Format, then select Time-of-day.)



Time taken: 0.001704 seconds

j) Print the two HTTP messages (GET and OK) referred to in question above. To do so, select Print from the Wireshark File command menu, and select the "Selected Packet Only" and "Print as displayed" radial buttons, and then click OK.

```
No.     Time              Source                Destination           Protocol Length Info
     10 10:59:47.838388    192.168.1.102         128.119.245.12        HTTP     555    GET /ethereal-labs/lab2-1.html HTTP/1.1
Frame 10: 555 bytes on wire (4440 bits), 555 bytes captured (4440 bits)
Ethernet II, Src: Dell_4f:36:23 (00:08:74:4f:36:23), Dst: LinksysG_da:af:73 (00:06:25:da:af:73)
Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.119.245.12
Transmission Control Protocol, Src Port: 4127, Dst Port: 80, Seq: 1, Ack: 1, Len: 501
Hypertext Transfer Protocol
No.     Time              Source                Destination           Protocol Length Info
     12 10:59:47.862531    128.119.245.12        192.168.1.102         HTTP     439    HTTP/1.1 200 OK  (text/html)
Frame 12: 439 bytes on wire (3512 bits), 439 bytes captured (3512 bits)
Ethernet II, Src: LinksysG_da:af:73 (00:06:25:da:af:73), Dst: Dell_4f:36:23 (00:08:74:4f:36:23)
Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.1.102
Transmission Control Protocol, Src Port: 80, Dst Port: 4127, Seq: 1, Ack: 502, Len: 385
Hypertext Transfer Protocol
Line-based text data: text/html (3 lines)
```

k) What is the destination physical address of the first packet captured? What device does it belong to? Show where in the capture would you find this information.

```
> Frame 10: 555 bytes on wire (4440 bits), 555 bytes captured (4440 bits)
v Ethernet II, Src: Dell_4f:36:23 (00:08:74:4f:36:23), Dst: LinksysG_da:af:73 (00:06:25:da:af:73)
  v Destination: LinksysG_da:af:73 (00:06:25:da:af:73)
      Address: LinksysG_da:af:73 (00:06:25:da:af:73)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  > Source: Dell_4f:36:23 (00:08:74:4f:36:23)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 4127, Dst Port: 80, Seq: 1, Ack: 1, Len: 501
> Hypertext Transfer Protocol
```

Destination Physical Address: 00:06:25:da:af:73

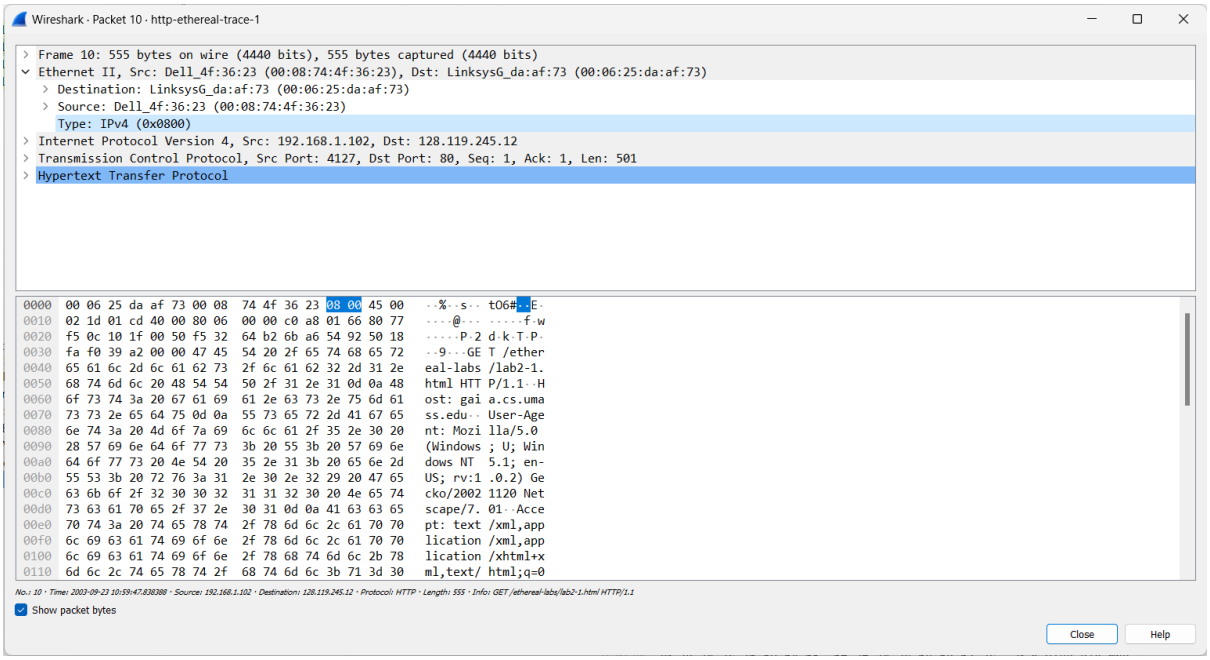
l) How many bytes of header does the first frame sent have? Show where in the capture would you find this information.

```
> Frame 10: 555 bytes on wire (4440 bits), 555 bytes captured (4440 bits)
> Ethernet II, Src: Dell_4f:36:23 (00:08:74:4f:36:23), Dst: LinksysG_da:af:73 (00:06:25:da:af:73)
v Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.119.245.12
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 541
    Identification: 0x01cd (461)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.102
    Destination Address: 128.119.245.12
> Transmission Control Protocol, Src Port: 4127, Dst Port: 80, Seq: 1, Ack: 1, Len: 501
```

Header Length: 20 bytes

m) By looking at the Ethernet header of a frame, can we determine if it contains an IP packet? Show where in the capture would you find this information.

Yes, we can determine if a frame contains an IP packet by looking at the Ethernet Header of a frame.
The following is the Ethernet Header of a HTTP packet

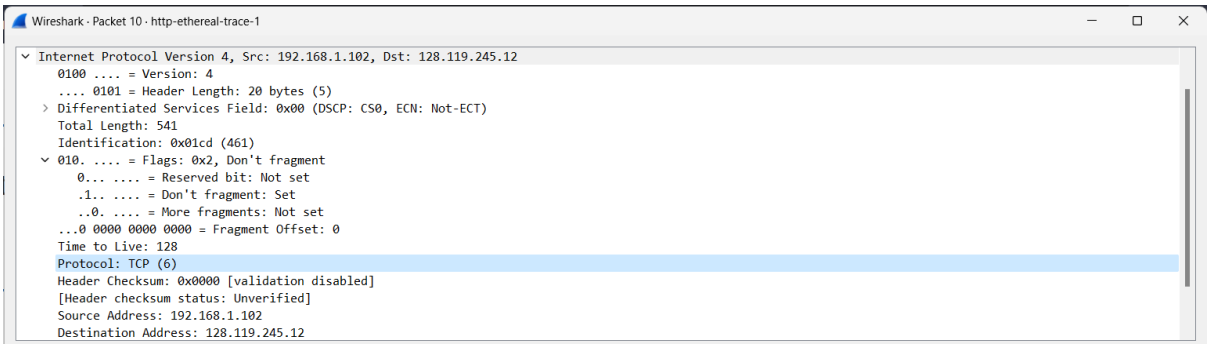


The packet shows IPv4 as the Type

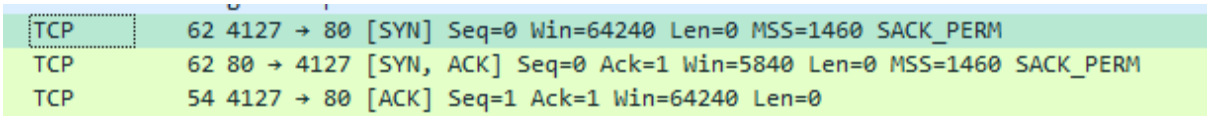
n) Is it possible to know if the first packet captured has TCP or UDP as transport protocol by looking at the IP header? Explain and show where in the capture would you find this information.

Yes, it is possible to see if the packet has TCP or UDP by looking at the IP header.

The following is the first packet captured, which has TCP.



o) In the SYN, ACK. What are the source and destination ports? Are these the same for the client and the server? Explain why.



TCP 3-way handshake for connection establishment first SYN packet from client

Source port: 4127

Destination port: 80

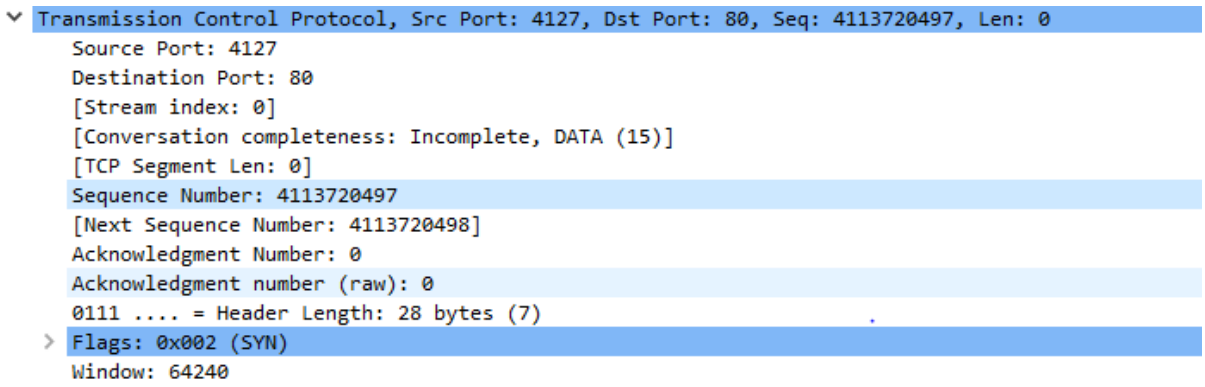
TCP 3-way handshake for connection establishment second SYN, ACK packet

Source port: 4127

Destination port: 80

p) Why does the Server Hello message sent by the server have 1 as a relative sequence number and 185 as a relative acknowledgement number.

q) Right-click a TCP capture -> TCP preferences -> Uncheck the box "Show relative sequence number." What is the first sequence number sent by the server to the client? Why is it not the 0 displayed by Wireshark?

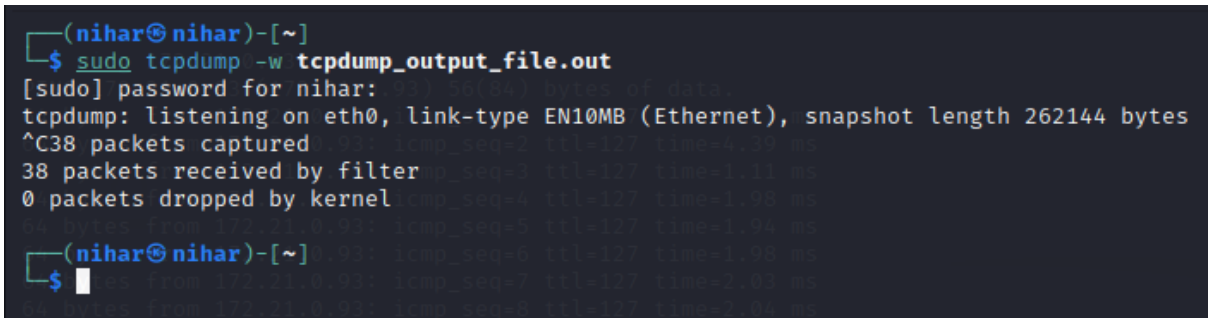


The first sequence number sent by the server to the client: 4113720497

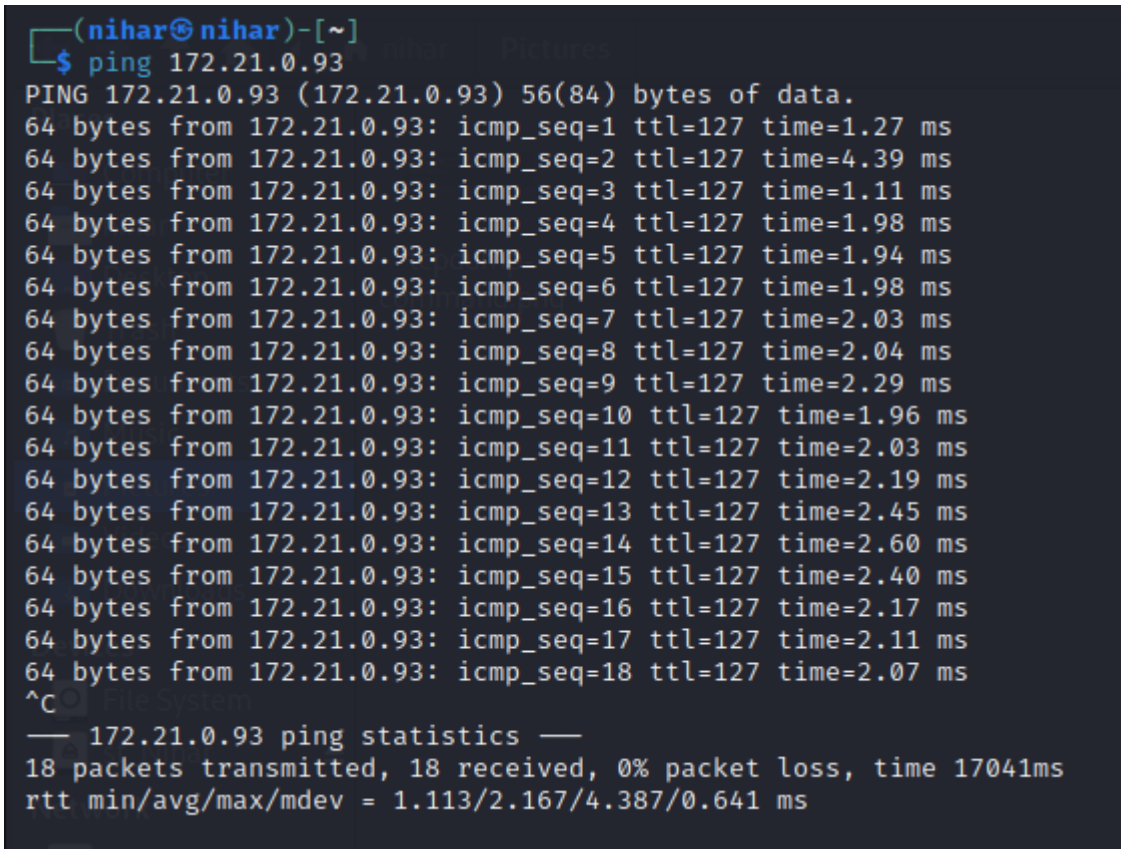
The sequence number is not taken zero for every TCP connection to avoid confusion and overlap between different connections, and prevent easy connection hijacking.

2. **This exercise is a simple exercise that only requires you to capture the tcpdump track. The problem requires you to either use two virtual machines on your laptop or two different machines in the computer lab ask the administrator for the host name of both the machines, if so. Then run the tcpdump command on one machine say PC1 (saving the output for your lab report) so that it monitors all the packets that contain the IP address of PC2 only and none else. Next, open a new terminal window on PC1 and execute ping command to PC2. It may be necessary to press Ctrl + C to terminate the tcpdump session. It may sometimes be best to simply redirect the output of tcpdump straight to a file and view it afterward with the more command or a text editor. Find out how can you do so. Run the command $tcpdump -enx -w tcpdump_out_file.out & do you see any output on the screen? Why?**

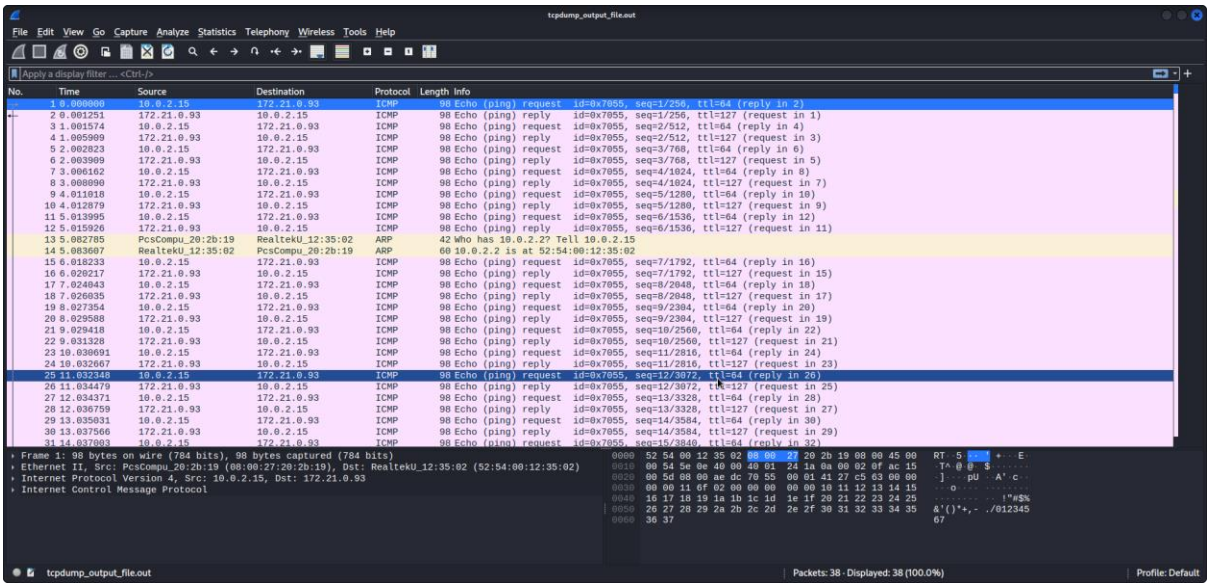
Step – 1 Start tcpdump



Step – 2 Ping request started



Step – 3 Captured Packets



3. **This question is in continuation of the question no 2. Run the command "telnet remote host". remotehost is the host name of either another virtual machine in your machine or it is the host name of any other machine in the network used in the lab (Ask the lab technical suport staff about the name of other machine). This command would generate some TCP trac. After you login to the remote machine, terminate the telnet session and terminate the tcpdump program.**

   • Click on a TCP packet from the list of captured packets in the wireshark window. Then go to the Edit menu and choose Mark Frame.

   • Go to the File menu and choose Print. In the Wireshark: Print dialog that pops up, check File, Plain Text, expand all levels, print detail and supress unmarked frames. Then, enter the output text file name, e.g., headers.txt, and click the OK button. The marked packet is now dumped into the text file, with a detailed list of the name and value of every field in all the three headers.

Step 1 – Telnet server creation

```
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~$ sudo systemctl restart xinetd.service
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~$ sudo systemctl status xinetd.service
● xinetd.service - LSB: Starts or stops the xinetd daemon.
     Loaded: loaded (/etc/init.d/xinetd; generated)
     Active: active (running) since Thu 2023-01-19 16:09:24 IST; 3s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 9712 ExecStart=/etc/init.d/xinetd start (code=exited, status=0/SUCCESS)
      Tasks: 2 (limit: 18847)
     Memory: 2.7M
     CGroup: /system.slice/xinetd.service
             ├─9410 in.telnetd: 172.21.3.209
             └─9722 /usr/sbin/xinetd -pidfile /run/xinetd.pid -stayalive -inetd_compat -inetd_ipv6

Jan 19 16:09:24 adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC xinetd[9722]: Reading included configuration file: /etc/xinetd.d/discard-udp [file=/etc/xinetd.d/discard-udp] [line=25]
Jan 19 16:09:24 adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC xinetd[9722]: Reading included configuration file: /etc/xinetd.d/echo [file=/etc/xinetd.d/echo] [line=14]
Jan 19 16:09:24 adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC xinetd[9722]: Reading included configuration file: /etc/xinetd.d/echo-udp [file=/etc/xinetd.d/echo-udp] [line=26]
Jan 19 16:09:24 adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC xinetd[9722]: Reading included configuration file: /etc/xinetd.d/servers [file=/etc/xinetd.d/servers] [line=14]
Jan 19 16:09:24 adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC xinetd[9722]: Reading included configuration file: /etc/xinetd.d/services [file=/etc/xinetd.d/services] [line=13]
Jan 19 16:09:24 adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC xinetd[9722]: Reading included configuration file: /etc/xinetd.d/telnet [file=/etc/xinetd.d/telnet] [line=13]
Jan 19 16:09:24 adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC xinetd[9722]: Reading included configuration file: /etc/xinetd.d/time [file=/etc/xinetd.d/time] [line=19]
Jan 19 16:09:24 adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC xinetd[9722]: Reading included configuration file: /etc/xinetd.d/time-udp [file=/etc/xinetd.d/time-udp] [line=28]
Jan 19 16:09:24 adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC xinetd[9722]: 2.3.15.3 started with libwrap loadavg labeled-networking options compiled in.
Jan 19 16:09:24 adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC xinetd[9722]: Started working: 1 available service
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~$ sudo ufw allow 23
Skipping adding existing rule
Skipping adding existing rule (v6)
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~$ sudo ufw allow from 172.21.3.209 to any port 23
Skipping adding existing rule
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~$
```

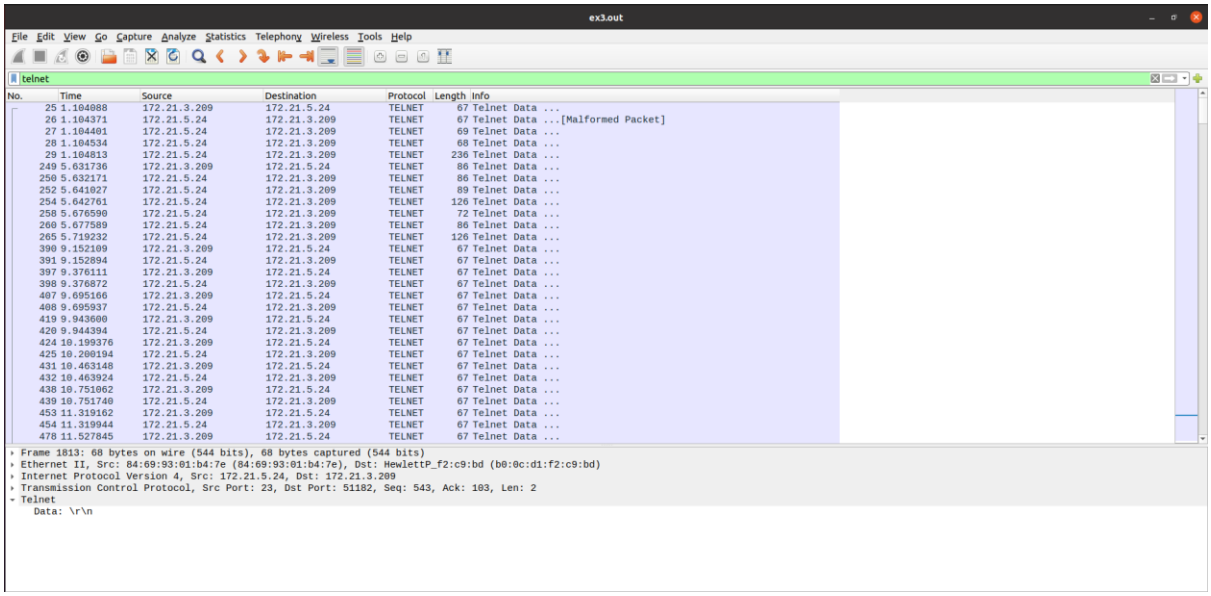
Step 2 – Starting of tcpdump server on server side

```
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~/Desktop$ tcpdump -enx -w ex3.out
tcpdump: eno1: You don't have permission to capture on that device
(socket: Operation not permitted)
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~/Desktop$ sudo tcpdump -enx -w ex3.out
tcpdump: listening on eno1, link-type EN10MB (Ethernet), capture size 262144 bytes
^C2071 packets captured
2078 packets received by filter
0 packets dropped by kernel
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~/Desktop$
```

Step 3 – Trying to connect from client side and getting access of server files

```
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~$ telnet 172.21.5.24
Trying 172.21.5.24...
Connected to 172.21.5.24.
Escape character is '^]'.
Ubuntu 20.04.3 LTS
adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC login: adminisarator
Password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.15.0-56-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

141 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Thu Jan 19 16:06:27 IST 2023 from 172.21.3.209 on pts/2
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~$ ls
 Android                 a.out   binary     bubblesort.c   Documents   linear.c            Mus
 AndroidStudioProjects   ass     binary.c   Desktop        Downloads   'linear search.c'    nam
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~$ cd Desktop/
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~/Desktop$ ls
ex3.out  tmp.out
```

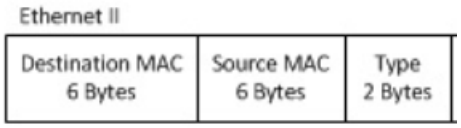
Step 4 – Captured Wireshark package



Now answer the following questions:

a) **Draw the format of the packet you saved, including the link, IP, and TCP headers, and identify the value of each field in these headers. Express the values in the decimal format.**

Ethernet Packet Format:

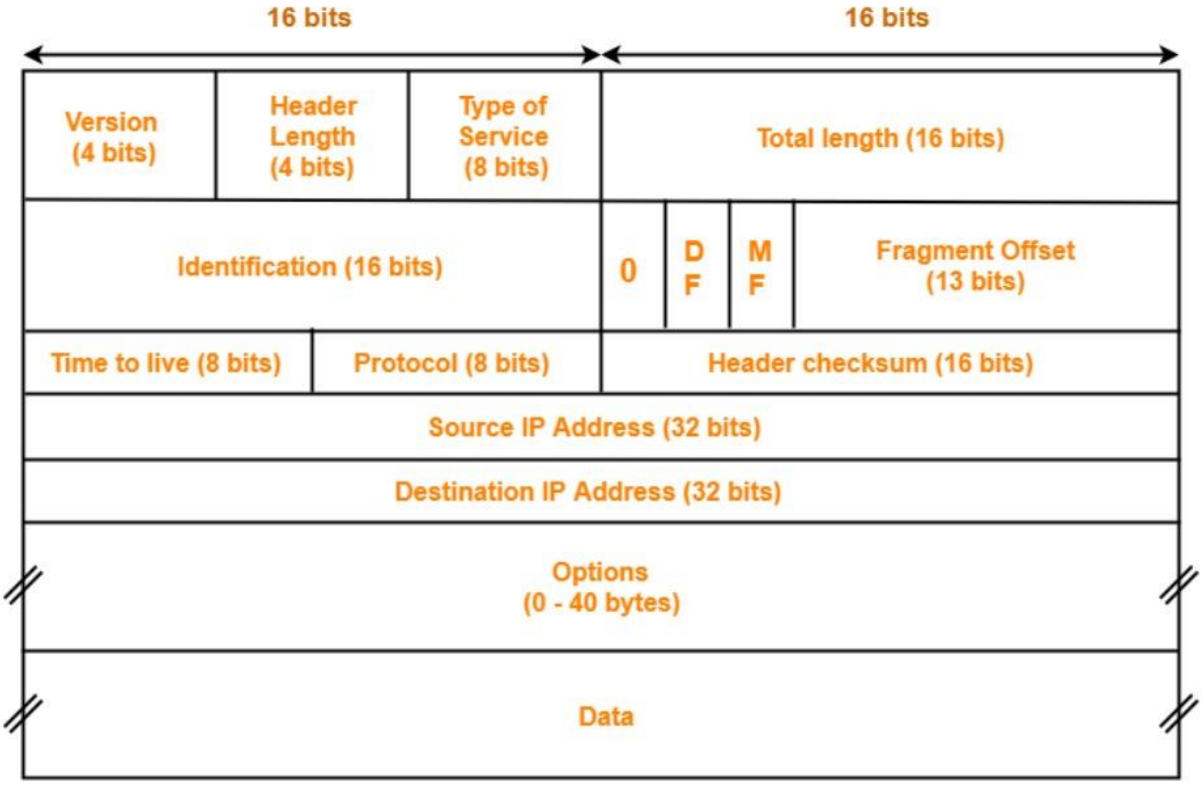


Destination: HP_01:b4:7e (84:69:93:01:b4:7e)

Source: HewlettP_f2:c9:bd (b0:0c:d1:f2:c9:bd)

Type: IPv4 (0x0800)

IP Packet Format:



Version: 4

Header Length: 20 bytes

Type of Service: 0

Total Length: 53

Identification: 0597111

Flags:

Do not Fragment: 1

More Fragment:

Fragment Offset: 0

Time to Live: 64

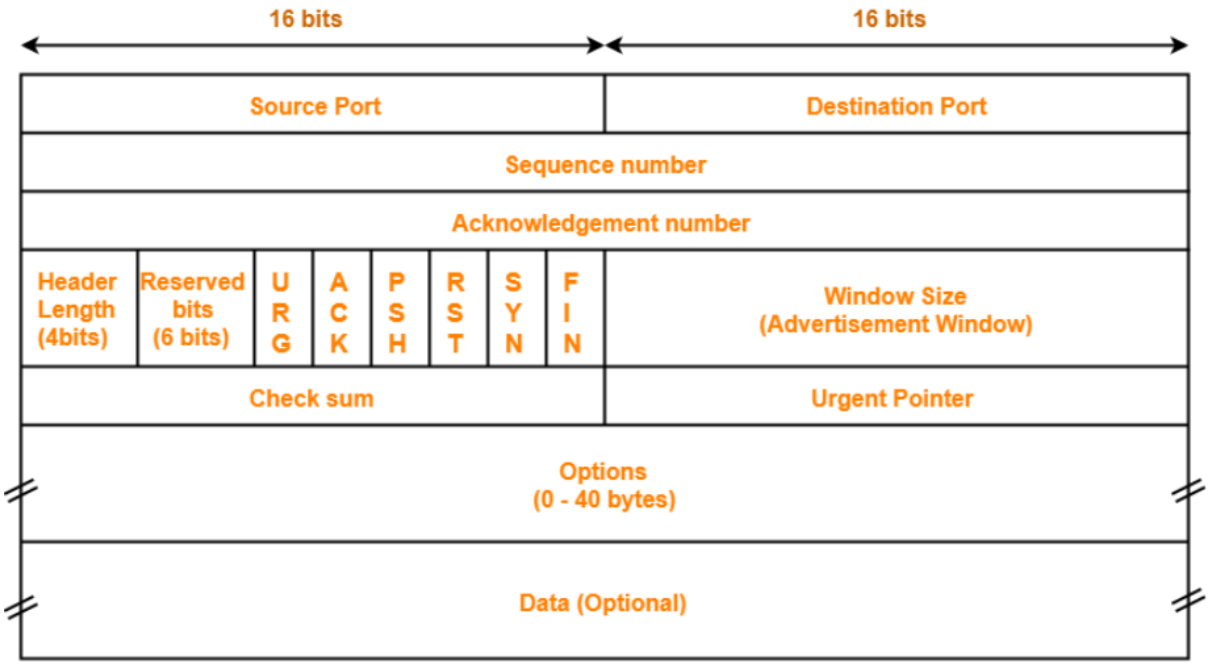
Protocol: TCP (6)

Header Checksum: 61475

Source Address: 172.21.3.209

Destination Address: 172.21.5.24

TCP Packet Format:



Source Port: 5182

Destination Port: 23

Sequence Number: 3109923138

Ack Number: 1773035314

Header Length: 32 Bytes

Reserved bits: 0

URG: 0

ACK: 1

PSH: 1

RST: 0

SYN: 0

FIN: 0

Window Size: 501

Check Sum: 37867

Urgent Pointer: 0

Options: 12 Bytes

b) What is the value of the protocol field in the IP header of the packet you saved? What is the use of the protocol field?

Protocol is an 8-bit field. It tells the network layer at the destination host to which protocol the IP datagram belongs to. In other words, it tells the next level protocol to the network layer at the destination side. Protocol number of ICMP is 1, IGMP is 2, TCP is 6 and UDP is 17.

In this case the protocol field containing number 6 signifies TCP protocol.

4. **In a manner similar to the Exercise no 3, now run tcpdump to capture an ARP request and an ARP reply and then use Wireshark to analyse the frames. If there are no arp requests and replies in the network, generate some using arpinga - remote - machine: After you see several ARP replies in the arping output, terminate the arping and the tcpdump program. Open the tcpdump trace using $wireshark - r exe4:out &: Print one ARP request and one ARP reply using wireshark.**

ARP request and ARP reply

Step - 1: Check configuration of receiver machine.

```
nisargdevani@nisargdevani-HP-Laptop-15-da0xxx:~$ ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.21.3.209  netmask 255.255.240.0  broadcast 172.21.15.255
        inet6 fe80::3b15:7c76:9058:f3dc  prefixlen 64  scopeid 0x20<link>
        ether b0:0c:d1:f2:c9:bd  txqueuelen 1000  (Ethernet)
        RX packets 97830  bytes 91147032 (91.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 60327  bytes 10048543 (10.0 MB)
        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 3146  bytes 6351675 (6.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3146  bytes 6351675 (6.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlo1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.42.0.1  netmask 255.255.255.0  broadcast 10.42.0.255
        inet6 fe80::885a:9e61:d834:9fec  prefixlen 64  scopeid 0x20<link>
        ether dc:f5:05:63:d8:97  txqueuelen 1000  (Ethernet)
        RX packets 29513  bytes 4132199 (4.1 MB)
        RX errors 0  dropped 5  overruns 0  frame 0
        TX packets 41494  bytes 43696519 (43.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

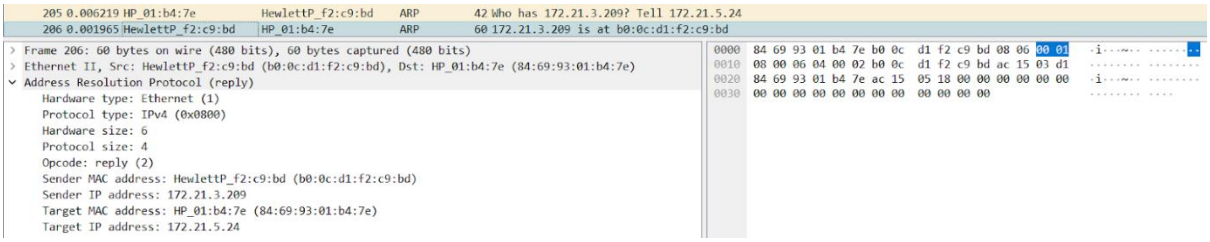
Step - 2: Start tcpdump on sender machine.

```
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~$ sudo tcpdump -enx -w exe4.out
tcpdump: listening on eno1, link-type EN10MB (Ethernet), capture size 262144 bytes
^C306 packets captured
306 packets received by filter
0 packets dropped by kernel
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~$ wireshark exe4.out &
[1] 4476
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~$
```

Step - 3: Ping to receiver machine from sender machine.

```
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~$ ping 172.21.3.209
PING 172.21.3.209 (172.21.3.209) 56(84) bytes of data.
64 bytes from 172.21.3.209: icmp_seq=1 ttl=64 time=1.83 ms
64 bytes from 172.21.3.209: icmp_seq=2 ttl=64 time=2.25 ms
64 bytes from 172.21.3.209: icmp_seq=3 ttl=64 time=2.00 ms
64 bytes from 172.21.3.209: icmp_seq=4 ttl=64 time=1.99 ms
64 bytes from 172.21.3.209: icmp_seq=5 ttl=64 time=2.24 ms
64 bytes from 172.21.3.209: icmp_seq=6 ttl=64 time=2.07 ms
64 bytes from 172.21.3.209: icmp_seq=7 ttl=64 time=2.01 ms
64 bytes from 172.21.3.209: icmp_seq=8 ttl=64 time=2.05 ms
64 bytes from 172.21.3.209: icmp_seq=9 ttl=64 time=0.891 ms
64 bytes from 172.21.3.209: icmp_seq=10 ttl=64 time=1.88 ms
64 bytes from 172.21.3.209: icmp_seq=11 ttl=64 time=1.88 ms
64 bytes from 172.21.3.209: icmp_seq=12 ttl=64 time=1.79 ms
^C
--- 172.21.3.209 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11016ms
rtt min/avg/max/mdev = 0.891/1.905/2.247/0.335 ms
adminisarator@adminisarator-HP-ProDesk-400-G7-Small-Form-Factor-PC:~$
```

Step - 4: Find ARP request/reply in captured packets.



Now answer the following questions:

(a) What is the value of the frame type field in an Ethernet frame carrying an ARP request and in an Ethernet frame carrying an ARP reply, respectively?

The value of the frame type field in an Ethernet frame carrying an ARP request is ARP and in an Ethernet frame carrying an ARP reply is ARP.

(b) What is the value of the frame type field in an Ethernet frame carrying an IP datagram captured in the previous exercise?

Frame type: Ipv4

(c) What is the use of the frame type field?

The frame type field is a field in the link-layer (layer 2) header of a packet that is used to identify the type of packet that is being transmitted. The frame type field is used by network devices to determine how to process a packet and what protocol is being used.

5. Explain briefly the purposes of the following tcpdump expressions.

i. tcpdump udp port 520

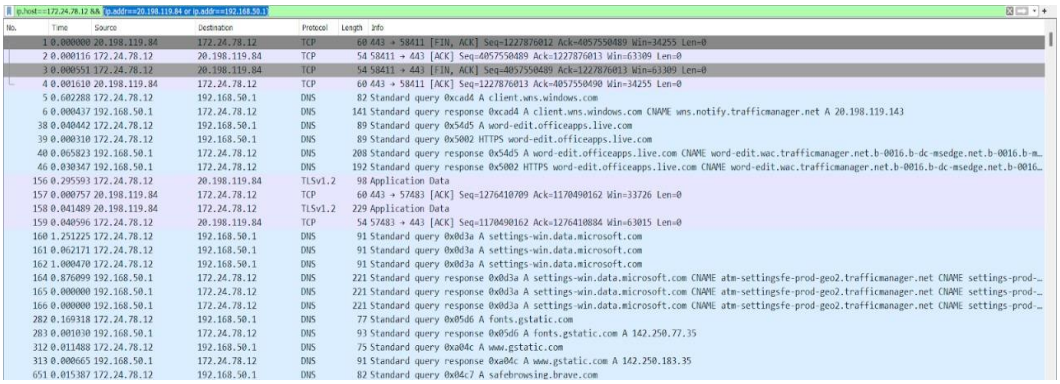
Tcpdump is used to capture and analyze network packets.
So, purpose of given expression is to capture UDP packet with port number 520 which is used by routing information protocol which uses hop count as a routing metric to find the best path between the source and the destination network.

ii. tcpdump -x -s 120 ip proto 89

-x command : When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex
-s sLen : Prints slen bytes of data from each packet rather than the default of 262144 bytes
Proto 89 is protocol code for OSPF.
So purpose of given expression is to capture and print 120 bytes data of each packet which is using OSPF protocol

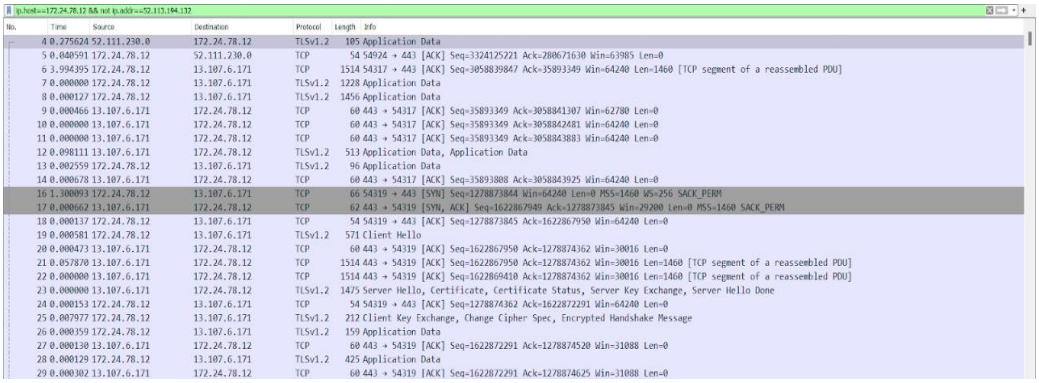
iii. tcpdump -x -s 70 host ip addr1 and (ip addr2 or ip addr3)

Command host ip_addr1 will match that address with source or destination address. After matching that command ip_addr2 or ip_adrr3 will search that ip in the remainder of the source or destination.
For example,

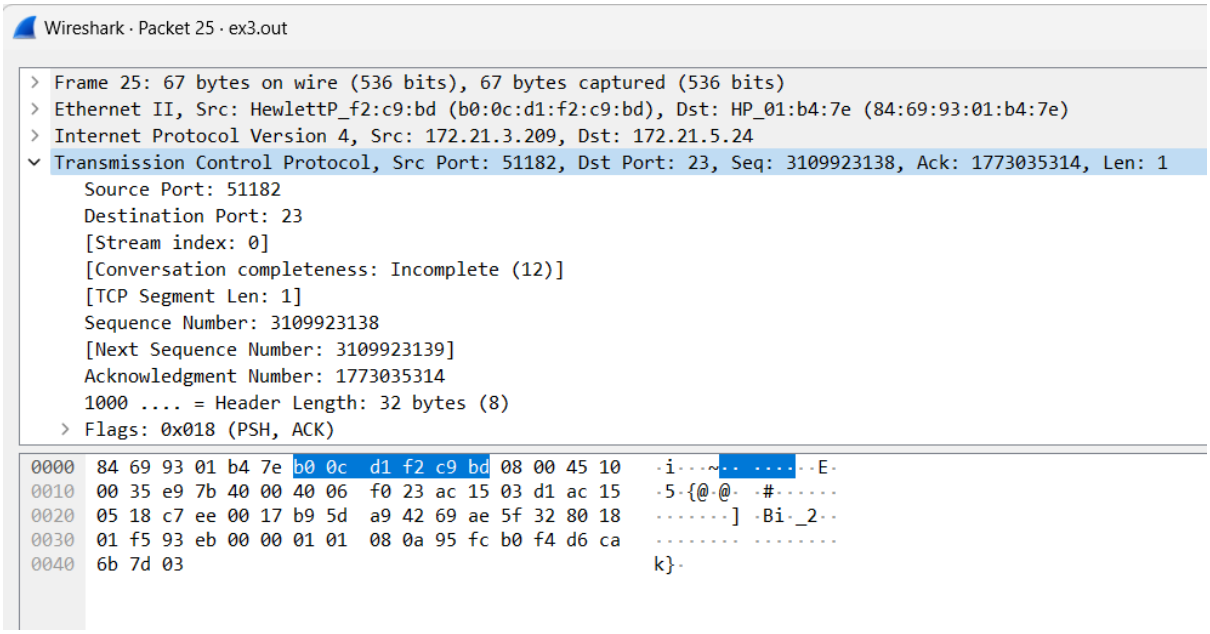


iv. tcpdump -x -s 70 host ip addr1 and not ip addr2

Command host ip_addr1 will match that address with source or destination address. After matching that command, ip_addr2 will search that ip not in remaining of the source or destination.
For example,


ip.host==172.24.78.12 && not ip.addr==52.113.194.132
No. Time Source Destination Protocol Length Info
4 0.275624 52.111.230.0 172.24.78.12 TLSv1.2 105 Application Data
5 0.040591 172.24.78.12 52.111.230.0 TCP 54 54924 → 443 [ACK] Seq=3324125221 Ack=2806716301 Win=63985 Len=0
6 3.994395 172.24.78.12 13.107.6.171 TCP 1514 54317 → 443 [ACK] Seq=3058839847 Ack=35893349 Win=64240 Len=1460 [TCP segment of a reassembled PDU]
7 0.000000 172.24.78.12 13.107.6.171 TLSv1.2 1228 Application Data
8 0.000127 172.24.78.12 13.107.6.171 TLSv1.2 1456 Application Data
9 0.000466 13.107.6.171 172.24.78.12 TCP 60 443 → 54317 [ACK] Seq=35893349 Ack=3058841307 Win=62780 Len=0
10 0.000000 13.107.6.171 172.24.78.12 TCP 60 443 → 54317 [ACK] Seq=35893349 Ack=3058842481 Win=64240 Len=0
11 0.000000 13.107.6.171 172.24.78.12 TCP 60 443 → 54317 [ACK] Seq=35893349 Ack=3058843883 Win=64240 Len=0
12 0.098111 13.107.6.171 172.24.78.12 TLSv1.2 513 Application Data, Application Data
13 0.002559 172.24.78.12 13.107.6.171 TLSv1.2 96 Application Data
14 0.000678 13.107.6.171 172.24.78.12 TCP 60 443 → 54317 [ACK] Seq=35893808 Ack=3058843925 Win=64240 Len=0
16 1.300093 172.24.78.12 13.107.6.171 TCP 66 54319 → 443 [SYN] Seq=1278873844 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
17 0.000662 13.107.6.171 172.24.78.12 TCP 62 443 → 54319 [SYN, ACK] Seq=1622867949 Ack=1278873845 Win=29200 Len=0 MSS=1460 SACK_PERM
18 0.000137 172.24.78.12 13.107.6.171 TCP 54 54319 → 443 [ACK] Seq=1278873845 Ack=1622867950 Win=64240 Len=0
19 0.000581 172.24.78.12 13.107.6.171 TLSv1.2 571 Client Hello
20 0.000473 13.107.6.171 172.24.78.12 TCP 60 443 → 54319 [ACK] Seq=1622867950 Ack=1278874362 Win=30016 Len=0
21 0.057870 13.107.6.171 172.24.78.12 TCP 1514 443 → 54319 [ACK] Seq=1622867950 Ack=1278874362 Win=30016 Len=1460 [TCP segment of a reassembled PDU]
22 0.000000 13.107.6.171 172.24.78.12 TCP 1514 443 → 54319 [ACK] Seq=1622869410 Ack=1278874362 Win=30016 Len=1460 [TCP segment of a reassembled PDU]
23 0.000000 13.107.6.171 172.24.78.12 TLSv1.2 1475 Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
24 0.000153 172.24.78.12 13.107.6.171 TCP 54 54319 → 443 [ACK] Seq=1278874362 Ack=1622872291 Win=64240 Len=0
25 0.007977 172.24.78.12 13.107.6.171 TLSv1.2 212 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
26 0.000359 172.24.78.12 13.107.6.171 TLSv1.2 159 Application Data
27 0.000130 13.107.6.171 172.24.78.12 TCP 60 443 → 54319 [ACK] Seq=1622872291 Ack=1278874520 Win=31088 Len=0
28 0.000129 172.24.78.12 13.107.6.171 TLSv1.2 425 Application Data
29 0.000302 13.107.6.171 172.24.78.12 TCP 60 443 → 54319 [ACK] Seq=1622872291 Ack=1278874625 Win=31088 Len=0

6. **Start tcpdump in a command window to capture packets between your machine and a remote host using: tcpdump -n -nn host your -host remote -host. Execute any TCP utility,telnet for example - as in the problem before, in another command window. When you see a TCP packet in the tcpdump output, terminate tcpdump and save its output. Now answer the following question:**



(a) What are the port numbers used by the remote and the local computer?

Port number used by local computer : 51182

Port number used by remote host : 23

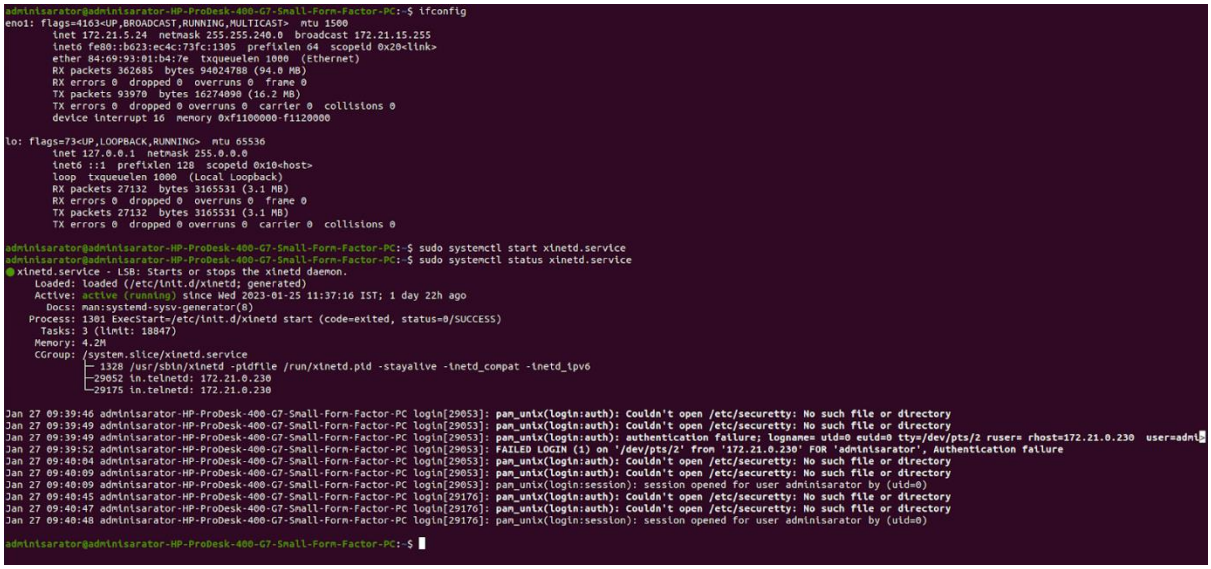
(b) Which machine's port number matches the port number listed for telnet in the /etc/services file? Note: In case telnet is not listed in the /etc/services file, use ssh utility.

Remote host's machine uses port number 23, which is assigned to the telnet protocol.

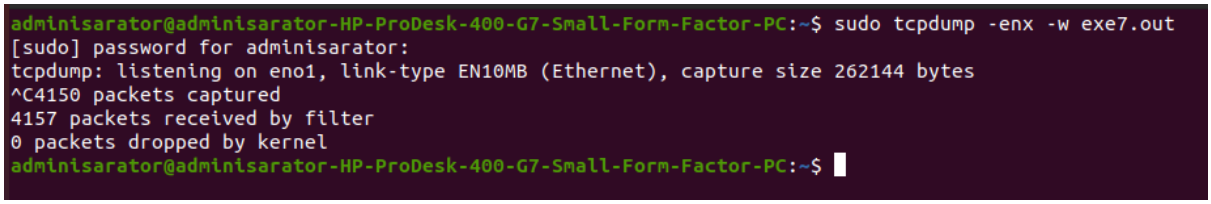
7. **Start tcpdump in one command window using tcpdump -n -nn host your-host remote-host: Then, telnet to the remote host from a second command window by typing telnet remotehost. Again issue the same command from a third command window. Now you are opening two telnet sessions to the same remote host simultaneously, from two different command windows. Check the port numbers being used on both sides of the two connections from the output in the tcpdump window. Save a TCP packet from each of the connections. Now answer the following questions:**

Telnet Configuration

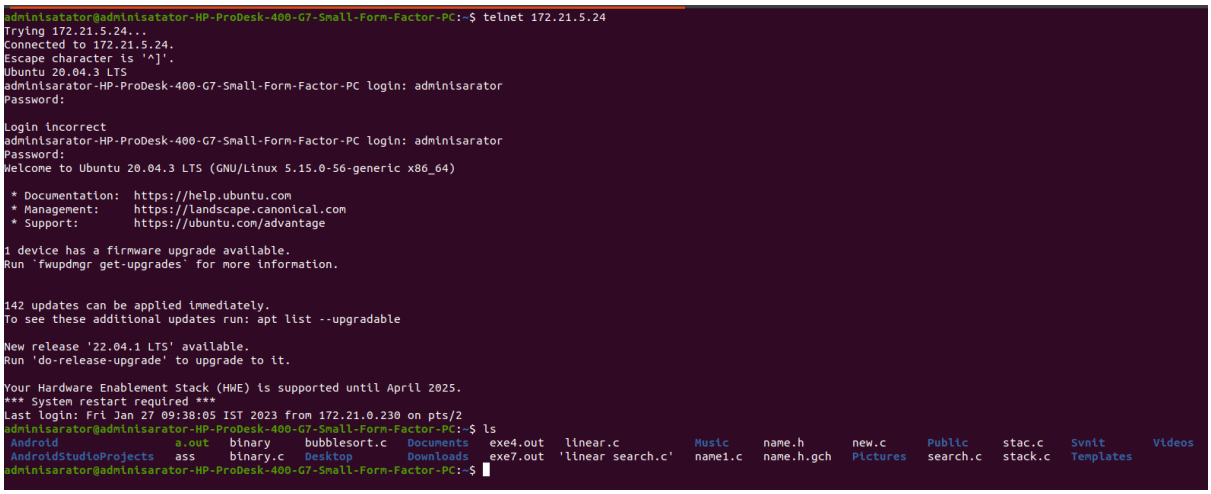
Step - 1: Created Telnet Server



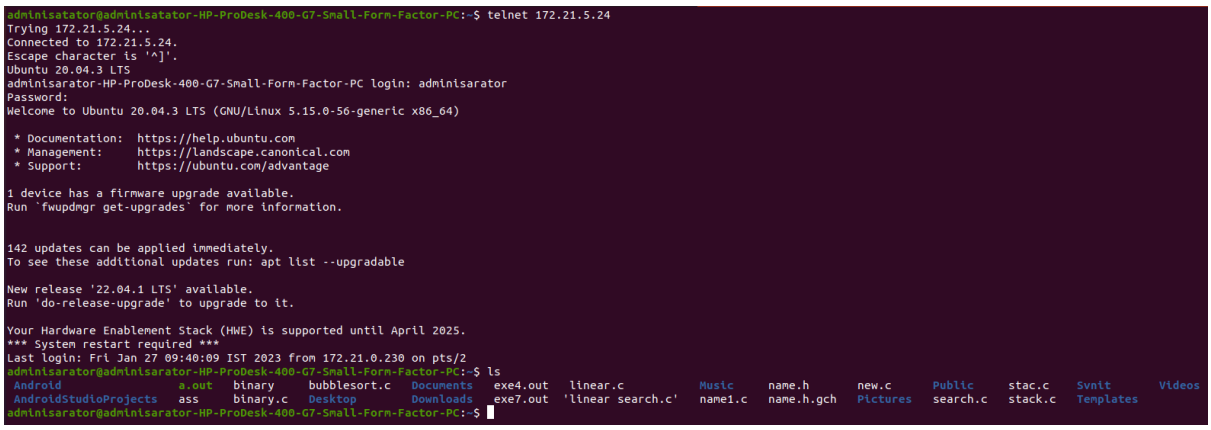
Step - 2: Tcpdump started on the server side.



Step - 3: Trying to connect to telnet server from client window1

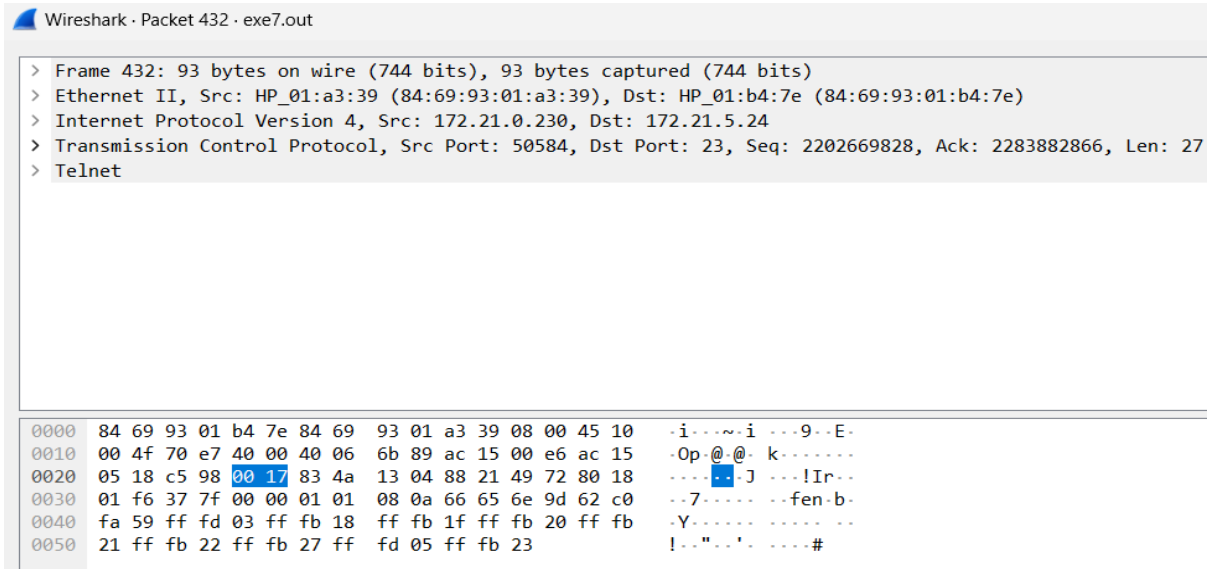


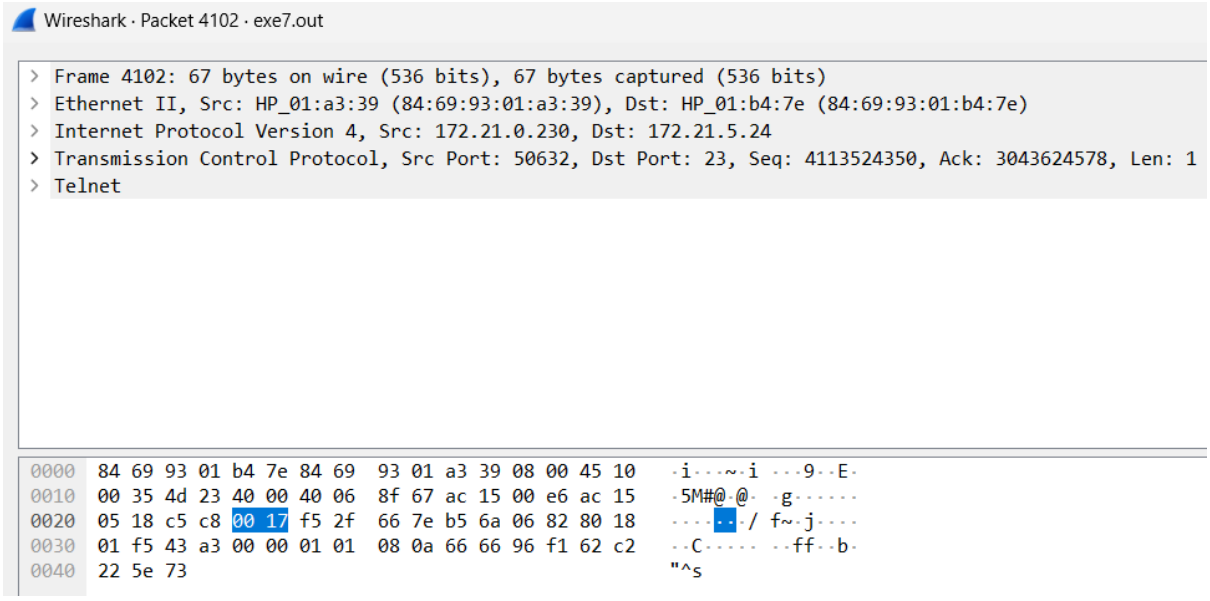
Step - 4: Trying to connect to telnet server from client window2



Step - 5: Analyze captured packets.

Now answer the following questions:





(a) When you have two telnet sessions with your machine, what port number is used on the remote machine? Are both sessions connected to the same port number on the remote machine?

23 port number is used on the remote machine, when we have two telnet sessions with our machine. Yes, both sessions connected to the same port number on the remote machine.

(b) What port numbers are used in your machine for the first and second telnet, respectively?

Port number used by the first telnet is 50584.
Port number used for the second telnet is 50632.

(c) What is the range of Internet-wide well-known port numbers? What is the range of well-known port numbers for Unix/Linux specific service? What is the range for a client port number? Compare your answer to the well-known port numbers defined in the /etc/services file. Are they consistent? In case they are not, try to discuss amongst peers and specify your view of the reason why they are not. Note: In case telnet is not listed in the /etc/services file, use ssh.

8. **Execute the traceroute command with www/yahoo.com as argument. Write down the IP address of yahoo.com that was used for the trace route. Determine the number of iterations required to determine route. Enlist the IP addresses of all the machines between the source and the destination. What is the average round trip time of the packet that reached the destination?**

```
┌──(nihar㉿nihar)-[~]
└─$ sudo traceroute -In yahoo.com
[sudo] password for nihar:
traceroute to yahoo.com (98.137.11.163), 30 hops max, 60 byte packets
 1  10.0.2.2  0.226 ms  0.194 ms  0.187 ms
 2  172.24.83.1  3.657 ms  4.333 ms  4.327 ms
 3  192.168.1.1  3.463 ms  3.586 ms  4.310 ms
 4  172.16.1.1  2.755 ms  2.922 ms  3.431 ms
 5  192.168.168.1  5.118 ms * *
 6  * 117.239.204.225  29.258 ms *
 7  172.24.193.226  29.519 ms  30.451 ms  32.573 ms
 8  * * *
 9  * 61.246.195.185  39.410 ms  39.394 ms
10  116.119.44.134  253.299 ms  253.681 ms *
11  * 206.72.210.195  245.962 ms  246.371 ms
12  209.191.64.125  249.053 ms  247.135 ms  248.340 ms
13  209.191.65.99  255.534 ms *  256.953 ms
14  209.191.65.19  274.631 ms  276.245 ms  276.870 ms
15  * 209.191.65.49  282.151 ms  281.993 ms
16  * 66.196.67.99  299.361 ms  281.065 ms
17  98.136.158.215  309.204 ms  309.533 ms *
18  98.136.159.243  279.201 ms *  282.520 ms
19  98.136.158.193  287.326 ms  278.767 ms 98.136.158.192  279.301 ms
20  98.137.11.163  279.682 ms  276.331 ms *
```

Ip of yahoo.com: 98.137.11.163
Number of iterations required to determine route: 20

IP addresses of all the machines between the source and the destination:
1 10.0.2.2
2 172.24.83.1
3 192.168.1.1
4 172.16.1.1
5 192.168.168.1
6 * 117.239.204.225
7 172.24.193.226
8 * * *
9 * 61.246.195.185
10 116.119.44.134
11 * 206.72.210.195
12 209.191.64.125
13 209.191.65.99
14 209.191.65.19
15 * 209.191.65.49
16 * 66.196.67.99

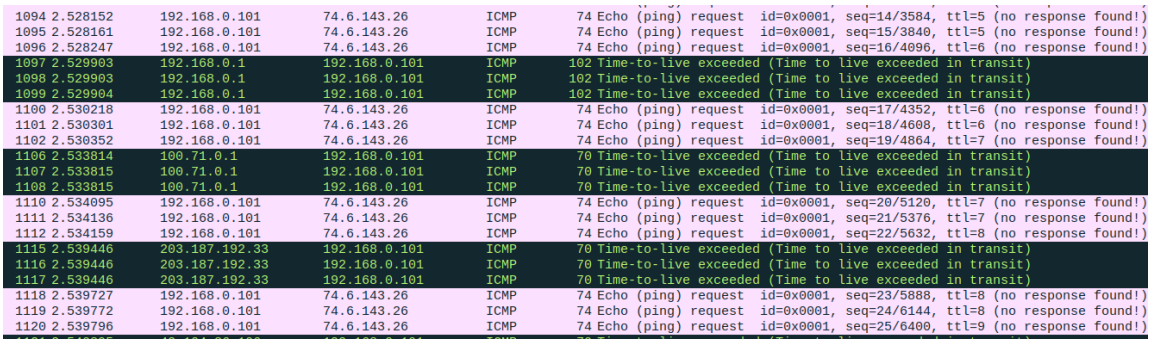
17 98.136.158.215
18 98.136.159.243
19 98.136.158.193
20 98.137.11.163

average round trip time of the packet: 276.5065

9. **With respect to the question above, run traceroute on one window of your OS and run tcpdump on the other window. Analyze the output of tcpdump. Answer the following questions giving appropriate high lighted snapshots in support of your answer :**

   (a) How many packets are sent by traceroute in each iteration ? How can you prove this using the tcpdump output.

   Three packet send by traceroute in each iteration



```
1094 2.528152   192.168.0.101   74.6.143.26     ICMP    74 Echo (ping) request  id=0x0001, seq=14/3584, ttl=5 (no response found!)
1095 2.528161   192.168.0.101   74.6.143.26     ICMP    74 Echo (ping) request  id=0x0001, seq=15/3840, ttl=5 (no response found!)
1096 2.528247   192.168.0.101   74.6.143.26     ICMP    74 Echo (ping) request  id=0x0001, seq=16/4096, ttl=6 (no response found!)
1097 2.529903   192.168.0.1     192.168.0.101   ICMP   102 Time-to-live exceeded (Time to live exceeded in transit)
1098 2.529903   192.168.0.1     192.168.0.101   ICMP   102 Time-to-live exceeded (Time to live exceeded in transit)
1099 2.529904   192.168.0.1     192.168.0.101   ICMP   102 Time-to-live exceeded (Time to live exceeded in transit)
1100 2.530218   192.168.0.101   74.6.143.26     ICMP    74 Echo (ping) request  id=0x0001, seq=17/4352, ttl=6 (no response found!)
1101 2.530301   192.168.0.101   74.6.143.26     ICMP    74 Echo (ping) request  id=0x0001, seq=18/4608, ttl=6 (no response found!)
1102 2.530352   192.168.0.101   74.6.143.26     ICMP    74 Echo (ping) request  id=0x0001, seq=19/4864, ttl=7 (no response found!)
1106 2.533814   100.71.0.1      192.168.0.101   ICMP    70 Time-to-live exceeded (Time to live exceeded in transit)
1107 2.533815   100.71.0.1      192.168.0.101   ICMP    70 Time-to-live exceeded (Time to live exceeded in transit)
1108 2.533815   100.71.0.1      192.168.0.101   ICMP    70 Time-to-live exceeded (Time to live exceeded in transit)
1110 2.534095   192.168.0.101   74.6.143.26     ICMP    74 Echo (ping) request  id=0x0001, seq=20/5120, ttl=7 (no response found!)
1111 2.534136   192.168.0.101   74.6.143.26     ICMP    74 Echo (ping) request  id=0x0001, seq=21/5376, ttl=7 (no response found!)
1112 2.534159   192.168.0.101   74.6.143.26     ICMP    74 Echo (ping) request  id=0x0001, seq=22/5632, ttl=8 (no response found!)
1115 2.539446   203.187.192.33  192.168.0.101   ICMP    70 Time-to-live exceeded (Time to live exceeded in transit)
1116 2.539446   203.187.192.33  192.168.0.101   ICMP    70 Time-to-live exceeded (Time to live exceeded in transit)
1117 2.539446   203.187.192.33  192.168.0.101   ICMP    70 Time-to-live exceeded (Time to live exceeded in transit)
1118 2.539727   192.168.0.101   74.6.143.26     ICMP    74 Echo (ping) request  id=0x0001, seq=23/5888, ttl=8 (no response found!)
1119 2.539772   192.168.0.101   74.6.143.26     ICMP    74 Echo (ping) request  id=0x0001, seq=24/6144, ttl=8 (no response found!)
1120 2.539796   192.168.0.101   74.6.143.26     ICMP    74 Echo (ping) request  id=0x0001, seq=25/6400, ttl=9 (no response found!)
```

   (b) Consider one specific iteration of traceroute invocation/iteration. For this specific iteration, what are the individual round trip times of each of the three probes sent ? What is the average round trip time ? Does it match with the round trip time returned by traceroute ?

   Round trip time for router: 100.71.0.1
   1st trip: 3.596 ms
   2nd trip: 3.514 ms
   3rd trip: 3.463 ms
   Avg: 3.525 ms

   (c) In each iteration of traceroute does it use the same port number for the destination? IF yes, reason why and if no, then also argue why it does so ?

   Traceroute send ICMP request and ICMP has no concept of ports

10. Download, study and analyse the NSL-KDD dataset (ref: https://www.unb.ca/cic/datasets/nsl.html). Is this a labelled dataset or unlabelled dataset? Give reasons for your answer. How many samples are there in the training and that in the testing dataset? Identify various attributes in this dataset. How many attack types are specified in this dataset? Which ones? Use the figure below to learn the attack types and different categories and identify them in the dataset. The study here prepares you with the groundwork to undertake the first problem in the next assignment.

(a) This is labelled dataset: because Dataset have target field attack-type.
(b) Training Sample: 125973, Testing Sample: 22544
(c) No. attack type in dataset: 23

Attacks: normal, neptune, warezclient, ipsweep, portsweep, teardrop, nmap, satan, smurf, pod, back, guess_passwd, ftp_write, multihop, rootkit, buffer_overflow, imap, warezmaster, phf, land, loadmodule, spy, perl

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df_train = pd.read_csv("./NSL-KDD/KDDTrain+.txt", header=None)
df_test = pd.read_csv("./NSL-KDD/KDDTest+.txt", header=None)
```

In [3]:
```
df_train
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	33	34	35	36	37	38	
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	0.17	0.03	0.17	0.00	0.00	0.00	(
1	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	0.88	0.00	0.00	0.00	(
2	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	0.00	0.00	1.00	1.00	(
3	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	0.03	0.04	0.03	0.01	(
4	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	0.00	0.00	0.00	0.00	(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
125968	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.06	0.00	0.00	1.00	1.00	(

In [4]:
```
df_train.shape
```

Out[4]:
```
(125973, 43)
```

In [5]:
```
df_test.shape
```

Out[5]:
```
(22544, 43)
```

In [6]:
```
columns=['duration','protocol_type','service','flag','src_bytes','dst_bytes','la
            ,'hot','num_failed_logins','logged_in','num_compromised','root_shell
            ,'num_file_creations','num_shells','num_access_files','num_outbound_
            ,'is_guest_login','count','srv_count','serror_rate','srv_serror_rate
            ,'same_srv_rate','diff_srv_rate','srv_diff_host_rate','dst_host_cour
            ,'dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_sr
            ,'dst_host_srv_diff_host_rate','dst_host_serror_rate','dst_host_srv_
            ,'dst_host_srv_rerror_rate','attack','outcome']
```

In [7]:
```
df_train.columns = columns
df_test.columns = columns
```

In [8]:
```
df_train['attack'].unique()
```

Out[8]:
```
array(['normal', 'neptune', 'warezclient', 'ipsweep', 'portsweep',
       'teardrop', 'nmap', 'satan', 'smurf', 'pod', 'back',
       'guess_passwd', 'ftp_write', 'multihop', 'rootkit',
       'buffer_overflow', 'imap', 'warezmaster', 'phf', 'land',
       'loadmodule', 'spy', 'perl'], dtype=object)
```

In [9]:
```
len(df_train['attack'].unique())
```

Out[9]:
```
23
```

In [10]:
```
def class_classify(data):
    output = []
    dos_check = ["back","land","neptune","pod","smurf","teardrop"]
    r2l_check = ["ipsweep","nmap","portsweep","satan"]
    u2r_check = ["buffer_overflow","loadmodule","perl","rootkit"]
    probe_check = ["ftp_write","guess_passwd","imap","multihop","phf","Snmpgetat

    for i in range(len(data)):

        if data.iloc[i]['attack'] in dos_check:
            output.append('Dos')

        elif data.iloc[i]['attack'] in r2l_check:
            output.append('R2L')

        elif data.iloc[i]['attack'] in u2r_check:
            output.append('U2R')

        else:
            output.append('Probe')

    return output
```

In [11]:
```
classs = class_classify(df_train)
df_train['class'] = classs
```

In [12]:
```
classs = class_classify(df_test)
df_test['class'] = classs
```

In [13]:
```
df_train
```

Out[13]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	u
0	0	tcp	ftp_data	SF	491	0	0	0	
1	0	udp	other	SF	146	0	0	0	
2	0	tcp	private	S0	0	0	0	0	
3	0	tcp	http	SF	232	8153	0	0	
4	0	tcp	http	SF	199	420	0	0	
...	...	...	...	...	...	...	...	...	
125968	0	tcp	private	S0	0	0	0	0	
125969	8	udp	private	SF	105	145	0	0	
125970	0	tcp	smtp	SF	2231	384	0	0	
125971	0	tcp	klogin	S0	0	0	0	0	
125972	0	tcp	ftp_data	SF	151	0	0	0	

125973 rows × 44 columns

In [14]:
```
df_test
```

Out[14]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment
0	0	tcp	private	REJ	0	0	0	0
1	0	tcp	private	REJ	0	0	0	0
2	2	tcp	ftp_data	SF	12983	0	0	0
3	0	icmp	eco_i	SF	20	0	0	0
4	1	tcp	telnet	RSTO	0	15	0	0
...	...	...	...	...	...	...	...	...
22539	0	tcp	smtp	SF	794	333	0	0
22540	0	tcp	http	SF	317	938	0	0
22541	0	tcp	http	SF	54540	8314	0	0
22542	0	udp	domain_u	SF	42	42	0	0
22543	0	tcp	sunrpc	REJ	0	0	0	0

22544 rows × 44 columns