Networks Lab Report
Assignment 3

Aim of Experiment : The objective of this assignment is to understand the TCP/IP protocol stack and the headers associated with different layers of the protocol stack by creating a FTP host and two FTP clients and sending an adequately large file from the host to the clients simultaneously and capturing the packets at host and the two clients. The analysis of these packets trace provide us information about various types of packet protocols, layers, header sizes and finally answer the query why we need both mac address and the IP for a network.

Procedure :

Assuming mininet is set up on the system in a virtualbox, first boot up mininet from the oracle vm virtualbox, now type the following and note the ip for eth0.

```
>ifconfig
```

Now download and place a big file to be sent through FTP for packet analysis in the home folder of the mininet after connecting to server and typing the IP of your virtual mininet.

Now, ssh login to your virtual mininet using local terminal (to create xterms for h1,h2 and h3), use ip you got from previous step from mininet terminal. And now for desired topology, we need to create custom topology for which we can use a python script.
In the python script, we create our desired topology using the mininet API and setting the bandwidth, loss and delay for the mentioned links.

Here, we need to first install FTPD in mininet, to do that we need no proxy network as proxy makes it difficult for the mininet CLI to communicate through the internet. So, we first disable the system proxy and connected to a no-proxy network with internet access like 3g mobile networks using a wifi hotspot. Now type the following in the terminal after SSHing to your virtual mininet :

```
>sudo dhclient eth1
>sudo apt-get install ftpd
```

Now, copy the script in the home folder of mininet after ssh login and use the following to create the desired topology using the python script

```
>sudo mn --custom ~/mininet/topo-2sw-3host.py --topo mytopo --link tc
```

We get the following output on the terminal :

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2
*** Adding links:
(1.00Mbit 1ms delay 1% loss) (1.00Mbit 1ms delay 1% loss) (h1, s1) (1.00Mbit 1ms delay 2% loss) (1.00Mbit 1ms delay 2% loss) (s1, h2) (1.00Mbit 1ms delay 2% loss) (1.00Mbit 1ms delay 2% loss) (s1, s2) (1.00Mbit 1ms delay 1% loss) (1.00Mbit 1ms delay 1% loss) (s2, h3)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(1.00Mbit 1ms delay 1% loss) (1.00Mbit 1ms delay 2% loss) (1.00Mbit 1ms delay 2% loss) (1.00Mbit 1ms delay 2% loss) (1.00Mbit 1ms delay 1% loss)
*** Starting CLI:
```

Now, from your terminal use

```
>xterm h1
>xterm h2
>xterm h3
```

Now, in your h1's xterm, type :

```
>sudo wireshark &                    //To run wireshark in the background
```

Now, in your h2's and h3's xterm, type :

```
>mkdir h2
```

>cd h2
>sudo wireshark & //To run wireshark in the background

In your h1's xterm type
>inetd
To start a FTP server at H1

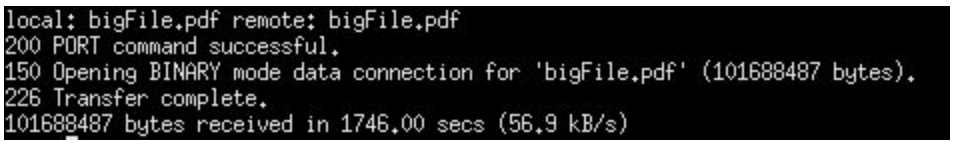
Now, in your h2's and h3's xterm type
>ftp 10.0.0.1
To start the ftp session and then the following commands in order:
>Name: mininet
>Password: mininet
ftp> get bigFile.pdf

Wait for about half an hour and after successful transfer
ftp>bye

The output generated in h2's and h3's xterm after successful transfer is as follows :

H2 :

H3:



Now open your wireshark windows and begin analysing the packet traces of H1, H2 and H3. The sample output in wireshark is of the following form :

1	0.000000000	32:39:93:ca:6d:0b	Broadcast	ARP	42 Who has 10.0.0.1? Tell 10.0.0.3
2	0.001088000	46:1f:ee:b4:96:69	32:39:93:ca:6d:0b	ARP	42 10.0.0.1 is at 46:1f:ee:b4:96:69
3	0.008148000	10.0.0.3	10.0.0.1	TCP	74 56209 > ftp [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=326043 TSec
4	0.009314000	10.0.0.1	10.0.0.3	TCP	74 ftp > 56209 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=
5	0.016331000	10.0.0.3	10.0.0.1	TCP	66 56209 > ftp [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=326047 TSecr=326046
6	0.050027000	10.0.0.1	10.0.0.3	FTP	138 Response: 220 mininet-vm FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.17) read
7	0.057869000	10.0.0.3	10.0.0.1	TCP	66 56209 > ftp [ACK] Seq=1 Ack=73 Win=29696 Len=0 TSval=326058 TSecr=326056
8	2.493551000	10.0.0.3	10.0.0.1	FTP	80 Request: USER mininet
9	2.494658000	10.0.0.1	10.0.0.3	TCP	66 ftp > 56209 [ACK] Seq=73 Ack=15 Win=29184 Len=0 TSval=326668 TSecr=326666
10	2.535624000	10.0.0.1	10.0.0.3	FTP	102 Response: 331 Password required for mininet.
11	2.548364000	10.0.0.3	10.0.0.1	TCP	66 56209 > ftp [ACK] Seq=15 Ack=109 Win=29696 Len=0 TSval=326679 TSecr=326678
12	4.449556000	10.0.0.3	10.0.0.1	FTP	80 Request: PASS mininet
13	4.453241000	10.0.0.1	10.0.0.3	TCP	66 ftp > 56209 [ACK] Seq=109 Ack=29 Win=29184 Len=0 TSval=327157 TSecr=327154
14	4.666650000	10.0.0.1	10.0.0.3	FTP	95 Response: 230 User mininet logged in.
15	4.674504000	10.0.0.3	10.0.0.1	TCP	66 56209 > ftp [ACK] Seq=29 Ack=138 Win=29696 Len=0 TSval=327212 TSecr=327211
16	4.675022000	10.0.0.3	10.0.0.1	FTP	72 Request: SYST
17	4.676207000	10.0.0.1	10.0.0.3	FTP	93 Response: 215 UNIX Type: L8 (Linux)
18	4.721973000	10.0.0.3	10.0.0.1	TCP	66 56209 > ftp [ACK] Seq=35 Ack=165 Win=29696 Len=0 TSval=327224 TSecr=327213

Observation :

1.The protocol hierarchy of H1 is as follows :

Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
▾ Frame	100.00 %	186824	100.00 %	221188581	0.661	0	0	0.000
▽ Ethernet	100.00 %	186824	100.00 %	221188581	0.661	0	0	0.000
Address Resolution Protocol	0.12 %	230	0.00 %	9660	0.000	230	9660	0.000
▽ Internet Protocol Version 4	99.88 %	186594	100.00 %	221178921	0.661	0	0	0.000
▽ Transmission Control Protocol	99.88 %	186594	100.00 %	221178921	0.661	101494	7048364	0.021
File Transfer Protocol (FTP)	0.02 %	31	0.00 %	2973	0.000	31	2973	0.000
FTP Data	45.53 %	85069	96.81 %	214127584	0.640	85069	214127584	0.640

DIfferent Protocols : (Lower to higher levels in protocol hierarchy)

1) Application layer: FTP/FTP-data (File Transfer Protocol)
2) Transport layer : TCP (Transmission Control Protocol)
3) Network layer : IPv4 (Internet Protocol version 4)
4) Data Link Layer : ARP (Address Resolution Protocol)

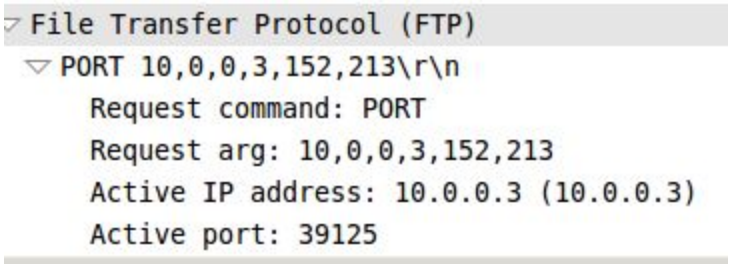
2.

- Application Layer :

  For FTP

  Header size - variable

For fields in FTP Header the screenshot of a particular packet captured at h1 in wireshark is attached.



For FTP-data

Header size and fields are not mentioned

- Transport Layer

For TCP

Header Size - 32 bytes ,

However for 1st TCP packet (with seq number=1) header length=40

For Fields present in TCP header

Following is a screenshot for a particular packet captured at h1 using wireshark. It shows all the fields present inside the TCP header along with the details.

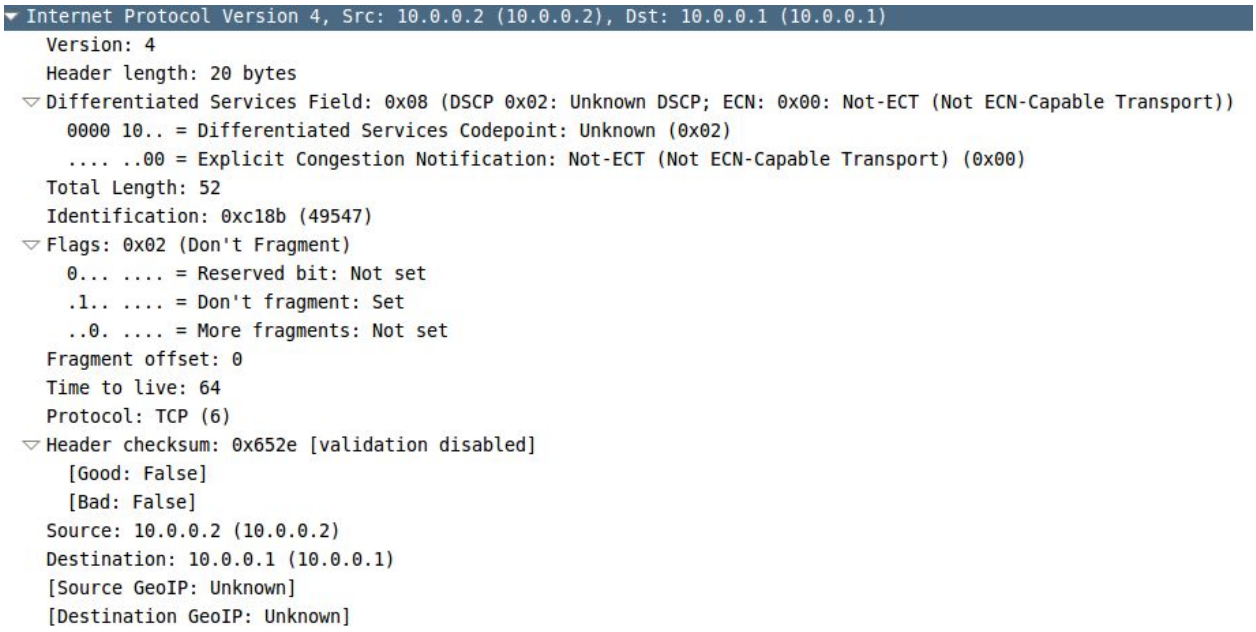
```
  Source port: ftp-data (20)
  Destination port: 39125 (39125)
  [Stream index: 2]
  Sequence number: 101688489    (relative sequence number)
  Acknowledgment number: 2    (relative ack number)
  Header length: 32 bytes
▽ Flags: 0x010 (ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Nonce: Not set
    .... 0... .... = Congestion Window Reduced (CWR): Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
  Window size value: 58
  [Calculated window size: 29696]
  [Window size scaling factor: 512]
▷ Checksum: 0x142a [validation disabled]
▽ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▽ No-Operation (NOP)
    ▽ Type: 1
        0... .... = Copy on fragmentation: No
        .00. .... = Class: Control (0)
        ...0 0001 = Number: No-Operation (NOP) (1)
  ▽ No-Operation (NOP)
    ▽ Type: 1
        0... .... = Copy on fragmentation: No
        .00. .... = Class: Control (0)
        ...0 0001 = Number: No-Operation (NOP) (1)
  ▽ Timestamps: TSval 531741, TSecr 531740
      Kind: Timestamp (8)
      Length: 10
      Timestamp value: 531741
      Timestamp echo reply: 531740
▷ [SEQ/ACK analysis]
```

- Network Layer

  For IPv4

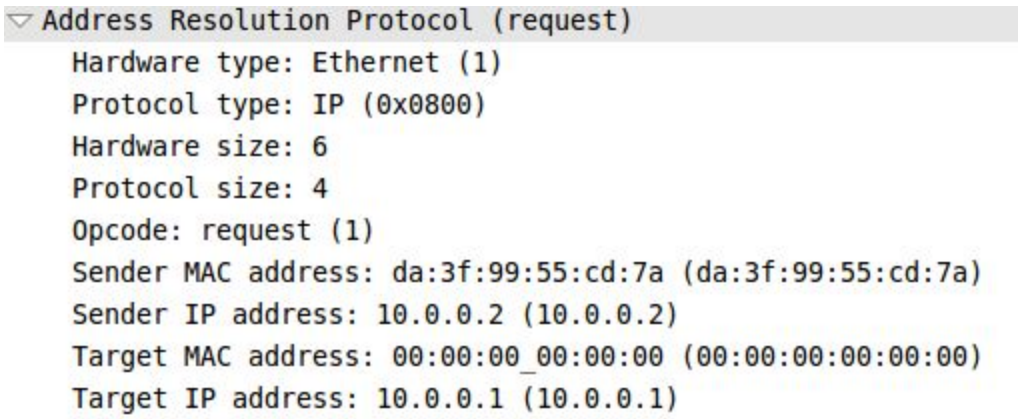
  Header size - 20 bytes

  For different fields present inside the header the screenshot for a particular packet captured at h1 using wireshark is attached.



- Data Link Layer

  The screenshot for ARP protocol with size and various fields is attached.



[Source IP, Destination IP,Source port and Destination port] for different end to end flows at the transport layer :

For H1:
['10.0.0.1', '10.0.0.3', 'ftp', '56209']
['10.0.0.1', '10.0.0.3', 'ftp-data', '57891']
['10.0.0.1', '10.0.0.2', 'ftp', '49980']
['10.0.0.1', '10.0.0.2', 'ftp-data', '40147']

For H2:
['10.0.0.2', '10.0.0.1', '49980', 'ftp']
['10.0.0.2', '10.0.0.1', '40147', 'ftp-data']
['10.0.0.1', '10.0.0.3', 'ftp', '56209']

For H3 :
['10.0.0.3', '10.0.0.1', '56209', 'ftp']
['10.0.0.3', '10.0.0.1', '57891', 'ftp-data']
['10.0.0.1', '10.0.0.2', 'ftp', '49980']

Port of ftp=21
Port of ftp-data=20

So, in total there are 8 different 4-tuples ([Source IP, Destination IP,Source port and Destination port]):
['10.0.0.1', '10.0.0.3', 'ftp', '56209']
['10.0.0.1', '10.0.0.2', 'ftp', '49980']
['10.0.0.1', '10.0.0.3', 'ftp-data', '57891']
['10.0.0.1', '10.0.0.2', 'ftp-data', '40147']
['10.0.0.2', '10.0.0.1', '49980', 'ftp']
['10.0.0.2', '10.0.0.1', '40147', 'ftp-data']
['10.0.0.3', '10.0.0.1', '56209', 'ftp']
['10.0.0.3', '10.0.0.1', '57891', 'ftp-data']

3.
Between H1 and H3 : (flow is bidirectional)
['10.0.0.1', '10.0.0.3', 'ftp', '56209']
['10.0.0.1', '10.0.0.3', 'ftp-data', '57891']
['10.0.0.3', '10.0.0.1', '56209', 'ftp']
['10.0.0.3', '10.0.0.1', '57891', 'ftp-data']

Port of ftp=21
Port of ftp-data=20

These are the unique tuples for <destination port,source port,source_ip,destination_ip> in same order.
Clearly the flow is bidirectional.

4.
For packets generated at h3:

Source MAC Address : 12:3a:bf:23:1e:d0
Destination MAC address : fe:2a:0a:3e:7a:0f

Mac address of h1: fe:2a:0a:3e:7a:0f

Clearly destination mac address of h3 is the mac address of h1

5.
For packets generated at h3:

Source IP : 10.0.0.3
Destination IP : 10.0.0.1

Also IP of h1 = 10.0.0.1
Clearly destination IP of h3 is the IP of h1

6.
IP address and MAC address identify the machine in different layers, IP address is used to identify the machine in network layer while the MAC address is used to identify the machine in Transport and underlying layers. We need MAC address to uniquely identify the machine in lower network hierarchy levels, since IP address directs to the location in network layer after which MAC address identifies the machine, it is also used to communicate in between machines in the same network layer.

MAC address is fixed by the manufacturer and hence is effectively random much like our name, which can be made unique by adding our date of birth, father's name ,mother's name to our "MAC address". But, IP address specifies our location in a network and is temporary while the MAC address is permanent. Think of IP address as your current residential address which is unique and uniquely identifies your location in the world.
Suppose, there was no IP address, how would the postman find you using just your name. However, the idea is not completely unheard of, only MAC address networks can be implemented at lower levels, we need IP addresses for larger networks like the internet to make it more effective and fast.

In our experiment also, first an ARP broadcast to the desired MAC address is made which then provides the IP address for further communication making it easier and faster.

Conclusion :

In this assignment we got to learn about the protocol hierarchy and we also learned to identify the network layers of different types of protocols observed in the packet trace of the FTP file transfer.
We also learned about the header fields, and the number of bit used for each type of protocol headers. We then tested the bidirectional network flow by analysing the packet trace of H1 and H3 where we observed that the packet headers from H1 to H3 was same as that from H3 to H1.
We then identified the source IP and MAC address of the host and matched it with that of the client, we then proceeded to provide an intuitive justification as to why two addresses where required when clearly for such a small network, only MAC address or for that matter, the IP address was sufficient

So, it gave a clear understanding of the protocol stack hierarchy and the headers associated with different protocols also we saw that besides MAC address, IP is also important for communication in the network and explored various explanations as to why both the addresses were necessary.

Authored by **:** **14CS10006 (Apurv Kumar)**
14CS30034 (Shubham Sharma)