Network Lab Report

Shuvayan Ghosh Dastidar
001810501044
JU-BCSE - III

Assignment 5:
Packet tracer and traffic analysis with Wireshark.

1.Generate some ICMP traffic by using the Ping command line tool to check the connectivity of a
neighbouring machine (or router). Note the results in Wireshark. The initial ARP request broadcast from your PC determines the physical MAC address of the network IP Address, and the ARP reply from the neighbouring system. After the ARP request, the pings (ICMP echo request and replies) can be seen.

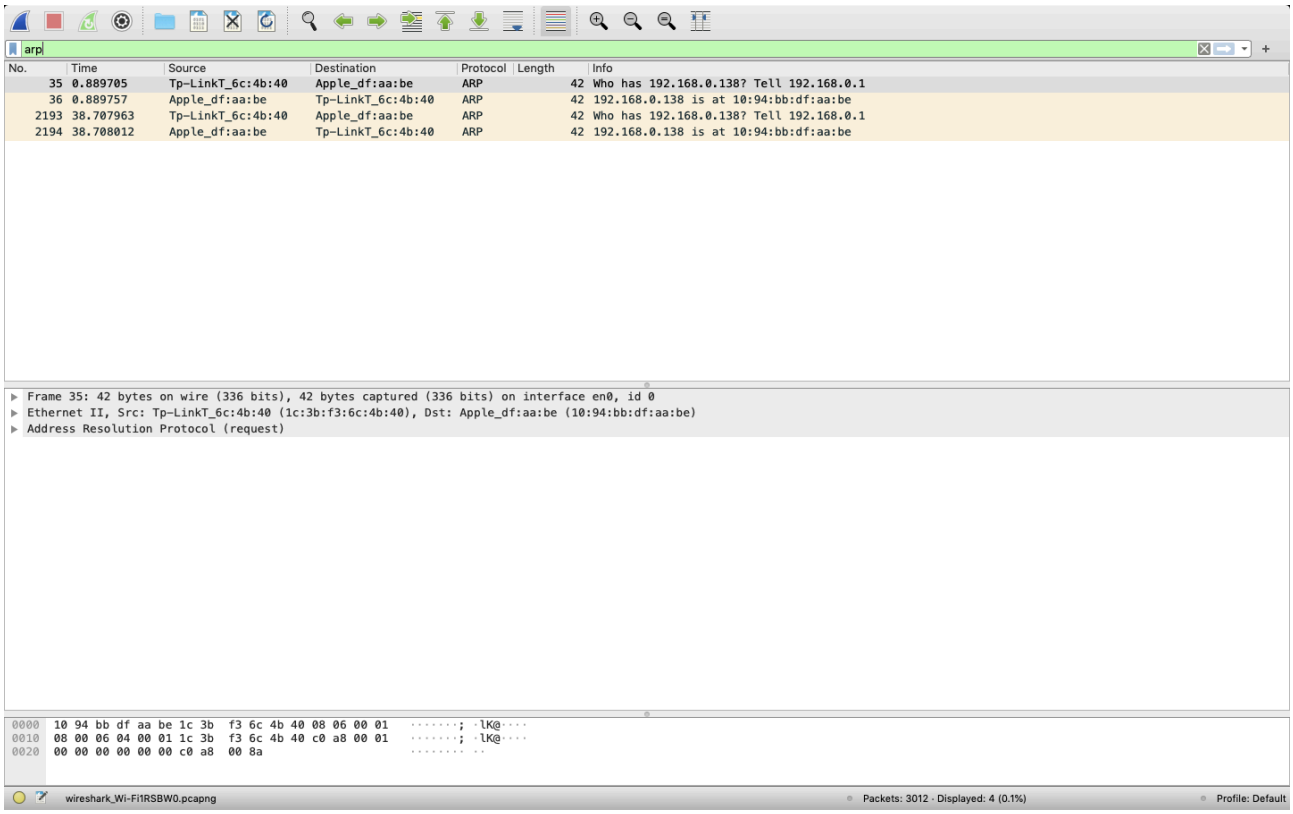
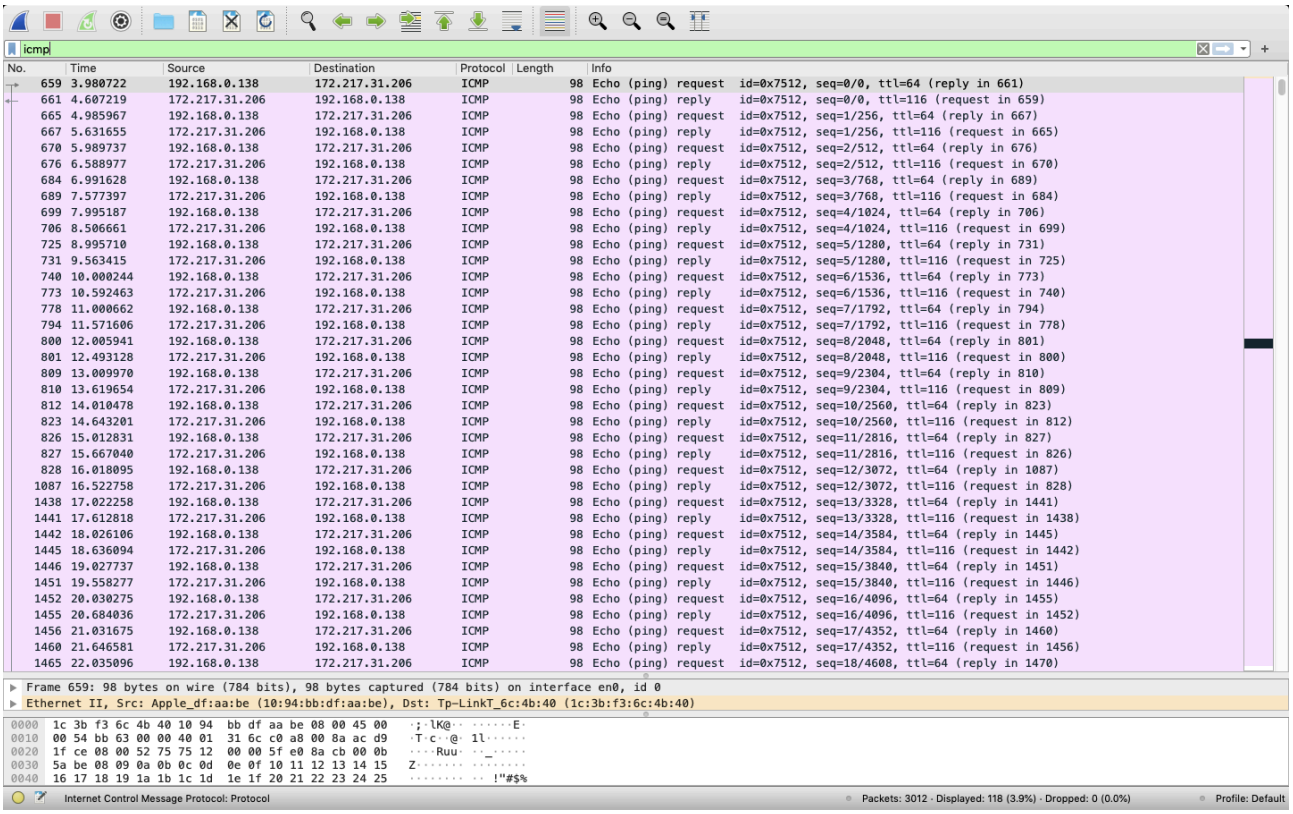


Fig1a: The different ARP requests generated as a part of locating MAC addresses of neighbouring machines.Tp-LinkT_6c is the Wifi-adaptor and the origin machine while Apple_df is the destination (Macbook Air).



icmp

No.	Time	Source	Destination	Protocol	Length	Info
659	3.980722	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=0/0, ttl=64 (reply in 661)
661	4.607219	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=0/0, ttl=116 (request in 659)
665	4.985967	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=1/256, ttl=64 (reply in 667)
667	5.631655	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=1/256, ttl=116 (request in 665)
670	5.989737	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=2/512, ttl=64 (reply in 676)
676	6.588977	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=2/512, ttl=116 (request in 670)
684	6.991628	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=3/768, ttl=64 (reply in 689)
689	7.577397	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=3/768, ttl=116 (request in 684)
699	7.995187	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=4/1024, ttl=64 (reply in 706)
706	8.506661	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=4/1024, ttl=116 (request in 699)
725	8.995710	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=5/1280, ttl=64 (reply in 731)
731	9.563415	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=5/1280, ttl=116 (request in 725)
740	10.000244	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=6/1536, ttl=64 (reply in 773)
773	10.592463	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=6/1536, ttl=116 (request in 740)
778	11.000662	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=7/1792, ttl=64 (reply in 794)
794	11.571606	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=7/1792, ttl=116 (request in 778)
800	12.005941	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=8/2048, ttl=64 (reply in 801)
801	12.493128	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=8/2048, ttl=116 (request in 800)
809	13.009970	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=9/2304, ttl=64 (reply in 810)
810	13.619654	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=9/2304, ttl=116 (request in 809)
812	14.010478	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=10/2560, ttl=64 (reply in 823)
823	14.643201	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=10/2560, ttl=116 (request in 812)
826	15.012831	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=11/2816, ttl=64 (reply in 827)
827	15.667040	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=11/2816, ttl=116 (request in 826)
828	16.018095	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=12/3072, ttl=64 (reply in 1087)
1087	16.522758	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=12/3072, ttl=116 (request in 828)
1438	17.022258	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=13/3328, ttl=64 (reply in 1441)
1441	17.612818	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=13/3328, ttl=116 (request in 1438)
1442	18.026106	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=14/3584, ttl=64 (reply in 1445)
1445	18.636094	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=14/3584, ttl=116 (request in 1442)
1446	19.027737	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=15/3840, ttl=64 (reply in 1451)
1451	19.558277	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=15/3840, ttl=116 (request in 1446)
1452	20.030275	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=16/4096, ttl=64 (reply in 1455)
1455	20.684036	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=16/4096, ttl=116 (request in 1452)
1456	21.031675	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=17/4352, ttl=64 (reply in 1460)
1460	21.646581	172.217.31.206	192.168.0.138	ICMP	98	Echo (ping) reply id=0x7512, seq=17/4352, ttl=116 (request in 1456)
1465	22.035096	192.168.0.138	172.217.31.206	ICMP	98	Echo (ping) request id=0x7512, seq=18/4608, ttl=64 (reply in 1470)

```
▶ Frame 659: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface en0, id 0
▶ Ethernet II, Src: Apple_df:aa:be (10:94:bb:df:aa:be), Dst: Tp-LinkT_6c:4b:40 (1c:3b:f3:6c:4b:40)

0000  1c 3b f3 6c 4b 40 10 94  bb df aa be 08 00 45 00   ·;·lK@·· ····· ··E·
0010  00 54 bb 63 00 00 40 01  31 6c c0 a8 00 8a ac d9   ·T·c··@· 1l····· ····
0020  1f ce 08 00 52 75 75 12  00 00 5f e0 8a cb 00 0b   ····Ruu· ··_····· ····
0030  5a be 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   Z····· ····· ····· ····
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ····· ····· ··· !"#$%
```

Internet Control Message Protocol: Protocol Packets: 3012 · Displayed: 118 (3.9%) · Dropped: 0 (0.0%) Profile: Default

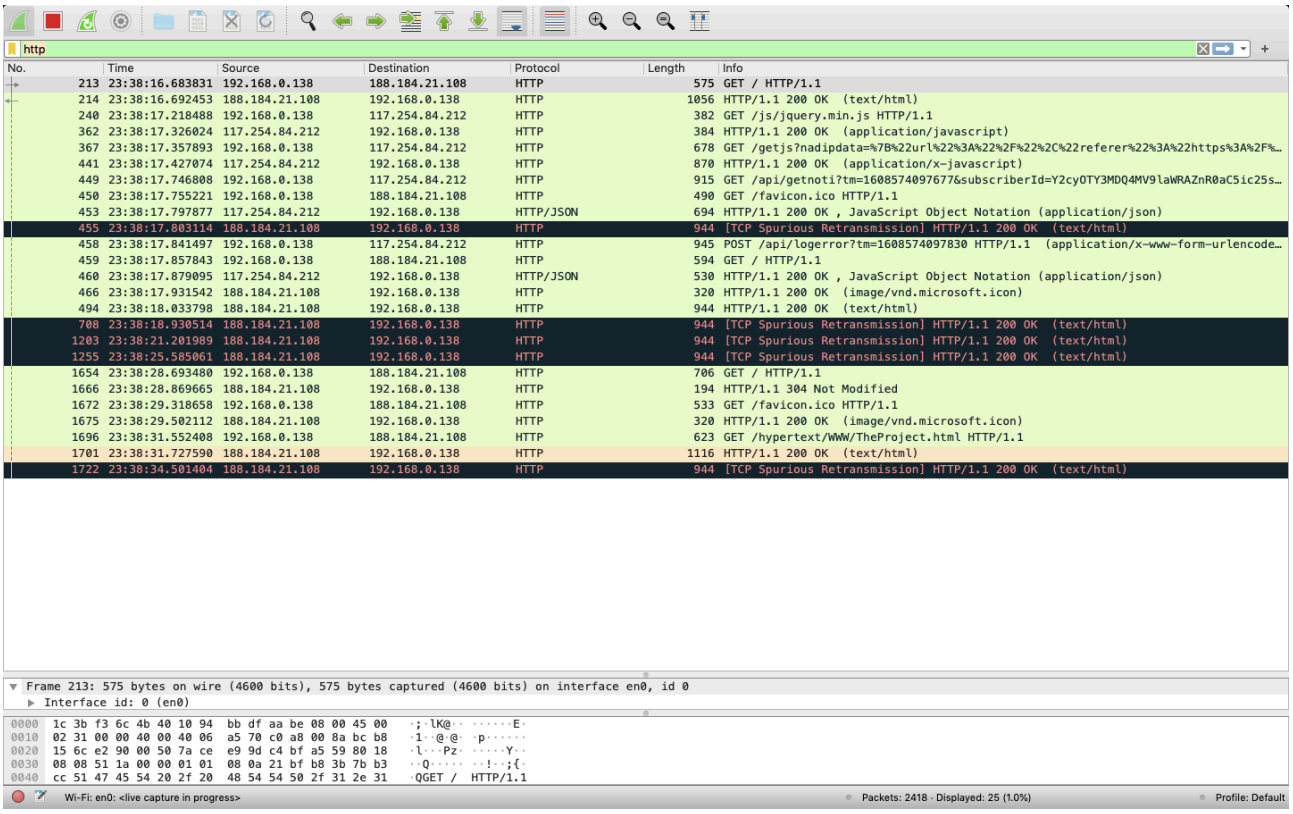
Fig1b: The different ICMP request-replies generated as a part of pinging google.com

2. Generate some web traffic and

A. find the list the different protocols that appear in the protocol column in the unfiltered packet-listing window of Wireshark.

1. ARP
2. DCP-AF
3. DNS
4. DHCP
5. ICMP
6. IGMPv2
7. MDNS
8. TCP
9. UDP
10. TLSv1.3
11. TLSv1.2

B. 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.



No.	Time	Source	Destination	Protocol	Length	Info
213	23:38:16.683831	192.168.0.138	188.184.21.108	HTTP	575	GET / HTTP/1.1
214	23:38:16.692453	188.184.21.108	192.168.0.138	HTTP	1056	HTTP/1.1 200 OK (text/html)
240	23:38:17.218488	192.168.0.138	117.254.84.212	HTTP	382	GET /js/jquery.min.js HTTP/1.1
362	23:38:17.326024	117.254.84.212	192.168.0.138	HTTP	384	HTTP/1.1 200 OK (application/javascript)
367	23:38:17.357893	192.168.0.138	117.254.84.212	HTTP	678	GET /getjs?nadipdata=%7B%22url%22%3A%22%2F%22%2C%22referer%22%3A%22https%3A%2F%…
441	23:38:17.427074	117.254.84.212	192.168.0.138	HTTP	870	HTTP/1.1 200 OK (application/x-javascript)
449	23:38:17.746808	192.168.0.138	117.254.84.212	HTTP	915	GET /api/getnoti?tm=1608574097677&subscriberId=Y2cyOTY3MDQ4MV9laWRAZnR0aC5ic25s…
450	23:38:17.755221	192.168.0.138	188.184.21.108	HTTP	490	GET /favicon.ico HTTP/1.1
453	23:38:17.797877	117.254.84.212	192.168.0.138	HTTP/JSON	694	HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
455	23:38:17.803114	188.184.21.108	192.168.0.138	HTTP	944	[TCP Spurious Retransmission] HTTP/1.1 200 OK (text/html)
458	23:38:17.841497	192.168.0.138	117.254.84.212	HTTP	945	POST /api/logerror?tm=1608574097830 HTTP/1.1 (application/x-www-form-urlencode…
459	23:38:17.857843	192.168.0.138	188.184.21.108	HTTP	594	GET / HTTP/1.1
460	23:38:17.879095	117.254.84.212	192.168.0.138	HTTP/JSON	530	HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
466	23:38:17.931542	188.184.21.108	192.168.0.138	HTTP	320	HTTP/1.1 200 OK (image/vnd.microsoft.icon)
494	23:38:18.033798	188.184.21.108	192.168.0.138	HTTP	944	HTTP/1.1 200 OK (text/html)
708	23:38:18.930514	188.184.21.108	192.168.0.138	HTTP	944	[TCP Spurious Retransmission] HTTP/1.1 200 OK (text/html)
1203	23:38:21.201989	188.184.21.108	192.168.0.138	HTTP	944	[TCP Spurious Retransmission] HTTP/1.1 200 OK (text/html)
1255	23:38:25.585061	188.184.21.108	192.168.0.138	HTTP	944	[TCP Spurious Retransmission] HTTP/1.1 200 OK (text/html)
1654	23:38:28.693480	192.168.0.138	188.184.21.108	HTTP	706	GET / HTTP/1.1
1666	23:38:28.869665	188.184.21.108	192.168.0.138	HTTP	194	HTTP/1.1 304 Not Modified
1672	23:38:29.318658	192.168.0.138	188.184.21.108	HTTP	533	GET /favicon.ico HTTP/1.1
1675	23:38:29.502112	188.184.21.108	192.168.0.138	HTTP	320	HTTP/1.1 200 OK (image/vnd.microsoft.icon)
1696	23:38:31.552408	192.168.0.138	188.184.21.108	HTTP	623	GET /hypertext/WWW/TheProject.html HTTP/1.1
1701	23:38:31.727590	188.184.21.108	192.168.0.138	HTTP	1116	HTTP/1.1 200 OK (text/html)
1722	23:38:34.501404	188.184.21.108	192.168.0.138	HTTP	944	[TCP Spurious Retransmission] HTTP/1.1 200 OK (text/html)

Fig2b : The different HTTP requests as shown in Wireshark.

The first HTTP GET request was sent at 23:38:16.218488 and the HTTP OK response was received at 23:38:17.326034. and thus it took 1.11 seconds.

C. What is the Internet address of the website? What is the Internet address of your computer?

The internet address of the website http://info.cern.ch/(had to look up an ancient website) is 188.184.21.108 and the internet address of my computer is 192.168.0.138.

D. Search back through your capture, and find an HTTP packet containing a GET command. Click on the packet in the Packet List Panel. Then expand the HTTP layer in the Packet Details Panel, from the packet.

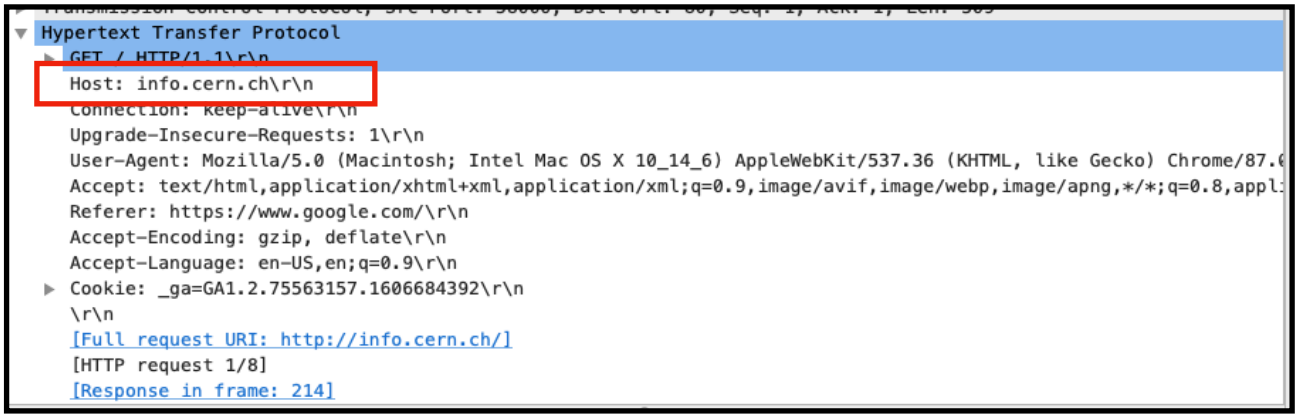
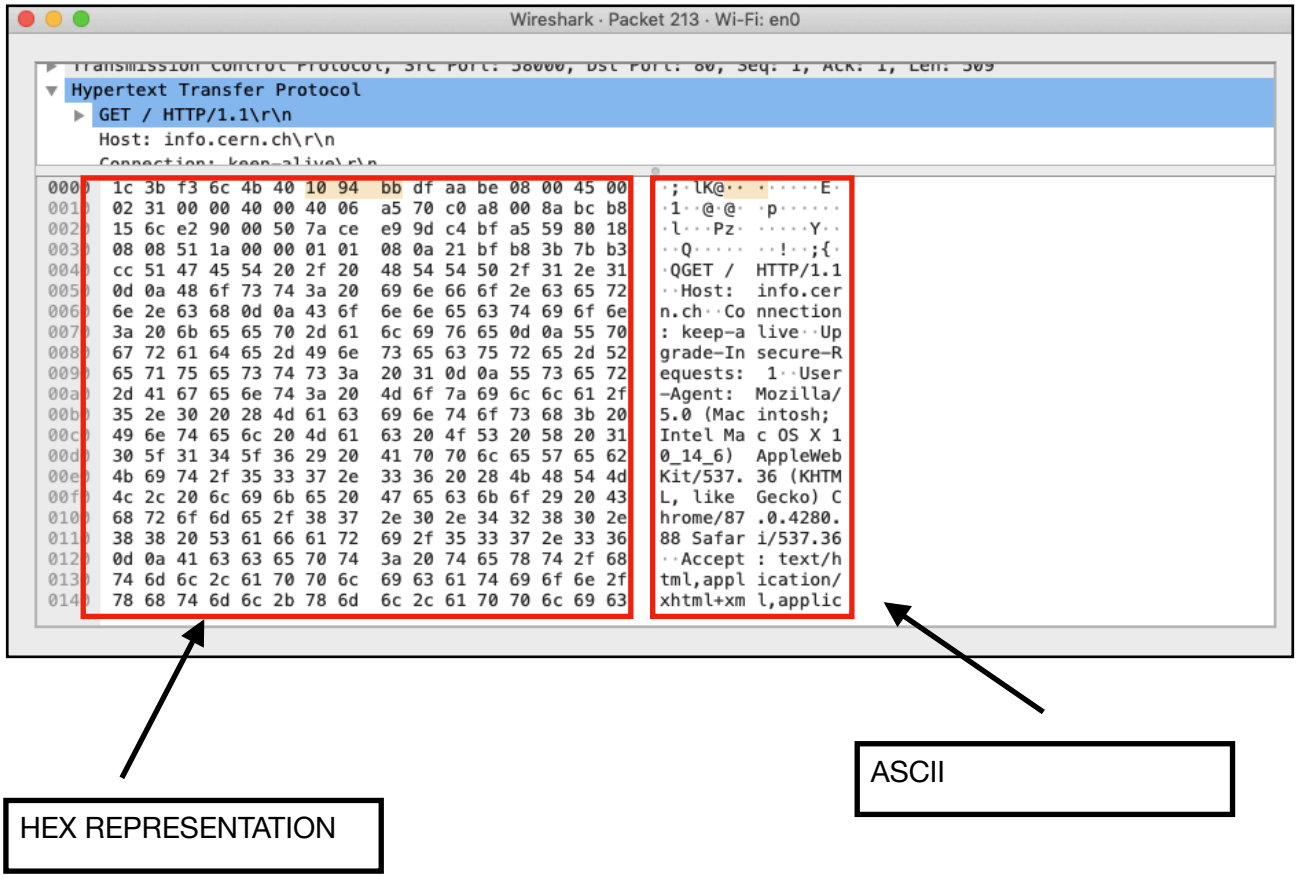


Fig.2d

E. Find out the value of the Host from the Packet Details Panel, within the GET command.

The figure 2d shows the hostname.- **http://info.cern.ch/**

3. Highlight the Hex and ascii representations of the packet in the Packets Byte Panel.



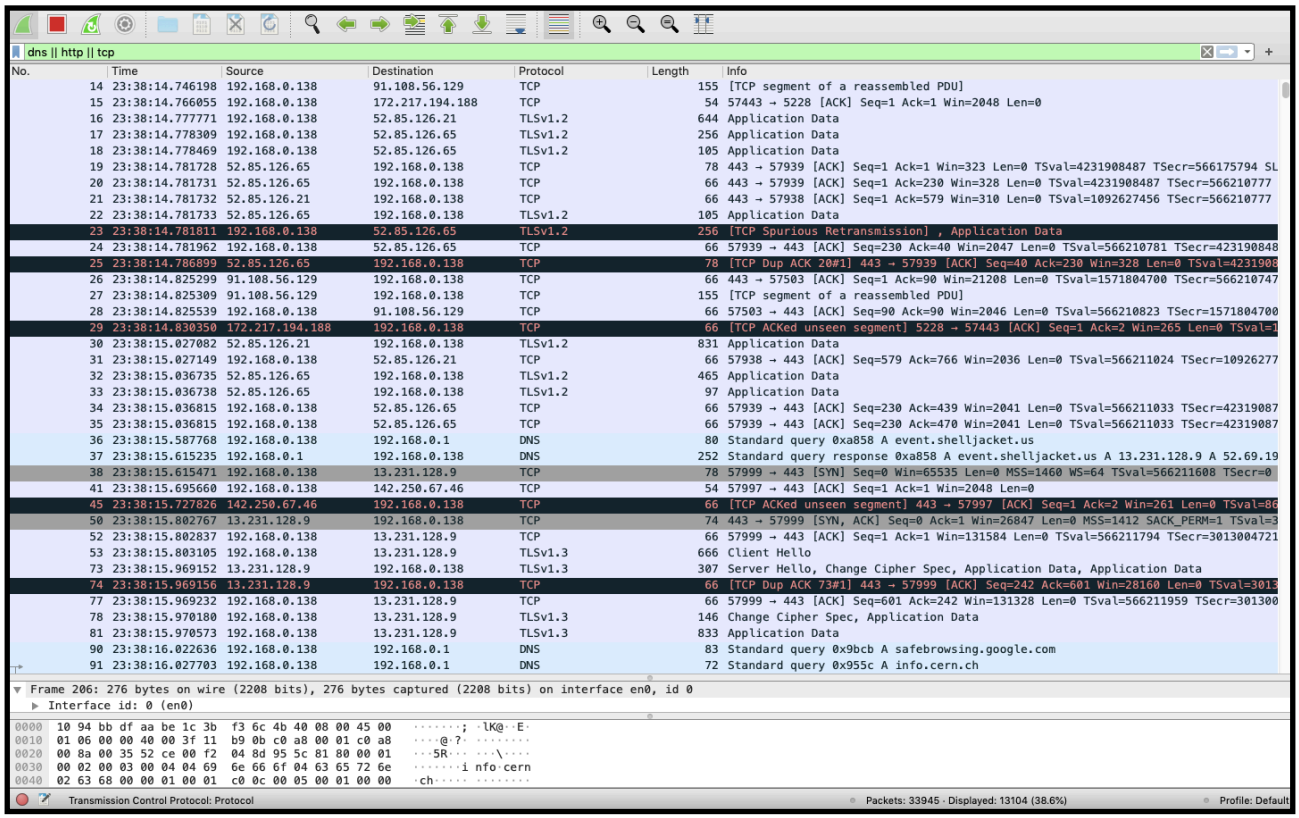
4. Find the first 4 bytes of the host parameter from the Packets Byte Panel.

The first four hex bytes are highlighted below.

```
1c 3b f3 6c 4b 40 10 94  bb df aa be 08 00 45 00   ·;·lK@·· ·····E·
02 31 00 00 40 00 40 06  a5 70 c0 a8 00 8a bc b8   ·1··@·@· ·p·····
15 6c e2 90 00 50 7a ce  e9 9d c4 bf a5 59 80 18   ·l···Pz· ·····Y··
08 08 51 1a 00 00 01 01  08 0a 21 bf b8 3b 7b b3   ··Q····· ··!··;{·
cc 51 47 45 54 20 2f 20  48 54 54 50 2f 31 2e 31   ·QGET /  HTTP/1.1
0d 0a 48 6f 73 74 3a 20  69 6e 66 6f 2e 63 65 72   ··Host:  info.cer
6e 2e 63 68 0d 0a 43 6f  6e 6e 65 63 74 69 6f 6e   n.ch··Co nnection
3a 20 6b 65 65 70 2d 61  6c 69 76 65 0d 0a 55 70   : keep-a live··Up
67 72 61 64 65 2d 49 6e  73 65 63 75 72 65 2d 52   grade-In secure-R
65 71 75 65 73 74 73 3a  20 31 0d 0a 55 73 65 72   equests:  1··User
2d 41 67 65 6e 74 3a 20  4d 6f 7a 69 6c 6c 61 2f   -Agent:  Mozilla/
35 2e 30 20 28 4d 61 63  69 6e 74 6f 73 68 3b 20   5.0 (Mac intosh;
49 6e 74 65 6c 20 4d 61  63 20 4f 53 20 58 20 31   Intel Ma c OS X 1
```

5.Filter packets with http, TCP, DNS and other protocols.

a. Find out what are those packets contain by following one of the conversations (also called network flows), select one of the packets and press the right mouse button..click on follow.



dns || http || tcp

No.	Time	Source	Destination	Protocol	Length	Info
14	23:38:14.746198	192.168.0.138	91.108.56.129	TCP	155	[TCP segment of a reassembled PDU]
15	23:38:14.766055	192.168.0.138	172.217.194.188	TCP	54	57443 → 5228 [ACK] Seq=1 Ack=1 Win=2048 Len=0
16	23:38:14.777771	192.168.0.138	52.85.126.21	TLSv1.2	644	Application Data
17	23:38:14.778309	192.168.0.138	52.85.126.65	TLSv1.2	256	Application Data
18	23:38:14.778469	192.168.0.138	52.85.126.65	TLSv1.2	105	Application Data
19	23:38:14.781728	52.85.126.65	192.168.0.138	TCP	78	443 → 57939 [ACK] Seq=1 Ack=1 Win=323 Len=0 TSval=4231908487 TSecr=566175794 SL
20	23:38:14.781731	52.85.126.65	192.168.0.138	TCP	66	443 → 57939 [ACK] Seq=1 Ack=230 Win=328 Len=0 TSval=4231908487 TSecr=566210777
21	23:38:14.781732	52.85.126.21	192.168.0.138	TCP	66	443 → 57938 [ACK] Seq=1 Ack=579 Win=310 Len=0 TSval=1092627456 TSecr=566210777
22	23:38:14.781733	52.85.126.65	192.168.0.138	TLSv1.2	105	Application Data
23	23:38:14.781811	192.168.0.138	52.85.126.65	TLSv1.2	256	[TCP Spurious Retransmission] , Application Data
24	23:38:14.781962	192.168.0.138	52.85.126.65	TCP	66	57939 → 443 [ACK] Seq=230 Ack=40 Win=2047 Len=0 TSval=566210781 TSecr=423190848
25	23:38:14.786899	52.85.126.65	192.168.0.138	TCP	78	[TCP Dup ACK 20#1] 443 → 57939 [ACK] Seq=40 Ack=230 Win=328 Len=0 TSval=423190
26	23:38:14.825299	91.108.56.129	192.168.0.138	TCP	66	443 → 57503 [ACK] Seq=1 Ack=90 Win=21208 Len=0 TSval=1571804700 TSecr=566210747
27	23:38:14.825309	91.108.56.129	192.168.0.138	TCP	155	[TCP segment of a reassembled PDU]
28	23:38:14.825539	192.168.0.138	91.108.56.129	TCP	66	57503 → 443 [ACK] Seq=90 Ack=90 Win=2046 Len=0 TSval=566210823 TSecr=1571804700
29	23:38:14.830350	172.217.194.188	192.168.0.138	TCP	66	[TCP ACKed unseen segment] 5228 → 57443 [ACK] Seq=1 Ack=2 Win=265 Len=0 TSval=1
30	23:38:15.027082	52.85.126.21	192.168.0.138	TLSv1.2	831	Application Data
31	23:38:15.027149	192.168.0.138	52.85.126.21	TCP	66	57938 → 443 [ACK] Seq=579 Ack=766 Win=2036 Len=0 TSval=566211024 TSecr=10926277
32	23:38:15.036735	52.85.126.65	192.168.0.138	TLSv1.2	465	Application Data
33	23:38:15.036738	52.85.126.65	192.168.0.138	TLSv1.2	97	Application Data
34	23:38:15.036815	192.168.0.138	52.85.126.65	TCP	66	57939 → 443 [ACK] Seq=230 Ack=439 Win=2041 Len=0 TSval=566211033 TSecr=42319087
35	23:38:15.036815	192.168.0.138	52.85.126.65	TCP	66	57939 → 443 [ACK] Seq=230 Ack=470 Win=2041 Len=0 TSval=566211033 TSecr=42319087
36	23:38:15.587768	192.168.0.138	192.168.0.1	DNS	80	Standard query 0xa858 A event.shelljacket.us
37	23:38:15.615235	192.168.0.1	192.168.0.138	DNS	252	Standard query response 0xa858 A event.shelljacket.us A 13.231.128.9 A 52.69.19
38	23:38:15.615471	192.168.0.138	13.231.128.9	TCP	78	57999 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=566211608 TSecr=0
41	23:38:15.695660	192.168.0.138	142.250.67.46	TCP	54	57997 → 443 [ACK] Seq=1 Ack=1 Win=2048 Len=0
45	23:38:15.727826	142.250.67.46	192.168.0.138	TCP	66	[TCP ACKed unseen segment] 443 → 57997 [ACK] Seq=1 Ack=2 Win=261 Len=0 TSval=86
50	23:38:15.802767	13.231.128.9	192.168.0.138	TCP	74	443 → 57999 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1412 SACK_PERM=1 TSval=3
52	23:38:15.802837	192.168.0.138	13.231.128.9	TCP	66	57999 → 443 [ACK] Seq=1 Ack=1 Win=131584 Len=0 TSval=566211794 TSecr=3013004721
53	23:38:15.803105	192.168.0.138	13.231.128.9	TLSv1.3	666	Client Hello
73	23:38:15.969152	13.231.128.9	192.168.0.138	TLSv1.3	307	Server Hello, Change Cipher Spec, Application Data, Application Data
74	23:38:15.969156	13.231.128.9	192.168.0.138	TCP	66	[TCP Dup ACK 73#1] 443 → 57999 [ACK] Seq=242 Ack=601 Win=28160 Len=0 TSval=3013
77	23:38:15.969232	192.168.0.138	13.231.128.9	TCP	66	57999 → 443 [ACK] Seq=601 Ack=242 Win=131328 Len=0 TSval=566211959 TSecr=301300
78	23:38:15.970180	192.168.0.138	13.231.128.9	TLSv1.3	146	Change Cipher Spec, Application Data
81	23:38:15.970573	192.168.0.138	13.231.128.9	TLSv1.3	833	Application Data
90	23:38:16.022636	192.168.0.138	192.168.0.1	DNS	83	Standard query 0x9bcb A safebrowsing.google.com
91	23:38:16.027703	192.168.0.138	192.168.0.1	DNS	72	Standard query 0x955c A info.cern.ch

Frame 206: 276 bytes on wire (2208 bits), 276 bytes captured (2208 bits) on interface en0, id 0
Interface id: 0 (en0)

```
0000  10 94 bb df aa be 1c 3b  f3 6c 4b 40 08 00 45 00   ·····; ·lK@··E·
0010  01 06 00 00 40 00 3f 11  b9 0b c0 a8 00 01 c0 a8   ····@·?· ·····
0020  00 8a 00 35 52 ce 00 f2  04 8d 95 5c 81 80 00 01   ···5R··· ···\····
0030  00 02 00 03 00 04 04 69  6e 66 6f 04 63 65 72 6e   ·····i nfo·cern
0040  02 63 68 00 00 01 00 01  c0 0c 00 05 00 01 00 00   ·ch····· ·····
```

Transmission Control Protocol: Protocol Packets: 33945 · Displayed: 13104 (38.6%) Profile: Default

fig 5a. Packets shown according to the applied filter.

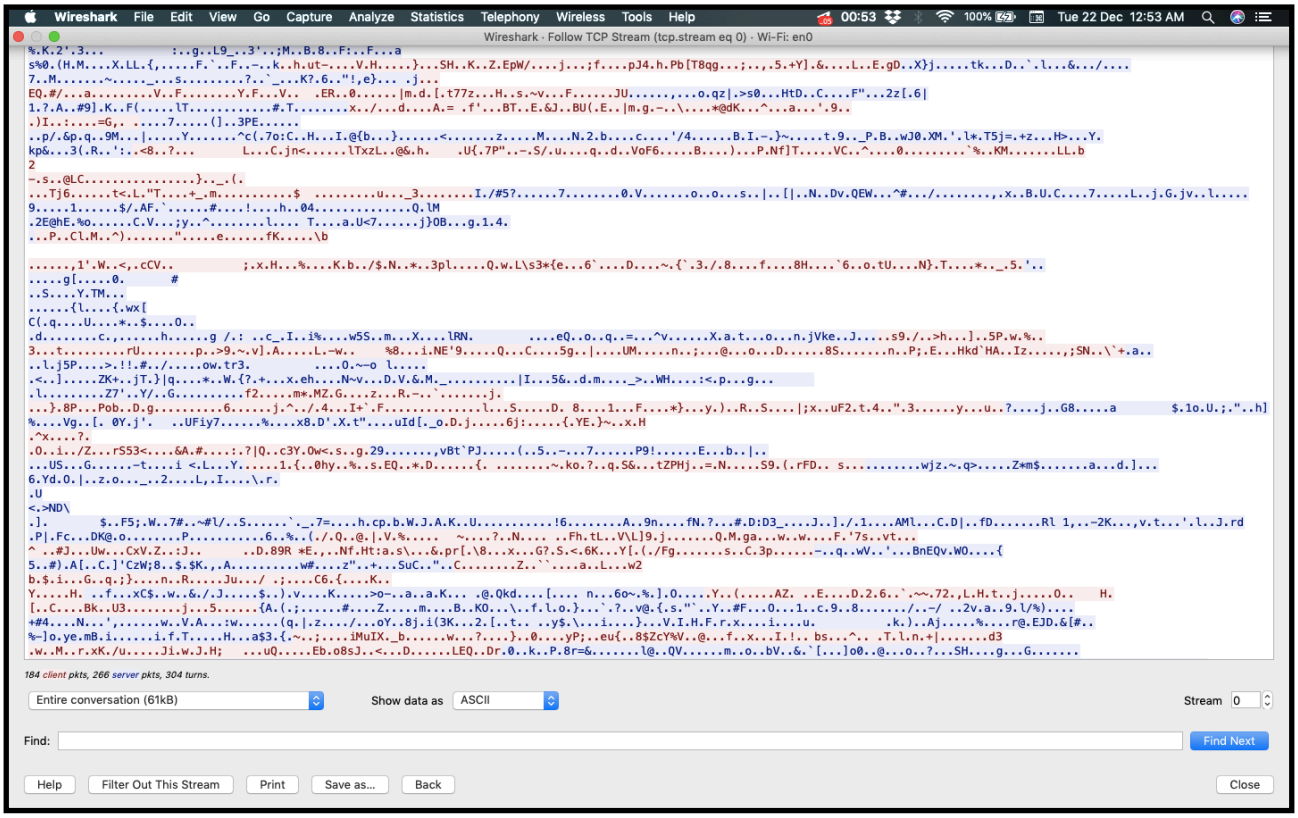
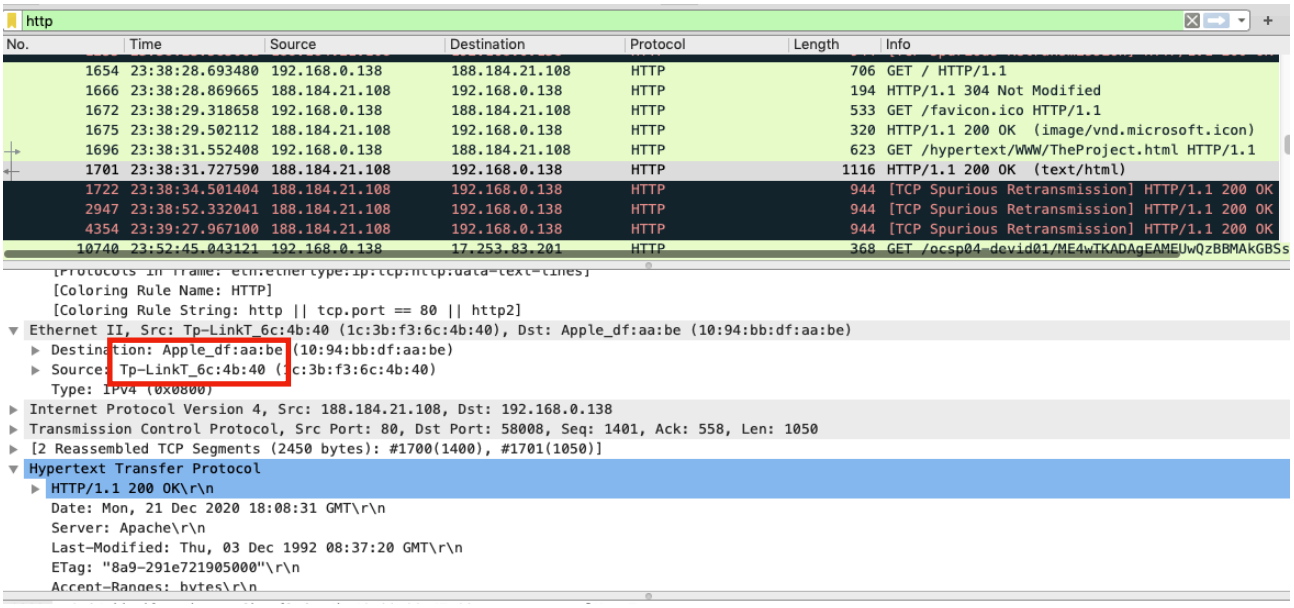


fig. 5a : Shows the stream of a tcp packet.

6. **Search through your capture, and find an HTTP packet coming back from the server (TCP Source Port == 80). Expand the Ethernet layer in the Packet Details Panel.**

fig6. Showing the ethernet details of a HTTP OK request.



7. What are the manufacturers of your PC's Network Interface Card (NIC), and the servers NIC?

PC's NIC : Apple_df

Servers's NIC : Tp-LinkT_6c

8. What are the Hex values (shown the raw bytes panel) of the two NICS Manufacturers OUIs?

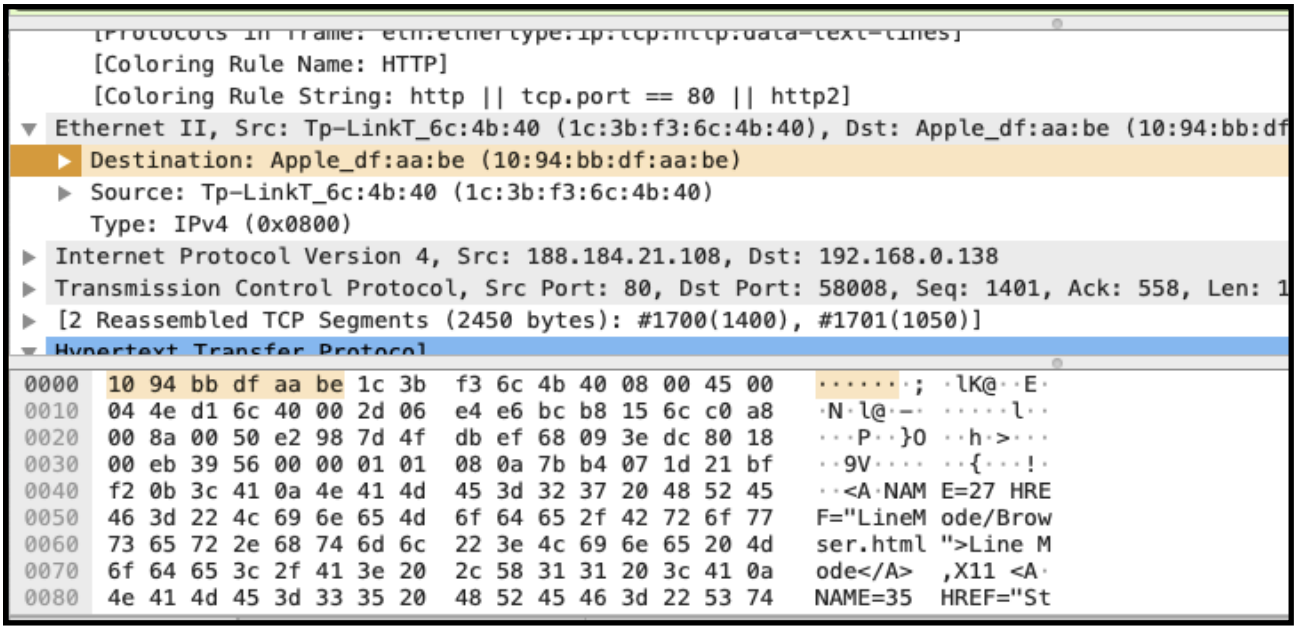


fig8a . Hex value of the source

```
    [Protocols in frame: eth:ethertype:ip:tcp:http:data-text-lines]
    [Coloring Rule Name: HTTP]
    [Coloring Rule String: http || tcp.port == 80 || http2]
▼ Ethernet II, Src: Tp-LinkT_6c:4b:40 (1c:3b:f3:6c:4b:40), Dst: Apple_df:aa:be (10:94:bb
  ▶ Destination: Apple_df:aa:be (10:94:bb:df:aa:be)
  ▶ Source: Tp-LinkT_6c:4b:40 (1c:3b:f3:6c:4b:40)
    Type: IPv4 (0x0800)
▶ Internet Protocol Version 4, Src: 188.184.21.108, Dst: 192.168.0.138
▶ Transmission Control Protocol, Src Port: 80, Dst Port: 58008, Seq: 1401, Ack: 558, Len
▶ [2 Reassembled TCP Segments (2450 bytes): #1700(1400), #1701(1050)]
▼ Hypertext Transfer Protocol

0000  10 94 bb df aa be 1c 3b  f3 6c 4b 40 08 00 45 00   ·····;· ·lK@··E·
0010  04 4e d1 6c 40 00 2d 06  e4 e6 bc b8 15 6c c0 a8   ·N·l@·-· ·····l··
0020  00 8a 00 50 e2 98 7d 4f  db ef 68 09 3e dc 80 18   ···P··}O ··h·>···
0030  00 eb 39 56 00 00 01 01  08 0a 7b b4 07 1d 21 bf   ··9V···· ··{···!·
0040  f2 0b 3c 41 0a 4e 41 4d  45 3d 32 37 20 48 52 45   ··<A·NAM E=27 HRE
0050  46 3d 22 4c 69 6e 65 4d  6f 64 65 2f 42 72 6f 77   F="LineM ode/Brow
0060  73 65 72 2e 68 74 6d 6c  22 3e 4c 69 6e 65 20 4d   ser.html ">Line M
0070  6f 64 65 3c 2f 41 3e 20  2c 58 31 31 20 3c 41 0a   ode</A>  ,X11 <A·
0080  4e 41 4d 45 3d 33 35 20  48 52 45 46 3d 22 53 74   NAME=35  HREF="St
```

fig8b . Hex value of the server.

9. Find the following statistics:

a. What percentage of packets in your capture are TCP, and give an example of the higher level protocol which uses TCP?

b. What percentage of packets in your capture are UDP, and give an example of the higher level protocol which uses UDP?

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	58496	100.0	35328357	54k	0	0	0
Ethernet	100.0	58496	2.3	818944	1265	0	0	0
Logical-Link Control	0.0	2	0.0	16	0	0	0	0
Data	0.0	2	0.0	8	0	2	8	0
Internet Protocol Version 6	2.8	1666	0.2	66640	102	0	0	0
User Datagram Protocol	2.7	1554	0.0	12432	19	0	0	0
Simple Service Discovery Protocol	2.6	1548	1.8	649988	1004	1548	649988	1004
Multicast Domain Name System	0.0	6	0.0	1726	2	6	1726	2
Internet Control Message Protocol v6	0.2	112	0.0	6432	9	112	6432	9
Internet Protocol Version 4	96.6	56532	3.2	1131500	1747	0	0	0
User Datagram Protocol	68.6	40144	0.9	321152	496	0	0	0
Simple Service Discovery Protocol	3.7	2149	2.1	751343	1160	2149	751343	1160
QUIC IETF	59.5	34821	79.6	28120465	43k	34448	27818355	42k
Network Time Protocol	0.1	36	0.0	1728	2	36	1728	2
NetBIOS Name Service	0.6	335	0.1	33228	51	335	33228	51
Multicast Domain Name System	1.2	720	0.1	34468	53	720	34468	53
Dynamic Host Configuration Protocol	0.0	9	0.0	2690	4	9	2690	4
Dropbox LAN sync Discovery Protocol	0.6	344	0.1	45752	70	344	45752	70
Domain Name System	1.0	560	0.1	49616	76	560	49616	76
Data	2.6	1543	0.6	215768	333	1543	215768	333
Transmission Control Protocol	27.4	16036	8.7	3073629	4747	9779	1259179	1945
Transport Layer Security	10.8	6299	7.1	2510670	3878	6190	2125976	3284
Malformed Packet	0.0	11	0.0	0	0	11	0	0
Hypertext Transfer Protocol	0.1	49	0.6	203545	314	19	8087	12
Online Certificate Status Protocol	0.0	6	0.0	15126	23	6	19743	30
Media Type	0.0	3	0.3	98804	152	3	99654	153
Line-based text data	0.0	13	1.1	375417	579	13	70071	108
JavaScript Object Notation	0.0	2	0.0	59	0	2	59	0
HTML Form URL Encoded	0.0	1	0.0	392	0	1	392	0
eXtensible Markup Language	0.0	2	0.0	712	1	2	1531	2
Compuserve GIF	0.0	3	0.0	129	0	3	129	0
Data	0.0	7	0.0	687	1	7	687	1
Internet Group Management Protocol	0.4	215	0.0	1784	2	215	1784	2
Internet Control Message Protocol	0.2	137	0.1	25970	40	3	108	0
NetBIOS Name Service	0.2	134	0.1	21038	32	134	21038	32
Address Resolution Protocol	0.5	296	0.0	8288	12	296	8288	12

No display filter.

Help Copy Close

Fig 9. Statistics of the entire transfer

a. TCP Packet Percentage : 27.4% . A high level protocol that uses TCP is HTTP.

b. UDP Packet Percentage : 68.6%. A high level protocol that uses UDP is DNS.

10.Find the traffic flow Select the Statistics->Flow Graph menu option. Choose General Flow and Network Source options, and click the OK button.

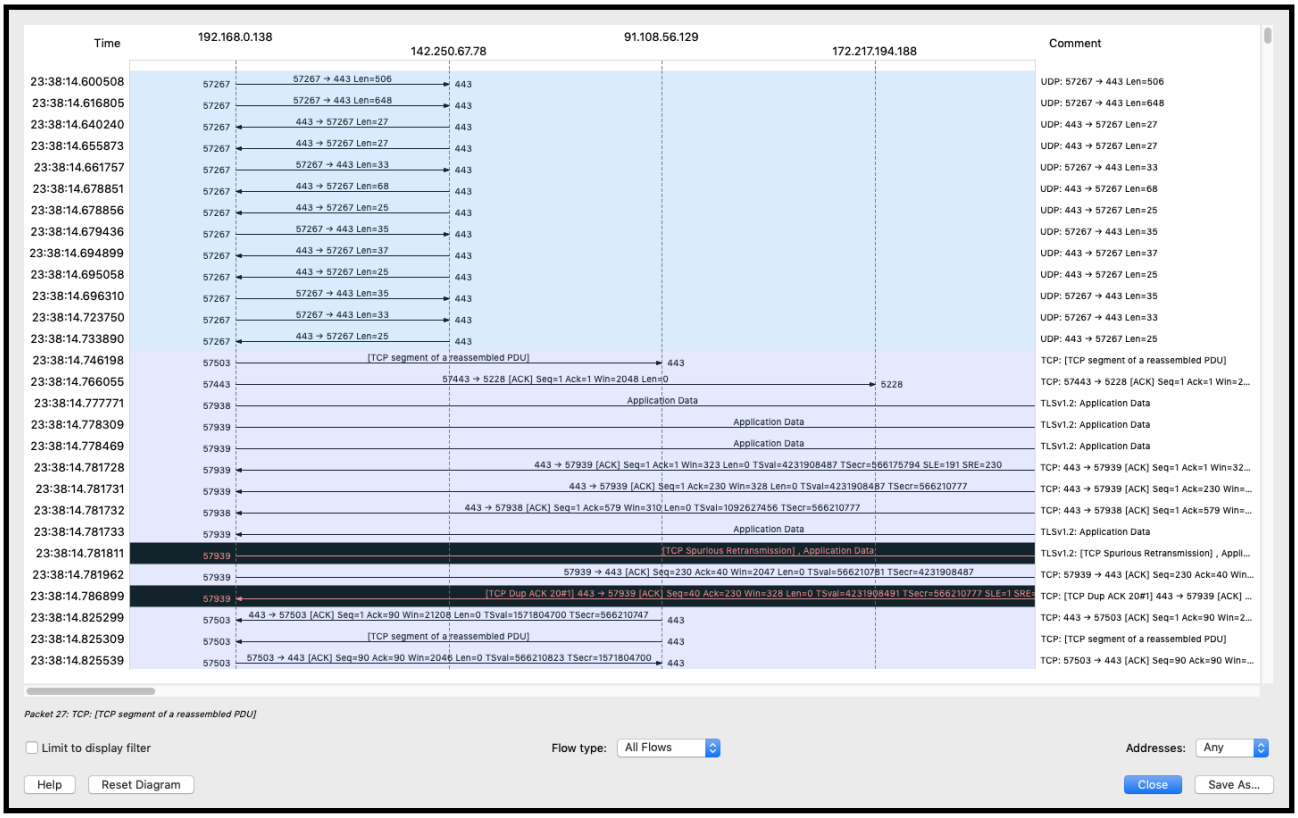


fig 10. Network Flow Graph