CS212 : Computer Networks Lab
Assignment 3

T Satwik
190030043

1 Question 1

The protocols used by the application at different layers are:

- OSCP (Application Layer) (hangouts supports OCSP packets, instead of HTTP, hence in wireshark we get the corresponding OCSP packets)
- DNS (Application layer, uses UDP protocol)
- TCP (Transport Layer)
- TLS (Transport Layer) (TLS is TLSv1.2 and TLSv1.3 in wireshark)
- IP (Network Layer) (used to know the source and destiantion IP addresses)

2 Question 2

The observed values for various fields of the various protocols are:

OSCP: OSCP is like HTTP hence we can get various data like, Request Method, Request URL, Request version, Accept-Language, User-Agent, etc. These values are in the screenshot in Figure 1

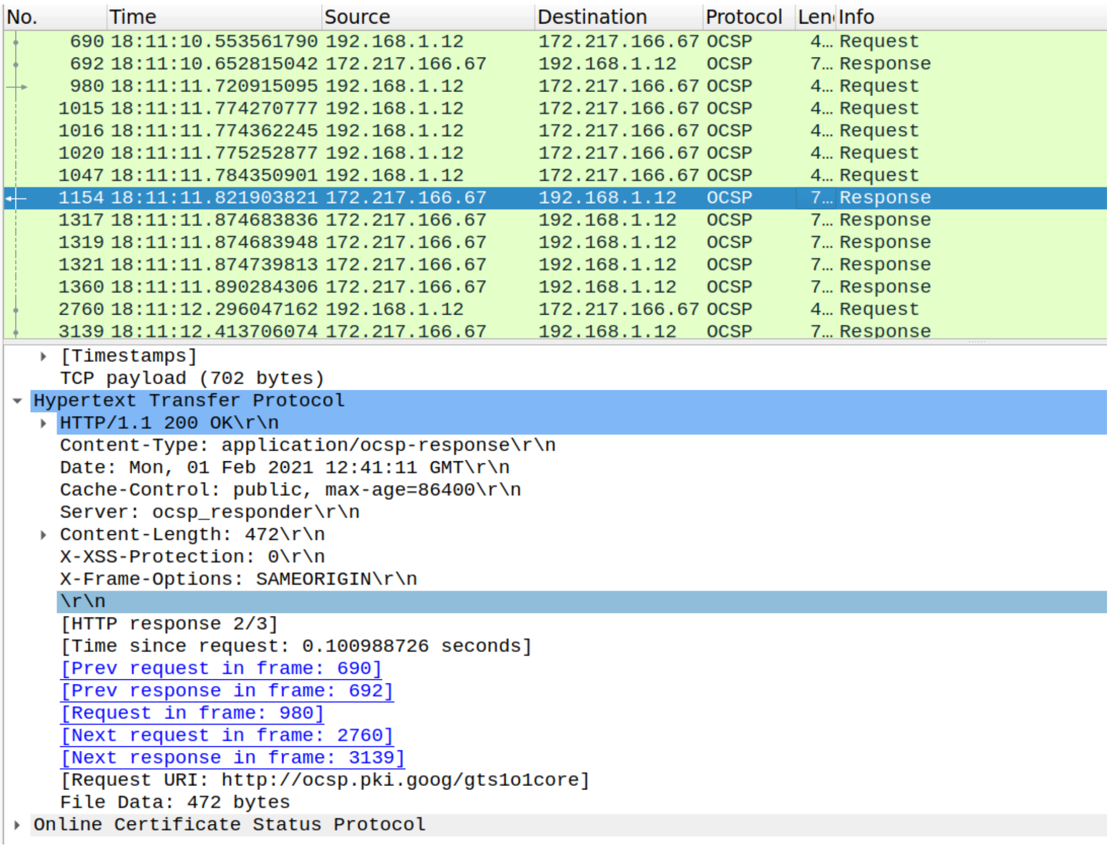
DNS: DNS gives the IP address corresponding the given URL. It used the UDP protocol. We can see the values in the Figure 2

TCP: TCP is a handshaking protocol, and establishes a 3-step-protocols. In the wireshark interface, we can see the various details like the source and destination ports, sequence numbers, acknowledgement numbers, windows size sum, checksum values, etc. We can see all these values in Figure 3

TLS: In addition to the TCP data as above, TLS (TLSv1.2 and TLSv 1.3) would give a transport layer security section which contains

- client hello,
- server hello change cipher spec and encrypted handshake message.
- change cipher spec and encrypted handshake message.

These add the additional security to the packets. We can see these packets and the values, in the Figure 4



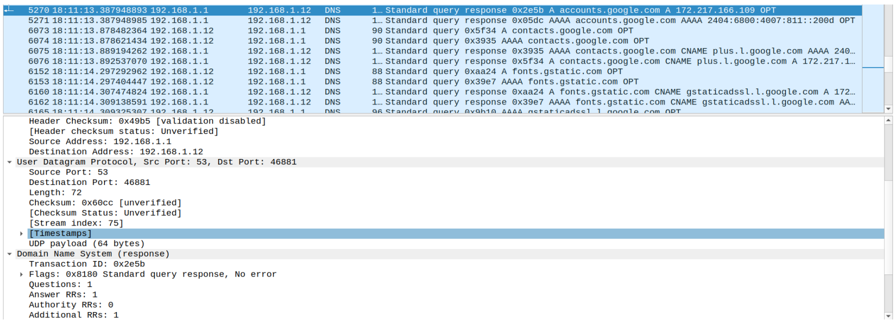
```
No.   Time                  Source          Destination     Protocol Len Info
  690 18:11:10.553561790 192.168.1.12    172.217.166.67 OCSP      4… Request
  692 18:11:10.652815042 172.217.166.67  192.168.1.12   OCSP      7… Response
  980 18:11:11.720915095 192.168.1.12    172.217.166.67 OCSP      4… Request
 1015 18:11:11.774270777 192.168.1.12    172.217.166.67 OCSP      4… Request
 1016 18:11:11.774362245 192.168.1.12    172.217.166.67 OCSP      4… Request
 1020 18:11:11.775252877 192.168.1.12    172.217.166.67 OCSP      4… Request
 1047 18:11:11.784350901 192.168.1.12    172.217.166.67 OCSP      4… Request
 1154 18:11:11.821903821 172.217.166.67  192.168.1.12   OCSP      7… Response
 1317 18:11:11.874683836 172.217.166.67  192.168.1.12   OCSP      7… Response
 1319 18:11:11.874683948 172.217.166.67  192.168.1.12   OCSP      7… Response
 1321 18:11:11.874739813 172.217.166.67  192.168.1.12   OCSP      7… Response
 1360 18:11:11.890284306 172.217.166.67  192.168.1.12   OCSP      7… Response
 2760 18:11:12.296047162 192.168.1.12    172.217.166.67 OCSP      4… Request
 3139 18:11:12.413706074 172.217.166.67  192.168.1.12   OCSP      7… Response
```

```
  ▸ [Timestamps]
    TCP payload (702 bytes)
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 200 OK\r\n
    Content-Type: application/ocsp-response\r\n
    Date: Mon, 01 Feb 2021 12:41:11 GMT\r\n
    Cache-Control: public, max-age=86400\r\n
    Server: ocsp_responder\r\n
  ▸ Content-Length: 472\r\n
    X-XSS-Protection: 0\r\n
    X-Frame-Options: SAMEORIGIN\r\n
    \r\n
    [HTTP response 2/3]
    [Time since request: 0.100988726 seconds]
    [Prev request in frame: 690]
    [Prev response in frame: 692]
    [Request in frame: 980]
    [Next request in frame: 2760]
    [Next response in frame: 3139]
    [Request URI: http://ocsp.pki.goog/gts1o1core]
    File Data: 472 bytes
▸ Online Certificate Status Protocol
```

Figure 1: OCSP packets screenshot

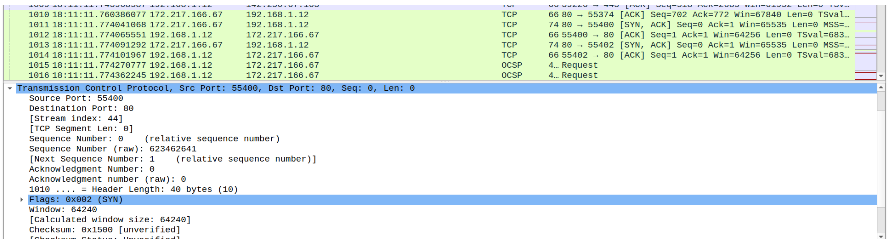
Internet Protocol Version 4 This is a section which contains the information regarding the source and destination ports. The corresponding values are in the Figure 5 .

Note: The Screenshot images are also attached in the zip files



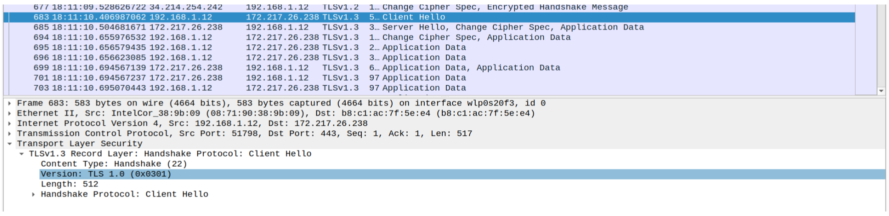
```
5270 18:11:13.387948893 192.168.1.1      192.168.1.12   DNS    1… Standard query response 0x2e5b A accounts.google.com A 172.217.166.109 OPT
5271 18:11:13.387948985 192.168.1.1      192.168.1.12   DNS    1… Standard query response 0x05dc AAAA accounts.google.com AAAA 2404:6800:4007:811::200d OPT
6073 18:11:13.878482364 192.168.1.12     192.168.1.1    DNS    90 Standard query 0x5f34 A contacts.google.com OPT
6074 18:11:13.878621434 192.168.1.12     192.168.1.1    DNS    90 Standard query 0x3935 AAAA contacts.google.com OPT
6075 18:11:13.889194262 192.168.1.1      192.168.1.12   DNS    1… Standard query response 0x3935 AAAA contacts.google.com CNAME plus.l.google.com AAAA 240…
6076 18:11:13.892537070 192.168.1.1      192.168.1.12   DNS    1… Standard query response 0x5f34 A contacts.google.com CNAME plus.l.google.com A 172.217.1…
6152 18:11:14.297292962 192.168.1.12     192.168.1.1    DNS    88 Standard query 0xaa24 A fonts.gstatic.com OPT
6153 18:11:14.297404447 192.168.1.12     192.168.1.1    DNS    88 Standard query 0x39e7 AAAA fonts.gstatic.com OPT
6160 18:11:14.307474824 192.168.1.1      192.168.1.12   DNS    1… Standard query response 0xaa24 A fonts.gstatic.com CNAME gstaticadssl.l.google.com A 172…
6162 18:11:14.309138591 192.168.1.1      192.168.1.12   DNS    1… Standard query response 0x39e7 AAAA fonts.gstatic.com CNAME gstaticadssl.l.google.com AA…
6165 18:11:14.309325307 192.168.1.12     192.168.1.1    DNS    96 Standard query 0x9b10 AAAA gstaticadssl.l.google.com OPT
    Header Checksum: 0x49b5 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.1
    Destination Address: 192.168.1.12
▾ User Datagram Protocol, Src Port: 53, Dst Port: 46881
    Source Port: 53
    Destination Port: 46881
    Length: 72
    Checksum: 0x60cc [unverified]
    [Checksum Status: Unverified]
    [Stream index: 75]
  ▸ [Timestamps]
    UDP payload (64 bytes)
▾ Domain Name System (response)
    Transaction ID: 0x2e5b
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 1
```

Figure 2: DNS packets screenshot



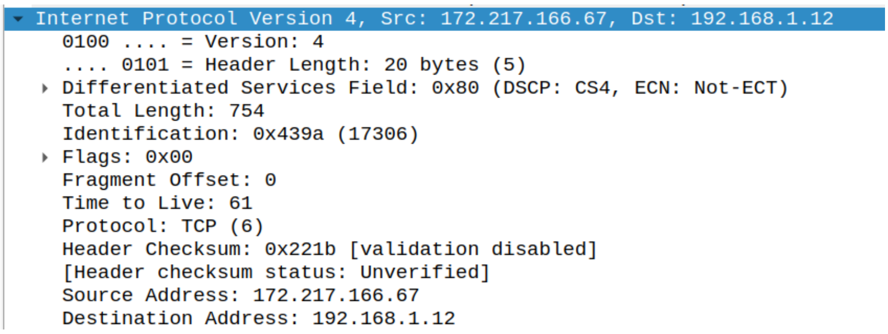
```
1010 18:11:11.760386077 172.217.166.67   192.168.1.12        TCP    66 80 → 55374 [ACK] Seq=702 Ack=772 Win=67840 Len=0 TSval…
1011 18:11:11.774041068 172.217.166.67   192.168.1.12        TCP    74 80 → 55400 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=…
1012 18:11:11.774065551 192.168.1.12     172.217.166.67      TCP    66 55400 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=683…
1013 18:11:11.774091292 172.217.166.67   192.168.1.12        TCP    74 80 → 55402 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=…
1014 18:11:11.774101967 192.168.1.12     172.217.166.67      TCP    66 55402 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=683…
1015 18:11:11.774270777 192.168.1.12     172.217.166.67      OCSP   4… Request
1016 18:11:11.774362245 192.168.1.12     172.217.166.67      OCSP   4… Request
▾ Transmission Control Protocol, Src Port: 55400, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 55400
    Destination Port: 80
    [Stream index: 44]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 623462641
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1010 .... = Header Length: 40 bytes (10)
  ▸ Flags: 0x002 (SYN)
    Window: 64240
    [Calculated window size: 64240]
    Checksum: 0x1500 [unverified]
    [Checksum Status: Unverified]
```

Figure 3: TCP packets screenshot



```
 677 18:11:09.528626722 34.214.254.242   192.168.1.12   TLSv1.2 1… Change Cipher Spec, Encrypted Handshake Message
 683 18:11:10.406987062 192.168.1.12     172.217.26.238 TLSv1.3 5… Client Hello
 685 18:11:10.504681671 172.217.26.238   192.168.1.12   TLSv1.3 3… Server Hello, Change Cipher Spec, Application Data
 694 18:11:10.655976532 192.168.1.12     172.217.26.238 TLSv1.3 1… Change Cipher Spec, Application Data
 695 18:11:10.656579435 192.168.1.12     172.217.26.238 TLSv1.3 2… Application Data
 696 18:11:10.656623085 192.168.1.12     172.217.26.238 TLSv1.3 3… Application Data
 699 18:11:10.694567139 172.217.26.238   192.168.1.12   TLSv1.3 6… Application Data, Application Data
 701 18:11:10.694567237 172.217.26.238   192.168.1.12   TLSv1.3 97 Application Data
 703 18:11:10.695070443 192.168.1.12     172.217.26.238 TLSv1.3 97 Application Data
▸ Frame 683: 583 bytes on wire (4664 bits), 583 bytes captured (4664 bits) on interface wlp0s20f3, id 0
▸ Ethernet II, Src: IntelCor_38:9b:09 (08:71:90:38:9b:09), Dst: b8:c1:ac:7f:5e:e4 (b8:c1:ac:7f:5e:e4)
▸ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 172.217.26.238
▸ Transmission Control Protocol, Src Port: 51798, Dst Port: 443, Seq: 1, Ack: 1, Len: 517
▾ Transport Layer Security
  ▾ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 512
    ▸ Handshake Protocol: Client Hello
```

Figure 4: TLS packets screenshot



```
▾ Internet Protocol Version 4, Src: 172.217.166.67, Dst: 192.168.1.12
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x80 (DSCP: CS4, ECN: Not-ECT)
    Total Length: 754
    Identification: 0x439a (17306)
  ▸ Flags: 0x00
    Fragment Offset: 0
    Time to Live: 61
    Protocol: TCP (6)
    Header Checksum: 0x221b [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.217.166.67
    Destination Address: 192.168.1.12
```

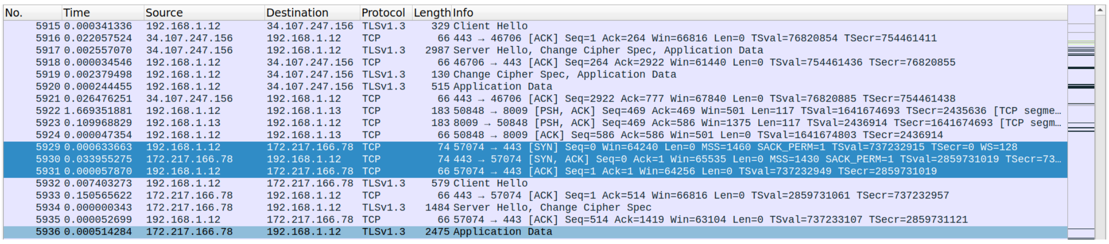
Figure 5: IP values screenshot

3 Question 3

The available functionalities for google hangout application/website are login make a call, send a message etc. When we use any of these functionalities, the number of TCP and TLSv1.2 values increases, and hence we can say that there are handshaking sequences involved in the packets corresponding to the functionality. I have attached another trace file in the submission along with the zip file(because of the 25MB limit), please refer to that file if necessary.

In the Figure 6, we can see that there is packet with SYN Flag from (192.)(i.e my system/client) to 172. (i.e google hangouts/server) and then there is packet from hangouts to my computer with SYN and ACK flags, which means that the server has acknowledged the syn packet from my computer, then there is ACK packet from my laptop to the hangouts server with ACK flag, acknowledging the syn packet from the server.

Hence there are 3 packets with SYN(client→server), SYN and ACK(server→client), ACK(client→server), hence it is a 3-step handshaking protocol.



No.	Time	Source	Destination	Protocol	Length	Info
5915	0.000341336	192.168.1.12	34.107.247.156	TLSv1.3	329	Client Hello
5916	0.022057524	34.107.247.156	192.168.1.12	TCP	66	443 → 46706 [ACK] Seq=1 Ack=264 Win=66816 Len=0 TSval=76820854 TSecr=754461411
5917	0.002557070	34.107.247.156	192.168.1.12	TLSv1.3	2987	Server Hello, Change Cipher Spec, Application Data
5918	0.000034546	192.168.1.12	34.107.247.156	TCP	66	46706 → 443 [ACK] Seq=264 Ack=2922 Win=61440 Len=0 TSval=754461436 TSecr=76820855
5919	0.002379498	192.168.1.12	34.107.247.156	TLSv1.3	130	Change Cipher Spec, Application Data
5920	0.000244455	192.168.1.12	34.107.247.156	TLSv1.3	515	Application Data
5921	0.026476251	34.107.247.156	192.168.1.12	TCP	66	443 → 46706 [ACK] Seq=2922 Ack=777 Win=67840 Len=0 TSval=76820885 TSecr=754461438
5922	1.669351881	192.168.1.12	192.168.1.13	TCP	183	50848 → 8009 [PSH, ACK] Seq=469 Ack=469 Win=501 Len=117 TSval=1641674693 TSecr=2435636 [TCP segme…
5923	0.109968829	192.168.1.13	192.168.1.12	TCP	183	8009 → 50848 [PSH, ACK] Seq=469 Ack=586 Win=1375 Len=117 TSval=2436914 TSecr=1641674693 [TCP segm…
5924	0.000047354	192.168.1.12	192.168.1.13	TCP	66	50848 → 8009 [ACK] Seq=586 Ack=586 Win=501 Len=0 TSval=1641674803 TSecr=2436914
5929	0.000633663	192.168.1.12	172.217.166.78	TCP	74	57074 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=737232915 TSecr=0 WS=128
5930	0.033955275	172.217.166.78	192.168.1.12	TCP	74	443 → 57074 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 TSval=2859731019 TSecr=73…
5931	0.000057870	192.168.1.12	172.217.166.78	TCP	66	57074 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=737232949 TSecr=2859731019
5932	0.007403273	192.168.1.12	172.217.166.78	TLSv1.3	579	Client Hello
5933	0.150565622	172.217.166.78	192.168.1.12	TCP	66	443 → 57074 [ACK] Seq=1 Ack=514 Win=66816 Len=0 TSval=2859731061 TSecr=737232957
5934	0.000000343	172.217.166.78	192.168.1.12	TLSv1.3	1484	Server Hello, Change Cipher Spec
5935	0.000052699	192.168.1.12	172.217.166.78	TCP	66	57074 → 443 [ACK] Seq=514 Ack=1419 Win=63104 Len=0 TSval=737233107 TSecr=2859731121
5936	0.000514284	172.217.166.78	192.168.1.12	TLSv1.3	2475	Application Data

Figure 6: Functionality packets screenshot

4 Question 4

The protocols like HTTP(OCSP), DNS, TCP, TLS are important because, they each perform a different task and are necessary for the functioning of the application as whole. DNS is used to get the IP address corresponding to the google.hangout url, so that we can establish a connection. HTTP gets data like request methods of the packets, accept language, browser information etc. TCP establishes a reliable and secure connection for data transfer with the handshaking protocol. TLS adds another layer of security by encrypting the data. IP is used to obtain the source and destination IP addresses Hence as we can see each protocol performs a different task and hence are relevant for the overall functioning of the application.

The usages of protocols is listed in tabular form in Table 1

5 Question 5

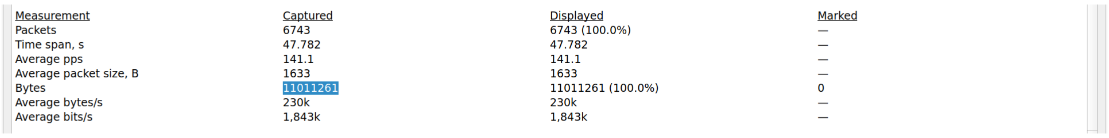
Statistics from your traces while performing experiments is shown in Figure 7

Throughput=11011261/47.782=230447.888326

RTT is different for different packets, hence I found the min and max values by arranging them in ascending order.
RTT ranges from 0.000000029 to 2.989372596.

Sl. No	**Protocol**	**Use**
1	HTTP(OCSP)	HTTP gets data like request methods of the packets, accept language, browser information etc
2	DNS	DNS is used to get the IP address corresponding to the google.hangout url, so that we can establish a connection.
3	TCP	TCP establishes a reliable and secure connection for data transfer with the handshaking protocol
4	TLS	TLS adds another layer of security by encrypting the data.
5	IP	IP is used to obtain the source and destination IP addresses

Table 1: Protocols and its uses



Measurement	Captured	Displayed	Marked
Packets	6743	6743 (100.0%)	—
Time span, s	47.782	47.782	—
Average pps	141.1	141.1	—
Average packet size, B	1633	1633	—
Bytes	11011261	11011261 (100.0%)	0
Average bytes/s	230k	230k	—
Average bits/s	1,843k	1,843k	—

Figure 7: Statistics screenshot

Packet Size is different for different packets, hence I found the min and max values by arranging them in ascending order.
Packet size ranges from 54 (for non ARP packets) to 15664.

Number of TCP packets lost = 54(obtained by the filter tcp.analysis.lost_segment)

Number of TCP packets = 6476 (cosidering the TLSv packets also)
Number of UDP packets = 203 (considering the DNS, MDNS packets also)

Number of packets with source as my laptop = 2850 (obtained by filter ip.src==192.168.1.12)
Number of packets with destination as my laptop = 3819 (obtained by filter ip.dst==192.168.1.12)

Number of responses received with respect to one request sent = 3819/2850
Number of responses received with respect to one request sent=1.34

All the values are tabulated in Table 2

Sl. No	**Parameter**	**Value**
1	Throughput	230447.888326
2	RTT	0.000000029 to 2.989372596
3	Packet Size	54 to 15664
4	TCP Packets Lost	54
5	Number of TCP packets	6476
6	Number of UDP packets	203
6	Number of responses per request	1.34

Table 2: Statistics

6 Question 6

The whole content is being sent from multiple locations, the list of those IP addresses is as follows:

- 172.217.166.109 (main)
- 142.250.67.163
- 216.58.203.46

It is common for servers to have multiple IP addresses, because when one of the IP address is down, the others can act as backup and also each IP address has its own merits and reputability, using different IP addresses would give the benefits of each of them.