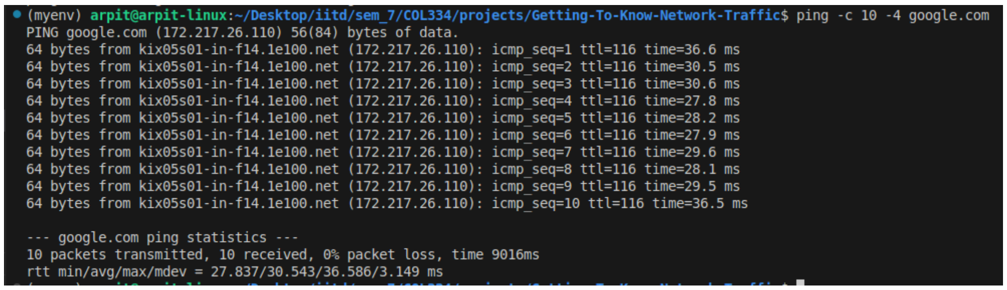
Assignment 1: Getting to Know Network Traffic

Arpit Prasad
COL334: Computer Network

August 22, 2025

1 Measurement Tools

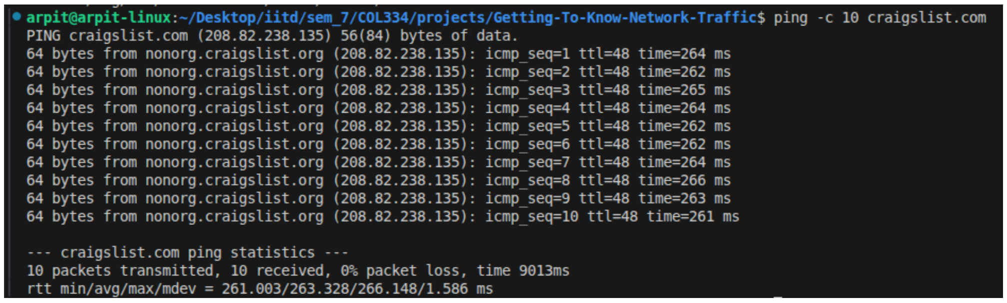
1.1 ping



```
● (myenv) arpit@arpit-linux:~/Desktop/iitd/sem_7/COL334/projects/Getting-To-Know-Network-Traffic$ ping -c 10 -4 google.com
  PING google.com (172.217.26.110) 56(84) bytes of data.
  64 bytes from kix05s01-in-f14.1e100.net (172.217.26.110): icmp_seq=1 ttl=116 time=36.6 ms
  64 bytes from kix05s01-in-f14.1e100.net (172.217.26.110): icmp_seq=2 ttl=116 time=30.5 ms
  64 bytes from kix05s01-in-f14.1e100.net (172.217.26.110): icmp_seq=3 ttl=116 time=30.6 ms
  64 bytes from kix05s01-in-f14.1e100.net (172.217.26.110): icmp_seq=4 ttl=116 time=27.8 ms
  64 bytes from kix05s01-in-f14.1e100.net (172.217.26.110): icmp_seq=5 ttl=116 time=28.2 ms
  64 bytes from kix05s01-in-f14.1e100.net (172.217.26.110): icmp_seq=6 ttl=116 time=27.9 ms
  64 bytes from kix05s01-in-f14.1e100.net (172.217.26.110): icmp_seq=7 ttl=116 time=29.6 ms
  64 bytes from kix05s01-in-f14.1e100.net (172.217.26.110): icmp_seq=8 ttl=116 time=28.1 ms
  64 bytes from kix05s01-in-f14.1e100.net (172.217.26.110): icmp_seq=9 ttl=116 time=29.5 ms
  64 bytes from kix05s01-in-f14.1e100.net (172.217.26.110): icmp_seq=10 ttl=116 time=36.5 ms

  --- google.com ping statistics ---
  10 packets transmitted, 10 received, 0% packet loss, time 9016ms
  rtt min/avg/max/mdev = 27.837/30.543/36.586/3.149 ms
```

Figure 1: Ten Pings to google.com



```
● arpit@arpit-linux:~/Desktop/iitd/sem_7/COL334/projects/Getting-To-Know-Network-Traffic$ ping -c 10 craigslist.com
  PING craigslist.com (208.82.238.135) 56(84) bytes of data.
  64 bytes from nonorg.craigslist.org (208.82.238.135): icmp_seq=1 ttl=48 time=264 ms
  64 bytes from nonorg.craigslist.org (208.82.238.135): icmp_seq=2 ttl=48 time=262 ms
  64 bytes from nonorg.craigslist.org (208.82.238.135): icmp_seq=3 ttl=48 time=265 ms
  64 bytes from nonorg.craigslist.org (208.82.238.135): icmp_seq=4 ttl=48 time=264 ms
  64 bytes from nonorg.craigslist.org (208.82.238.135): icmp_seq=5 ttl=48 time=262 ms
  64 bytes from nonorg.craigslist.org (208.82.238.135): icmp_seq=6 ttl=48 time=262 ms
  64 bytes from nonorg.craigslist.org (208.82.238.135): icmp_seq=7 ttl=48 time=264 ms
  64 bytes from nonorg.craigslist.org (208.82.238.135): icmp_seq=8 ttl=48 time=266 ms
  64 bytes from nonorg.craigslist.org (208.82.238.135): icmp_seq=9 ttl=48 time=263 ms
  64 bytes from nonorg.craigslist.org (208.82.238.135): icmp_seq=10 ttl=48 time=261 ms

  --- craigslist.com ping statistics ---
  10 packets transmitted, 10 received, 0% packet loss, time 9013ms
  rtt min/avg/max/mdev = 261.003/263.328/266.148/1.586 ms
```

Figure 2: Ten Pings to craigslist.com

1. **Protocols Used**: Ping uses ICMP protocol. It sends the Echo Request packet to the destination host and recieves Echo Reply packet from the same. It sits on the third layer of the protocol stack, which is the network layer.

2. **Latency**:

(a) **Avg Latency of Craigslist**: 263.328 ms

(b) **Avg Latency of Google**: 30.543 ms

(c) Google's host has **smaller RTT** than Craigslist

(d) **Reason for different latencies of websites**:

   i. Google has more number of hosts than Craigslist, which splits newtwork traffic

   ii. Google's traffic might directly peer with most of the ISPs in the same tier of internet hierarchy (Regional IPSs).

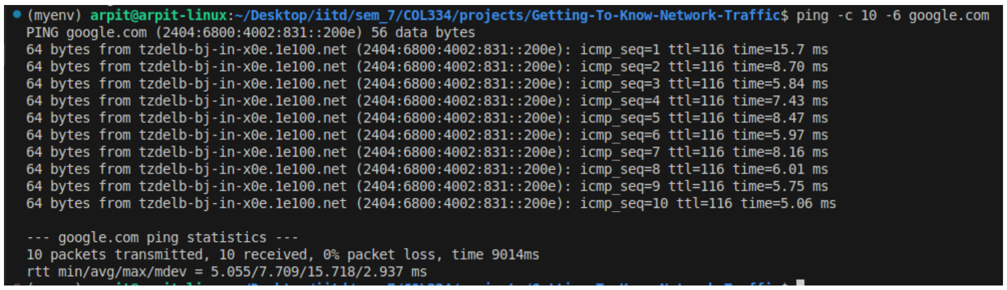
(e) **Reason for differnt latencies across pings for same website**:

   i. Length of Queue for service at destination host is not constant in time and varies according to the number of user who requested for service before we place any request (queueing of packets, at the host)

   ii. Network congestion is not constant, hence each switch in the network may not have same number of packets it has to route across differnt time

   iii. Differnt routes may be taken for different pings leading to different paths and hence latencies

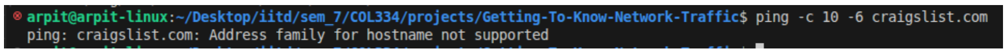
3. **Using IPv6 for both websites**



```
(myenv) arpit@arpit-linux:~/Desktop/iitd/sem_7/COL334/projects/Getting-To-Know-Network-Traffic$ ping -c 10 -6 google.com
PING google.com (2404:6800:4002:831::200e) 56 data bytes
64 bytes from tzdelb-bj-in-x0e.1e100.net (2404:6800:4002:831::200e): icmp_seq=1 ttl=116 time=15.7 ms
64 bytes from tzdelb-bj-in-x0e.1e100.net (2404:6800:4002:831::200e): icmp_seq=2 ttl=116 time=8.70 ms
64 bytes from tzdelb-bj-in-x0e.1e100.net (2404:6800:4002:831::200e): icmp_seq=3 ttl=116 time=5.84 ms
64 bytes from tzdelb-bj-in-x0e.1e100.net (2404:6800:4002:831::200e): icmp_seq=4 ttl=116 time=7.43 ms
64 bytes from tzdelb-bj-in-x0e.1e100.net (2404:6800:4002:831::200e): icmp_seq=5 ttl=116 time=8.47 ms
64 bytes from tzdelb-bj-in-x0e.1e100.net (2404:6800:4002:831::200e): icmp_seq=6 ttl=116 time=5.97 ms
64 bytes from tzdelb-bj-in-x0e.1e100.net (2404:6800:4002:831::200e): icmp_seq=7 ttl=116 time=8.16 ms
64 bytes from tzdelb-bj-in-x0e.1e100.net (2404:6800:4002:831::200e): icmp_seq=8 ttl=116 time=6.01 ms
64 bytes from tzdelb-bj-in-x0e.1e100.net (2404:6800:4002:831::200e): icmp_seq=9 ttl=116 time=5.75 ms
64 bytes from tzdelb-bj-in-x0e.1e100.net (2404:6800:4002:831::200e): icmp_seq=10 ttl=116 time=5.06 ms

--- google.com ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 5.055/7.709/15.718/2.937 ms
```

Figure 3: Ten Pings to google.com using IPv6



```
arpit@arpit-linux:~/Desktop/iitd/sem_7/COL334/projects/Getting-To-Know-Network-Traffic$ ping -c 10 -6 craigslist.com
ping: craigslist.com: Address family for hostname not supported
```

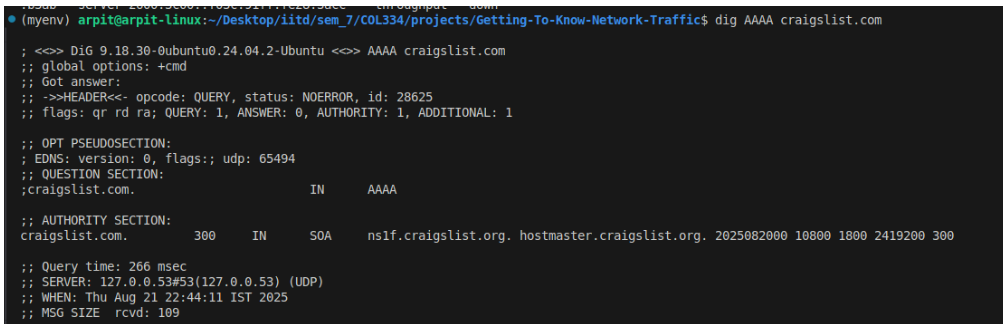
Figure 4: Error when pinging craigslist.com

(a) **How to force IPv6**: pass a flag -6 to force ping to follow IPv6

(b) **Result**: IPv6 was supported by Google's host but not by Craigslist's host

(c) **Why** $ping - 6$ **Failed for Craigslist's host**:

   i. When checking the IPv6 address for craigslist.com using dig AAAA craigslist.com, my computer does not find any IPv6 address as can be seen from Fig. 5, hence does not know which address to resolve to. Therefore, forcing ping to follow IPv6 Addresing cannot be executed.



```
(myenv) arpit@arpit-linux:~/Desktop/iitd/sem_7/COL334/projects/Getting-To-Know-Network-Traffic$ dig AAAA craigslist.com

; <<>> DiG 9.18.30-0ubuntu0.24.04.2-Ubuntu <<>> AAAA craigslist.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28625
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;craigslist.com.                        IN      AAAA

;; AUTHORITY SECTION:
craigslist.com.         300     IN      SOA     ns1f.craigslist.org. hostmaster.craigslist.org. 2025082000 10800 1800 2419200 300

;; Query time: 266 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Thu Aug 21 22:44:11 IST 2025
;; MSG SIZE  rcvd: 109
```

Figure 5: dig tool being used to find the IPv6 Address of craigslist.com

ii. Also, since IPv6 worked for Google's host, implies that my computer and Google's host both support IPv6. If a failure of Address Family support has occured it must have occured on Craigslist's server. This implies Craigslist's server does not support IPv6 addresses.

4. **Max size of the packets**

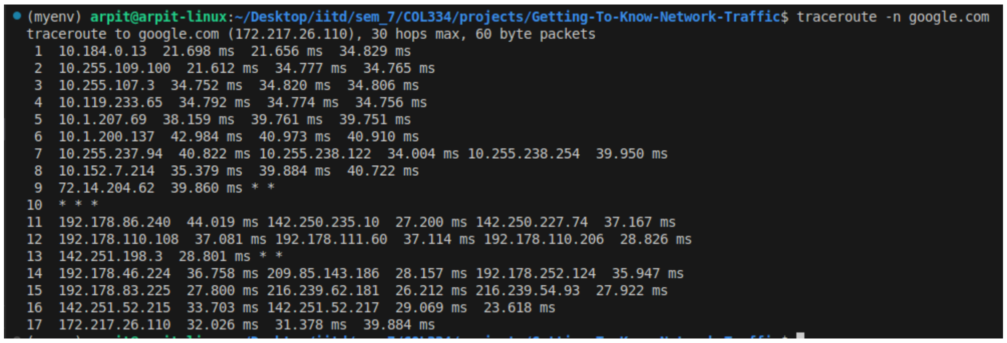
   (a) Max Size = 65535 bits

   (b) The length of the field - "total length" in the packet structure - which indicates the size of the payeload is 16, for both IPv4 and IPv6 Addressing in ping packets. Hence, the total payload size = $2^{16} - 1$ (the 1 excludes all bits zero, which implies zero size payload) = 65535 bits. Theoretically, the protocol allows these many number of bits to be sent as data in the payload.

1.2 traceroute

1. IPv4 Address for google.com : 172.217.26.110

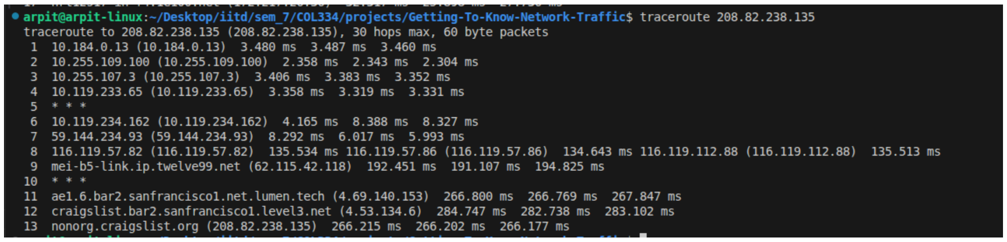
2. IPv4 Address for craigslist.com : 208.82.238.135



```
(myenv) arpit@arpit-linux:~/Desktop/iitd/sem_7/COL334/projects/Getting-To-Know-Network-Traffic$ traceroute -n google.com
traceroute to google.com (172.217.26.110), 30 hops max, 60 byte packets
 1  10.184.0.13  21.698 ms  21.656 ms  34.829 ms
 2  10.255.109.100  21.612 ms  34.777 ms  34.765 ms
 3  10.255.107.3  34.752 ms  34.820 ms  34.806 ms
 4  10.119.233.65  34.792 ms  34.774 ms  34.756 ms
 5  10.1.207.69  38.159 ms  39.761 ms  39.751 ms
 6  10.1.200.137  42.984 ms  40.973 ms  40.910 ms
 7  10.255.237.94  40.822 ms 10.255.238.122  34.004 ms 10.255.238.254  39.950 ms
 8  10.152.7.214  35.379 ms  39.884 ms  40.722 ms
 9  72.14.204.62  39.860 ms * *
10  * * *
11  192.178.86.240  44.019 ms 142.250.235.10  27.200 ms 142.250.227.74  37.167 ms
12  192.178.110.108  37.081 ms 192.178.111.60  37.114 ms 192.178.110.206  28.826 ms
13  142.251.198.3  28.801 ms * *
14  192.178.46.224  36.758 ms 209.85.143.186  28.157 ms 192.178.252.124  35.947 ms
15  192.178.83.225  27.800 ms 216.239.62.181  26.212 ms 216.239.54.93  27.922 ms
16  142.251.52.215  33.703 ms 142.251.52.217  29.069 ms  23.618 ms
17  172.217.26.110  32.026 ms  31.378 ms  39.884 ms
```

Figure 6: Trace Route of sending packet to google.com

A **Number of Hops**:

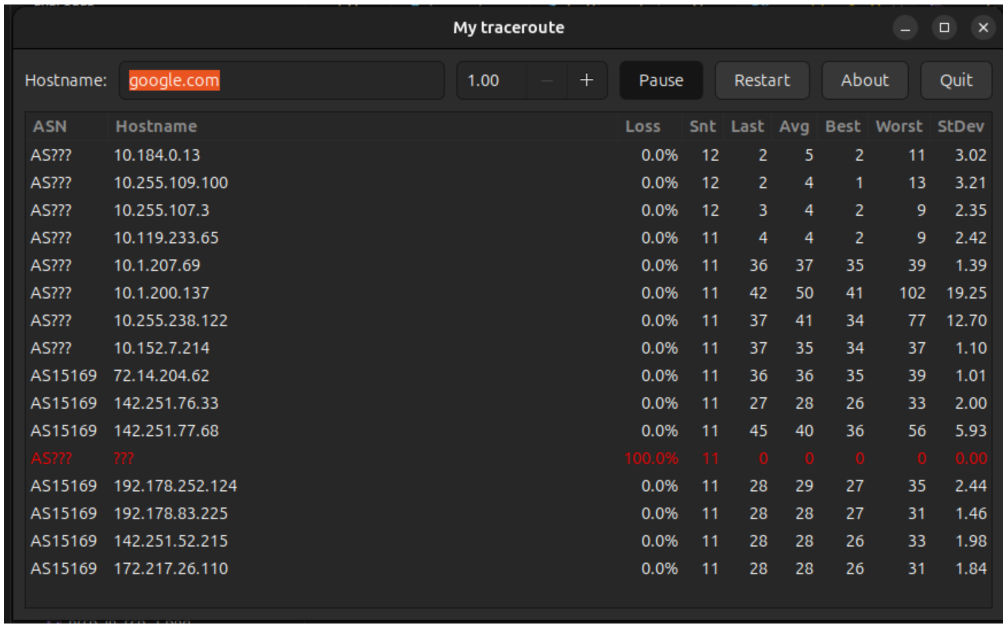


```
arpit@arpit-linux:~/Desktop/iitd/sem_7/COL334/projects/Getting-To-Know-Network-Traffic$ traceroute 208.82.238.135
traceroute to 208.82.238.135 (208.82.238.135), 30 hops max, 60 byte packets
 1  10.184.0.13 (10.184.0.13)  3.480 ms  3.487 ms  3.460 ms
 2  10.255.109.100 (10.255.109.100)  2.358 ms  2.343 ms  2.304 ms
 3  10.255.107.3 (10.255.107.3)  3.406 ms  3.383 ms  3.352 ms
 4  10.119.233.65 (10.119.233.65)  3.358 ms  3.319 ms  3.331 ms
 5  * * *
 6  10.119.234.162 (10.119.234.162)  4.165 ms  8.388 ms  8.327 ms
 7  59.144.234.93 (59.144.234.93)  8.292 ms  6.017 ms  5.993 ms
 8  116.119.57.82 (116.119.57.82)  135.534 ms 116.119.57.86 (116.119.57.86)  134.643 ms 116.119.112.88 (116.119.112.88)  135.513 ms
 9  mei-b5-link.ip.twelve99.net (62.115.42.118)  192.451 ms  191.107 ms  194.825 ms
10  * * *
11  ae1.6.bar2.sanfrancisco1.net.lumen.tech (4.69.140.153)  266.800 ms  266.769 ms  267.847 ms
12  craigslist.bar2.sanfrancisco1.level3.net (4.53.134.6)  284.747 ms  282.738 ms  283.102 ms
13  nonorg.craigslist.org (208.82.238.135)  266.215 ms  266.202 ms  266.177 ms
```

Figure 7: Trace Route of sending packet to craigslist.com

Table 1: Number of Hops for Websites

Host	Number of Hops
google.com	17
craigslist.com	13



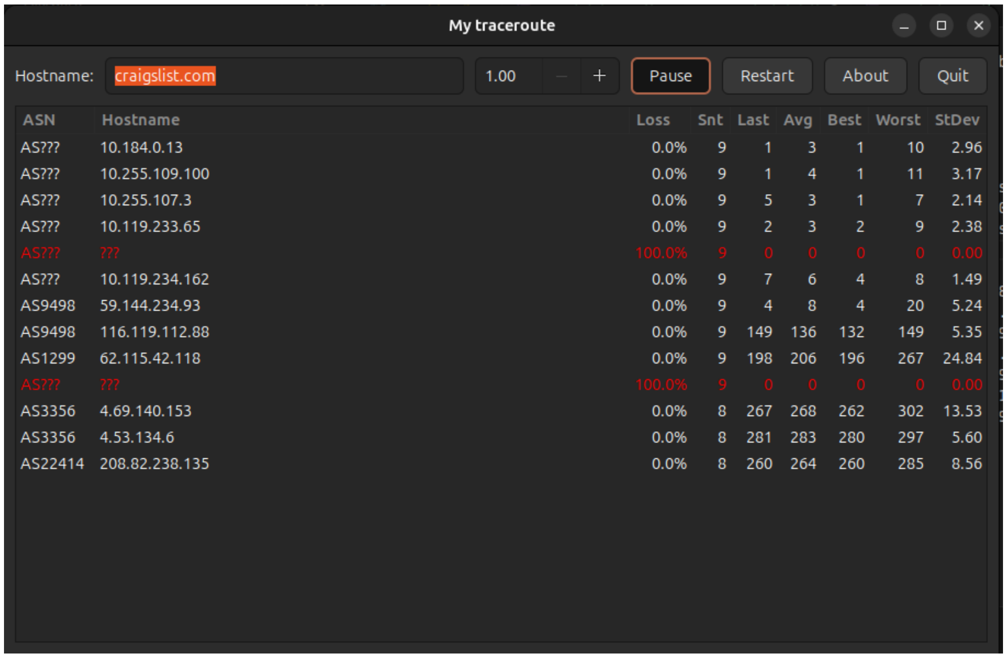
```
ASN      Hostname          Loss    Snt  Last  Avg  Best  Worst  StDev
AS???    10.184.0.13        0.0%    12     2    5     2     11   3.02
AS???    10.255.109.100     0.0%    12     2    4     1     13   3.21
AS???    10.255.107.3       0.0%    12     3    4     2      9   2.35
AS???    10.119.233.65      0.0%    11     4    4     2      9   2.42
AS???    10.1.207.69        0.0%    11    36   37    35     39   1.39
AS???    10.1.200.137       0.0%    11    42   50    41    102  19.25
AS???    10.255.238.122     0.0%    11    37   41    34     77  12.70
AS???    10.152.7.214       0.0%    11    37   35    34     37   1.10
AS15169  72.14.204.62       0.0%    11    36   36    35     39   1.01
AS15169  142.251.76.33      0.0%    11    27   28    26     33   2.00
AS15169  142.251.77.68      0.0%    11    45   40    36     56   5.93
AS???    ???              100.0%    11     0    0     0      0   0.00
AS15169  192.178.252.124    0.0%    11    28   29    27     35   2.44
AS15169  192.178.83.225     0.0%    11    28   28    27     31   1.46
AS15169  142.251.52.215     0.0%    11    28   28    26     33   1.98
AS15169  172.217.26.110     0.0%    11    28   28    26     31   1.84
```

Figure 8: Autonomous System Number (denoted as ASN here) for each IP Address in the trace route of google.com (produced using the tool: mtr)

B **Explanation for "*"**: Some servers do not cater to traceroute packets, for security reasons and traffic control, hence do not send the Time Exceeded packet back to the source and hence, we do not have information about this node. This node is represented in the traceroute by "*"

C **Multiple IP Addresses for the same Hop Count**: *traceroute* sends three packets for each hop. The three packets may opt for independent routes, depending on the congestion of the network. traceroute lists all the unique ip addresses of the nodes encountered by the three packets

D The following tables (Table 2 for google.com and Table 3 for craigslist.com) **lists the**



```
ASN      Hostname                     Loss  Snt  Last  Avg  Best  Worst  StDev
AS???    10.184.0.13                  0.0%    9     1    3     1     10   2.96
AS???    10.255.109.100               0.0%    9     1    4     1     11   3.17
AS???    10.255.107.3                 0.0%    9     5    3     1      7   2.14
AS???    10.119.233.65                0.0%    9     2    3     2      9   2.38
AS???    ???                        100.0%    9     0    0     0      0   0.00
AS???    10.119.234.162               0.0%    9     7    6     4      8   1.49
AS9498   59.144.234.93                0.0%    9     4    8     4     20   5.24
AS9498   116.119.112.88               0.0%    9   149  136   132    149   5.35
AS1299   62.115.42.118                0.0%    9   198  206   196    267  24.84
AS???    ???                        100.0%    9     0    0     0      0   0.00
AS3356   4.69.140.153                 0.0%    8   267  268   262    302  13.53
AS3356   4.53.134.6                   0.0%    8   281  283   280    297   5.60
AS22414  208.82.238.135               0.0%    8   260  264   260    285   8.56
```

Figure 9: Autonomous System Number (denoted as ASN here) for each IP Address in the trace route of craigslist.com (produced using the tool: mtr)

IP Addresses and their correspoding RTTs and Geolocations

Table 2: IP Addresses and their GeoLocations for google.com. Note: NA means Not Available from the respective method listed in the column

Sl No	IP Address	DNS	DNS Geolocation	Maxmind Geolocation	RTT (ms)
1	10.184.0.13	NA	NA	NA	8.457
2	10.255.109.100	NA	NA	NA	9.288
3	10.255.107.3	NA	NA	NA	9.235
4	10.119.233.65	NA	NA	NA	9.184
5	10.1.207.69	NA	NA	NA	37.041
6	10.1.200.137	NA	NA	NA	42.454
7	10.255.238.122	NA	NA	NA	34.988
8	10.152.7.214	NA	NA	NA	37.466
9	*	NA	NA	NA	*
10	*	NA	NA	NA	*
11	72.14.233.58	NA	NA	United States (US), North America	39.810
12	192.178.110.204	NA	NA	United States (US), North America	27.622
13	*	NA	NA	NA	142.251.198.3
14	192.178.252.110	NA	NA	United States (US), North America	31.866
15	216.239.54.93	NA	NA	United States (US), North America	34.751
16	142.251.52.215	NA	NA	United States (US), North America	26.822
17	172.217.26.110	kix05s01-in-f14.1e100.net or kix05s01-in-f110.1e100.net or tzdelb-bj-in-f14.1e100.net	Osaka Japan or Tanzania	United States (US), North America	27.744

(a) **For google.com** : most of the network devices that relay the packet are private and hence very less information is obtained about them. However we observe a delta in 5th hop, therefore we can assume that the packet has crossed the country and can be confirmed with RTTs in later hops.

(b) **For craigslist.com** : The bigger deltas in RTT are observed when there is a change in country. For eg., from Table 3, we observe that upto Bangalore the RTT was 5.77 ms but as the relay proceeded to the US, the RTT becomes signficanly higher, 197.05 ms. (Here, Geolocation is referenced from Maxmind Database)

Table 3: IP Addresses and their GeoLocations for craigslist.com. Note: NA means Not Available from the respective method listed in the column

Sl No	IP Address	DNS	DNS Geolocation	Maxmind Geolocation	RTT (ms)
1	10.184.0.13	NA	NA	NA	685.733
2	10.255.109.100	NA	NA	NA	685.597
3	10.255.107.3	NA	NA	NA	685.551
4	10.119.233.65	NA	NA	NA	685.507
5	*	NA	NA	NA	*
6	10.119.234.162	NA	NA	NA	685.376
7	59.144.234.93	NA	NA	Bengaluru, Karnataka, India	5.715
8	116.119.112.88	NA	NA	India	138.551
9	62.115.42.118	mei-b5-link.ip.twelve99.net	Meridian, Mississippi, USA	France	197.050
10	*	NA	NA	NA	*
11	4.69.140.153	ae1.6.bar2.SanFrancisco1.net.lumen.tech	San Francisco, California, United States (US), North America	United States, North America	268.401
12	4.53.134.6	CRAIGSLIST.bar2.SanFrancisco1.Level3.net	San Francisco, California, United States (US), North America	San Francisco, California, United States (US), North America	270.387
13	208.82.238.135	nonorg.craigslist.org	San Francisco, California, United States (US), North America	San Francisco, California, United States (US), North America	259.620

Hence, from the above explanation, the data intuitively makes sense.

E **Three Tier Architecture**:

(a) **craigslist.com** : Here we observe the three tier architecture clearly. Since first the packet travels from Delhi to Bangalore (which is a 2nd tier ISP transfer), then the exchange is observed from Bangalore to France and France to San Fransico which are a 1st Tier ISP Transfer. Finally 2nd and 3rd tier transfers occur for the packet to reach craigslist.com server

(b) **google.com** : Here, we do not observe the three tier architecture clearly. Most of the transfers are through private ip addresses. Google peers its packets in the same level of hierarchy (the Regional ISPs). This is the reason why we observe so many regional ISPs (Unites States (US), North America).

2 Network Traffic Collection and Analysis

2.1 Traffic Capture

A **Median Time taken** for the DNS request-response to comlete: **42.438ms**. **Note**: Median was taken on query response times observed in the pcap file when applied with the filter of DNS. (This was done using python script, code present in traffic_analysis.py func: get_dns_query_response_times)

B **HTTP**

(a) **Number of HTTP Requests** = 363 (fitler used: *http.request*)

(b) **How webpages are structured**: The webpage is structured like a tree. The root of the tree is the webpage itself. Root's children are the first level abstraction in the webpage, such as the title, the box that contains green ticks (as present in the website httpvshttps.com) etc.. The next level with each abstraction, contains the green ticks itself, in the case of the abstraction mentioned previously.

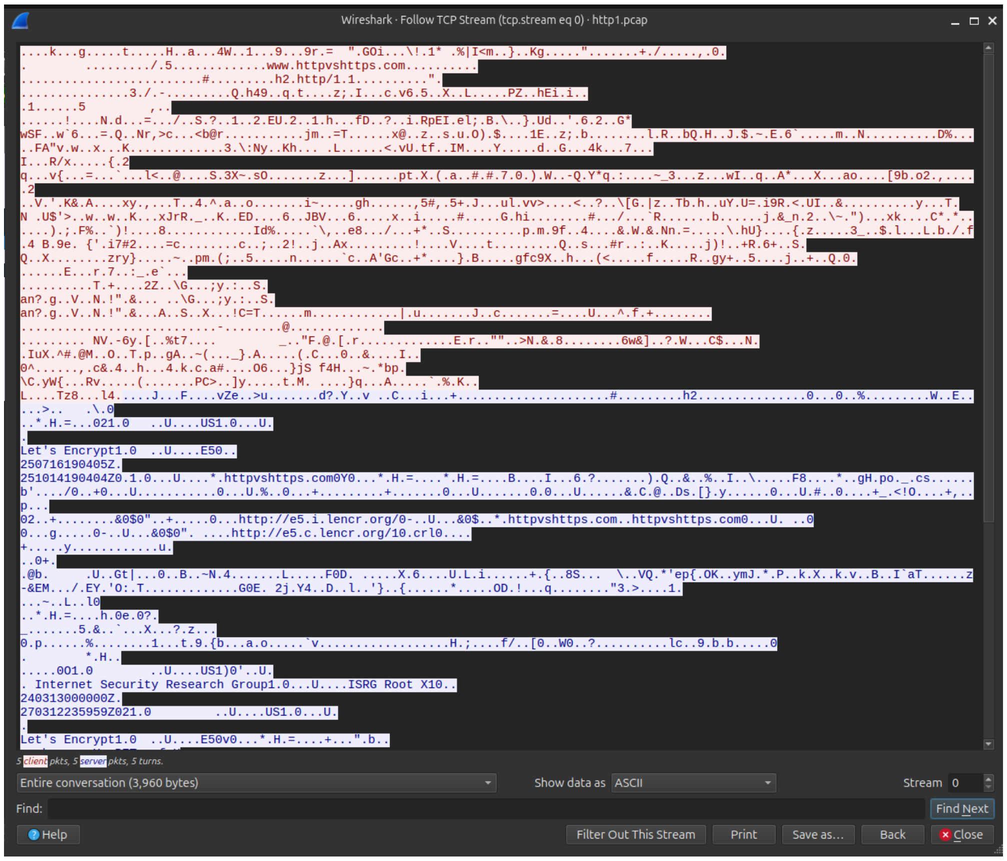
(c) **How browsers render complex pages with multiple images and files**: The browser recursively calls on the nodes of the tree (mentioned above) and fetches the required informaiton, and displays them according to their HTML Code. This way it is able to render complex images and texts. This was observed from the way images were downloaded and logged onto the wireshark framework

C TCP Connection

(a) **Number of TCP Connections** = 31 (filter used: $tcp.flags.syn == 1 and tcp.flags.ack == 0$)

(b) **Are Number of HTTP Conbnections==Number of TCP Connections**: No, they are generally not equal, however they are related. In one TCP Connection multiple HTTP requests can be made.

(c) **Content Object Fetch over same TCP Connection**: Yes, some contents gets fetched over the same TCP Connection. This can be supported from the fact that HTTP transfer are made with HTTP/1.1, which implies sustained TCP Connection for multiple HTTP transfers. This was verified using the filter *tcp.stream* == 0 where we checked the upstream flow. This showed multiple to and fro packet flows.



Wireshark · Follow TCP Stream (tcp.stream eq 0) · http1.pcap

```
....k...g.....t.....H..a...4W..1...9...9r.=  ".GOi...\!.1* .%|I<m..}..Kg....."......+./.....,.0.
.        ........./.5............www.httpvshttps.com.........
...........................#.........h2.http/1.1..........".
...............3./.-.........Q.h49..q.t....z;.I...c.v6.5..X..L.....PZ..hEi.i..
.1......5         ,..
......!....N.d...=.../..S.?..1..2.EU.2..1.h...fD..?..i.RpEI.el;.B.\..}.Ud..'.6.2..G*
wSF..w`6...=.Q..Nr,>c...<b@r...........jm..=T......x@..z..s.u.O).$....1E..z;.b........l.R..bQ.H..J.$.~.E.6`.....m..N..........D%...
..FA"v.w..x...K.............3.\:Ny..Kh...  .L......<.vU.tf..IM....Y.....d..G...4k...7...
I...R/x.....{.2
q...v{...=...`...l<..@....S.3X~.sO.......z...].....pt.X.(.a..#.#.7.0.).W..-Q.Y*q.:....~_3...z...wI..q..A*...X...ao....[9b.o2.,....
.2
..V.'.K&.A....xy.,...T..4.^.a..o.......i~.....gh......,5#,.5+.J...ul.vv>....<..?..\[G.|z..Tb.h..uY.U=.i9R.<.UI..&..........y...T.
N .U$'>..w..w..K...xJrR._..K..ED....6..JBV...6.....x..i.....#.....G.hi........#.../...`R.......b......j.&_n.2..\~.")...xk....C*.*..
....).;.F%..`)!....8............Id%.....`\,..e8.../...+*..S..........p.m.9f..4....&.W.&.Nn.=.....\.hU}....{.z.....3_..$.l...L.b./.f
.4 B.9e. {'.i7#2....=c........c..;..2!..j..Ax.........!....V....t.........Q..s...#r..:..K.....j)!..+R.6+..S.
Q..X........zry}.....~..pm.(;..5.....n......`c..A'Gc..+*....}.B.....gfc9X..h...(<.....f.....R..gy+..5....j..+..Q.0.
......E...r.7..:_.e`...
..........T.+....2Z..\G...;y.:..S.
an?.g..V..N.!".&...  ..\G...;y.:..S.
an?.g..V..N.!".&...A..S..X...!C=T......m............|.u.......J..c.......=....U...^.f.+........
............................-........@............
.........  NV.-6y.[..%t7....         _.."F.@.[.r.............E.r..""..>N.&.8........6w&]..?.W...C$...N.
.IuX.^#.@M..O..T.p..gA..~(..._}.A.....(.C...0..&....I..
0^......,.c&.4..h...4.k.c.a#....O6...}jS f4H...~.*bp.
\C.yW{...Rv.....(.......PC>..]y.....t.M.  ....}q...A.....`.%.K..
L....Tz8...l4.....J...F....vZe..>u.......d?.Y..v ..C...i...+....................#.........h2...............0...0..%.........W..E..
...>..   .\.0
..*.H.=...021.0   ..U....US1.0...U.
.
Let's Encrypt1.0  ..U....E50..
250716190405Z.
251014190404Z0.1.0...U....*.httpvshttps.com0Y0...*.H.=....*.H.=....B....I...6.?.......).Q..&..%..I..\.....F8....*..gH.po._.cs......
b'..../0..+0...U...........0...U.%..0...+.........+.......0...U.......0.0...U......&.C.@..Ds.[}.y......0...U.#..0....+_.<!O....+,..
p...
02..+........&0$0"..+.....0...http://e5.i.lencr.org/0-..U...&0$..*.httpvshttps.com..httpvshttps.com0...U. ..0
0...g.....0-..U...&0$0". ....http://e5.c.lencr.org/10.crl0....
+.....y............u.
..0+.
.@b.      .U..Gt|...0..B..~N.4.......L.....F0D.  .....X.6....U.L.i......+.{..8S...  \..VQ.*'ep{.OK..ymJ.*.P..k.X..k.v..B..I`aT......z
-&EM.../.EY.'O:.T.............G0E. 2j.Y4..D..l..'}..{......*.....OD.!...q........"3.>....1.
...~..L..l0
..*.H.=....h.0e.0?.
_.......5.&..`...X...?.z...
0.p......%........1...t.9.{b...a.o.....`v.................H.;....f/..[0..W0..?..........lc..9.b.b.....0
.        *.H..
.....0O1.0        ..U....US1)0'..U.
. Internet Security Research Group1.0...U....ISRG Root X10..
240313000000Z.
270312235959Z021.0        ..U....US1.0...U.
.
Let's Encrypt1.0  ..U....E50v0...*.H.=....+...".b..
```

5 client pkts, 5 server pkts, 5 turns.

Entire conversation (3,960 bytes)   Show data as ASCII   Stream 0

D HTTPs packet trace

(a) **Is HTTP traffic there?** No, there is no HTTP Traffic (filter used: *http*)

(b) **Content transfer of HTTP and JavaScript files? and why?**: There is no content tranfer of HTTP and JavaScript. HTTPs is secured data transfer, therefor the file content is encrypted and hence not visible without the key to the file. This is the reason why wireshark does not show this content.

(c) **dns traffic**: Yes, DNS Traffic is present. DNS is required for look ups to convert web addresses to their correspoding IP Addresses. Once the ip addresses for the websites are resolved no more DNS Traffic is present for that particular domain name. Note: I was able to see DNS traffic because DoH, i.e., DNS over HTTPs is not enabled on my browser. Hence, DNS Traffic could not tunnel through HTTPs. (filter used: *dns*)

(d) **Number of tcp connections logged** = 7

(e) **Is Number of TCP Conbnections in HTTPS case == Number of TCP Connections in HTTP case?**: They are not equal. However the HTTPS case has smaller number of TCP Connections. A potential reason is multiplexing, allowing many HTTPS request to be sent simultaneously. Also, apart from TCP Handshake, https use TLS Handshake, which is expensive and hence tend to perform this Handshake lesser number of times. Due to this lesser number of TCP Connections are established.

2.2 Performance Analysis

E HTTP vs HTTPs

(a) **Comparision of time taken for downloads**: HTPP::17.132s and HTTPs::1.614 (93% faster than HTTP)

(b) **Observation from the plots**:

i. The download plot for https.pcap has significant throughput for download in comparision to that for http.pcap. Since the amount of data downloaded is the same, if the time of download reduces (as inferred from the time on the website and also the x axis of the plots), the throughput increases. The download throughput are upto 100 times more in https than http

ii. Time of downloads can also be justified from the RTT plot. RTT is in general, across all times on wall clock, smaller in https.pcap case than in comparision to http.pcap. This implies smaller time for data transfers from https.pcap than http.pcap

(c) **Assumption for RTT Plots**: We consider the completion of Round Trips for all those messages (with sequence number S and length L) that are stored in senders memory in list of pending ACKs from receiver, when an Acknowledgment Message is received from the receiver at the sender containing Acknowledgment Number A s.t., $A >= S + L$. This accounts for retransmission in all cases, either the packet getting lost when transferring upstream or downstream when receiving ACK.

(d) **Note**: RTT values are recorded at the time of ACK of the sent message

(e) **Note**: The images for throughputs have been found for each second 1, 2, 3, ... on the wall clock

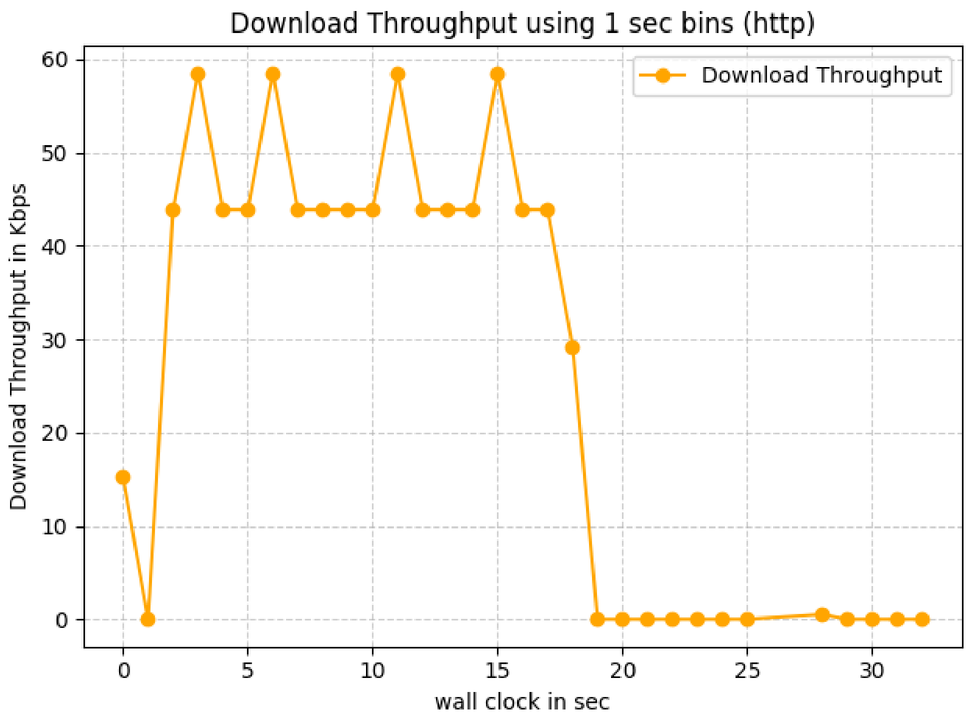


Figure 10: Download Throughput using http.pcap

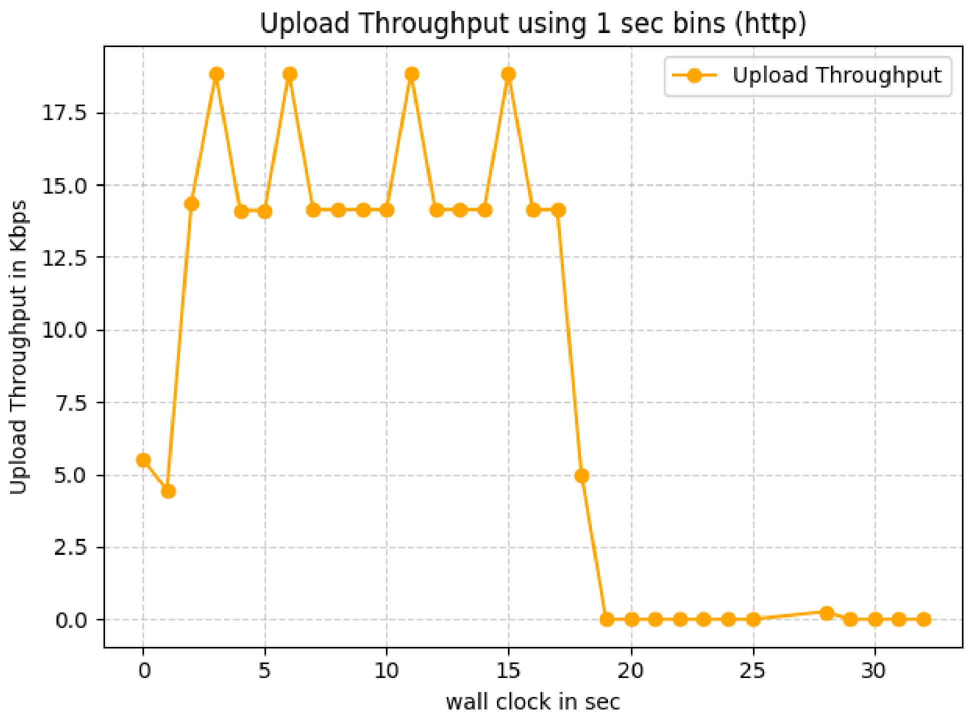


Figure 11: Upload Throughput using http.pcap

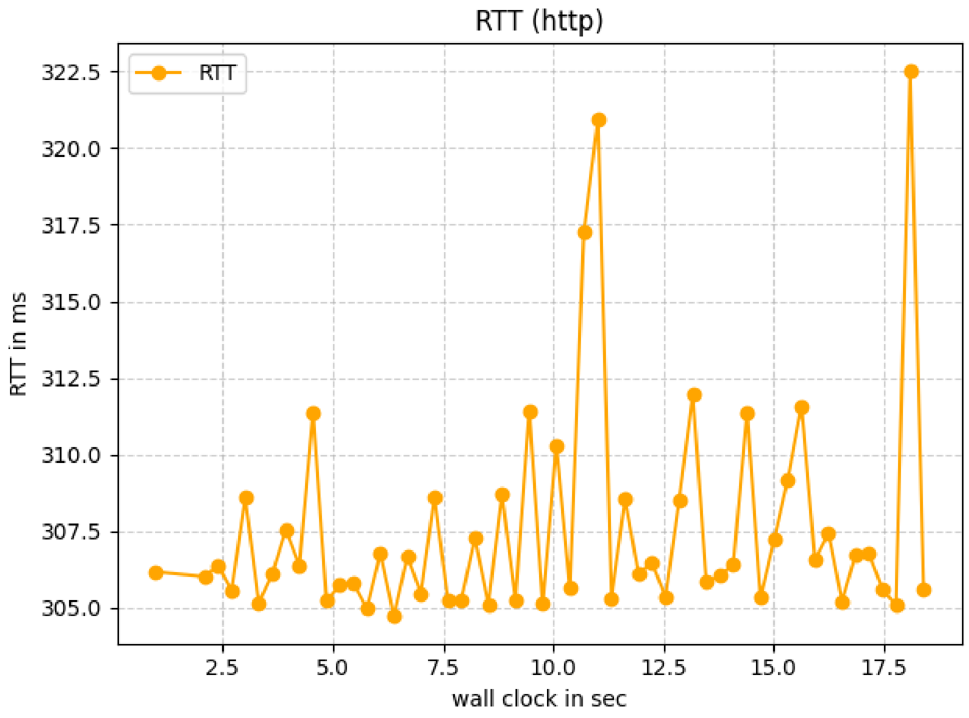


Figure 12: RTT using http.pcap

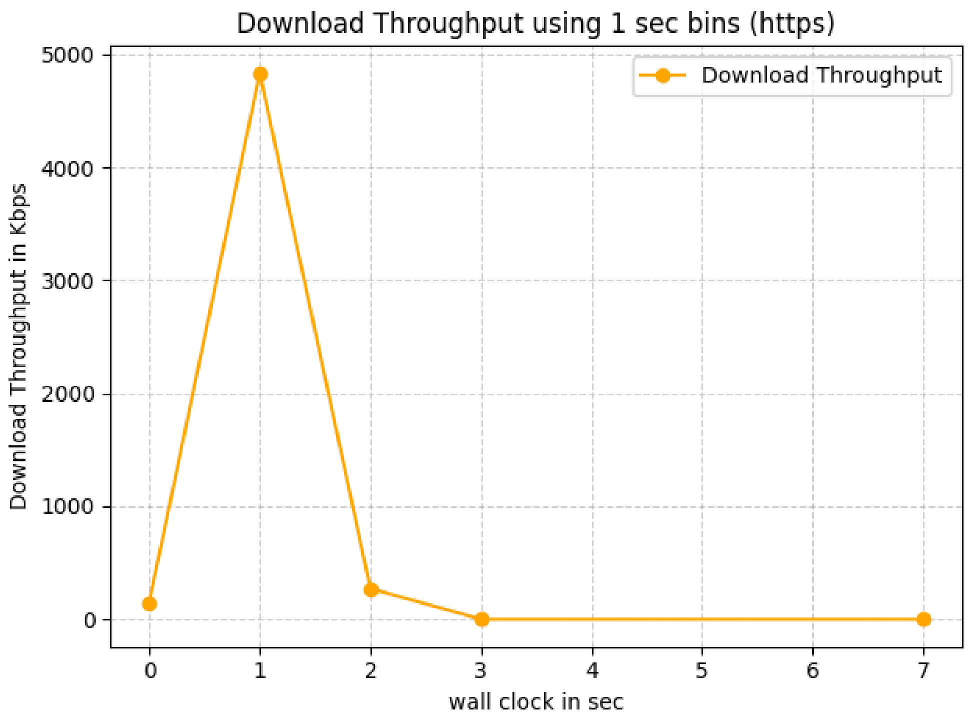


Figure 13: Download Throughput using https.pcap

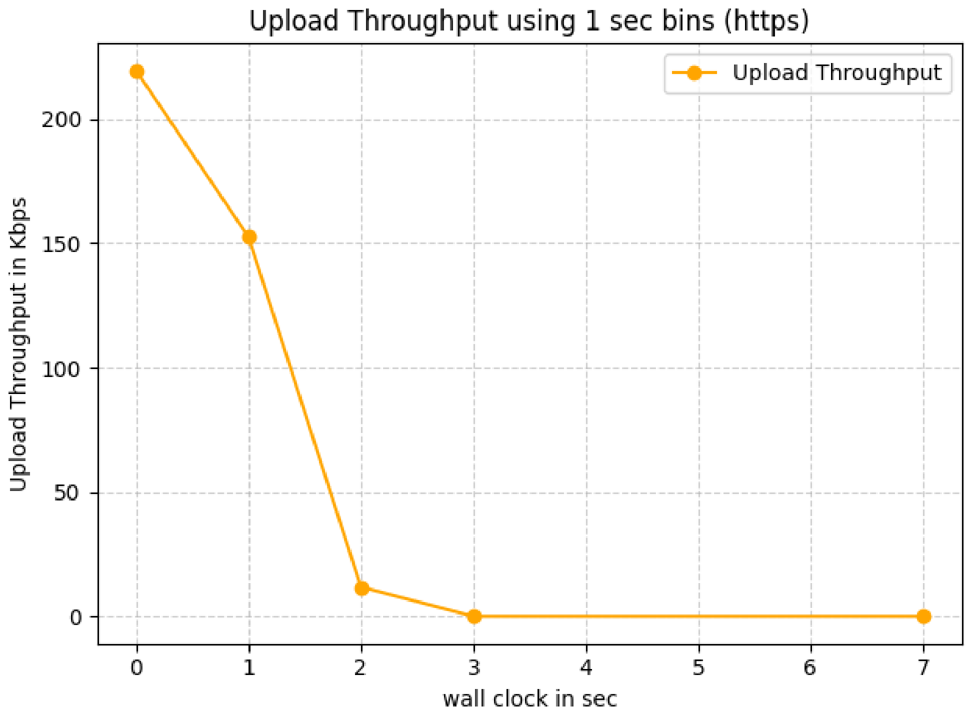


Figure 14: Upload Throughput using https.pcap

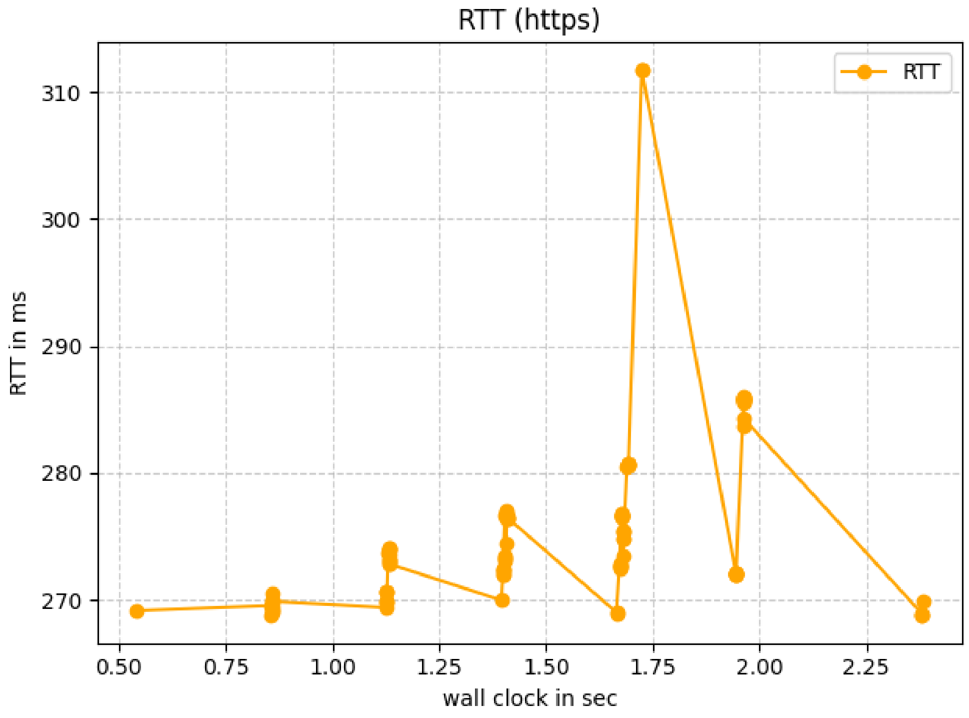


Figure 15: RTT using https.pcap