COL334 Assignment 1: Report

Aneeket Yadav

August 2024

This report describes the observations, insights and methods of analysis employed in network data collection, traffic analysis and measurement. See link to .pcapng file for Part-2.

1 Measurement Tools

1.1 Ping

A. Average ping latencies for `google.com` and `sigcomm.org` on IITD-wifi were found to be **87.327 ms** and **352.485 ms** respectively. On cellular hotspot network, they were **42.371 ms** and **564.346 ms** respectively.

- Difference between ping latencies of `google.com` and `sigcomm.org` on the same network may due to the following reasons:
  - Google being one of the largest CDNs, often has direct peering agreements with many ISPs and other networks, which can reduce the number of hops and improve routing efficiency compared to networks that do not have such agreements.
  - Google's CDN caches content close to users, reducing the distance data needs to travel.
  - Google's network infrastructure is designed to optimize routing paths and reduce latency which includes advanced routing algorithms and traffic management techniques.
- Difference between the average ping latency of the same website on IITD-wifi and cellular network is due to data packets may taking different routes to reach the destination, affecting latency. Fewer number of hops generally imply lower latency. As we will soon see using traceroute, IITD-wifi connection offers lesser number of hops both in case of both websites. In general, Wi-Fi networks are likely to have lower ping latency compared to mobile hotspots due to their stable infrastructure, dedicated local network, and fewer variations in signal quality

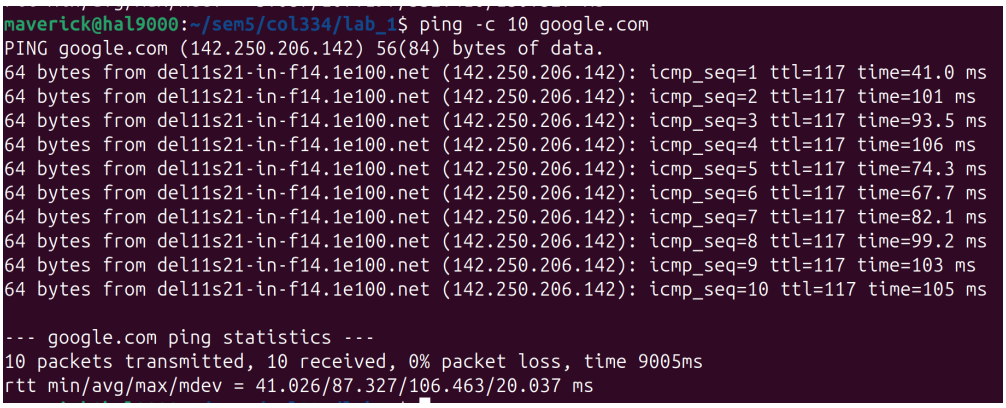

Figure 1: Ping results for google.com on IITD-wifi

```
maverick@hal9000:~/sem5/col334/lab_1/ss$ ping -c 10 sigcomm.org
PING sigcomm.org (190.92.158.4) 56(84) bytes of data.
64 bytes from server.hosting3.acm.org (190.92.158.4): icmp_seq=1 ttl=49 time=318 ms
64 bytes from server.hosting3.acm.org (190.92.158.4): icmp_seq=2 ttl=49 time=314 ms
64 bytes from server.hosting3.acm.org (190.92.158.4): icmp_seq=3 ttl=49 time=312 ms
64 bytes from server.hosting3.acm.org (190.92.158.4): icmp_seq=4 ttl=49 time=407 ms
64 bytes from server.hosting3.acm.org (190.92.158.4): icmp_seq=5 ttl=49 time=338 ms
64 bytes from server.hosting3.acm.org (190.92.158.4): icmp_seq=6 ttl=49 time=352 ms
64 bytes from server.hosting3.acm.org (190.92.158.4): icmp_seq=7 ttl=49 time=312 ms
64 bytes from server.hosting3.acm.org (190.92.158.4): icmp_seq=8 ttl=49 time=313 ms
64 bytes from server.hosting3.acm.org (190.92.158.4): icmp_seq=9 ttl=49 time=314 ms
64 bytes from server.hosting3.acm.org (190.92.158.4): icmp_seq=10 ttl=49 time=544 ms

--- sigcomm.org ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 311.703/352.485/544.295/69.913 ms
```

Figure 2: Ping results for sigcomm.org on IITD-wifi

```
maverick@hal9000:~/sem5/col334/lab_1/ss$ ping -c 10 sigcomm.org
PING sigcomm.org (64:ff9b::be5c:9e04) 56 data bytes
64 bytes from server.hosting3.acm.org (64:ff9b::be5c:9e04): icmp_seq=1 ttl=43 time=613 ms
64 bytes from server.hosting3.acm.org (64:ff9b::be5c:9e04): icmp_seq=2 ttl=43 time=612 ms
64 bytes from server.hosting3.acm.org (64:ff9b::be5c:9e04): icmp_seq=3 ttl=43 time=636 ms
64 bytes from server.hosting3.acm.org (64:ff9b::be5c:9e04): icmp_seq=4 ttl=43 time=455 ms
64 bytes from server.hosting3.acm.org (64:ff9b::be5c:9e04): icmp_seq=5 ttl=43 time=479 ms
64 bytes from server.hosting3.acm.org (64:ff9b::be5c:9e04): icmp_seq=6 ttl=43 time=400 ms
64 bytes from server.hosting3.acm.org (64:ff9b::be5c:9e04): icmp_seq=7 ttl=43 time=730 ms
64 bytes from server.hosting3.acm.org (64:ff9b::be5c:9e04): icmp_seq=8 ttl=43 time=754 ms
64 bytes from server.hosting3.acm.org (64:ff9b::be5c:9e04): icmp_seq=9 ttl=43 time=368 ms
64 bytes from server.hosting3.acm.org (64:ff9b::be5c:9e04): icmp_seq=10 ttl=43 time=596 ms

--- sigcomm.org ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9643ms
rtt min/avg/max/mdev = 368.355/564.346/753.934/125.977 ms
maverick@hal9000:~/sem5/col334/lab_1/ss$ ping -c 10 google.com
PING google.com (2404:6800:4002:821::200e) 56 data bytes
64 bytes from del12s04-in-x0e.1e100.net (2404:6800:4002:821::200e): icmp_seq=1 ttl=57 time=37.0 ms
64 bytes from del12s04-in-x0e.1e100.net (2404:6800:4002:821::200e): icmp_seq=2 ttl=57 time=34.9 ms
64 bytes from del12s04-in-x0e.1e100.net (2404:6800:4002:821::200e): icmp_seq=3 ttl=57 time=43.6 ms
64 bytes from del12s04-in-x0e.1e100.net (2404:6800:4002:821::200e): icmp_seq=4 ttl=57 time=42.3 ms
64 bytes from del12s04-in-x0e.1e100.net (2404:6800:4002:821::200e): icmp_seq=5 ttl=57 time=40.4 ms
64 bytes from del12s04-in-x0e.1e100.net (2404:6800:4002:821::200e): icmp_seq=6 ttl=57 time=38.7 ms
64 bytes from del12s04-in-x0e.1e100.net (2404:6800:4002:821::200e): icmp_seq=7 ttl=57 time=37.4 ms
64 bytes from del12s04-in-x0e.1e100.net (2404:6800:4002:821::200e): icmp_seq=8 ttl=57 time=76.7 ms
64 bytes from del12s04-in-x0e.1e100.net (2404:6800:4002:821::200e): icmp_seq=9 ttl=57 time=32.1 ms
64 bytes from del12s04-in-x0e.1e100.net (2404:6800:4002:821::200e): icmp_seq=10 ttl=57 time=40.6 ms

--- google.com ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9016ms
rtt min/avg/max/mdev = 32.060/42.371/76.687/11.894 ms
```

Figure 3: Ping results for google.com and sigcomm.org on mobile hotspot

B.
- The ping utility primarily uses the Internet Control Message Protocol (ICMP) to measure the round-trip time (RTT) for packets sent from the source to the destination and back. ICMP is part of the IP suite and operates on the network layer to for control and error detection. In the context of ping, the protocol operates as follows:
  - ICMP Echo Request sent to the target IP address.
  - The target system receives the Echo Request and responds with ICMP Echo Reply messages.
  - Time taken(RTT) for Echo Request to travel to the target and for the Echo Reply to return is noted.
  - Ping also provides information on packet loss and jitter (variability in response time) over successive pings.

- The theoretically maximum ping packet size will vary with IP protocol:

  IPv4 :
  IPv4 max packet size = 65535 bytes
  IP min header size = 20 bytes
  ICMP header size = 8 bytes
  Theoretical maximum = 65535 - 20 - 8 = 65507 bytes

  IPv6 :
  IPv6 max packet size = 65535 bytes
  IP header size(fixed) = 40 bytes
  ICMP header size = 8 bytes
  Theoretical maximum = 65535 - 40 - 8 = 65487 bytes

- Ping could not be performed at the theoretical upper limit. We need to consider the **Maximum Transmission Unit(MTU)**, which is a property of the network. Typically, routers, switches, NICs and wireless adapters have their default MTUs. Standard MTU for ethernet is 1500 bytes. However, other research networks may have much larger MTUs. For convenience, I define 'ping size' as the number I enter as '-s' parameter for a ping command. Considering a 20 byte long IPv4 header and 8 byte long ICMP header, this would allow maximum ping size to be 1472 bytes. On IITD-wifi, this was the limit beyond which packet loss occurred for `google.com`. However, the returned packet size was truncated to 76 bytes. Suppose we start pinging incrementally starting at a ping size of 1-byte, then returned packet is displayed excluding the IP header to be of size 9-bytes. Similarly, the returned packet is of size $x+8$ bytes for ping size of x bytes for $1 \leq x \leq 68$. On the other hand, when the ping routine was monitored using `wireshark`, something strange was noted. Suppose we set ping size = 3, then `wireshark` shows the size of the sent packet as 45 bytes. Along with the 11 byte ICMP message, it contains a 20-byte header and 14 additional bytes. In fact, this turns out to be the pattern for $1 \leq x \leq 1472$ i.e for $x = 1472$, a 1514 byte request is sent. It is not clear how this happens when we know MTU of ethernet to be 1500 bytes. The size of the received packet(as seen by `wireshark`) is same as size of sent packet for $1 \leq x \leq 68$. For $69 \leq x \leq 1472$, the size of the received packet(as seen by `wireshark`) is 110 bytes which though truncated, does not explain why we see 76 bytes received in the terminal. For $x > 1472$, the size of the sent packet(as seen by `wireshark`) is $(x - 1472 + 34)$ bytes and no reply is detected, i.e packet loss occurs. As for why truncation occurs $x > 68$ onwards, it may be that once you reach a size that is going to be fragmented, the remote host is replying with only what it can fixed amount of data. In view of the above discussion, I define max ping size as the value of ping size for which a reply is received albeit truncated. Therefore, the max ping size observed for `google.com` was 1472 on IITD-wifi and 1232 on mobile hotspot. For `sigcomm.org`, these figures were 35512 and 1452 respectively.

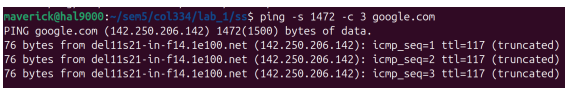

Figure 4: Max ping for google.com on IITD-wifi - IPv4

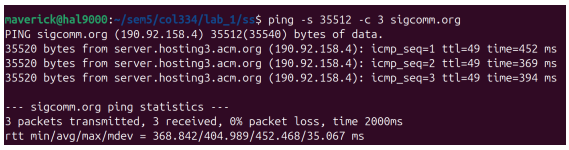

Figure 5: Max ping for sigcomm.org on IITD-wifi - IPv4

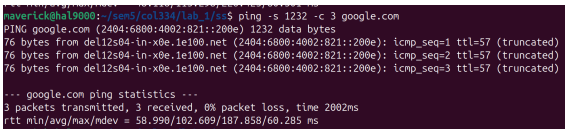

Figure 6: Max ping for google.com on mobile hotspot - IPv4

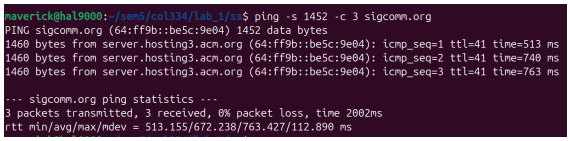

Figure 7: Max ping for sigcomm.org on mobile hotspot - IPv4

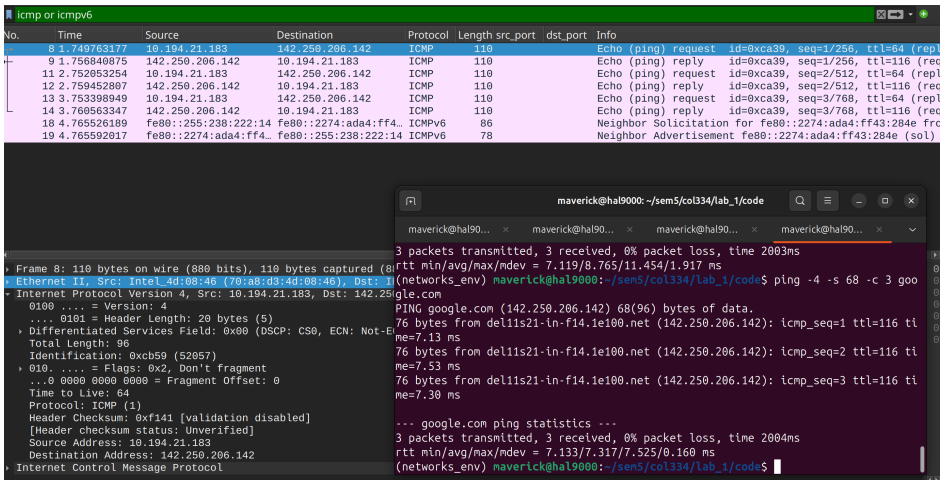

Figure 8: Wireshark results for google.com on IITD-wifi - IPv4, packet size = 68

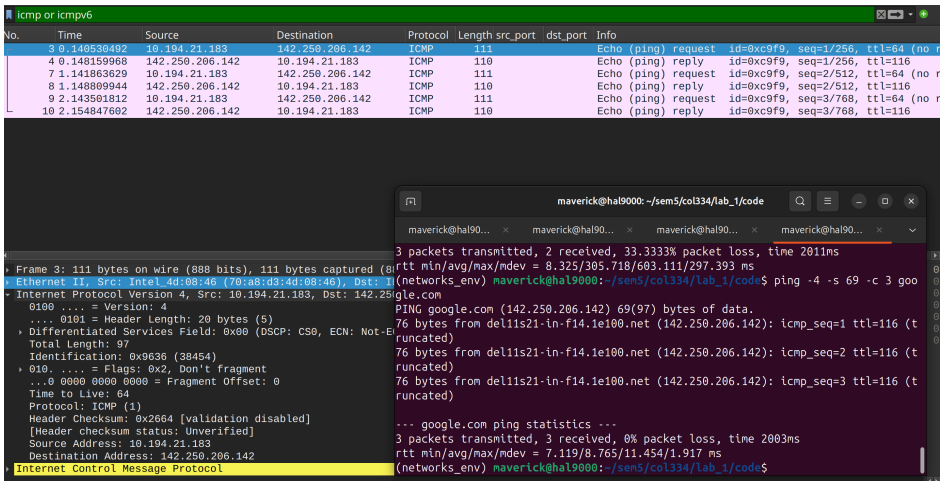

Figure 9: Wireshark results for google.com on IITD-wifi - IPv4, packet size = 69

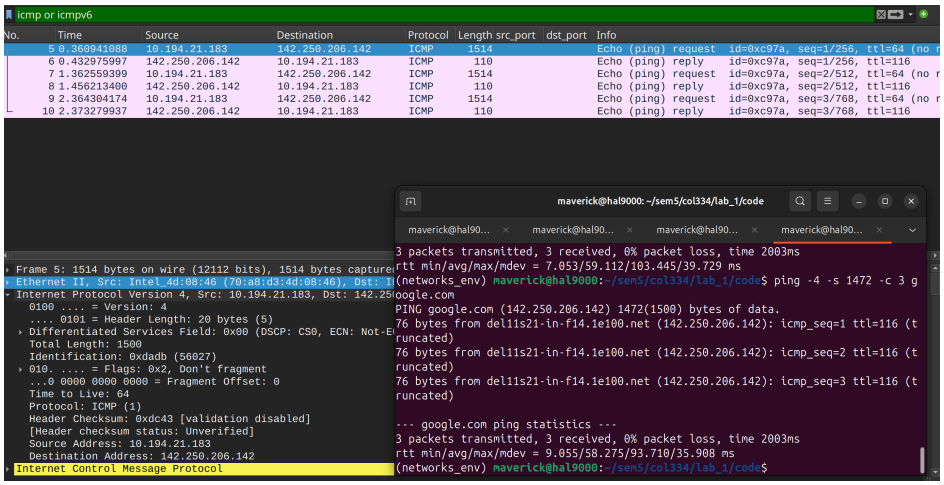

Figure 10: Wireshark results for google.com on IITD-wifi - IPv4, packet size = 1472

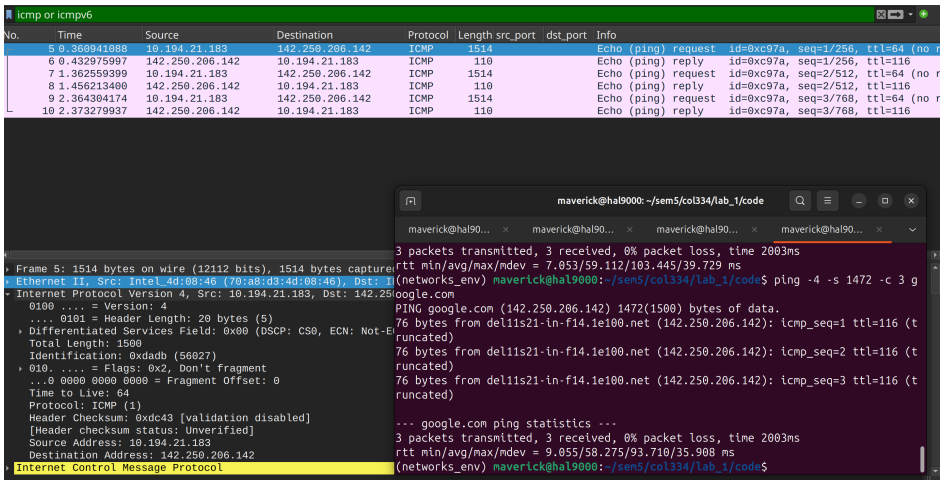

Figure 11: Wireshark results for google.com on IITD-wifi - IPv4, packet size = 1473

C.

- When we force ping to use IPv6, firstly in case of a successful ping, the length of the sent packet is 54 bytes larger than the ICMP encapsulation which itself is packet size plus 8 bytes.
- In case of `google.com` on IITD-wifi, we still observe that for $1 \leq x \leq 68$, the received packet is of same size as sent packet and ping is successful. For $69 \leq x \leq 1452$, ping is successful but length of received packet is less than that of sent packet. For $x > 1452$, length of sent packet(as seen by `wireshark`) is $(x - 1452 + 74)$ bytes and pings fails. Note that this similar to the practical ping limit based on MTU of ethernet, since header length of IPv6 is 40 bytes which is 20 bytes more than that of IPv4, hence 20 bytes lesser for max ping size. Similar outcomes are encountered with mobile data, though the max packet size is 1232.
- As for `sigcomm.org`, max packet size on cellular network was 1452 bytes. Effects similar to `google.com` were observed. However, ping failed on IITD-wifi providing the reason that address family for hostname is unsupported.

```
maverick@hal9000:~/sem5/col334/lab_1/ss$ ping -6 google.com
PING google.com (2404:6800:4002:82c::200e) 56 data bytes
^C
--- google.com ping statistics ---
53 packets transmitted, 0 received, 100% packet loss, time 53234ms
```

Figure 12: Max ping for google.com on IITD-wifi - IPv6

```
maverick@hal9000:~/sem5/col334/lab_1/ss$ ping -6 -s 10 -c 3 sigcomm.com
ping: sigcomm.com: Address family for hostname not supported
```

Figure 13: Failed ping for sigcomm.org on IITD-wifi - IPv6

```
maverick@hal9000:~/sem5/col334/lab_1/ss$ ping -6 -s 1232 -c 3 google.com
PING google.com (2404:6800:4002:82c::200e) 1232 data bytes
76 bytes from del11s21-in-x0e.1e100.net (2404:6800:4002:82c::200e): icmp_seq=1 ttl=57 (truncated)
76 bytes from del11s21-in-x0e.1e100.net (2404:6800:4002:82c::200e): icmp_seq=2 ttl=57 (truncated)
76 bytes from del11s21-in-x0e.1e100.net (2404:6800:4002:82c::200e): icmp_seq=3 ttl=57 (truncated)

--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 55.396/175.488/301.711/100.651 ms
```

Figure 14: Max ping for google.com on mobile hotspot - IPv6

```
maverick@hal9000:~/sem5/col334/lab_1/ss$ ping -6 -s 1452 -c 3 sigcomm.org
PING sigcomm.org (64:ff9b::be5c:9e04) 1452 data bytes
1460 bytes from server.hosting3.acm.org (64:ff9b::be5c:9e04): icmp_seq=1 ttl=41 time=1120 ms
1460 bytes from server.hosting3.acm.org (64:ff9b::be5c:9e04): icmp_seq=2 ttl=41 time=818 ms
1460 bytes from server.hosting3.acm.org (64:ff9b::be5c:9e04): icmp_seq=3 ttl=41 time=433 ms

--- sigcomm.org ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2042ms
rtt min/avg/max/mdev = 432.503/790.253/1119.845/281.311 ms, pipe 2
```

Figure 15: Max ping for sigcomm.org on mobile hotspot - IPv6

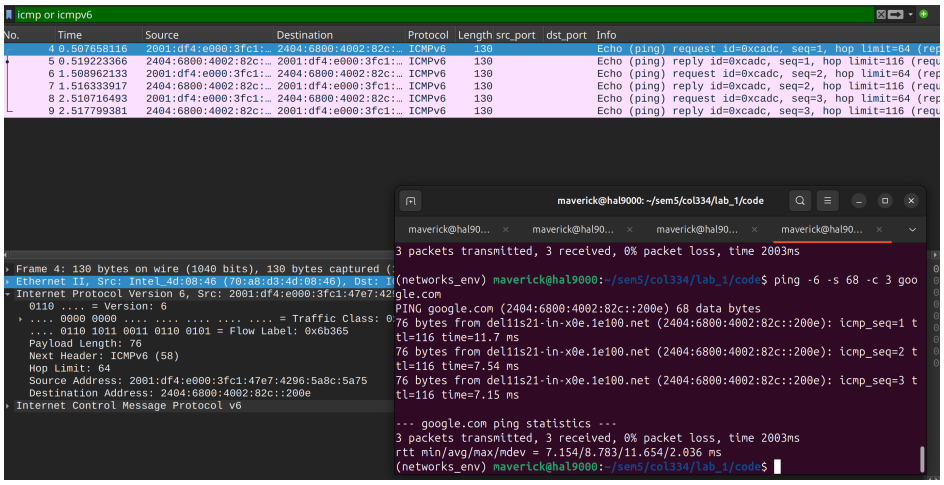

Figure 16: Wireshark results for google.com on IITD-wifi - IPv6, packet size = 68

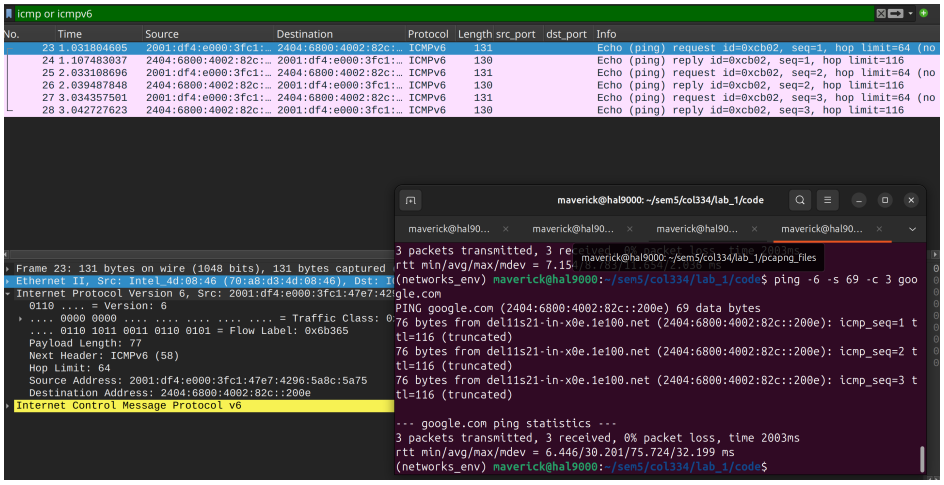

Figure 17: Wireshark results for google.com on IITD-wifi - IPv6, packet size = 69

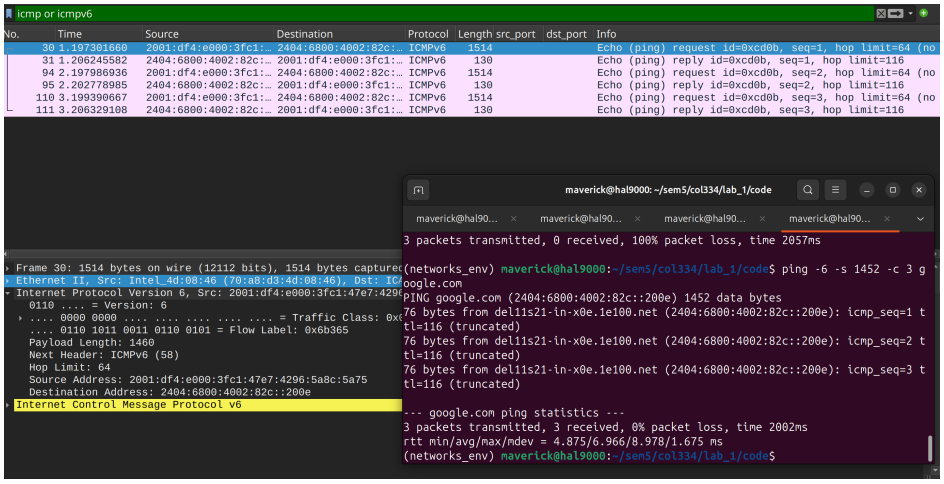

Figure 18: Wireshark results for google.com on IITD-wifi - IPv6, packet size = 1452

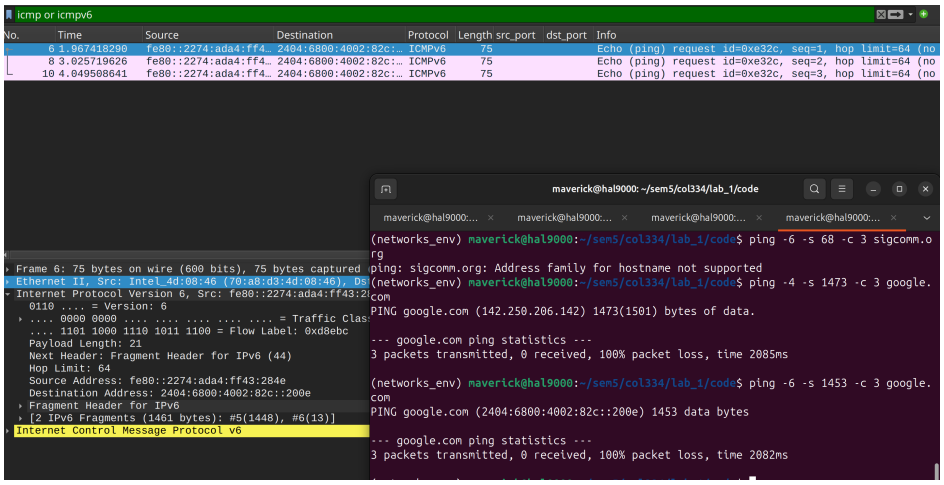

Figure 19: Wireshark results for google.com on IITD-wifi - IPv6, packet size = 1453

1.2 TraceRoute

The following traceroute results were obtained:

A. `google.com` experienced 9 hops on IITD-wifi and 15 on cellular network whereas `sigcomm.org` made 15 hops on IITD-wifi and 18 on cellular network. The list of autonomous systems detected and their ASNs were as follows:

- `google.com` on IITD-wifi: AS15169 Google LLC(hops 5-9)
- `sigcomm.org` on IITD-wifi: AS55836 Reliance Jio(hops 6,10), AS3356 Level 3 Parent LLC(hops 11-13), AS55293(hops 14-15)
- `google.com` on cellular network : AS15169 Google LLC(hops 10-15)
- `sigcomm.org` on cellular: AS64049 250 North Bridge Road, 16-01 Raffles City Tower Singapore (11-13), AS3356 Level 3 Parent LLC(hops 14-16), AS55293(hops 17-18)

B. The '*' character is generally observed in traceroute due to

- **Timeout**: The router or network device did not send a response back within the expected time frame. This could be due to network congestion, high latency, or the router being slow to respond.
- **Filtering or Blocking**: Some routers or firewalls are configured to block or drop ICMP (Internet Control Message Protocol) packets, which are used by traceroute to probe the route. These devices might be configured to ignore such probes to prevent network scanning or for security reasons.
- **Unreachable Network**(NA in this case): There could be a network issue causing the probe packets to be lost or the routers to be unreachable.

C. Did not observe multiple IP addresses for the same hop count anywhere. However, this may occur due the following reasons:

- **Load Balancing** : This means that different packets might take different routes, leading to different IP addresses appearing at the same hop count.
- **Anycast Routing** : This is a routing method where multiple servers share the same IP address, and the network directs packets to the nearest or best-performing server. For a particular hop, one might see different IPs if the packets are routed to different anycast nodes.

```
maverick@hal9000:~/sem5/col334/lab_1/ss$ traceroute google.com
traceroute to google.com (142.250.206.142), 64 hops max
  1   10.184.0.13  3.687ms  6.154ms  5.719ms
  2   10.255.107.3  7.351ms  9.533ms  7.883ms
  3   10.119.233.65  54.118ms  17.374ms  29.204ms
  4   *  *  *
  5   10.119.234.162  9.651ms  5.619ms  6.878ms
  6   72.14.194.160  6.434ms  6.666ms  8.015ms
  7   192.178.80.159  10.933ms  9.782ms  7.050ms
  8   142.251.76.199  7.413ms  7.368ms  7.673ms
  9   142.250.206.142  6.771ms  11.892ms  11.098ms
maverick@hal9000:~/sem5/col334/lab_1/ss$ traceroute sigcomm.org
traceroute to sigcomm.org (190.92.158.4), 64 hops max
  1   10.184.0.13  8.386ms  6.387ms  5.339ms
  2   10.255.107.3  7.332ms  5.973ms  6.204ms
  3   10.119.233.65  4.945ms  3.027ms  5.418ms
  4   *  *  *
  5   10.119.234.162  116.152ms  11.177ms  9.847ms
  6   136.232.148.177  9.841ms  33.909ms  8.969ms
  7   *  *  *
  8   *  *  *
  9   *  *  *
 10   49.45.4.103  317.952ms  303.809ms  305.728ms
 11   4.69.151.134  411.880ms  406.543ms  411.329ms
 12   4.69.202.222  407.566ms  419.526ms  399.224ms
 13   4.31.124.142  407.828ms  410.490ms  415.178ms
 14   69.48.136.9  412.961ms  523.574ms  386.054ms
 15   190.92.158.4  527.442ms  394.238ms  410.463ms
```

(a) Traceroutes on IITD-wifi

```
maverick@hal9000:~/sem5/col334/lab_1/ss$ traceroute google.com
traceroute to google.com (142.250.206.142), 64 hops max
  1   192.168.149.42  2.108ms  1.800ms  1.694ms
  2   255.0.0.0  50.477ms  39.497ms  39.555ms
  3   255.0.0.2  30.530ms  40.040ms  39.618ms
  4   255.0.0.3  38.952ms  40.212ms  39.818ms
  5   192.168.225.195  39.698ms  51.414ms  28.580ms
  6   192.168.216.30  40.215ms  192.168.216.28  39.713ms  39.832ms
  7   *  *  *
  8   *  *  *
  9   *  *  *
 10   74.125.48.196  32.173ms  37.644ms  39.883ms
 11   142.251.54.117  41.944ms  39.703ms  89.821ms
 12   142.251.76.197  28.264ms  39.629ms  39.878ms
 13   142.251.76.199  41.649ms  29.491ms  49.745ms
 14   142.251.255.57  47.972ms  39.617ms  50.702ms
 15   142.250.206.142  39.100ms  50.969ms  42.333ms
maverick@hal9000:~/sem5/col334/lab_1/ss$ traceroute sigcomm.org
traceroute to sigcomm.org (190.92.158.4), 64 hops max
  1   192.168.149.42  2.720ms  1.428ms  1.859ms
  2   255.0.0.0  76.556ms  24.405ms  49.897ms
  3   255.0.0.2  30.261ms  53.721ms  25.934ms
  4   255.0.0.3  48.937ms  39.655ms  30.512ms
  5   192.168.225.194  29.255ms  39.492ms  30.976ms
  6   192.168.216.24  44.228ms  192.168.216.26  33.821ms  192.168.216.24  44.702ms
  7   *  *  *
  8   *  *  *
  9   *  *  *
 10   *  *  *
 11   103.198.140.64  154.041ms  82.470ms  67.074ms
 12   103.198.140.207  314.244ms  307.034ms  307.065ms
 13   103.198.140.207  307.027ms  307.407ms  306.841ms
 14   4.69.202.222  307.078ms  409.707ms  306.910ms
 15   4.69.202.222  410.036ms  614.186ms  306.847ms
 16   4.31.124.142  408.927ms  409.346ms  409.834ms
 17   69.48.136.9  521.174ms  399.899ms  331.511ms
 18   190.92.158.4  332.360ms  768.889ms  408.903ms
```

(b) Traceroutes on mobile hotspot

D. The required IP address is 10.184.0.13. When trying to ping this IP using cellular network, the process keeps waiting indefinitely and thus had to be killed. No reply packets were received.

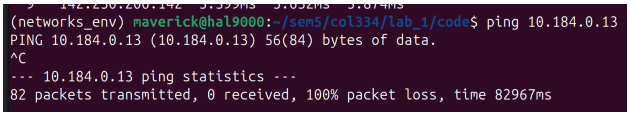

Figure 21: Failed ping from cellular network at 1st hop of IITD-wifi

E.
- Traceroute to `google.com` on IITD-wifi reveals a 2-tier architecture. The local Tier 3 might has connected directly to Tier 1(Google) i.e IITD may have direct agreements with major ISPs, bypassing Tier 2. In this case, the absence of multiple Tier 2 hops indicates efficient routing through direct peering agreements.
- Traceroute to `google.com` on cellular network may have a tier-2 or tier-3 architecture depending on what jobs 7-9 represent. Before hop 7, no autonomous system is found and just after hop 9, 74.125.48.196 is owned by Google LLC.
- Traceroute to `sigcomm.org` on IITD-wifi reveals a 3-tier architecture which is evident from the discussion of autonomous systems on this route earlier.
- Traceroute to `sigcomm.org` on cellular network shows a 3-tier architecure. See autonomous systems

F. The results of geo-location of some IPs observed in the traceroute have been listed on the next page.

IP Address	Location	Network	Postal Code	Approximate Latitude / Longitude*, and Accuracy Radius	ISP / Organization
10.184.0.13	⚠ The IP address '10.184.0.13' is a reserved IP address (private, multicast, etc.).				
10.255.107.3	⚠ The IP address '10.255.107.3' is a reserved IP address (private, multicast, etc.).				
10.119.233.65	⚠ The IP address '10.119.233.65' is a reserved IP address (private, multicast, etc.).				
10.119.234.162	⚠ The IP address '10.119.234.162' is a reserved IP address (private, multicast, etc.).				
72.14.195.56	United States (US), North America	72.14.194.0/23	-	37.751, -97.822 (1000 km)	Google
142.251.54.111	United States (US), North America	142.251.48.0/20	-	37.751, -97.822 (1000 km)	Google
142.251.76.197	United States (US), North America	142.251.76.0/22	-	37.751, -97.822 (1000 km)	Google
142.250.206.142	Florida, United States (US), North America	142.250.206.0/23	-	28.6344, -81.6221 (1000 km)	Google Servers
136.232.148.177	New Delhi, National Capital Territory of Delhi, India (IN), Asia	136.232.148.0/22	110043	28.652, 77.1663 (5 km)	Jio
49.45.4.103	New Delhi, National Capital Territory of Delhi, India (IN), Asia	49.45.4.0/23	110043	28.652, 77.1663 (1000 km)	Jio, Reliance Jio Infocomm Pte Singapore
4.7.26.61	San Bernardino, California, United States (US), North America	4.7.26.0/24	92407	34.2098, -117.3997 (20 km)	Lumen
4.69.202.222	United States (US), North America	4.69.200.0/22	-	37.751, -97.822 (1000 km)	Lumen
69.48.136.9	United States (US), North America	69.48.136.0/23	-	37.751, -97.822 (1000 km)	A2 Hosting
190.92.158.4	Michigan, United States (US), North America	190.92.152.0/21	-	42.4652, -83.3713 (1000 km)	A2 Hosting

Figure 22: Maxmind geolocation results for some observed IPs

2 Network Data Collection and Header Analysis

The experiment was conducted in two configurations : app-to-app and website-to-website. This allowed us to gain multiple insights when analysing the traffic. In the app-to-app scenario, the operating systems were MacOS and Windows 11 while in website-to-website, Windows 11 was replaced by Ubuntu 24.04.

A. **Protocol Hierarchy**

- Website-to-website : Based on the network traffic summary shown in the image, the following protocols are likely used by a Teams call, along with their packet counts and percentages:
  (a) Network Layer
    - **IPv6:** 124,095 packets (99.0%)
    - **IPv4:** 1,107 packets (0.9%)

  (b) Transport Layer
    - **UDP (User Datagram Protocol):** 124,111 packets (99.1%)
      * IPv6 UDP: 123,877 packets (98.9%)
      * IPv4 UDP: 234 packets (0.2%)
    - **TCP (Transmission Control Protocol):** 979 packets (0.8%)
      * IPv6 TCP: 116 packets (0.1%)
      * IPv4 TCP: 863 packets (0.7%)

  (c) Application Layer
    - **SRTP (Secure Real-time Transport Protocol):** 2,793 packets (2.2%)
    - **QUIC IETF:** 30 packets (0.0%)
    - **MDNS (Multicast Domain Name System):** 12 packets (0.0%)
    - **STUN (Session Traversal Utilities for NAT):** 296 packets (0.2%)
      * IPv6 STUN: 2 packets (0.0%)
      * IPv4 STUN: 2 packets (0.0%)
      * Within SRTP: 292 packets (0.2%)
    - **DHCPv6:** 4 packets (0.0%)
    - **IGMP (Internet Group Management Protocol):** 8 packets (0.0%)
    - **HTTP (Hypertext Transfer Protocol):** 3 packets (0.0%)
      * IPv6 HTTP: 1 packet (0.0%)
      * IPv4 HTTP: 2 packets (0.0%)

  (d) Analysis
  The majority of the traffic appears to be UDP-based, which is typical for real-time communications like video calls. SRTP is commonly used for securing voice and video communications, which aligns with a Teams call. The presence of STUN suggests NAT traversal techniques are being used, which is common for peer-to-peer communications in applications like Teams. IPv6 technology is being predominantly used by default.

- App-to-app : Based on the network traffic summary shown in the image, the following protocols are likely used by a Teams call, along with their packet counts and percentages:
  (a) Network Layer
    - **IPv6:** 519 packets (1.1%)
    - **IPv4:** 45,655 packets (98.8%)

  (b) Transport Layer
    - **UDP (User Datagram Protocol):** 124,111 packets (99.1%)
      * IPv6 UDP: 62 packets (0.1%)
      * IPv4 UDP: 45225 packets (97.9%)
    - **TCP (Transmission Control Protocol):** 979 packets (0.8%)

    - IPv6 TCP: 446 packets (1.0%)
    - IPv4 TCP: 417 packets (0.9%)
  (c) Application Layer
    - **MDNS (Multicast Domain Name System):** 12 packets (0.0%)
    - **IPv4 STUN (Session Traversal Utilities for NAT):** 26011 packets (56.3%)
    - **IGMP (Internet Group Management Protocol):** 13 packets (0.0%)
    - **RTP (Real Time Transport Control Protocol):** 335 packets (0.7%)
  (d) Analysis The majority of the traffic appears to be UDP-based, which is typical for real-time communications like video calls, a large part of which is STUN(which being an application layer protocol is not being utilised in the website-to-website case since an application layer is not necessarily needed there) .IPv4 technology is being predominantly used by default.

B. **Endpoints** :

- App-to-app : No direct connection between the hosts was observed during the call.
  Endpoint for Sanyam: IP (192.168.28.83) and network (Home Wi-Fi)
  Endpoint for Aneeket: IP (192.168.29.51) and network (Cellular data hotspot)
  They are different endpoints. In the Teams call, both participants had private IP addresses indicating they were behind NAT devices. This prevented direct P2P connections. However, they were both communicating with the same Microsoft server (IP 4.190.207.114). MS Teams uses NAT traversal techniques like STUN (for discovering public IPs) and TURN (for relaying media) to facilitate communication. Relay TURN servers are employed to manage and route traffic between participants. This ensures connectivity even when direct connections are blocked by NAT and firewall restrictions.
- Website-to-website :
  Endpoint for Sanyam: IP (2405:201:4005:b0ab:444:a366:80b3:b95a) and network (Sanyam's Home Wi-Fi)
  Endpoint for Aneeket: IP (2405:201:4005:b0ab:2e0:c67b:e3a9:19e7) and network (Sanyam's Home Wi-Fi)
  In this case, a direct connection between the two endpoints was observed which was established probably for performance and connectivity reasons.

C. **Audio and Video packets**

- The criteria used to identify audio and video packets consists of identifying the right ports and packet lengths. Typically, video requires much greater amount of data to be transferred per unit time, which is why its packets are more frequent and larger(average packet size about 1000 bytes and 8-10 packets transferred in one go). Audio on the contrary is comparitively lighter(with average packets size about 150 bytes and one packet transferred in one go).
- On the app, the user irrespective of whether on sending or receiving end, uses a port in the range 50000-50019 for audio packets and 50020-50039 for video transfer. This can also be found on Microsoft's forum. MS Teams's UDP port for audio/video sending/receiving remained fixed at 3480. During the call, Sanyam's audio port was 50015 and video port was 50030. But for Aneeket, audio and video ports were same : 57946(the reason for the same is not clear).
- However, this differed for website-to-website communication where the same port was assigned for both audio and video packet transfer. Sanyam's assigned UDP port is 61080 and Aneeket's port is 58656.

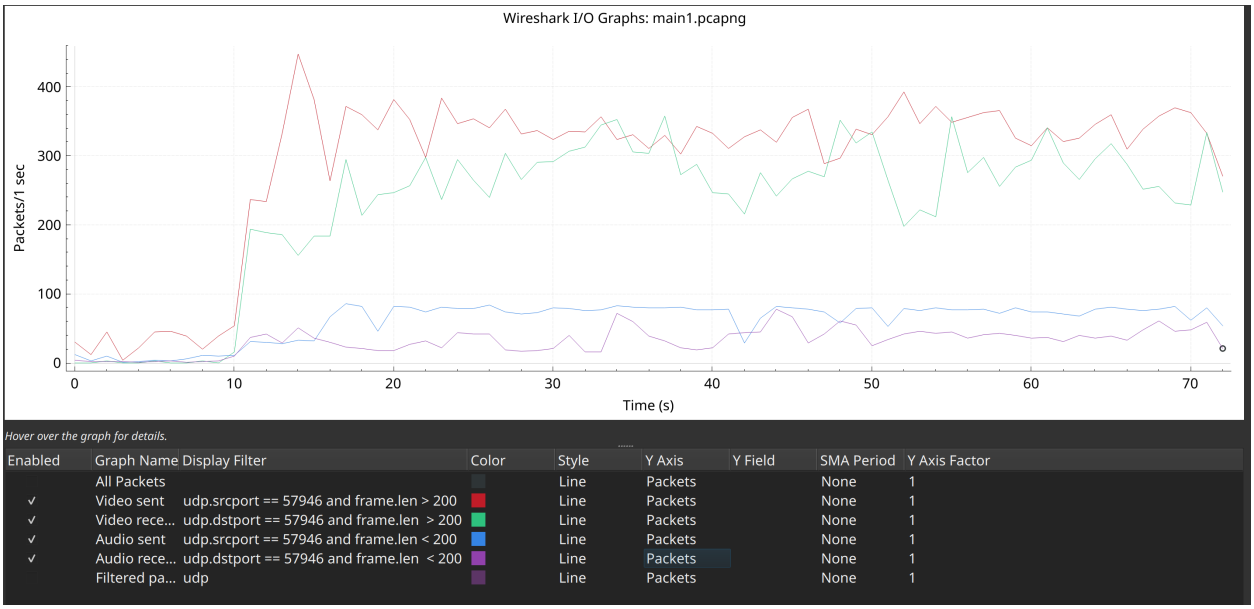

Figure 23: Traffic at Aneeket's end in app-to-app mode in packets/sec

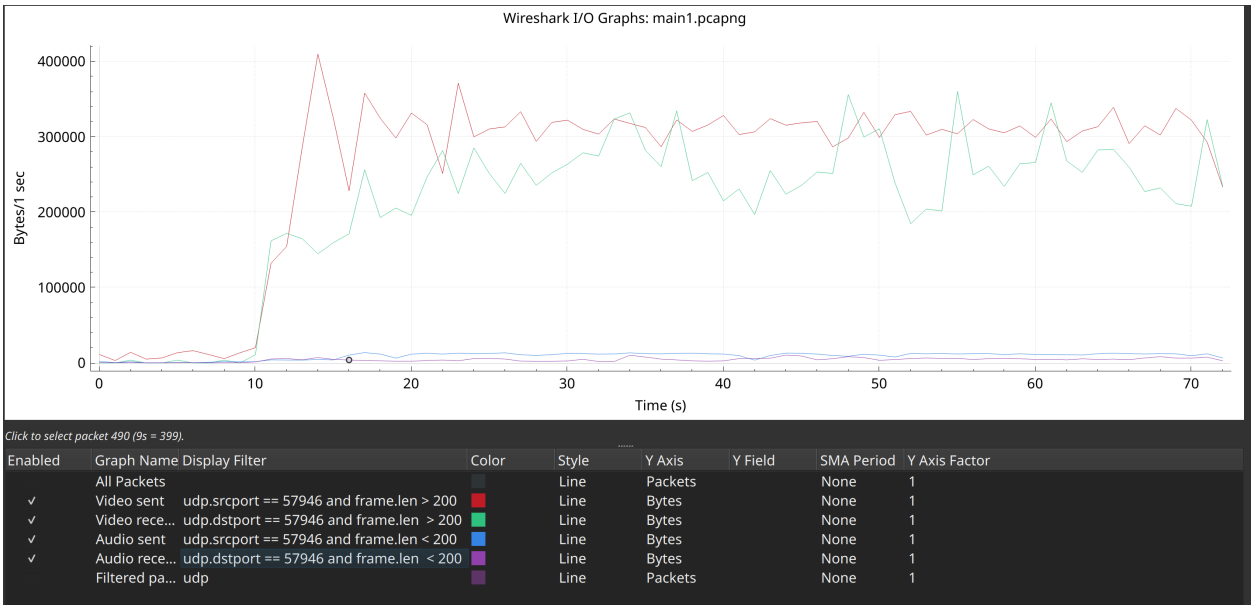

Figure 24: Traffic at Aneeket's end in app-to-app mode in bytes/sec

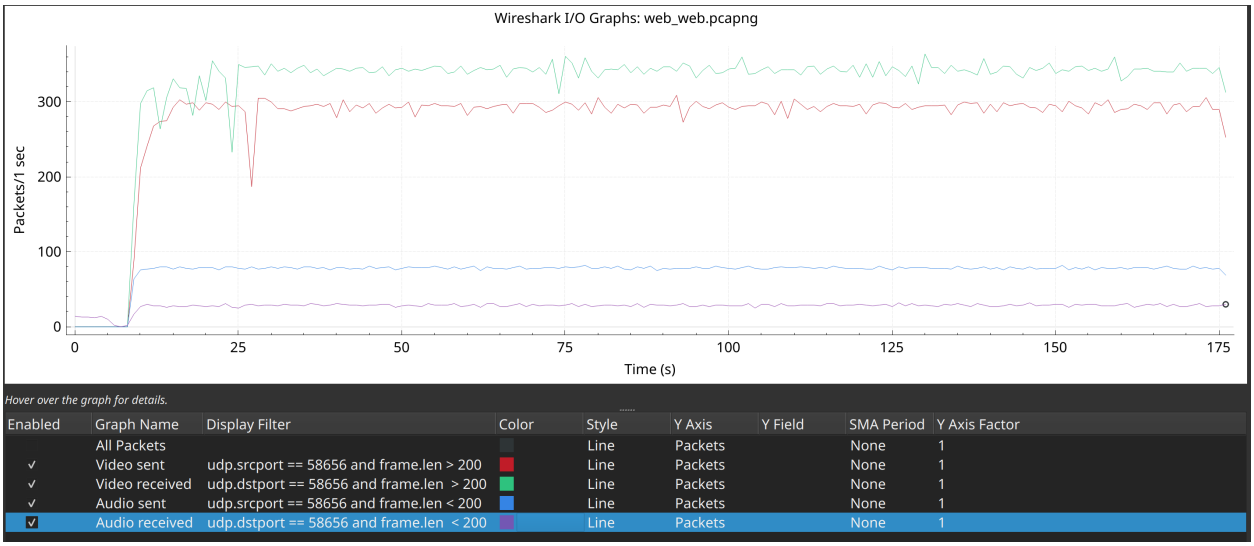

Figure 25: Traffic at Aneeket's end in website-to-website mode in packets/sec

Figure 26: Traffic at Aneeket's end in website-to-website mode in bytes/sec

3 Traffic Analysis and Network Performance

3.1 Logic/Algorithm

1. Load the `.pcapng` file and parse the packets.
2. Initialize an empty list `handshakes`.
3. For each packet in the `.pcapng` file:
   (a) Check if the packet is a TLS handshake.

(b) If the handshake type is Client Hello and the information contains `"ndt7-mlab"`:
    i. Append the packet to the `handshakes` list.

4. Verify that there are atleast two handshakes. If not, exit with error.
5. Identify handshakes that are more than 5 seconds apart:
   (a) For each pair of consecutive handshakes in the list:
      i. Calculate the time difference between the handshakes.
      ii. If the time difference is greater than 5 seconds:
         A. Mark the first handshake as `handshake_1`.
         B. Mark the last handshake in the pair as `handshake_2`.
6. Collect all TCP packets between `handshake_1` and `handshake_2`.
7. Determine the client and server IPs:
   (a) Extract the source IP from `handshake_1` and set it as `client`.
   (b) Extract the destination IP from `handshake_1` and set it as `server`.
8. Count the number of TCP packets from server to client and client to server:
   (a) Initialize counters `server_to_client_count` and `client_to_server_count` to 0.
   (b) For each TCP packet between `handshake_1` and `handshake_2`(practically, we can optimise this step by considering the traffic in both directions for 0.5 sec only):
      i. If the packet is from `server` to `client`, increment `server_to_client_count`.
      ii. If the packet is from `client` to `server`, increment `client_to_server_count`.
9. Classify `handshake_1` as download or upload handshake:
   (a) If `server_to_client_count` > `client_to_server_count`, classify `handshake_1` as a download handshake.
   (b) Otherwise, classify it as an upload handshake.
10. Find the last TCP packet from server to client that has `tcp.flags.fin == 1` after `handshake_2`, call it `close_packet`
11. Compute average download speed or upload speed:
   (a) If `handshake_1` is a download handshake:
      i. Average download speed = $\frac{\text{Server to client traffic between } \texttt{handshake\_1} \text{ and } \texttt{handshake\_2}}{\text{time difference between } \texttt{handshake\_1} \text{ and } \texttt{handshake\_2}}$.
      ii. Average upload speed = $\frac{\text{Client to server traffic between } \texttt{handshake\_2} \text{ and } \texttt{close\_packet}}{\text{time difference between } \texttt{handshake\_2} \text{ and } \texttt{close\_packet}}$.
   (b) If `handshake_1` is an upload handshake:
      i. Average upload speed = $\frac{\text{Client to server traffic between } \texttt{handshake\_1} \text{ and } \texttt{handshake\_2}}{\text{time difference between } \texttt{handshake\_1} \text{ and } \texttt{handshake\_2}}$.
      ii. Average download speed = $\frac{\text{Server to client traffic between } \texttt{handshake\_2} \text{ and } \texttt{close\_packet}}{\text{time difference between } \texttt{handshake\_2} \text{ and } \texttt{close\_packet}}$.
12. Output the calculated speeds and any other relevant information.

3.2 Results

- Speed test traffic percentage = **65.42369839787821%**
- Average Download throughput = **29.5236 Mbps**
- Average Upload throughput = **8.5179 Mbps**

3.3 Graphs

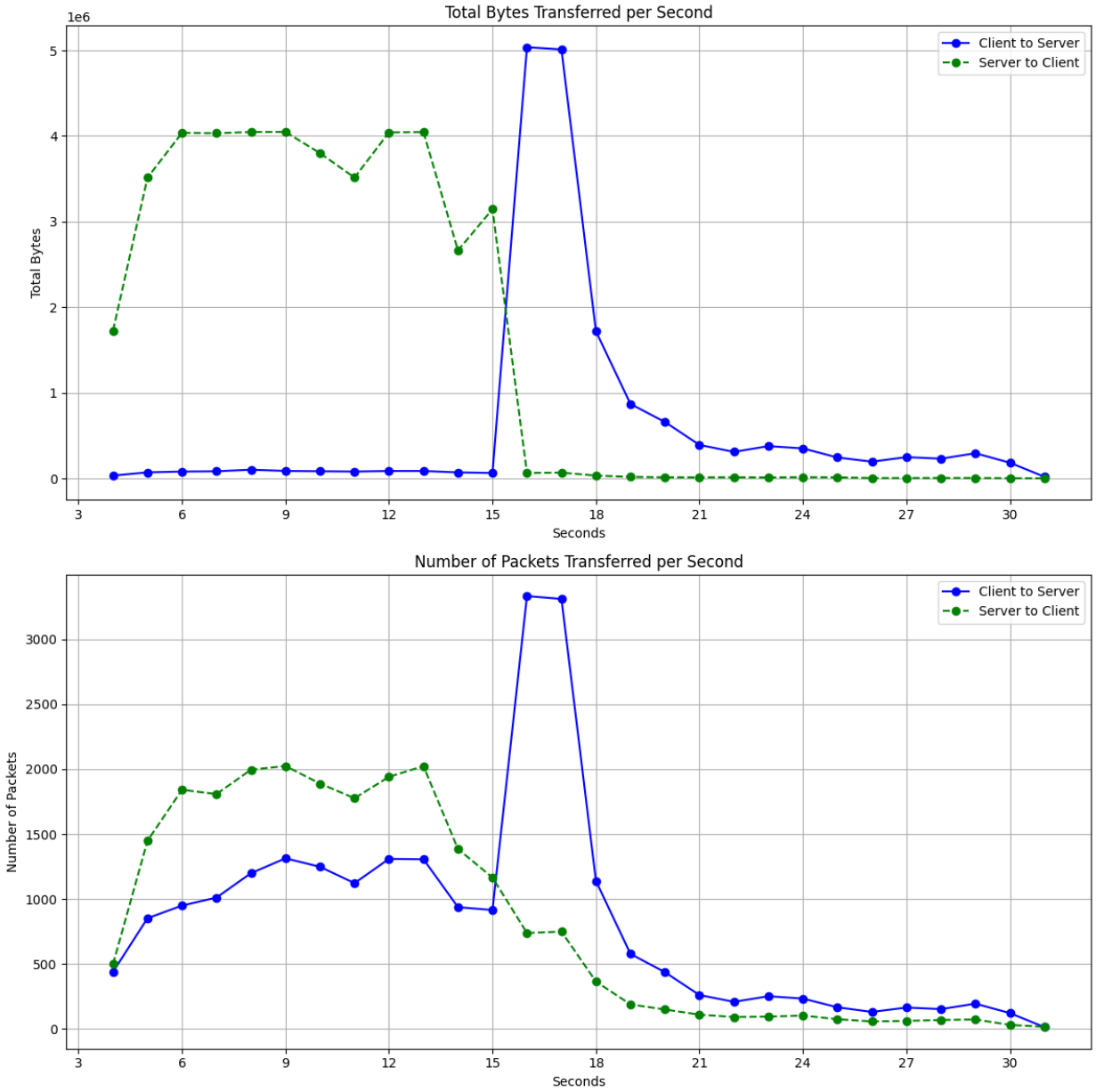

Figure 27: Client to Server and Server to Client traffic observed during the session