CS-359 Assignment-11

Name: Ammaar Ahmad

Roll No.: 1801CS08

In this assignment, we experimented with various statistical values which related to the performance of Internet Protocols like Throughput, Round trip time, Packet Size, Number of packets lost, Number of TCP, UDP Packets, Number of responses with respect to requests etc.

For this purpose, we captured data packets from and to facebook.com at two different parts of the day. And on this data, we applied various functionalities of Wireshark to analyze performance.

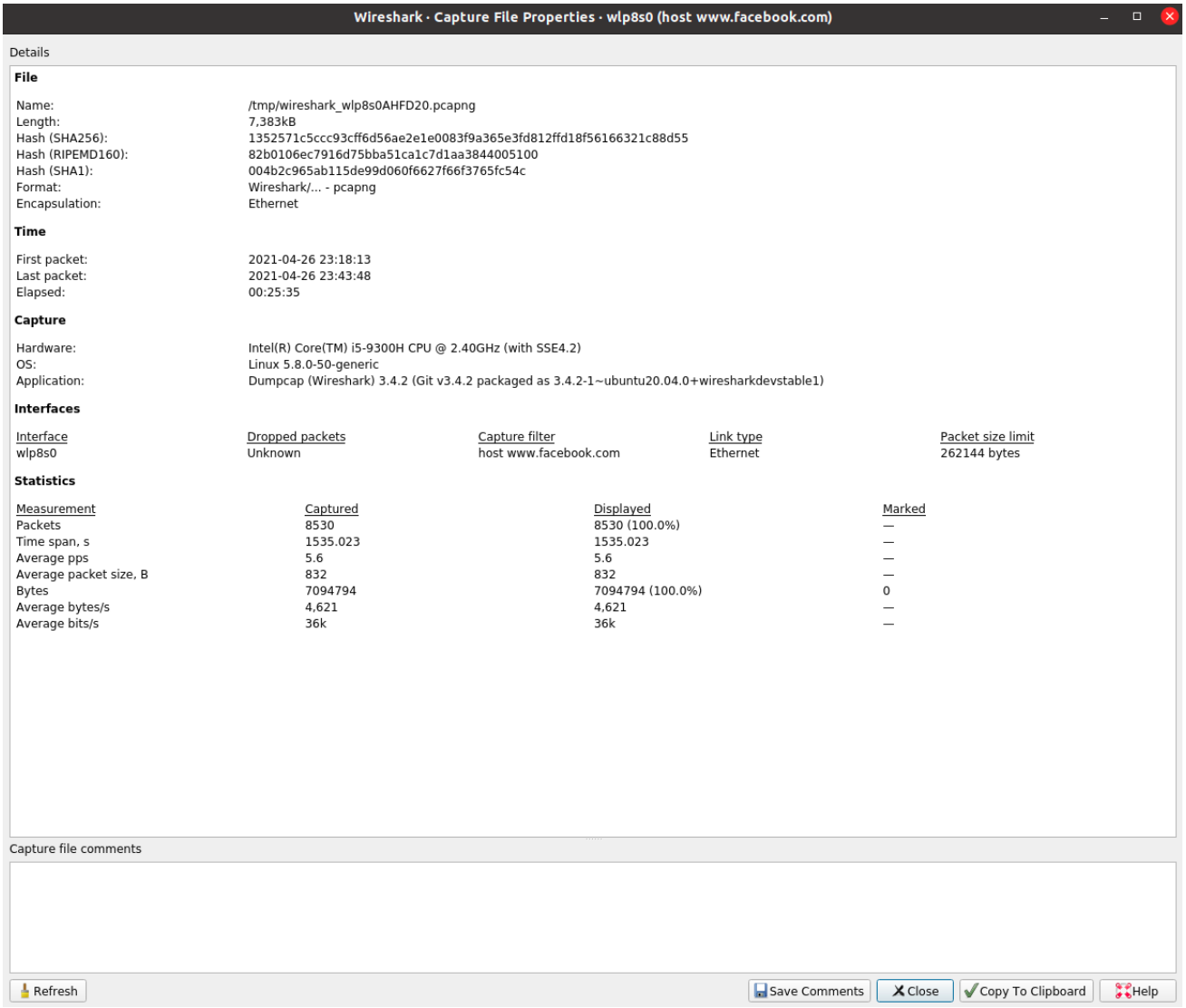
Capturing Packets

1. Capture Filter: **host www.facebook.com** and then start capturing
7. Open a tab and go to facebook.com and perform actions until at least 2000 packets.

Throughput

Throughput tells you how much data was transferred from a source at any given time and bandwidth tells you how much data could theoretically be transferred from a source at any given time.

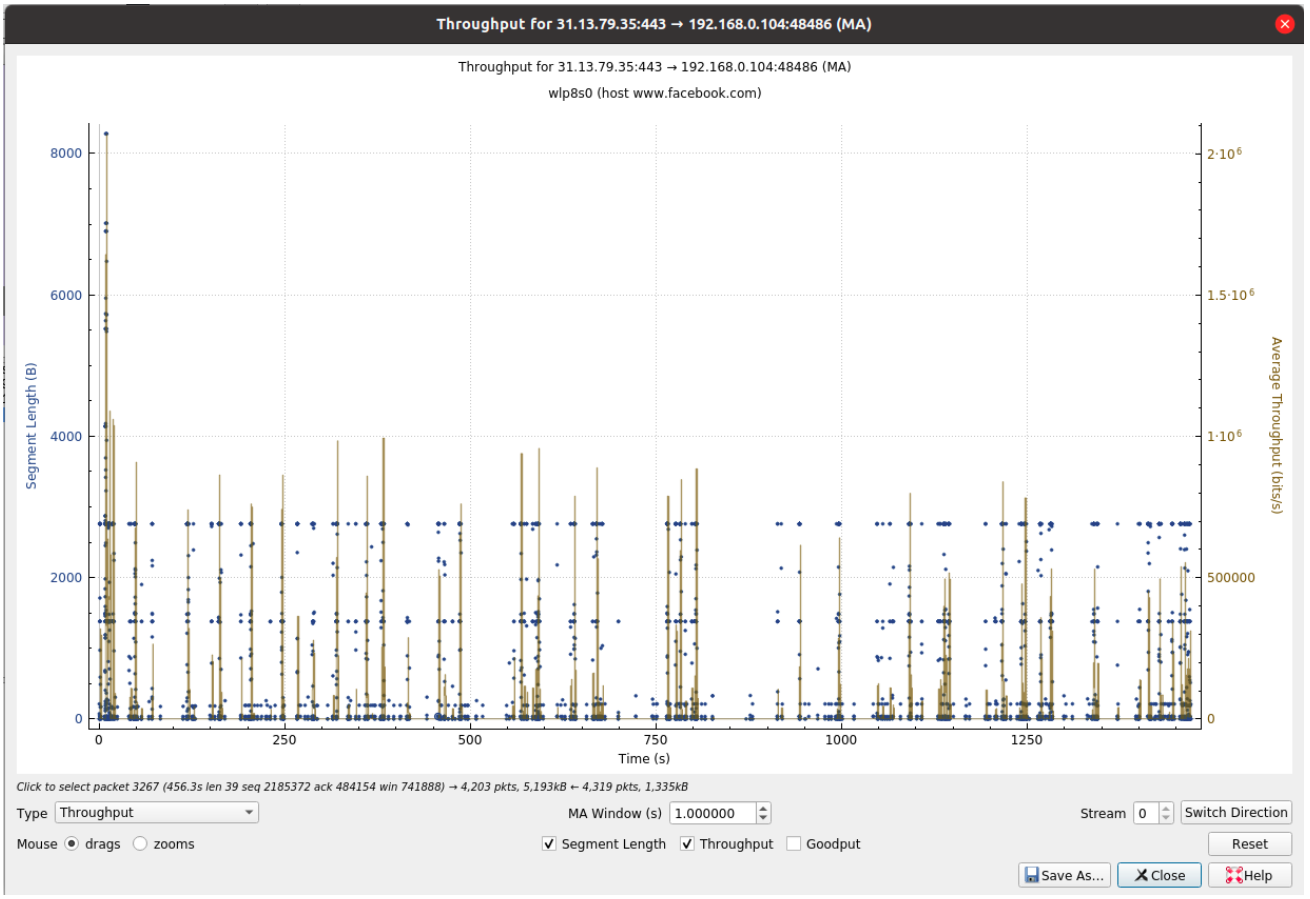
We can measure average throughput and graphical plot of throughputs of various packets using Wireshark
For summary of transfers, got to **“Statistics -> Capture File Properties”**



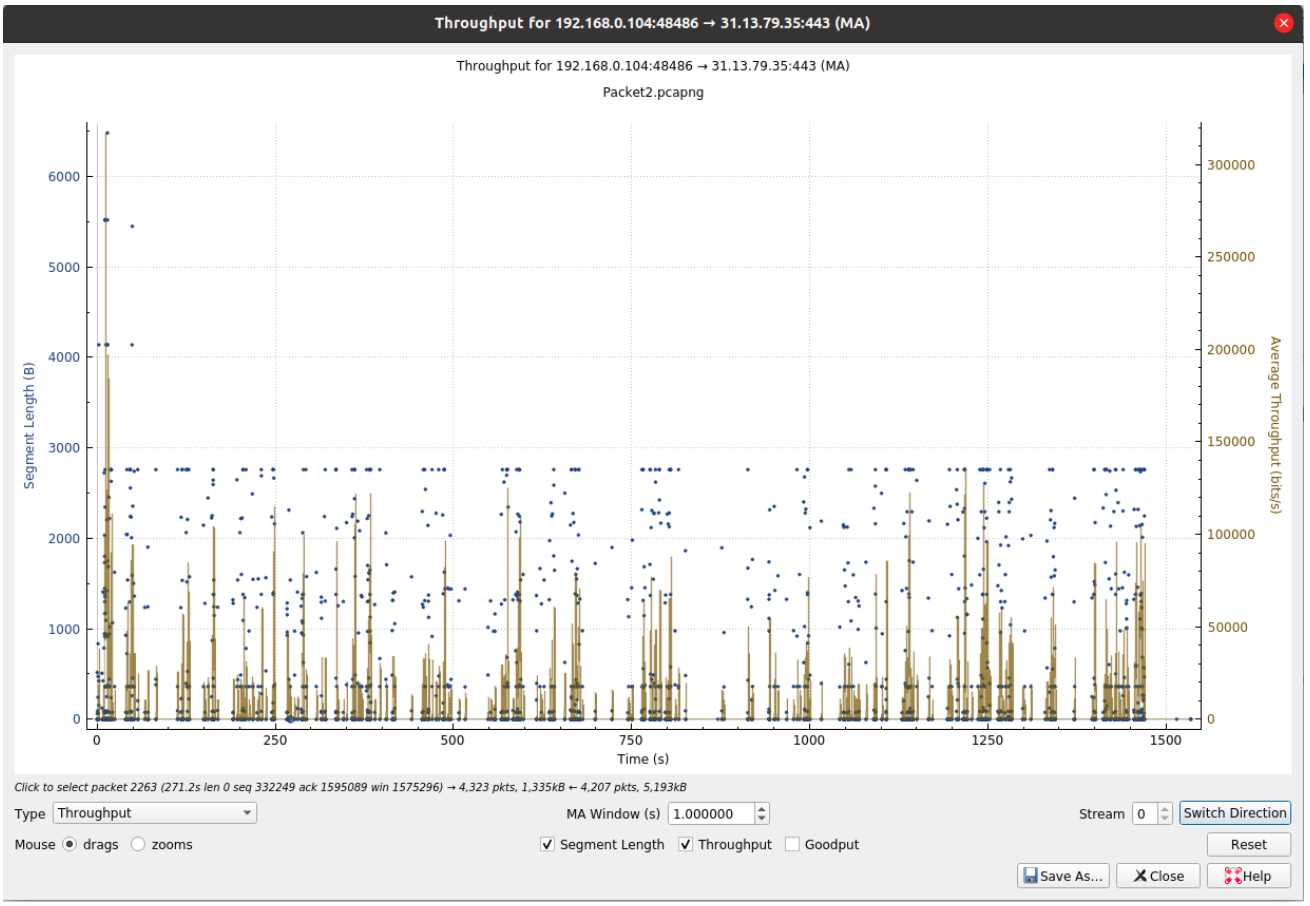
The above picture shows that Average rate of transfer in **bits/sec is 36k.** Which is average throughput

We can also view the distribution of Throughputs of various packets sent and received from Facebook to the machine.

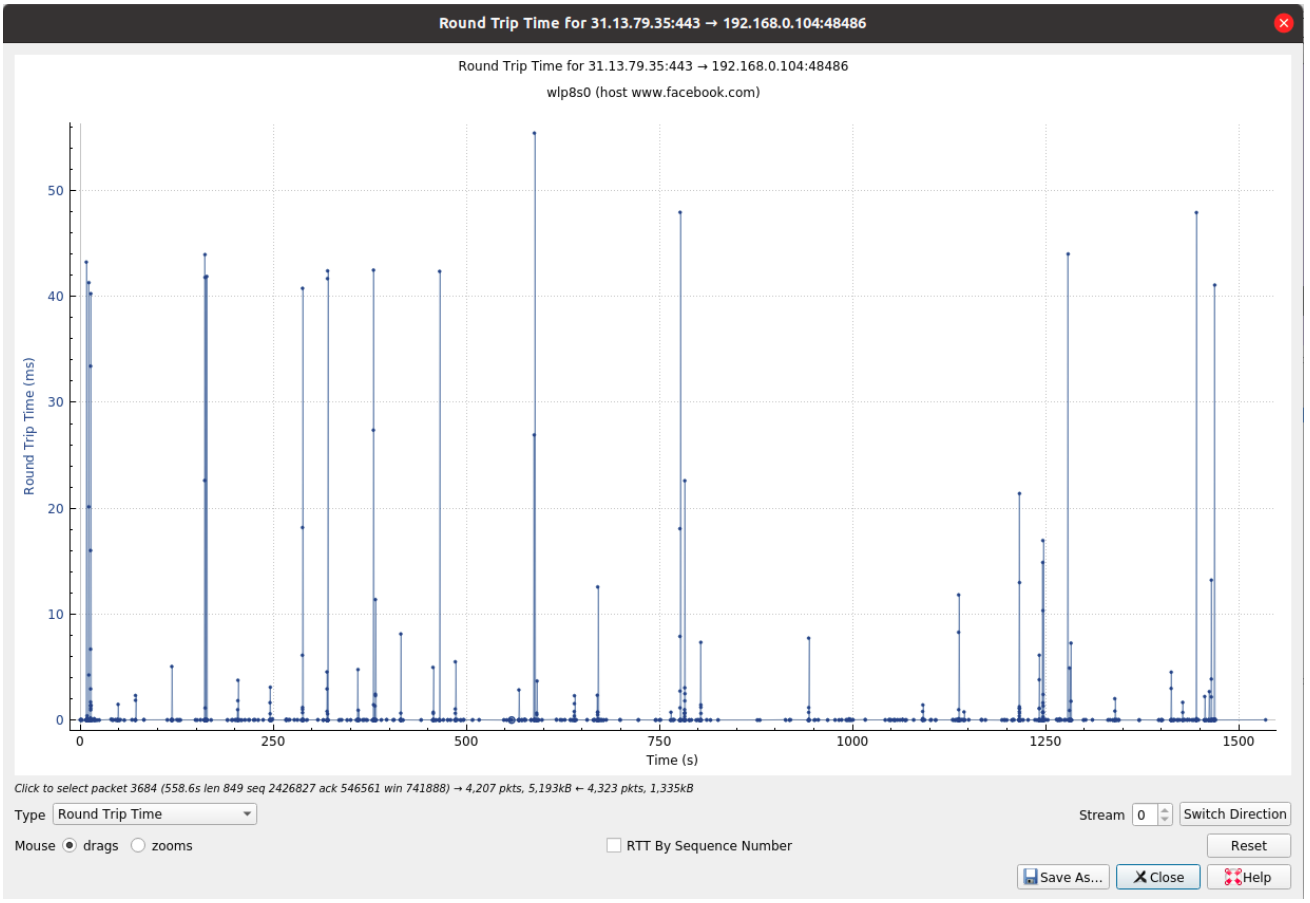
We can do this by **“Statistics -> TCP Stream Graphs -> Throughput”**



We can see in the above graph that there are a lot of spikes in speeds varying from 10^4 range to 8 * 10^5



We can see in the above graph that there are a lot of spikes in speeds varying from 5*10^4 range to 15 * 10^4



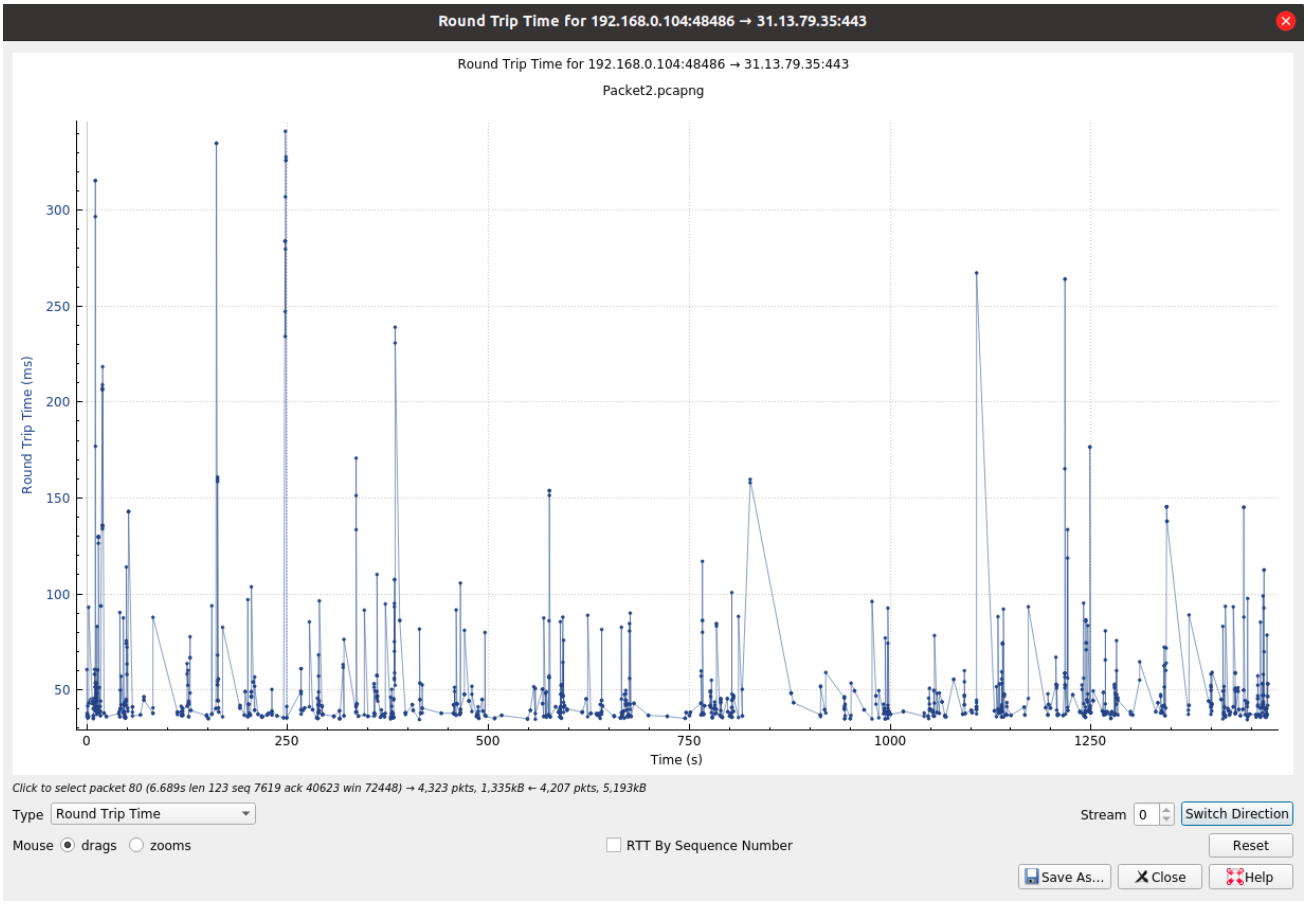
Round Trip Time

Round Trip Time (RTT) is the length time it takes for a data packet to be sent to a destination plus the time it takes for an acknowledgment of that packet to be received back at the origin.

It Is possible to plot the graphs similar to those of throughputs for to and fro packets in Wireshark.

We can do so by **“Statistics -> TCP Stream Graphs -> Round Trip Time”**

The following graph plots the RTT of packets going from machine to facebook. We can see they vary between 0.1ms to 10 ms with major chunks lying from 0 to 4.
The packets coming from Facebook have very less RTT. Because of the high speed servers of Facebook and the acknowledgement is also sent immediately from our machine

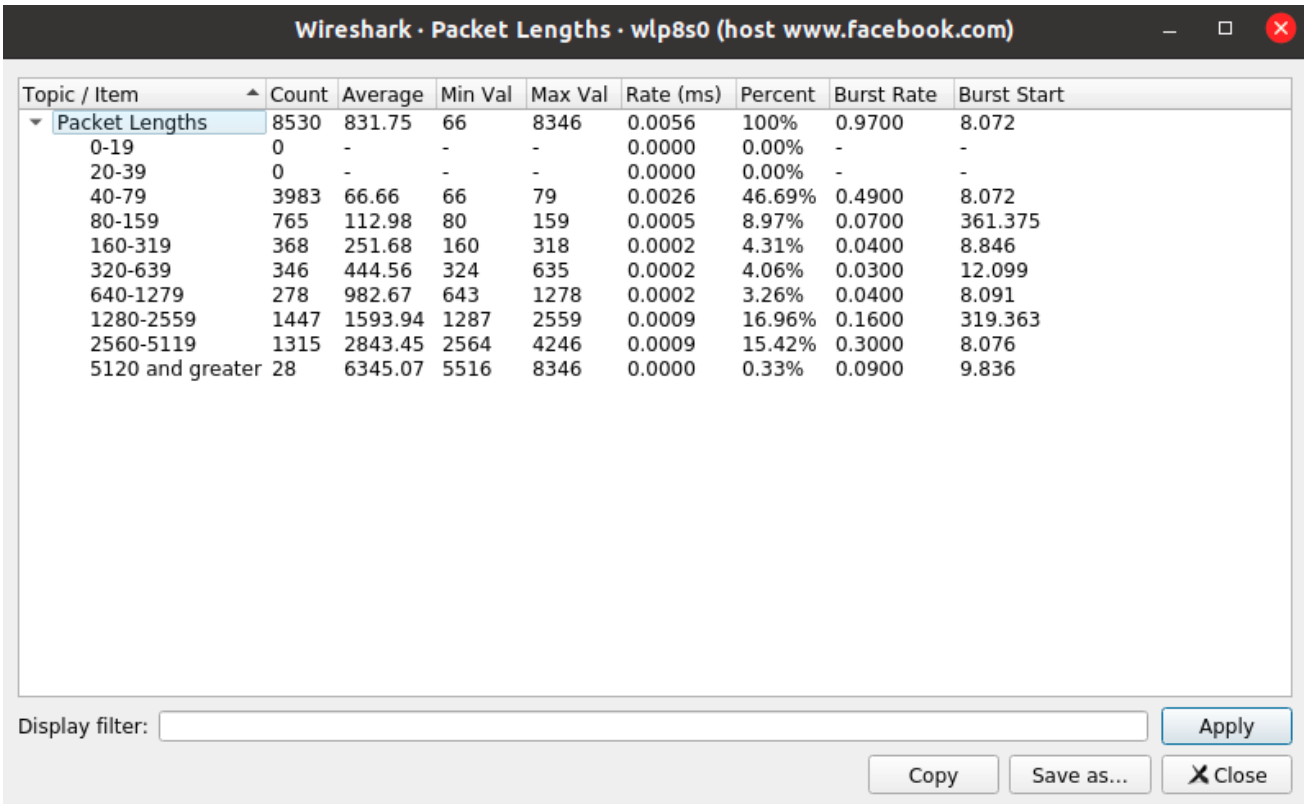


The following graph plots the RTT of packets going from machine to facebook. We can see they vary between 0.1 ms to 100 ms with major chunks lying from 0 to 10.

Packet Length

The summary of amount of data carried by packets can be displayed using **“Statistics -> Packet lengths”**

Following are the results for data collected at different times



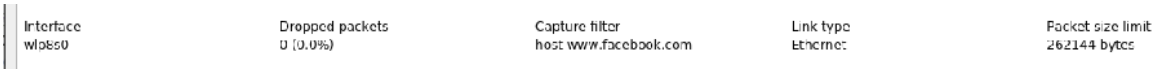
In the above data, we can see the average length of packet is 831.,75 bytes

Packets Lost

We can check the number of packets lost using the **“Statistics -> Capture File Properties”**

The summary shows many details including Dropped packets.

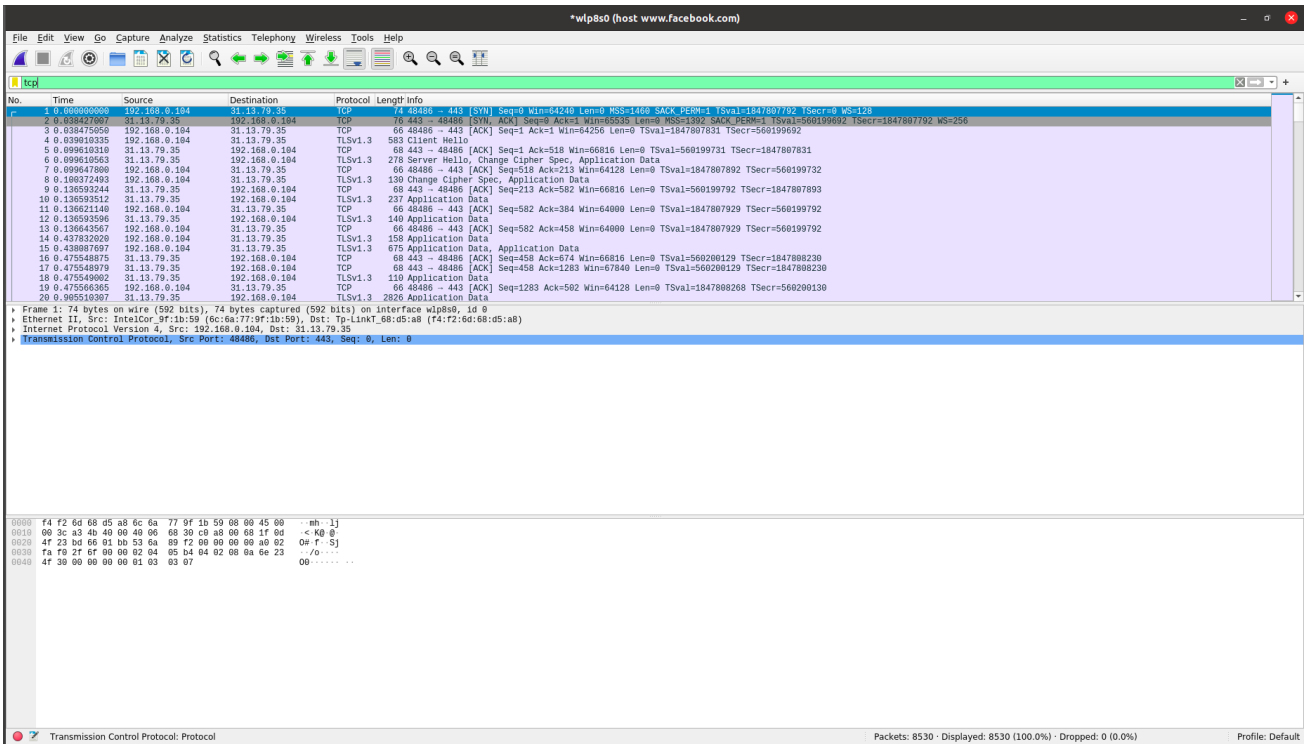
Luckily, due to the reliable Facebook servers, we get 0 dropped packets.



TCP Packets

Since Facebook Only uses TCP Protocol, there isn't much use of data from only Facebook. Therefore for the experiment of Capturing TCP Packets and UDP Packets, we remove the capture filter and collect general data.

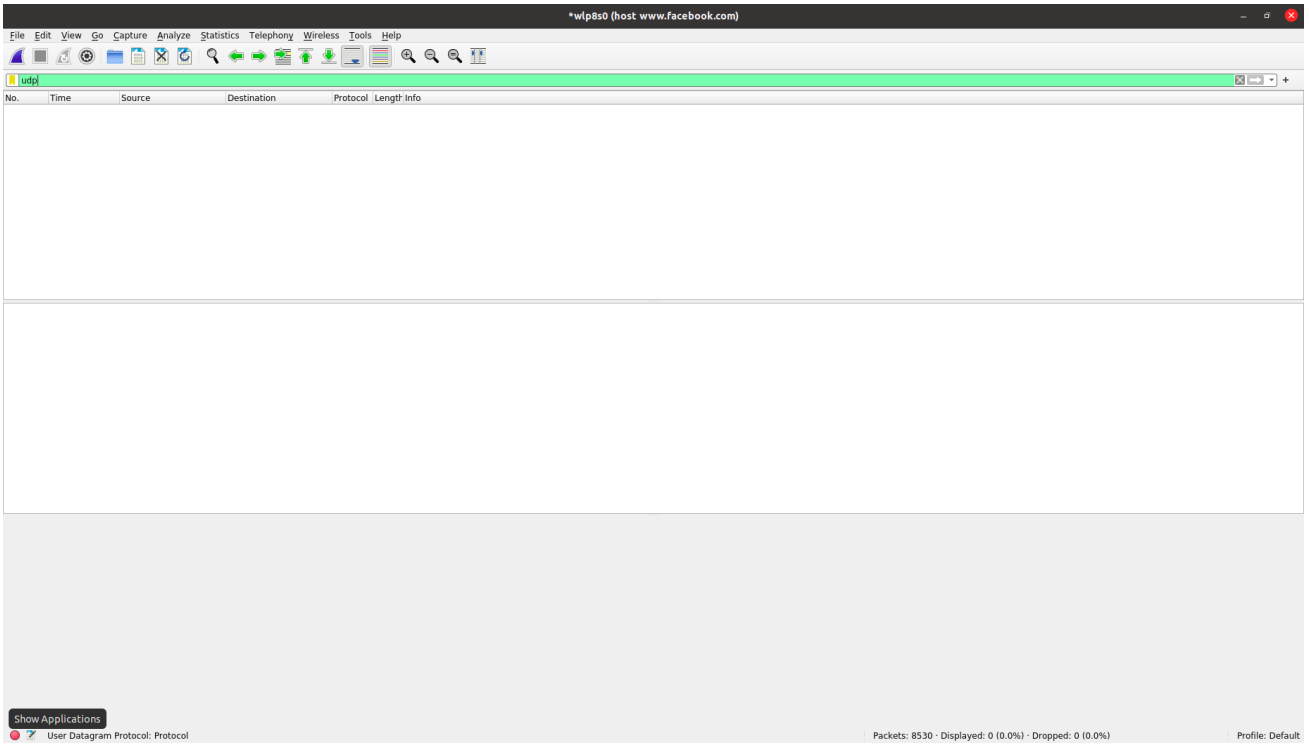
We can use the Display filter **"tcp"** to view only TCP Packets.



We can see that 8530 packets out of 8530 captured packets are TCP protocol.

i.e. 100% of packets follow TCP protocol.

UDP Packets



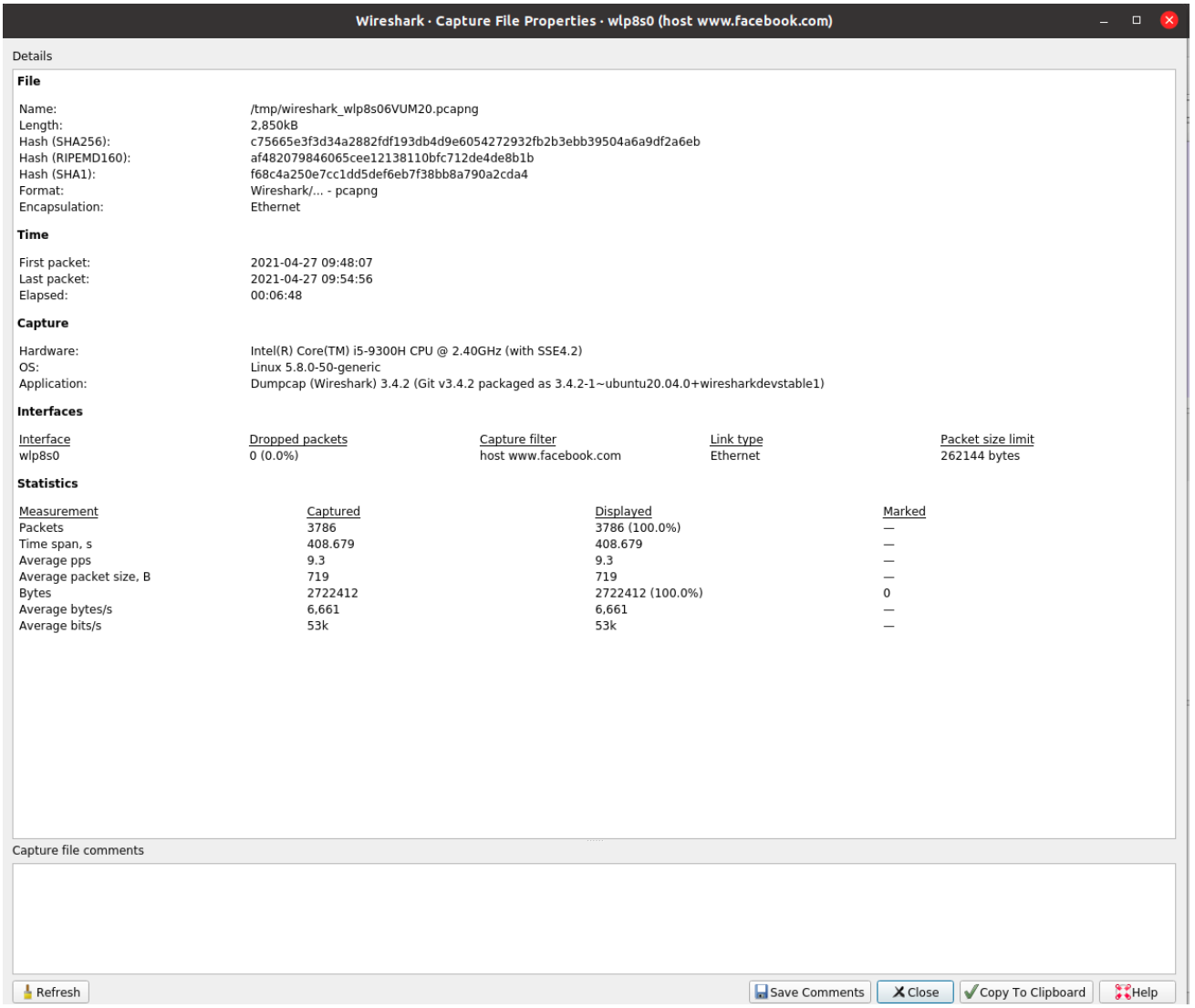
Similarly, we can capture the UDP Packets using the display filter **“udp”** We can see that none of the packets (0% of packets) out of 8530 packets follow UDP Protocol.

All the above experiments were performed 2nd time next morning

Throughput

Throughput tells you how much data was transferred from a source at any given time and bandwidth tells you how much data could theoretically be transferred from a source at any given time.

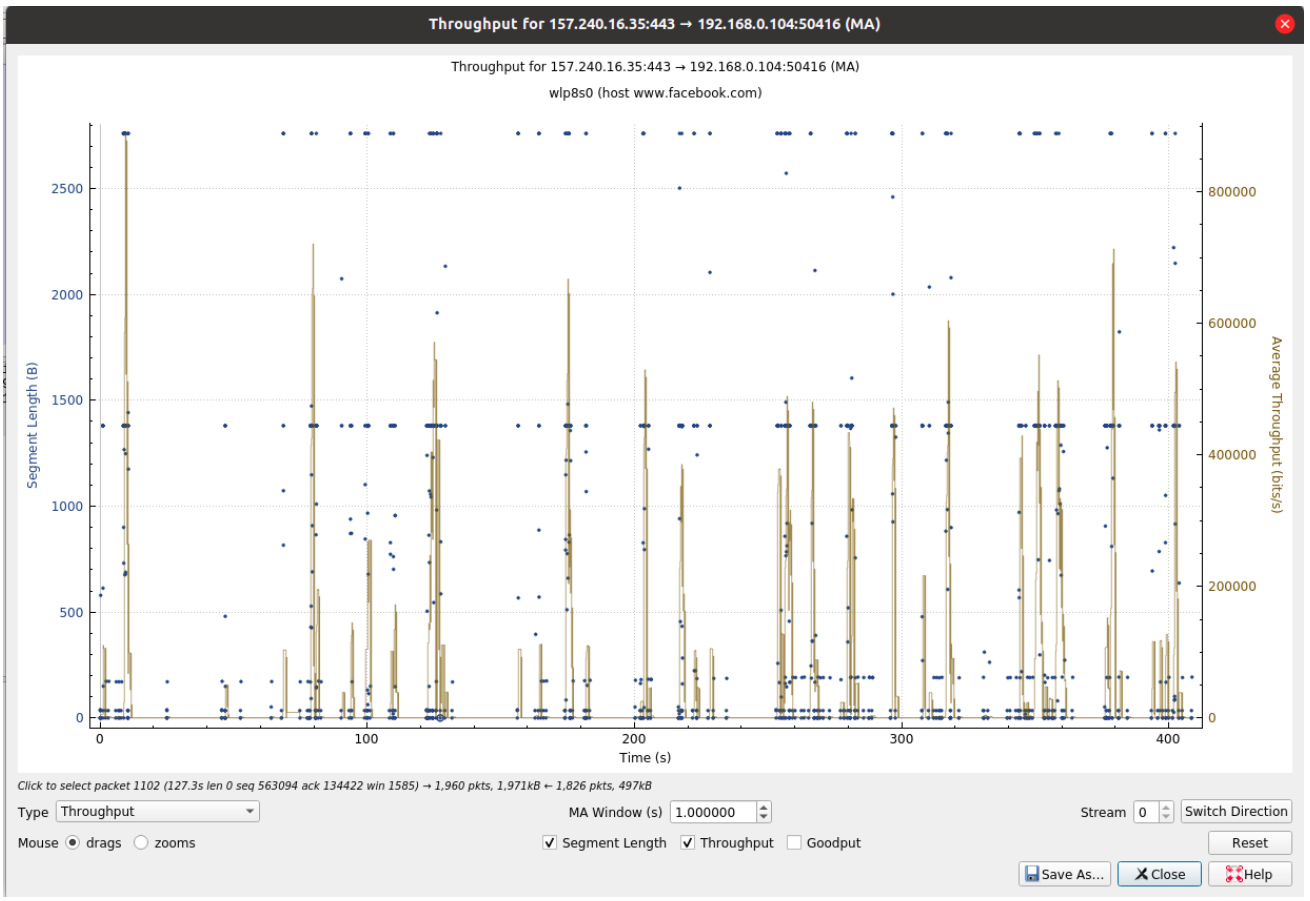
We can measure average throughput and graphical plot of throughputs of various packets using Wireshark
For summary of transfers, got to **“Statistics -> Capture File Properties”**



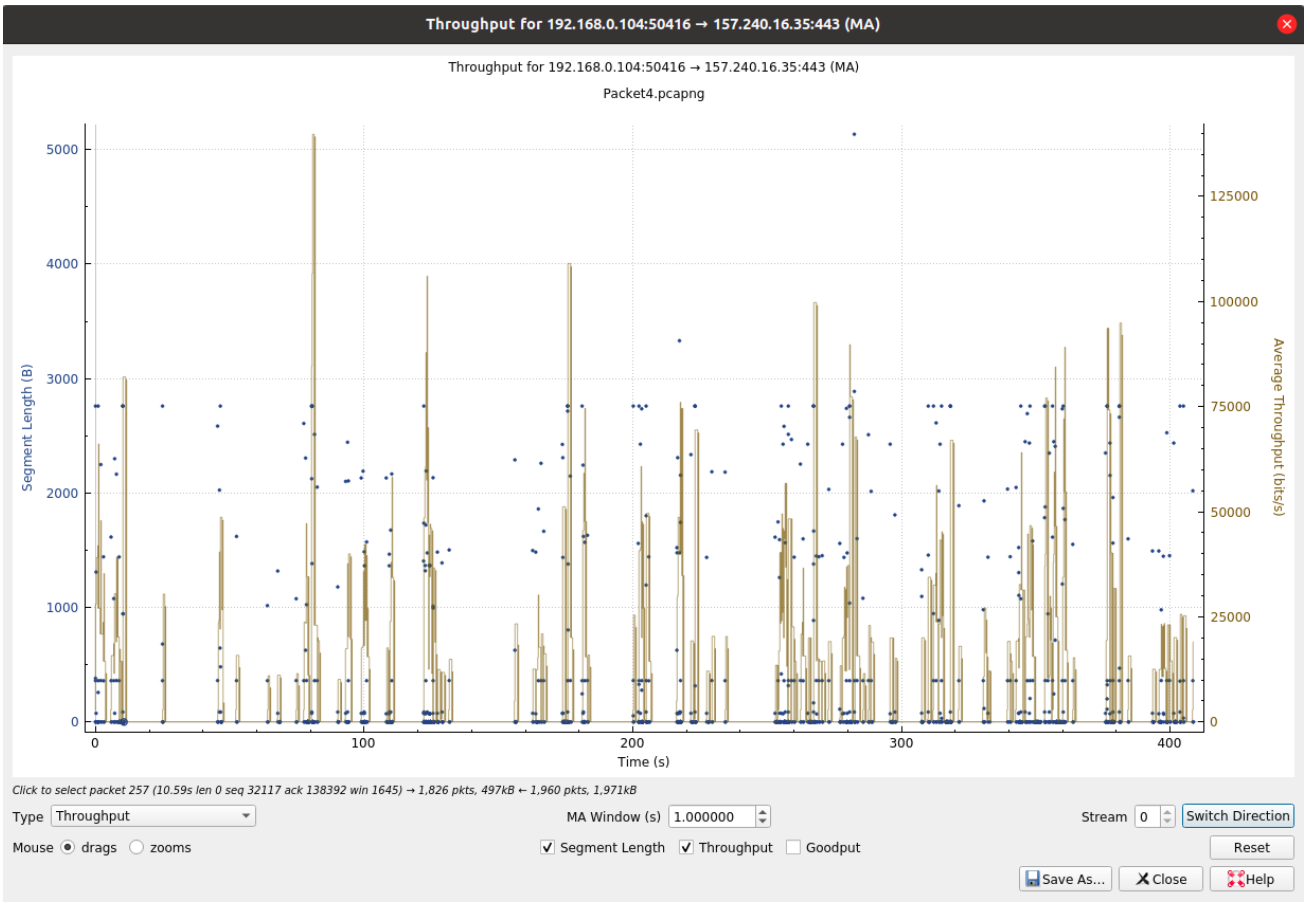
The above picture shows that Average rate of transfer in **bits/sec is 53k.** Which is average throughput

We can also view the distribution of Throughputs of various packets sent and received from Facebook to the machine.

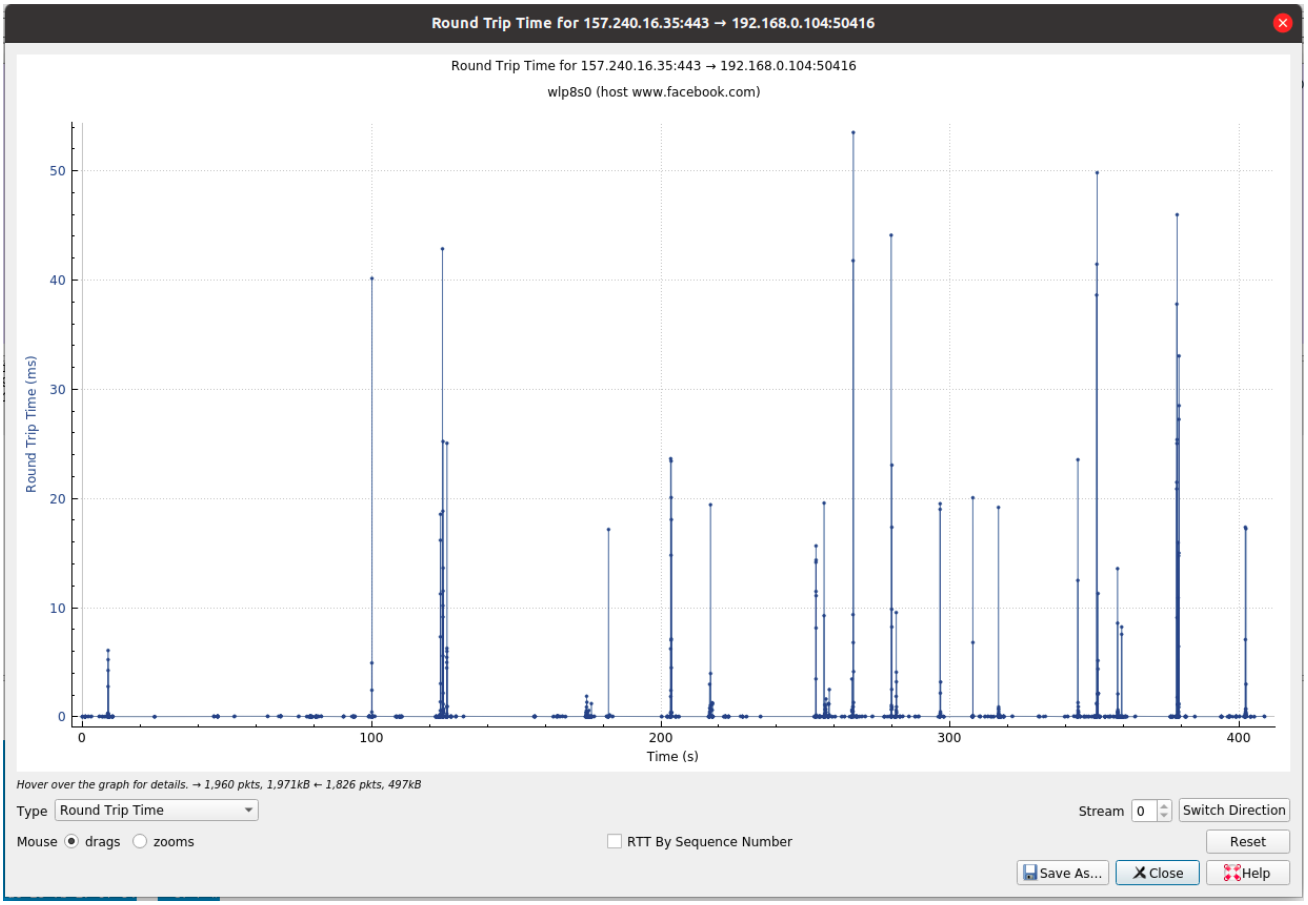
We can do this by **“Statistics -> TCP Stream Graphs -> Throughput”**



We can see in the above graph that there are a lot of spikes in speeds varying from 2*10^5 range to 8*10^5



We can see in the above graph that there are a lot of spikes in speeds varying from 2.5*10^4 range to 7.5*10^4



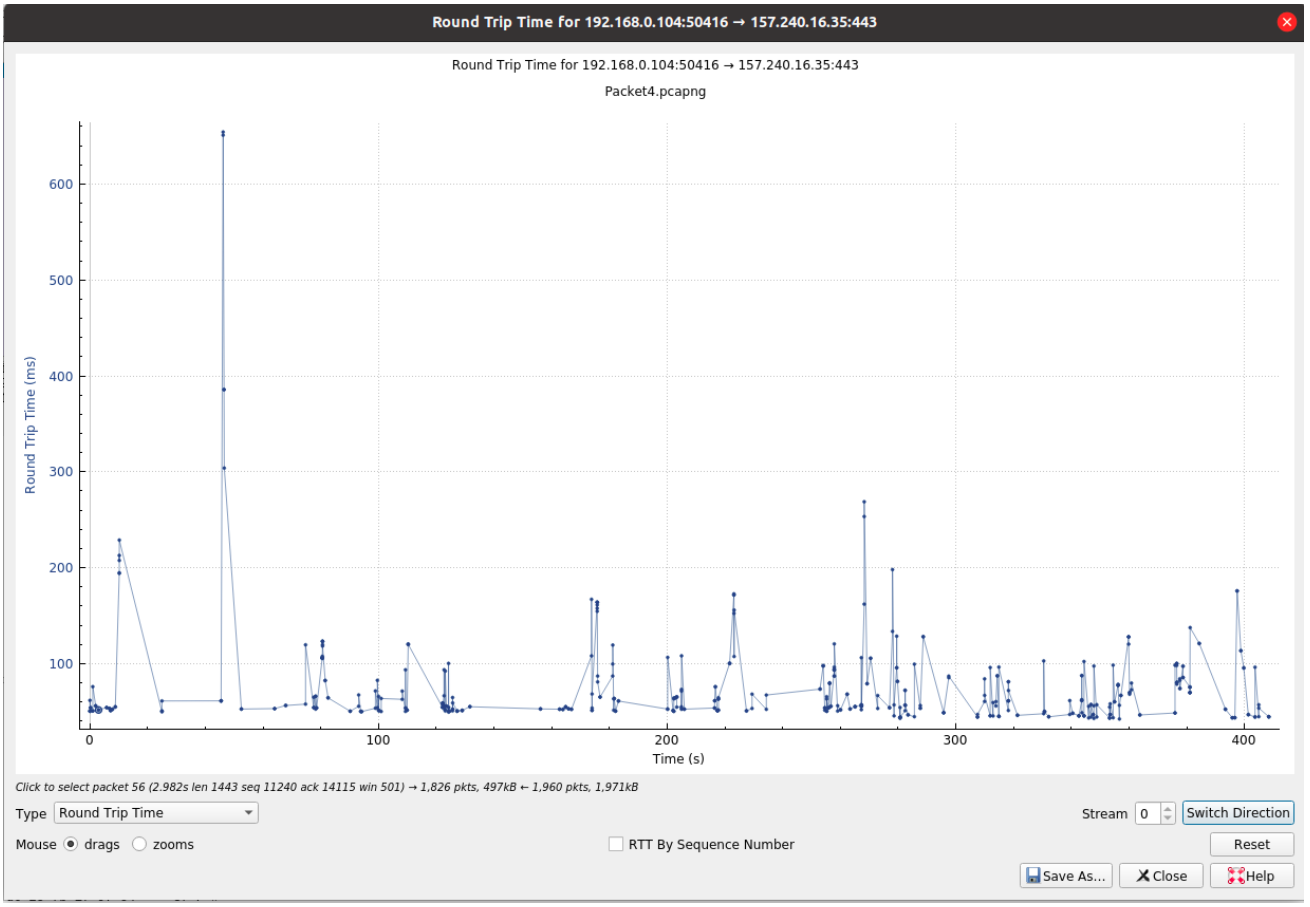
Round Trip Time

Round Trip Time (RTT) is the length time it takes for a data packet to be sent to a destination plus the time it takes for an acknowledgment of that packet to be received back at the origin.

It Is possible to plot the graphs similar to those of throughputs for to and fro packets in Wireshark.

We can do so by **“Statistics -> TCP Stream Graphs -> Round Trip Time”**

The following graph plots the RTT of packets going from machine to facebook. We can see they vary between 0.1ms to 10 ms with major chunks lying from 0 to 4.
The packets coming from Facebook have very less RTT. Because of the high speed servers of Facebook and the acknowledgement is also sent immediately from our machine

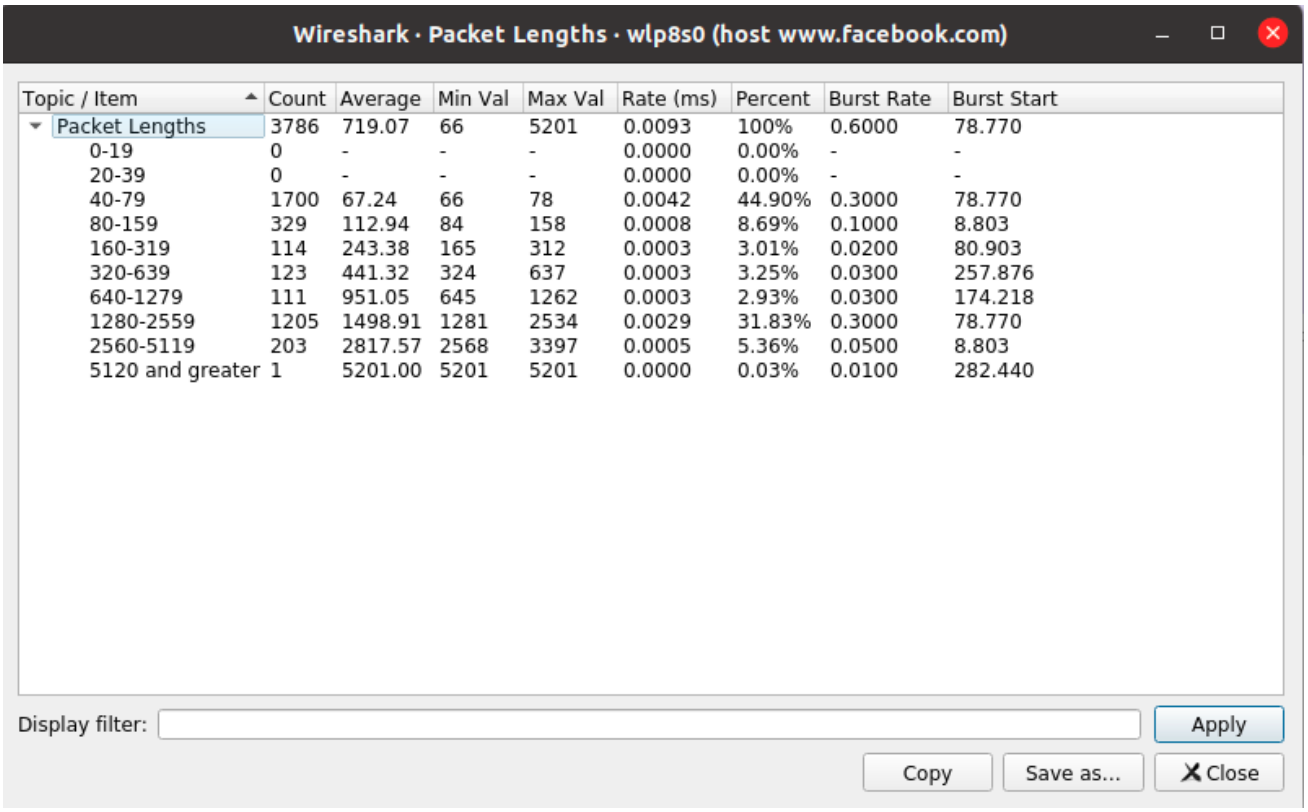


The following graph plots the RTT of packets going from machine to facebook. We can see they vary between 10 ms to 40 ms with major chunks lying from 20 to 30 .

Packet Length

The summary of amount of data carried by packets can be displayed using **“Statistics -> Packet lengths”**

Following are the results for data collected at different times



In the above data, we can see the average length of packet is 719.07 bytes

Packets Lost

We can check the number of packets lost using the **“Statistics -> Capture File Properties”**

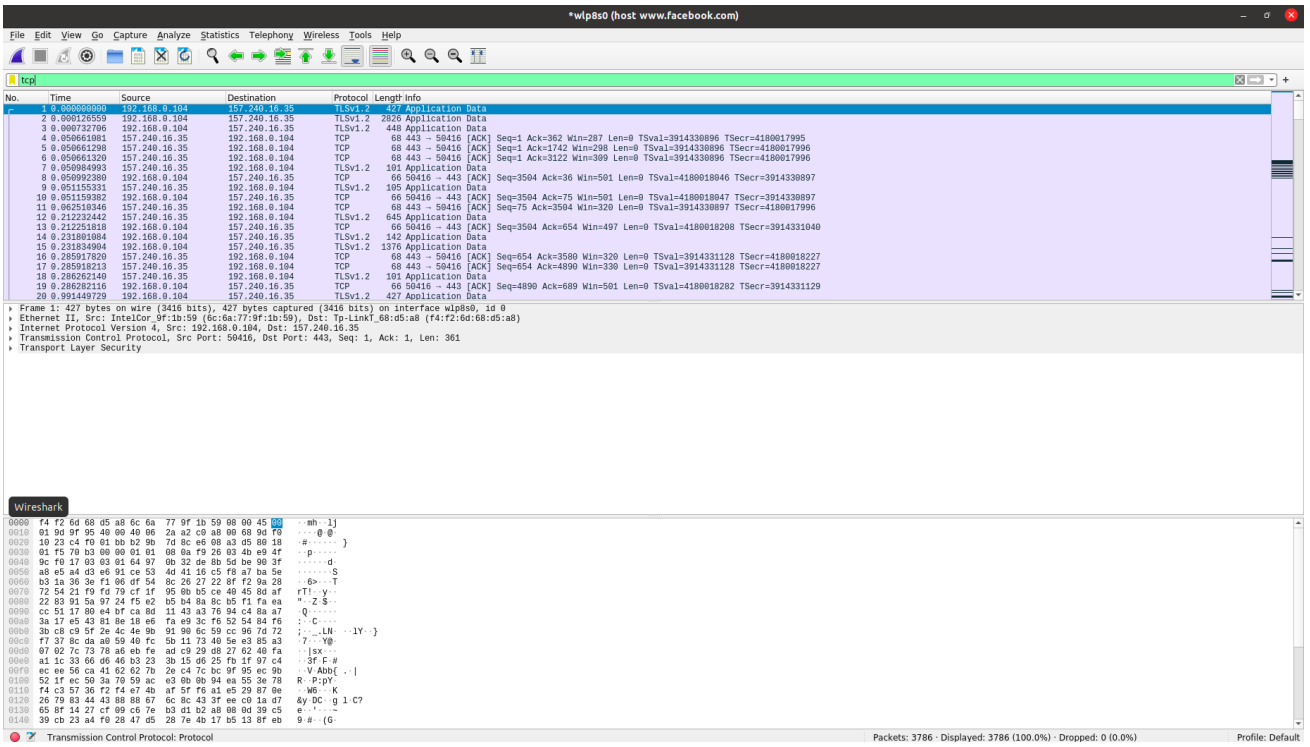
The summary shows many details including Dropped packets.

Luckily, due to the reliable Facebook servers, we get 0 dropped packets.

TCP Packets

Since Facebook Only uses TCP Protocol, there isn't much use of data from only Facebook. Therefore for the experiment of Capturing TCP Packets and UDP Packets, we remove the capture filter and collect general data.

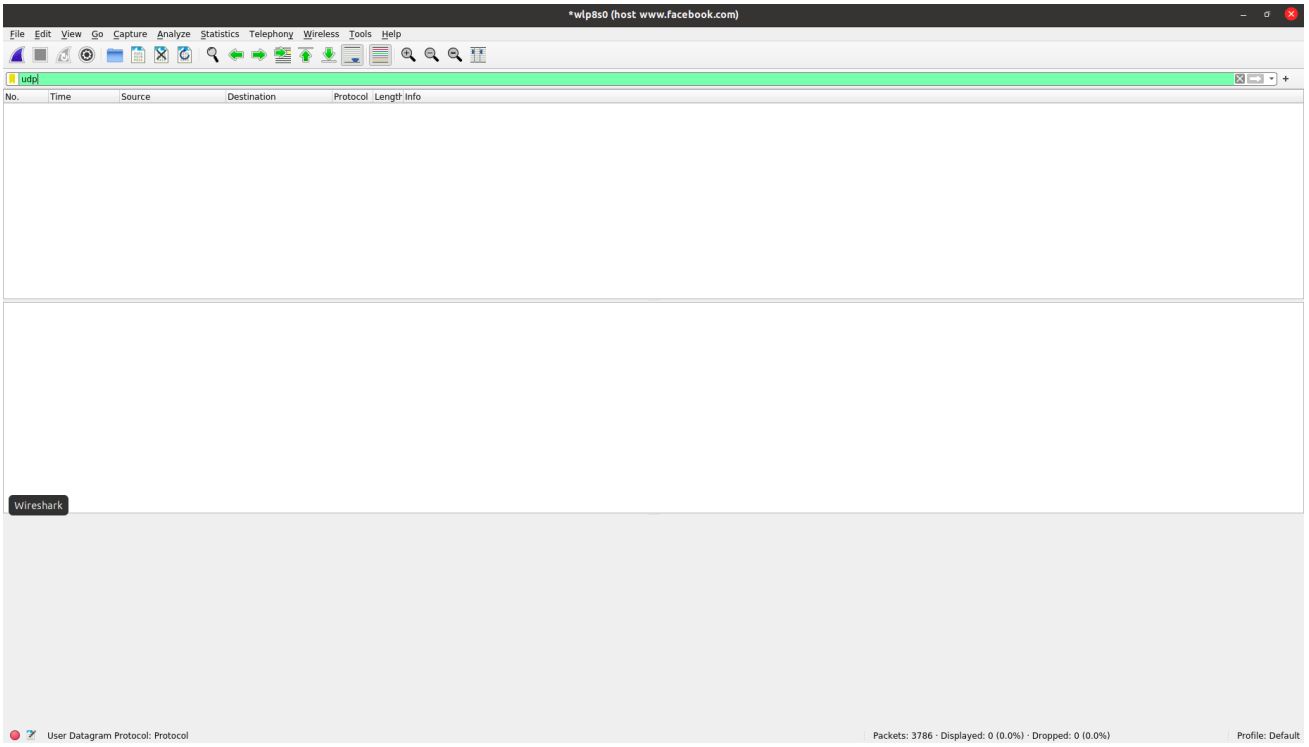
We can use the Display filter **"tcp"** to view only TCP Packets.



We can see that 3786 packets out of 3786 captured packets are TCP protocol.

i.e. 100% of packets follow TCP protocol.

UDP Packets



Similarly, we can capture the UDP Packets using the display filter **“udp”** We can see that none of the packets (0% of packets) out of 3786packets follow UDP Protocol.

Request-Response

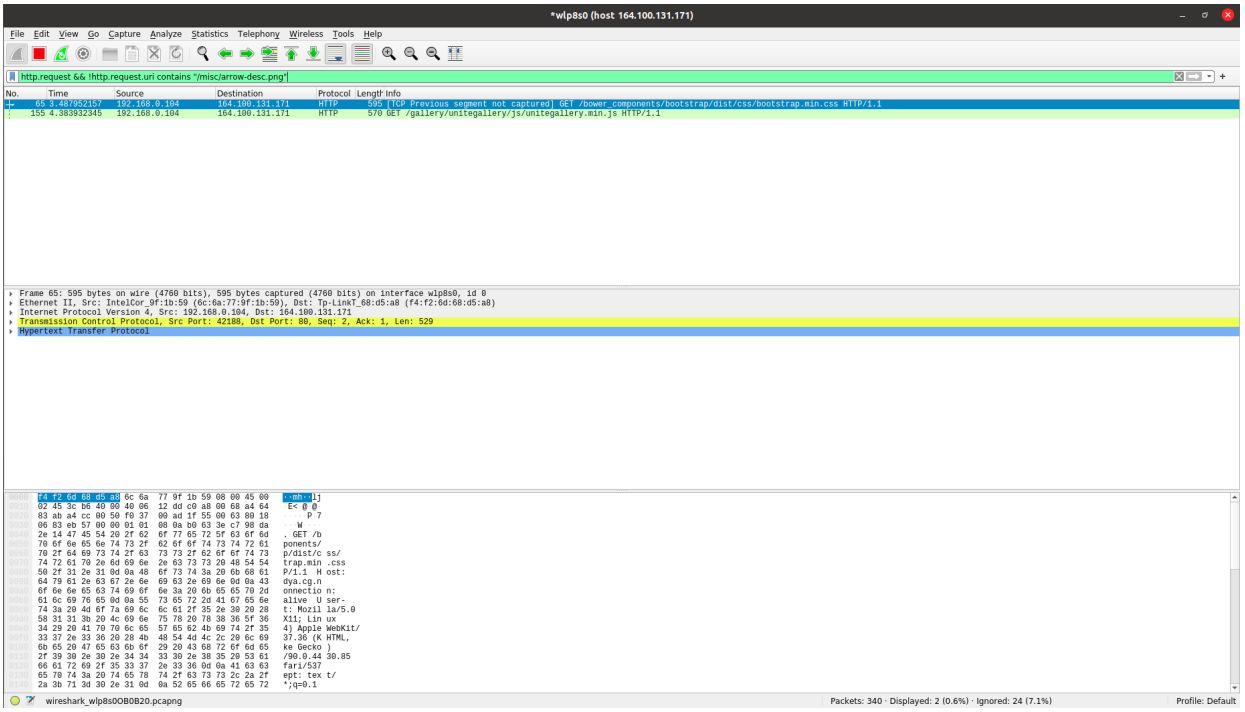
As a part of this experiment, we are also required to capture Request Response pairs. i.e. How many Responses would be there for one particular Request.

Therefore, we need HTTP GET data. Unfortunately, Facebook follows HTTPS protocol, and therefore we need to change our source for this experiment. A website which uses HTTP is essential.

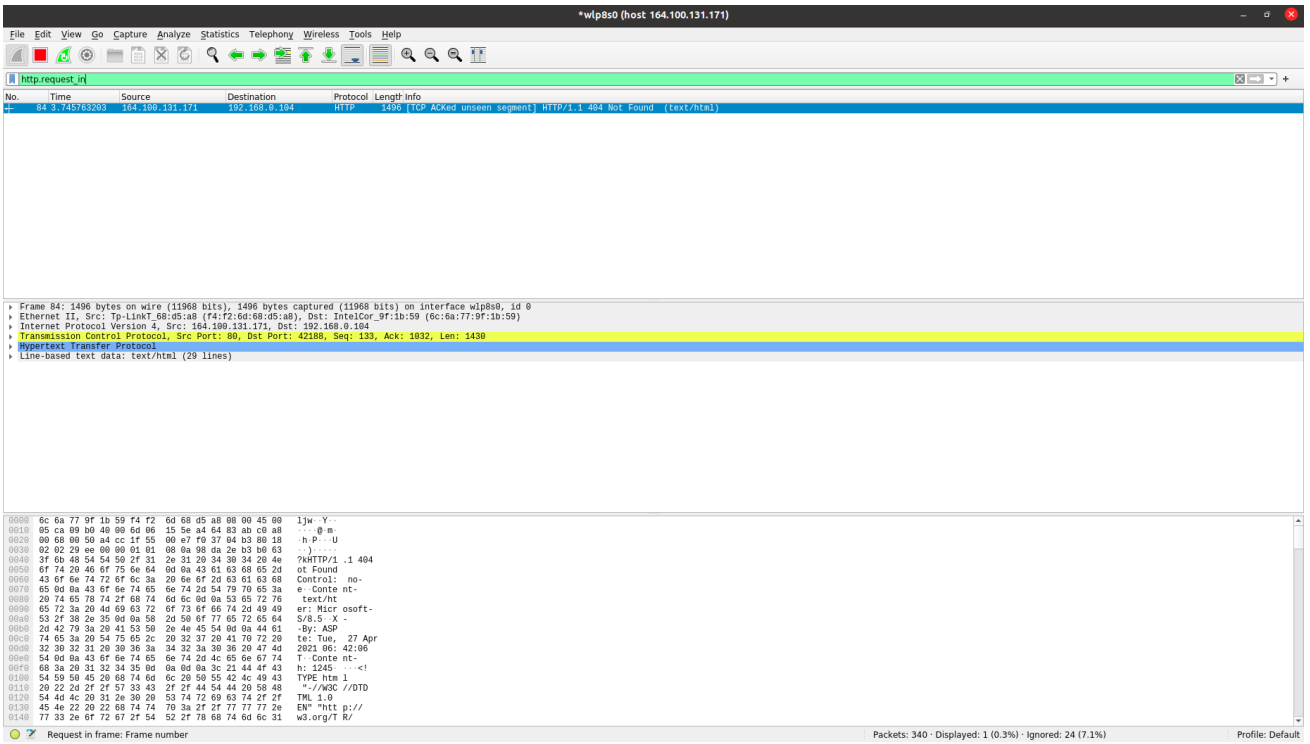
For this experiment, we use a site http://khadya.cg.nic.in/Directorate_AboutUs_Hn.aspx IP address is 157.140.2.239. We can capture its data using the **“host 164.100.131.171”** capture filter.
Now that we have the data, we need to perform the following steps to capture its response packets

1. From the data, display the request packets using **“http.request”** display filter
2. From the resulting list, choose one random request and display it. I have used “http.request.uri contains "/misc/arrow-desc.png"” for this experiment.

3. Next step involves removing all the other requests and their responses. This could be done using **“http.request && !http.request.uri contains "/misc/arrow-desc.png"”** display filter and remove that list using **“Edit -> Ignore all displayed”**



4. Now there is only the response for the request we are considering. We can displa the response using **“http.request_in”** display filter



We can now see the singular response for our request.