CS-359 Assignment-11

Name: P. V. Sriram

Roll No.: 1801CS37

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. Firstly, we need the IP address of Facebook.com to apply capture filter.
2. We can do so by using the ping facebook.com command
3. We can also find out our own IP address using the ifconfig command
4. Facebook => 69.171.250.35
5. PC => 10.0.2.15
6. After we have this data, we can use capture filter as host 69.171.250.35 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”**

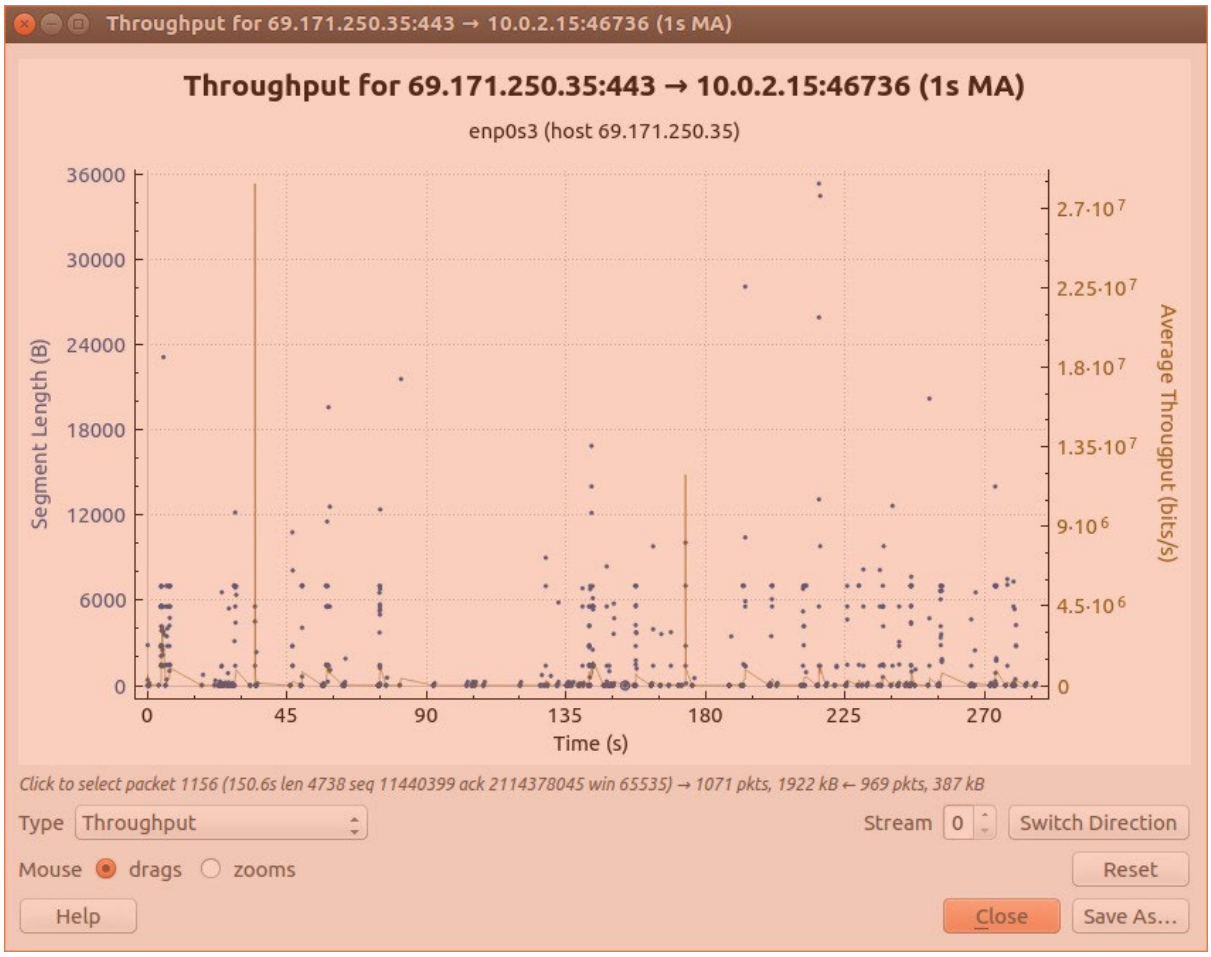
Statistics

Measurement	Captured	Displayed	Marked
Packets	2084	2084 (100.0%)	N/A
Time span, s	286.756	286.756	N/A
Average pps	7.3	7.3	N/A
Average packet size, B	1167.5	1167.5	N/A
Bytes	2433996	2433996 (100.0%)	0
Average bytes/s	8488	8488	N/A
Average bits/s	67 k	67 k	N/A

The above picture shows that Average rate of transfer in **bits/sec is 67k.** Which is average throughput

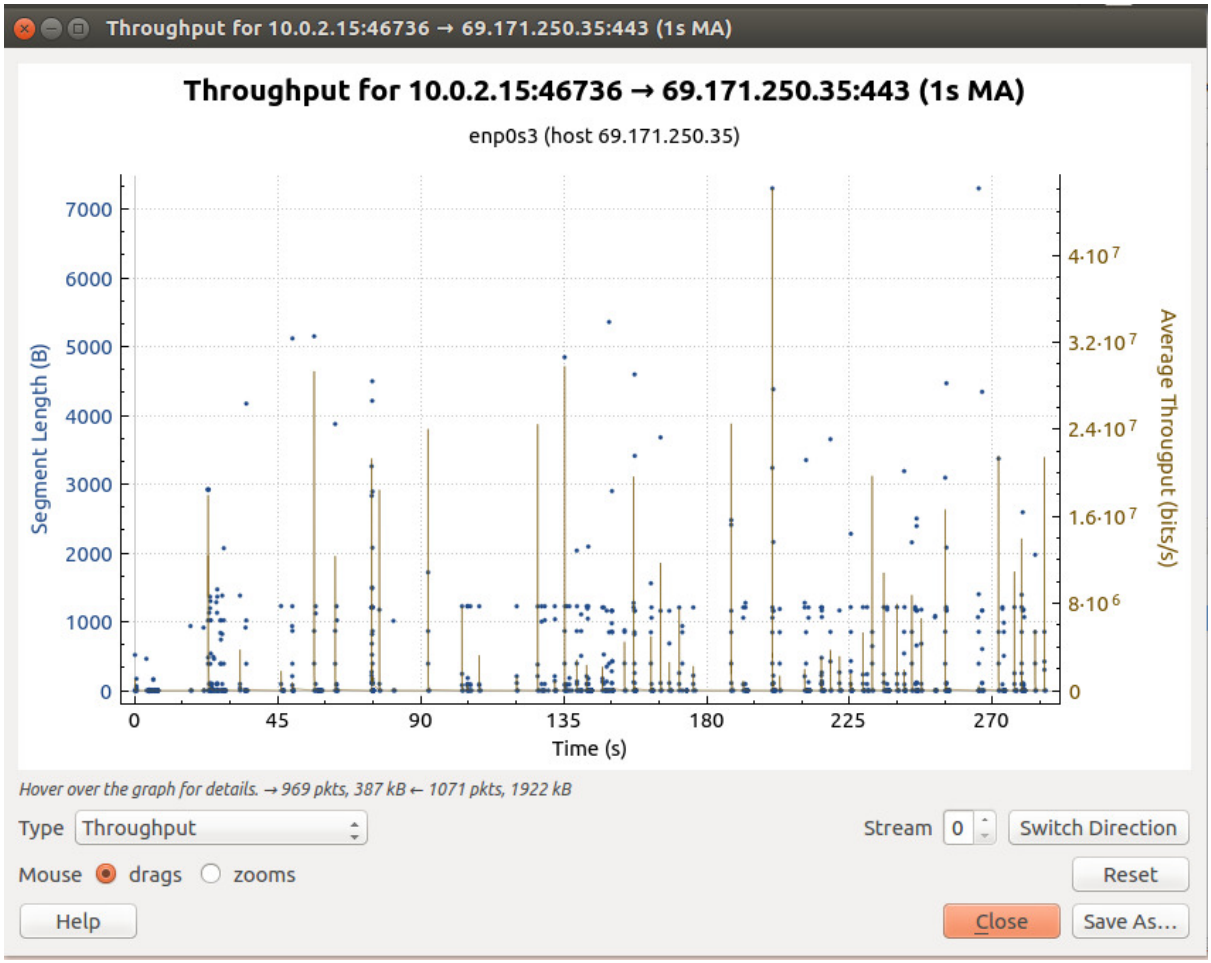
We can also view the distribution of Throughputs of various packets going coming from Facebook to the machine.

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 10^6 range to 10^7

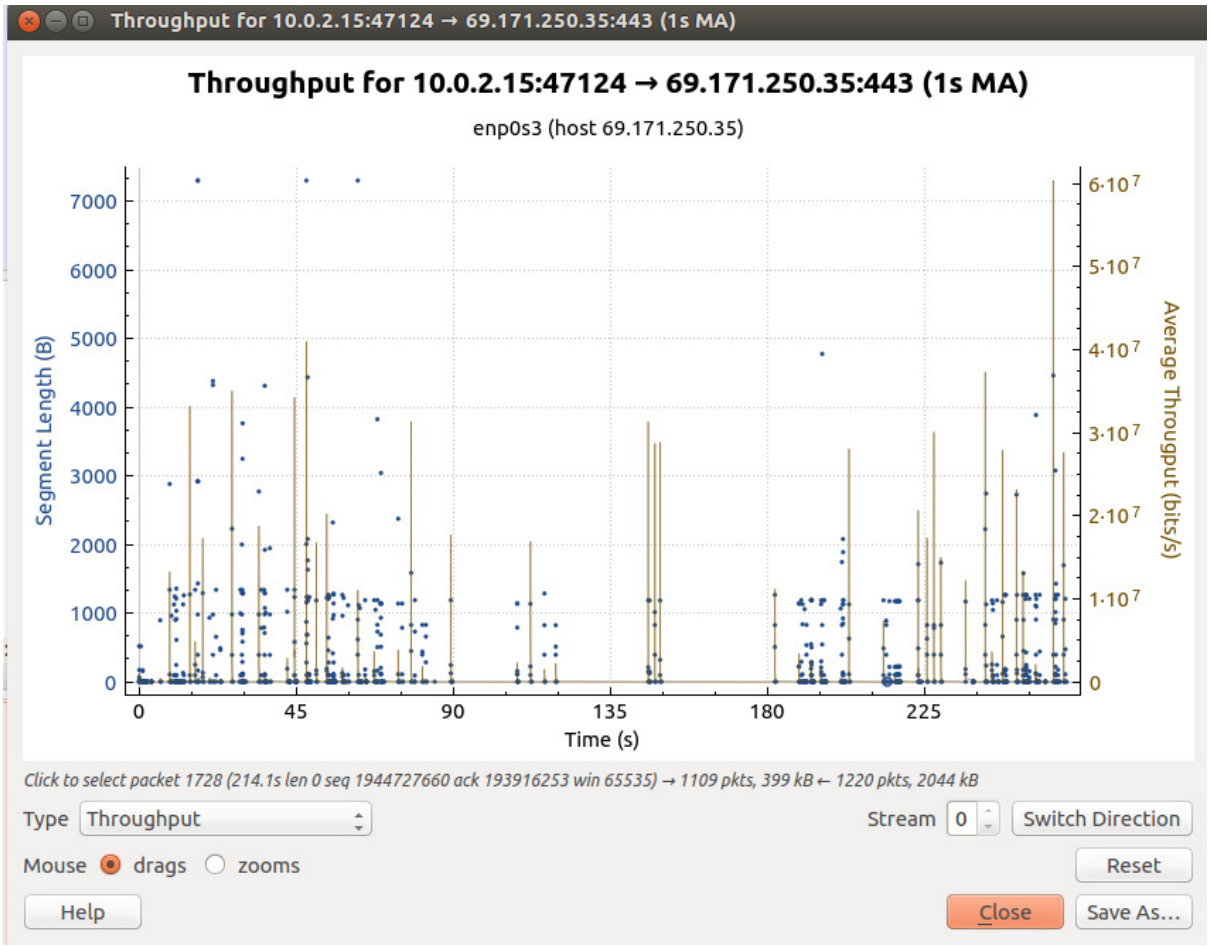
We can also view at packets going from machine to Facebook by switching directions

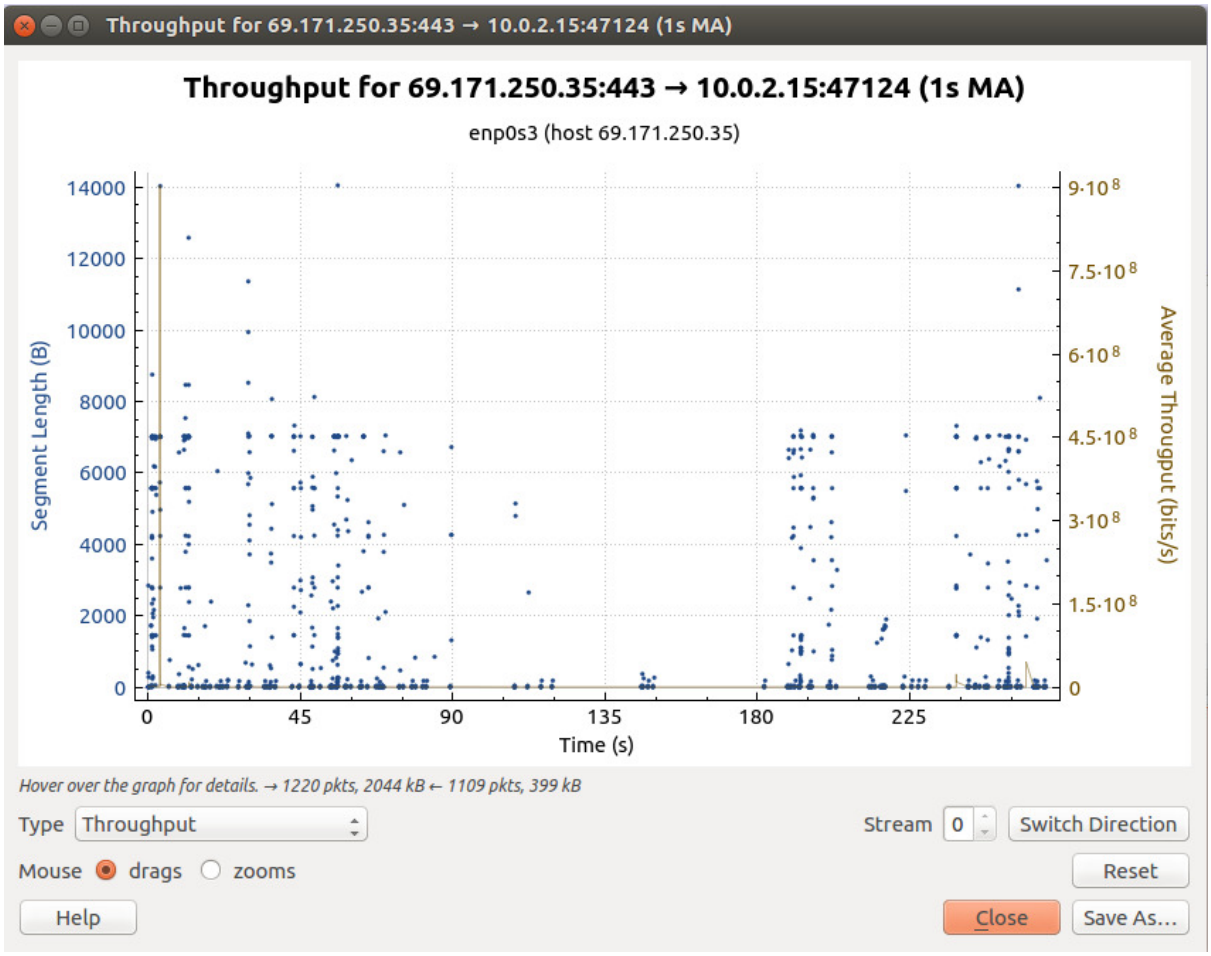


Similarly, we can repeat the experiment at another duration of the day. The following are the results.

Statistics

Measurement	Captured	Displayed	Marked
Packets	2329	2329 (100.0%)	N/A
Time span, s	265.495	265.495	N/A
Average pps	8.8	8.8	N/A
Average packet size, B	1104.5	1104.5	N/A
Bytes	2572872	2572872 (100.0%)	0
Average bytes/s	9690	9690	N/A
Average bits/s	77 k	77 k	N/A





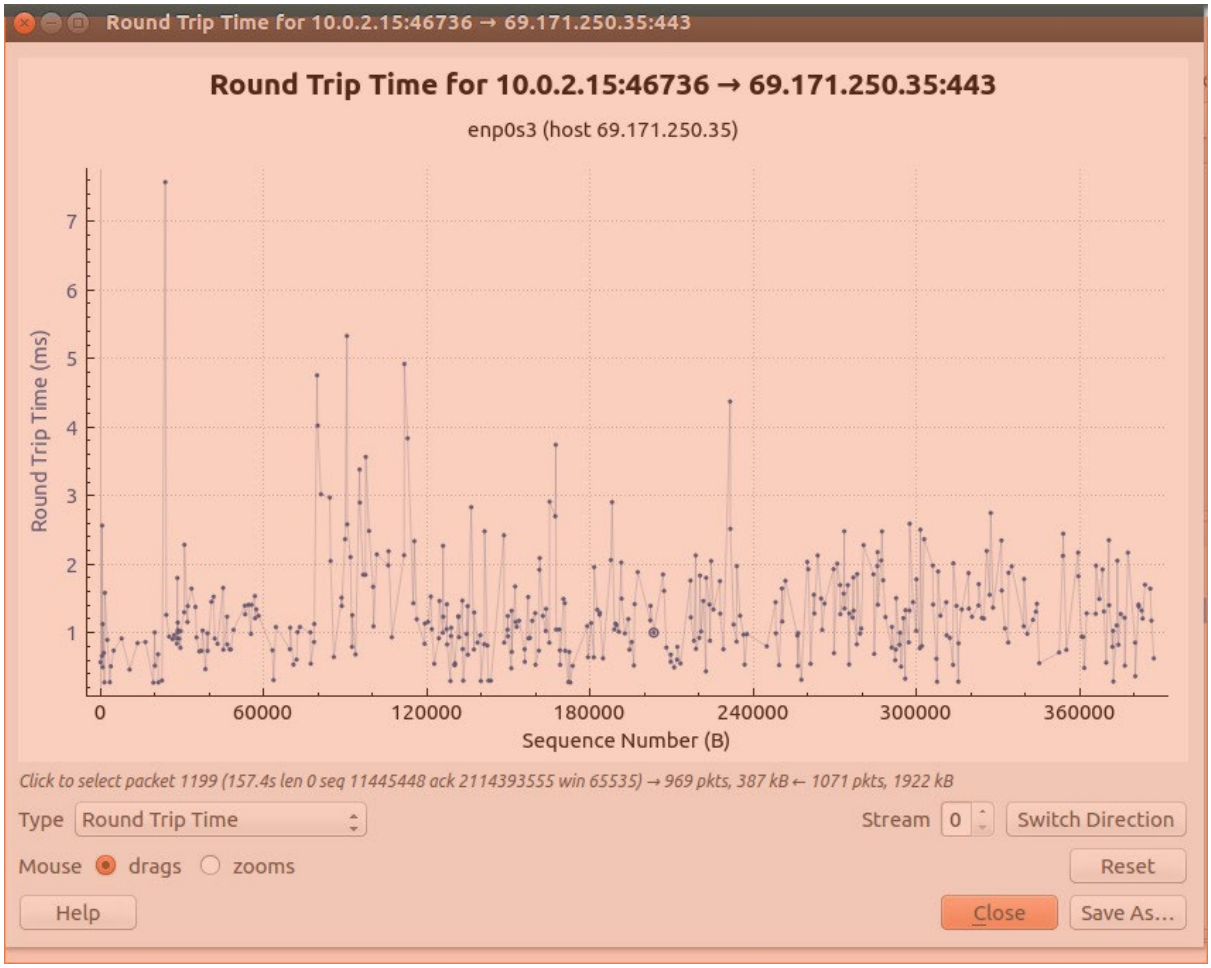
Round Trip Time

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.

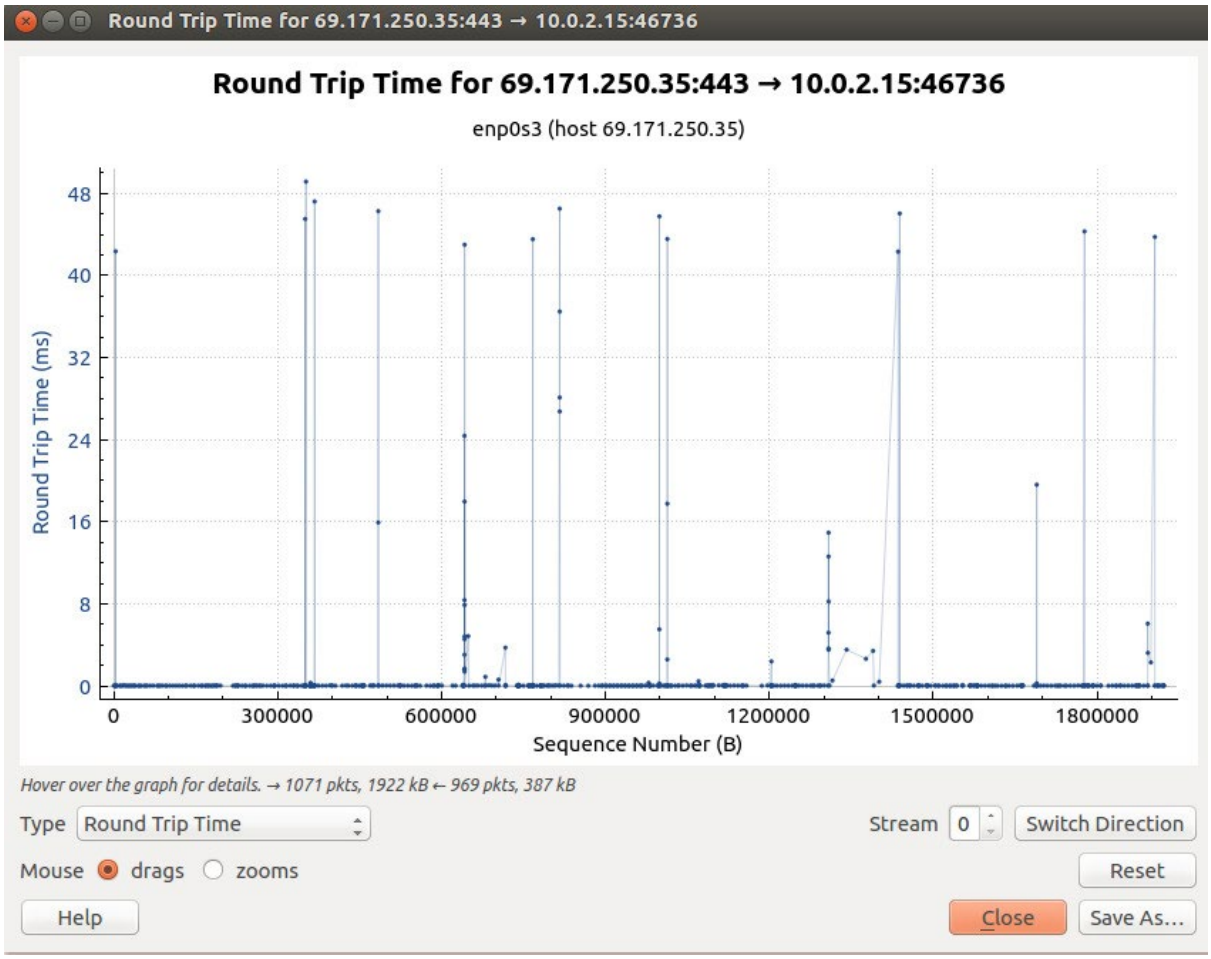
It Is possible to plot the graphs similar to those of throughputs for to and fro packets in Wireshark.

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

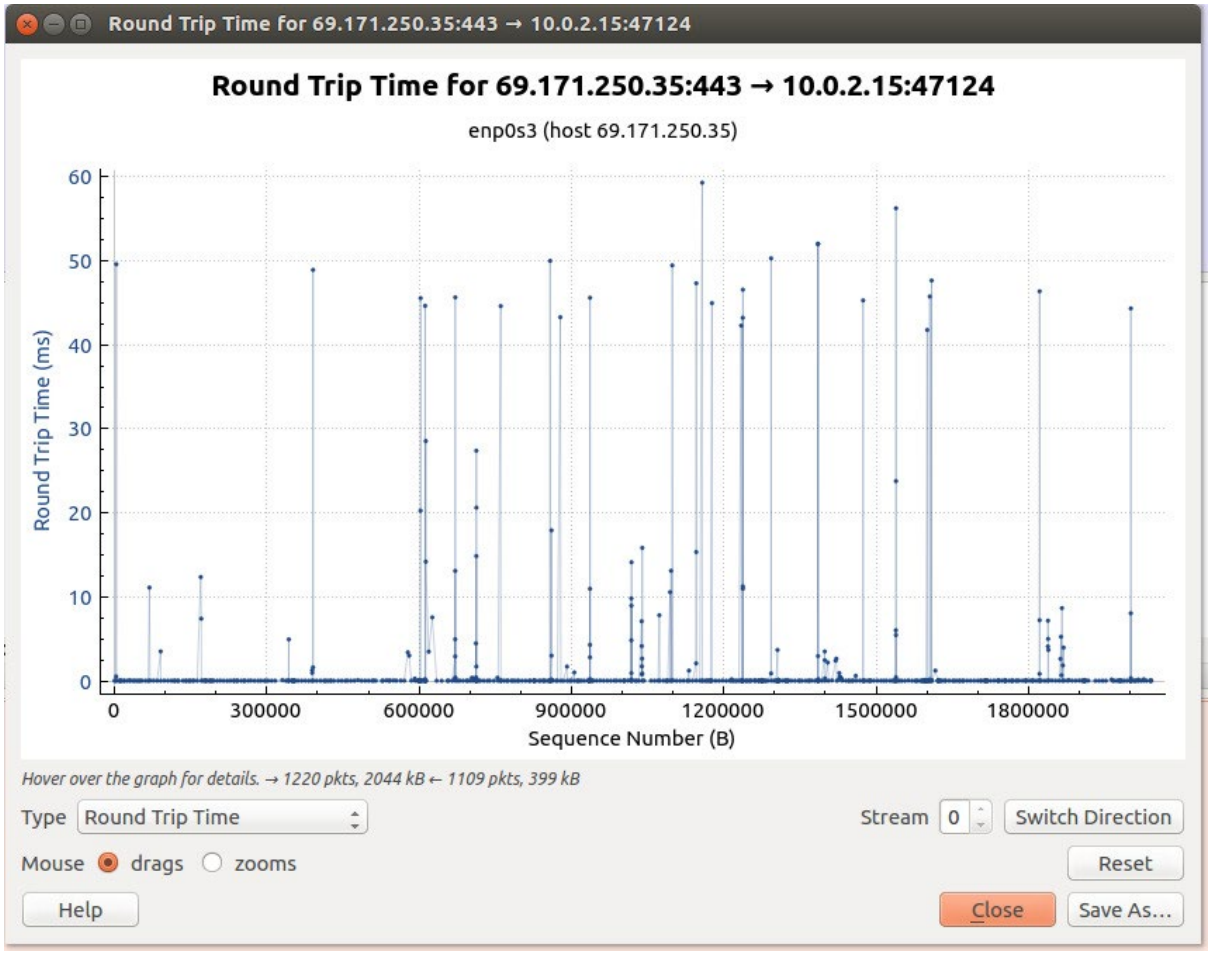
The following graph plots the RTT of packets going from machine to facebook. We can see they vary between 0.1ms to 7 ms with major chunk lying from 0 to 2.

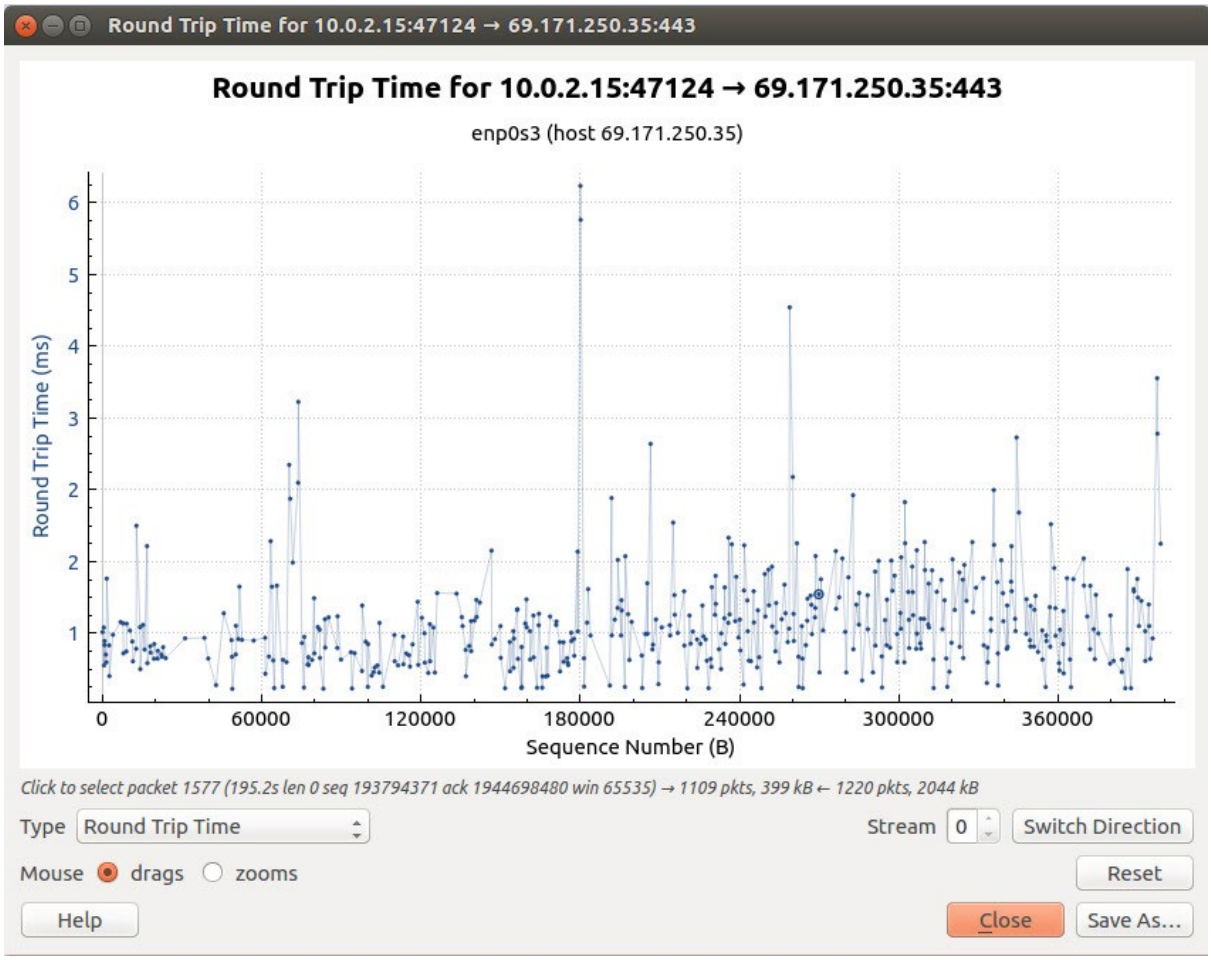

Round Trip Time for 10.0.2.15:46736 → 69.171.250.35:443
Round Trip Time for 10.0.2.15:46736 → 69.171.250.35:443
enp0s3 (host 69.171.250.35)
Round Trip Time (ms)
1
2
3
4
5
6
7
0
60000
120000
180000
240000
300000
360000
Sequence Number (B)
Click to select packet 1199 (157.4s len 0 seq 11445448 ack 2114393555 win 65535) → 969 pkts, 387 kB ← 1071 pkts, 1922 kB
Type
Round Trip Time
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...

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



Similar experiment could be replicated at another time of the day by performing same steps.

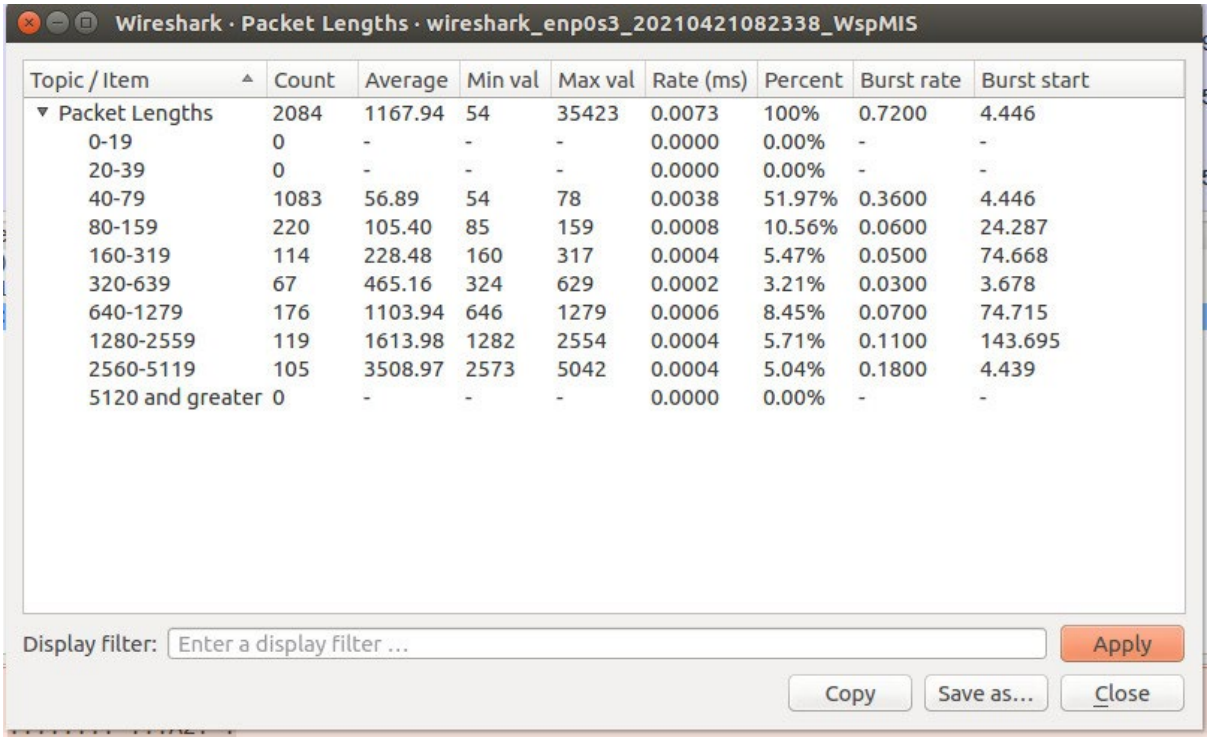

Round Trip Time for 69.171.250.35:443 → 10.0.2.15:47124
Round Trip Time for 69.171.250.35:443 → 10.0.2.15:47124
enp0s3 (host 69.171.250.35)
Round Trip Time (ms)
60
50
40
30
20
10
0
0
300000
600000
900000
1200000
1500000
1800000
Sequence Number (B)
Hover over the graph for details. → 1220 pkts, 2044 kB ← 1109 pkts, 399 kB
Type
Round Trip Time
Stream
0
Switch Direction
Mouse
drags
zooms
Reset
Help
Close
Save As...



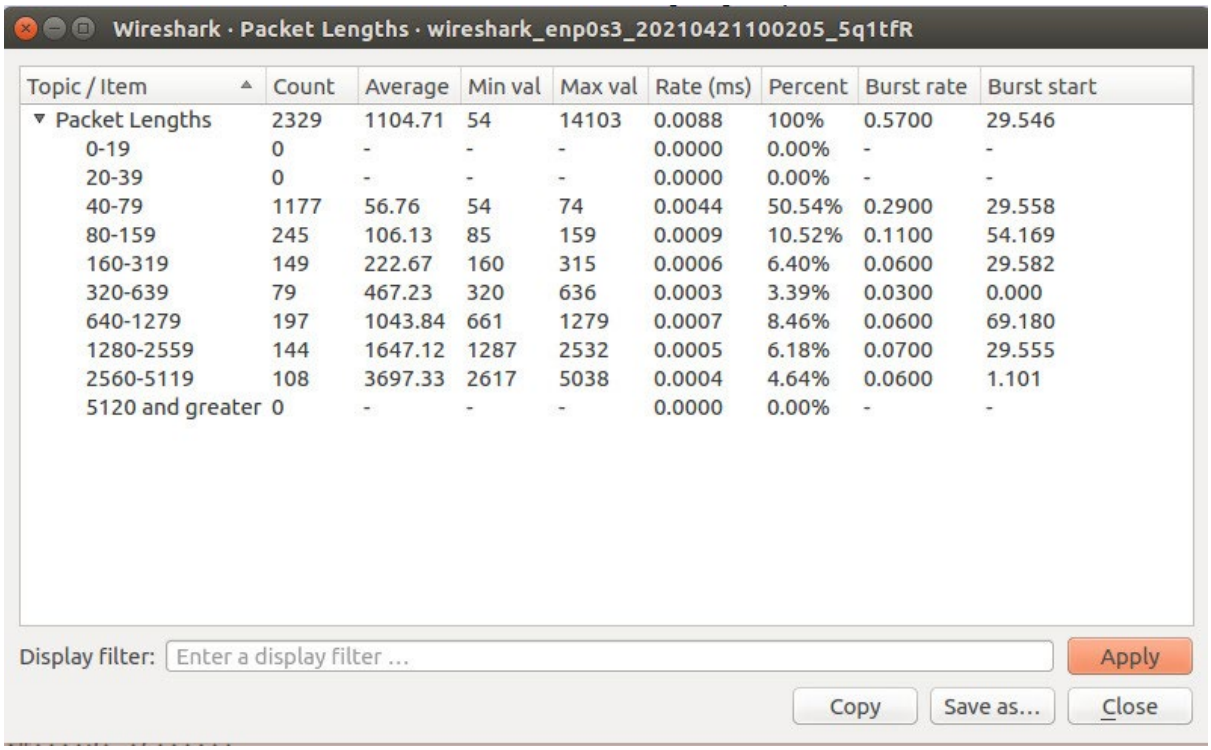
Packet Length

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 1167.94 bytes



In the above data, we can see the average length of packet is 1104.71 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.

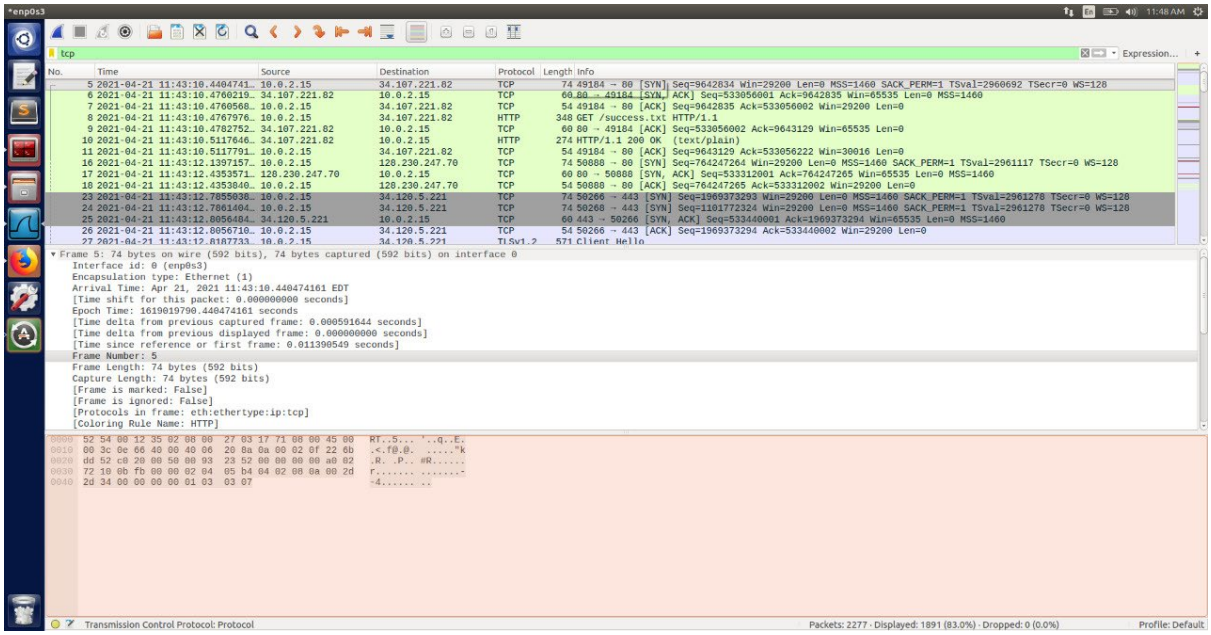
Luckily, due to the reliable Facebook servers, we get 0 dropped packets.

Interface	Dropped packets	Capture filter	Link type	Packet size limit
enp0s3	0 (0 %)	host 69.171.250.35	Ethernet	262144 bytes

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 at only TCP Packets.

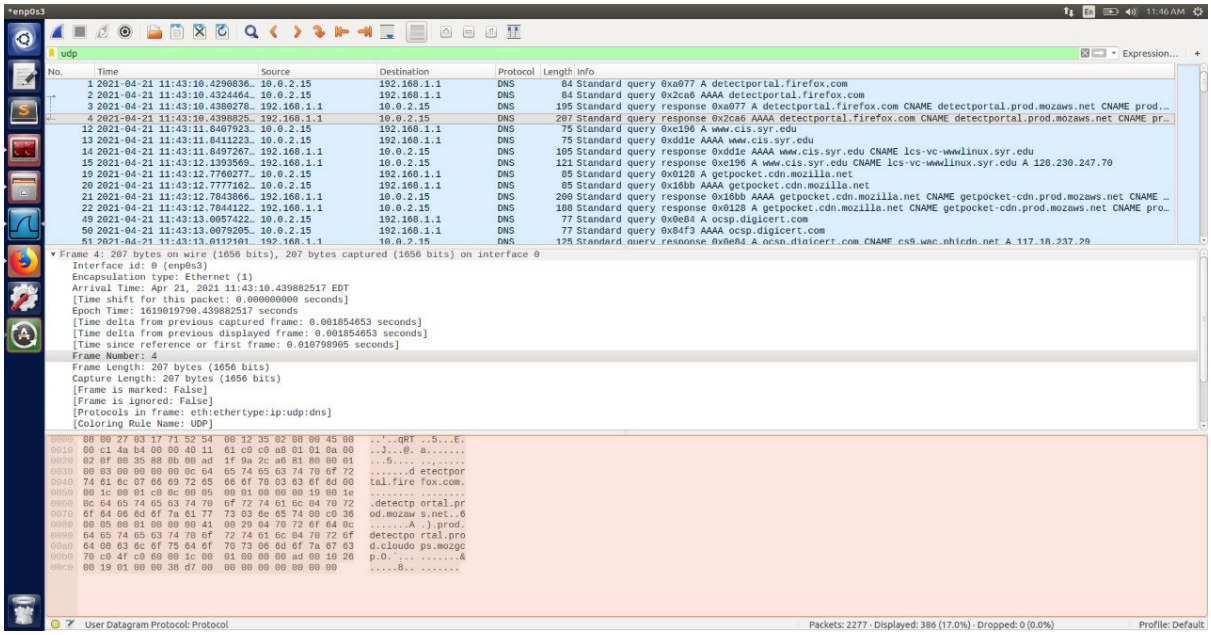


We can see that 1891 packets out of 2277 captured packets are TCP protocol.

i.e. 83% of packets follow TCP protocol.

UDP Packets

Similarly, we can capture the UDP Packets using the display filter **“udp”**



We can see that 386 packets (17% of packets) out of 2277 packets follow UDP Protocol.

Request-Response

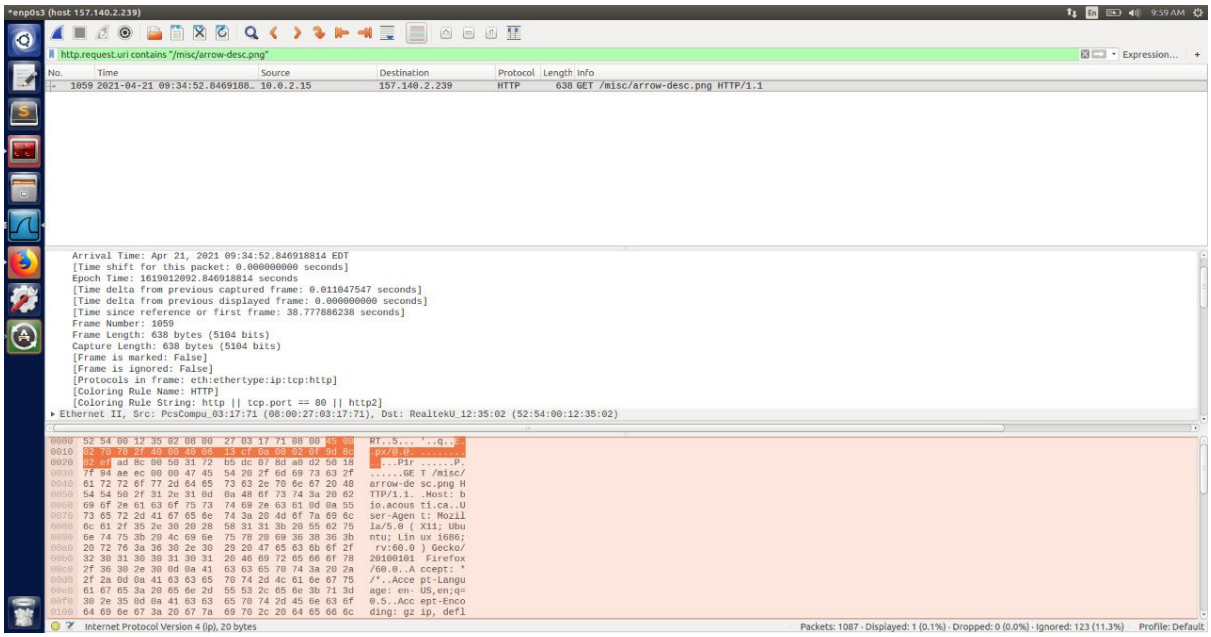
As a part of this experiment, we are also required to capture Request Response pairs. i.e. How many Responses would be there for one particular Request.

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.

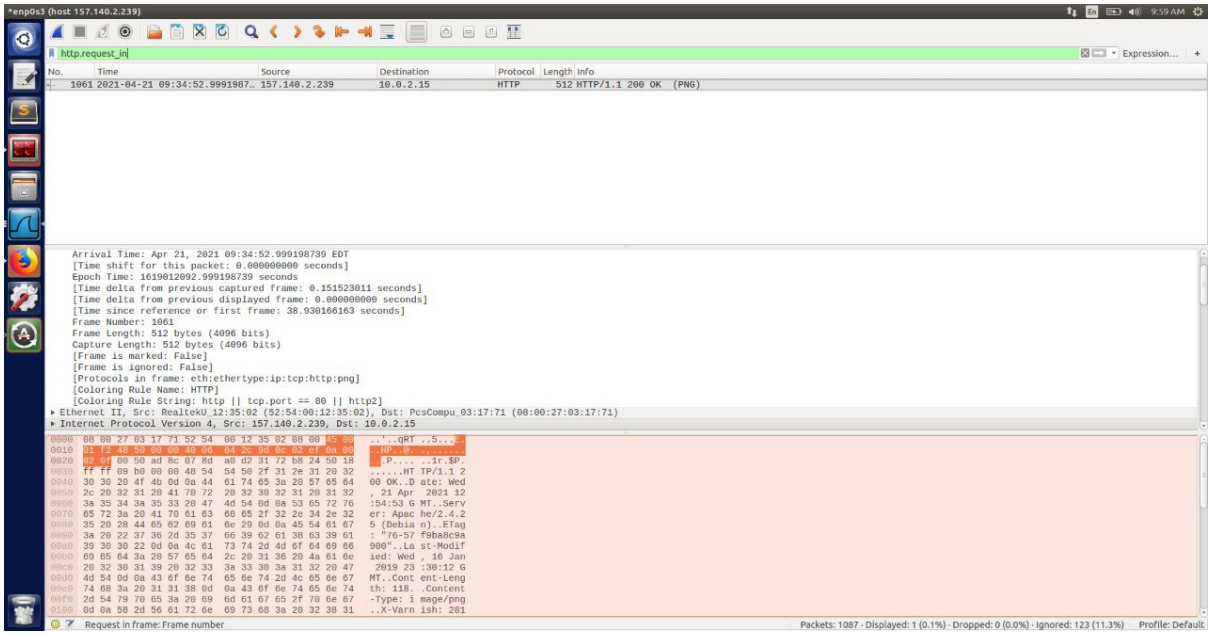
For this experiment, we use a site bio.accousti.ca. IP address is 157.140.2.239. We can capture its data using **“host 157.140.2.239”** capture filter.

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 display the response using **“http.request_in”** display filter



We can now see the singular response for our request.