The Project in total consists of 4 python files :

1. topo-2sw-4h.py
2. Normal_Traffic.py
3. Attack_Traffic.py
4. Congestion_Window.py

Hardware Requirements:

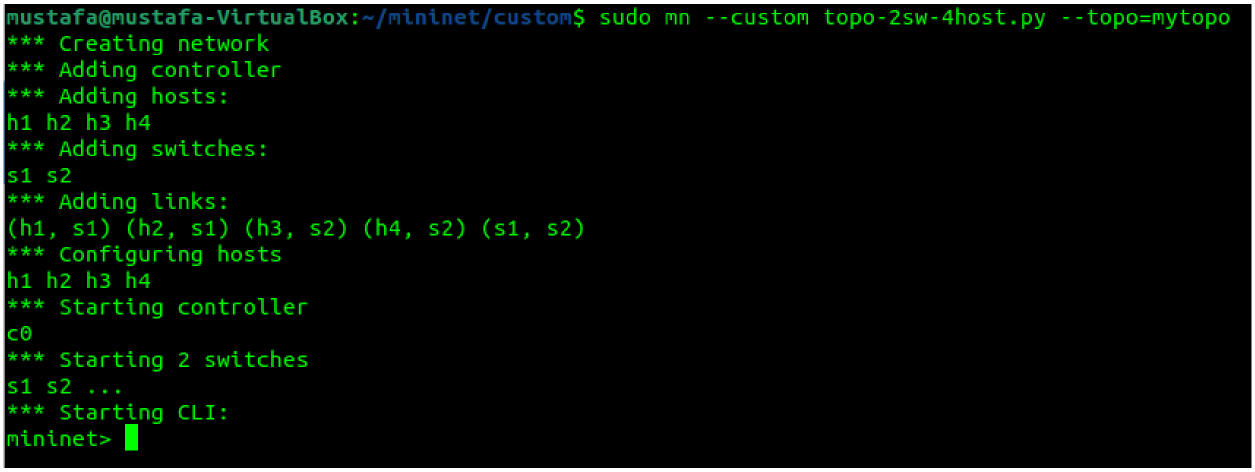
A Machine running Linux natively or on a VM.

Software Requirements:

1. Linux (This project was tested on an Ubuntu 22.04 LTS VM on top of a Machine Running Windows 10)
2. Mininet with Wireshark (We used version 2.3.0)
3. Python3 (comes built in with Linux)

Instructions to run the project:

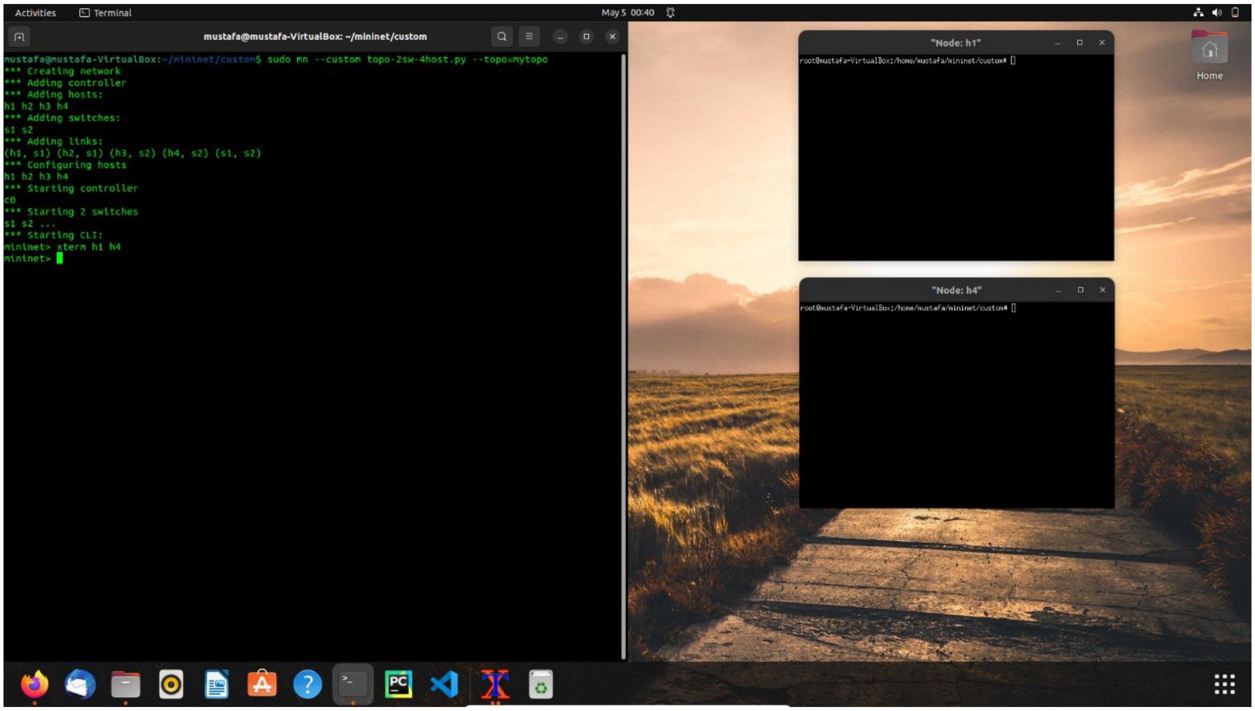
1. The first step is to run the custom topology in Mininet. Copy all 4 above files to the mininet/custom directory. Make sure it is in that directory or else it won't run. Once it is copied , open a terminal in the mininet/custom directory and type the command,
   `sudo mn --custom topo-2sw-4h.py –topo=mytopo`



```
mustafa@mustafa-VirtualBox:~/mininet/custom$ sudo mn --custom topo-2sw-4host.py --topo=mytopo
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> 
```

This will start our mininet topology, alternatively, one can pass the 'pingall' command to test the topology.

2. The 2nd step is to start the flow of traffic from one host to another host. For our project, we selected to send traffic from host h4 to host h1. To do this, we first we need individual terminals for each node, to do this pass the command,
   xterm h1 h4



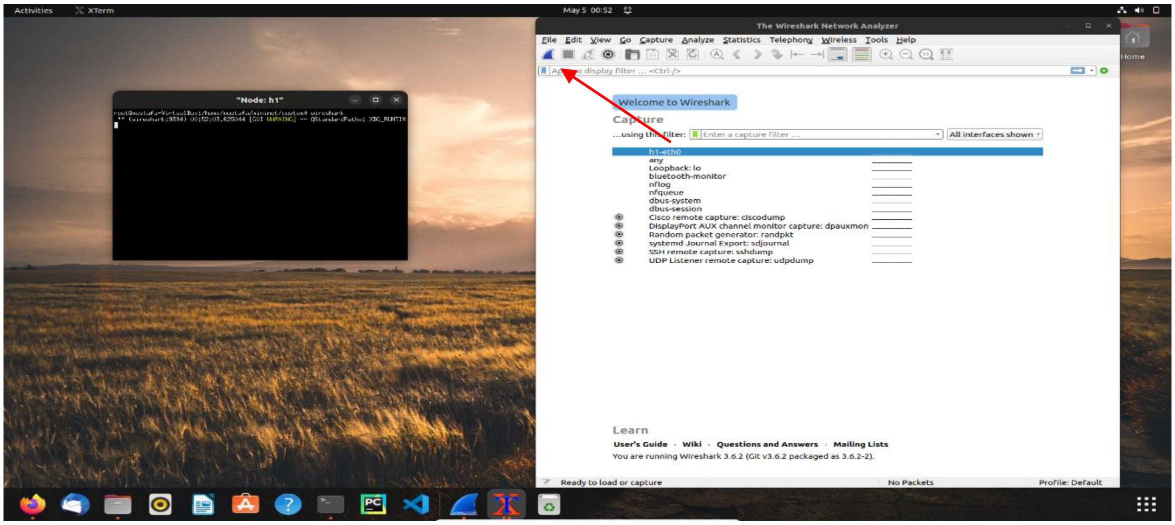
```
mustafa@mustafa-VirtualBox:~/mininet/custom$ sudo mn --custom topo-2sw-4host.py --topo=mytopo
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> xterm h1 h4
mininet> 
```

```
root@mustafa-VirtualBox:/home/mustafa/mininet/custom# 
```

```
root@mustafa-VirtualBox:/home/mustafa/mininet/custom# 
```

We will get 2 xterm windows, one for host h1 and one for host h4.

3. For packet Analysis, we will use Wireshark, we launch Wireshark for host h1 by typing in the h1 terminal command wireshark



Activities
XTerm
May 5 00:52
"Node: h1"
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Welcome to Wireshark
Capture
…using this filter:
Enter a capture filter …
All interfaces shown
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
Home

```
root@mustafa-VirtualBox:/home/mustafa/mininet/custom# wireshark
 ** (wireshark:9394) 00:52:03,625044 [GUI WARNING] -- QStandardPaths: XDG_RUNTIM
```

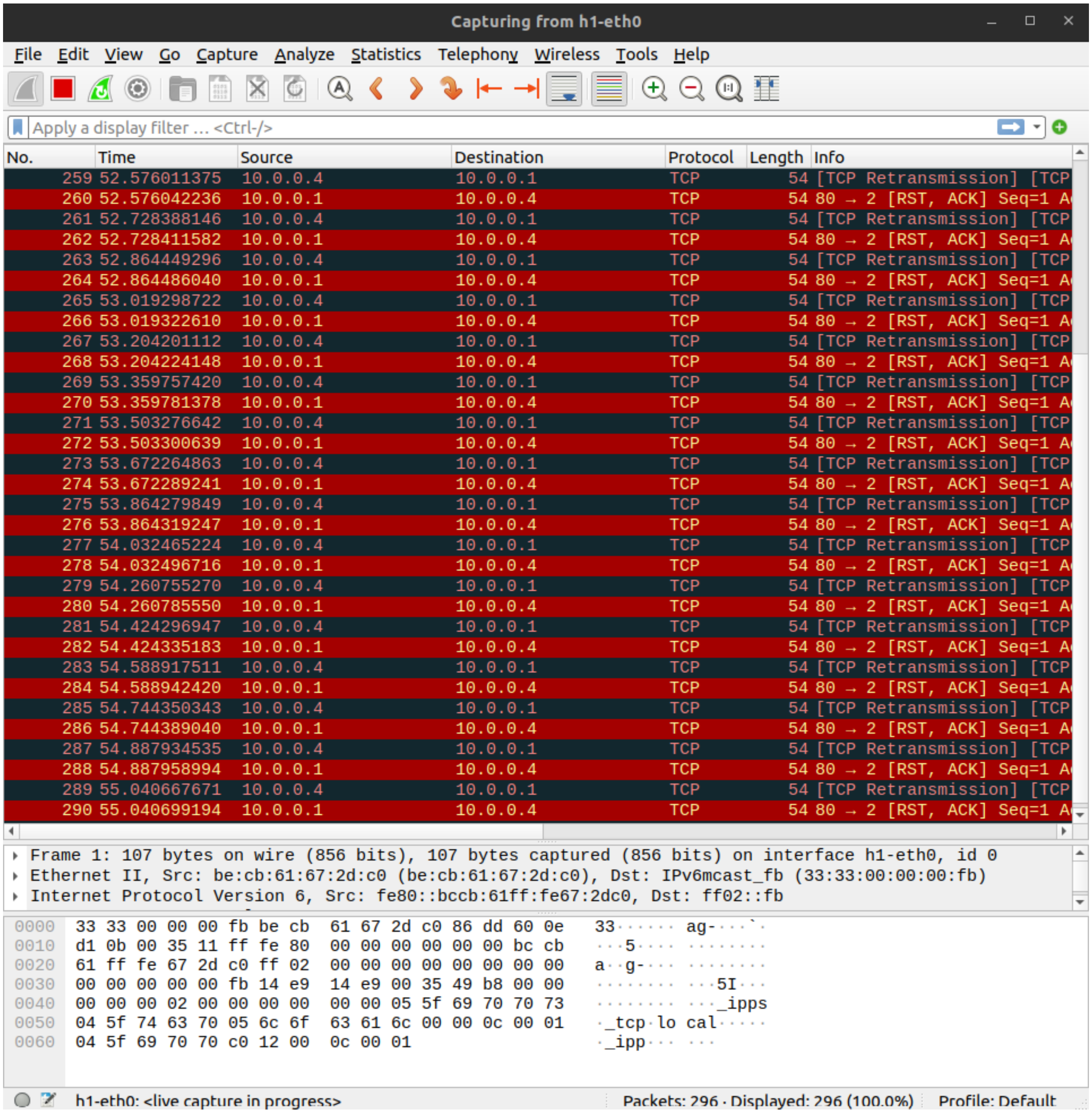
h1-eth0
any
Loopback: lo
bluetooth-monitor
nflog
nfqueue
dbus-system
dbus-session
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump

You are running Wireshark 3.6.2 (Git v3.6.2 packaged as 3.6.2-2).

Ready to load or capture   No Packets   Profile: Default

Once wireshark launches, start packet capture by pressing the blue button on top-left (pointed by the arrow in the image above).

4. Navigate back to the h4 terminal window, and now run the Normal_Traffic.py file to see normal traffic flow by using the command python3 Normal_Traffic.py then enter “10.0.0.4” as the source and “10.0.0.1” as the destination. (don’t include double quotes)
5. The packets start getting sent and we can observe in Wireshark that packets are captured:


Capturing from h1-eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

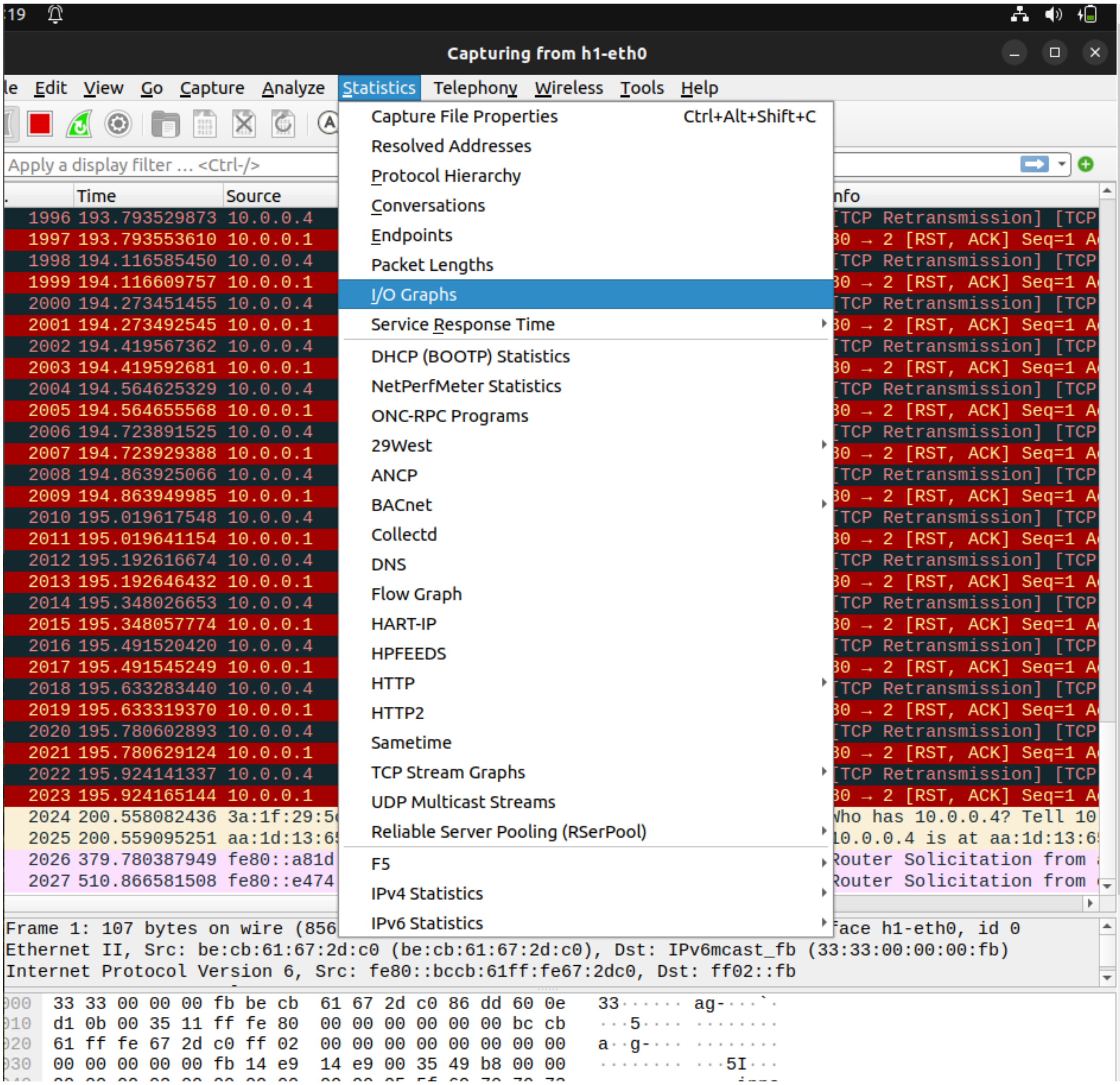
```
No.       Time            Source           Destination      Protocol Length Info
      259 52.576011375   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      260 52.576042236   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      261 52.728388146   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      262 52.728411582   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      263 52.864449296   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      264 52.864486040   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      265 53.019298722   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      266 53.019322610   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      267 53.204201112   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      268 53.204224148   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      269 53.359757420   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      270 53.359781378   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      271 53.503276642   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      272 53.503300639   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      273 53.672264863   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      274 53.672289241   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      275 53.864279849   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      276 53.864319247   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      277 54.032465224   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      278 54.032496716   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      279 54.260755270   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      280 54.260785550   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      281 54.424296947   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      282 54.424335183   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      283 54.588917511   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      284 54.588942420   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      285 54.744350343   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      286 54.744389040   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      287 54.887934535   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      288 54.887958994   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
      289 55.040667671   10.0.0.4         10.0.0.1         TCP        54 [TCP Retransmission] [TCP
      290 55.040699194   10.0.0.1         10.0.0.4         TCP        54 80 → 2 [RST, ACK] Seq=1 A
```

```
› Frame 1: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface h1-eth0, id 0
› Ethernet II, Src: be:cb:61:67:2d:c0 (be:cb:61:67:2d:c0), Dst: IPv6mcast_fb (33:33:00:00:00:fb)
› Internet Protocol Version 6, Src: fe80::bccb:61ff:fe67:2dc0, Dst: ff02::fb
```

```
0000  33 33 00 00 00 fb be cb  61 67 2d c0 86 dd 60 0e   33······ ag-···`·
0010  d1 0b 00 35 11 ff fe 80  00 00 00 00 00 00 bc cb   ···5···· ········
0020  61 ff fe 67 2d c0 ff 02  00 00 00 00 00 00 00 00   a··g-··· ········
0030  00 00 00 00 00 fb 14 e9  14 e9 00 35 49 b8 00 00   ········ ···5I···
0040  00 00 00 02 00 00 00 00  00 00 05 5f 69 70 70 73   ········ ···_ipps
0050  04 5f 74 63 70 05 6c 6f  63 61 6c 00 00 0c 00 01   ·_tcp·lo cal·····
0060  04 5f 69 70 70 c0 12 00  0c 00 01                  ·_ipp··· ···
```

h1-eth0: <live capture in progress> Packets: 296 · Displayed: 296 (100.0%) Profile: Default

6. Go to statistics tab on the toolbar and select IOGraph:

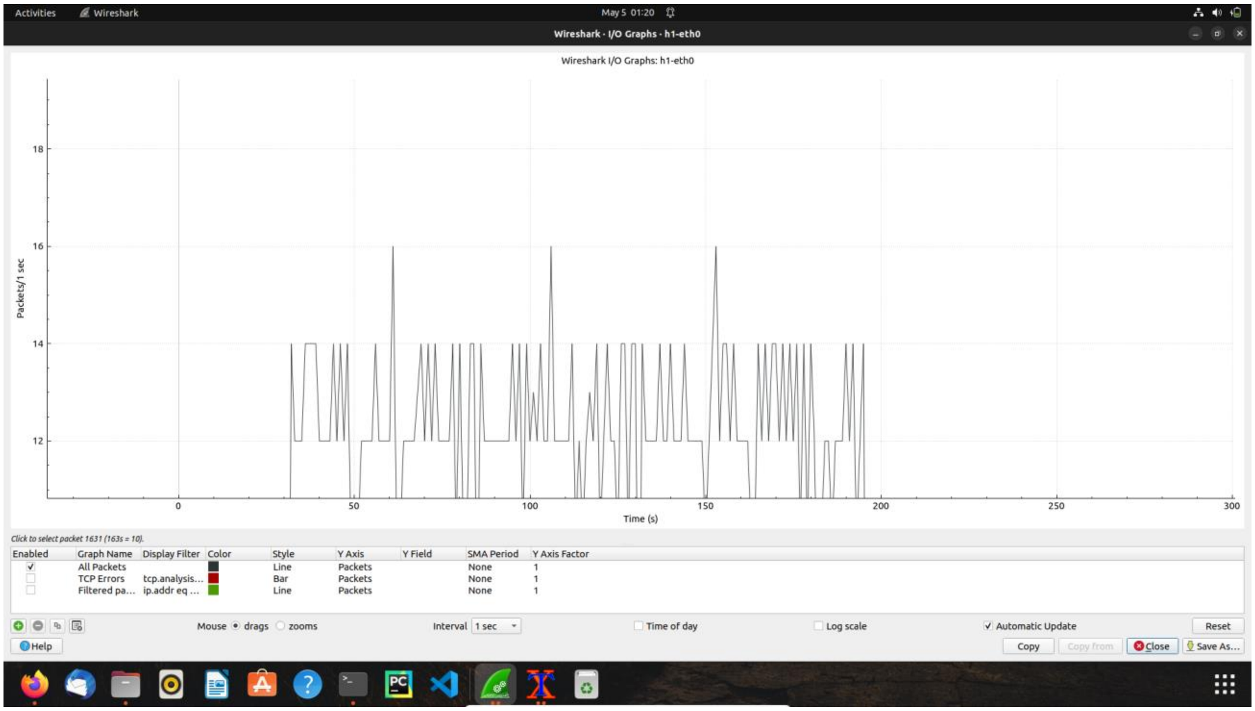


```
1996 193.793529873 10.0.0.4      [TCP Retransmission] [TCP
1997 193.793553610 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
1998 194.116585450 10.0.0.4      [TCP Retransmission] [TCP
1999 194.116609757 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
2000 194.273451455 10.0.0.4      [TCP Retransmission] [TCP
2001 194.273492545 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
2002 194.419567362 10.0.0.4      [TCP Retransmission] [TCP
2003 194.419592681 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
2004 194.564625329 10.0.0.4      [TCP Retransmission] [TCP
2005 194.564655568 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
2006 194.723891525 10.0.0.4      [TCP Retransmission] [TCP
2007 194.723929388 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
2008 194.863925066 10.0.0.4      [TCP Retransmission] [TCP
2009 194.863949985 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
2010 195.019617548 10.0.0.4      [TCP Retransmission] [TCP
2011 195.019641154 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
2012 195.192616674 10.0.0.4      [TCP Retransmission] [TCP
2013 195.192646432 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
2014 195.348026653 10.0.0.4      [TCP Retransmission] [TCP
2015 195.348057774 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
2016 195.491520420 10.0.0.4      [TCP Retransmission] [TCP
2017 195.491545249 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
2018 195.633283440 10.0.0.4      [TCP Retransmission] [TCP
2019 195.633319370 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
2020 195.780602893 10.0.0.4      [TCP Retransmission] [TCP
2021 195.780629124 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
2022 195.924141337 10.0.0.4      [TCP Retransmission] [TCP
2023 195.924165144 10.0.0.1      30 → 2 [RST, ACK] Seq=1 A
2024 200.558082436 3a:1f:29:5    Who has 10.0.0.4? Tell 10
2025 200.559095251 aa:1d:13:6    10.0.0.4 is at aa:1d:13:6
2026 379.780387949 fe80::a81d    Router Solicitation from
2027 510.866581508 fe80::e474    Router Solicitation from
```

```
Frame 1: 107 bytes on wire (856      face h1-eth0, id 0
Ethernet II, Src: be:cb:61:67:2d:c0 (be:cb:61:67:2d:c0), Dst: IPv6mcast_fb (33:33:00:00:00:fb)
Internet Protocol Version 6, Src: fe80::bccb:61ff:fe67:2dc0, Dst: ff02::fb
```

```
000  33 33 00 00 00 fb be cb  61 67 2d c0 86 dd 60 0e   33······ ag-···`·
010  d1 0b 00 35 11 ff fe 80  00 00 00 00 00 00 bc cb   ···5···· ········
020  61 ff fe 67 2d c0 ff 02  00 00 00 00 00 00 00 00   a··g-··· ········
030  00 00 00 00 00 fb 14 e9  14 e9 00 35 49 b8 00 00   ········ ···5I···
```

We can observe the graph pattern for the normal traffic:



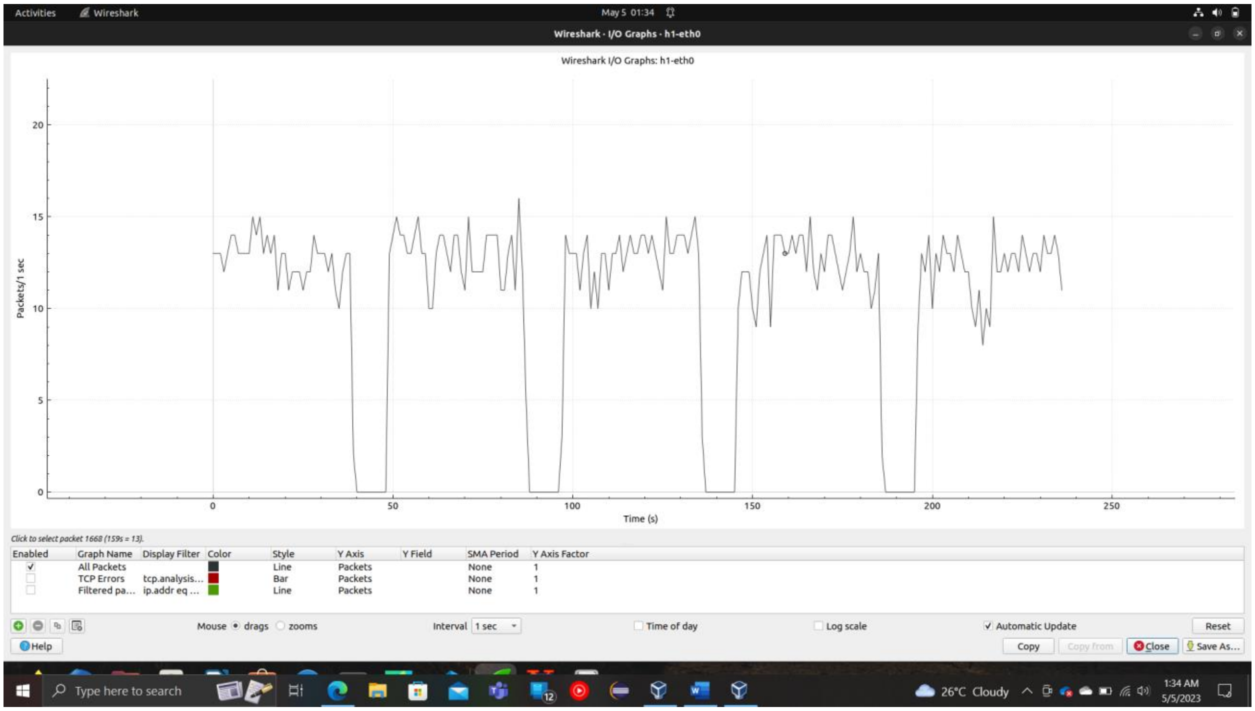
Click to select packet 1631 (163s = 10).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	All Packets		■	Line	Packets		None	1
□	TCP Errors	tcp.analysis…	🟥	Bar	Packets		None	1
□	Filtered pa…	ip.addr eq …	🟩	Line	Packets		None	1

Mouse ● drags ○ zooms Interval 1 sec □ Time of day □ Log scale ☑ Automatic Update

7. Repeat steps 4-6 again but use the command `python3 Attack_Traffic.py`

   We will get the following graph output which demonstrate DDoS attack traffic:

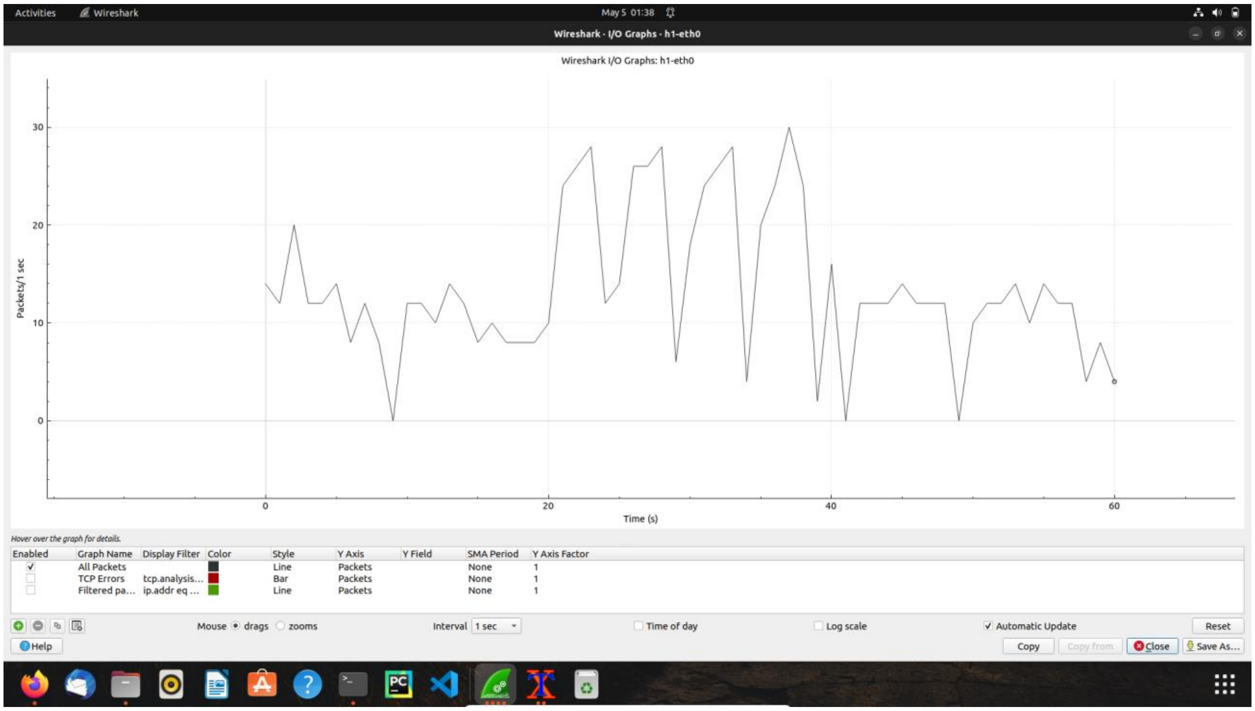


Click to select packet 1668 (159s = 13).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	All Packets		■	Line	Packets		None	1
□	TCP Errors	tcp.analysis…	🟥	Bar	Packets		None	1
□	Filtered pa…	ip.addr eq …	🟩	Line	Packets		None	1

Mouse ● drags ○ zooms Interval 1 sec □ Time of day □ Log scale ☑ Automatic Update

8. Now in terminal h4 , run python3 Congestion_Window.py (this program does not require any source or destination id)
9. Observe the Wireshark graph:



Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	All Packets		■	Line	Packets		None	1
□	TCP Errors	tcp.analysis…	🟥	Bar	Packets		None	1
□	Filtered pa…	ip.addr eq …	🟩	Line	Packets		None	1

Mouse ● drags ○ zooms Interval 1 sec □ Time of day □ Log scale ☑ Automatic Update

10. This concludes the project, quit Mininet by pressing ctrl+z in terminal and (optional) perform a Mininet clean up by using command `sudo mn -c`