Laboratory: 3

Background Traffic

Introduction

This report includes generation of simulated network background traffic by replaying packet traces collected from Clemson Campus network such that it could be used for DDOS attack detection studies. Here, the number of packets generated and the expected duration of the time values are changed in order to make the simulated background traffic as close to the real background traffic. “Tshark” is used to capture the Clemson traffic (either by time duration condition or by using number of packets duration) and “Tcpreplay” suite is used from open-source utilities for replaying the captured network traffic for creating simulated network traffic.

For this, we set up an environment as described below and ssh to snifffer to capture the campus packets.

Some similar tools like Tshark are: Cloud Shark, SolarWinds

Setup and Process

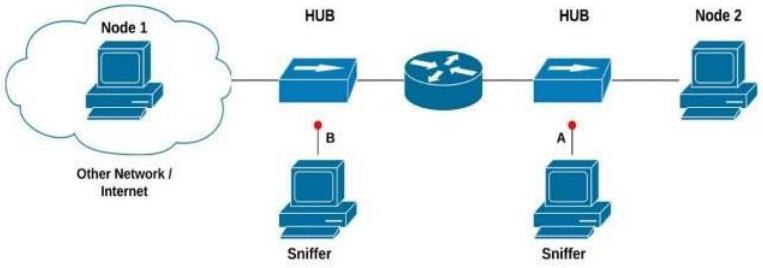

Figure1: Network set-up

Here, as shown above, node 1 (192.168.10.10) is machine from lab in clemson and node 2 (192.168.20.13)is machine from lab in charleston. All the machines are connected over the HUB and the red dots are thesniffing points (192.168.10.9). Snooping machine that I will be using for replaying the campus traffic and capturing the simulated traffic (192.168.10.111) is connected to the same sub-net.

We login to sniffer machine using ssh to capture the Clemson network traffic using Tshark with the following command (figure: 2)

$ tshark -i enp3s1 -c 10000 -w campus_traffic.pcap -F libpcap

Here, the interface used is “enp3s1” and the capture contains 10000 packets. “campus_traffic.pcap” is

the captured pcap file name of the campus network.

```
[shriyas@clemson9 ~]$ tshark -i enp3s1 -c 10000 -w campus_traffic.pcap -F libpca
p
Capturing on 'enp3s1'
10000
```

Figure 2: Capturing Clemson network

This capture is then copied to spoof machine where the python script for plotting time-series graph is located, by using the following command. (Figure: 3)

scp shriyas@192.168.10.9:~/campus_traffic.pcap .

```
root@cnc:~/lab3scripts# ls
a.txt                       python_setup.sh     scapy_example.py
bittwist.sh                 rand_arr_time.py    setup_bittwist.sh
campus_traffic.csv          rand_arr_time.pyc   shriyas.py
campus_traffic.pcap         README              time_series.py
mengine.py                  read_pcap.py        time_series.pyc
plot_time_series_example.py read_pcap.pyc
```

Figure 3: campus_traffic in spoof machine

Now, before replaying this traffic we start the capture to listen in interface lo in spoof machine to get the packets from the simulated network. Once the capture is started, we use the following command to reply the campus traffic is spoof machine. (Figure: 4)

tcpreplay --intf1=lo campus_traffic.pcap

```
root@cnc:~/lab3scripts# tcpreplay --intf1=lo campus_traffic.pcap
Warning in sendpacket.c:sendpacket_open_pf() line 669:
Unsupported physical layer type 0x0304 on lo.  Maybe it works, maybe it wont.  S
ee tickets #123/318
sending out lo
processing file: campus_traffic.pcap
Actual: 10000 packets (10960552 bytes) sent in 7.68 seconds.           Rated: 1
427155.2 bps, 10.89 Mbps, 1302.08 pps
Statistics for network device: lo
        Attempted packets:         10000
        Successful packets:        10000
        Failed packets:            0
        Retried packets (ENOBUFS): 0
        Retried packets (EAGAIN):  0
```

Figure 4: Replaying campus traffic

Here, we can see all the 10000 packets captured previously are replayed successfully. This replayed traffic is captured on loopback interface by using the following command. I captured for 15000 packets in my case. Capturing is shown below. (Figure: 5)

```
root@cnc:~# tshark -i lo -c 15000 -w replay.pcap -F libpcap
Running as user "root" and group "root". This could be dangerous.
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:44: dofile has been disabled due to ru
nning Wireshark as superuser. See https://wiki.wireshark.org/CaptureSetup/Captur
ePrivileges for help in running Wireshark as an unprivileged user.
Capturing on 'Loopback'
9307

^C
1375 packets dropped
root@cnc:~#
```

Figure 5: Capturing replayed traffic

We can see above that some packets are dropped. At this point, looking at the captured packet number and packets dropped we can know that this simulated traffic is already different from the real captured traffic. For a better understanding of how different they are, time-series graph is plotted for the real traffic and replayed traffic on different arriving time algorithm as shown below:

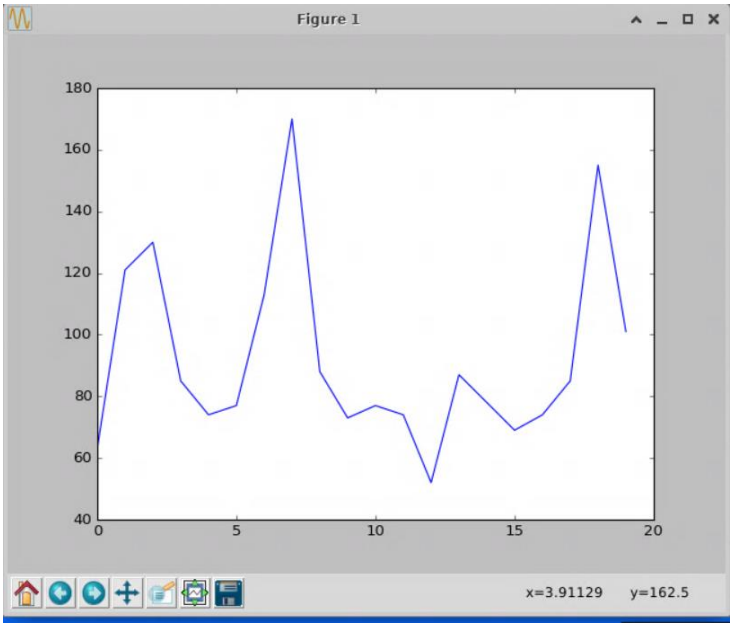

Figure 6: Time-series graph for real campus Traffic

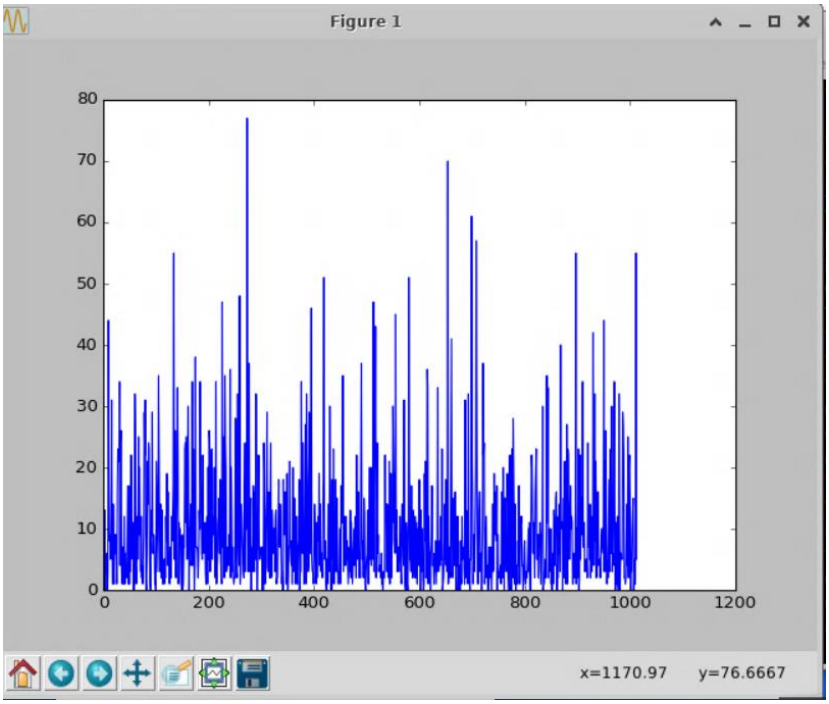

Figure 6: Time-series graph for simulated Traffic at 3rd (poisson) arriving time algorithm

Here, in the above graph,3rd arriving time algorithm from the code is used for plotting simulated traffic with 10000 number of packets and expected duration of 1000 seconds. (Figure: 7)

```
def example2_plot_time_series_from_simulated_data():
    """
    This is an example usage of rand_arr_time.py
    rand_arr_time.py generate a random integer list
    """
    option=3                              # There are 6 options in total, each us
e different method to generate random arriving times
    Number_of_packets=10000                # Number of packets generated
    expected_duratation=1000                 # Expected length of time in seconds
    arrive_time=rand_arr_time.rand_arr_time(option,Number_of_packets,expected_du
ratation)  # Get packet arrive time, with option 3, 100000 packets, expected in
1000 seconds.
    time_series.plot_time_series(arrive_time)   # Plot time series using packets
 arrive time
```

Figure 7: Meta details of the time-series graph in figure 6

Below is the time-series graph for simulated Traffic at 4th arriving time algorithm from the code with 10000 number of packets and expected duration of 1000 seconds. (Figure: 8)

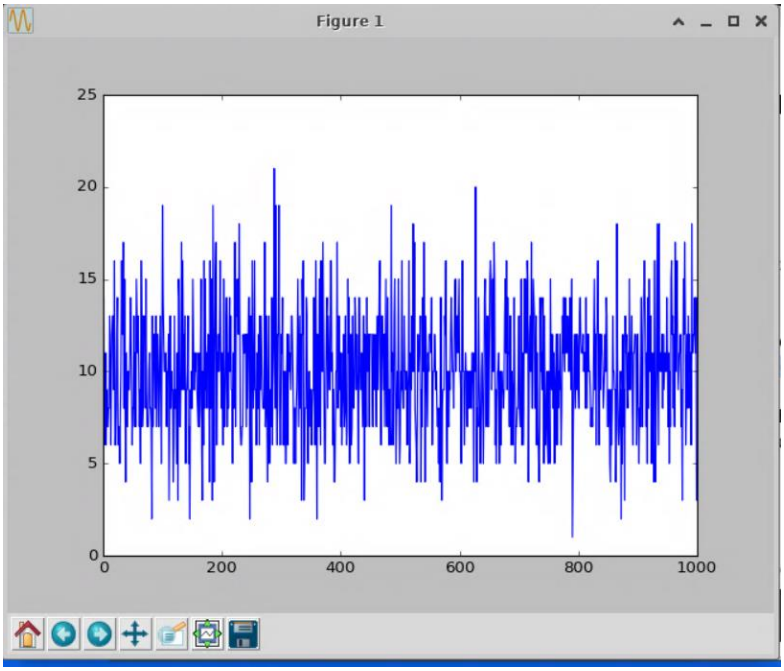

Figure 8: Time-series graph for simulated Traffic at 4rd (exponential) arriving time algorithm

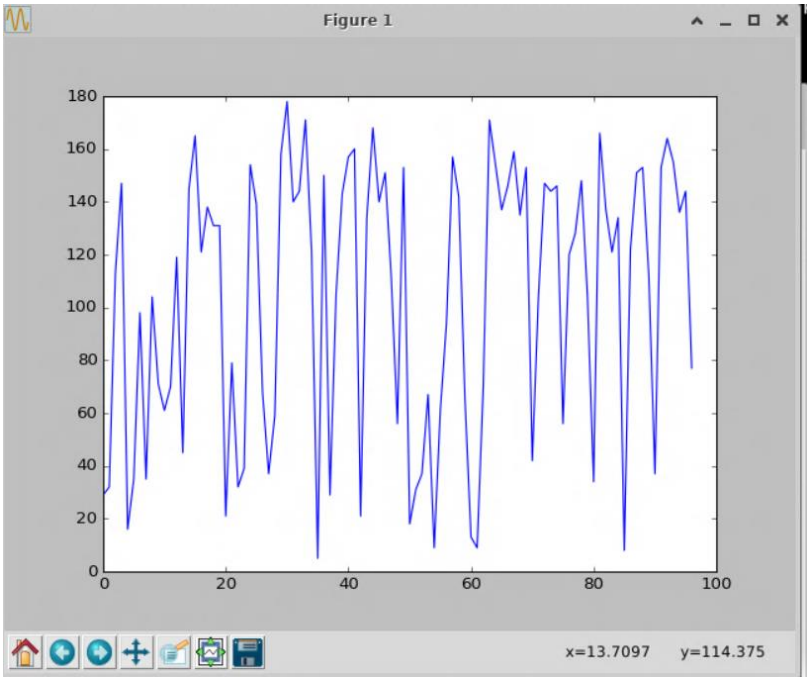

Figure 9: Time-series graph for simulated Traffic at 6rd (random) arriving time algorithm

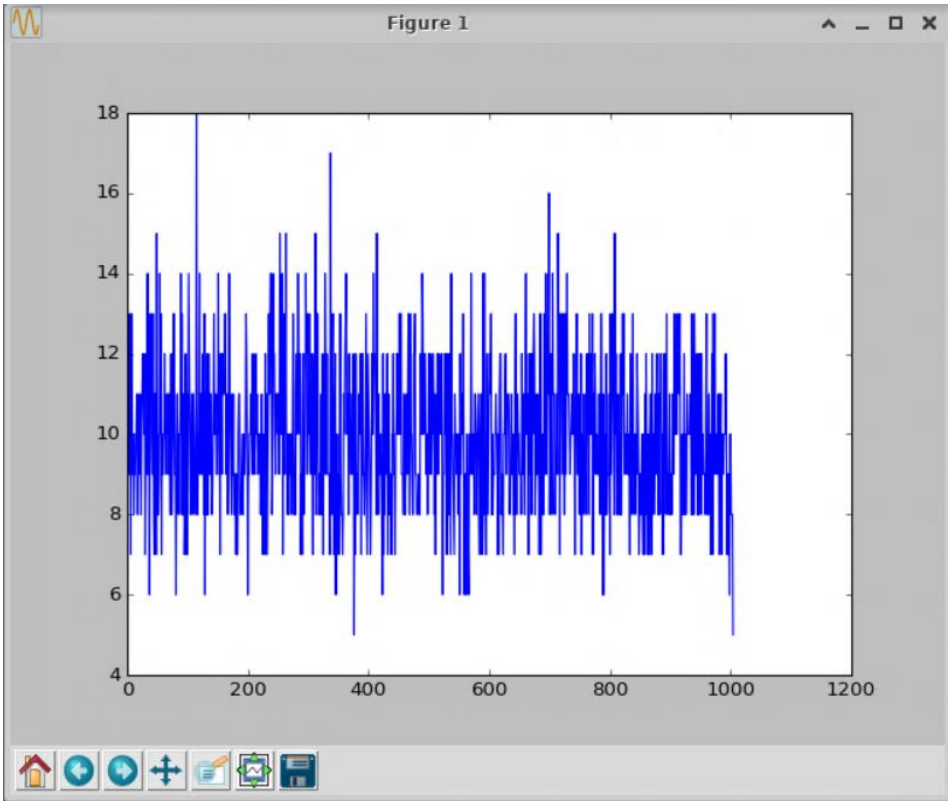

Figure 10: Time-series graph for simulated Traffic at 1st (Uniform) arriving time algorithm

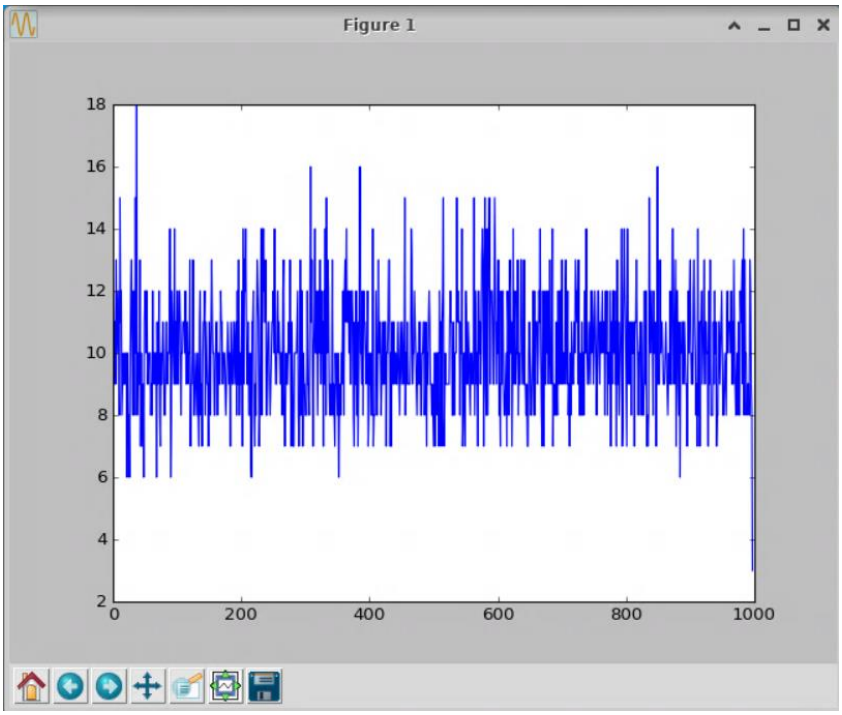

Figure 11: Time-series graph for simulated Traffic at 2st (tnorm) arriving time algorithm

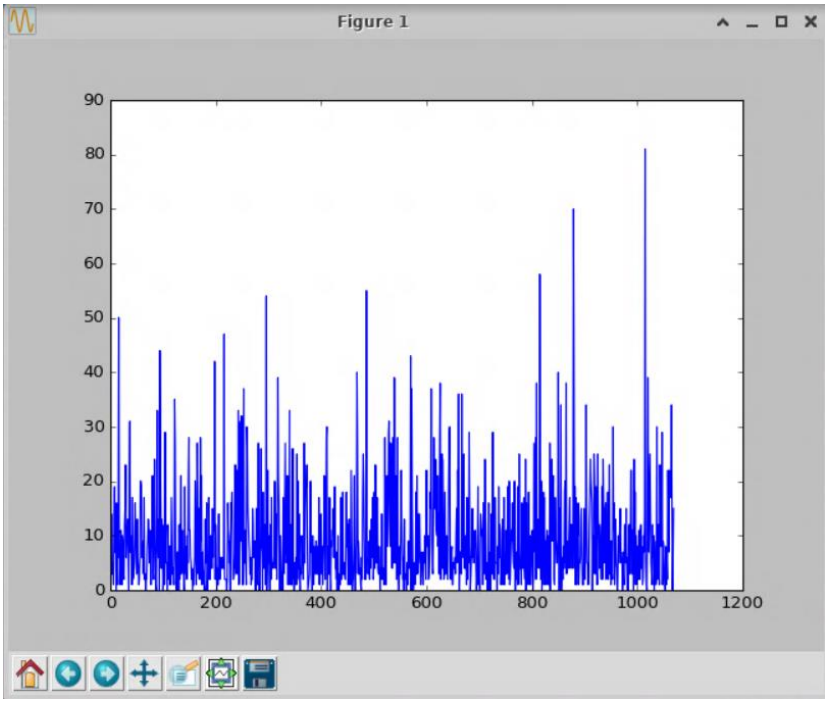

Figure 12: Time-series graph for simulated Traffic at 5st (poisson with random expectation of interval) arriving time algorithm

Now, I've used the following values to get the simulated traffic as close to the real traffic. (Figure: 13)

```
    option=6                            # There are 6 options in total, each use
different method to generate random arriving times
    Number_of_packets=2000                # Number of packets generated
    expected_duratation=100                 # Expected length of time in seconds
    arrive_time=rand_arr_time.rand_arr_time(option,Number_of_packets,expected_du
ratation)  # Get packet arrive time, with option 3, 100000 packets, expected in
1000 seconds.
    time_series.plot_time_series(arrive_time)   # Plot time series using packets
 arrive time
```

Figure 13: Meta details of the time-series graph in figure 14

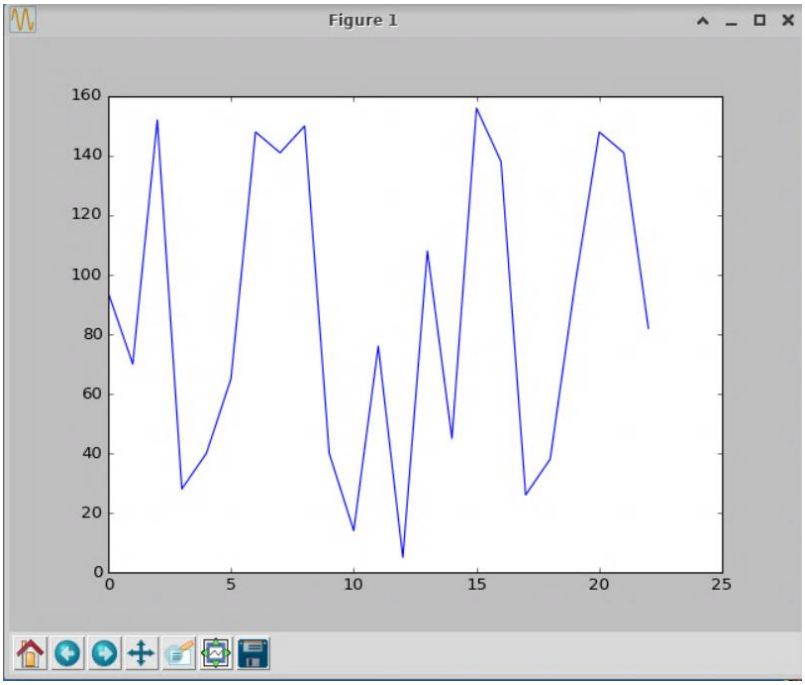

Figure 14: Time-series graph for simulated traffic close to real background traffic

Questions

1. Plot the background traffic datasets generated/used in three scenarios where x-axis shows time, and the y-axis shows the number of packets received. Compare the figures and discuss their differences.

Plotted graphs for 3 different background traffic datasets scenarios are shown below:

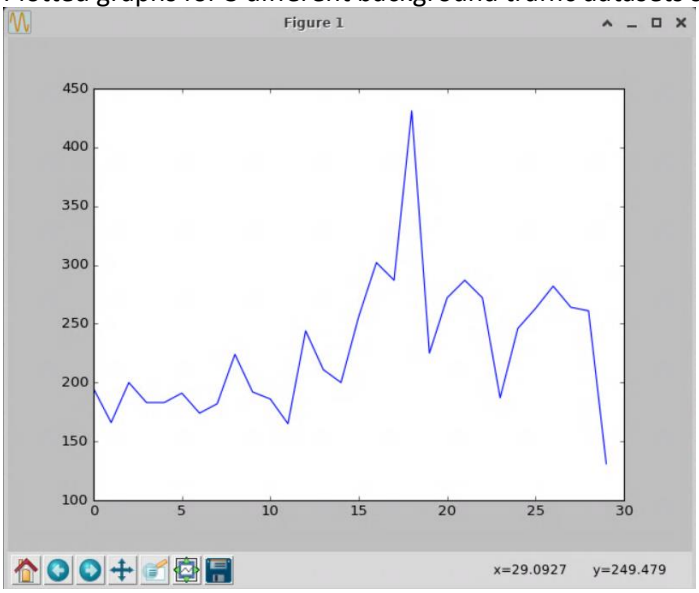

Figure 15: Scenario 1

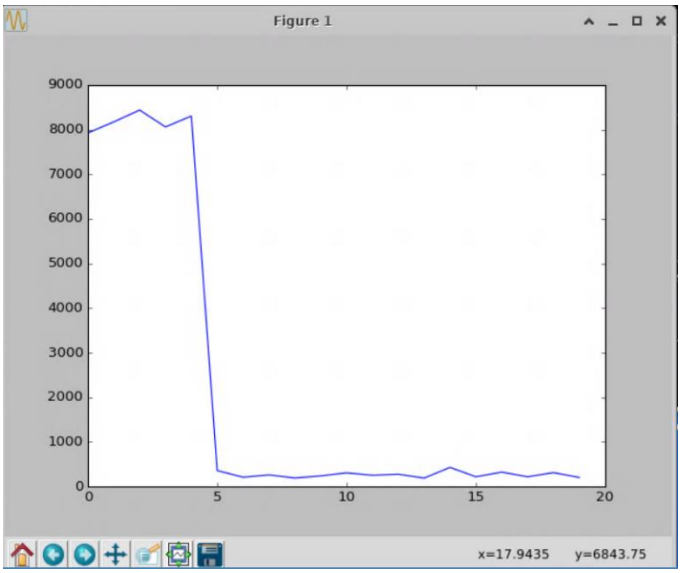

Figure 16: Scenario 2

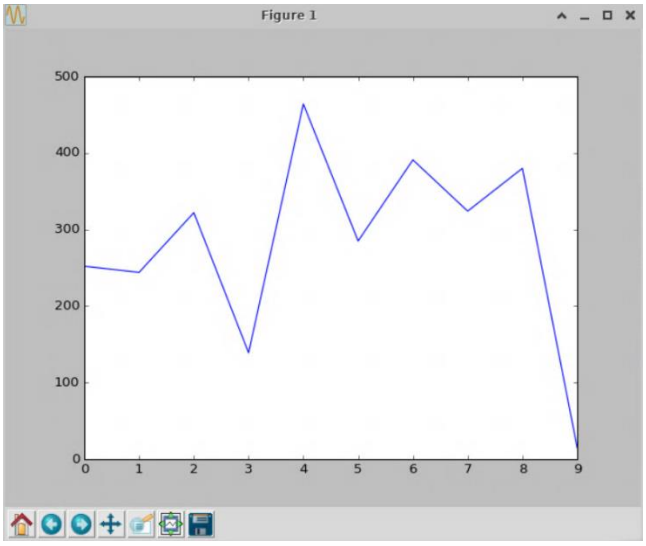

Figure 17: Scenario 3

By analyzing the above graphs, we can see that, in scenario 1 the time duration of the traffic flow captured is 30 seconds and the highest number of packet flow is seen between 15 to 20 seconds of the capture but the over all flow of traffic is moderate and there is no major drop in the flow of packets over the captured network and the last packet is transferred on 29.0927 second. Also, the total number of packets captured are 6862.

Whereas, in scenario 2, we can see the time duration of the traffic flow captured is 20 seconds and the number of packets capture is much more than scenario 1 and scenario 3 (which is 44861). The highest number of packet flow here is seen between 0 to 5 seconds and a major drop in the flow of packets over the capture is seen on 5th second. The last packet transferred here is on 17.9435 second.

Now in scenario 3, the time duration of the traffic flow captured is 9 seconds and the number of packets captured are 2816. The highest number of packet flow here is seen at 4th second. And the major drop in

the flow of packets over the capture is seen between 8 and 9 seconds. The last packet transferred here is on 9th second.

2. Network background traffic statistics generation script (plot_time_series_example.py) generates datasets using different probability distribution functions (PDF). Compare datasets generated using different PDFs. Discuss the difference between datasets and the data-set converted from operational network data.

Datasets generated using tnorm, poisson PDFs and data-set converted from operational network data is shown below:

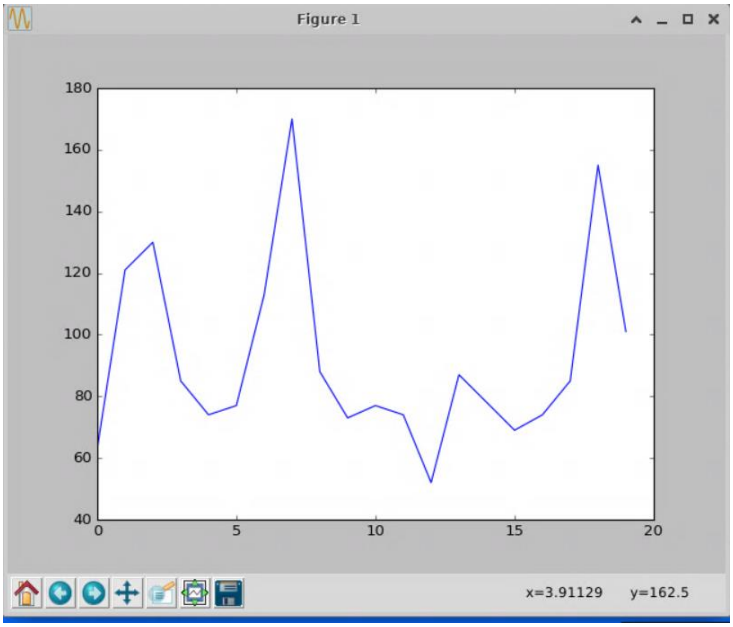

Figure 18: Data-set converted from operational network data.

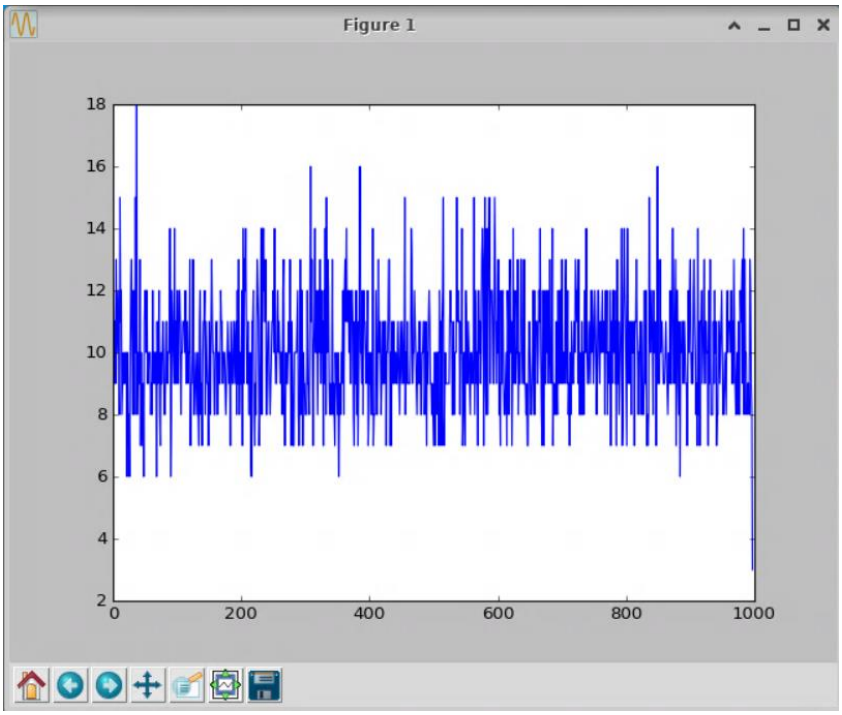

Figure 19: Data set generated using Tnorm probability distribution function

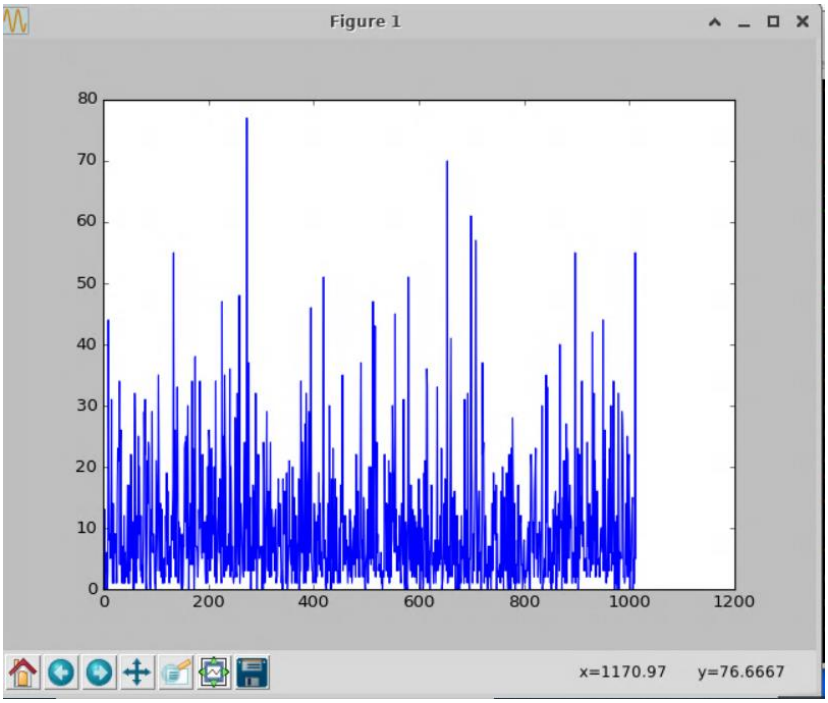

Figure 20: Data set generated using poisson probability distribution function

Here, in figure 18, we can see the time duration of the traffic flow captured is 20 seconds. The highest number of packet flow is seen between 5 to 10 seconds of the over all flow of the capture. All the packets observed lie in the range of 50 to 170. The last packet is transferred on 19th second and the total number of packets captured are 9307.

Whereas in figure 2, the time duration of the traffic flow captured is 1000 seconds and the total number of packets captured that are plotted are 10000. Here, all the packets observed lie in the range of 3 to 18, but majority of the packets are in 6 to 16 range and the highest number of packet flow is seen between 0 to 200 seconds. The last packet is transferred on 1000^{th} second.

Now from figure 3, All the packets observed lie in the range of 0 to 78 and the total number of packets captured are 9307. The time duration of the traffic flow captured is 1000 seconds and the total number of packets captured that are plotted are 10000. The highest number of packet flow is seen between 200 to 400 seconds. The last packet is transferred on 1000^{th} second.

3. What kind of traffic did you run between hosts in the second scenario (Replay a pcap file with tcpreplay). Justify why you think it can represent operational network traffic.

The traffic that was run between the hosts in the second scenario was Simulated network traffic. It is the next best choice for having network traffic for study purposes, first choice being operational network traffic.

It can represent operational network traffic, since the packets captured that were used to generate the replay traffic where from the actual working network. Here, the time of the day, the allowed network services and the average number of users which effects the traffic generated, are from the real-life scenario. Hence, the simulated traffic is as close as it could get to the operational network traffic.

4. You can replay and forward captured pcap files on a link in a controlled network environment to use as background traffic. If you perform a DDoS attack on this link, you can observe the effects of the DDoS attack without jeopardizing the operational network. However, some of the effects cannot be observed with replayed/forwarded background traffic. List some of these effects and explain the reason why they can not be observed.

One of the effects that cannot be observed with replayed/forwarded background traffic is, responsive behavior in the traffic, for example, SYN attack cannot be observed since there would be no response from the victim in replayed traffic, and thus the half way handshakes are not achieved. This is due to the reason that, replayed traffic will blindly transmit data in a preconfigured sequence and doesn't adapt to prevailing traffic conditions.

One of the other effects that cannot be observed is network-wide coherence. This is due to reason that replayed traffic gets too sparse or too congested at times which results in the inability to produce the network-wide coherence required for a few observation points to sense shifts in traffic patterns.

5. Number of packets received by a node on the network is one of the popular metrics used in DDoS detection applications. In this assignment, we focused on packet count statistics. Discuss what other metrics that can be used for DDoS detection.

The following metrics can be used for DDoS detection: Hartley entropy, Shannon entropy, Renyi's entropy and Generalized entropy. These metrics can be used to describe characteristics of network traffic and an appropriate metric facilitates building an effective model to detect DDoS attacks.

In general, here, when the difference between entropy of flow count and mean value of entropy is greater than the threshold value that is updated adaptively based on traffic pattern, DDoS attack could be detected.

References:

[1] http://cs.uccs.edu/~jkalita/papers/2014/BhuyanMonowarIC32014.pdf
[2] https://pdf.sciencedirectassets.com/280203/
[3] http://cseweb.ucsd.edu/~kvishwanath/papers/swingusenix08.pdf