Topic : Study the impact of initial congestion window and TCP window scaling using Flent .

Timeline of the project :

1) First we learnt how to use github and became familiar to it.
2) We studied about TCP congestion and Window Scaling from HPBN Textbook.
3) Installed flent and ran some test cases(rrul test, tcp_bidirectional, tcp_upload).
4) Studied establishing test bed and its components involved .
5) Studied the how to change the initial congestion window and enable and disable tcp window scaling.
6) Finally we ran the test cases for different initial congestion window values with TCP window scaling enabled and disabled respectively.
7) At last, we analysed the ellipses graph obtained in flent gui.

Steps involved in Flent Installation :

1. Packages Required:
All required packages you can install in one command only
sudo apt-get install python-dev python-setuptools python-matplotlib python-pip pyqt4-dev-tools netperf

2. Download FLENT:
Download latest version from github
https://github.com/tohojo/flent

3. Installation:

i] Unzip downloaded folder

```
unzip flent-master.zip
```

ii] Install

```
cd flent-master
make
sudo make install
```

Steps to run FLENT:

You must run netperf on two computers - server and client.

1. Server: Netperf needs to be started in “server mode” to listen for commands from the Client. To do this, install netperf on the Server computer, then enter:

```
netserver &
```

Note: Instead of installing netperf on a local server, you may substitute the netserver that is running on netperf.bufferbloat.net by using -H netperf.bufferbloat.net in the commands below.

2. Client: Install netperf, then install flent on your Client computer. When you invoke flent on the Client, it will connect to the specified netserver (-H) and carry out the measurements. Here are some useful commands:

a) RRUL: Create the standard graphic image used by the Bufferbloat project to show the down/upload speeds plus latency in three separate charts:

```
flent rrul -p all_scaled -l 60 -H 10.1.1.1 -t RRUL_test -o RRUL.png
```

b) CDF: A Cumulative Distribution Function plot showing the probability that ping times will be below a bound:

```
flent rrul -p ping_cdf -l 60 -H 10.1.1.1 -t CDF_test -o CDF.png
```

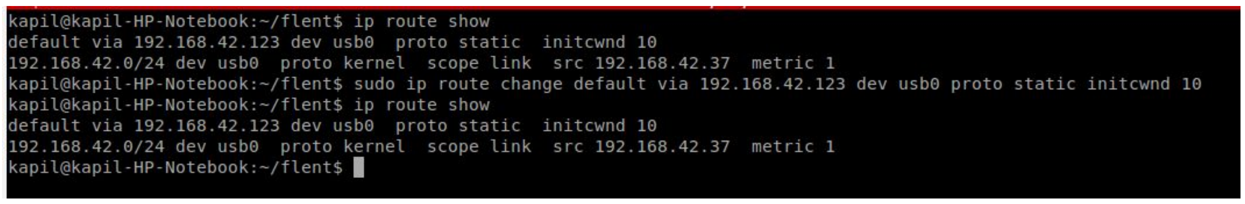
c) TCP Upload: Displays TCP upload speed and latency in two charts:

```
flent tcp_upload -p totals -l 60 -H 10.1.1.1 -t TCP_upload_test -o
tcp_upload.png
```

d) TCP Download: Displays TCP download speeds and latency in two charts:

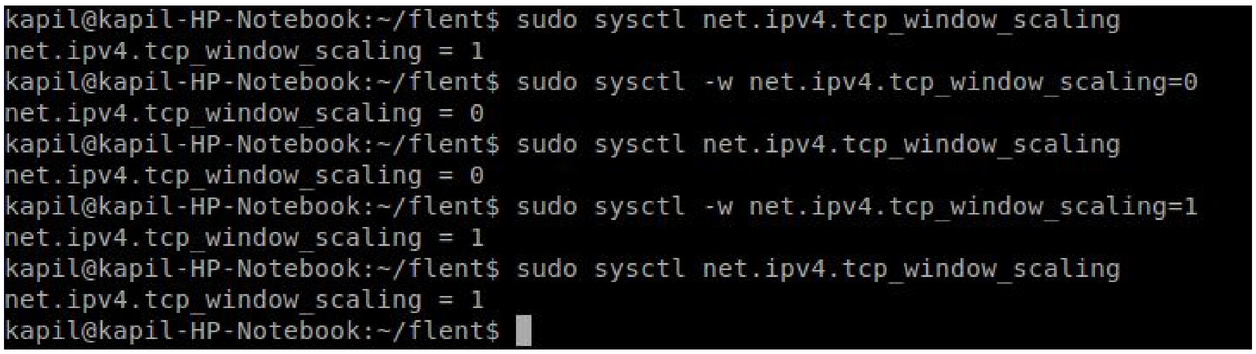
flent tcp_download -p totals -l 60 -H 10.1.1.1 -t TCP_download_test -o tcp_download.png
The output of each of these commands is a graphic (PNG) image along with a data file in the current working directory that can be used to re-create the plot, either from the command line, or by loading them into the GUI.
Run flent-gui to start the GUI

Commands involved in changing Initial Congestion Window :
Adjusting the value of the initcwnd setting on Linux is simple. Assuming we want to set it to 10:
Step 1: check route settings.
Make a note of the line starting with default.
Step 2: Change the default settings.
Paste the current settings for default and add *initcwnd 10* to it.
Step 3: Verify



```
kapil@kapil-HP-Notebook:~/flent$ ip route show
default via 192.168.42.123 dev usb0  proto static  initcwnd 10
192.168.42.0/24 dev usb0  proto kernel  scope link  src 192.168.42.37  metric 1
kapil@kapil-HP-Notebook:~/flent$ sudo ip route change default via 192.168.42.123 dev usb0 proto static initcwnd 10
kapil@kapil-HP-Notebook:~/flent$ ip route show
default via 192.168.42.123 dev usb0  proto static  initcwnd 10
192.168.42.0/24 dev usb0  proto kernel  scope link  src 192.168.42.37  metric 1
kapil@kapil-HP-Notebook:~/flent$
```

Commands involved in enabling and disabling the TCP Window Scaling:
To check the current status we use the first command and to enable it set it to 1 and to disable it set it to 0 as shown below :-



```
kapil@kapil-HP-Notebook:~/flent$ sudo sysctl net.ipv4.tcp_window_scaling
net.ipv4.tcp_window_scaling = 1
kapil@kapil-HP-Notebook:~/flent$ sudo sysctl -w net.ipv4.tcp_window_scaling=0
net.ipv4.tcp_window_scaling = 0
kapil@kapil-HP-Notebook:~/flent$ sudo sysctl net.ipv4.tcp_window_scaling
net.ipv4.tcp_window_scaling = 0
kapil@kapil-HP-Notebook:~/flent$ sudo sysctl -w net.ipv4.tcp_window_scaling=1
net.ipv4.tcp_window_scaling = 1
kapil@kapil-HP-Notebook:~/flent$ sudo sysctl net.ipv4.tcp_window_scaling
net.ipv4.tcp_window_scaling = 1
kapil@kapil-HP-Notebook:~/flent$
```

Commands to run test case and check result in Flent GUI :

./run-flent rrul-l 60 -H 172.162.10.2 -t e1

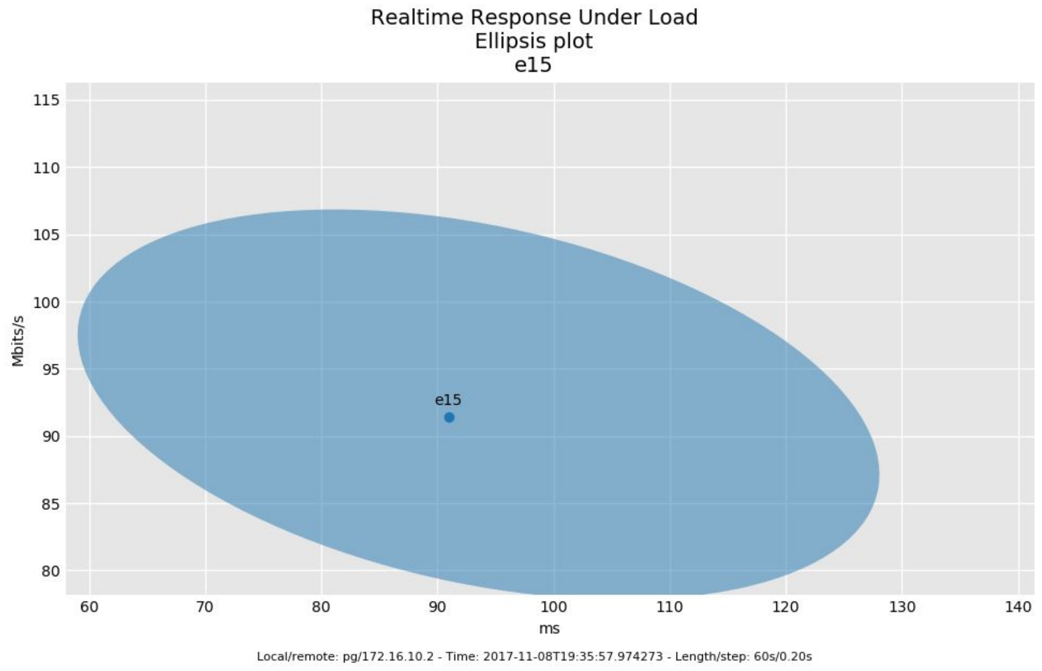
Where rrul is test, 60 means we will run the test case for 60 sec, 172.162.10.2 is the IP Address of the server machine and e1 is just a text in which e means tcp window scaling “enabled” and 1 means the size of initial congestion window.

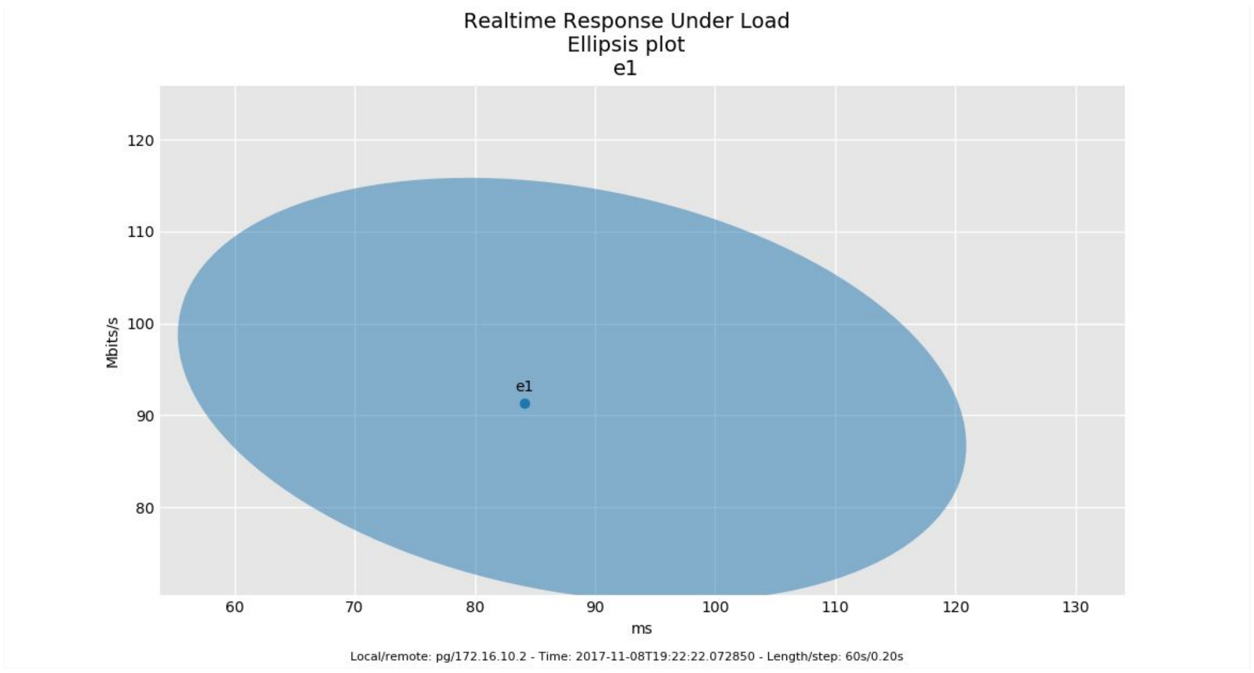
Analysis of graphs (ellipsis) obtained :

In this analysis section, there are mainly two cases :

1. Impact of Initial Congestion Window
2. Impact of TCP Window Scaling

Case 1 : When we just change the size of initial congestion window say from 1 to 15 keeping Window scaling unchanged,





We know that the ellipse shown here covers 60 to 70 percent of the points of bandwidth-delay obtained by running the test for certain period of time (1 minute as per standard). Also greater the area of the ellipse, more is the variance obtained.

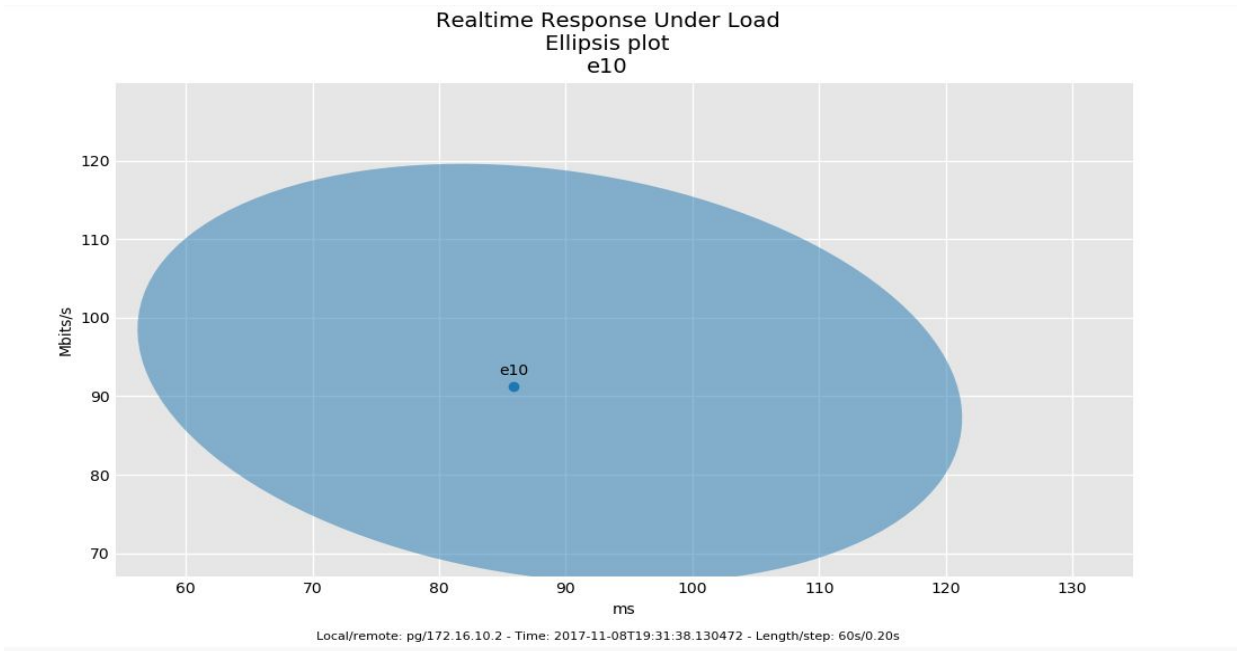
Now if we compare these two cases e1 and e15 (i.e. TCP Window Scaling enabled and initial congestion window values of 1 and 15)

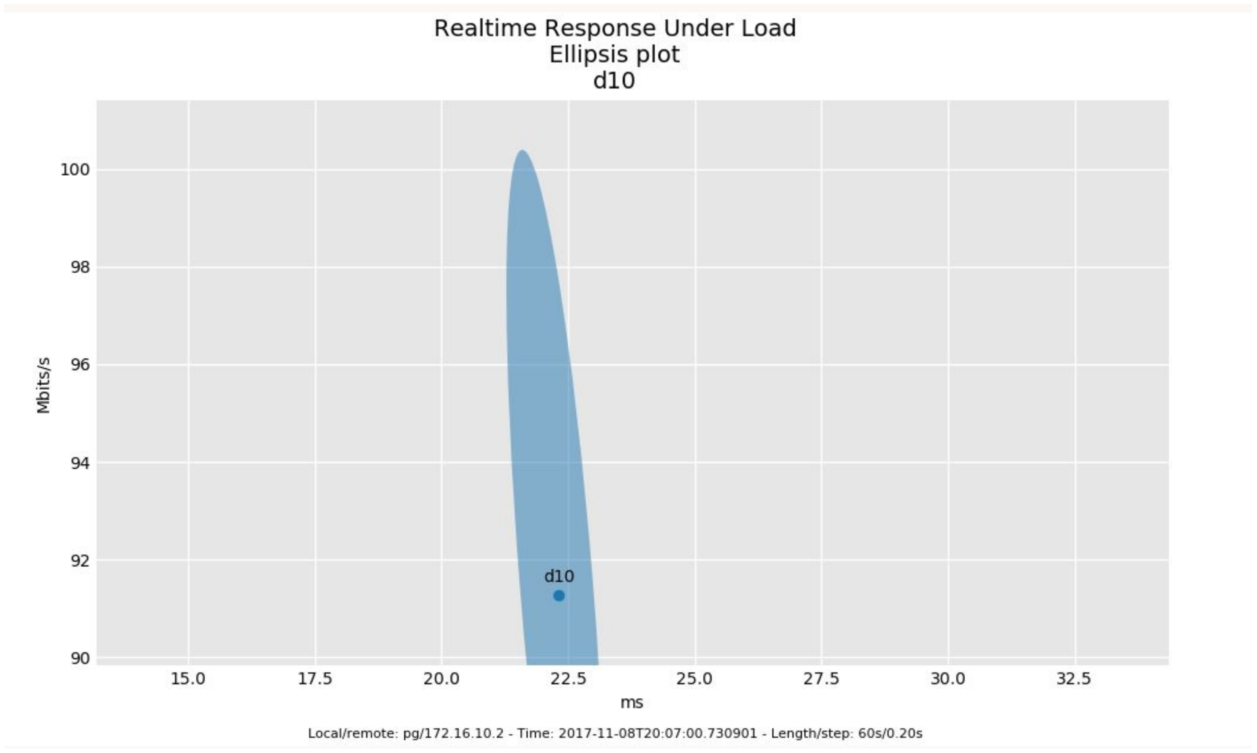
a) Variance is more in e1 than e15.
b) In e15, the graph is more towards top-left corner than in e1 case implying that the algorithm ran better in that case.
c) The highest throughput obtained in e1 is around 115 Mbits/s while it is around 107 Mbits/sec in e15. Such a behaviour is because of bottleneck situation in case where there is higher value of initial congestion window.
d) Also the least delay observed in e15 case is around 59 ms while it is around 55 ms in e1 case.

Similar comparison can be drawn with TCP Window Scaling disabled.

Case 2 : When we just enable/disable the TCP Window Scaling keeping the initial congestion window unchanged.

Consider the case where Initial Congestion Window is 10 i.e. e10 and d10 .





Clearly, there is huge difference between shapes of the two ellipses.

1. The ellipse in e10 has considerably large amount of area showing a great amount of variance in bandwidth-delay as compared to the sleek and slim ellipse obtained in case of d10.
2. The highest throughput obtained with enabled tcp window scaling is around 120 Mbits/s while its just around 103 Mbits/s with disabled tcp window scaling.
3. As per the graphs, There is not much deviations in delay in disabled window scaling. It is kind of concentrated around 22.5 ms. However, there is huge variation in delays when tcp window scaling was enabled.
4. Also considering the shape of the graph, the algorithm with enabled scaling ran better as evident from its inclination towards top-left corner, however, the ellipse obtained is very less slanted towards top-left corner when tcp window scaling was disabled.