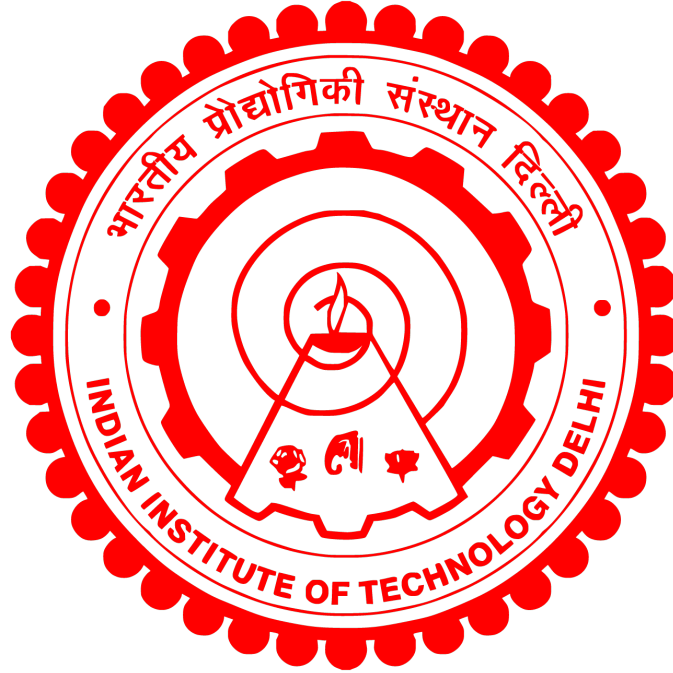


COL-724 : Assignment 1: Bufferbloat



Advanced Computer Networks

COL724

A1

Samyak Jain: 2020CS50667

Written under guidance of ***Prof. Huzur Saran***

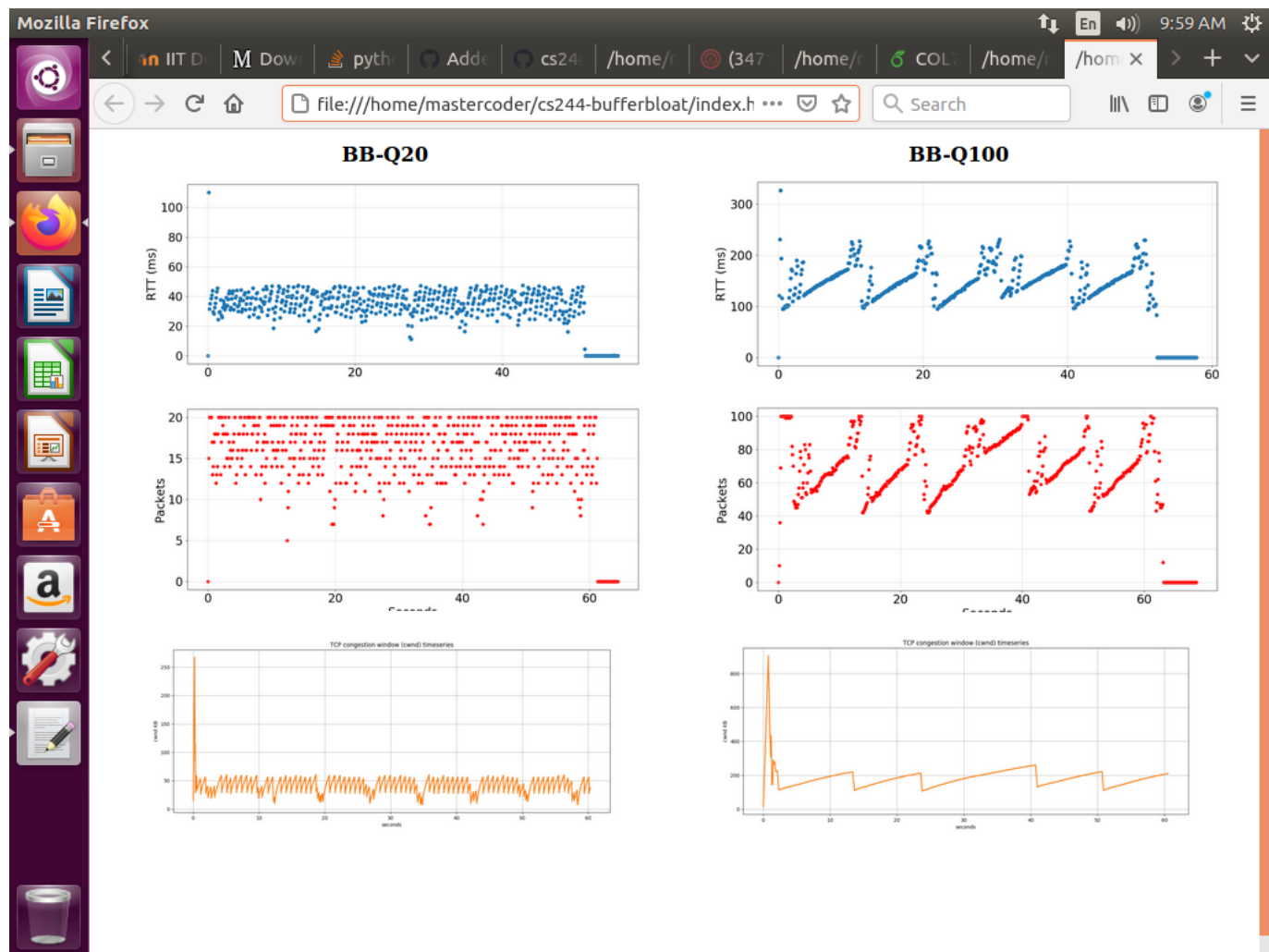
Keywords: iperf, ping, bufferbloat, router size, RTT

1 Report questions

1.1 Figures and Graphs

Following graphs are present in a very particular order:

1. For queue length as 20, RTT vs time (top left).
2. For queue length as 20, number of packets vs time (middle left).
3. For queue length as 20, congestion window for tcp vs time(bottom left).
4. For queue length as 100, RTT vs time (top right).
5. For queue length as 100, number of packets vs time (middle right).
6. For queue length as 100, congestion window for tcp vs time(bottom right).



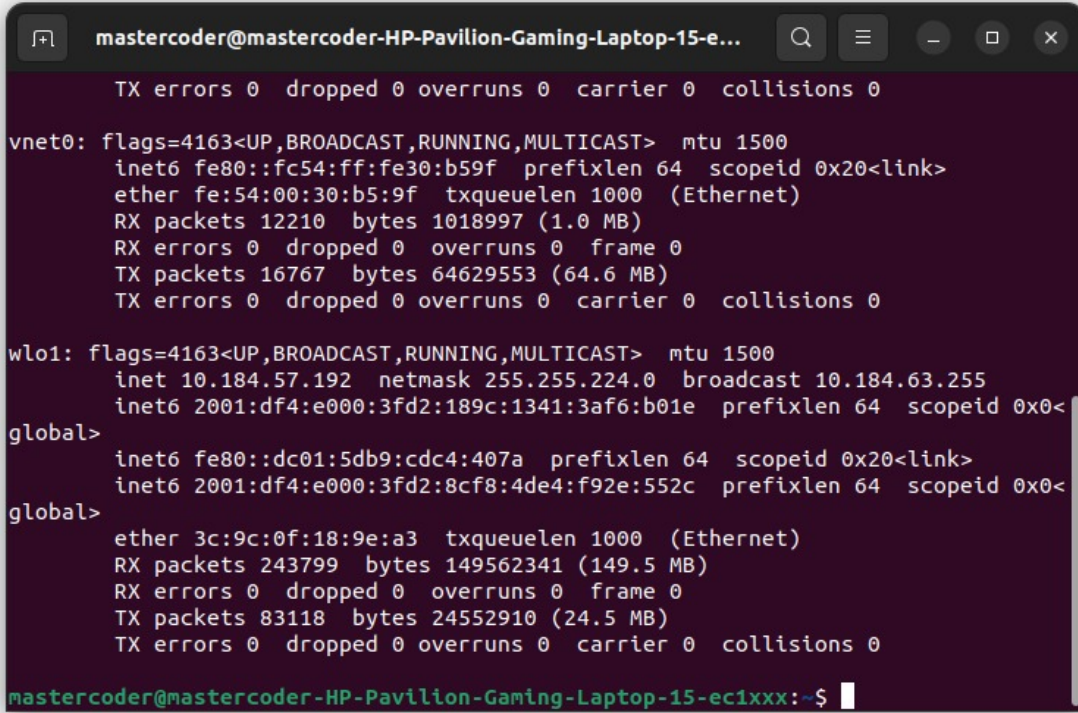
Following is the quantitative value of the mean and standard deviation obtained for queue len as 20 and 100 respectively.

```
mastercoder@mastercoder-Standard-PC-i440FX-PIIX-1996: ~/cs244-bufferbloat
mastercoder@mastercoder-Standard-PC-i440FX-PIIX-1996:~/cs244-bufferbloat$ sudo sh run.sh
net.ipv4.tcp_no_metrics_save = 1
net.ipv4.tcp_mem = 10240 87380 268435456
net.ipv4.tcp_congestion_control = reno
h1 h1-eth0:s0-eth1
h2 h2-eth0:s0-eth2
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
Starting iperf server...
59.0s left...
51.3s left...
43.6s left...
36.3s left...
28.6s left...
20.5s left...
13.3s left...
5.6s left...
('LEN of fetch_times: ', 24)
('Avg is: ', 0.5368333333333334)
('STD is: ', 0.03073605555555553)
Killed
saving to bb-q20/cwnd-iperf.png
saving to bb-q20/q.png
net.ipv4.tcp_congestion_control = reno
h1 h1-eth0:s0-eth1
h2 h2-eth0:s0-eth2
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
Starting iperf server...
59.0s left...
50.9s left...
40.7s left...
30.8s left...
20.4s left...
11.4s left...
1.8s left...
('LEN of fetch_times: ', 21)
('Avg is: ', 1.1663333333333334)
('STD is: ', 0.2462914603174604)
```

1.2 A1

The fetch times are proportional to the queue lengths(also called router buffer length) in router and hence as the queue length increases we see an increase in the fetch times. When queue size is 20, the average fetch time is 0.5368 while in the case of queue size as 100, this becomes 1.116 (ms). Larger buffers induce larger latencies.

1.3 A2



```
mastercoder@mastercoder-HP-Pavilion-Gaming-Laptop-15-e...
TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

vnet0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet6 fe80::fc54:ff:fe30:b59f  prefixlen 64  scopeid 0x20<link>
    ether fe:54:00:30:b5:9f  txqueuelen 1000  (Ethernet)
    RX packets 12210  bytes 1018997 (1.0 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 16767  bytes 64629553 (64.6 MB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

wlo1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.184.57.192  netmask 255.255.224.0  broadcast 10.184.63.255
    inet6 2001:df4:e000:3fd2:189c:1341:3af6:b01e  prefixlen 64  scopeid 0x0<
global>
    inet6 fe80::dc01:5db9:cdc4:407a  prefixlen 64  scopeid 0x20<link>
    inet6 2001:df4:e000:3fd2:8cf8:4de4:f92e:552c  prefixlen 64  scopeid 0x0<
global>
    ether 3c:9c:0f:18:9e:a3  txqueuelen 1000  (Ethernet)
    RX packets 243799  bytes 149562341 (149.5 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 83118  bytes 24552910 (24.5 MB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

mastercoder@mastercoder-HP-Pavilion-Gaming-Laptop-15-ec1xxx:~$
```

Max queue length is 1000 (indicated by txqueuelen) on my device as reported by *ifconfig*. Therefore, multiplying by packet size(mtu) and dividing by bandwidth gives waiting time. Therefore max waiting time = $\frac{1000 \cdot 1500 \cdot 8}{100 \cdot 10^6} = 0.12s$

1.4 A3

RTT is linearly proportional to the queue size. Due to larger waiting times per packet in large size buffers(just because more packets can now be pushed in), the RTT per packet is higher and described the by the formula, it is in fact linear.

$$RTT = queue_len * delay \quad (1)$$

1.5 A4

Following are some ways to reduce bufferbloat delays:

1. Reduce the size of buffers. Smaller buffer wont have long queues because of smaller capacities and hence smaller waiting times.
2. We can use a policy similar to RED (random early discard) which drops a few packets probabilistically (this probability might depend on how many packets are present within the queue at the moment) and thus not allowing long queues and thus queueing delays.

3. Also for different traffic, we can have different queues and thus one queue wont get overfilled and cause long queuing delays.
4. Other Active Queue Management methods like congestion signalling (one example discussed in class was ECN bit usage), dynamic queue management (adjusting size of buffer itself depending on network conditions) etc could also be employed to reduce bufferbloat

1.6 A5

The 2 graphs are almost similar, match almost completely in shape and values. Small differences are present possibly due to order in which we run the 3 flows, background processes running on system (causing minute delays via OS or NIC) (these are also called system load), somerandomness present in the network conditions, initial conditions of running processes and network.

