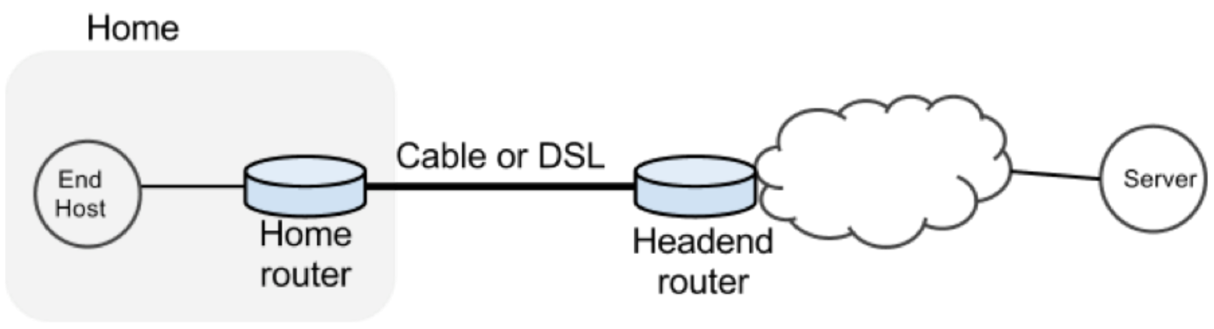
Lab 3: Bufferbloat

Name: ______PALAK BHONSLE_____________ID: _N10140736_______ Date: _______04/08/2016______

1. Objectives

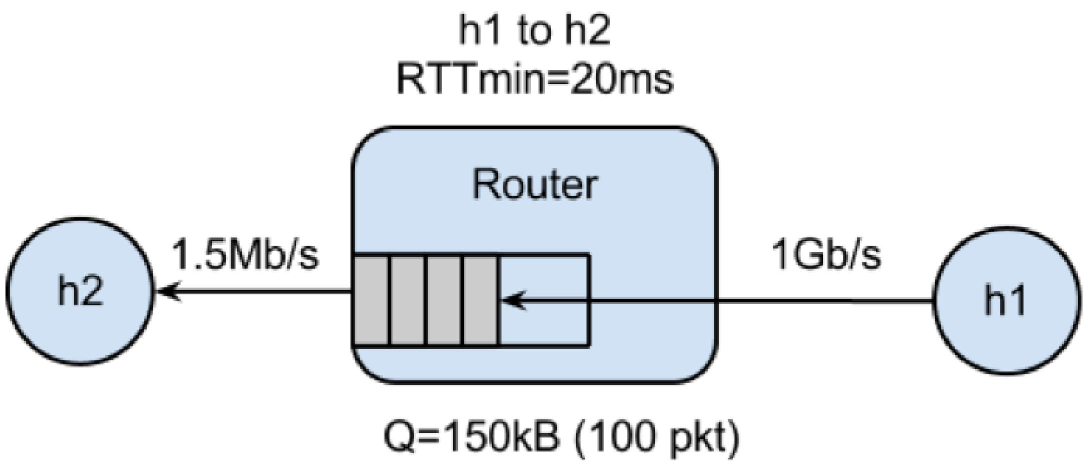
- Understand the bufferbloat problem
- Observe the bufferbloat problem on mininet

2.2 Home Network Emulation Setup



Aim is to study what happens when we download data from a remote server to the End Host in this home network.

2.3 BufferBloat Environment Setup



Buffer in the router is 150kB (150*1024=153600 bytes) equivalent to 100 packets which implies each packet consist of 1536 bytes.

To setup lab environment the following commands were run

```
git clone https://github.com/bovenyan/bufferbloat
```

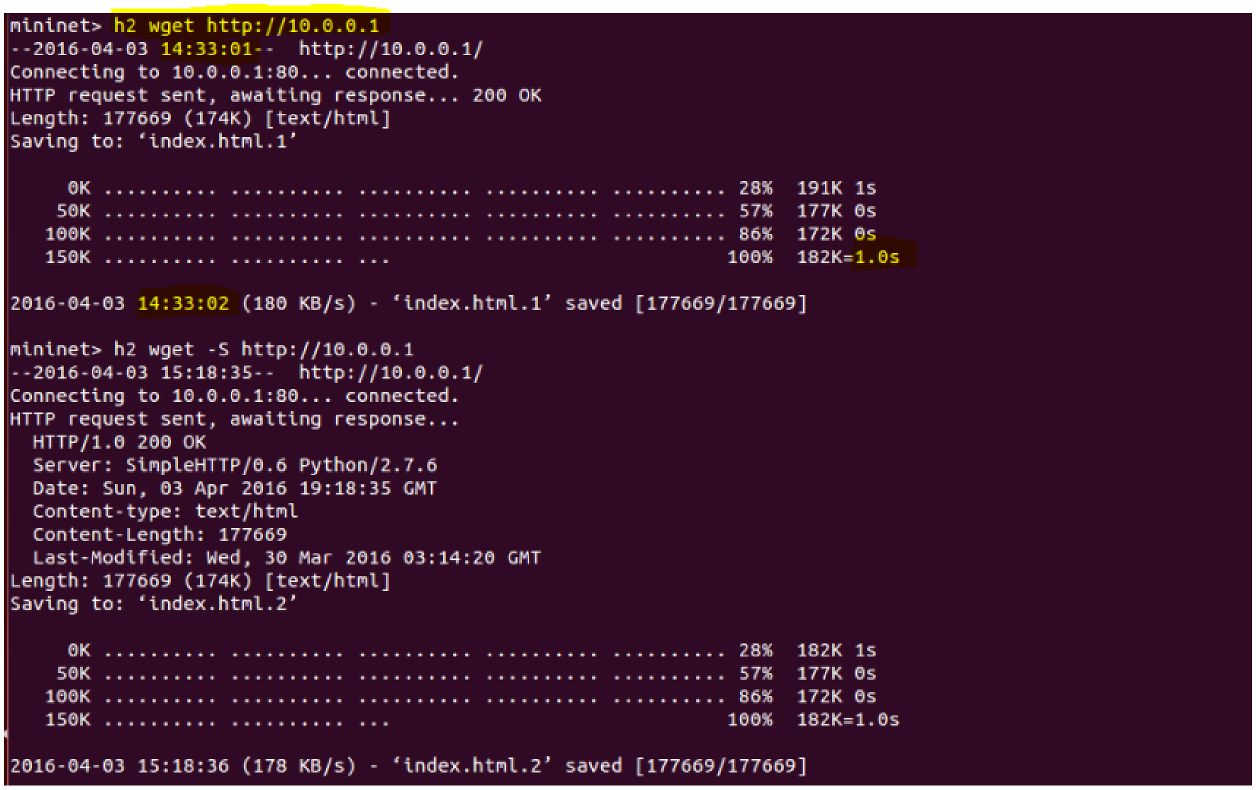
Run the emulator,

```
> cd bufferbloat/
> sudo ./run.sh
```

2.4 Exercise 1: Sketch the CWND of Web page downloads and "Streaming Videos"

1. Aim to measure how long it takes to download a web page from H1,

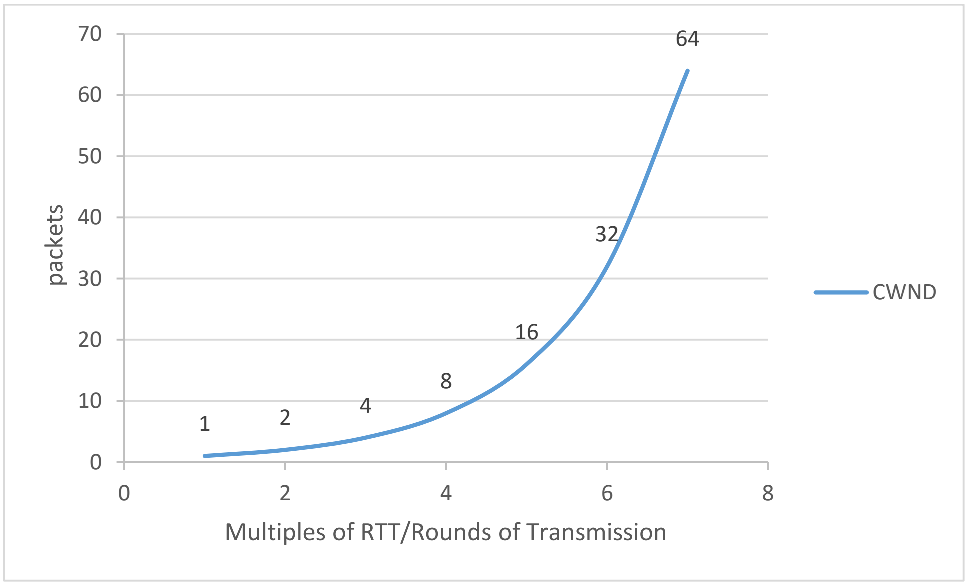
```
mininet> h2 wget http://10.0.0.1

It takes 1 second to download a webpage from H1.
```



```
mininet> h2 wget http://10.0.0.1
--2016-04-03 14:33:01--  http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 177669 (174K) [text/html]
Saving to: ‘index.html.1’

     0K .......... .......... .......... .......... .......... 28%  191K 1s
    50K .......... .......... .......... .......... .......... 57%  177K 0s
   100K .......... .......... .......... .......... .......... 86%  172K 0s
   150K .......... .......... ...                             100%  182K=1.0s

2016-04-03 14:33:02 (180 KB/s) - ‘index.html.1’ saved [177669/177669]

mininet> h2 wget -S http://10.0.0.1
--2016-04-03 15:18:35--  http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response...
  HTTP/1.0 200 OK
  Server: SimpleHTTP/0.6 Python/2.7.6
  Date: Sun, 03 Apr 2016 19:18:35 GMT
  Content-type: text/html
  Content-Length: 177669
  Last-Modified: Wed, 30 Mar 2016 03:14:20 GMT
Length: 177669 (174K) [text/html]
Saving to: ‘index.html.2’

     0K .......... .......... .......... .......... .......... 28%  182K 1s
    50K .......... .......... .......... .......... .......... 57%  177K 0s
   100K .......... .......... .......... .......... .......... 86%  172K 0s
   150K .......... .......... ...                             100%  182K=1.0s

2016-04-03 15:18:36 (178 KB/s) - ‘index.html.2’ saved [177669/177669]
```

2. **Sketch how you think cwnd evolves over time at H1. Mark multiples of RTT on the x-axis:**
   Wget is a short flow.
   TCP flow starts with slow start phase, according to slow start phase CWND is set to one initially, on first successful acknowledgement the window increments to 2 and after subsequent RTTs there is exponential growth of CWND.

So number of packets transmitted each RTT varies from 1, 2, 4, 8, 16, 32, 64, 128……
TCP flows enter Congestion Avoidance when CWND>=SSThresh.
Content length is 177669 bytes equal to 174KB and it takes 1 second to download. In Slow Start CWND increases exponentially each RTT (20 ms).The maximum buffer occupancy is 100 packets =150KB.The initial Threshold value for slow start phase is the initial advertised window [reffered RFC 5681-TCP Congestion Control] size(awnd) which in our case is 150 KB or 100 packets.
Summing the number of packets received each RTT 1+2+4+8+16+32+64=127 packets (127*1536=195072=190.5KB) 127 packets is equal to 190.5 KB of data which is more than data requested via wget(174KB).Since threshold value is not reached and there is no 3 duplicate ack or retransmission timeout so the TCP flow never leaves the slow start phase.



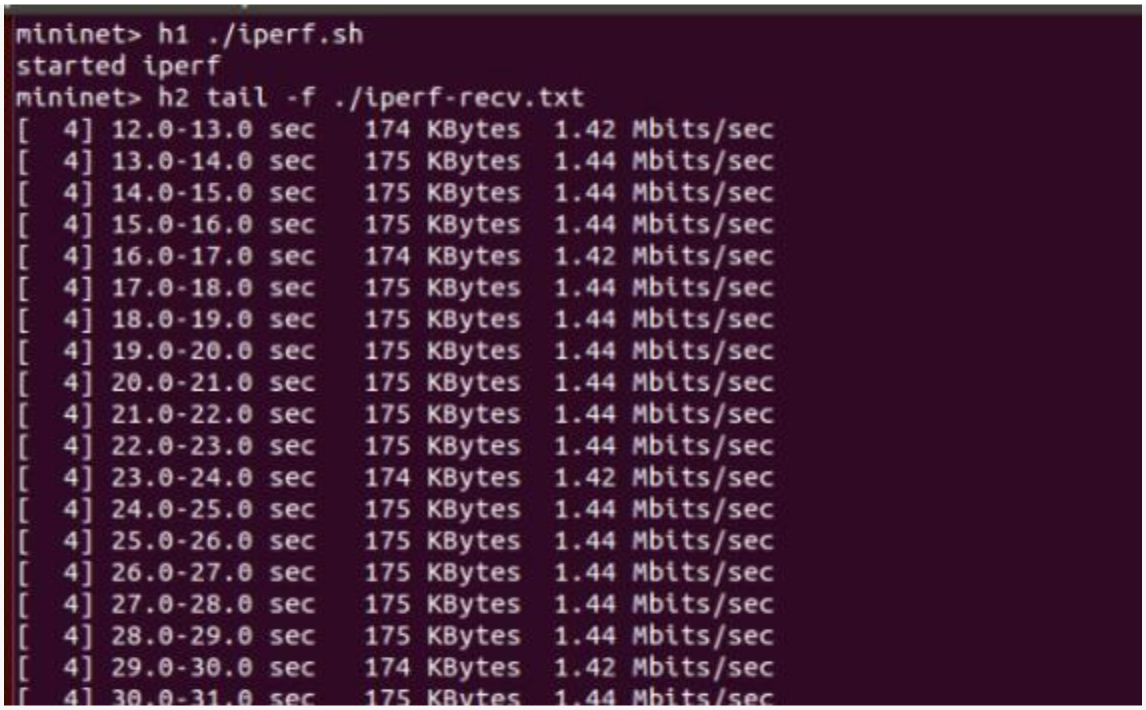
Aim: To see how the dynamics of a long flow (which enters the AIMD phase). We can generate long flows using the iperf command, which is wrapped in a script.

```
mininet> h1 ./iperf.sh .
```

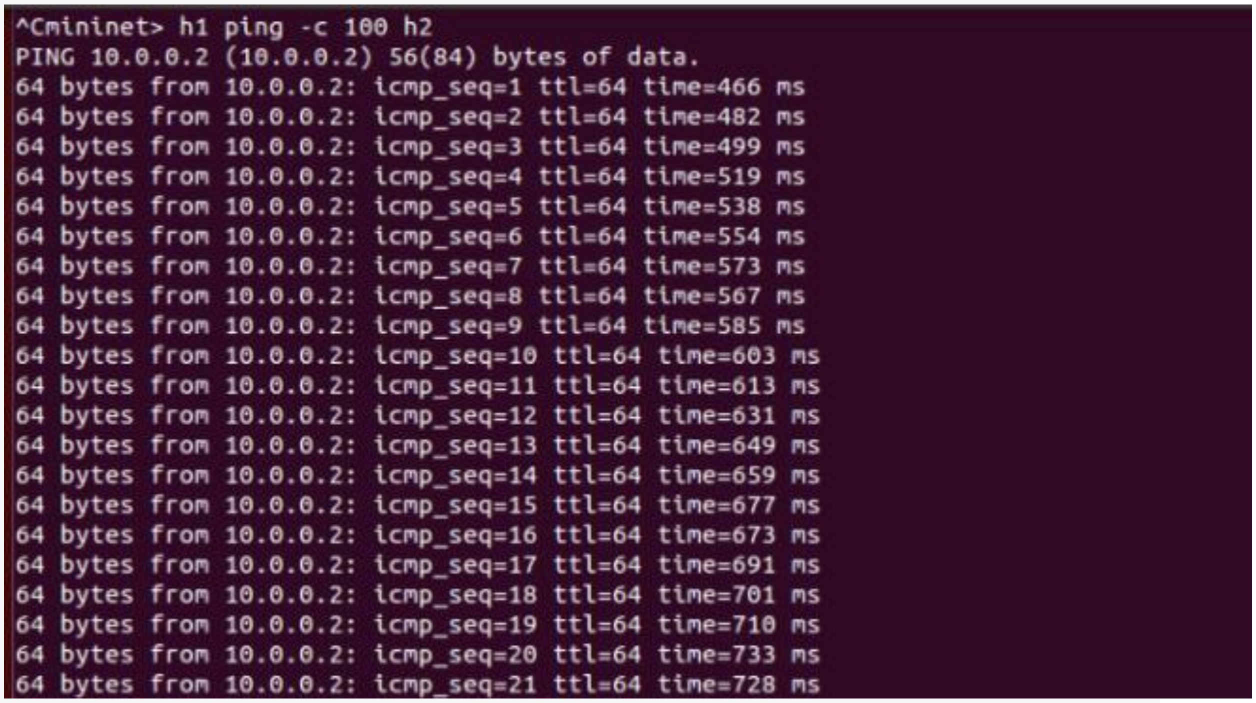
The throughput of TCP flow from H1 to H2 can be seen by running.

```
mininet> h2 tail -f ./iperf-recv.txt
```

To see how CWND evolves over time at H1 .We use ping command to measure the delay .after starting the iperf.
Use command: `mininet> h1 ping -c 100 h2.`
Below is the screenshot after executing these commands.



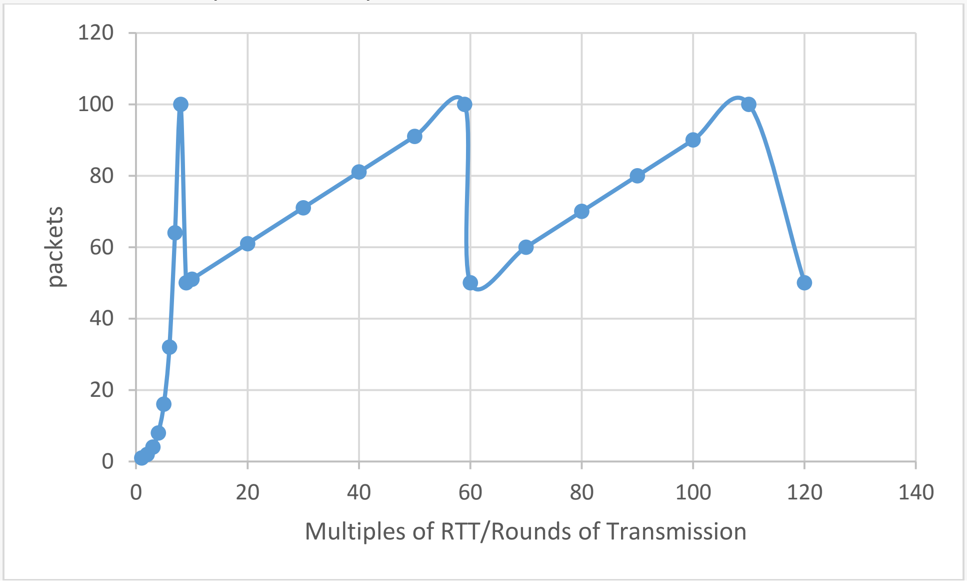
```
mininet> h1 ./iperf.sh
started iperf
mininet> h2 tail -f ./iperf-recv.txt
[  4] 12.0-13.0 sec   174 KBytes  1.42 Mbits/sec
[  4] 13.0-14.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 14.0-15.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 15.0-16.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 16.0-17.0 sec   174 KBytes  1.42 Mbits/sec
[  4] 17.0-18.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 18.0-19.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 19.0-20.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 20.0-21.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 21.0-22.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 22.0-23.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 23.0-24.0 sec   174 KBytes  1.42 Mbits/sec
[  4] 24.0-25.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 25.0-26.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 26.0-27.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 27.0-28.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 28.0-29.0 sec   175 KBytes  1.44 Mbits/sec
[  4] 29.0-30.0 sec   174 KBytes  1.42 Mbits/sec
[  4] 30.0-31.0 sec   175 KBytes  1.44 Mbits/sec
```



```
^Cmininet> h1 ping -c 100 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=466 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=482 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=499 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=519 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=538 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=554 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=573 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=567 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=585 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=603 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=613 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=631 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=649 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=659 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=677 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=673 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=691 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=701 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=710 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=733 ms
64 bytes from 10.0.0.2: icmp_seq=21 ttl=64 time=728 ms
```

3. Sketch how you think cwnd evolves over time at H1. You might find it useful to use ping to measure how the delay evolves over time, after the iperf has started:

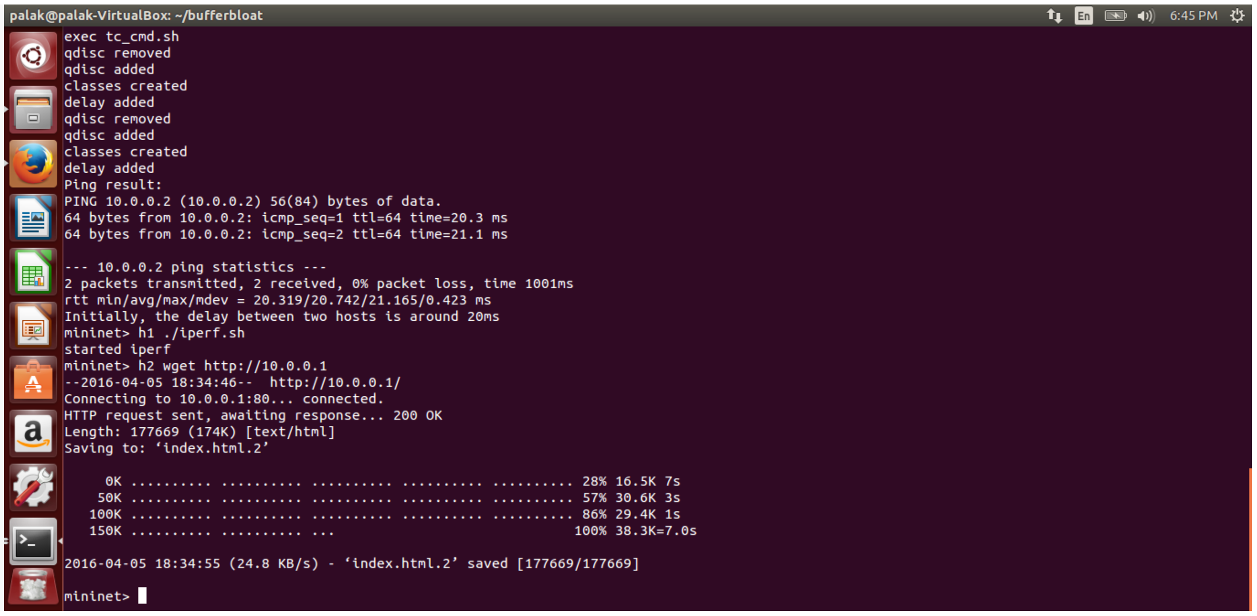
The ping rate is equivalent to the effective latency between H1 and H2.howevr due to congestion sometimes systems have low latency and more RTT.
Iperf is a long flow as compared to wget so the flow enters AIMD (Additive increase multiplicative decrease)
Once it reaches 100 packets congestion occurs as the receiver buffer gets fully occupied (CWND=sssthreshold=100 packets).Thus due to congestion it enters to MID (Multiplicative decrease) and thus reducing CWND to (Half of ssthreshold=50 packets).In additive increase increase CWND by 1 MSS every RTT.



To see how our long-lived iperf flow affects our web page download, download the webpage again - while iperf is running. Write down how long it takes.

```
mininet> h2 wget http://10.0.0.1
```

4. 7 seconds (how long the iperf flow affects our webpage download)



palak@palak-VirtualBox: ~/bufferbloat

```
exec tc_cmd.sh
qdisc removed
qdisc added
classes created
delay added
qdisc removed
qdisc added
classes created
delay added
Ping result:
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=21.1 ms

--- 10.0.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 20.319/20.742/21.165/0.423 ms
Initially, the delay between two hosts is around 20ms
mininet> h1 ./iperf.sh
started iperf
mininet> h2 wget http://10.0.0.1
--2016-04-05 18:34:46--  http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 177669 (174K) [text/html]
Saving to: ‘index.html.2’

     0K .......... .......... .......... .......... .......... 28% 16.5K 7s
    50K .......... .......... .......... .......... .......... 57% 30.6K 3s
   100K .......... .......... .......... .......... .......... 86% 29.4K 1s
   150K .......... .......... ...                             100% 38.3K=7.0s

2016-04-05 18:34:55 (24.8 KB/s) - ‘index.html.2’ saved [177669/177669]

mininet> 
```

5. **Why does the web page take so much longer to download? Please write your explanation below.**

The web page takes longer time to download because

1. Previously only wget command was being executed, which led to the entire bandwidth being allocated for wget so it took only 1 second to download.   But now, we are even doing an iperf to the host. So, the bandwidth is shared between iperf and wget.
2. The buffer which is limited to 100 packets is also shared between iperf and wget .

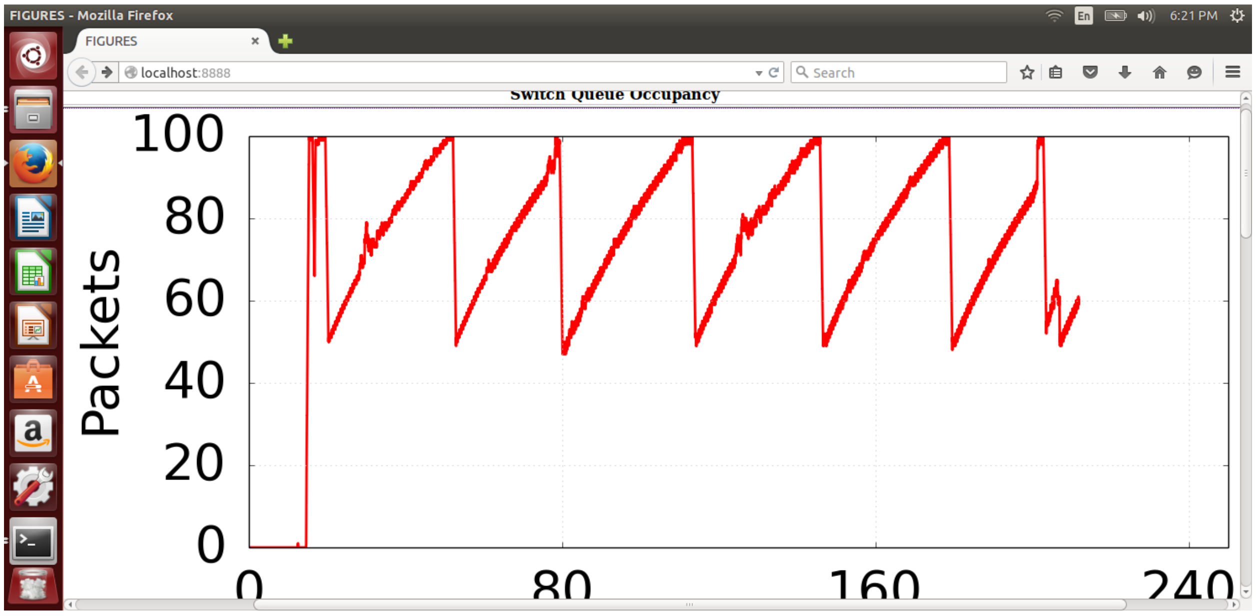
Due to all these reasons the webpage takes so much time to download.

2.5 Measuring the real cwnd and buffer occupancy values.

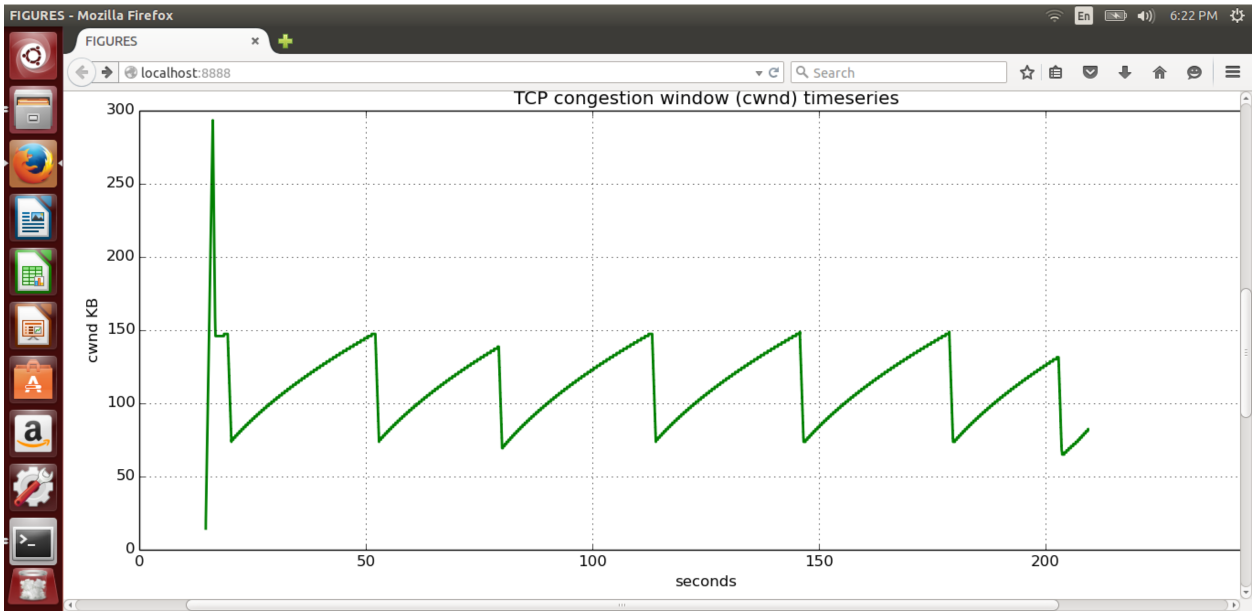
The buffer in the Headend router is so large that when it fills up with iperf packets, it delays the short wget.

6.Adjust command line parameters to generate the figure you want.

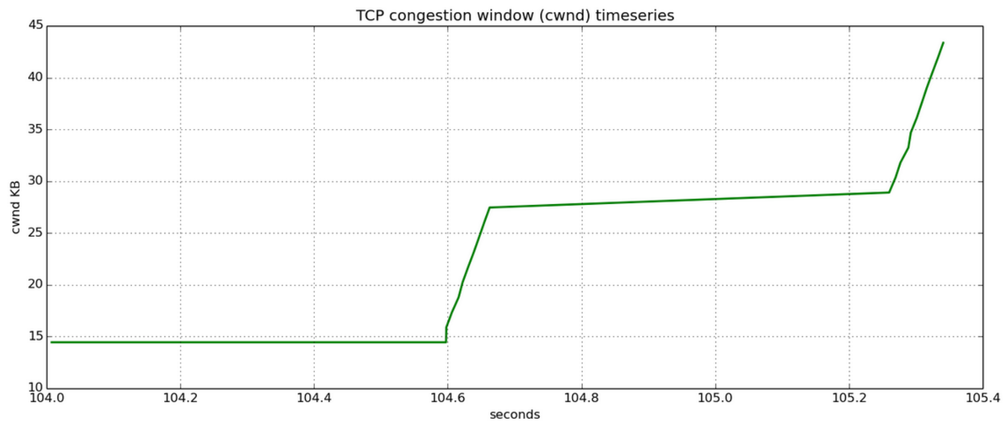
SWITCH QUEUE OCCUPANCY



TCP CWND FOR IPERF



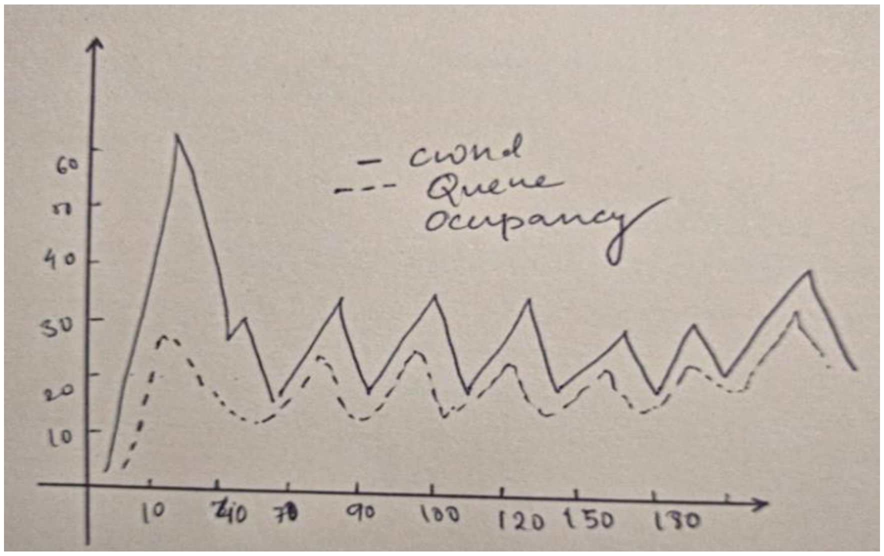
TCP CWND FOR WGET



2.6 Alleviate BufferBloat by making the buffer smaller

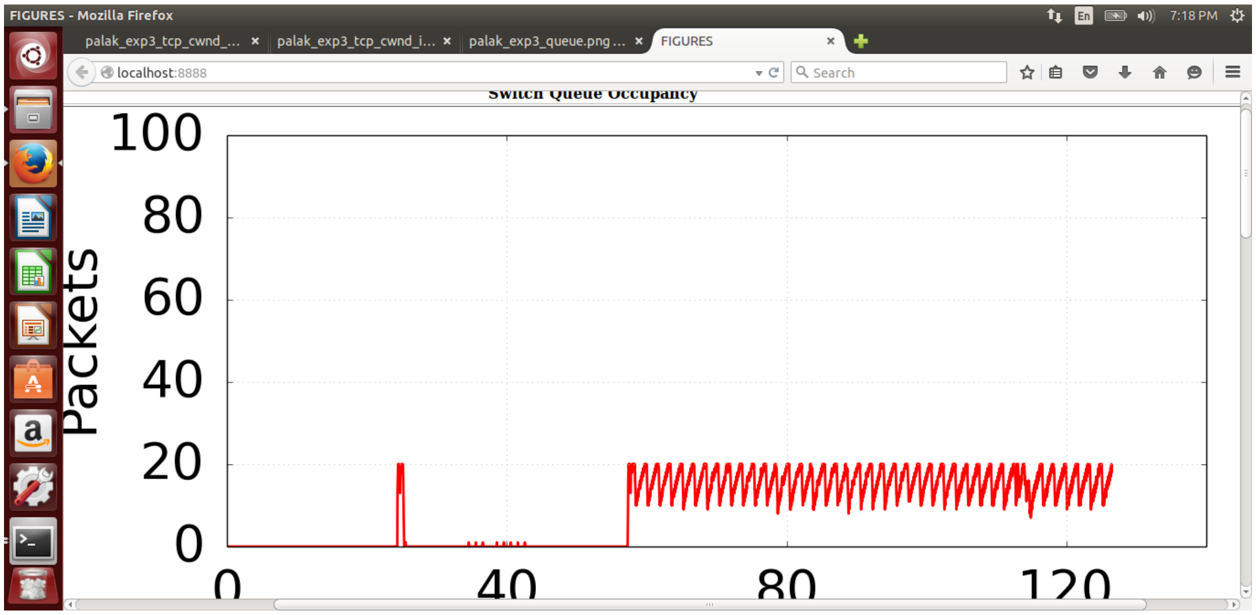
AIM :Make the router buffer smaller. Reduce it from 100 packets to 20 packets

7. What do you think the cwnd and queue occupancy will be like in this case?

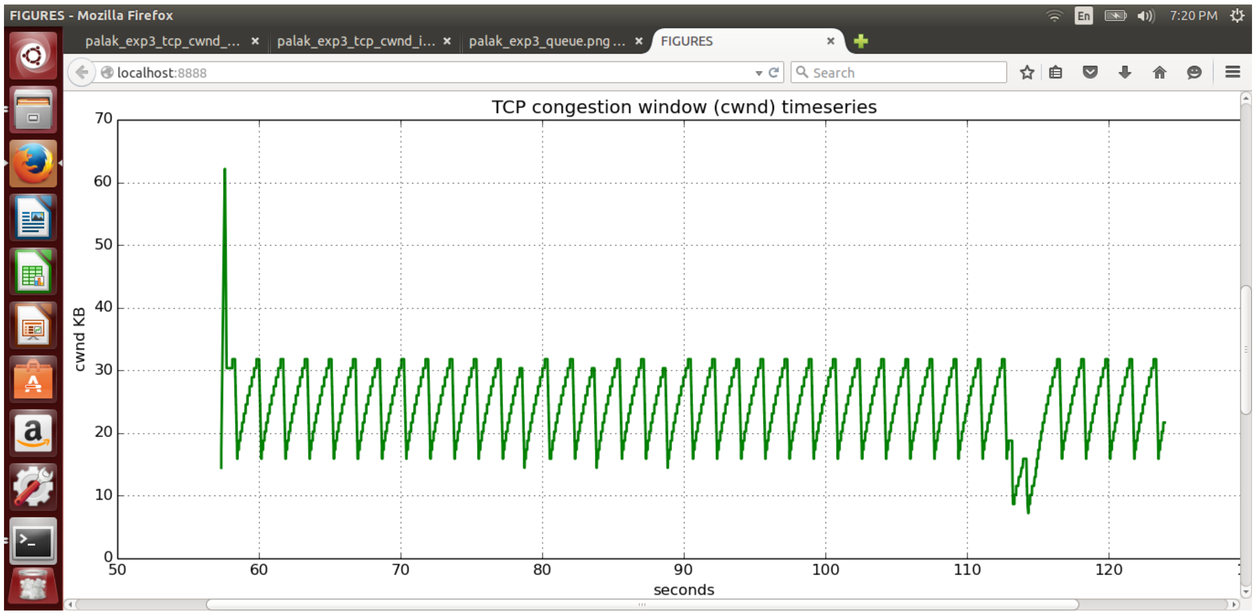


8. Plot the figure for cwnd and queue occupancy, this time using the script "./plot_figures_minq.sh"

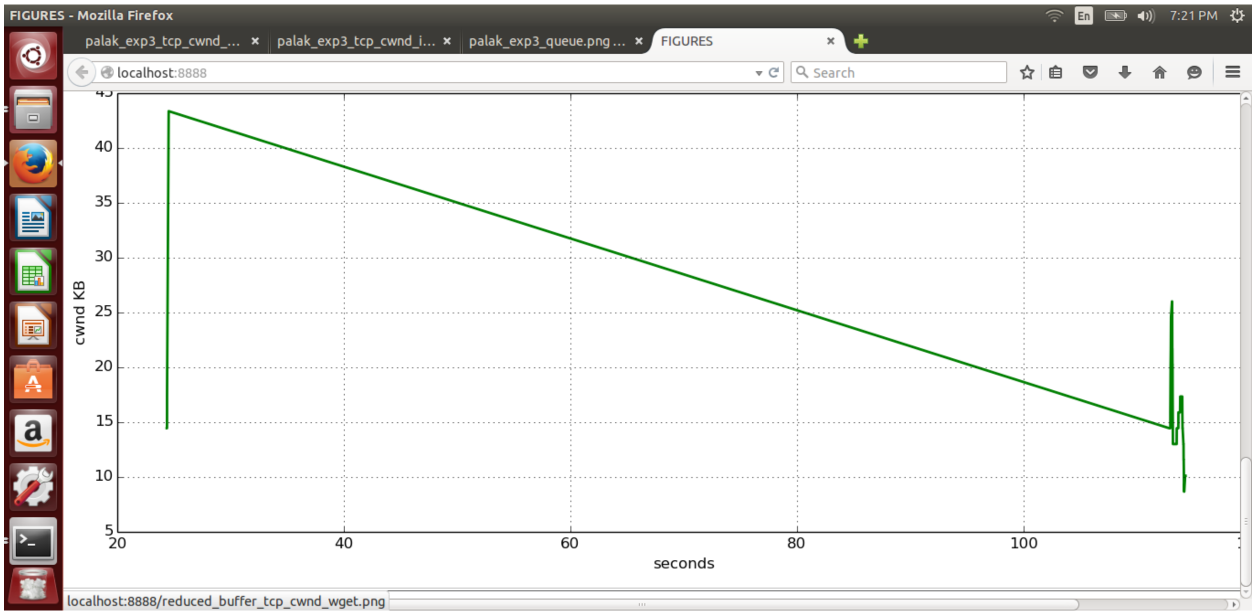
SWITCH QUEUE OCCUPANCY



TCP CWND FOR IPERF



TCP CWND FOR WGET



9. Why does reducing the queue size reduce the download time for wget? Please put your explanation below.

Reducing queue size reduces the download time for wget.

Suppose initially the queue size was 100 packets, total iperf packet 500(long flow) ,total wget packet 20 (short flow) each packet takes 10 ms to be delivered from h1 to h2.let's assume queue takes equal percentage of packets for each flow.

1st round 50 packets (iperf):20 packets (wget):30 packets (iperf)

Total 100 packets, therefore time taken by 20 packets of wget to be delivered is 10*100=1000ms or 1 sec.

Now assume reduced queue size 50 packets, total iperf packet 500(long flow) ,total wget packet 20 (short flow) each packet takes 10 ms to be delivered from h1 to h2.let's assume queue takes equal percentage of packets for each flow.

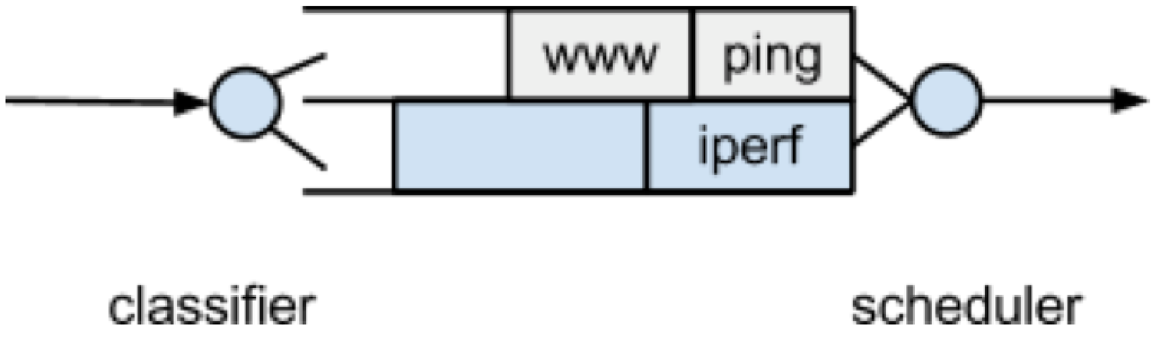
1st round 25 packets (iperf):20 packets (wget):5 packets(iperf)

Total 50 packets, therefore time taken by 20 packets of wget to be delivered is 10*50=500 ms or 0.5 seconds

Clearly the time taken by reduced queue size is less. If the buffer size is reduced, it would decrease of the original time to get to the top of stack and get delivered. Since buffer size is reduced to 20 packets the packets are stored for shorter time and are queued quickly.

2.7 Try different queues

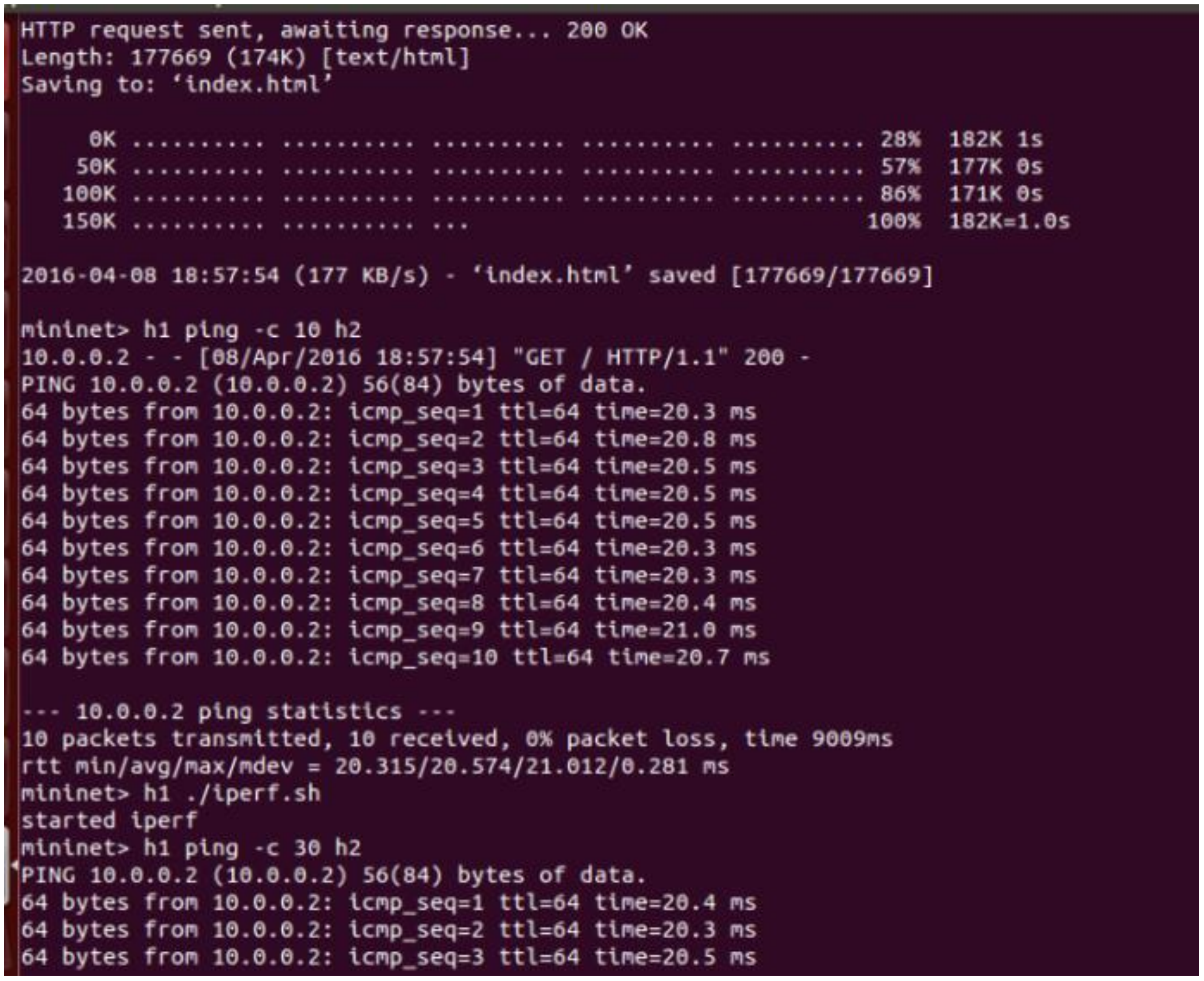
For this experiment, we put the iperf and wget/ping packets into separate queues in the Headend router. The scheduler implements fair queueing so that when both queues are busy, each flow will receive half of the bottleneck link rate.



10. You should see the ping delay (and the wget download time) doesn't change much before and after we start the iperf.

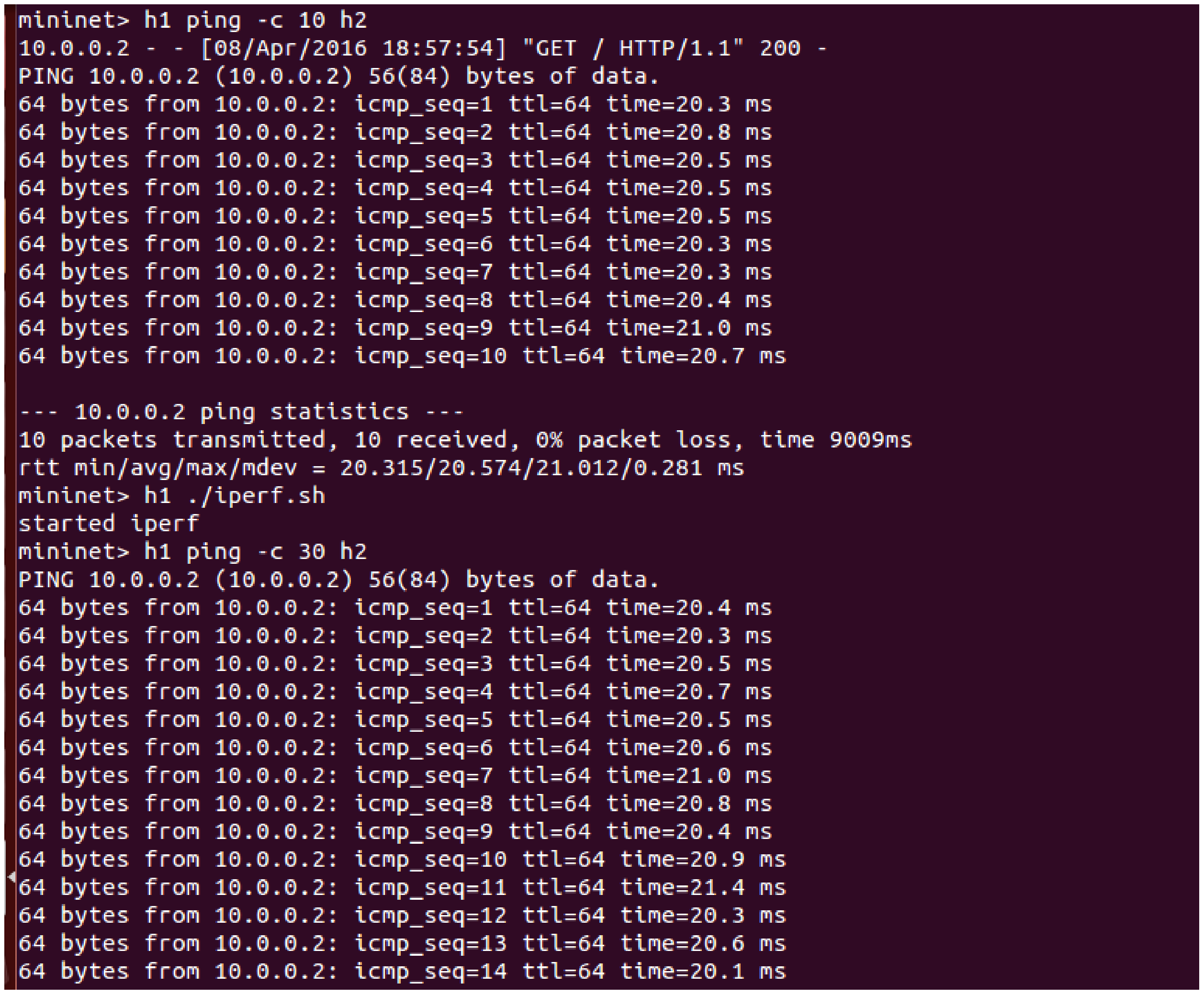
It can be observed from the outputs below that ping delay and wget download time doesn't change much before and after we start iperf. It remains 1 second for both the situations .This is because we put the iperf and wget/ping packets into separate queues in the Headend router. The scheduler implements fair queueing so that when both queues are busy, each flow will receive half of the bottleneck link rate.

Wget download and ping delay before iperf.

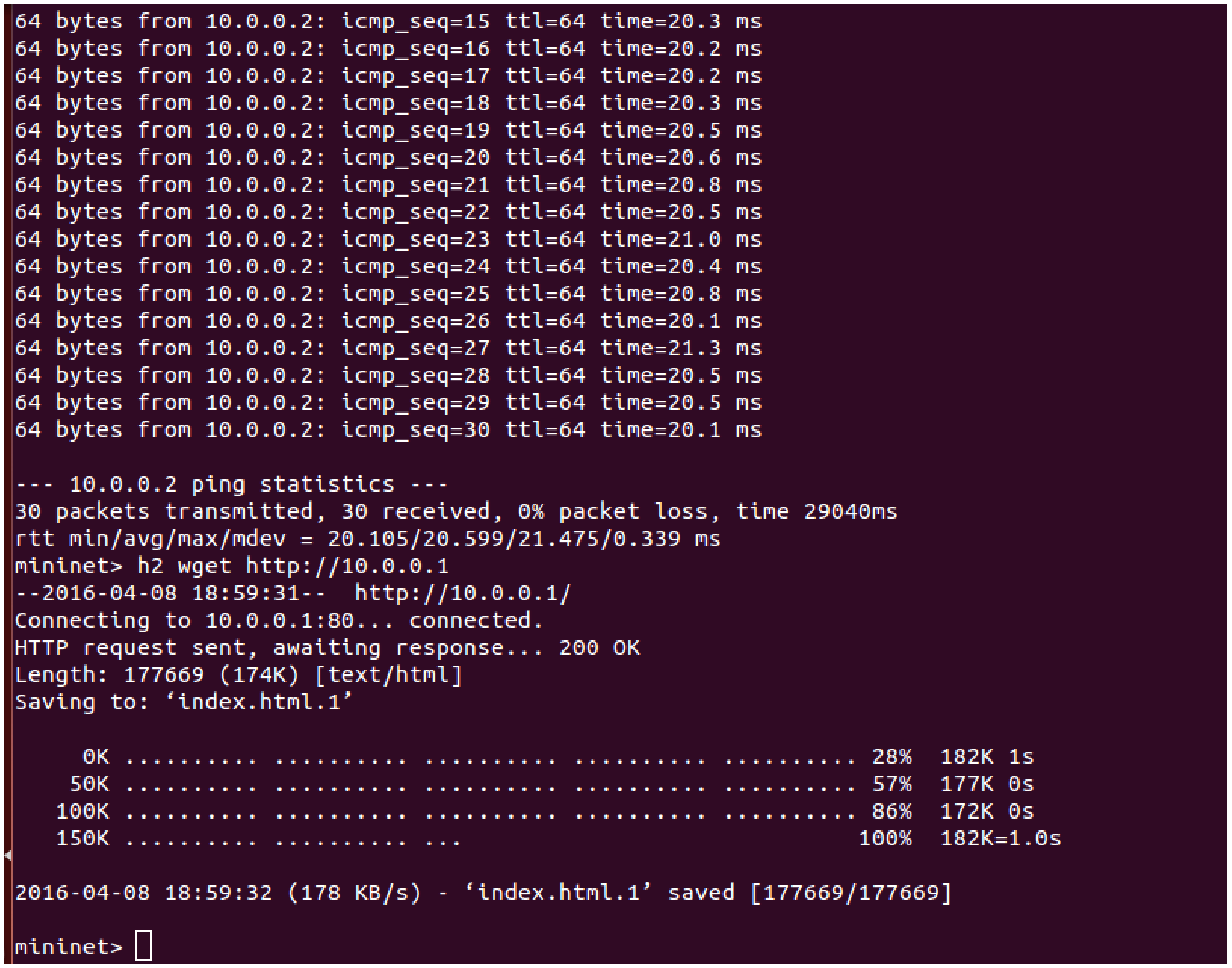


```
HTTP request sent, awaiting response... 200 OK
Length: 177669 (174K) [text/html]
Saving to: ‘index.html’

     0K .......... .......... .......... .......... .......... 28%  182K 1s
    50K .......... .......... .......... .......... .......... 57%  177K 0s
   100K .......... .......... .......... .......... .......... 86%  171K 0s
   150K .......... .......... ...                             100%  182K=1.0s

2016-04-08 18:57:54 (177 KB/s) - ‘index.html’ saved [177669/177669]

mininet> h1 ping -c 10 h2
10.0.0.2 - - [08/Apr/2016 18:57:54] "GET / HTTP/1.1" 200 -
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=20.8 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=20.5 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=20.5 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=20.5 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=20.4 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=21.0 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=20.7 ms

--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9009ms
rtt min/avg/max/mdev = 20.315/20.574/21.012/0.281 ms
mininet> h1 ./iperf.sh
started iperf
mininet> h1 ping -c 30 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=20.4 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=20.5 ms
```

Wget download and ping delay after iperf.

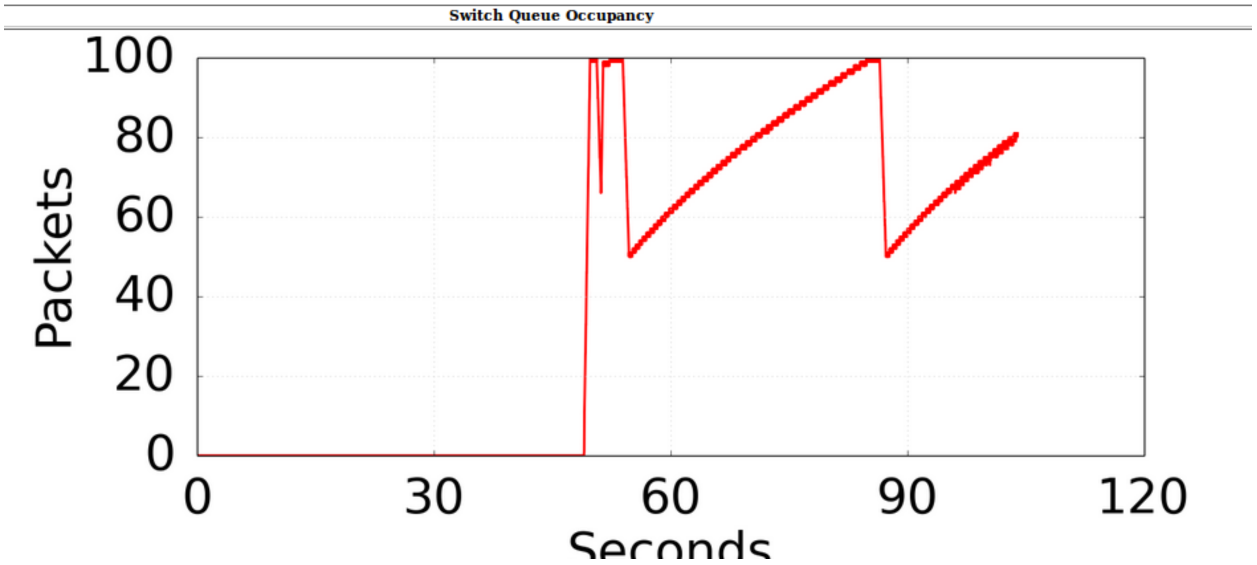


```
mininet> h1 ping -c 10 h2
10.0.0.2 - - [08/Apr/2016 18:57:54] "GET / HTTP/1.1" 200 -
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=20.8 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=20.5 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=20.5 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=20.5 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=20.4 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=21.0 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=20.7 ms

--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9009ms
rtt min/avg/max/mdev = 20.315/20.574/21.012/0.281 ms
mininet> h1 ./iperf.sh
started iperf
mininet> h1 ping -c 30 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=20.4 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=20.5 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=20.7 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=20.5 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=20.6 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=21.0 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=20.8 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=20.4 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=20.9 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=21.4 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=20.6 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=20.1 ms
```



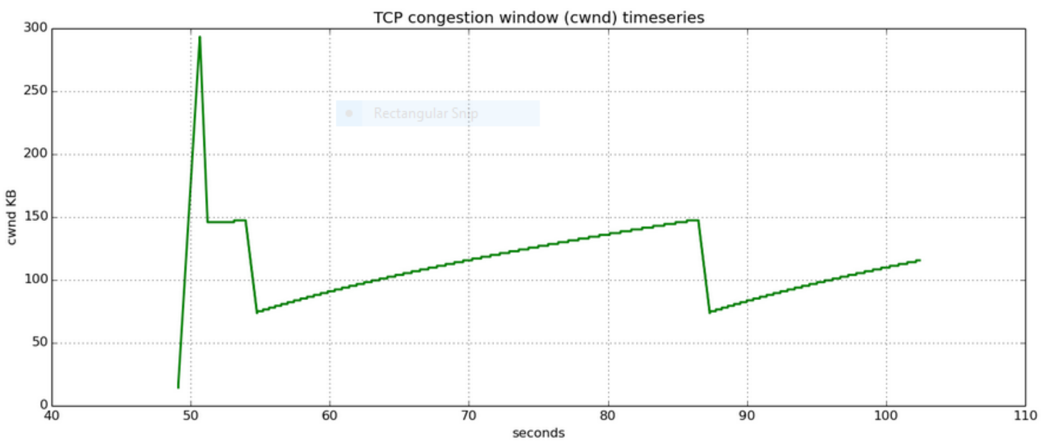
```
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=20.2 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=20.2 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=20.5 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=20.6 ms
64 bytes from 10.0.0.2: icmp_seq=21 ttl=64 time=20.8 ms
64 bytes from 10.0.0.2: icmp_seq=22 ttl=64 time=20.5 ms
64 bytes from 10.0.0.2: icmp_seq=23 ttl=64 time=21.0 ms
64 bytes from 10.0.0.2: icmp_seq=24 ttl=64 time=20.4 ms
64 bytes from 10.0.0.2: icmp_seq=25 ttl=64 time=20.8 ms
64 bytes from 10.0.0.2: icmp_seq=26 ttl=64 time=20.1 ms
64 bytes from 10.0.0.2: icmp_seq=27 ttl=64 time=21.3 ms
64 bytes from 10.0.0.2: icmp_seq=28 ttl=64 time=20.5 ms
64 bytes from 10.0.0.2: icmp_seq=29 ttl=64 time=20.5 ms
64 bytes from 10.0.0.2: icmp_seq=30 ttl=64 time=20.1 ms

--- 10.0.0.2 ping statistics ---
30 packets transmitted, 30 received, 0% packet loss, time 29040ms
rtt min/avg/max/mdev = 20.105/20.599/21.475/0.339 ms
mininet> h2 wget http://10.0.0.1
--2016-04-08 18:59:31--  http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 177669 (174K) [text/html]
Saving to: ‘index.html.1’

     0K .......... .......... .......... .......... .......... 28%  182K 1s
    50K .......... .......... .......... .......... .......... 57%  177K 0s
   100K .......... .......... .......... .......... .......... 86%  172K 0s
   150K .......... .......... ...                             100%  182K=1.0s

2016-04-08 18:59:32 (178 KB/s) - ‘index.html.1’ saved [177669/177669]

mininet> 
```

iperf Queue Occupancy:



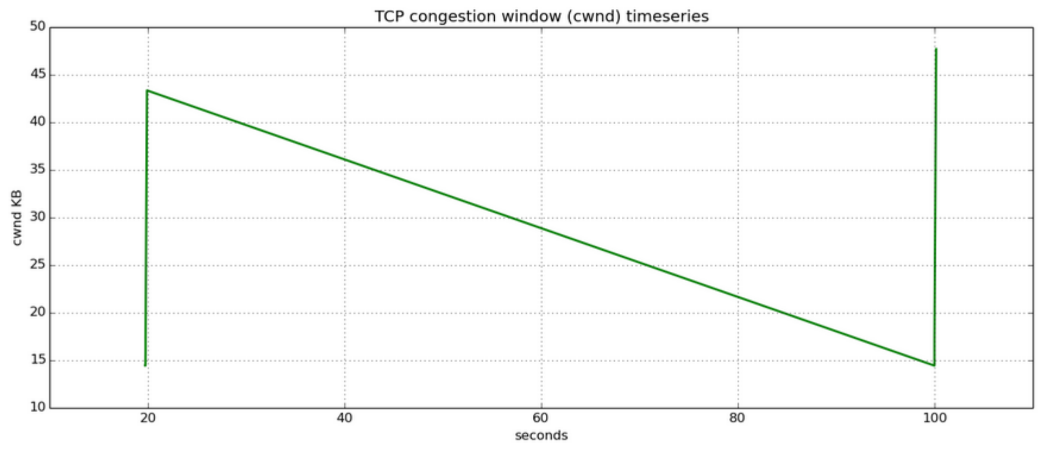
TCP cwnd:

TCP CWND for iperf



cwnd wget:

TCP CWND for wget


TCP congestion window (cwnd) timeseries
cwnd KB
seconds