Lab 3: Bufferbloat

Name: Pratik Anil Thakker ID: N13522702 Date: 11/12/2015

1. Objectives

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

1. Lab Tutorial

1.1 Reference to go through

1) BufferBloat: https://github.com/mininet/mininet/wiki/Bufferbloat
2) Iperf: https://iperf.fr/
3) Google: A universal reference

1.2 Home Network Emulation Setup

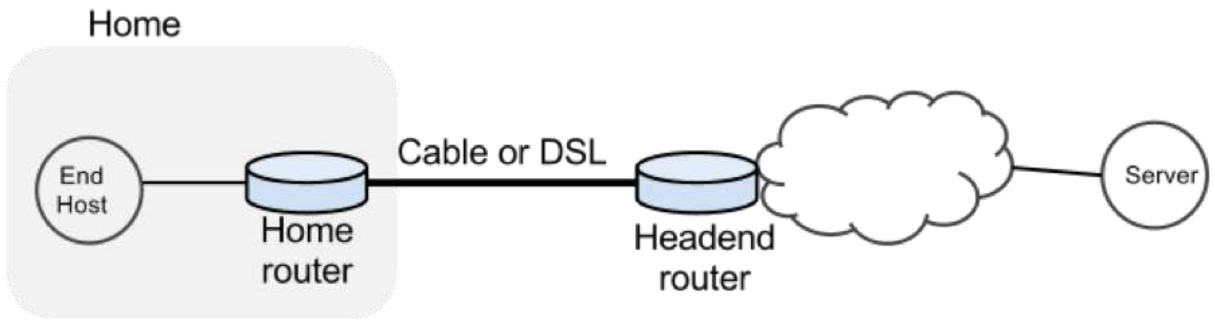


Figure 1 Typical home network environment

In this exercise we will study the dynamics of TCP in home networks. Take a look at Figure 1 which shows a “typical” home network with a Home Router connected to an end host. The Home Router is connected via Cable or DSL to a Headend router at the Internet access provider’s office. We are going to study what happens when we download data from a remote server to the End Host in this home network.

In a real network it's hard to measure cwnd (because it's private to the Server) and the buffer occupancy (because it's private to the router). To make our measurement job easier, we are going to emulate the network in Mininet.

1.3 BufferBloat Environment Setup

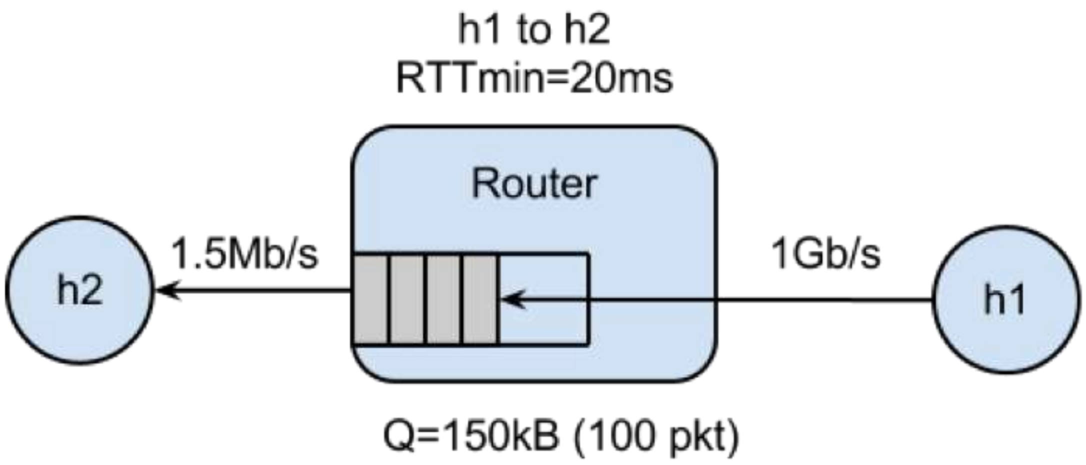


Figure 2. BufferBloat Environment Setup

In this lab, the environment is already provided as git repository. Get the repository by cloning it using the following command,

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

Run the emulator,

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

After Mininet is running, you can measure the delay from H1 to H2 with the command:

```
> mininet> h1 ping –c 10 h2
```

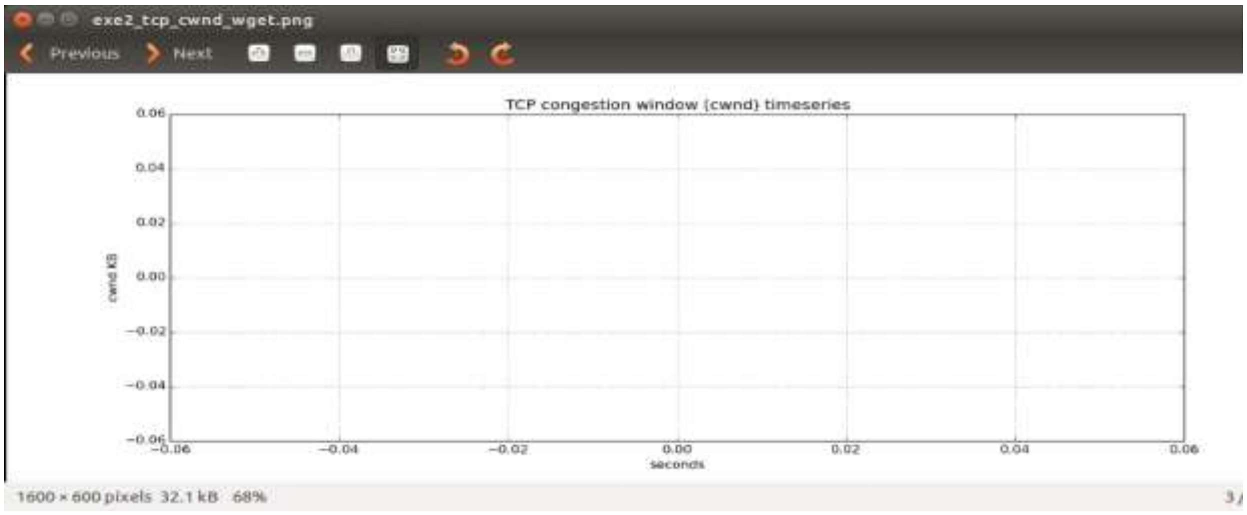
1.4 Exercise 1: Sketch the CWND of Web page downloads and "Streaming Videos"

Measure how long it takes to download a web page from H1,

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

1. Answer: 1 second.

2. Sketch how you think cwnd evolves over time at H1. Mark multiples of RTT on the x-axis:



To see how the dynamics of a long flow (which enters the AIMD phase) differs from a short flow (which never leaves slow-start), we are going to repeat Part 2 for a "streaming video flow". Instead of actually watching videos on your machine, we are going to set up a long-lived high speed TCP connection instead, to emulate a long-lived video flow. You can generate long flows using the iperf command, and we have wrapped it in a script which you can run as follows:

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

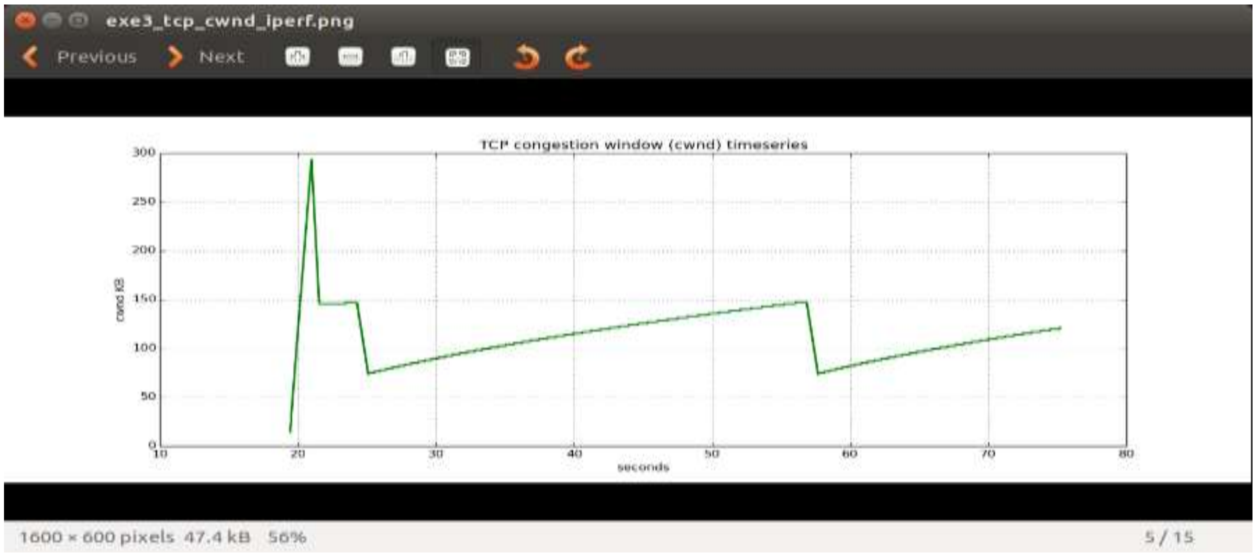
You can see the throughput of TCP flow from H1 to H2 by running:

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

You can quit viewing throughput by pressing CTRL-C.

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:

```
mininet> h1 ping -c 100 h2
```



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. Answer: 4.5 seconds

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

Answer: Since iperf generates tcp and udp traffic in the network the web page takes longer time to download.

1.5 Measuring the real cwnd and buffer occupancy values.

Stop and restart Mininet and the monitor script, then re-run the above experiment as follows.

```
mininet> exit
bash# sudo ./run.sh
```

Monitor TCP CWND and Buffer Occupancy in Mininet. In another bash terminal, go to bufferbloat directory and type the following giving a name for your experiment.

```
bash# ./monitor.sh <EXP_NAME>
```

Don’t worry if you see “ERROR: Module tcp_probe does not exist in /proc/modules”, it just means this module is not previously loaded.

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

(wait for 70 seconds …)

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

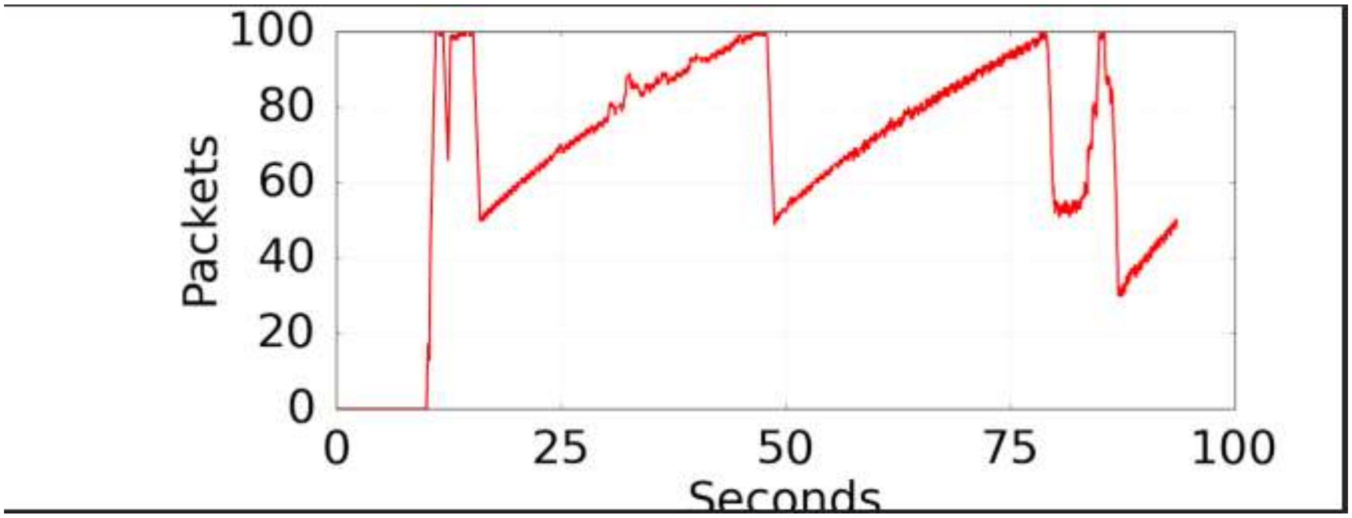
Wait for the wget to complete, then stop the python monitor script followed by the instructions on the screen. The cwnd values are saved in:

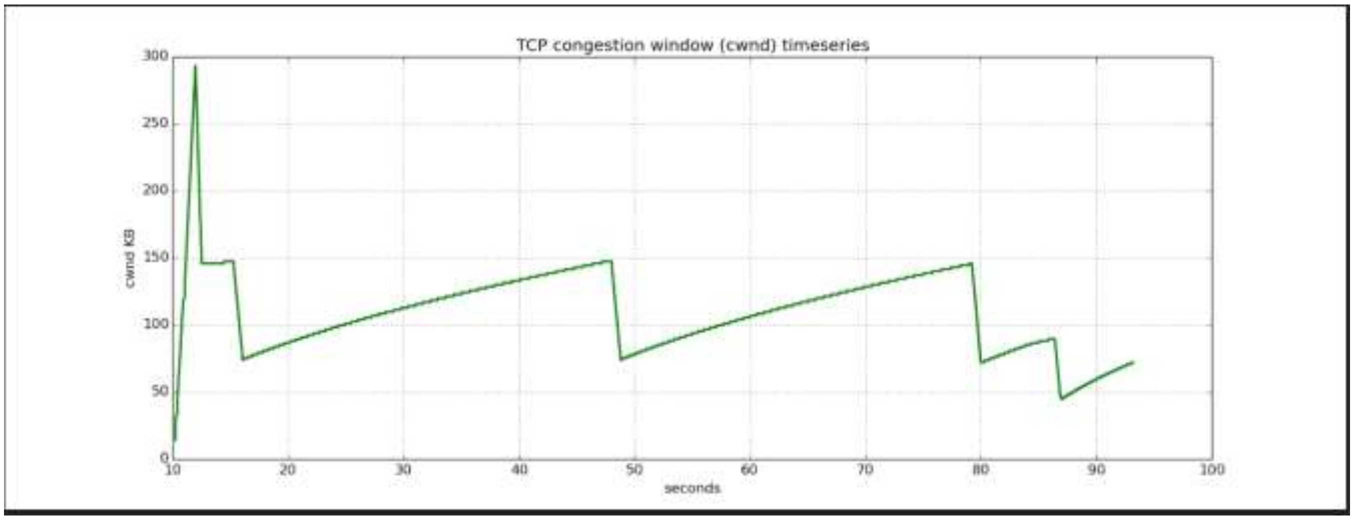
_tcpprobe.txt and the buffer occupancy in _sw0-qlen.txt

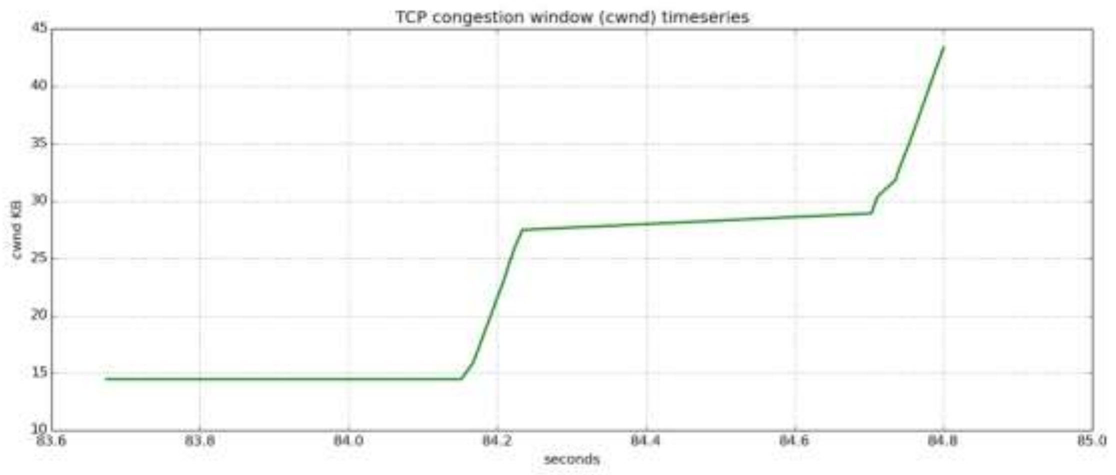
Plot the TCP cwnd and queue occupancy from the output file

```
bash# ./plot_figures.sh <EXP_NAME>
```

6. Adjust command line parameters to generate the figure you want. The script will also host a webserver on the machine and you can use the url the script provided to access to your figures if it is a remote machine w/ public IP. Sample Figures. If you are unable to see the cwnd, ensure you run wget after you started the monitor.sh script. By now you will have realized that the buffer in the Headend router is so large that when it fills up with iperf packets, it delays the short wget flow. Next we'll look at two ways to reduce the problem.







1.6 Alleviate BufferBloat by making the buffer smaller

Make the router buffer smaller. Reduce it from 100 packets to 20 packets. Stop any running Mininet and start Mininet again, but this time we will make the buffers 20 packets long instead:

```
prompt> sudo ./run-minq.sh
```

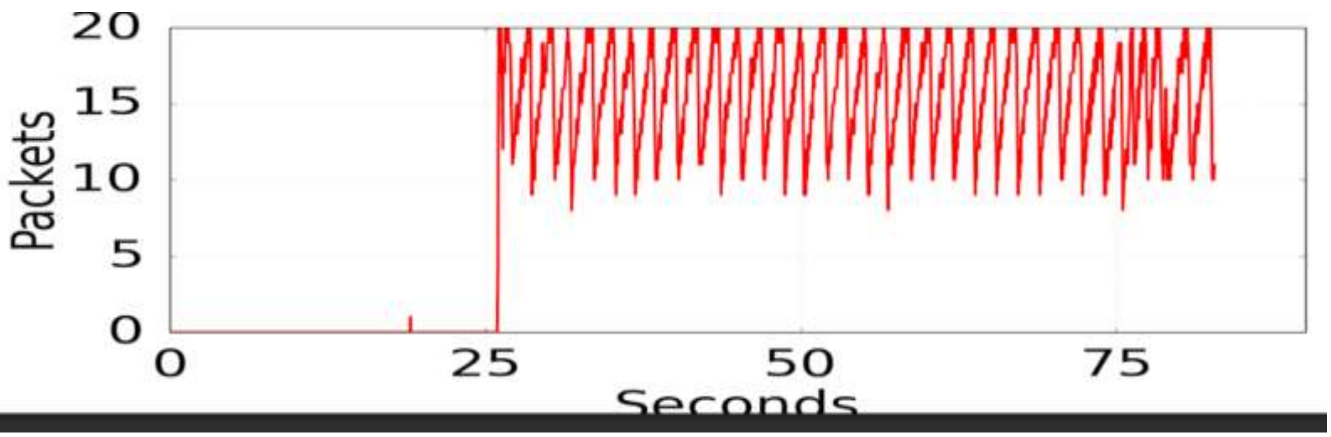
Let’s also run the monitor script on the side:

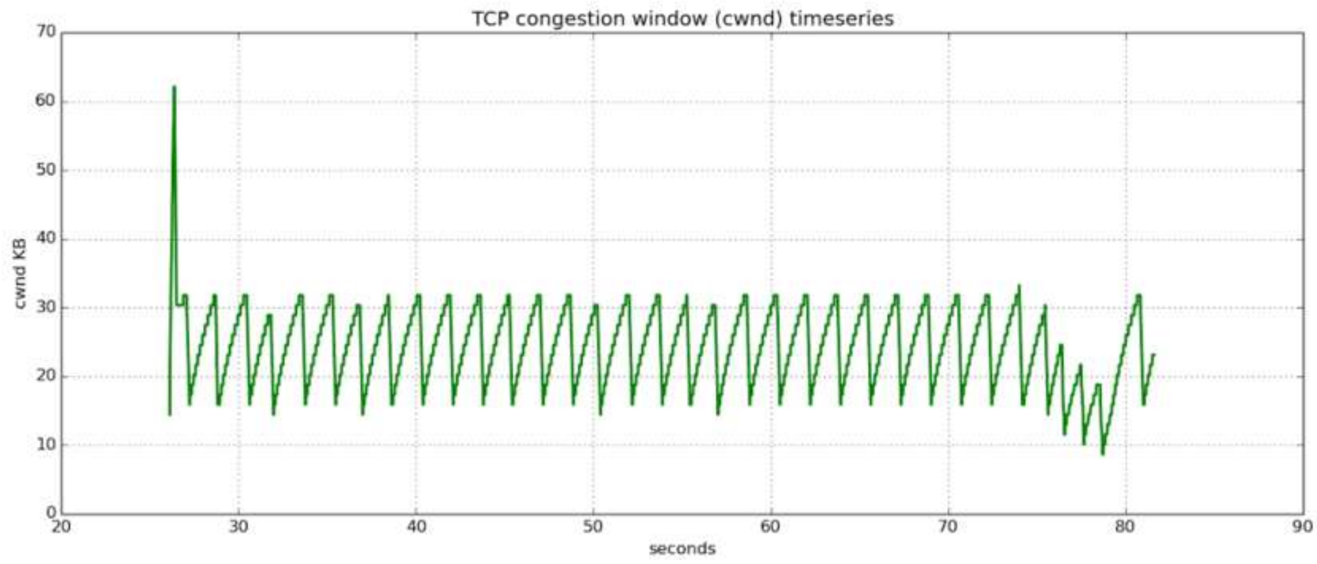
```
prompt> sudo ./monitor.sh <EXP_NAME>
```

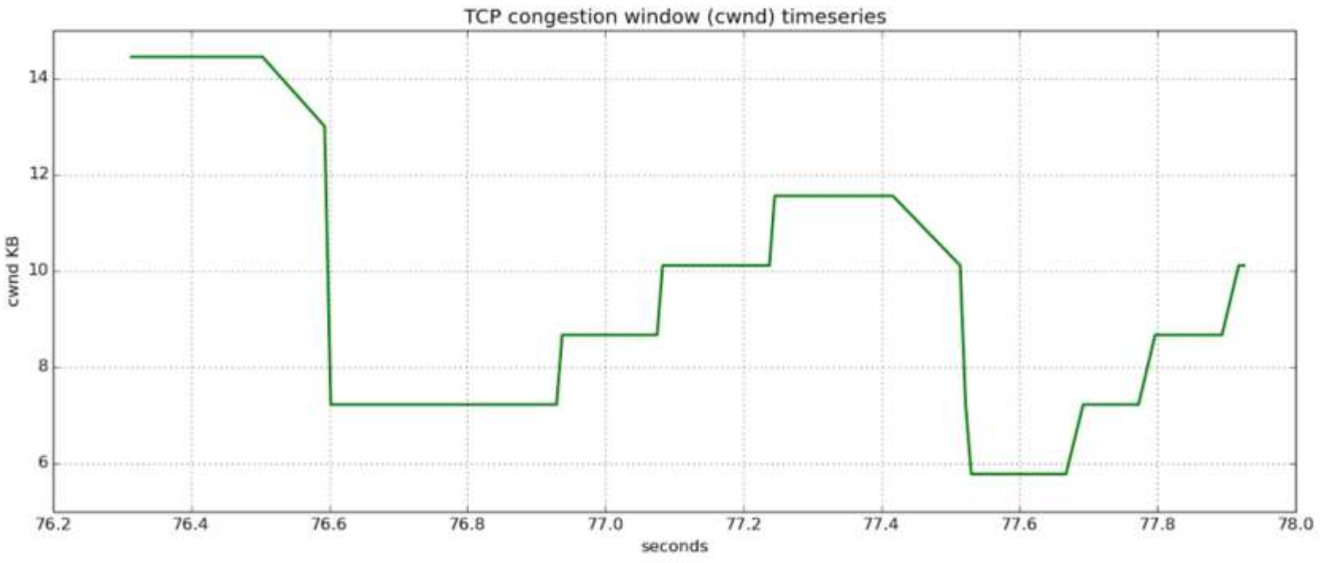
Repeat the steps in Parts 2 and 3:

```
mininet> h2 wget http://10.0.0.1
mininet> h1 ping -c 10 h2
mininet> h1 ./iperf.sh
mininet> h1 ping -c 30 h2
mininet> h2 wget http://10.0.0.1
```

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"

```
prompt> ./plot_figures_minq.sh
```

Then again, use the url to see your figures. Sample figures.

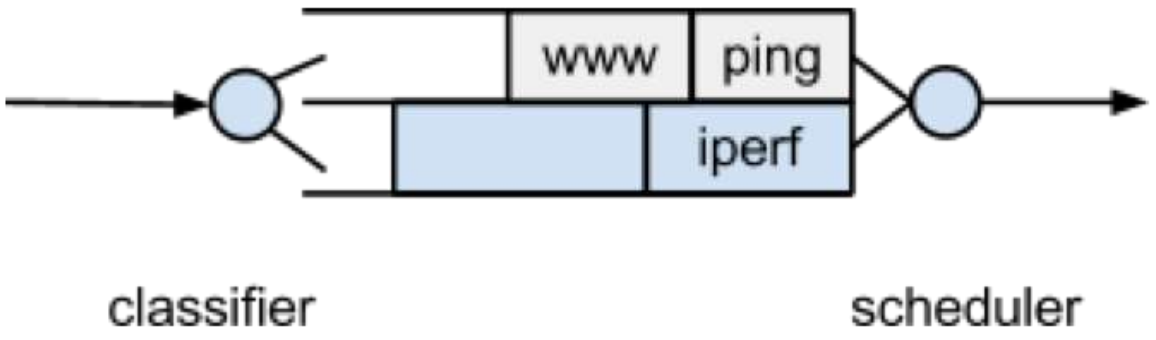
9. Why does reducing the queue size reduce the download time for wget? Please put your explanation below.

Answer: Reducing the queue size results in less time the packet has to wait in the queue and hence this reduces the download time for wget.

1.7 Try different queues

The problem seems to be that packets from the short flow are stuck behind a lot of packets from the long flow. What if we maintain a separate queue for each flow and then put iperf and wget traffic into 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.



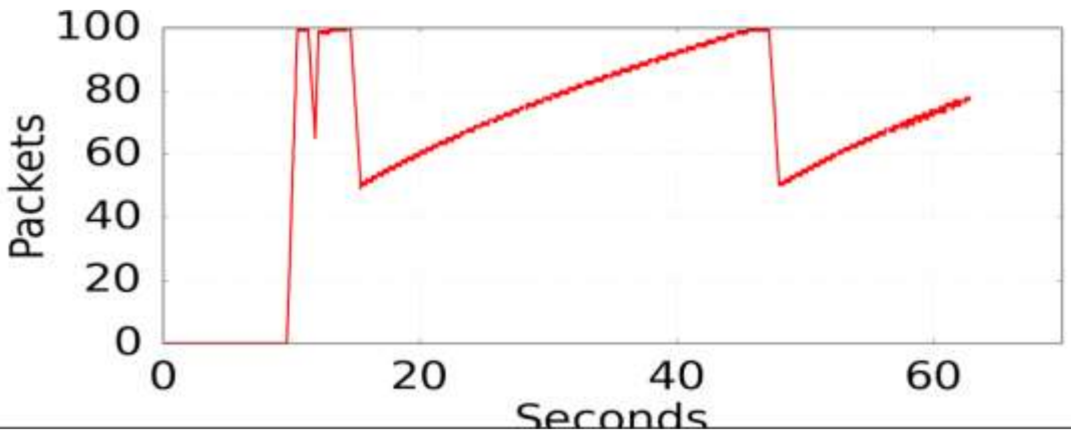
Start Mininet again, but this time we will create two queues, one for each type of traffic.

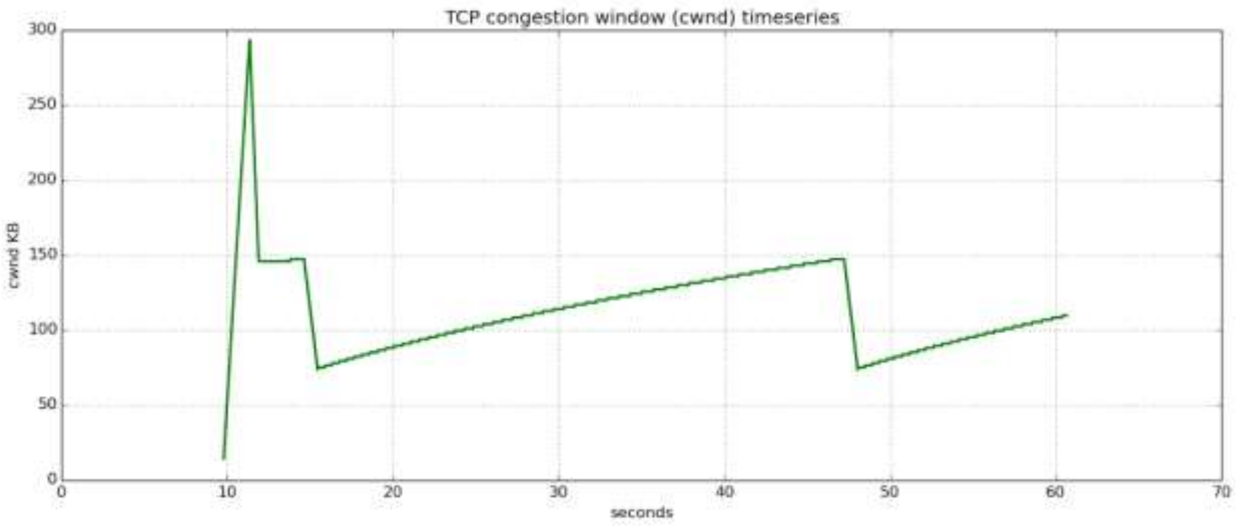
```
prompt> sudo ./run-diff.sh
```

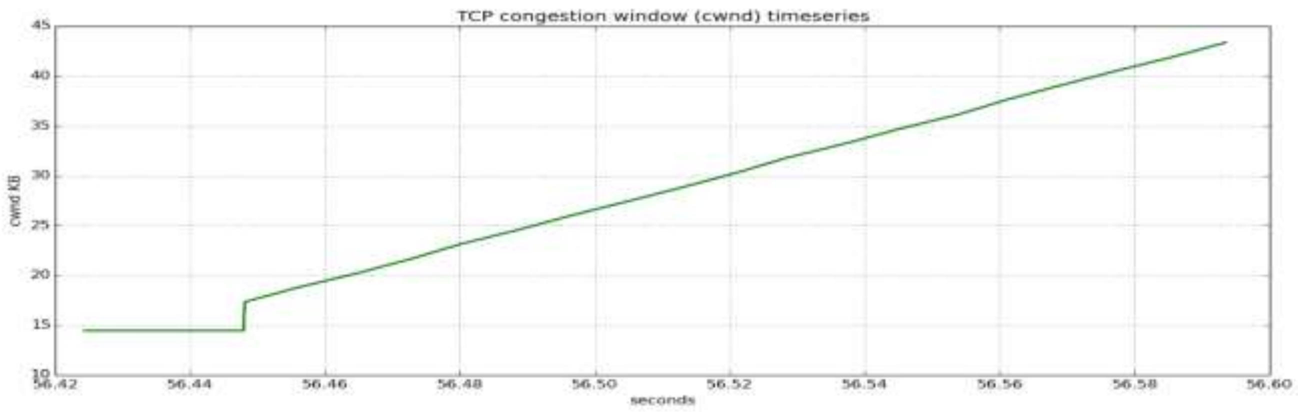
Repeat the steps in Parts 2 and 3

```
mininet> h2 wget http://10.0.0.1
mininet> h1 ping -c 10 h2
mininet> h1 ./iperf.sh
mininet> h1 ping -c 30 h2
mininet> h2 wget http://10.0.0.1
```

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


Packets
Seconds


TCP congestion window (cwnd) timeseries
cwnd KB
seconds


TCP congestion window (cwnd) timeseries
cwnd KB
seconds