

Lab 4: TCP Congestion Control

50.012 Networks

Hand-out: 8 Oct

Hand-in: 30 Oct (Saturday 11:59pm)

1 Introduction

In this lab, you will experiment with TCP congestion control and study the effect of router buffer sizes. The learning objectives include:

- Understand TCP's congestion control mechanism.
- Learn about the mininet tool and get familiar with network emulation.
- Understand why larger buffers can be disadvantageous.

Acknowledgement: Part of this lab is based on the Stanford CS244 Class lab1.

2 Setup

2.1 mininet

- This exercise assumes that you have a running mininet installation on Ubuntu 14.04.
- Please refer to the lab 4 preparation guide (from eDimension) on how to setup the VM environment and install mininet.
- Alternatively, you can download a pre-built VM image with mininet already installed:
`https://sutdapac-my.sharepoint.com/:u:/g/personal/bowen_liu_mymail_sutd_edu_sg/Ec2T8hhxBb1FpTcBkVFA_lwB9I6eZ7m_IC8FnxZ024ccng?e=g1jWb2`
For the pre-built VM, the Ubuntu password is aaa.
- Note that we need this older version of Ubuntu 14.04 because our script relies on the tcp_probe kernel module, which is no longer available in later versions.
- Also, note that the pre-build VM uses Python2.7 for invoking the mininet (e.g., the provided bufferbloat.py file in the lab4.zip and the webserver.py it calls), but you should be able to use Python3 for the remaining provided code. Hence, you may need to install both Python2 and Python3 in your environment.

What is Mininet?

- Mininet is a network emulator (mostly written in Python), that allows you to define network topologies and to connect emulated hosts.

- The emulated hosts run on your OS, and can execute your client/server applications easily.
- You can also easily change link parameters such a packet loss rate, delay, and bandwidth.
- Mininet is mainly intended for experimenting with Software Defined Networks and OpenFlow, but it is also convenient for this lab session.
- More details at <http://mininet.org>, in particular <http://mininet.org/walkthrough/>
- If you like videos: some basic things of this sheet are also demo'ed in <https://www.youtube.com/watch?v=jmlgXaocwiE>

2.2 Lab4 setup

- Download the lab4.zip from eDimension and unpack to some local folder in your VM, e.g., ~/lab4/.
- Open a terminal, and change into that folder.

Explore mininet:

- Start up mininet:

```
sudo ./run.sh
```

Note that if you are using the provided VM image, the first time you execute `sudo ./run.sh` , there may be some errors, just execute the command again it should work.

- Get familiar with mininet commands using mininet's help system:

```
mininet> help [topic]
```

- Use the following mininet command to spawn two terminals, belonging to two nodes in the emulated network.

```
mininet> xterm h1 h2
```

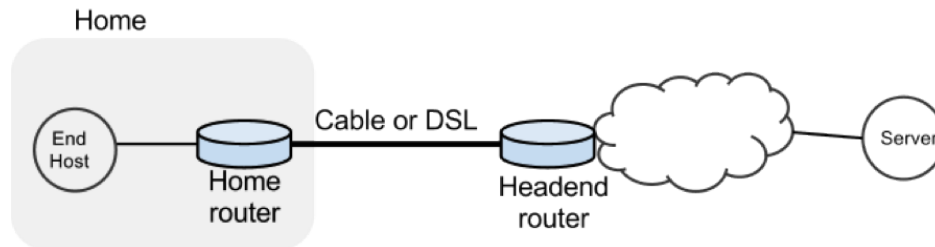
- Using these terminals, you can run applications or perform ping operations on the emulated hosts.
- To close running commands like `ping -c 100`, try CTRL-C.
- To warm up, try to find out the IP addresses of h1 and h2.
- Note that `mininet` allows to run a command from its command prompt as if the command is run from any virtual host. For example, if we want to ping h2 from h1 we can run:

```
mininet> h1 ping h2
```

- To leave/close Mininet, try CTRL-D on the main command line

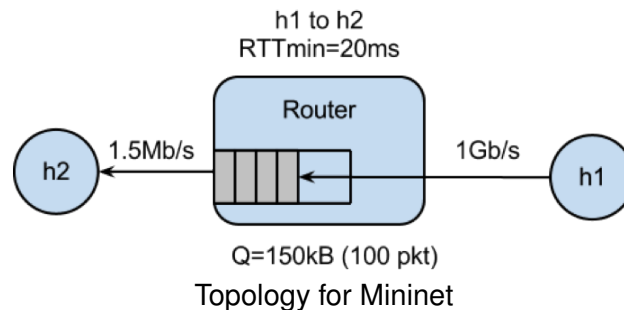
3 TCP congestion control and the impact of buffer size

In this exercise we will study the dynamics of TCP in home networks. Take a look at the figure below 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.



Problem overview

In a real network it is hard to measure the TCP congestion window $cwnd$ at the server (because it is private to the Server) and the buffer occupancy at the router (because it is private to the router). Luckily, we have mininet, which allows us to get these values easily.



Topology for Mininet

3.1 Simple TCP connection (wget)

- Start up mininet

```
sudo ./run.sh
```

- In the running mininet emulation, test how long it takes to download a webpage on h1 from h2.
- wget's visualization may be a bit confusing. You can find the total time by subtracting the start time from the end time.

```
mininet> h2 wget h1
```

- We discussed the TCP congestion control, what is your expectation on how the $cwnd$ changes during this HTTP session?

3.2 With parallel load (iperf)

- To see how the dynamics of a long flow (e.g., streaming a video) differs from a short flow (which may not leave the slow-start state), we are going to set up a long-lived high speed TCP connection, to emulate a long-lived video streaming flow.
- You can generate long flows using the `iperf` command. `iperf` is a tool for performing network throughput measurements. It can test either TCP or UDP throughput. To perform an `iperf` test the user must establish both a server (to discard traffic) and a client (to generate traffic). We have wrapped `iperf` in a script that 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.
- Think about the congestion window of the TCP stream again. How will it look for a long-established connection?
- You can also use ping to observe RTT while `iperf` is running

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

The Impact on the Short Flow

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

```
mininet> h2 wget h1
```

- Why does the web page take so much longer to download?

3.3 Measuring the cwnd and buffer occupancy values

We provided a script that lets you measure cwnd and buffer occupancy values. We are going to re-run a couple of the experiments and plot the values.

1. Restart Mininet

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

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

2. Monitor TCP CWND and Buffer Occupancy in Mininet

- Open a second terminal tab, type the following command and give a name to your experiment.

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

- In your first tab with the running mininet, start iperf again

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

- (wait for 70 seconds ...)

```
mininet> h2 wget h1
```

- 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 <EXP_NAME>_tcpprobe.txt and the buffer occupancy in <EXP_NAME>_sw0-qlen.txt.

3. Plot CWND and Queue Occupancy

- Plot the TCP cwnd and queue occupancy from the output file

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

- 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 provided by the script to access your figures.
- If you are unable to see the cwnd, ensure you run wget after you start the monitor.sh script.

3.4 Smaller buffer: from 100 packets to 20 packets

- Stop any running Mininet. This time we will make the buffer at the router smaller (from 100 packets in previous experiments to 20 packets), by using the following script:

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

- Repeat the earlier simple tests

```
mininet> h2 wget h1  
mininet> h1 ping -c 10 h2
```

- Do the results change?
- Close mininet, restart the experiment and start monitoring again

```
sudo ./run-minq.sh  
sudo ./monitor.sh <EXP_NAME>
```

- Re-run the experiment:

```
mininet> h1 ./iperf.sh  
mininet> h1 ping -c 30 h2  
mininet> h2 wget h1
```

- What do you think the cwnd and queue occupancy will be like in this case?
- Plot the figure for cwnd and queue occupancy, this time using:

```
./plot_figures_minq.sh <EXP_NAME>
```

- Then again, use the url to see your figures. Why does reducing the queue size reduce the download time for wget?

4 What to hand in

4.1 eDimension submission:

- Please provide a writeup (PDF format with your name) that includes the following information:
 - What is the normal time required to download the webpage on h1 from h2?
 - What was your initial expectation for the congestion window size over time?
 - After starting iperf on h1, did you observe something interesting in the ping RTT?
 - After starting iperf on h1, why does the web page take so much longer to download?
 - Please provide the figures for the first experiment (with qlen 100).
 - * Please comment on what you can see in the figures.
 - Please provide the figures for the second experiment (with qlen 20).
 - * Please comment on what you can see in the figures and what is different from the previous experiment. Explain the reason behind the difference.