CS 485/ECE 440/CS 585 Fall 2017 Lab 4, part 1

Lab 4 part 1 is due by 11:59pm on Monday, November 27th, 2017. Part 1 is worth 100 points, and part 2 will be worth 100 points, so in total Lab 4 is worth 200 lab points. Part 1 is intended to be a simple assignment, there are a lot of moving parts but try to keep it simple and make your writeup as simple as possible.

You'll submit one files (a PDF writeup) as an attachment to an email to crandall@cs.unm.edu. Failure to follow these instructions will result in a zero on the assignment. Send real email attachments, when you send "SharePoint" or Google Drive links or whatever, I won't click on those links. Each student will submit independently.

You are expected to do your own work. From setting up the VMs to capturing PCAPs for your figures to writing up the writeup, for all phases of this project you should do your own work. Any instance of not doing your own work will be considered cheating. If you're not sure whether something will be considered cheating or not, ask me before you do it. For the writeup, all graphics and figures used in your writeup should be your own (no using other people's graphics, not even if you cite it), and all writing should be your own writing. You are encouraged to discuss the assignment with your classmates at any level of abstraction you like, so long as two things are true: 1) nobody else but you is typing on the keyboard or doing anything to configure your VMs; 2) you're not typing anything or making any changes that you don't understand. As long as those two things are true, feel free to explain to each other how the subnetting is working, compare quagga and ripd configurations, look at each other's network configurations, share ideas for troubleshooting, or anything to help each other get your networks operating correctly. Exchanging tools, source code that existed before the assignment was assigned, and general thoughts about approaches to specific problems is okay. As a reminder of the course policy, if you cheat on any assignment in this class including this assignment (cheating includes, but is not limited to, representing somebody else's work as your own or having someone else do the assignment for you) you will receive an F in the class. If you want to share source code written for the assignment with a classmate, you should get my permission first and share it with the whole class.

DO NOT START ON LAB 4 UNTIL YOU'RE DONE WITH LAB 3.

For Lab 4 part 1, you'll need to follow these overall steps, which are explained in more detail below:

1. Get yourself networked to two more classmates besides your Lab 3 partner, so three total, using dynamic routing.

2. Set up your middle router to limit bandwidth with a pair of token bucket filters (one for each of eth1 and eth2), and then also set up your fifth machine (the one with .50 as its IP address, gummo for me) to use Reno instead of CUBIC as its TCP congestion control algorithm.

3. Collect as much evidence as you can that you're connected to the class network through three other students. This can include traceroutes, PCAPs, dumps of routing tables, or whatever.

4. Create PCAPs (probably using iperf) where your first machine is the client and your fifth machine is the server, and vice versa (*i.e.*, your first machine is the server and your fifth machine is the client) that you can use to illustrate the difference between TCP CUBIC and TCP Reno.

For Step 1, DO NOT USE IP ADDRESSES THAT WEREN'T ASSIGNED TO YOU OR SOMEONE

YOU CONNECT DIRECTLY TO. In order to connect to two more students with dynamic routing, you'll follow the same steps as you did for Lab 3. *I.e.*, you'll email Rudy to add eth4 and eth5 connections to the new "partners," you'll configure those interfaces on new /28s that aren't in use, update your quagga and ripd configuration files, and then restart networking. You can select what other two students you connect to to reach three total, and you can connect to a fourth student if you like. Feel free to ask people to see evidence that their network is working as per the end of Lab 3 before agreeing to connect to them.

For Step 2, detailed instructions are at the end of this document. Make sure you do not make any spelling errors in your /etc/network/interfaces files.

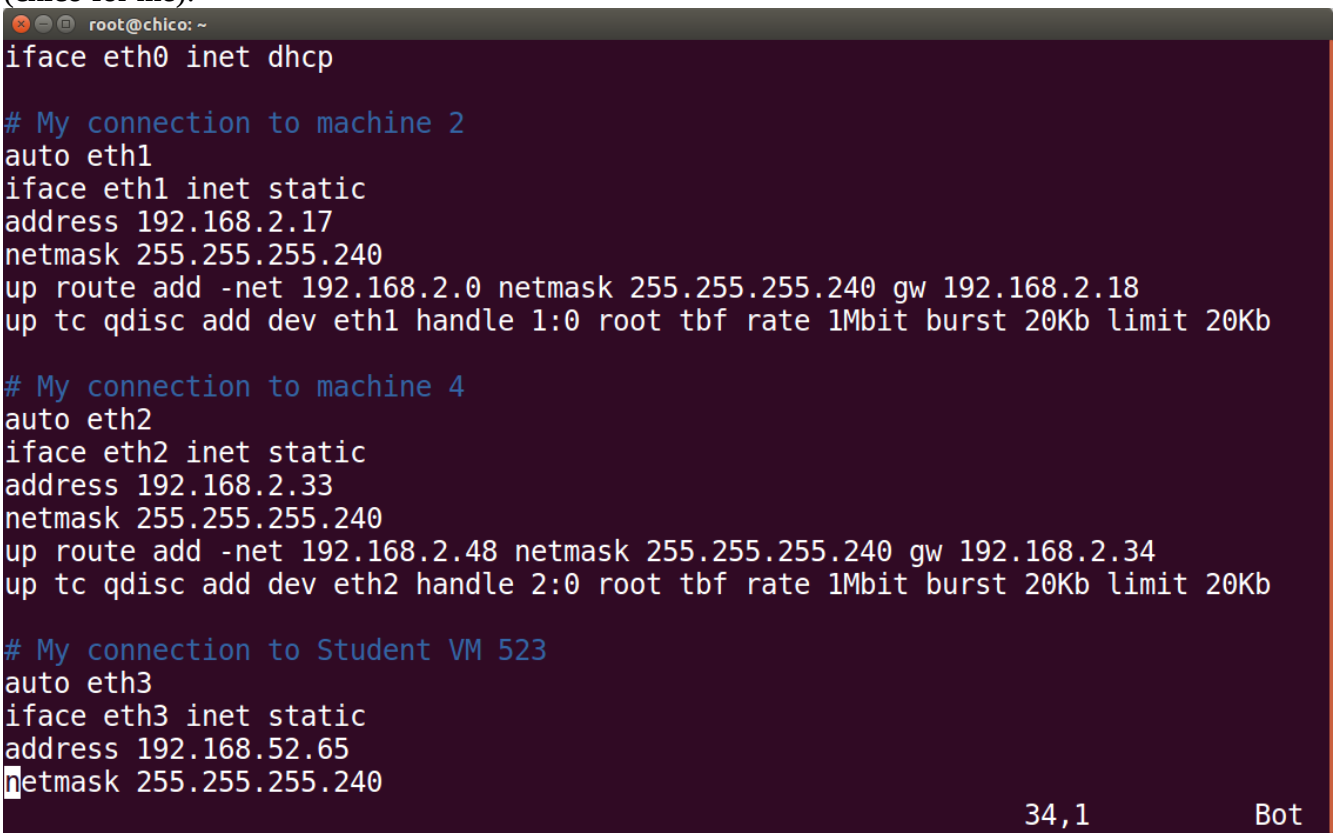
For Step 3. you can use screencaps of traceroutes and routing table dumps, redirect them to text output and import them into your document, or whatever you like. No need to send me any files other than your PDF writeup. PCAPs are not strictly necessary, but you can screencap PCAPs or Wireshark graphics and such.

For Step 4, keep it simple. Just use iperf in both directions and try to illustrate the difference between Reno and CUBIC by highlighting the difference in two graphics, *e.g.*, the bandwidth over time graph (which you can screencap from Wireshark). You can keep it really simple, circling the additive increase of Reno and the cubic function of CUBIC and saying, "Look, they're different because of how they look here" is all you really need to do.

Create a writeup showing (1) your evidence that you're connected to the class network as well as (2) the difference between Reno and CUBIC. Keep it simple, and try to use just a few figures and not many words. Remember that I have to grade 74-ish of these. This assignment sounds complicated, but it really doesn't need to be. Just show me a few graphics and say, "Figure 1 shows me tracerouting through student X to student A, Figure 2 shows me tracerouting through student Y to student B, and Figure 3 shows me tracerouting through student Z to student C. Figure 4 is a screencap of my routing table dump in a tiny font, and Figure 5 is a screencap of this cool Wireshark feature that shows my connections into the network." Then for the next part just say, "In Figure 6 I've circled the additive increase of Reno, and in Figure 7 I've circled the cubic function used by CUBIC." That's really all the writing you need to do. If your writing is similar to mine, that's no big deal (just don't plagiarize from anybody but me ;-).

Detailed instructions for Step 2 follow on the next page…

First, add transport control (tc) queuing disciplines (qdiscs) for eth1 and eth2 on your middle router (chico for me):



The handle is just in case you want to attach more tc rules (you don't). Root is the root tc rule that you want to attach to. TBF stands for token bucket filter, it's a way of limiting bandwidth (Google it for details). Then there's the rate you want to limit bandwidth to and a couple of other parameters related to burstiness. The effect of these rules is that any traffic going out on eth1 or eth2 will be limited to 1 megabit per second of bandwidth.

Now restart your networking on your machine 3 with…

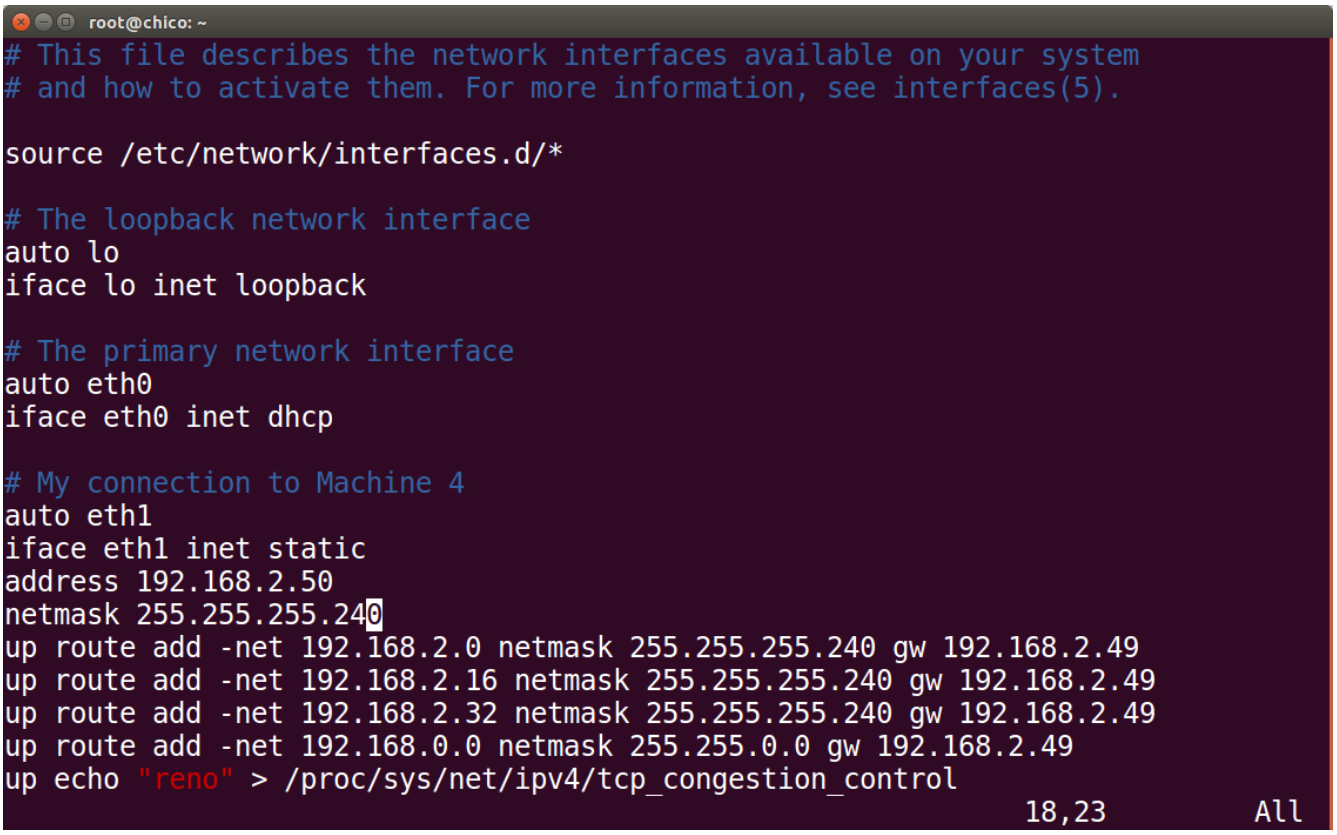
```
sudo /etc/init.d/networking/restart
```

...and then use iperf on your machines 1 and 5 to test that the bandwidth limits are working in both directions:

```
root@chico: ~
cs585user@192.168.2.2's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-97-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

36 packages can be updated.
1 update is a security update.


*** System restart required ***
Last login: Wed Nov  1 10:27:01 2017 from 172.16.0.2
cs585user@groucho:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.2.2 port 5001 connected with 192.168.2.50 port 49568
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec  4.33 GBytes  3.72 Gbits/sec
[  5] local 192.168.2.2 port 5001 connected with 192.168.2.50 port 49570
[  5]  0.0-14.1 sec  1.62 MBytes   967 Kbits/sec
```

```
root@chico: ~

*** System restart required ***
Last login: Wed Nov  1 10:29:03 2017 from 172.16.0.2
cs585user@gummo:~$ iperf 192.168.2.2
iperf: ignoring extra argument -- 192.168.2.2
Usage: iperf [-s|-c host] [options]
Try `iperf --help' for more information.
cs585user@gummo:~$ iperf -c 192.168.2.2
------------------------------------------------------------
Client connecting to 192.168.2.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.2.50 port 49568 connected with 192.168.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  4.33 GBytes  3.72 Gbits/sec
cs585user@gummo:~$ iperf -c 192.168.2.2
------------------------------------------------------------
Client connecting to 192.168.2.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.2.50 port 49570 connected with 192.168.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-12.1 sec  1.62 MBytes  1.13 Mbits/sec
cs585user@gummo:~$
```

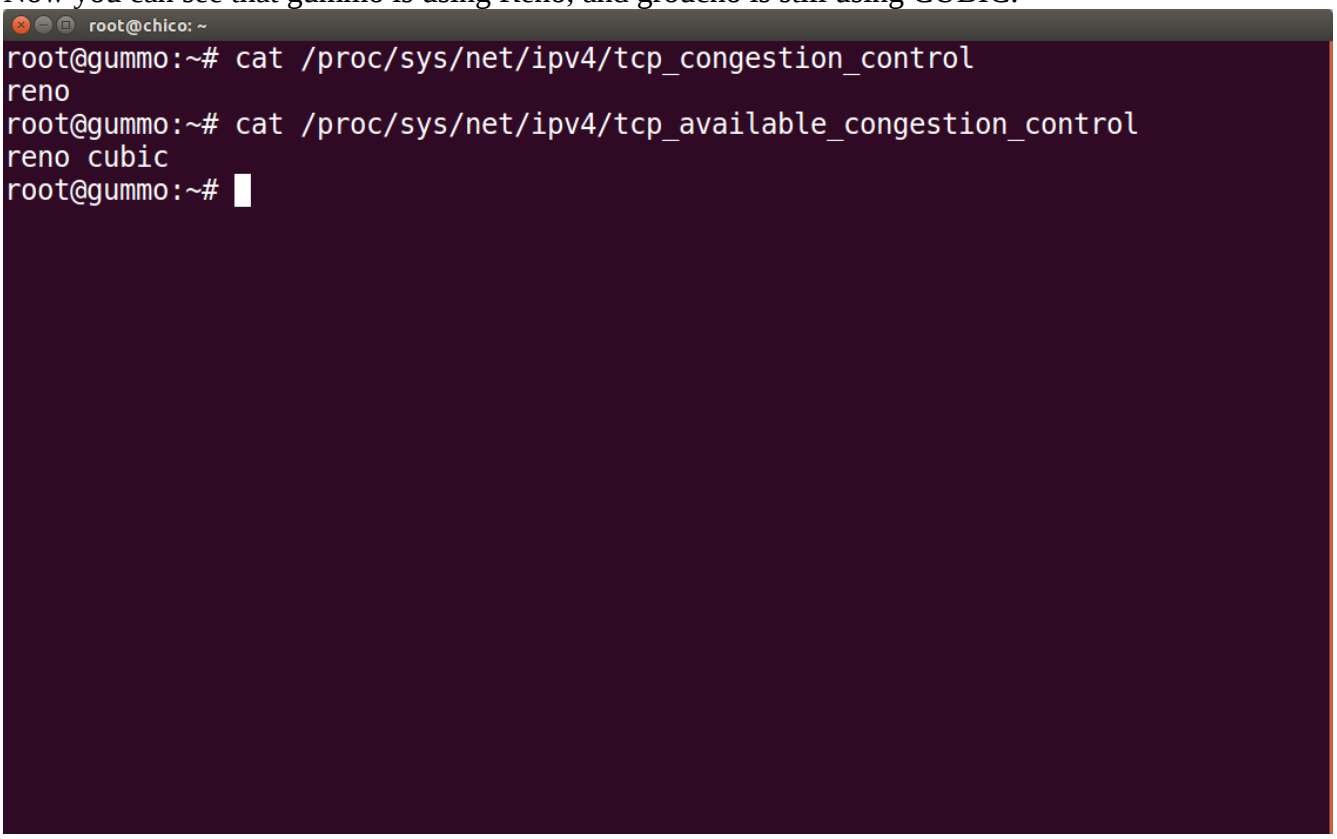
Note that the previous two screencaps only tested in one direction, switch which is the client and which the server to test in the other. Next, we want to configure machine 5 (gummo for me) to use TCP Reno instead of TCP CUBIC (CUBIC is the default for Linux) as the TCP congestion control algorithm. Edit /etc/network/interfaces on that machine to look like this:

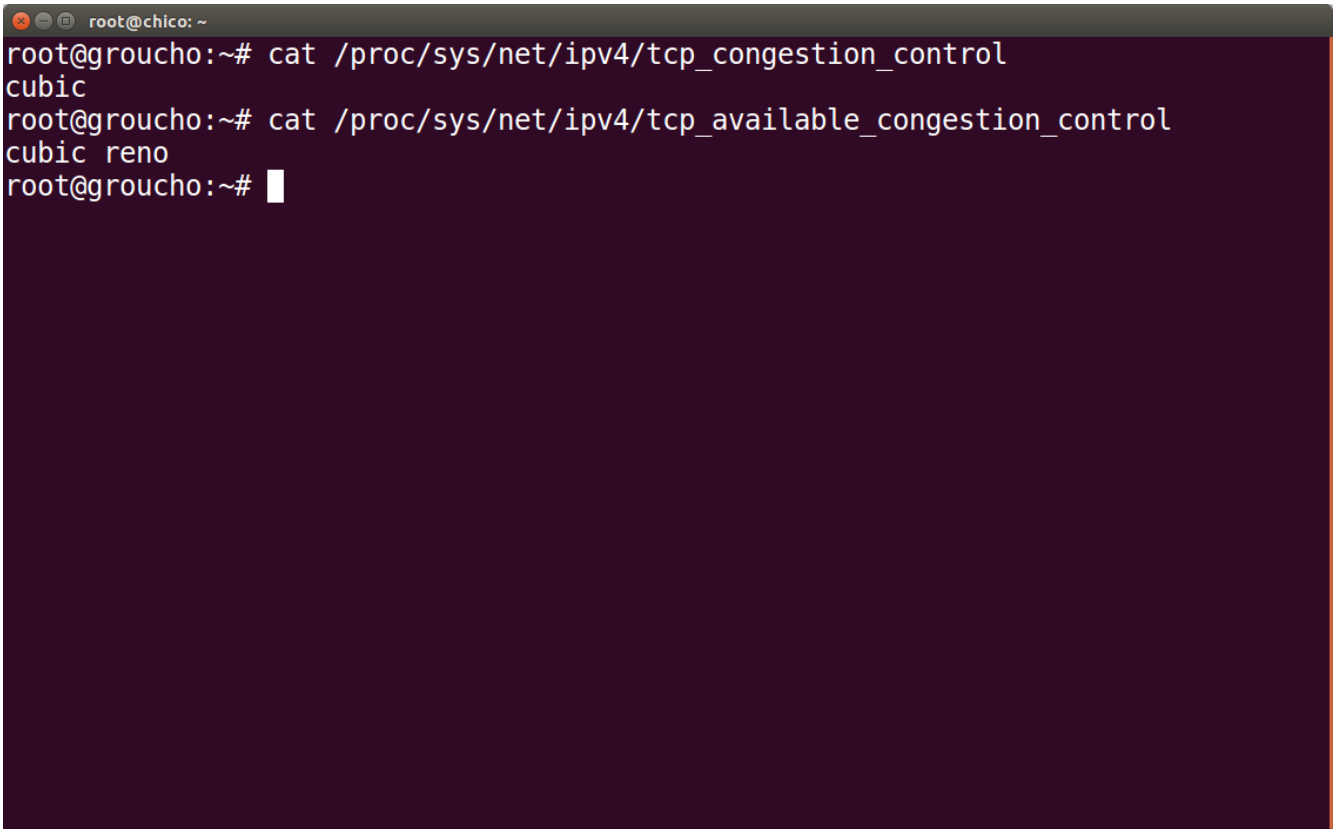


We added the last line that updates a kernel variable using the /proc filesystem to make it so that all TCP sockets on this machine default to Reno instead of CUBIC. Restart networking on gummo:

```
sudo /etc/init.d/networking/restart
```

Now you can see that gummo is using Reno, and groucho is still using CUBIC:





If you repeat the iperf tests in both directions now, you should be able to see a difference between Reno and CUBIC already in that CUBIC is more aggressive about finding available unused bandwidth. Now just find a way to graph that (I suspect that a pair of bandwidth over time graphs in Wireshark will be good enough).