Performance Study of TCP Congestion Control

EE 382N, Spring 2019

Dr. Ramesh Yerraballi

The University of Texas at Austin

Due: Sunday 4/7 (11:59pm)

Objective

Study the performance of various TCP congestion-control algorithms using `mininet`.

Environment

To study performance of end-to-end applications, one can build real applications and insert performance monitoring code in them and create periodic dumps and analyze the dumps offline. This requires that you run your client and server in two different locations in the Internet, a highly impractical proposition. An alternative is to simulate the client and server applications, the TCP/IP stack and the Internet, clearly another complex and time-consuming endeavour.

The most practical alternative is to use the technology of Virtualization to create an environment that emulates the Internet. Mininet (mininet.org) is one such tool that allows you to create an "Instant Virtual Network on your Computer".

The setup of the testbed requires the following steps:

1. Install and Virtualization software - VirtualBox is free
2. Download and Install the mininet (32-bit version) vdi - See Option1 here: http://mininet.org/download/
3. Boot up the VM you just created (give it a name) and make some updates to it:
    1. Run `sudo apt-get update`
    2. Install VirtualBox guest additions (`sudo apt-get install virtualbox-guest-dkms`)
    3. Install the X Window system (`sudo apt-get install xinit lxde`) (optional)
    4. Change the name of the machine from `mininet` to a name of your choice:
        1. modify the `/etc/hostname` and `/etc/hosts` file to accomplish this
        2. Reboot
4. Change the Network settings for the VM to be "`Bridge Adapter`" from the default which is "`NAT`"

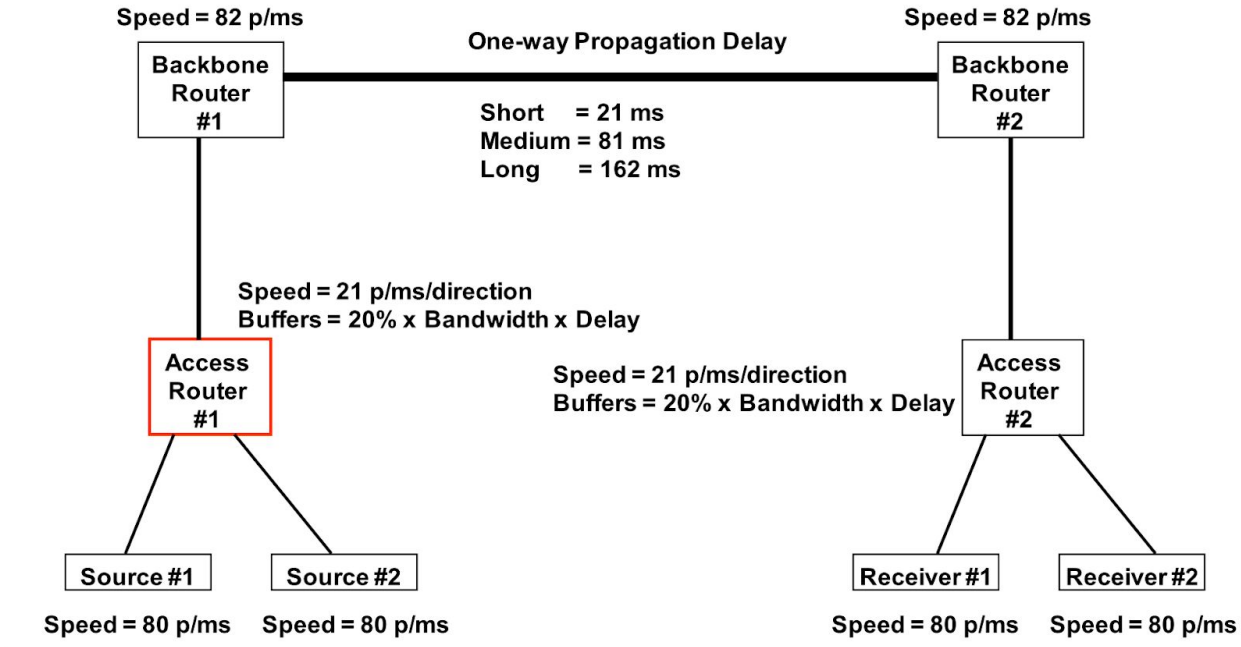
Performance Study

The topic of TCP congestion-control has been studied extensively with many (15) implementations in my version(4.4.0) of Linux:

```
ramesh@linuxmint ~/EE461S/s19/pintos-starter/userprog/build $ ls /lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp*
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_bic.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_cdg.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_dctcp.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_diag.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_highspeed.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_htcp.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_hybla.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_illinois.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_lp.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_probe.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_scalable.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_vegas.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_veno.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_westwood.ko
/lib/modules/4.4.0-21-generic/kernel/net/ipv4/tcp_yeah.ko
```

Your job is to compare four Congestion Control algorithms . Two of them must be Reno and Cubic. The other two must be two most recent algorithms implemented in Linux. You must write a brief writeup on each of the algorithms summarizing their main features.

that you can get to run in your Linux environment and compare them as done in Chapter 5 of this online book from NIST: https://www.nist.gov/document-15389. Using the dumbbell topology shown in Figure 5-6 (reproduced here:)



The specific plots are:

1. Cwnd vs time for all 4 TCP Congestion Control algorithms. Plot for 3 different propagation delays on the backbone (short:21ms, medium:81ms and Long:162ms).

2. TCP Fairness showing how the two TCP connections Bandwidth are equal over a long period of time.

Your report must comment on the plots stating your observations. Also, compare your plots against those in the NIST book which itself attempts to reproduce results from the paper by Li et. al [1]

Tools

These are the tools you will need to get the comparison plots:

1. **mininet**: To get the most out of mininet you will need to use the python API: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#working
2. **iperf**: To create a long-running TCP connection that transfers large amounts of data between a client and a server. (iperf3 is supposed to be a newer, better version but iperf will do). Here is an excellent document on how to use iperf in mininet: http://csie.nqu.edu.tw/smallko/sdn/iperf_mininet.htm
3. **tcpprobe**: Is a module that records the state of a TCP connection in response to incoming packets. It works by inserting a hook into the `tcp_recv` processing path so that the congestion window (`cwnd`) and sequence number can be captured. Here is an example of how to use tcp_probe: https://wiki.linuxfoundation.org/networking/tcpprobe and here is a desciption of the output that tcp_probe generates:

Field	Explanation
`Time`	Time (in seconds) since beginning of probe output
`Sender`	Source address and port of the packet, as IP:port
`Receiver`	Destination address and port of the packet, as IP:port
`Bytes`	Bytes in packet
`Next`	Next send sequence number, in hex format
`Unacknowledged`	Smallest sequence number of packet send but unacknowledged, in hex format
`Send CWND`	Size of send congestion window for this connection (in MSS)
`Slow start threshold`	Size of send congestion window for this connection (in MSS)
`Send window`	Send window size (in MSS). Set to the minimum of send CWND and receive window size
`Smoothed RTT`	Smoothed estimated RTT for this connection (in ms)
`Receive window`	Receiver window size (in MSS), received in the lack ACK. This limit prevents the receiver buffer from overflowing, i.e. prevents the sender from sending at a rate that is faster than the receiver can process the data.

Note: using iperf3 will save you from having to use tcp_probe but there have been some reported bugs in the iperf3 tool.

Submission

Your submission must have two things in it:
1. A document with you plots and discussion.
2. All mininet code and corresponding scripts you wrote to generate the plots.

Upload a single zip file with the above contents.

References

1. Paper: “Experimental Evaluation of TCP Protocols for High-Speed Networks”, by Yee-Ting Li, Douglas Leith and Robert N. Shorten [http://www.hamilton.ie/net/eval/ToNfinal.pdf]
2. Paper: “Performance Analysis of TCP Congestion Control Algorithms”, by Habibullah Jamal, Kiran Sultan[http://www.wseas.us/journals/cc/cc-27.pdf]