Assignment 4 - Parking Lot

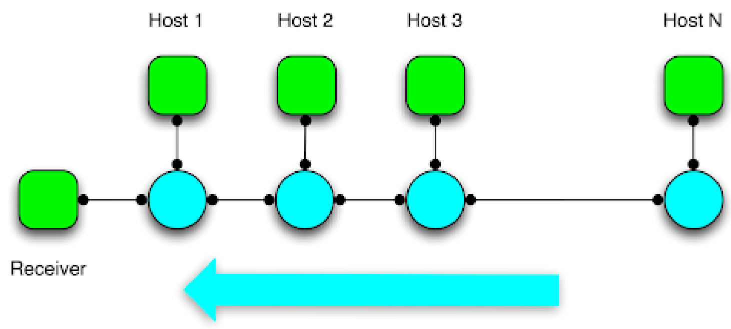
Goal

In this assignment, you will build another custom topology and learn about TCP congestion control. You'll also learn about the TCP sawtooth and see how TCP works to share bandwidth across multiple flows [1].

Over 80% of peak traffic on the Internet is transmitted via TCP [2]. Therefore, it is important to understand how TCP's congestion control helps prevent congestion collapse in networks. Congestion collapse occurs when packets use a large amount of bandwidth in one section of the network but are dropped at a downstream link. This means the network is congested at the downstream link and the senders must reduce their sending rate for packets to not be dropped. TCP's congestion control responds to these packet drops and enables a congested section of the network to recover.

Overview

To explore congestion control, you will create the following topology for N = 1, 2, 3, 4, 5 at a link speed of 10 Mbps per link and a delay of 1 ms (note the 10 Mbps speed differs from the lecture video to ensure the topology runs smoothly on a virtual machine). You will then need to generate simultaneous TCP Reno flows using `iperf`, from each of the sender hosts to the lone receiver, and record the achieved throughput. Note that `iperf` defaults to TCP Reno so you do not need to set a command line option. A provided script will plot the time series of throughput vs time for each sender, for each experiment (N = 1, 2, 3, 4, 5).



Directions

1. Log in to your Mininet instance and change to the `assignment-4` directory. Then install the following packages for this assignment:
   `sudo apt-get -y install python-argparse`
   `sudo easy_install termcolor`
2. Update to the latest assignment code:
   `git commit -a -m "Saving work"`
   `git pull --rebase`
3. Now open the `parkinglot.py` file. You will need to complete two functions within the file to complete the assignment. Both functions are marked with `TODO` comments for your convenience.
4. Complete the `__init__` function of the `ParkingLotTopo` class. The class defines the topology used in the assignment. The framework code creates a parking lot topology for N=3. Your task is to generalize it for any N >= 1.
5. Complete the `run_parkinglot_expt` function. The function generates TCP flows between the senders and the receiver using `iperf`, and monitors the throughput of each flow. Your task is to start and stop `iperf` for the additional senders in the topology.
6. To verify your topology code works correctly use the following command:
   `sudo python parkinglot.py --bw <link_bandwidth> --dir <output_dir> -t <expt_duration> -n <n>`

7. When you're confident your code works correctly, run `parkinglot-sweep.sh` to generate a set of plots and the necessary log files:
   ```
   sudo ./parkinglot-sweep.sh
   ```
   Observe how the cwnd.png shows the congestion window size for TCP changing over time and the rate.png plot shows the bandwidth across the switches over time.
   You can view the log files and plots using the Python web server and navigating to `http://ip_address:8000`:
   ```
   python -m SimpleHTTPServer
   ```
8. To complete the assignment, submit your topology file and the `bwm.txt` file, which contains the bandwidth measurements, for each of N = 2, 3, 4, and 5 on the assignment submission page.
9. Before you move on, take a stab at answering the quiz questions below and feel free to discuss your responses in the forum.

Quiz questions

1. See the cwnd plot for N = 1. Why is the "additive increase" part of the sawtooth line curved and not straight?
2. Now, see the cwnd plots for other values of N. As N increases, should the cumulative cwnd sawtooth show higher or lower variance? Why?
3. In your opinion, what should each host's bandwidth share be? In your rate plots (rate.png) for various values of N, what bandwidth share does each host get? If all hosts used UDP instead, what share would each host get?
4. If one of the hosts started more than 1 flow to the receiver, how would it affect other flows? What if that host used UDP instead of multiple TCP flows?

Rubric

This rubric is here to help you understand the expectations for the assignment. It is the same rubric that the person evaluating your project will use. We recommend you look at the rubric before you begin working on your project and again before you decide to submit it.

Criteria	**Does Not Meet Expectations**	**Meets Expectations**	**Exceeds Expectations**
Parking lot code			
Complete the topology as described in steps 3 and 4.	The topology is not generalized for N >= 1. The iperf flows are not started between the receivers and host.	The topology is generalized for N >= 1 and the iperf flows are correctly started.	There is no "Exceeds Expectations" option for this criteria.
Topology bandwidth output			
Output looks reasonable.	Output does not pass the sanity check when it is submitted. The bandwidth logs do not show equal share of bandwidth between flows.	Output does pass the sanity check. The bandwidth logs show the hosts sharing an equal share of the total bandwidth.	There is no "Exceeds Expectations" option for this criteria.

Notes

[1] Based on CS 244 assignment.
[2] Sandvine Global Internet Phenomena Report, Fall 2011. TCP is largely video over HTTP (along with some web browsing and P2P.) BitTorrent over uTP and UDP accounts for the bulk of UDP traffic.

POPULAR NANODEGREE PROGRAMS +

STUDENT RESOURCES +

UDACITY +

INQUIRIES +