Logout

My Workspace | CS-6250-O01 FALL14 | CS-6300-O01 FALL14

ASSIGNMENTS

- Home
- Syllabus
- Announcements
- Resources
- Assignments
- Gradebook
- Email Archive
- Roster
- Site Info
- Section Info
- Piazza
- Help

Assignment 3 - Parking lot - Returned

Title	Assignment 3 - Parking lot
Student	Telfort, Dominique Gayot
Submitted Date	Sep 12, 2014 11:12 pm
Grade	11.5 (max 10.0)

Instructions

Assignment 3 - Parking Lot

Goal

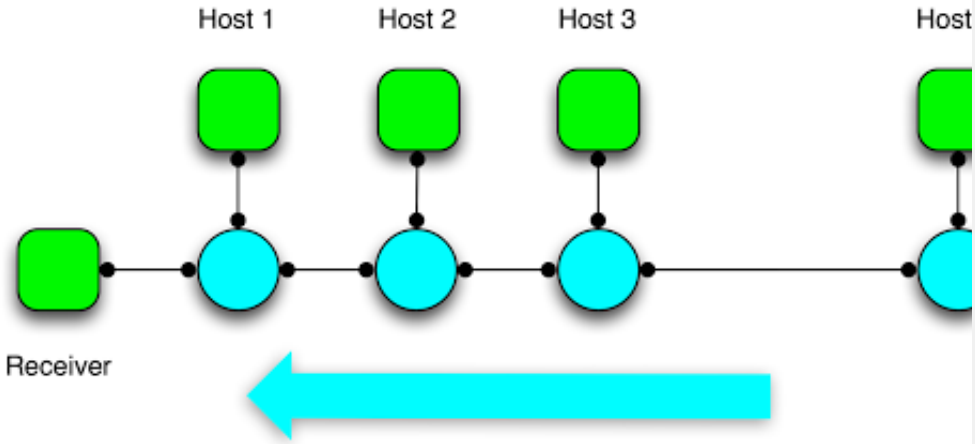
In this assignment, you will build another custom topology and learn about TCP congestion control. You'll also learn about the TCP sawtooth and see how TCP works to share bandwidth across multiple flows [1].

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 CUBIC flows using `iperf`, from each of the sender hosts to the lone receiver, and record the achieved throughput. Note that `iperf` defaults to TCP CUBIC 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-3` 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: (without the '<' and '>' `sudo python parkinglot.py --bw <link_bandwidth> --dir <ou`
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. Please answer the quiz questions below in a file `quiz.txt`. Answers should be of the format `1. X` where X is the answer.
9. To complete the assignment, submit your topology file, quiz answers, and the `bwm.txt` files, which contains the bandwidth measurements, for each of N = 2, 3, 4, and 5 on T-Square. Rename each `bwm.txt` file `bwmN.txt`, so for N = 2, you should submit `bwm2.txt`.

Quiz questions

1. See the cwnd plot for N = 1. Why is the "additive increase" part of the sawtooth line curved and not straight?

   A. Data variance from time to time, should be straight.

   B. It is called “slow-start”, and it is also known as the exponential growth phase, TCP congestion window is increased by the number of ACK received. C. For that part, TCP congestion window is increased by one every round.

   D. It is known as the linear growth phase

   E. None of above

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?

   A. Lower variance, it is easier to assign bandwidth equally to more TCP flows.

   B. Higher variance, it is more difficult to assign bandwidth equally to more TCP flows.

   C. Higher variance, for about tens of TCP flows, usually they are

synchronized, therefore high variance would be observed.

D. Lower variance, for about thousands of TCP flows, usually they are unsynchronized, therefore lower variance would be observed.

E. None of above

3. What should each host's bandwidth share be for TCP flows?

   A. Hosts should share bandwidth equally.

   B. Host which starts TCP flow earlier would get more bandwidth.

   C. Host which is near to the receiver would get more bandwidth.

   D. Hosts would share bandwidth randomly.

   E. None of above

4. If all hosts used UDP instead, what share would each host get?

   A. They also should share bandwidth equally.

   B. Half of them will share bandwidth equally, the other half will get 0 bandwidth.

   C. They would share bandwidth randomly.

   D. The first host starting UDP flow would get the total bandwidth.

   E. None of above

5. If one of the hosts started more than 1 TCP flow to the receiver, how would it affect other flows?

   A. Nothing new would happen

   B. Other flows would get 0 bandwidth, because one host tries to send more than one TCP flow.

   C. Other flows would get more bandwidth.

   D. Bandwidth will be shared in this way, each flow would get its

Copyright 2003-2011 The Sakai Foundation. All rights reserved. Portions of Sakai are copyrighted by other parties as described in the Acknowledgments screen.
T-Square - gatech-sakai-2-8-x-10 - Sakai 2.8.x (Kernel 1.2.5)- Server pinch8.lms.gatech.edu