

Hands-On Assignment 2

*Instructor: Prof. Nick Feamster**College of Computing, Georgia Tech*

This problem set has 2 parts: Setting up software routers, and injecting link failures. Answer the parts as clearly and concisely as possible. You may discuss ideas with others in the class, but your solutions and presentation must be your own. Do not look at anyone else's solutions or copy them from anywhere. (Please refer to the Georgia Tech honor code, posted on the course Web site).

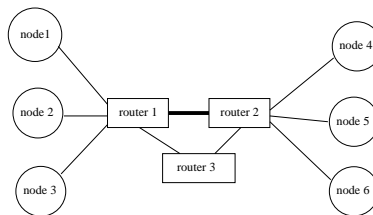
Turn in your solutions in on **November 3, 2008** by 11:59pm (please email PDF with subject "CS4251 Hands-On 2").

You are welcome to work with one other partner on this problem if you want, provided you (1) work out all the problems yourself; (2) type up your own assignment; (3) list on your assignment hand-in the name of the person you worked with.

Experience with Software Routers in Emulab

This problem will give you hand-on experience with Layer 2 and Layer 3 protocols, and with setting up network experiments in Emulab. The experimental configuration language is based on Tcl, which is the same as that which is used for *ns*, a network simulator.

In this problem, you will configure the Quagga (<http://www.quagga.net/>) software router's OSPF module to connect the nodes in your topology.



Setting Up Your Routers

1. Set up the point-to-point Emulab topology shown above. Your topology should now be as before: each node should be able to ping its immediate neighbors, but no other nodes in the topology.
2. Install and configure OSPF on each of the nodes in the topology. Configure the topology so that each edge has unit link cost. (If you want to save work, it's fine if you only install OSPF on *node1*, *node4*, and routers 1–4, but you will see more of the flooding effects if you install everywhere. You can also save work by using a Perl, Ruby, or Python script to generate your configuration.)
3. How often do you see OSPF "Hello" messages? How often do you see an LSA message for any single link?

Turn in:

- Your OSPF configuration for one of the nodes.
- Answers to the questions above.

Convergence Under Failure

1. Set up a mechanism to automatically inject *link failures* on the bold link in the topology. Explain how you achieved your mechanism. List two other ways that you might have emulated a link failure.
 - How long does OSPF take to converge to the new topology?
 - How might you reduce convergence time? Which Quagga command would you use to accomplish this?

(*Hint:* There are multiple ways to achieve this, some of which are provided by Emulab itself.)

2. Set up a mechanism for injecting *probabilistic packet loss* on the bold link in the topology. This may not be so straightforward: You will have to somehow figure out how to intercept traffic before it traverses the link and drop some fraction of the packets.

You may find the following features helpful:

- Linux divert sockets.
<http://www.faqs.org/docs/Linux-mini/Divert-Sockets-mini-HOWTO.html>
 - The `RandomSample` Click element.
3. Use `iperf` (<http://sourceforge.net/projects/iperf>) to send a constant bit rate TCP stream between two hosts on either end of your topology (i.e., traffic should cross both intermediate nodes). Run a 15-second transfer for each loss rate.

Vary the packet loss rate from zero to 20%. At what packet loss rate does TCP start to experience poor throughput?

Turn in:

- Description of how you failed the links (include scripts if appropriate).
- Your divert sockets configuration.
- Answers to the questions above, including a table showing TCP throughput vs. loss rate.