*CSE 473 – Introduction to Computer Networks*

# Studio 5

(*Adapted from Jon Turner’s Studios*)

In this studio, you’ll be implementing a simple overlay IP forwarder that routes packets randomly. Start by claiming one of the studio reservations by opening the provided configuration file and selecting “commit” using one of the studio accounts.

1. Start by reviewing the provided source code with your group. Make sure you understand what all the parts are doing and how they fit together. Note that the interface provided by the *Substrate* includes a link number. Make sure you understand what these link numbers signify and how they are used (you may want to do the second part first, then come back to this). After you have done so, you may want to look at the description at the end of this document that outlines the functionality of the different scripts and configuration files, and provides some insight into where various capabilities are provided in the code.
2. In the *net* sub-directory, you will find a set of configuration files that define an overlay IP network that runs on top of the ONL hosts. Draw a picture of the network topology on the white board, labeling each the nodes *r*1, *r*2*,* etc. Label the connecting links with their “costs”. Next to each node, write it’s IP address in the overlay network and the name of the ONL host on which it runs.
3. Complete the implementation of the Forwarder thread. When sending (or forwarding) a packet, just pick a link at random using the Random.nextInt() method. You may assume that every node has exactly three links, numbered 0, 1 and 2. Packets addressed to “this” node should have their payloads delivered to SrcSnk. When forwarding a packet decrement the TTL and discard if the TTL is zero. Whenever this happens, print a message along with a copy of the packet. When first creating a packet, initialize the TTL to 15.
4. Test your program using the provided script1 (you may need to edit the script to set the “root” directory) by typing

./script1 .1 10 static debug or bash script1 .1 10 static debug

Look at the statistics in the log file: for *r*1. For each of the nodes it sends test packets to, what is the minimum delay reported? What is the maximum delay? What is the ratio between the two? Considering the link costs in your network graph, does this surprise you at all?

Determine the number of times that a node forwarded a packet with a TTL of 2. An easy way to do this is to type

grep -A1 sending log\* | grep “ttl=2” | wc

Here,wc (stands for “word count”) is a simple program that counts the number of lines, words and characters in its input. Make sure you understand how this works. Repeat this for TTLs of 3, 4, 5, 6 and 7. See anything curious?

How many times are packets discarded because their TTLs hit zero?

1. In this part, you will be using a “discard filter” that causes one of the ONL routers to discard all the packets it received on one of the inter-router links. First, run script1by typing

./script1 1 10 static or bash script1 1 10 static

Notice the values of the packet counts in the SrcSnk statistics. Now, click on port 1 of ONL router number 1 (the left-hand router) and select the “Filter Table” item. Examine the filter and note the check box at the right end. Clicking on this check box “turns on” the filter, causing the router to discard all packets received on this port. Look at your network diagram and which host connects to port1 to make sure you know which packets are blocked by this filter (note that this filter only blocks packets that it receives). Do this, and select “Commit” from the File menu. Then rerun the script. Observe the difference in the SrcSnk statistics and in the monitoring display. Why are a different number of packets sent between each node? Why do the average delays change the way they do? To turn off the filter, uncheck the box and select Commit again.

1. Try running the script without the “static” argument. How do the delay statistics compare to the earlier run? What do you expect would be the average delay between a pair of adjacent routers? You’ll need to check out the Substrate code to answer this.

Configuration Files and Scripts

Each “router” that is instantiated in this studio uses a configuration file to make set up easier and allow the use of fewer arguments in the run command. Each file looks something like:

hostIp: 192.168.4.2

myIp: 1.1.0.1

neighbor: 1.2.0.1 192.168.7.1 .02

neighbor: 1.4.0.1 192.168.5.2 .06

neighbor: 1.6.0.1 192.168.2.4 .01

destination: 1.3.0.1

destination: 1.5.0.1

The meaning of each line is as follows:

hostIp -This is the IP address of the host running the router program. In ONL, it should always be in the subnet 192.168.0.0/16.

myIp -This is the IP address of this router in the overlay network that you are creating by running this program on multiple hosts.

neighbor -This defines a neighbor to this router in the overlay network. The format for this is:

neighbor: (IP address in overlay network) (Host IP address) (cost)

Note that cost is really “delay”, which can be either static (set in TestRouter) or variable as implemented in Sender (delays will always be static in the studio, but not in the lab).

destination -This is the IP address in the overlay network of another router that this router wants to send packets to.

The script used for the studio will ssh into six hosts and starts running a router on each of them. The script will then wait for 20 seconds longer than the runtime argument, then it will print out the statistics that each of the routers produced while running the program.