

CSCI 558L Fall 2011 Lab#6

Due Date: October 8th, 11:59pm

Andrew Goodney - goodney@usc.edu - Young Cho - youngcho@isi.edu

1 Internet Routing

This project will let you gain some experience setting up static routing on a small network. The NS file you should use is attached as listing 1. We will use a network of 4 hosts and 3 routers. Note the 2nd to last line of the NS file: `$ns rtrproto Manual`. Normally this is set to `static`, and the DETER system sets the kernel routing tables for all of your hosts so they can talk to each other. Here we set it to `manual`, which means the hosts and routers do not know about the network topology. Your task is to properly configure the routers so the 4 hosts can exchange packets.

We will also be using a piece of software called Quagga (<http://www.quagga.net/>). Quagga implements several routing protocols (RIP, OSPF, BGP, etc) and provides a ‘Cisco-like’ command-line interface for configuring and interacting with the router software. Quagga runs on top of most modern UNIX-like operating systems and is made up of several encapsulated software pieces called daemons. Quagga does not actually route packets, it uses information from the implemented protocols to update the kernel routing table. The piece of Quagga we’ll use for this assignment is called **zebra**. Zebra is the management daemon responsible for updating the kernel routing table. When used with the other routing daemons (`ospfd`, `ripd`, `bgpd`) **zebra** takes updates from these daemons and applies them to the kernel routing table. However, **zebra** can also be used for managing static routes.

This homework will require you to build and use the Quagga software. This handout will not provide extremely detailed instructions, just enough to get you started. This way we hope you’ll learn the most.

This assignment uses 10 total DETER nodes so you MUST swap out or terminate your experiments when you are not using them!

- Begin by starting the experiment with the NS file shown in listing 1. Can **node1** ping **node3**? Can **node1** ping **rtr1**?
- Where does the *default* route on **node1** point? Where does the *default* route on **node3** point? All of the nodes in DETER are connected to the control network (with IP address 192.168.X.X), we need to ‘pretend’ this connection doesn’t exist. Change the *default* route for the 4 hosts to their respective router (**rtr1** or **rtr3** respectively). Use the **route** command to do so. This should be the only change necessary on the 4 hosts. Include a print out of the kernel routing table before and after you make this change. Important: before you change the *default* route, enter the following command:

```
sudo route add -host 192.168.253.1 gw 192.168.1.254
```

Otherwise once the *default* route is changed, you won't be able to connect to your node from `users.isi.deterlab.net`.

- Download the Quagga source code to your `users.deterlab.net` account. On each router node use the `mkextrafs.pl` script to make a filesystem at `/mnt`. Copy the Quagga source to `/mnt` and unpack it (also on each node).
- Build the Quagga software by executing `./configure` and `make` in the Quagga source directory. Do not install the software using `make install`. Do this on all three router nodes.
- You should now have the Quagga software built. Create a configuration file for `zebra` in `/mnt`. Call it `zebra.conf`. Below is an example:

```
!  
! Example zebra configuration  
!  
hostname zebrad  
password zebra  
!  
interface eth5  
ip address 10.10.1.1/24  
!  
interface eth7  
ip address 10.1.0.3/24  
!  
interface eth9  
ip address 10.10.2.2/24  
!  
interface lo  
ip forwarding  
!  
log stdout  
line vty  
!
```

In this file we need to tell `zebra` about the interfaces we're going to use. Note we ignore the control network. You will probably have to change the names of the interfaces to match the specific machine you are assigned by DETER.

- Perform a similar procedure on `rtr2` and `rtr3`. You will need to create the configuration file yourself. Once you have it correct, include the configuration files in your report.
- Now we need to start `zebra` on the routers. Change into the `zebra` subdirectory in the Quagga source code. Execute the following command:

```
sudo ./zebra -d -u root -f /mnt/conf/zebra.conf
```

- Now the fun begins! Lets ignore **rtr2** for now. On **rtr1** and **rtr3** we need to set static routes that tell the routers where to find the subnet behind the other router. So we need to tell **rtr1** about the 10.1.2.0/24 subnet and we need to tell **rtr3** about the 10.1.0.0/24 subnet. Look at the Quagga documentation (it's online) and figure out how to do this. Start by connecting to the **zebra** daemon:

```
telnet localhost 2601
```

To enter commands that change the configuration use **enable** followed by **configure terminal**. It should take only one command on each router (**rtr1** and **rtr3**) to enable the two LANs to exchange packets.

- Use **ping** to verify you can now exchange packets between the LANs. About how much RTT latency is there? Does this make sense?
- Include in your report the commands you used to configure the routes in **zebra**, a **traceroute** between **node1** and **node4**, the routing tables in the routers, and an explanation of what you did and why it works.
- Now lets look at **rtr2**. Can **rtr2** ping any of the nodes in the LANs? Why not? Add two routes to **rtr2** such that **rtr2** can ping nodes in the LANs. Include the commands you used and the routing table in **rtr2** in your report.
- Now go back and add a secondary route through **rtr2** on **rtr1** and **rtr3**. When you do **show ip route** on either **rtr1** or **rtr3** you should see two lines in the routing table for one entry. Similar to:

```
S>* 10.1.0.0/24 [1/0] via 10.10.1.1, eth7
                via 10.10.3.2, eth10
```

- Does adding these secondary routes change the path a packet takes from LAN to LAN? (i.e. did the **traceroute** from **node1** to **node4** change?)
- Now set the loss ratio on **link0** (the link between **rtr1** and **rtr3** to 1.0. Can you still ping between the LANs? What does **traceroute** show?
- Now change the metric (distance) for the routes in **rtr1** and **rtr3**. Set the route through **rtr2** to have a metric of 1 and change the other route (through **rtr1** or **rtr3** respectively) to have a metric of 5. When you have it correct, part of the output from **show ip route** on **rtr1** should look similar to:

```
S>* 10.1.2.0/24 [1/0] via 10.10.2.1, eth9
S   10.1.2.0/24 [5/0] via 10.10.1.2, eth5
```

- Can you now ping from LAN to LAN? About how much delay is there now? Does this make sense? How about **traceroute**? Your **traceroute** should have one hop that does not return (i.e. you get * * *). Explain why changing the metric 'fixed' the network? Explain why **traceroute** still has an error?

1.1 Discussion

- What does setting the `link0` loss to 1.0 simulate?
- How long do you think it would take you to change or update the routes manually if you worked at an ISP and a link between routers went down? Is this a problem? What if you had to change the routes on 5 routers, 10 routers? Does manually updating the routing tables scale well?
- How would protocols like OSPF solve this problem?

Listing 1: Lab #6 NS file

```

set ns [new Simulator]
source tb_compat.tcl

# Nodes
set rtr1 [$ns node]
tb-set-node-os $rtr1 Ubuntu1004-STD
set rtr2 [$ns node]
tb-set-node-os $rtr2 Ubuntu1004-STD
set rtr3 [$ns node]
tb-set-node-os $rtr3 Ubuntu1004-STD
set node1 [$ns node]
tb-set-node-os $node1 Ubuntu1004-STD
set node2 [$ns node]
tb-set-node-os $node2 Ubuntu1004-STD
set node3 [$ns node]
tb-set-node-os $node3 Ubuntu1004-STD
set node4 [$ns node]
tb-set-node-os $node4 Ubuntu1004-STD

# Links
set link0 [$ns duplex-link $rtr1 $rtr3 1000000.0kb 5.0ms DropTail]
tb-set-ip-link $rtr1 $link0 10.10.1.1
tb-set-ip-link $rtr3 $link0 10.10.1.2
set link1 [$ns duplex-link $rtr2 $rtr3 1000000.0kb 5.0ms DropTail]
tb-set-ip-link $rtr3 $link1 10.10.3.1
tb-set-ip-link $rtr2 $link1 10.10.3.2
set link2 [$ns duplex-link $rtr1 $rtr2 1000000.0kb 5.0ms DropTail]
tb-set-ip-link $rtr2 $link2 10.10.2.1
tb-set-ip-link $rtr1 $link2 10.10.2.2

# Lans
set lan0 [$ns make-lan "$node1 $rtr1 $node2" 1000000.0kb 0.0ms]
tb-set-ip-lan $node1 $lan0 10.1.0.1
tb-set-node-lan-bandwidth $node1 $lan0 1000000.0kb
tb-set-ip-lan $rtr1 $lan0 10.1.0.3
tb-set-node-lan-bandwidth $rtr1 $lan0 1000000.0kb
tb-set-ip-lan $node2 $lan0 10.1.0.2
tb-set-node-lan-bandwidth $node2 $lan0 1000000.0kb
set lan1 [$ns make-lan "$rtr3 $node3 $node4" 1000000.0kb 0.0ms]
tb-set-ip-lan $rtr3 $lan1 10.1.2.1
tb-set-node-lan-bandwidth $rtr3 $lan1 1000000.0kb
tb-set-ip-lan $node3 $lan1 10.1.2.3
tb-set-node-lan-bandwidth $node3 $lan1 1000000.0kb
tb-set-ip-lan $node4 $lan1 10.1.2.4
tb-set-node-lan-bandwidth $node4 $lan1 1000000.0kb

$ns rtproto Manual
$ns run

```