CSCI 558L – Laboratory Assignment #6: Custom Router
(Group Assignment)

Instructor: Young H. Cho
T.A.: Siddharth Bhargav

Due date: October 15, 2012 at 11:55pm

In this assignment you will build a simple IP router in user space (a.k.a your own code in C, C++ or other language that provides a libpcap and raw socket interface). See listing 1 for the NS file to get you started. In order to build an IP router, you must be able to capture a packet sent to your router, determine the next hop and then send the packet to that next hop.

You will also split your group into two sub-teams of 2 students to independently build IP routers on the first week to demo the progress at mid-point then merge the work on the second week to demo the final router on the due date during the TA's office hours.

To do each step:

- Capture packets: familiarize yourself with libpcap.
  Start here: http://www.tcpdump.org/#documentation. The third tutorial by Tim Carst is a good start.

- Determine the next hop: familiarize yourself with the theory and practice of IP routing. The last assignment should have been a good start.

- Send the packet to the next hop: familiarize yourself with the raw sockets feature provided by Linux (specifically a socket with PF PACKET as the type), or the packet injection feature of libpcap.

Note: This assignment uses 8 total DETER nodes so you MUST swap out or terminate your experiments when you are not using them!

- Start by instantiating the NS file as an experiment. Your code will eventually go in the node labeled usRTR (for user-space router). However, for now set the default route in each of the nodeX machines to their respective router node (similar to the previous assignment). Then set routes in the three router nodes so that all nodes and routers can ping each other. Note: the routes to reach node5 and node6 should be collapsed into one CIDR route. What is this route in CIDR notation? Which routers did you use it on? When you include this route in your router it must be stored in collapsed form.

- Observe the routing table in the usRTR node. This is the routing table you'll need to duplicate. Start by removing all of the routes that point to non-local IP addresses. Remember to leave the route to 192.168.253.1 or you'll lose connectivity to your node. Note which machines can no longer communicate.

- Use libpcap and raw sockets to build a router. Follow the following steps:

    - Build a routing table. It should contain the destination network, the next-hop IP, and the MAC address of the next-hop. Getting the MAC address can be tricky (google for

SIOCGARP or parse /proc/net/arp). Remember you'll need to send at least one packet to the next-hop router to get its MAC address in the arp table.

- Capture all packets sent to the router.

- If the packet is destined to a locally connected IP or a broadcast or multicast address, ignore it. The Linux kernel will have dealt with it already. { If the packet is destined to an IP address that matches a route in your table, do the following:

  - Set the source MAC address to the MAC address of the ethernet adapter on your router that will be sending the packet.
  - Set the destination MAC address to the MAC address of the next-hop.
  - Decrement the TTL of the packet. If you decrement the TTL to zero, you'll need to send and ICMP message back to the host (and drop the packet).
  - Use a raw socket to write the modified packet out the proper ethernet adapter.

- You will need to figure out how to stop the Linux kernel on the usRTR node from sending an ICMP "Destination Unreachable" message to the source of a packet on your network. Research ICMP and use TCPDump to observe the usRTR node generating these messages. Why are these generated? Why does this cause a problem? How can you block these messages? Answer these questions in your report. (Hint: you'll need to use a simple firewall rule)

- You must implement enough ICMP support to get traceroute to work properly. At a minimum, you'll need to implement ICMP message "Time Exceeded" (Type 11, Code 0).

- Test your router: once you get it working correctly your network should be fully connected again.

- Now collect the following stats for the competition aspect of this assignment: Packets per Second and Throughput

  - Calculate packets per second by using iperf to send the smallest possible UDP packet. Increase the sending rate until you start to drop packets. Use the output of iperf to calculate the maximum packets per second your router is capable of supporting.

  - Calculate throughput in a similar manner, except use the largest UDP packet as possible and use the output of iperf to calculate the throughput in megabits per second. For this assignment, use mega as 106. Answer the following question in your report: why are file transfer people interested in throughput with mega = 10242 while network people use 106?

- We used one CIDR route in this assignment. Why is CIDR important (i.e. what problem did it solve)?

You must demonstrate the progress made on your router on October 8th and demonstrate your working router including packet routing, performance tests, and correct ICMP responses with traceroute on October 15th during the TA's office hours.

Listing 1

```
set ns [new Simulator]
source tb_compat.tcl

# Nodes
set node0 [$ns node]
tb-set-node-os $node0 Ubuntu1004-STD
set node3 [$ns node]
tb-set-node-os $node3 Ubuntu1004-STD
set node4 [$ns node]
tb-set-node-os $node4 Ubuntu1004-STD
set node5 [$ns node]
tb-set-node-os $node5 Ubuntu1004-STD
set node6 [$ns node]
tb-set-node-os $node6 Ubuntu1004-STD
set rtr1 [$ns node]
tb-set-node-os $rtr1 Ubuntu1004-STD
set rtr2 [$ns node]
tb-set-node-os $rtr2 Ubuntu1004-STD
set usRTR [$ns node]
tb-set-node-os $usRTR Ubuntu1004-STD

# Links
set link0 [$ns duplex-link $node0 $usRTR 1000000.0kb 0.0ms DropTail]
tb-set-ip-link $node0 $link0 10.10.0.1
tb-set-ip-link $usRTR $link0 10.10.0.2
set link1 [$ns duplex-link $node5 $rtr2 1000000.0kb 0.0ms DropTail]
tb-set-ip-link $node5 $link1 10.1.2.3
tb-set-ip-link $rtr2 $link1 10.1.2.2
set link2 [$ns duplex-link $node6 $rtr2 1000000.0kb 0.0ms DropTail]
tb-set-ip-link $rtr2 $link2 10.1.3.2
tb-set-ip-link $node6 $link2 10.1.3.6

# Lans
set lan0 [$ns make-lan "$rtr1 $rtr2 $usRTR" 1000000.0kb 0.0ms]
tb-set-ip-lan $rtr1 $lan0 10.99.0.1
tb-set-ip-lan $rtr2 $lan0 10.99.0.2
tb-set-ip-lan $usRTR $lan0 10.99.0.3
set lan1 [$ns make-lan "$rtr1 $node3 $node4" 1000000.0kb 0.0ms]
tb-set-ip-lan $rtr1 $lan1 10.1.0.1
tb-set-ip-lan $node3 $lan1 10.1.0.3
tb-set-ip-lan $node4 $lan1 10.1.0.4

$ns rtproto Manual
$ns run
```