CS 640 Introduction to Computer Networks **Fall'25**

Assignment 3

Due at 11:59pm on: Nov 13

Overview

For this assignment, you will modify your virtual router to build a routing table using distance vector routing. With this change, your virtual router will no longer depend on a static route table.

Learning Outcomes

After completing this programming assignment, students should be able to:

1. Write code that constructs and deconstructs packets containing multiple layers of protocols
2. Explain how the distance vector (DV) routing work

1 Getting Started

You will use the same environment and code base as Assignment 2. You should create a copy of your entire `assign2` and name it `assign3`:

```
cp -r ~/assign2 ~/assign3
```

You can use the version of `Router.java`, `RouteTable.java` and `Switch.java` you wrote for Assignment 2. If you've forgotten the commands to start Mininet, POX, or your virtual router, you should refer back to Assignment 2.

As you complete this assignment, you may want to use `tcpdump` to examine the headers of packets sent/received by hosts. This will help you identify potential issues in your implementation. To record network traffic with `tcpdump` on a specific host, open an xterm window:

```
mininet> xterm h1
```

Then start tcpdump in that xterm:

```
sudo tcpdump -n -vv -e -i h1-eth0
```

This will cause tcpdump to print brief messages on the packets sent/received on the interface `h1-eth0`. You'll need to change the host number included in the interface (`-i`) argument to match the host on which you're running tcpdump. tcpdump is also capable of storing the packet contents into a capture dump for offline analysis with `-w`. Consult the manual of tcpdump (with `man tcpdump`) for advanced usage. Come to office hours if you have issue debugging with `tcpdump`.

2 Implement RIP

For this part of the assignment, you will implement distance vector routing to build, and update, your router's route table. Specifically, you will implement a simplified version of the Routing Information Protocol v2 (RIPv2). Details on the RIPv2 protocol are available from RFC2453 and Network Sorcery RFC Sourcebook. If you're not sure how distance vector routing works, you should read Section 3.3.2 of the textbook or the lecture slides on distance vector routing.

Starting RIP

Your router should only run RIP when a static route table is not provided (via the `-r` argument when running `VirtualNetwork.jar`). You should modify `Main.java` and/or `Router.java` to appropriately start RIP when required.

When your router starts, you should add entries to the route table for the subnets that are directly reachable via the router's interfaces. This subnets can be determined based on the IP address and netmask associated with each of the router's interfaces. These entries should have no gateway.

RIP Packets

The `RIPv2` and `RIPv2Entry` classes in the `net.floodlightcontroller.packet` package define the format for RIPv2 packets. All RIPv2 packets should be encapsulated in UDP packets whose source and destination ports are 520 (defined as a constant `RIP_PORT` in the UDP class). When sending RIP requests and unsolicited RIP responses, the destination IP address should be the multicast IP address reserved for RIP (`224.0.0.9`) and the destination Ethernet address should be the broadcast MAC address (`FF:FF:FF:FF:FF:FF`). When sending a RIP response for a specific RIP request, the destination IP address and destination Ethernet address should be the IP address and MAC address of the router interface that sent the request.

RIP Operation

Your router should send a RIP request out all of the router's interfaces when RIP is initialized. Your router should send an unsolicited RIP response out all of the router's interfaces every 10 seconds thereafter.

You should update the `handlePacket(...)` method in the `Router` class to check if an arriving IP packet is of protocol type UDP, and a UDP destination port of 520. Packets that match this criteria are RIP requests or responses. Your router should update its route table based on these packets, and send any necessary RIP response packets.

Your router should time out route table entries for which an update has not been received for more than 30 seconds. You should never remove route entries for the subnets that are directly reachable via the router's interfaces.

Your implementation does not need to be a complete standards-compliant implementation of RIPv2. You should implement basic distance vector routing as discussed in the textbook and in-class, using RIPv2 packets as the format for messages exchanged between routers.

Testing RIP

To test your router's control plane, you will need a topology with more than one router: `pair_rt.topo`, `triangle_rt.topo`, `triangle_with_sw.topo`, or `linear5.topo`. You should not include the `-r` argument when starting your routers, since your router should construct its route table using RIP, rather than using a statically provided route table. Start your implementation as you would in Assignment 2 Part 3, but omit the static routing table in the parameters `-r rtable.rX` so that your router knows it should use RIP to find out on its own:

```
java -jar VirtualNetwork.jar -v r1 -a arp_cache
```

You may use the same method you tested with your implementation in Assignment 2 Part 3.

To test whether your RIP implementation can safely fail-over in a redundant topology (i.e. with loop), `triangle_rt.topo` is a good starting point. This topology has three routers connected in a ring. You might want to bring down one of the links (say `r1-r2`) by running the `link r1 r2 down` in the mininet terminal, then verify that h1 can still reach h2 even if link r1-r2 has failed. Feel free to create other topologies to test your implementation.

Submission Instructions

You must submit a single gzipped tar of your `src` directory containing the Java source files for your virtual switch and router. Please submit the entire `src` directory; do not submit any other files or directories. To create the tar file, run the following command, replacing `username1` and `username2` with the CS username of each group member:

```
tar -czvf username1_username2.tgz src
```

Upload the tar file to the Assignment 3 dropbox on canvas. Please submit only one tar file per group.