

Final Project

Implement Routing Information Protocol on the *Simple Router*

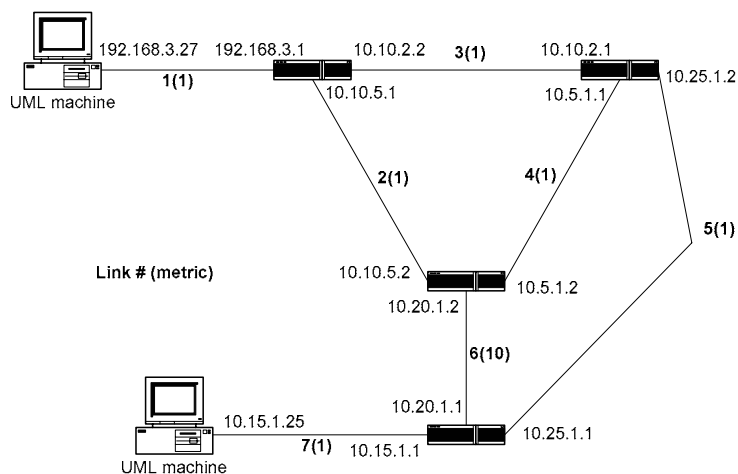
In this project, you are going to take the *simple router* designed and implemented as part of programming assignment 2 and implement a routing information protocol on it. You are encouraged to use the augmented simple router of programming assignment 3, if you have that version fully implemented and debugged.

For the routing information protocol, please refer to RFC 2453 (RIP version 2). This should be considered as the authoritative source for the protocol. You are requested to implement the split horizon with poisoned reverse and triggered updates. You need not be concerned about the following aspects that are covered in the above document:

- compatibility between RIPv1 and RIPv2 (just consider everything to be RIPv2 enabled)
- no authentication of messages
- no routing tag (routing tag as defined in RFC 2453)

Your simple router is supposed to read a configuration file to obtain various different parameters including initial routes (these are the routes that are present to the devices that are directly connected to the router). In addition, the configuration file can include information about the “infinity” value, disable/enable RIP, and any other parameter you find necessary to configure a router. You are encouraged to follow the style of the Zebra (www.zebra.org) configuration files (Zebra is an open-source routing daemon for Linux and BSD derived Unix).

Use the following topology to test the router. Any two devices in the figure below are connected through a *hub* (UML virtual switch running in hub mode) (for example, although not shown, the UML machine is connected to the router through a hub). The numbers beside each link gives the link number and the metric that corresponds to the link.



The metric can be considered as the cost of using the link. The objective of routing protocol is to determine the minimum cost path for sending packets from one point to another.

For your convenience, the project is decomposed into the following sub modules.

1. Modify the UML virtual switch (hub) code such that it will hold the metric and status of the segment within it. That is, when a router connects to the hub, it can probe the hub to know the metric and also the link status (i.e., whether a link is active or inactive). Ideally, I would like you to augment the VPL library to add this feature. You might have to add some code into the hub (UML virtual switch) module. If you find this a very *hard* task, I will (reluctantly) accept alternate solutions that use some other daemons.
2. Implement RIPv2 without split horizon and poisoned reverse. Deploy a virtual network that implements the given topology with the given parameters for link metrics. Use traceroute to show the path taken by a packet from one UML to other through the network.
3. Change the metrics (I assume you are able to change the metrics without rebooting the hubs) at the hubs such that the link costs give a different minimum cost path. Illustrate that your routing protocol works correctly by showing that the new path is taken by packets from one UML to other. The traceroute should be used to again show the new path.
4. Set your update timer (interval between two consecutive RIP update messages) coarsely (about 30 seconds) so that you can observe or measure the impact of route instabilities caused by RIPv2 convergence issues. Fail link 5 in the above topology. This should mean packets cannot cross that link because the link is unavailable and also the cost is infinite. Due to the counting to infinity issue, packets will not reach from one UML to other. Demonstrate this in using your routers and the above topology.
5. Implement triggered updates that expedite the convergence process in the face of the counting to infinity phenomenon. Demonstrate that the convergence of the routes takes place much quickly for the same infinity value (choose a value of 16 as infinity).
6. Implement split horizon and poisoned reverse in your router. With these functions, demonstrate that routing instability does not take place in the router.

You may build a tracing/logging functionality into the router to show routing fluctuations if you find it difficult to demonstrate using real packet traversal scenarios.

Finally, as part of this project, you are required to submit a 5 to 10 page design document for the simple router. This document should cover all aspects of the simple router design starting from programming assignment 2 to the final project. If you are attempting the extra-credit part, you are encouraged to include the discussion of that in this document as well.

Tentative grading scheme:

Design document [20 points]: Content selection, organization, language, presentation will be taken into consideration when this document is marked. Do not copy material and figures from the handouts. You are expected to provide your own interpretation of how the simple router works, why it works. The conclusions and future work section (1 page) should talk about future enhancements you propose to the simple router. Your discussion should also provide ideas on how these enhancements can be designed and implemented. It is important to convey your ideas very clearly in this document. It is your responsibility to write the document such that the grader does not have problems understanding the document. If the grader does not understand what is written, you are very likely to lose points!

Demonstration [20 points]: Questions will be asked during the demonstration. You are expected to be able to answer those questions. These points are dependent on your ability to complete the demonstration (group activity), answer questions (individual activity), and overall presentation (group activity).

Code organization and in-code documentation [10 points]: Modular organization of code. Makefiles, man pages, in-code documentation. *3 points allocated exclusively to those that present a nroff formatted man page (use MAN macros).*

Correctness and completeness of code [50 points]: This is based on the extent your program meets the requirements laid out in this document. Correct and working implementation is essential. If your solution is not working or partially working, you will be graded for partial credit, which does not exceed 25 points.