COSC 364
RIP Assignment Report

29 April 2025

Jordan Chubb 62646119
Wenlong Zong 42718910

Percentage contribution

- Jordan Chubb - 60%
- Wenlong Zong - 40%

List of contributions

Jordan Chubb

- Argument parsing
- Configuration file processing
- Socket setup
- RipManager and RoutingTableEntry classes
- Automatic testing

Wenlong Zong

- Timers
- Packet building
- Manual testing
- Dijkstra's algorithm (automatic testing)

Test discussion

Configuration Files

Correctness of configuration files is checked thoroughly in a specific module. Functions within this module are tested using the doctest module. This module can also test the validity of multiple configuration files at once, ensuring that they are all compatible with each other. The test checks that router-id's are unique and are between 1 and 64000, all port numbers are between 1024 and 64000, and metrics are between 1 and 16. Each output of every configuration file is checked that it matches up to exactly one other configuration file, and that the metrics between them are the same. For automatic testing, all the automatically generated configuration files are checked to be valid. The expected outcome of invalid configuration files is that the program will fail to launch and that an appropriate error message will be displayed, which is what was found during testing.

Manual Testing

Configuration files were generated for the example network in the assignment specification. It was expected that the routing tables would converge to the exact topology in the provided example. The outcome was that they converged correctly.
Different routers were then stopped with the expectation that routers who had the stopped-router as a next-hop would then timeout this route and set the metric to 16. This metric of 16 would then propagate to the other routes, and all routers who had the metric at 16 would start the deletion process before deleting the route from their table. The outcome was that this was exactly the case.

Automatic Testing

The automatic testing program works by launching multiple router daemons as individual processes which are then monitored. The network topology is randomly generated by the program, configuration files are made and saved based on this, and then the daemons are started. The

program monitors the daemon processes through their stdout, which uses a json-encoded list to communicate their entire routing table. For each router daemon, a minimum spanning tree is generated based on the previously generated network topology using Dijkstra's algorithm. The minimum spanning tree is compared against the current routers routing table to ensure that the routing table costs are equal to the minimum costs from Dijkstra's algorithm.
The automatic tests include:

- 100 routers, fully-connected (each router is connected to every other router)
- 100 routers, sparsely-connected (each router connects to one other router)

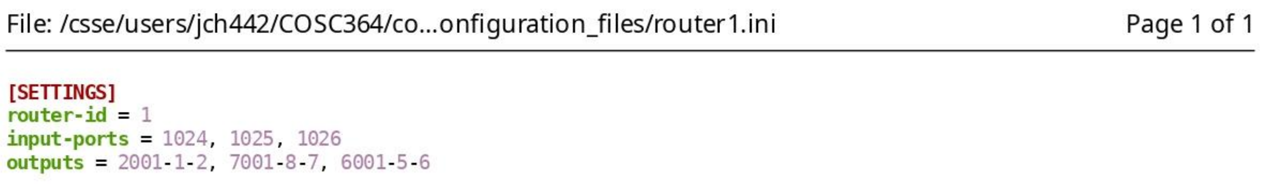
For each of these tests, the program waits until the network has converged correctly, and then once it has, it changes the topology by stopping (crashing) or starting 50 random routers, and then waits for convergence again. It repeats this stopping/starting step several times, waiting for convergence after each change. For these tests we expect every router to eventually converge. Our test results found that the network would converge every time for all tests. This showed that our daemon was correctly converging after topology changes, even for very large networks with groups of routers being disconnected from other groups of routers.

During the convergence process of automatic testing it was discovered that convergence would take over 1 minute after stopping/starting random routers and that the CPU usage was 100% for a 30 second period. It was considered that this may have been caused by the routers waiting for their routes to timeout before updating them. In section 3.9.2 of RIP RFC specification, there is an optional heuristic that switches to a new route if the metric is the same and it is at least halfway to expiry. This is a potential method to improve the long convergence time. This optional heuristic was implemented, but it did not improve the problem.

The problem was then considered to be caused by a large number of triggered updates being sent by routers, which in turn caused other routers to also send triggered updates, making the problem worse. Section 3.10.1 of RIP RFC specification was then implemented, and the outcome was that the CPU usage would no longer go to 100% during the convergence process. This showed that the triggered updates were the source of this problem.

Packet Testing
Malformed packets are randomly generated to check that all packets fit the message format. The RIP packets are expected with command number 2, version number 2, must be zero section for 2 bytes and followed with payload of 1 to 25 RIP entries. Each RIP entry should have address family identifier AF_INET(2), must be zero section for 2 bytes, destination IPv4 address, 2 must be zero sections and the current metric for the destination. All malformed packets are expected to be dropped and that is achieved by the program.

Example configuration file



File: /csse/users/jch442/COSC364/co...onfiguration_files/router1.ini   Page 1 of 1

```
[SETTINGS]
router-id = 1
input-ports = 1024, 1025, 1026
outputs = 2001-1-2, 7001-8-7, 6001-5-6
```

Source code