2021

COSC 364: Asg1

RIP ROUTING PROTOCOL IMPLEMETATION

Bach Vu (25082165)

Charlie Hunter (27380476)

Project Contribution:

Bach Vu (25082165) (60%)

- Creation of Router Class

- Creating of socket, Sending and Receiving

- Creation of Timer.py/Garbage collection

- Printing routing table

- Implement: Triggered update (shorter route/no-response), split horizon

Charlie Hunter (27380476) (40%)

- Reading Configuration files, raising errors if found in files

- Creating and processing packets,

- Updating routing table for new and shorter routes

- Documentation

- Tests: Basic Functionality, Trigged update and garbage collection, Larger Network, Demo Network tests

Questions

Which aspects of your overall program (design or implementation) do you consider particularly well done?

The *daemon.py*, *daemon_sup.py*, *router.py* and *timer.py* all keep functionality of the code separate so debugging and reading code is easier. The ***daemon.py*** file is the main file where the code runs, while ***daemon_sup.py*** has functions to assist the daemon file (reading config, creating/processing packets, etc). The ***router.py*** file has the Router Class where updates to the routing table, checking for expired routes and printing of the routing table and keeping all information about each route (Outputs, inputs, and timeouts) occurs. The ***timer.py*** file checks for regular update and garbage timeouts, also where times for each router is stored.

Which aspects of your overall program (design or implementation) could be improved?

We did not create automated test, so for each minor change in source code we must manually monitor how routing table get updated on each router. The program is still linear coding, even though counter value is kept independent for each task there may be small delay before the task executed: function call, timestamp checking, ...

How have you ensured atomicity of event processing?

The best we could find to do to ensure some level of atomicity was to create a short timeout on the select, and a guard statement for each task to check if it has waited from last execution (task timeout). This was the best option but is not concurrency, just a way to minimize computer resources while waiting for an input to be sent to the socket. This can be seen in the *Daemon.py* in the receive function on line 53, where timeout is set to 0.025 to block the current cycle if no packet arrives. If we increase thread block timeout, less CPU resource consume as less amount of repeated task being call, so the execution time can be offset by a larger value.

Have you identified any weaknesses of the RIP routing protocol?

Two weakness of the RIP routing protocol are the maximum hop count of 15 and lots of network traffic when update is multicast. The RIP protocol can only be used in a small network where each router can reach maximum 15 hops away. This means in large networks where hops between routers are more than 15 hops away the RIP protocol is not able to be used. Another problem is the high level of network traffic due to updates being sent between routers periodically, or when there is a major change in the network layout.

RIP operate on application layers, so its rely on lower level to send/receive accurate datagram. Even so, a hardware problem may change the routing table memory even if the daemon work perfectly. In practical, a router need 180s to detect a deadlink, so it may have discarded many packets that it supposed to forward to the lost-connection destination (wrong advertise).

Testing

Three tests networks where made for testing each with a specific functionality to determine the code ran correctly. Test one, was a basic functionally test on a small network, test two was

a triggered update and garbage collection test, and test three was to test convergence on a large network. Trivial test where also carried out on the demo network provided to us.

Test 1 – Basic Functionality test.

The first test conducted can be seen on the network below in *figure 1.* This test was to test the basic functionality of code. Firstly router one and router two where ran and connected to one another. After this router three was ran, and router one and two both found a shorter path to each other via router three. The process of router one can be seen below in *figures 2 to 5*.

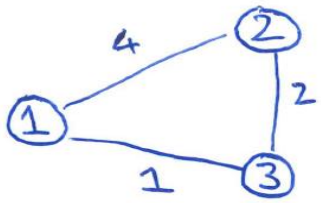


Figure 1 – Basic functionality test network

```
====================================================================
|                --ROUTING TABLE [19:21:18.810]--                  |
|  Dest.   | Next Hop |  Metric  | Time (s) |       Notes         |
|------------------------------------------------------------------|
|    1     |    -     |    0     |  0.001   |  ['Time Active']    |
====================================================================
```

Figure 2 – router one starting

```
====================================================================
|                --ROUTING TABLE [19:21:38.966]--                  |
|  Dest.   | Next Hop |  Metric  | Time (s) |       Notes         |
|------------------------------------------------------------------|
|    1     |    -     |    0     |  20.157  |  ['Time Active']    |
|    2     |    2     |    4     |  1.136   |  ['Reset timer']    |
====================================================================
```

Figure 3 – router one converging to router two

```
====================================================================
|                --ROUTING TABLE [19:21:49.049]--                  |
|  Dest.   | Next Hop |  Metric  | Time (s) |       Notes         |
|------------------------------------------------------------------|
|    1     |    -     |    0     |  30.240  |  ['Time Active']    |
|    2     |    2     |    4     |  11.218  |  ['Reset timer']    |
|    3     |    3     |    1     |  0.834   |   ['New dest.']     |
====================================================================
```

Figure 4 – Router three turning on and converging to the network

```
                --ROUTING TABLE [19:22:09.255]--
 Dest.   | Next Hop |  Metric  | Time (s) |       Notes
|-------------------------------------------------------|
   1     |    -     |    0     |  50.445  |  ['Time Active']
   2     |    3     |    3     |  3.920   | ['Shorter route']
   3     |    3     |    1     |  3.920   |  ['Reset timer']
```

Figure 5 – Router one connecting to router two via router three for a smaller cost

Test 2 – Trigged Update and Garbage Collection

The second test conducted can be seen below on *figure 6*. The purpose of this test was to test Trigged update and garbage collection. Firstly all routers where ran so that they could all converge. Once convergence has occurred router three was turned off. Routers one and two both receive a timeout from router three they understand that router three is dead and stop advertising that router three and four and set both the metric to router three and four to 16. After a garbage timeout both routers one and two will remove routers three and four from there routing table. Router ones process of this can be seen below in *figures 7 to 10*. The outcome worked as excepted.

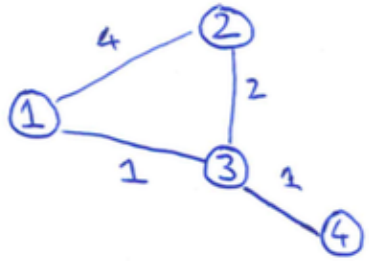
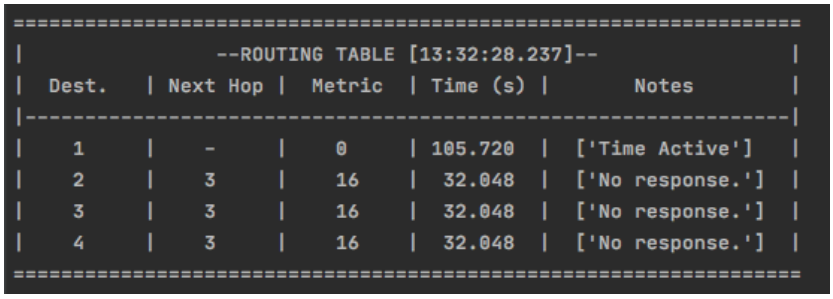


Figure 6 – Split horizon with poisoned reverse test network

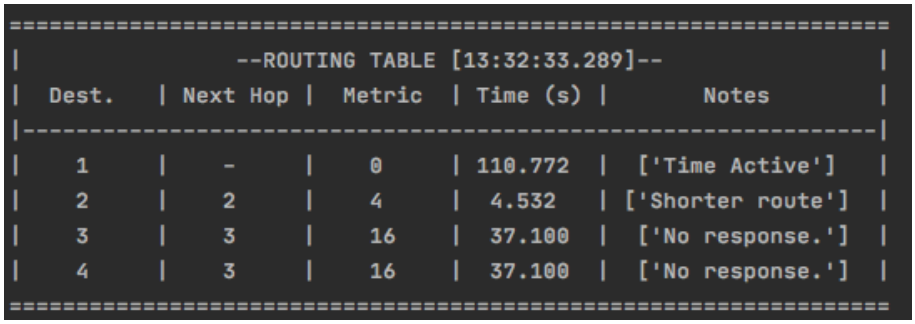
```
                --ROUTING TABLE [13:31:17.822]--
 Dest.   | Next Hop |  Metric  | Time (s) |       Notes
|-------------------------------------------------------|
   1     |    -     |    0     |  35.305  |  ['Time Active']
   2     |    3     |    3     |  5.012   |  ['Reset timer']
   3     |    3     |    1     |  5.012   |  ['Reset timer']
   4     |    3     |    2     |  5.012   |  ['Reset timer']
```

Figure 7 – Routing table of route one after convergence of all routers



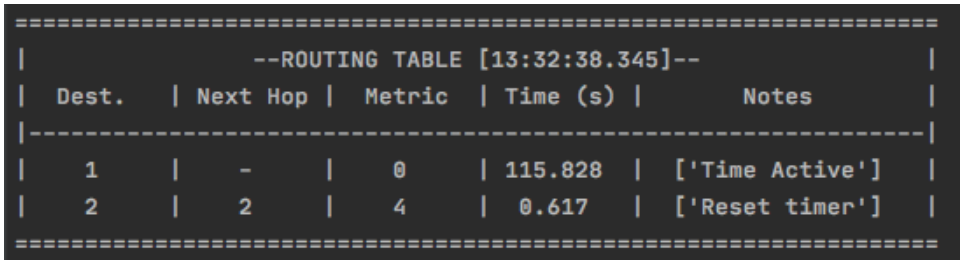
--ROUTING TABLE [13:32:28.237]--				
Dest.	Next Hop	Metric	Time (s)	Notes
1	-	0	105.720	['Time Active']
2	3	16	32.048	['No response.']
3	3	16	32.048	['No response.']
4	3	16	32.048	['No response.']

Figure 8 – routing table of route one after route three is turned off, and 30 seconds of no response from router three.



--ROUTING TABLE [13:32:33.289]--				
Dest.	Next Hop	Metric	Time (s)	Notes
1	-	0	110.772	['Time Active']
2	2	4	4.532	['Shorter route']
3	3	16	37.100	['No response.']
4	3	16	37.100	['No response.']

Figure 9 – routing table of route one re-routing to route 2



--ROUTING TABLE [13:32:38.345]--				
Dest.	Next Hop	Metric	Time (s)	Notes
1	-	0	115.828	['Time Active']
2	2	4	0.617	['Reset timer']

Figure 10 – routing table of route one, after garbage time where both route four and route three are removed from the routing table

Test 3 – A larger network test with five routers.

The third test was to test how the network would go with a larger network of 5 routers. This test was to make sure in a larger network different from the provided demo network the code would still run correctly. The network can be seen below in *figure 11*. The correct routing tables after convergence can also be seen for each router in *figures 12-16*.

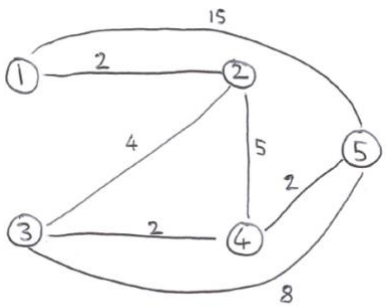


Figure 11 - five routers test network

```
================================================================
|                --ROUTING TABLE [13:19:54.330]--              |
|  Dest.   | Next Hop |  Metric  | Time (s) |      Notes       |
|--------------------------------------------------------------|
|    1     |    -     |    0     | 166.470  |  ['Time Active']  |
|    2     |    2     |    2     |  1.130   |  ['Reset timer']  |
|    5     |    2     |    9     |  1.130   |  ['Reset timer']  |
|    3     |    2     |    6     |  1.130   |  ['Reset timer']  |
|    4     |    2     |    7     |  1.130   |  ['Reset timer']  |
================================================================
```

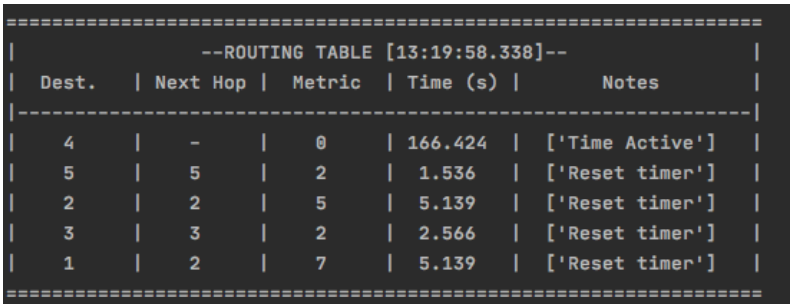
Figure 12 – Routing table for route one

```
================================================================
|                --ROUTING TABLE [13:19:56.084]--              |
|  Dest.   | Next Hop |  Metric  | Time (s) |      Notes       |
|--------------------------------------------------------------|
|    2     |    -     |    0     | 166.448  |  ['Time Active']  |
|    3     |    3     |    4     |  0.312   |  ['Reset timer']  |
|    4     |    4     |    5     |  4.846   |  ['Reset timer']  |
|    1     |    1     |    2     |  4.741   |  ['Reset timer']  |
|    5     |    4     |    7     |  4.846   |  ['Reset timer']  |
================================================================
```

Figure 13 – Routing table for route two

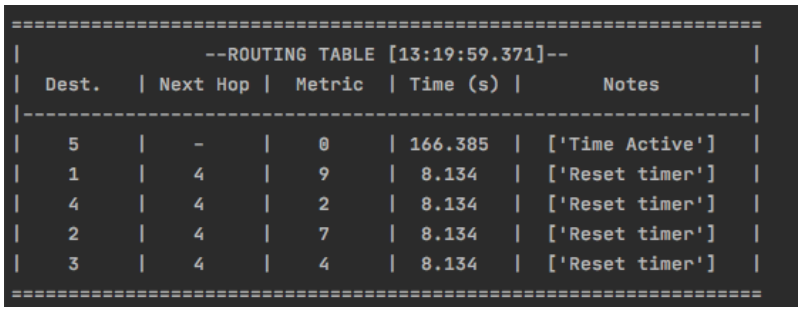
```
================================================================
|                --ROUTING TABLE [13:19:57.111]--              |
|  Dest.   | Next Hop |  Metric  | Time (s) |      Notes       |
|--------------------------------------------------------------|
|    3     |    -     |    0     | 166.334  |  ['Time Active']  |
|    4     |    4     |    2     |  5.873   |  ['Reset timer']  |
|    2     |    2     |    4     |  3.911   |  ['Reset timer']  |
|    1     |    2     |    6     |  3.911   |  ['Reset timer']  |
|    5     |    4     |    4     |  5.873   |  ['Reset timer']  |
================================================================
```

Figure 14 – Routing table for route three



```
================================================================
|               --ROUTING TABLE [13:19:58.338]--               |
|  Dest.   | Next Hop |  Metric  | Time (s) |       Notes       |
|--------------------------------------------------------------|
|    4     |    -     |    0     | 166.424  |  ['Time Active']  |
|    5     |    5     |    2     |  1.536   |  ['Reset timer']  |
|    2     |    2     |    5     |  5.139   |  ['Reset timer']  |
|    3     |    3     |    2     |  2.566   |  ['Reset timer']  |
|    1     |    2     |    7     |  5.139   |  ['Reset timer']  |
================================================================
```

Figure 15 – Routing table for route four



```
================================================================
|               --ROUTING TABLE [13:19:59.371]--               |
|  Dest.   | Next Hop |  Metric  | Time (s) |       Notes       |
|--------------------------------------------------------------|
|    5     |    -     |    0     | 166.385  |  ['Time Active']  |
|    1     |    4     |    9     |  8.134   |  ['Reset timer']  |
|    4     |    4     |    2     |  8.134   |  ['Reset timer']  |
|    2     |    4     |    7     |  8.134   |  ['Reset timer']  |
|    3     |    4     |    4     |  8.134   |  ['Reset timer']  |
================================================================
```

Figure 16 – Routing table for route five

Example configuration file

Below an example configuration file for a router can be seen, the example shows router one.

```
router-id: 1
input-ports: 51002, 51003, 51004, 51005, 51006, 51007
outputs: 51007-57001-8-7, 51006-56001-5-6, 51002-52001-1-2
timer: 5
```

Input-ports is identified by 5X00Y, where X is equal to the router Id of the sending router and Y is the router Id of the receiving router.

Outputs is defined by X-Y-Z-A, where X is the input port being sent from, Y is the Output Port being sent to, Z is the metric value, and A is the router Id being sent to (who owns port Y).

Timer is optional field, indicating regular update period (in sec). Other timeout values are scale based on this value: print period, garbage & detect deadlink task, ... For demo purpose, timer is set to 5.

Source code