

COSC 364

Internet Technologies and Engineering

First Assignment

Jemin Lee 7548762

Xiang Ji 22246256

The percentage contribution:

Jemin Lee 60%

Xiang Ji 40%

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

When we design our program, we refer to the object-oriented design ideas, such as it creates a router for every router. we separate create message , send message and receive message to three functions. In this way, code will be easy to read, modify and test for everyone in the group. Another aspect of the program we consider well down is we use different strings(update, trigger) to distinguish the update is trigger or periodic. This made the program more fast and effective when the program ran.

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

Due to this we know the importance of the split horizon with poisoned reverse in route loops, so we took this method. When using this method, the information received by the router is used to update the routing table and then sent out through all interfaces. However, when a routing table item that has already arrived from an interface is sent through the same interface, its metric is set to 16. Maybe this method isn't the best approach, but it will be able to run as expected.

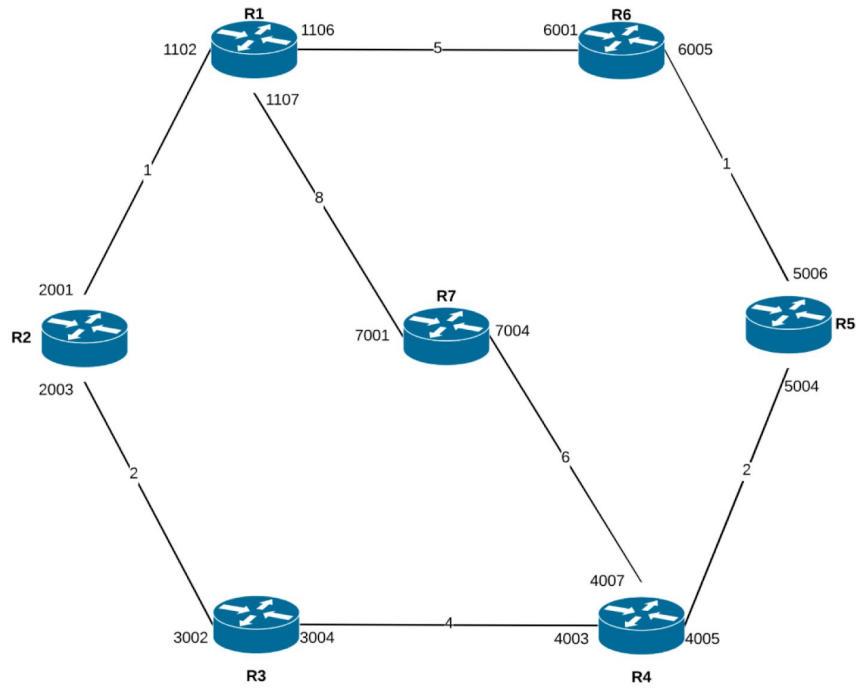
How have you ensured atomicity of event processing?

We have a bind_socket function for each port. In this way it guarantees every process is respective.

Our testing plans :

In fact, we did a lot of small tests when we wrote the program. We would add print statements to display the process and compare them with the expected results to determine whether the program is running as expected.

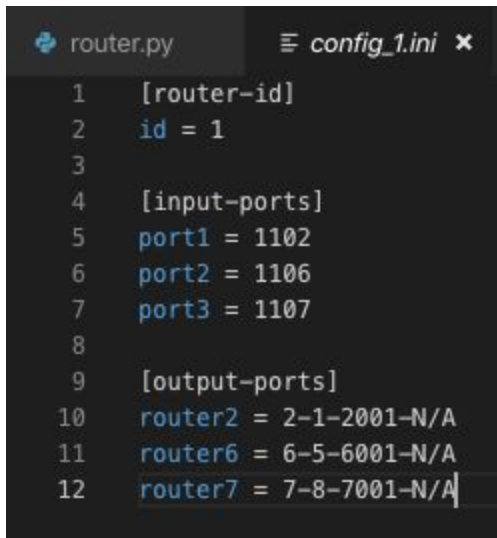
network for testing :



Startup configuration

When we are ready, we start to check whether the program can read all configures of the router. For this test, we start one router and check if the program properly reads the startup configuration.

The format of the configuration file looks like below (configuration file 1):



```
router.py  config_1.ini x
1  [router-id]
2  id = 1
3
4  [input-ports]
5  port1 = 1102
6  port2 = 1106
7  port3 = 1107
8
9  [output-ports]
10 router2 = 2-1-2001-N/A
11 router6 = 6-5-6001-N/A
12 router7 = 7-8-7001-N/A
```

router 1 startup configuration example

It is an ini type file, and consists of router-ID number, its port numbers that router listens to, and routes for its direct neighbors.

Routes consist of destination router name as key, and destination router_ID - cost - port number - next hop as value.

Therefore, when router-1 is started, the expected outcome is, the terminal displaying router 2, 6 and 7 as directly connected.

```
[Jemins-MBP:COSEC364_RIP-Routing-Project jeminlee$ python3 router.py config_1.ini ]
RIP Router Demon
RIP Version: 2
!
!
!
Show Routes
Router ID: 1, 13:34:06.503439
router 2 directly connected, 2001
router 6 directly connected, 6001
router 7 directly connected, 7001
!
!
13:34:06.503522, Update message sent to port 2001
13:34:06.503522, Update message sent to port 6001
13:34:06.503522, Update message sent to port 7001
```

router 1- displaying routes after startup

As expected, the terminal displays the expected outcome. Therefore the program is reading the startup configuration correctly. Additionally, we can see that the router sends update messages to its neighbors initially. We expect this to repeat periodically.

Send Periodic Update

The next step we check if routers perform periodic updates. We start only two routers, let two routers learn new routes. The routers send messages every 12 seconds. Router 1 and 2 will be started, we will check if routers send messages every 12 seconds and learn routes that are valid at this point. Expected outcomes are:

- Each router sending periodic update
- Router 1 learning a route to router 3
- Router 2 learning routes to router 6 and 7

```
COSC364_RIP-Routing-Project -- bash -- 80x53
Jemins-MBP:COSC364_RIP-Routing-Project jeminlee$ python3 router.py config_1.ini
RIP Router Demon
RIP Version: 2
!
!
!
Show Routes
Router ID: 1, 15:27:14.586370
router 2 directly connected, 2001
router 6 directly connected, 6001
router 7 directly connected, 7001
!
15:27:14.586457, Update message sent to port 2001
15:27:14.586457, Update message sent to port 6001
15:27:14.586457, Update message sent to port 7001
!
15:27:14.587655, message from router 2
!
updating route, message from router2
!
3 reachable via Port 2001 , Next Hop: 2, Metric: 4
!
Invalid timer started for router3
!
!
Show Routes
Router ID: 1, 15:27:15.277782
router 6 directly connected, 6001
router 7 directly connected, 7001
router 2 directly connected, 2001
router 3 reachable via Port 2001, Next Hop: 2 Metric: 4
!
15:27:27.278375, Update message sent to port 2001
15:27:27.278375, Update message sent to port 6001
15:27:27.278375, Update message sent to port 7001
!
15:27:27.279910, message from router 2
!
!
updating route, message from router2
!
!
!
Show Routes
Router ID: 1, 15:27:39.285097
router 6 directly connected, 6001
router 7 directly connected, 7001
router 3 reachable via Port 2001, Next Hop: 2 Metric: 4
router 2 directly connected, 2001

COSC364_RIP-Routing-Project -- bash -- 81x53
Jemins-MBP:COSC364_RIP-Routing-Project jeminlee$
Jemins-MBP:COSC364_RIP-Routing-Project jeminlee$
Jemins-MBP:COSC364_RIP-Routing-Project jeminlee$ python3 router.py config_2.ini
RIP Router Demon
RIP Version: 2
!
!
!
Show Routes
Router ID: 2, 15:27:15.276943
router 1 directly connected, 1102
router 3 directly connected, 3002
!
15:27:15.277045, Update message sent to port 1102
15:27:15.277045, Update message sent to port 3002
!
!
Show Routes
Router ID: 2, 15:27:27.278286
router 1 directly connected, 1102
router 3 directly connected, 3002
!
15:27:39.282953, Update message sent to port 1102
15:27:39.282953, Update message sent to port 3002
!
15:27:39.284253, message from router 1
!
!
updating route, message from router1
!
6 reachable via Port 1102 , Next Hop: 1, Metric: 7
!
Invalid timer started for router6
!
updating route, message from router1
!
7 reachable via Port 1102 , Next Hop: 1, Metric: 10
!
Invalid timer started for router7
!
!
!
Show Routes
Router ID: 2, 15:27:39.286460
router 3 directly connected, 3002
router 1 directly connected, 1102
router 6 reachable via Port 1102, Next Hop: 1 Metric: 7
router 7 reachable via Port 1102, Next Hop: 1 Metric: 10
```

execution of router 1 and 2

Routers are expected to send its routing update immediately after startup, so router 2 does not get an update from router 1 initially because router 1 was started before router 2. Nevertheless, we can see that routers send periodic updates via the attached ports(in the red box), and each router gets messages, and updates their own routing table (in the green box). Therefore, we can conclude that the program can perform periodic updates and routing updates.

Invalidation & Flush timer

Now we turn off one router from the same condition. We can see if the other router invalidates, and flush the aged routes. The timers set for invalidate timer and flush timer are 45 and 60 seconds in this test. The expected outcome is one router will age out all routes eventually.

The screenshot below indicates that a message was received from router 2.

```
Show Routes
Router ID: 1, 17:54:44.746606
router 2 directly connected, 2001
!
!
17:54:56.751886, Update message sent to port 2001
17:54:56.751886, Update message sent to port 6001
17:54:56.751886, Update message sent to port 7001
!
17:54:56.753485, message from router 2
```

last message received at 5:54:57

After approximately 45 seconds, the route is invalidated, and displayed as -

“router 2 possibly down”.

```
17:55:41.756783, router2 invalid
!
!
!
Show Routes
Router ID: 1, 17:55:44.763938
route to router 2 possibly down
```

Invalid route (at 5:55:42)

Eventually, after 60 seconds, the route is deleted from the router's routing table.

```
.
Show Routes
Router ID: 1, 17:56:32.777177
route to router 2 possibly down
17:56:41.760067 route to router2 deleted
```

flush route (at 5:56:33)

Route Convergence & Triggered Update

The next test is to remove one router from the full topology. Through this test, if the routers can communicate with each other, the remaining routers can converge and create a new link table, and routers implement routing protocol RIPv2

After startup, we wait for the routers to update their routing tables.

```
!
Show Routes
Router ID: 1, 18:20:52.919537
rotuer 3 reachable via Port 2001, Next Hop: 2 Metric: 4
rotuer 5 reachable via Port 2001, Next Hop: 2 Metric: 12
rotuer 4 reachable via Port 2001, Next Hop: 2 Metric: 9
rotuer 6 reachable via Port 2001, Next Hop: 2 Metric: 14
router 2 directly connected, 2001
router 7 directly connected, 7001
```

router 1 routing table

```
!
Show Routes
Router ID: 2, 18:20:05.670589
rotuer 4 reachable via Port 3002, Next Hop: 3 Metric: 7
rotuer 6 reachable via Port 3002, Next Hop: 3 Metric: 12
rotuer 7 reachable via Port 1102, Next Hop: 1 Metric: 10
rotuer 5 reachable via Port 3002, Next Hop: 3 Metric: 10
router 1 directly connected, 1102
router 3 directly connected, 3002
!
```

router 2 routing table

```
!
Show Routes
Router ID: 3, 18:20:06.259511
rotuer 5 reachable via Port 4003, Next Hop: 4 Metric: 7
rotuer 7 reachable via Port 4003, Next Hop: 4 Metric: 11
rotuer 1 reachable via Port 2003, Next Hop: 2 Metric: 4
rotuer 6 reachable via Port 4003, Next Hop: 4 Metric: 9
router 2 directly connected, 2003
router 4 directly connected, 4003
!
```

router 3 routing table

```
!
Show Routes
Router ID: 4, 18:19:42.834729
rotuer 6 reachable via Port 5004, Next Hop: 5 Metric: 4
rotuer 2 reachable via Port 3004, Next Hop: 3 Metric: 7
rotuer 1 reachable via Port 3004, Next Hop: 3 Metric: 9
router 3 directly connected, 3004
router 5 directly connected, 5004
router 7 directly connected, 7004
!
```

router 4 routing table


```

Show Routes
Router ID: 5, 18:18:31.259331
rotuer 1 reachable via Port 6005, Next Hop: 6 Metric: 7
rotuer 3 reachable via Port 4005, Next Hop: 4 Metric: 7
rotuer 7 reachable via Port 4005, Next Hop: 4 Metric: 9
rotuer 2 reachable via Port 6005, Next Hop: 6 Metric: 9
router 4 directly connected, 4005
router 6 directly connected, 6005
!

```

router 5 routing table

```

Show Routes
Router ID: 6, 18:18:16.810766
rotuer 7 reachable via Port 5006, Next Hop: 5 Metric: 11
rotuer 2 reachable via Port 1106, Next Hop: 1 Metric: 7
rotuer 3 reachable via Port 5006, Next Hop: 5 Metric: 9
rotuer 4 reachable via Port 5006, Next Hop: 5 Metric: 4
router 1 directly connected, 1106
router 5 directly connected, 5006
!

```

router 6 routing table

```

Show Routes
Router ID: 7, 18:19:52.880100
rotuer 6 reachable via Port 4007, Next Hop: 4 Metric: 11
rotuer 2 reachable via Port 1107, Next Hop: 1 Metric: 10
rotuer 3 reachable via Port 4007, Next Hop: 4 Metric: 11
rotuer 5 reachable via Port 4007, Next Hop: 4 Metric: 9
router 1 directly connected, 1107
router 4 directly connected, 4007
!

```

router 7 routing table

We can see that all routers form a full routing table, so we turned off router 6.

The expected outcome is that the routers age out the route for route 6, and send triggered updates to outgoing ports, and when other routers get them, invalidate the route for 6, and send the triggered update out to their own egress ports except to the ports where the message came from.

We will provide the screenshots of two routers (5 and 4).

```

Show Routes
Router ID: 5, 18:19:31.294002
rotuer 1 reachable via Port 6005, Next Hop: 6 Metric: 7
rotuer 3 reachable via Port 4005, Next Hop: 4 Metric: 7
rotuer 7 reachable via Port 4005, Next Hop: 4 Metric: 9
rotuer 2 reachable via Port 6005, Next Hop: 6 Metric: 9
router 6 directly connected, 6005
router 4 directly connected, 4005
!
RIP trigger message sent to port 4005
!
18:19:40.276393, router6 invalid
!

```

router 5 routing table before invalidation (logs for invalidation timer, and RIP trigger update message sent)

```
Show Routes
Router ID: 5, 18:19:43.301776
rotuer 1 reachable via Port 6005, Next Hop: 6 Metric: 7
rotuer 3 reachable via Port 4005, Next Hop: 4 Metric: 7
rotuer 7 reachable via Port 4005, Next Hop: 4 Metric: 9
rotuer 2 reachable via Port 6005, Next Hop: 6 Metric: 9
route to router 6 possibly down
router 4 directly connected, 4005
!
```

Router 5 routing table after invalidation

We can see that the route to router 5 is invalidated, and displayed as “possibly down”

Now, we check router 4, if it received triggered update from router 5, and invalidates the route.

```
Show Routes
Router ID: 4, 18:19:42.834729
rotuer 6 reachable via Port 5004, Next Hop: 5 Metric: 4
rotuer 2 reachable via Port 3004, Next Hop: 3 Metric: 7
rotuer 1 reachable via Port 3004, Next Hop: 3 Metric: 9
router 3 directly connected, 3004
router 5 directly connected, 5004
router 7 directly connected, 7004
!
!
18:19:54.835681, Update message sent to port 3004
18:19:54.835681, Update message sent to port 5004
18:19:54.835681, Update message sent to port 7004
!
18:19:54.837630, message from router 3
!
!
updating route, message from router3
!
updating route, message from router3
!
recieved trigger update from router 5
[{5: 'trigger'}, ['6', '16', '6005', 'N/A']]
!
```

router 4 routing table, log for trigger update received from router 5

```

.
18:20:18.261838, router6 down according to trigger update from router 4
!
!
!
Show Routes
Router ID: 3, 18:20:18.262548
rotuer 5 reachable via Port 4003, Next Hop: 4 Metric: 7
rotuer 7 reachable via Port 4003, Next Hop: 4 Metric: 11
rotuer 1 reachable via Port 2003, Next Hop: 2 Metric: 4
route to router 6 possibly down
router 2 directly connected, 2003
router 4 directly connected, 4003
.

```

Router 4 fixing its routing table from triggered update

Lastly, we checked if all routers delete the route.

```

!
Show Routes
Router ID: 4, 18:21:18.903119
rotuer 2 reachable via Port 3004, Next Hop: 3 Metric: 7
rotuer 1 reachable via Port 3004, Next Hop: 3 Metric: 9
router 3 directly connected, 3004
router 5 directly connected, 5004
router 7 directly connected, 7004
18:21:27.834888 route to router6 deleted
!

```

Router 4 deletes the route to router 6

We can see that router 4 has been deleted, so the test is successful and it tells us that routers can dynamically update their routing tables and send triggered updates.

Infinite Metric, 16

In `show_routes` function and `create_message` function we can see that 16 is the infinite metric by printing route id down when the receive metric is 16, and in `update_table`, when the cost of a route is larger than 15, the route is ignored.

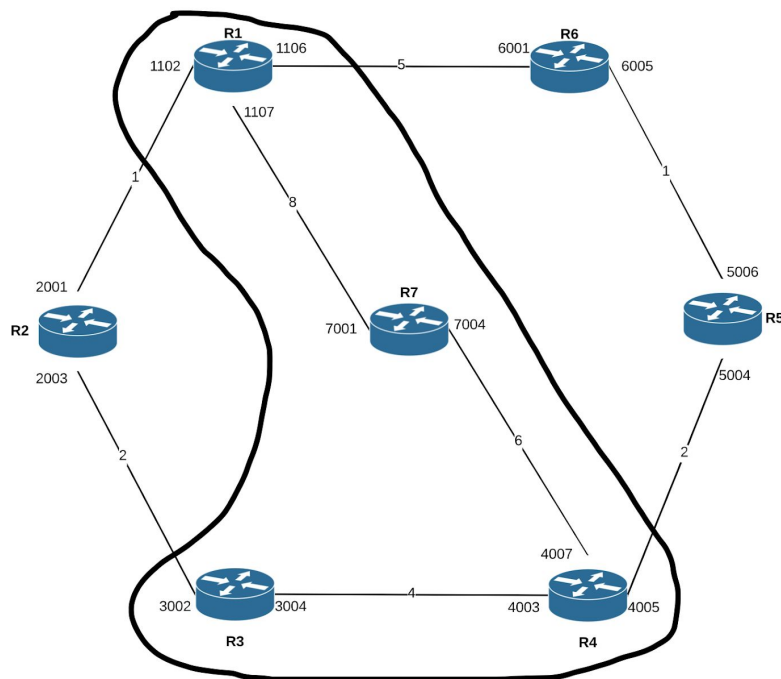
In `show_routes()`

```
if metric == "16":  
    print('route to router {} possibly down'.format(ID))
```

In `update_table()`

```
if potential_new_cost > 15:  
    print("!")
```

For this test, we will only start router 1, 3, 4, 7. This is to test if routers save routes that cost more than 16. For example, for router 1 to reach router 3 through router 7 in this environment, the cost is 20, meaning unreachable.



```
!
Show Routes
Router ID: 1, 19:02:07.504711
rotuer 4 reachable via Port 7001, Next Hop: 7 Metric: 15
router 7 directly connected, 7001
!
```

Router 1 routing table

```
Show Routes
Router ID: 3, 19:02:07.506176
route to router 5 possibly down
rotuer 7 reachable via Port 4003, Next Hop: 4 Metric: 11
router 4 directly connected, 4003
```

Router 3 routing table

As we can see, two routers cannot reach each other, so we can conclude that the max metric of 16 threshold is working for all routers.

Split-Horizon

We show how RIP routers send the whole routing table to their neighbours by printing it. Then we have to find out how messages are created. In this way, the program implements split-horizon to avoid sending routes to the same port where the routes are learned from the same port.

To test this, we print the message content when routers send messages to other routers. To show this, we will provide a screenshot of router 1.

```
Show Routes
Router ID: 1, 19:08:04.833783
router 2 directly connected, 2001
router 6 directly connected, 6001
router 7 directly connected, 7001
!
!
[{1: 'update'}, {'router6': ['6', '5', '6001', 'N/A'], 'router7': ['7', '8', '7001', 'N/A']}]
19:08:04.833879, Update message sent to port 2001
[{1: 'update'}, {'router2': ['2', '1', '2001', 'N/A'], 'router7': ['7', '8', '7001', 'N/A']}]
19:08:04.833879, Update message sent to port 6001
[{1: 'update'}, {'router2': ['2', '1', '2001', 'N/A'], 'router6': ['6', '5', '6001', 'N/A']}]
19:08:04.833879, Update message sent to port 7001
,
```

Router 1 update messages to its egress ports

Currently, router 1's routing table indicates that it is directly connected to router 2, 6, and 7. The update message to them should be different, as they are shown in the screenshot above.

To router 2 (port 2001):

- Router 6 route (port 6001)
- Router 7 route (port 7001)

To router 6 (port 6001):

- Router 2 route (port 2001)
- Router 7 route (port 7001)

To router 7 (port 7001):

- Router 2 route (port 2001)
- Router 6 route (port 6001)