COSC364 22-S1 Assignment Report

Name: MENG ZHANG
Student ID: 71682325

Nameng: ZHENG CHAO
Student ID: 21671773

Questions:

1. **A percentage contribution of each partner to the overall project**

   MENG ZHANG (71682325): 60%
   ZHENG CHAO (21671773): 40%

2. **For each partner please give a brief list of contributions**

   MENG ZHANG (71682325):
   - Program structure design (coworking)
   - Program test design (coworking)
   - Implementation of following modules:
     - rip_main.py
     - rip_init.py
     - rip_router.py
     - IO_parser.py
     - forwarding_route.py
     - rip_packet.py

   ZHENG CHAO (21671773):
   - Program structure design (coworking)
   - Program test design (coworking)
   - Implementation of following modules:
     - IO_formatter.py
     - network_interface.py

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

   Adequate time and efforts have been devoted to the design of the program. Particularly, the program is well organised in the pattern of object oriented programming, with the Router class as the core of the whole program which binds the data of the routing table with its related methods. Incorporated by an Interface object simulating a network interface, a Router object can advertise, maintain, and manipulate its routing table data and implement the rip protocol. With classes like Route and RipPacket, as well as other supported functional modules (i.e. IO_parser.py, IO_formatter.py), the program is maintainable and extensible.

4. **Which aspects of your overall program (design or implementation) could be Improved?**

   Both the design and the implementation of the program could be improved, but with more time the implementation of the program, especially the incoming message processing part of the Router class could be done in a dryer and more systematic way. In addition, the output of the program could be further improved, i.e. only important new messages are updated and printed out, so that the scrolling messages could be more readable.

5. **How have you ensured the atomicity of event processing?**

   The program is single-threaded and runs in an infinite while loop (refer to the program entry point rip_main.py), in which ROUTER.receive_routes(), ROUTER.advertise_all_routes_periodically(), and ROUTER.check_timeout_entries_periodically() are invoked in order. To implement multiple timer-driven events and ensure their atomicity at the same time, each timer-driven event is assigned a unique timestamp variable, i.e. __regular_advertise_timer (refer to rip_router.py), and an event method is only invoked when its elapsed time is equal to or greater than its preset period. e.g. ROUTER.advertise_all_routes_periodically() is invoked every 3 seconds plus a random offset time; if its elapsed time is less than its preset period, the method skips and continues to the following operation.It is worth to mention that in the ROUTER.receive_routes() method, a select.select() function is called to pick up incoming packets from a given number of input sockets and will block the whole process when there is no incoming messages, in order to save CPU consumption. However, it may also delay other timer-driven events. To minimise this side effect, a timeout (i.e. 0.5 sec) is passed to select.select() so that it will block the process at most 0.5 sec in each loop.

6. **Have you identified any weaknesses of the RIP routing protocol?**

- The RIP protocol is limited to small networks whose longest path is 15 hops, due to its infinity (16) configuration. As a result, it's almost impossible to apply RIP protocol to large networks. If a bigger infinity value is chosen, i.e. 1000, to allow a larger network, the process of convergence could become extremely slow, and also violates backwards compatibility.
- The RIP protocol is designed to only use fixed metrics to compare alternative routes. If real-time costs such as network delay, load, reliability ,etc are used, it could cause highly frequent triggered updates, which would result in massive network load.
- RIP protocol lacks the mechanism to divide large networks into smaller subnetworks like OSPF protocol. The size of the message each node processes and advertises would increase proportional to the size of the whole network, which makes the protocol not as scalable as OSPF.
- The backwards compatibility for RIP protocol version 1 could cause security problems, as version 1 lacks basic authentication mechanisms.

Testing

Test 1: basic init functionality of a single router

- Case 1:
  Description: Check if a router with a valid router config file can be started correctly.
  Steps: start rip_main.py with router1_config.txt
  Expected result:
    - A router is created without error messages and runs in an infinite loop.
    - A routing table containing 1 entry (router 1 itself) is printed out periodically.
    - A regular update message is printed out for each of the three neighbours. (router 2/6/7), indicating matching router id, output port and timestamp.
    - Period is randomised and printed out each time a regular update is advertised.
    - A timeout checking message is printed out periodically with its timestamp.

  Status: Passed

```
 cosc364_assignment git:(main) > python3 rip_main.py router1_config.txt
Starts RIP Daemon...
Set Router regular update period to 3.96
Created Router 1
Sends all routes to Router2 [6020] at 00:46:56.62
Sends all routes to Router6 [6061] at 00:46:56.62
Sends all routes to Router7 [6071] at 00:46:56.62

=========================================================================
|                     Router 01 RIP ROUTING TABLE                       |
=========================================================================
|   Dest   |   Next   |   Metric   |  Timeout  |  Garbage  |   State   |
-------------------------------------------------------------------------
|    1     |    -     |     0      |     -     |     -     |  active   |
-------------------------------------------------------------------------

Set Router regular update period to 3.85
Checking timeout entries at 00:46:59.62
Sends all routes to Router2 [6020] at 00:47:00.62
Sends all routes to Router6 [6061] at 00:47:00.62
Sends all routes to Router7 [6071] at 00:47:00.62
```

- Case 2:
  Description: Check if a router with an invalid route config file can be started or not.
  Steps: start rip_main.py with router1_config_false.txt in which the ratio between period and timeout is not 6 as specified in assignment specs.
  Expected result:
    - The router does not start
    - An error message is printed out, indicating the timer value in the config file is invalid.

  Status: passed

```
 cosc364_assignment git:(main) > python3 rip_main.py router1_config_false.txt
Starts RIP Daemon...
The ratio timeout vs period should be 6
Some value of the config file is invalid
Traceback (most recent call last):
  File "/home/mz/Documents/repositories/cosc364/cosc364_assignment/rip_main.py", line 28, in <module>
    ROUTER = rip_router_init(config_file_name)
  File "/home/mz/Documents/repositories/cosc364/cosc364_assignment/rip_init.py", line 23, in rip_router_init
    router = Router(config['router_id'],
TypeError: 'NoneType' object is not subscriptable
```

Conclusion of test 1:

- The tested program can correctly parse a valid config file, start a router daemon and run into an infinite loop, printing out the expected routing table and update messages on time.
- The tested program won't start a router with an invalid config file and can print out an error message relevant to the invalid parts of the false config file.

Test 2: basic routing functionality of a 3-router network

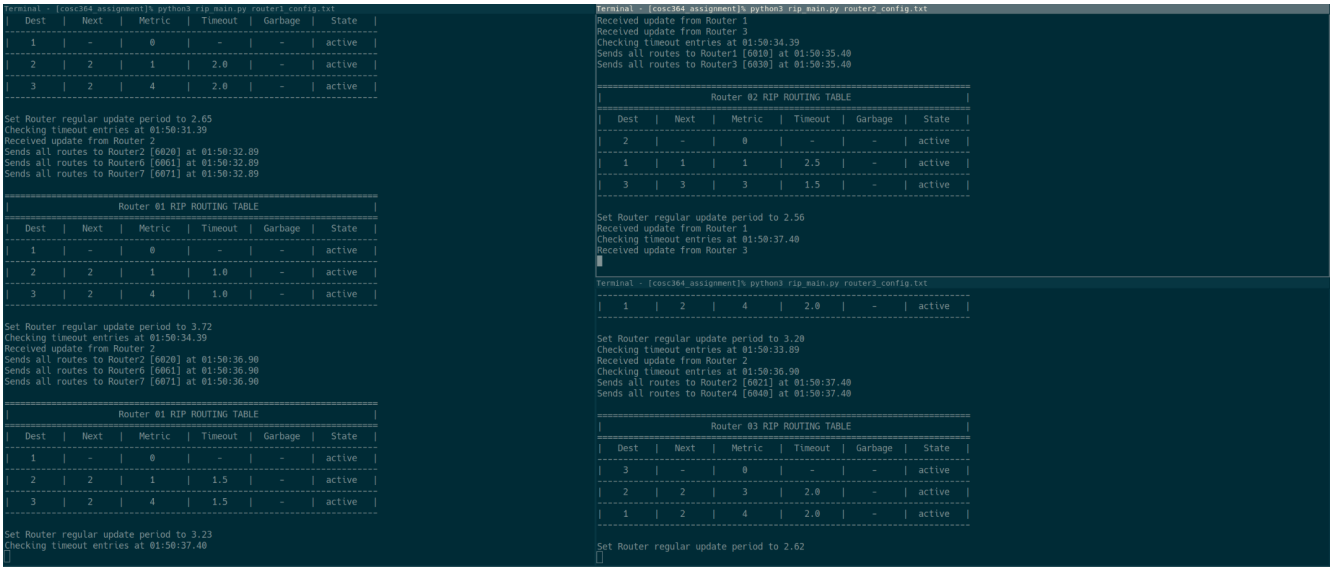
- Case 1: Discover neighbours and add new routes

  Description: Check if 3 routers can be started, discover their neighbours, and have their routing tables converge

  Steps: start router1, router2 and router3

  Expected result:
  - All 3 routers can be started without error messages.
  - All 3 routers can discover their neighbours after their first loop
  - Router 1 gets a route to router 3 through router 2
  - Router 3 gets a route to router 1 through router 2
  - All 3 routers have their routing tables converge in a short time.

  Status: Passed



- Case 2: Invalid route

  Description: Check if an invalid route can be timed out and removed after garbage collection time.

  Steps:
  1. Start router1, router2 and router3
  2. After the routing tables of the 3 routers converge, stop router 3

  Expected result:

  After the route to router 3 is invalid:
  - The timeout timer in the routing tables increments correctly
  - The garbage collection timer of the route to router 3 starts after the timeout timer expires, the route's state flag is labelled "dying", and the metric to router 3 is changed to infinity (16).

- When the garbage collection timer starts, an update for the invalid route is triggered, and a message of sending a triggered update is printed out with its timestamp.
- After the garbage collection timer expires, the route to router 3 is removed from the routing table of router 1 and router 2.
- No routing loops between router 1 and router 2

Status: Passed

```
Set Router regular update period to 2.98
Received update from Router 1
Checking timeout entries at 02:21:34.14
Triggered update for invalid route
Sends triggred update to Router1 [6010] at 02:21:34.14
Sends triggred update to Router3 [6030] at 02:21:34.14

=================================================================================
|                        Router 02 RIP ROUTING TABLE                            |
=================================================================================
|   Dest   |   Next   |   Metric   |  Timeout  |  Garbage  |   State   |
---------------------------------------------------------------------------------
|    2     |    -     |     0      |     -     |     -     |  active   |
---------------------------------------------------------------------------------
|    1     |    1     |     1      |    1.0    |     -     |  active   |
---------------------------------------------------------------------------------
|    3     |    3     |     16     |   18.5    |     0     |   dying   |
---------------------------------------------------------------------------------
```

- Case 3: Recover route
  Description: Check if a route can be recovered before/after being removed.
  Steps:
  1. Start router1, router2 and router3
  2. After the routing tables of the 3 routers converge, stop router 3
  3. Reconnect router 3 after the garbage-collection timer starts and before the route is removed from the routing table of router 2 and router 1
  4. After the routing tables of the 3 routers converge, stop router 3 again
  5. Reconnect router 3 after the route to router 3 is removed from the routing tables of router 2 and router 1

  Expected result:
  - Router 3 can be restarted
  - The route to router 3 can be added back to the routing tables of router 1 and router 2, and the stage flag is changed to “active” after router 3 is reconnected to the network
  - The routing tables of router 1/2/3 converge in a short time at last.

  Status: Passed

Conclusion of test 2:

- The tested routers can discover all their neighbours immediately after they are started.
- The tested routers can exchange and process routing table data from their neighbours and add new routes to their routing table.

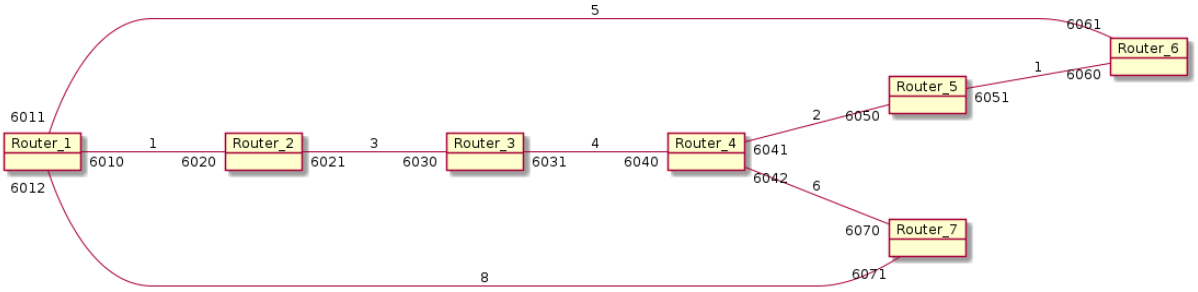
- When a route invalid, its directly-connected neighbour can start a garbage-collection process for the route after its timeout expires and send a triggered update to the neighbours immediately. When the garbage-collection timer expires, the invalid route can be removed from the routing table.
- No routing loop happens when an invalid route occurs at one end of a 3-router network, which means the implementation of split-horizon with poisoned reverse works in the program.
- A removed router can be connected back to the network during the garbage-collection process or after the route is removed.
- The routing tables can converge after new neighbours/routes are discovered, invalid routes are removed, and removed routers are reconnected.

Test 3: Routing functionality of a 7-router network

- Case 1: the shortest route

Description: Check if a router would choose the shortest route from multiple paths to the same destination

Steps: start router 1-7



Expected result:

- In a short time, all the routing tables can converge
- Each router can always choose the shortest route to other routers

Status: Passed

```
Checking timeout entries at 03:34:56.82
Received update from Router 7
Sends all routes to Router2 [6020] at 03:34:58.83
Sends all routes to Router6 [6061] at 03:34:58.83
Sends all routes to Router7 [6071] at 03:34:58.83

=================================================================
|                  Router 01 RIP ROUTING TABLE                  |
=================================================================
|   Dest   |   Next   |   Metric   |  Timeout  |  Garbage  |   State   |
-----------------------------------------------------------------
|    1     |    -     |     0      |     -     |     -     |  active   |
-----------------------------------------------------------------
|    2     |    2     |     1      |    2.5    |     -     |  active   |
-----------------------------------------------------------------
|    3     |    2     |     4      |    2.5    |     -     |  active   |
-----------------------------------------------------------------
|    4     |    2     |     8      |    2.5    |     -     |  active   |
-----------------------------------------------------------------
|    5     |    6     |     6      |    3.5    |     -     |  active   |
-----------------------------------------------------------------
|    6     |    6     |     5      |    3.5    |     -     |  active   |
-----------------------------------------------------------------
|    7     |    7     |     8      |    1.0    |     -     |  active   |
-----------------------------------------------------------------

Set Router regular update period to 2.38
```

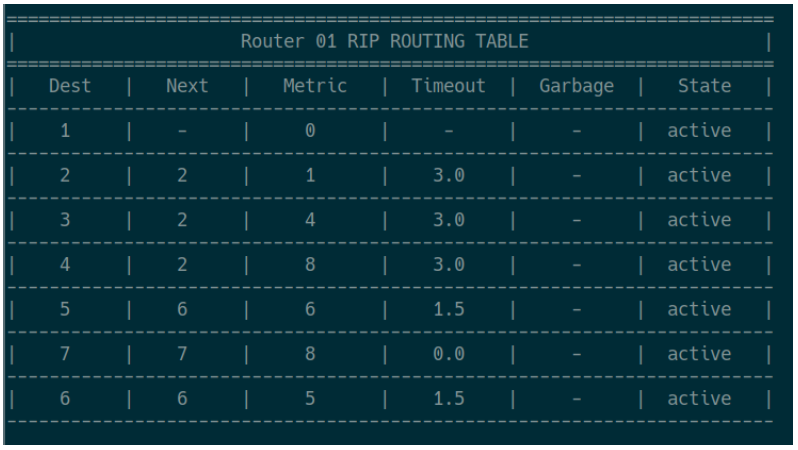
- Case 2: the shortest route after router removed

  Description: Check if a router would choose another optimal route after a neighbour through which a shortest route has passed is removed

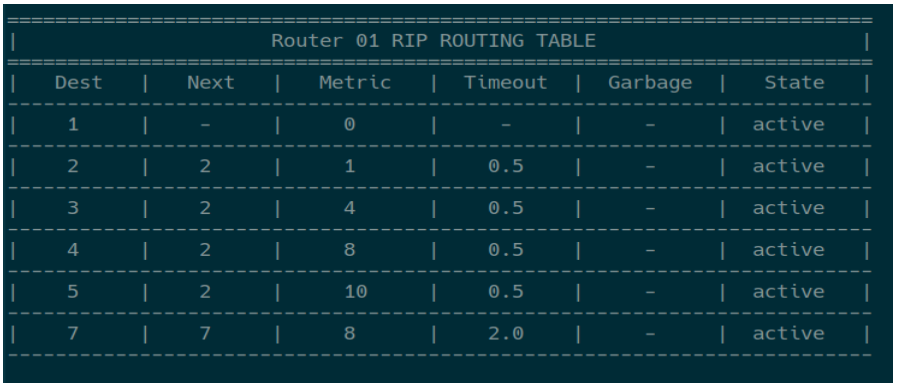
  Steps:
  - Start router 1-7
  - Wait until all routing tables converge
  - Remove router 6
  - Wait until all routing tables converge
  - Reconnect router 6
  - Wait until all routing tables converge

  Expected result:
  - Before removing router 6, router 1 chooses router 6 as the next hop to get to router 5
  - After removing router 6, router 1 chooses router 2 as the next hotp to get to router 5
  - After reconnecting router 6, router 1 again chooses router 6 as the next hop to get to router 5

  Status: Passed



Router 01 RIP ROUTING TABLE					
Dest	Next	Metric	Timeout	Garbage	State
1	-	0	-	-	active
2	2	1	3.0	-	active
3	2	4	3.0	-	active
4	2	8	3.0	-	active
5	6	6	1.5	-	active
7	7	8	0.0	-	active
6	6	5	1.5	-	active



Router 01 RIP ROUTING TABLE					
Dest	Next	Metric	Timeout	Garbage	State
1	-	0	-	-	active
2	2	1	0.5	-	active
3	2	4	0.5	-	active
4	2	8	0.5	-	active
5	2	10	0.5	-	active
7	7	8	2.0	-	active

Conclusion of test 3:

- When multiple paths are available to the same destination, a router can always choose the shortest route
- When a router through which a shortest route has passed is removed, a router can always choose another shortest route available to the same destination

- The routing tables can alway converge at last.

From test 1-3, we can conclude that the tested program can implement the rip protocol features specified in the assignment specs.

One example configuration file for the example network of Figure 1.

The configuration file for router 4: