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*.

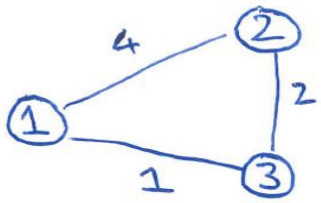


Figure 1 – Basic functionality test network

```
====================================================================
|                --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.

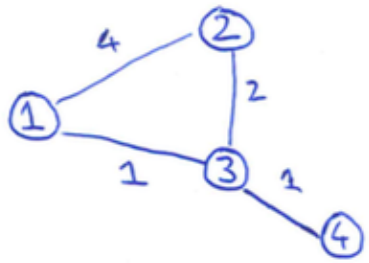
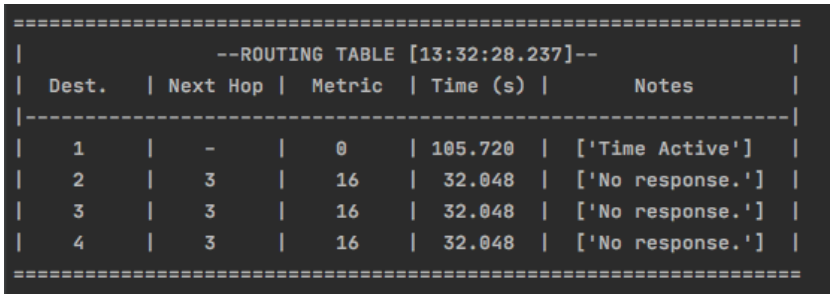


Figure 6 – Split horizon with poisoned reverse test network

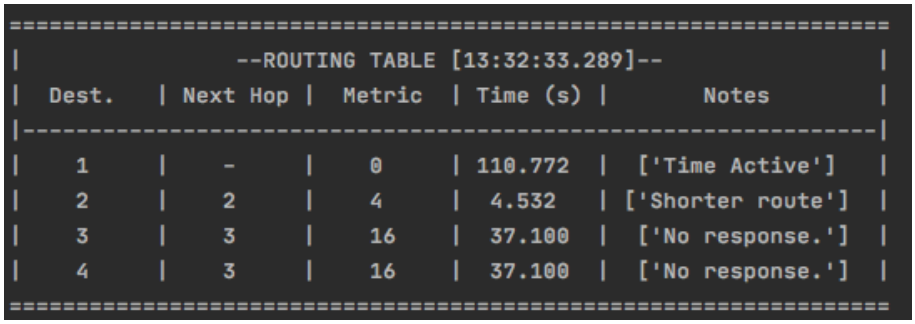
```
================================================================
|                --ROUTING TABLE [13:31:17.822]--               |
|  Dest.   | Next Hop |  Metric  | Time (s) |       Notes       |
|--------------------------------------------------------------|
|    1     |    -     |    0     |  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



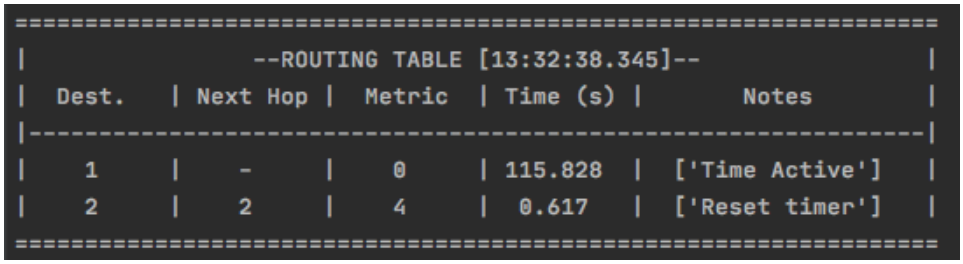
--ROUTING TABLE [13:32:28.237]--				
Dest.	Next Hop	Metric	Time (s)	Notes
1	-	0	105.720	['Time Active']
2	3	16	32.048	['No response.']
3	3	16	32.048	['No response.']
4	3	16	32.048	['No response.']

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*.

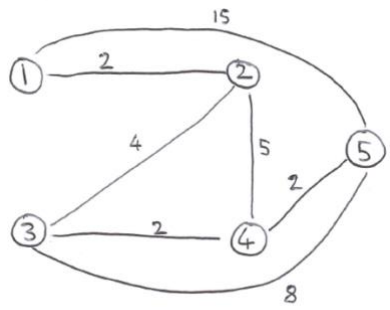


Figure 11 - five routers test network

```
================================================================
|                --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']  |
================================================================
```

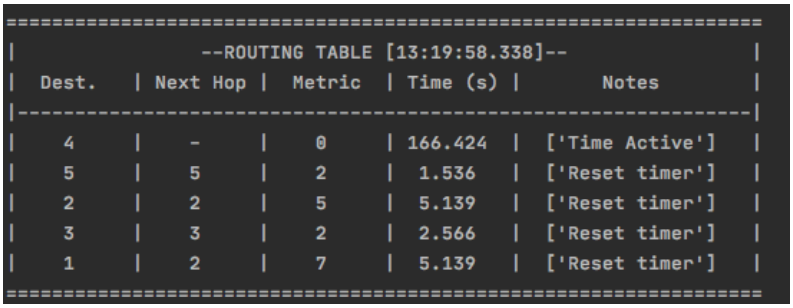
Figure 12 – Routing table for route one

```
================================================================
|                --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

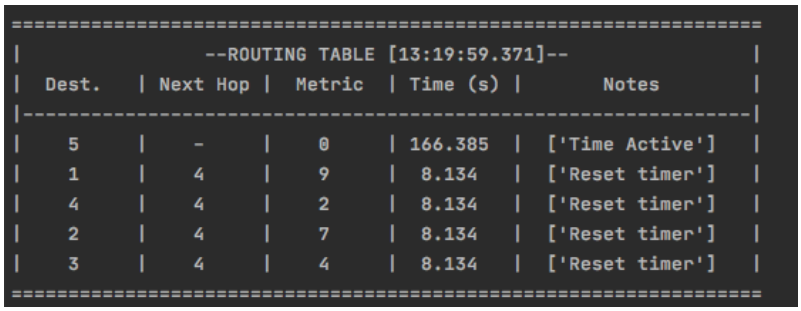
```
================================================================
|                --ROUTING TABLE [13:19:57.111]--               |
|  Dest.   | Next Hop |  Metric  | Time (s) |       Notes       |
|--------------------------------------------------------------|
|    3     |    -     |    0     | 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

```python
"""
Assignment 1: RIP protocol
Team: Bach Vu (25082165), Charlie Hunter (27380476)
Router main program/Daemon.py
"""
########## Header ##########
from daemon_sup import *
import socket, time, select
import sys, random # must use
import traceback # optional features
from router import Router, RTimer

LocalHost = "127.0.0.1"
ROUTER = None # Router Obj
SOCKETS = {} # Enabled Interfaces

########## Body ##########
def createSocket():
    for port in ROUTER.INPUT_PORTS:
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock.bind((LocalHost, port))
        SOCKETS[port] = sock

def send(mode):
    """ Send forwarding table to neighbour routers """
    for destID, link in ROUTER.OUTPUT_PORTS.items():
        table = ROUTER.get_routing_table(destID, mode)
        if len(table) == 0:
            # entry may got updated while delay & prepare package
            continue
        message = create_rip_packet(table)
        dest = (LocalHost, link[0])
        SOCKETS[link[2]].sendto(message, dest)

    print(f"Routing Table ({mode}) sent to neighbours at {strCurrTime()}.\n")
    if mode == ROUTER.REGULAR_UPDATE:
        ROUTER.reset_timer(RTimer.PERIODIC_TIMEOUT)


def send_periodic():
    mode = "None"
    if ROUTER.is_expired(RTimer.PERIODIC_TIMEOUT, getTime()):
        # Regular update
        mode = ROUTER.REGULAR_UPDATE
    elif ROUTER.has_expired_entry(getTime()):
        # Triggered update. Known NoResponse links don't trigger this
        mode = ROUTER.EXPIRED_UPDATE
    else:
        return

    send(mode)

def receive(timeout = 0.025):
    """ Return True if some data received """
    readable, _, _ = select.select(SOCKETS.values(), [], [], timeout)
    for sock in readable:
        receiver = sock.getsockname()
        data, sender = sock.recvfrom(1024)
```

```
59          if not ROUTER.is_expected_sender(sender, receiver):
60              print(f"Droped message on {sender} → {receiver} link!")
61              continue
62          routes = process_rip_packet(data)
63          print(f"Received ROUTES {str(routes)} at {strCurrTime(getTime())} from {sender}"
    )
64          triggered_update = ROUTER.update_route_table(routes, getTime())
65          if triggered_update:
66              # Next time, the record become Reset Timer
67              send(Router.FAST_ROUTE_UPDATE)
68
69  def garbage_collection():
70      ROUTER.garbage_collection(getTime())
71
72  ########## Program ##########
73  def init_router():
74      global ROUTER # include this if modifying global variable
75      filename = sys.argv[1]
76      rID, inputs, outputs, timeout = read_config(filename)
77
78      #  Router instance with default routing table
79      ROUTER = Router(rID, inputs, outputs, getTime(True), timeout)
80      ROUTER.print_hello()
81
82      # First time notice to neighbours
83      createSocket()
84
85  if __name__ == "__main__":
86      try:
87          init_router()
88          while True:
89              ROUTER.print_route_table(getTime())
90              send_periodic()
91              garbage_collection()
92              receive()
93
94      except IndexError:
95          print("Error: Config file is not provided!")
96      except FileNotFoundError:
97          print("Error: given Config file not found!")
98      except ValueError as v_err:
99          print("Warning:", v_err)
100     except socket.error as s_err:
101         print("Error:", s_err)
102     except KeyboardInterrupt:
103         print("\n******** Daemon exit successfully! Router shuting down... ********")
104     except Exception as e:
105         traceback.print_exc() # Traceback unknown error
106         print("Program exited unexpectedly.\n")
107     finally:
108         print()
109         sys.exit()
```

```python
"""
Assignment 1: RIP protocol
Team: Bach Vu (25082165), Charlie Hunter (27380476)
Router support function/daemon_sup.py
"""
import os, sys
import numpy as np
from datetime import datetime

FILE_EXTENSION = ".txt"

def read_config(filename):
    rID, inputs, outputs, timeout = None, None, None, None
    if filename.endswith(FILE_EXTENSION):
        config_file = open(filename)
    else:
        config_file = open(filename + FILE_EXTENSION)
    config_data = config_file.readlines()

    for line in config_data:
        head, data = line.split(':')
        if head == "router-id":
            rID = int(data)
            if not 0 < rID or rID > 64000:
                raise ValueError("Router ID must be between 1 and 64000.")
        elif head == "input-ports":
            inputs = [int(port) for port in data.rstrip().split(',')]
            if not is_valid_ports(inputs):
                raise ValueError("Invalid input port(s) in config data.\nPorts must be between 1024 and 64000.")
        elif head == "outputs":
            outputs = [port.strip() for port in data.rstrip().split(',')]
            ports = [int(output.split('-')[1]) for output in outputs]
            if not is_valid_ports(ports):
                raise ValueError("Invalid output port(s) in config data.\nPorts must be between 1024 and 64000.")
        elif head == "timer":
            timeout = int(data)
            if not 0 < timeout or timeout > 30:
                raise ValueError("Timeout must be between 1 and 30.")
    return rID, inputs, outputs, timeout

def is_valid_ports(ports):
    ports = np.array(ports)
    return np.all((ports ⩾ 1024) & (ports ⩽ 64000))

def create_rip_packet(table):
    header = create_rip_head()
    body = bytearray()
    for entry in table:
        new_entry = create_rip_entry(entry)
        body += new_entry
    return header + body

def create_rip_head(TTL=0):
    "Creates the 4 byte header"
    command = 1
    verison = 2
```

```
    command = command.to_bytes(1, byteorder='big')
    verison = verison.to_bytes(1, byteorder='big')
    reserve = (TTL+1).to_bytes(2, byteorder='big')
    return command + verison + reserve

def create_rip_entry(entry):
    "Creates the 20 byte body of packet"
    address_fam, zero = 0, 0
    afi         = address_fam.to_bytes(2, byteorder='big')
    route_tag   =        zero.to_bytes(2, byteorder='big')
    dest        =    entry[0].to_bytes(4, byteorder='big') # routerID
    subnet      =        zero.to_bytes(4, byteorder='big')
    next_hop    =    entry[1].to_bytes(4, byteorder='big')
    metric      =    entry[2].to_bytes(4, byteorder='big')
    return afi + route_tag + dest + subnet + next_hop + metric

def process_rip_packet(packet):
    command = int.from_bytes(packet[0:1], byteorder='big')
    version = int.from_bytes(packet[1:2], byteorder='big')
    if command =≠ 1 or version =≠ 2:
        return []

    routes = []
    entry_count = (len(packet)-4)//20
    for i in range(entry_count):
        si = i*20 + 4 # entry_start_index
        dest_id  = int.from_bytes(packet[si+4:si+8], byteorder='big')
        next_hop = int.from_bytes(packet[si+12:si+16], byteorder='big')
        metric   = int.from_bytes(packet[si+16:si+20], byteorder='big')
        routes.append((dest_id, next_hop, metric))

    return routes


def strCurrTime(time=None):
    if time is None:
        return datetime.now().strftime('%H:%M:%S')
    else:
        return time.strftime('%H:%M:%S')

def getTime(as_float=False):
    """ Get current time as float or object """
    if as_float:
        return datetime.now().timestamp()
    else:
        return datetime.now()
```

```python
"""
Assignment 1: RIP protocol
Team: Bach Vu (25082165), Charlie Hunter (27380476)
Router main program/router.py
"""
from timer import RTimer
from daemon_sup import strCurrTime

class Router:
    EXPIRED_UPDATE = "expired"
    REGULAR_UPDATE = "periodic"
    FAST_ROUTE_UPDATE = "Poison enhance"
    def __init__(self, rID, inputs, outputs, startTime, timeout):
        _timeout = timeout if timeout is not None else 5
        self.timer = RTimer(_timeout)
        self._garbages = {} # (dest, time since expired)

        self.ROUTER_ID = rID
        self.INPUT_PORTS = inputs

        self._ROUTING_TABLE = {}  # {Dest: nxt Hop, metric, time, note}
        self._ROUTING_TABLE[rID] = ["-", 0, startTime, "Time Active"]

        self.OUTPUT_PORTS = {} # (dest, cost, port_to_send)
        for output in outputs:
            from_port, to_port, cost, dest = output.split('-')
            from_port, to_port, cost, dest = int(from_port), int(to_port), int(cost), int
(dest)
            self.OUTPUT_PORTS[dest] = (to_port, cost, from_port)

    def get_routing_table(self, dest, mode):
        entries = []
        for key, val in self._ROUTING_TABLE.items():
            if dest == val[0] or dest == key:
                # don't re-advertise info from a hop (Split horizon)
                # dest == key not needed, but can reduce packet size
                continue
            if mode == "expired" and val[1] != 16:
                # triggered update, contain expired entries only (pg29)
                continue
            if mode == "Poison enhance" and val[3] != "Shorter route":
                continue

            new_metric = val[1] + self.OUTPUT_PORTS[dest][1]
            if new_metric > 15 and val[1] != 16:
                continue
            new_metric = min(new_metric, 16)
            entries.append((key, self.ROUTER_ID, new_metric))
        return entries

    def update_route_table(self, routes, utime):
        update_flag = False
        for route in routes:
            dest, nxtHop, metric = route
            new_entry = [nxtHop, metric, utime.timestamp(), ""]
            exist_entry = self._ROUTING_TABLE.get(dest, None)

            if not self._need_update(new_entry, exist_entry):
```

```
58                 continue
59
60             self._ROUTING_TABLE[dest] = new_entry
61             if new_entry[3] == "Shorter route":
62                 # trigger update with small delay. Not needed for small delay network.
63                 update_flag = True
64
65             # updated dest entry could be in garbage collecting
66             self._garbages.pop(dest, None)
67         return update_flag
68
69     def _need_update(self, new_entry, exist_entry):
70         """ For fancy purpose of taking note when update an entry
71             return True if new entry is valid to be updated
72         """
73         if exist_entry is None:
74             if new_entry[1] == 16:
75                 # Don't worry about dead link of unknown dest
76                 return False
77             new_entry[3] = "New dest."
78         else:
79             if new_entry[1] < exist_entry[1]:
80                 new_entry[3] = "Shorter route"
81
82             elif new_entry[1] == 16:
83                 if exist_entry[1] == 16:
84                     # already receive this link dead
85                     return False
86                 elif exist_entry[0] != new_entry[0]:
87                     # link dead is not currently in route table
88                     return False
89                 # 1st time known dest (metric < 16) has dead link
90                 new_entry[3] = "Link dead."
91
92             elif new_entry[1] == exist_entry[1]:
93                 new_entry[3] = "Reset timer"
94                 if new_entry[0] != exist_entry[0]:
95                     # New route, reset timer still
96                     new_entry[3] = "Same cost"
97
98             else:
99                 # ["Slower route."], not update
100                return False
101
102        return True
103
104    def garbage_collection(self, gtime):
105        if not self.timer.is_expired(RTimer.GARBAGES_TIMEOUT, gtime):
106            return False
107
108        for item, time in self._garbages.copy().items():
109            if self.timer.is_expired(RTimer.GARBAGE_TIMEOUT, gtime, time):
110                self._ROUTING_TABLE.pop(item, None)
111                self._garbages.pop(item)
112                print(f"Removed dead link to {item} at {strCurrTime(gtime)}")
113        self.timer.reset_timer(RTimer.GARBAGES_TIMEOUT)
114
115    def has_expired_entry(self, etime):
```

```
116         if not self.timer.is_expired(RTimer.ENTRIES_TIMEOUT, etime):
117             """ Trigger once if multilink die in short period """
118             return False
119 
120         garbage_found = 0
121         for dest,entry in self._ROUTING_TABLE.items():
122             if dest == self.ROUTER_ID:
123                 continue
124 
125             _, metric, ttl, _ = entry
126             if metric == 16:
127                 if dest in self._garbages.keys():
128                     # Waiting to be removed, skip to avoid sending same info to network
129                     continue
130                 self._garbages[dest] = etime.timestamp()
131                 garbage_found += 1
132 
133             elif self.timer.is_expired(RTimer.ENTRY_TIMEOUT, etime, ttl):
134                 entry[1], entry[3] = 16, "No response."
135                 self._ROUTING_TABLE[dest][1] = 16 # set to infinity
136                 self._garbages[dest] = etime.timestamp()
137                 print(f"Found expired link to {dest} at {strCurrTime(etime)}")
138                 garbage_found += 1
139 
140         self.timer.reset_timer(RTimer.ENTRIES_TIMEOUT)
141         # print(garbage_found)
142         return garbage_found > 0
143 
144     def is_expected_sender(self, sender, receiver):
145         """ Avoid unwanted broadcast/malicious pecket """
146         for link in self.OUTPUT_PORTS.values():
147             if sender[1] == link[0] and receiver[1] == link[2]:
148                 return True
149         return False
150 
151     def print_hello(self):
152         print("-"*66)
153         print(f"Router {self.ROUTER_ID} is running ...")
154         print("Input ports:", self.INPUT_PORTS)
155         print("Output ports:")
156         for dest, link in self.OUTPUT_PORTS.items():
157             print(f"    {link} to Router ID {dest}")
158         print("-"*66)
159         print("Use Ctrl+C or Del to shutdown.")
160         print()
161 
162     def print_route_table(self, ptime):
163         if not self.timer.is_expired(RTimer.PRINT_TIMEOUT, ptime):
164             return
165 
166         print("="*66)
167         print("|{:16}--{} [{}]--{:16}|".format(" ", "ROUTING TABLE", strCurrTime(ptime
    ), " "))
168         print("|{:^10}|{:^10}|{:^10}|{:^10}|{:^20}|".format(
169             "Dest.", "Next Hop", "Metric", "Time (s)", "Notes"))
170         print("|" + "-"*64 + "|")
171         for dest, record in self._ROUTING_TABLE.items():
172             hop, cost, log_time, note = record
```

```
            duration = ptime.timestamp() - log_time
            print("|{:^10}|{:^10}|{:^10}|{:^10.3f}|{:^20}|".format(
                dest, hop, cost, duration, str(note)))
        print("="*66)
        self.timer.reset_timer(RTimer.PRINT_TIMEOUT)

    def reset_timer(self, mode):
        self.timer.reset_timer(mode)

    def is_expired(self, mode, curr_time):
        return self.timer.is_expired(mode, curr_time)


```

```python
"""
Assignment 1: RIP protocol
Team: Bach Vu (25082165), Charlie Hunter (27380476)
Timer main program/timer.py
"""
import random, time

class RTimer:
    PRINT_TIMEOUT = 0
    PERIODIC_TIMEOUT = 1
    ENTRY_TIMEOUT = 2
    GARBAGE_TIMEOUT = 3
    ENTRIES_TIMEOUT = 4
    GARBAGES_TIMEOUT = 5
    def __init__(self, base):
        self._timeout = base
        self._time_logs = [-1.0, -1.0, -1.0, -1.0, -1.0, -1.0]

    def get_print_timeout(self):
        """ How often to print routing table """
        return self._timeout * 1/2

    def get_periodic_timeout(self):
        """ From config, Ripv2 value 30 +- (0,5) """
        return self._timeout * (1-random.uniform(-1/5, 1/5))

    def get_entry_timeout(self):
        """ Expiry of a routing entry. Ripv2 value 180 """
        return self._timeout * 6

    def get_garbage_timeout(self):
        """ Delete expired entry delay. Ripv2 value 120 """
        return self._timeout * 4

    def get_entry_check_timeout(self):
        """ Router periodic check expired entries """
        return 1 #self._timeout / 5

    def get_garbage_check_timeout(self):
        """ Router periodic check expired garbage """
        return 1 # self._timeout / 5

    def reset_timer(self, mode):
        self._time_logs[mode] = time.time()

    def is_expired(self, mode, curr_time, ttl=None):
        """ Check time log """
        curr_time = curr_time.timestamp()
        if ttl is not None:
            self._time_logs[mode] = ttl
        if self._time_logs[mode] == -1:
            return True

        timeout_value = [self.get_print_timeout, self.get_periodic_timeout,
            self.get_entry_timeout, self.get_garbage_timeout,
            self.get_entry_check_timeout, self.get_garbage_check_timeout]
        time_elapsed = curr_time - self._time_logs[mode]
        return time_elapsed >= timeout_value[mode]()
```

Plagiarism Declaration

This form needs to accompany your COSC 364 assignment submission.

I understand that plagiarism means taking someone else's work (text, program code, ideas, concepts) and presenting them as my own, without proper attribution. Taking someone else's work can include verbatim copying of text, figures/images, or program code, or it can refer to the extensive use of someone else's original ideas, algorithms or concepts.

I hereby declare that:

- My assignment is my own original work. I have not reproduced or modified code, figures/images, or writings of others without proper attribution. I have not used original ideas and concepts of others and presented them as my own.
- I have not allowed others to copy or modify my own code, figures/images, or writings. I have not allowed others to use original ideas and concepts of mine and present them as their own.
- I accept that plagiarism can lead to consequences, which can include partial or total loss of marks, no grade being awarded and other serious consequences, including notification of the University Proctor.

Name: Bach Vu & Charlie Hunter

Student ID: 25082165 & 27380676

Signature: 

Date: 15/04/2021