COSC364

Assignment 1

April 27, 2018

Kate Chamberlin 54384616

SHAN KOO 44001993

Contents

[Percentage Contribution 1](#_Toc512617523)

[Questions 1](#_Toc512617524)

[Testing 1](#_Toc512617525)

[Appendices 4](#_Toc512617526)

[Appendix 1: ConfigParser.py 4](#_Toc512617527)

[Appendix 2: Packet.py 6](#_Toc512617528)

[Appendix 3: Router.py 8](#_Toc512617529)

[Appendix 4: Config files and run file 14](#_Toc512617530)

# Percentage Contribution

Kate 50

Shan 50

# Questions

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

The configuration parser (ConfigParser.py) and the encoding/decoding of packets. We felt that the configuration parser was well-done as it was cleanly-written code that was laid out well. It utilised code that was already in the library (configparser) to reduce complexity.

We also felt that the way we serialised the packets was well-done (encode/decode in Packet.py). Using the struct module in the standard library was extremely useful in ensuring that the header was correctly padded. The way that we encoded/decoded our packets was also very efficient for parsing the RT entries directly into our main Router code.

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

We would like to improve the functions ‘update’ and ‘update\_routing\_table’. This is largely because the nested ‘for’ loops and ‘if’ statements reduced readability and made it more complex to make small adjustments to our code.

We also thought that we could have been better with our modularisation – some of our functions contain code that could most likely have been put into more suitable functions if we had been more forward-thinking with our planning of this assignment.

**How have you ensured atomicity of event processing?**

We have ensured atomicity by running each type of event in a new thread. For example, our timers were in separate threads to our periodic updates, to ensure that none of our essential functions were blocked. This meant that all sockets and timers could perform simultaneously, without losing packets or missing timer calls.

In addition to the threading, we used thread locking to ensure that no important data was over-written. Our routing table for example was locked so that only one thread could modify it at a time.

# Testing

Within the Packet.py module we tested thoroughly to ensure that the encoding/decoding worked correctly, and none of the information was lost as it was being converted in the serialization process. For this we tested empty packets, full packets, triggered-update packets and packets with invalid variables. These tests resulted in the expected outcome – a decoded dictionary containing the RTEs that we input for the tests. For example, entering RTEs {3: [4, 5], 6:[3, 2]} and encoding and decoding it resulted in {3: [4, 5], 6:[3, 2]}, and so on. We also tested the split horizon code within this module to ensure that it sent metrics of infinity to the routers it had learned the route from. Some values (such as the version number) we did not test since the brief stated that they would be constant.

The ConfigParser.py module was tested with several input values, both valid and invalid. Similar to the Packet.py testing, some variables that were stated as constant in the brief did not have relevant exceptions coded into ConfigParser since they were assumed to be correct. However, our testing initially showed that we had not raised our port number error correctly – port numbers below 1024 were still considered valid. We therefore altered our code and after that had no issues in the rest of our tests.

Router.py contained several, more complex tests. We tested the initialisation quite thoroughly with expected results for our neighbour list, metrics, port lists, socket creation and sending empty packets to ensure that the connections were fine. For example, creating a router with a given configuration file would print out the neighbours, both port lists, both socket lists, and their given metrics. This showed us that there were no issues with module imports or parsing, so we then moved on to our threading.

Our thread testing involved printing an active count of our threads while code was running and printing the output from those threads on a regular basis. Initially we were having difficulties with the threads as they kept throwing a ‘bootleg’ error that we could not trace back. The thread count was also increasing with time, and thus increasing CPU load which is undesirable. This error was not fatal however and it kept looping through the code. Eventually we realised that we had made a syntactical error by putting an argument into our threaded function call, when instead we needed to parse them in as a list. Once that was working we then culled the main thread to check if our threads were still running independently and they did continue to run as expected. At this stage we had not implemented thread locking.

Router filling/updating was the most complex as we needed to test our routing table update functions. Since they were nested it took considerably longer than our previous testing. We began by testing how the neighbours filled the routing table with expected results, and then tried to test the convergence of the entire demonstration network. The convergence was not working however, and some of the routers were maintaining routes with higher metrics than expected. This was because we were updating local variables rather than the global routing table, and once we fixed that our network converged correctly.

We also had higher metrics than expected elsewhere, but after creating new configuration files we realised that this was due to incorrect ports and metrics in the files. This was easily solved. After manually calculating the expected next-hops and metrics using the distance-vector algorithm, we could see that our convergence was correct, with each router printing expected routing tables (bar one or two instances where the router had chosen an alternate route with the same metric).

We then tested the robustness of our RIP implementation by culling and restarting routers. We were having difficulty at this point with our time-out working as expected, as our garbage collection was being over-written with each periodic update. When a router was dropped, the time-out would initialise, count up, and once it hit the maximum value it would set the route to infinity, but the garbage collection would not move past our time period (two seconds in testing). This was solved by adjusting our main loop, as we had a superfluous condition that was over-writing the garbage collection.

After this, the program performed as expected. Convergence, robustness, and timers all worked well. However, we realised that our atomicity could have been better, as some threads were printing to the terminal in the middle of another thread’s print call. We therefore implemented thread locking and tested that it was working by adjusting timer variables. Without thread locking this resulted in multiple threads printing at once, however once the thread locking was implemented this no longer happened.

# Appendices

## Appendix 1: ConfigParser.py

"""RIP ROUTING ASSIGNMENT - COSC364

ConfigParser.py - code for parsing from config files into the router.

Authors: Shan Koo and Kate Chamberlin

Due date: 27/04/2018, 11:59pm

Date of last edit: 26/04/2018 """

**import** configparser

**import** sys

MAX\_PORT **=** 64000

MIN\_PORT **=** 1024

MAX\_ID **=** 64000

MIN\_ID **=** 1

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Function to parse the user configuration of the router.

# @param filename the filename of the config text file

# @return configurations in format [self ID, [input ports], [output ports]]

# where output ports are of format [port, metric, peer ID]

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** get\_config**(**filename**):**

all\_ports **=** **[]**

config\_list **=** **[]**

output\_ports **=** **[]**

config **=** configparser**.**ConfigParser**()**

config**.**read**(**filename**)**

# Read the router ID

router\_id **=** int**(**config**.**get**(**'Router'**,** 'router-id'**))**

# Check validity of router id

**if** router\_id **<** MIN\_ID **or** router\_id **>** MAX\_ID**:**

**raise** Exception**(**"Error - Router ID must be between 1 and 64000"**)**

# Read the input ports

input\_ports **=** config**.**get**(**'Router'**,** 'input-ports'**).**split**(**" "**)**

all\_ports **=** list**(**input\_ports**)**

#read the output ports

outputs\_split **=** config**.**get**(**'Router'**,** 'output-ports'**).**split**(**" "**)**

**for** output **in** outputs\_split**:**

output\_data **=** output**.**split**(**"-"**)**

all\_ports**.**append**(**output\_data**[**0**])**

output\_ports**.**append**(**output\_data**)**

# Check validity of all ports

check\_ports**(**all\_ports**)**

config\_list**.**append**(**router\_id**)**

config\_list**.**append**(**input\_ports**)**

config\_list**.**append**(**output\_ports**)**

**return** config\_list

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Function to check the validity of all ports

# @param ports\_list the list of all ports both in and out

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** check\_ports**(**ports\_list**):**

# Check the port number

**for** port **in** ports\_list**:**

port **=** int**(**port**)**

**if** port **<** MIN\_PORT **or** port **>** MAX\_PORT**:**

**raise** Exception**(**"Error - Port number must be between 1024 and 64000"**)**

# Check for duplicates

**if** len**(**set**(**ports\_list**))** **!=** len**(**ports\_list**):**

raise Exception("Error - Duplicate port number")

## Appendix 2: Packet.py

"""RIP ROUTING ASSIGNMENT - COSC364

Packet.py - code for packet struct and relevant functions.

Authors: Shan Koo and Kate Chamberlin

Due date: 27/04/2018, 11:59pm

Date of last edit: 26/04/2018 """

**import** socket

**import** struct

TAG **=** 0 # Since there is no IGP/BGP routing, always 0

COMMAND **=** 2 # No request packets so always 2

VERSION **=** 2 # RIPv2

AFI **=** socket**.**AF\_INET # Address Family for IPv4

INFINITY **=** 16 # Infinity metric

# set format and calculate sizes, see

# https://docs.python.org/2/library/struct.html#format-characters

HEADER\_FORMAT **=** "!BBH"

RTE\_FORMAT **=** "!HHIII"

HEADER\_SIZE **=** struct**.**calcsize**(**HEADER\_FORMAT**)**

RTE\_SIZE **=** struct**.**calcsize**(**RTE\_FORMAT**)**

**class** **Packet:**

src **=** 0

dst **=** 0

rtes **=** **{}**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Initialises the packet

# @param src the source id

# @param dst the dst id

# @param routing\_table the routing table

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** \_\_init\_\_**(**self**,** src**,** dst**,** routing\_table**):**

self**.**src **=** src

self**.**dst **=** dst

self**.**rtes **=** routing\_table

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Function to encode the packet into a binary string

# @return the encoded packet

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** encode**(**self**):**

metric **=** 0

# Pack the header into a binary format given by RFC

encoded\_packet **=** struct**.**pack**(**HEADER\_FORMAT**,** COMMAND**,** VERSION**,** self**.**src**)**

**for** key **in** self**.**rtes**.**keys**():**

**if** **(**key **!=** self**.**dst**):** # Doesn't send its own route

nxt\_hop **=** self**.**rtes**[**key**][**0**]**

# Implement split horizon with poisoned reverse

**if** **(**self**.**dst **==** nxt\_hop**):**

metric **=** INFINITY

**else:**

metric **=** self**.**rtes**[**key**][**1**]**

# Pack each RTE and add it to the binary packet

encoded\_packet **+=** struct**.**pack**(**RTE\_FORMAT**,** AFI**,** TAG**,**

key**,** nxt\_hop**,** metric**)**

**return** encoded\_packet

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Function to decode the packet from binary string.

# @param filename the filename of the config text file

# @return the decoded RTEs in format: {dest: [next hop, metric]}

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** decode**(**self**,** data**):**

num\_rtes **=** int**((**len**(**data**)** **-** HEADER\_SIZE**)** **/** RTE\_SIZE**)**

decoded\_rte\_table **=** **{}**

# Unpack the header

header **=** struct**.**unpack\_from**(**HEADER\_FORMAT**,** data**)**

self**.**COMMAND **=** header**[**0**]**

self**.**VERSION **=** header**[**1**]**

self**.**src **=** header**[**2**]**

# Unpack each RTE, beginning from the first one.

i **=** HEADER\_SIZE

**while** i **<** len**(**data**):**

rte **=** struct**.**unpack\_from**(**RTE\_FORMAT**,** data**[**i**:])**

# Check validity of RTE

**if** rte**[**0**]** **==** AFI **and** rte**[**1**]** **==** TAG **and** rte**[**4**]** **>=** 1 **and** rte**[**4**]** **<=**16**:**

addr **=** rte**[**2**]**

nxt\_hop **=** rte**[**3**]**

metric **=** rte**[**4**]**

decoded\_rte\_table**[**addr**]** **=** **[**nxt\_hop**,** metric**]**

i **+=** RTE\_SIZE #increment by size of one RTE

**else:**

i **+=** RTE\_SIZE

**return** decoded\_rte\_table

## Appendix 3: Router.py

"""RIP ROUTING ASSIGNMENT - COSC364

Router.py - Main code for virtual routers.

Authors: Shan Koo and Kate Chamberlin

Due date: 27/04/2018, 11:59pm

Date of last edit: 27/04/2018 """

**import** select

**import** random

**import** socket

**import** os**.**path

**import** threading

**import** time

**import** sys

**import** ConfigParser

**from** Packet **import** Packet

#enumeration for the dictionary format (no spaces to ensure difference)

NEXTHOP **=** 0

METRIC **=** 1

RCF **=** 2

TIMEOUT **=** 3

GARBAGECOLL **=** 4

PORT **=** 0

#constants

HOST **=** "127.0.0.1"

INFINITY **=** 16

INVALID **=** 16

TIME\_BLOCK **=** 15

PERIODIC\_UPDATE **=** 30 **/** TIME\_BLOCK

TIME\_OUT **=** 180 **/** TIME\_BLOCK

GARBAGE\_COLLECTION **=** 120 **/** TIME\_BLOCK

**class** **Router:**

# Local variables

lock **=** threading**.**RLock**()**

router\_id **=** 0 # Router ID of this router

input\_socks **=** **[]** # List of input sockets

rt\_tbl **=** **{}** # Dict of format {dest: [next hop, metric, RCF, timeout, garbage collection]}

neighbours **=** **{}** # Dict of format {router ID: [port, metric]

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Initialise the router

# @param config\_file the router configuration file

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** \_\_init\_\_**(**self**,** config\_file**):**

# Parse configurations

config\_list **=** ConfigParser**.**get\_config**(**config\_file**)**

self**.**router\_id **=** config\_list**[**0**]** # Parse router ID

# Parse and set input ports

**for** port **in** config\_list**[**1**]:** #line 2, input ports

port **=** int**(**port**)**

socket **=** self**.**create\_socket**(**port**)**

self**.**input\_socks**.**append**(**socket**)**

# Parse and set output ports

**for** port**,** metric**,** router **in** config\_list**[**2**]:** #line 3, output ports

router **=** int**(**router**)**

port **=** int**(**port**)**

metric **=** int**(**metric**)**

self**.**neighbours**[**router**]** **=** **[**port**,** metric**]**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Gets the metric of a neighbour

# @param router\_id the router id

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** get\_neighbour\_metric**(**self**,** router\_id**):**

**return** self**.**neighbours**[**router\_id**][**METRIC**]**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Gets the port of a neighbour

# @param router\_id the router id

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** get\_neighbour\_port**(**self**,** router\_id**):**

**return** self**.**neighbours**[**router\_id**][**PORT**]**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Creates socket for the port number

# @param port the port number

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** create\_socket**(**self**,** port**):**

sock **=** socket**.**socket**(**socket**.**AF\_INET**,** socket**.**SOCK\_DGRAM**)**

sock**.**setsockopt**(**socket**.**SOL\_SOCKET**,** socket**.**SO\_REUSEADDR**,** 1**)**

sock**.**setblocking**(False)**

sock**.**bind**((**HOST**,** port**))**

**print(**"Socket " **+** str**(**port**)** **+** " created"**)**

**return** sock

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Sends the packet

# @param packet the packet

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** send\_packet**(**self**,** packet**):**

encoded\_packet **=** packet**.**encode**()**

**try:**

# Using first socket as default

self**.**input\_socks**[**0**].**sendto**(**encoded\_packet**,** **(**HOST**,**

self**.**neighbours**[**packet**.**dst**][**PORT**]))**

**except** Exception**:**

**print(**"Could not send packet to destination."**)**

**return**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Triggers update

# @param src the source router id that triggered the update

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** trigger\_update**(**self**,** src**):**

changed **=** **{}**

**for** dest **in** self**.**rt\_tbl**.**keys**():**

**if** self**.**rt\_tbl**[**dest**][**RCF**]** **==** 1**:**

changed**[**dest**]** **=** self**.**rt\_tbl**[**dest**]**

self**.**rt\_tbl**[**dest**][**RCF**]** **=** 0

**if** len**(**changed**)** **>** 0**:**

delay **=** random**.**randint**(**1**,** 5**)**

thread **=** threading**.**Timer**(**delay**,** self**.**init\_trigger\_update**,**

args **=** **[**changed**,** src**])**

thread**.**daemon **=** **True**

thread**.**start**()**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Function to send trigger update to the neighbours

# @param changed a list of changed route(s)

# @param src the source destination that calls the trigger update

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** init\_trigger\_update**(**self**,** changed**,** src**):**

**for** neighbour **in** self**.**neighbours**:**

**if** neighbour **!=** src**:**

packet **=** Packet**(**self**.**router\_id**,** neighbour**,** changed**)**

self**.**send\_packet**(**packet**)**

**return**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Sends periodic update to neighbours

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** send\_update**(**self**):**

period **=** float**(**random**.**randint**(**8**,** 12**)** **/** 10**)**

thread **=** threading**.**Timer**(**period **\*** PERIODIC\_UPDATE**,** self**.**send\_update**)**

thread**.**daemon **=** **True**

thread**.**start**()**

**for** neighbour **in** self**.**neighbours**:**

packet **=** Packet**(**self**.**router\_id**,** neighbour**,** self**.**rt\_tbl**)**

self**.**send\_packet**(**packet**)**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Process incoming packet

# @param data the packet

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** read\_packet**(**self**,** data**):**

in\_packet **=** Packet**(**0**,** self**.**router\_id**,** **{})** # Init class as placeholder

rte\_table **=** in\_packet**.**decode**(**data**)** # Decode the data

packet\_src **=** in\_packet**.**src

self**.**update\_rt\_tbl**(**packet\_src**,** rte\_table**)** # Update the routing table

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Starts time out

# @param packet\_src the source router id

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** start\_time\_out**(**self**,** packet\_src**):**

**for** dest **in** self**.**rt\_tbl**.**keys**():**

**if** self**.**rt\_tbl**[**dest**][**NEXTHOP**]** **==** packet\_src **or** dest **==** packet\_src**:**

**if** **(**self**.**rt\_tbl**[**dest**][**METRIC**]** **<** INFINITY**):**

self**.**rt\_tbl**[**dest**][**TIMEOUT**]** **=** time**.**time**()**

self**.**init\_time\_out**(**packet\_src**)**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Updates routing table according to RIP protocol

# @param packet\_src the source router id

# @param rtes the routing entries received in a packet

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** update\_rt\_tbl**(**self**,** packet\_src**,** rtes**):**

self**.**lock**.**acquire**()** #locking mechanism for threads

keys **=** self**.**rt\_tbl**.**keys**()**

neighbours **=** self**.**neighbours**.**keys**()**

nxt\_hop **=** packet\_src

nxt\_hop\_metric **=** self**.**get\_neighbour\_metric**(**nxt\_hop**)**

# add a new route

**if** packet\_src **not** **in** keys**:**

self**.**update\_route**(**nxt\_hop**,** nxt\_hop**,** nxt\_hop\_metric**,** 0**)**

**for** dest **in** rtes**.**keys**():**

metric **=** rtes**[**dest**][**1**]**

new\_metric **=** min**(**nxt\_hop\_metric **+** metric**,** INFINITY**)**

# Route does not exist

**if** dest **not** **in** keys**:**

**if** new\_metric **<** INFINITY**:**

self**.**rt\_tbl**[**dest**]** **=** **[**nxt\_hop**,** new\_metric**,** 0**,** 0**,** 0**]**

# Route exist

**else:**

rt\_nxt\_hop **=** self**.**rt\_tbl**[**dest**][**NEXTHOP**]**

rt\_metric **=** self**.**rt\_tbl**[**dest**][**METRIC**]**

rt\_garbage **=** self**.**rt\_tbl**[**dest**][**GARBAGECOLL**]**

# Valid route

**if** rt\_metric **<** INFINITY**:**

# Same next hop

**if** nxt\_hop **==** rt\_nxt\_hop**:**

# Metric changed

**if** new\_metric **!=** rt\_metric**:**

# Route becomes invalid

**if** rt\_metric **<** INFINITY **and** new\_metric **>=** INFINITY**:**

**if** self**.**rt\_tbl**[**dest**][**GARBAGECOLL**]** **==** 0**:**

self**.**rt\_tbl**[**dest**][**METRIC**]** **=** new\_metric

self**.**rt\_tbl**[**dest**][**RCF**]** **=** 1

self**.**rt\_tbl**[**dest**][**GARBAGECOLL**]** **=** time**.**time**()**

self**.**init\_gbg\_coll**(**dest**)**

# Route metric changed

**else:**

self**.**update\_route**(**dest**,** nxt\_hop**,** new\_metric**,** 0**)**

# Different next hop

**else:**

# New optimal path found

**if** new\_metric **<** rt\_metric**:**

self**.**update\_route**(**dest**,** nxt\_hop**,** new\_metric**,** 0**)**

# Invalid route

**else:**

# Another route found

**if** new\_metric **<** INFINITY**:**

self**.**update\_route**(**dest**,** nxt\_hop**,** new\_metric**,** 0**)**

self**.**start\_time\_out**(**packet\_src**)**

self**.**trigger\_update**(**packet\_src**)**

self**.**print\_routing\_table**()**

self**.**lock**.**release**()**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Updates routing table entry

# @param dest the destination id

# @param nxt\_hop the next hop id

# @param new\_metric the metric

# @param rcf the route change flag

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** update\_route**(**self**,** dest**,** nxt\_hop**,** new\_metric**,** rcf**):**

self**.**rt\_tbl**[**dest**]** **=** **[**nxt\_hop**,** new\_metric**,** rcf**,** 0**,** 0**]**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Function to check time out of an entry

# @param src the source router id

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** check\_time\_out**(**self**,** src**):**

self**.**lock**.**acquire**()**

**for** dest **in** self**.**rt\_tbl**.**keys**():**

**if** dest **==** src **or** self**.**rt\_tbl**[**dest**][**NEXTHOP**]** **==** src**:**

# Route is invalid

**if** time**.**time**()** **-** self**.**rt\_tbl**[**dest**][**TIMEOUT**]** **>** TIME\_OUT**:**

self**.**rt\_tbl**[**dest**][**METRIC**]** **=** INFINITY

self**.**rt\_tbl**[**dest**][**RCF**]** **=** 1

**if** self**.**rt\_tbl**[**dest**][**GARBAGECOLL**]** **==** 0**:**

self**.**rt\_tbl**[**dest**][**GARBAGECOLL**]** **=** time**.**time**()**

self**.**init\_gbg\_coll**(**dest**)**

self**.**trigger\_update**(**src**)**

self**.**lock**.**release**()**

**return**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Function to check garbage collection of an entry

# @param dest the destination id

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** check\_gbg\_coll**(**self**,** dest**):**

self**.**lock**.**acquire**()**

**if** self**.**rt\_tbl**[**dest**][**4**]** **!=** 0**:**

# Route removed

**if** **(**time**.**time**()** **-** self**.**rt\_tbl**[**dest**][**GARBAGECOLL**])** **>**

GARBAGE\_COLLECTION**:**

**del** self**.**rt\_tbl**[**dest**]**

self**.**lock**.**release**()**

**return**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Function to start thread to check time out

# @param src the source id

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** init\_time\_out**(**self**,** src**):**

thread **=** threading**.**Timer**(**TIME\_OUT**,** self**.**check\_time\_out**,** args **=** **[**src**])**

thread**.**daemon **=** **True**

thread**.**start**()**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Function to start thread to check garbage collection

# @param dest the destination id

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** init\_gbg\_coll**(**self**,** dest**):**

thread **=** threading**.**Timer**(**GARBAGE\_COLLECTION**,**

self**.**check\_gbg\_coll**,**

args **=** **[**dest**])**

thread**.**daemon **=** **True**

thread**.**start**()**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Prints routing table in format:

# Destination, Next Hop, Metric, Time-out, Garbage Collection

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** print\_routing\_table**(**self**):**

template **=** "{0:^15d} | {1:^12d} | {2:^10d} | {3:^12.2f} | {4:^20.2f}"

**print(**"Router {0}"**.**format**(**self**.**router\_id**))**

**print(**"{0:^15s} | {1:^12s} | {2:^10s} | {3:^12s} | {4:^20s}"**.**format**(**

"Destination"**,** "Next Hop"**,** "Metric"**,** "Time Out"**,**

"Garbage Collection"**).**rstrip**())**

**for** dest **in** self**.**rt\_tbl**.**keys**():**

**if** self**.**rt\_tbl**[**dest**][**GARBAGECOLL**]** **==** 0**:**

**print(**template**.**format**(**dest**,** self**.**rt\_tbl**[**dest**][**NEXTHOP**],**

self**.**rt\_tbl**[**dest**][**METRIC**],**

time**.**time**()** **-** self**.**rt\_tbl**[**dest**][**TIMEOUT**],**

0**).**rstrip**())**

**else:**

**print(**template**.**format**(**dest**,** self**.**rt\_tbl**[**dest**][**NEXTHOP**],**

self**.**rt\_tbl**[**dest**][**METRIC**],**

time**.**time**()** **-** self**.**rt\_tbl**[**dest**][**TIMEOUT**],**

time**.**time**()** **-** self**.**rt\_tbl**[**dest**]**

**[**GARBAGECOLL**]).**rstrip**())**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Runs the router

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** run**(**self**):**

self**.**send\_update**()**

**while** **True:**

read\_ready**,**

write\_ready**,**

except\_ready **=** select**.**select**(**self**.**input\_socks**,** **[],** **[])**

**for** sock **in** read\_ready**:**

data**,** src **=** sock**.**recvfrom**(**512**)**

self**.**read\_packet**(**data**)**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Main function to start the router

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**def** main**():**

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

**try:**

**if** len**(**sys**.**argv**)** **<** 2**:**

**print(**"No file given"**)**

sys**.**exit**(**0**)**

filename **=** str**(**sys**.**argv**[-**1**])**

**if** os**.**path**.**exists**(**filename**):**

router **=** Router**(**filename**)**

router**.**run**()**

**else:**

**print(**"File does not exist"**)**

sys**.**exit**(**0**)**

**except** **(**KeyboardInterrupt**,** SystemExit**):**

sys**.**exit**(**0**)**

main**()**

## Appendix 4: Config files and run file

**[Router]**

router-id**:** 1

input-ports**:** 1111 1115 1118

output-ports**:** 2001-1-2 7008-8-7 6005-5-6

**[Router]**

router-id**:** 2

input-ports**:** 2001 2003

output-ports**:** 3003-3-3 1111-1-1

**[Router]**

router-id**:** 3

input-ports**:** 3003 3004

output-ports**:** 4004-4-4 2003-3-2

**[Router]**

router-id**:** 4

input-ports**:** 4002 4004 4006

output-ports**:** 3004-4-3 5002-2-5 7006-6-7

**[Router]**

router-id**:** 5

input-ports**:** 5001 5002

output-ports**:** 4002-2-4 6001-1-6

**[Router]**

router-id**:** 6

input-ports**:** 6001 6005

output-ports**:** 1115-5-1 5001-1-5

**[Router]**

router-id**:** 7

input-ports**:** 7006 7008

output-ports**:** 1118-8-1 4006-6-4

#!/bin/sh

gnome-terminal -e "python3 Router.py config\_1.ini"

gnome-terminal -e "python3 Router.py config\_2.ini"

gnome-terminal -e "python3 Router.py config\_3.ini"

gnome-terminal -e "python3 Router.py config\_4.ini"

gnome-terminal -e "python3 Router.py config\_5.ini"

gnome-terminal -e "python3 Router.py config\_6.ini"

gnome-terminal -e "python3 Router.py config\_7.ini"