COSC 364
Internet Technologies and Engineering
Assignment 1

Kayle Ransby
34043590

Sjaak Flick
36406121

Kayle Ransby: 34043590
Sjaak Flick: 36406121

Percentage contribution of each partner:

- Kayle Ransby: 55%
- Sjaak Flick: 45%

Which aspects of our program do we consider particularly well done?

- The configuration files can accommodate standard python comments (# Comment) and whitespace characters. Input parameters can also be given in any order, and unknown parameters will be ignored.
- Each daemon in the “network” is a single python class, which makes it easier to diagnose issues related to the operation of the daemon.
- The intervals for the required timers in the RIP protocol are stored in the demon class. These intervals can be divided by a common factor allowing the speed at which the demon operates to increase. This made debugging and testing our code much simpler.
- When a router goes offline, each of its neighbours will delete the route in their route tables once both the dead timer and garbage timers expire.

Which aspects of our program could be improved?

- The usage of the threading module makes the program work as intended, but is quite inefficient in terms of overall speed of the program
- Our configuration parameters can be declared multiple times but only the first occurance will be applied. An improvement could be to allow multiple declarations of the same parameter

How have you ensured atomicity of event processing?

We have ensured atomicity of event processing through the use of timers from the python threading library. Once a timer is started, it will wait for a specified amount of time before executing a function with the parameters you give it. We are using threading timers for periodic updates, route timeout and route garbage collection. Because these timers are running on separate threads, they do not interfere with the main loop of the routing demon. This allows for reliable and interference free execution of events.

Weaknesses of the RIP routing protocol:

One major issue with RIP is the count to infinity problem; this can be mitigated with the use of split horizon with poisoned reverse and triggered updates.
The entire routing table is sent with every periodic update, this is quite inefficient in many ways. It wastes bandwidth transmitting larger sized packets and the router’s are not gaining any new information the majority of the time.
If they do gain new information, it would be better to send the new information as a separate packet.

Kayle Ransby: 34043590
Sjaak Flick: 36406121

Program testing:

Test: Processing configuration files
Expected output:
If all of the parameters are within the ranges allowed and the correct syntax is used for comments in the files, the daemon should start and run without issues.
Otherwise, an error should be printed on the screen and the daemon should close
Actual Output:
An issue encountered is that while the error message would be printed, there would also be another error message relating to the startup of the periodic timers.
Conclusion:
When an error occurs, we must first check that the periodic timers are running before trying to close them.

Test: “Switching off” a particular router in the network.
The purpose of this test is to determine if the demons recalculate the metrics of any routes affected by the router going offline.
Expected Output:
Any entries to the now disconnected router that exist on the network would linger on for as long as the dead-timer + garbage collector would allow for.
Any entries in the route table that went through the router that was switched off, would then have to be recalculated.
Actual Output:
The behaviour of the routing demons when this test was performed was to be expected. Because there’s no updates coming from the router that’s been turned off, the timeout variable ran down to 0. This triggered the garbage collection timer, once the value of the garbage collector went to 0, the route was deleted from the table.
Conclusion:
Our implementation of this feature was correct.

Test: Periodic updates.
Expected Output:
Within a set interval, a periodic update packet would be sent to each of a demon’s neighbours, and in turn, a demon would receive packets from their neighbouring demons. The intended behaviour of sending these packets is to tell each neighbouring demon that it’s still alive, and sending the entire route table along with it.
Actual output:
What was seen is that packets were sent in a consistent pattern and that the dead timers were updated when there was a packet received.
Conclusion:
Our implementation of the periodic update was working as intended.

Kayle Ransby: 34043590
Sjaak Flick: 36406121

Test: Triggered updates
Expected output:
When a route becomes unavailable, either through its “timeout” timer or a received packet. Or a better route (with a smaller metric) is found. A triggered update should be sent to all neighbouring routers informing them about the route change.
Actual output:
Whenever a triggered update occurs, all routers continuously send packets to each other and never return to the periodic update interval (the terminals just become a blur of updates)
Conclusion:
There is an error in the way our triggered update is implemented. Triggered updates were implemented just after split horizon with poisoned reverse, meaning there was an error with the metric of 16 (infinity) being sent between neighbours.

Test: timeout and garbage collection timers
Expected output:
When an update packet is received, the “timeout” timer is reset to (180 / division_factor) seconds. When an update packet has not been received in the last (180 / division_factor) seconds, the garbage collection timer should start. Once that timer goes down to 0, the route should be deleted from the table.
Actual output:
The behaviour observed was as expected.
Conclusion:
This aspect of our design was working as designed.

Test: Route entries.
Expected output:

- When a route is added to the table, each entry should display the destination ID, the metric, it’s outgoing interface and the timers associated with the route.
- If a better route is found (with a lower metric), it should replace the current route in the table and send a triggered update to its neighbours.

Actual output:
Initially, the router was learning each of the routes the other had via the periodic updates, but there was a design issue relating to how the routes were being stored. They were being stored in a dictionary with a (routerID, port) tuple as the key. An unexpected result of this is that there was more than 1 route for each destination router in the route table.
This was dealt with by changing the key to just the destination, and the value to be a (metric, port) tuple. Once this was done, the routers were adding and changing values to the table as expected.
Conclusion:
The initial design of the route table was error prone, but when the issue was found and dealt with, the tables were working as expected.

Kayle Ransby: 34043590
Sjaak Flick: 36406121

Example configuration file:

Configuration file for router 4 in the example network:

router-id 4
input-ports 1403, 1405, 1407
outputs 1304-4-3, 1504-2-5, 1704-6-7

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: Kayle Ransby

Student ID: 34043590

Signature: 

Date: 03/05/2020

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: Siyuan Qiu

Student ID: 36406121

Signature: 

Date: 08-5-2020