

# **COSC364**

## **Assignment 1**

### **RIP Routing**

Tristan Christie  
Student ID: 16801546

Clinton Walker  
Student ID: 55356003

Contribution per person:

Tristan Christie-50%

Tristan contributed to the project by creating the config file reader script, functions that create update packets, some of the main function, triggered updates, the report, and tests

Clinton Walker-50%

Clinton contributed to the project by creating functions that bind given port numbers to sockets, most of the main function, the routing table and routing table updates, and functions to process incoming packets

Some of the aspects of our overall program that were particularly well done include the config file reader which allows for a wide range of comments in the config file, the Packet creating functions which is in big endian and follows the format given in the RIP specification document.

Some of the aspects of our overall program that we feel could have been improved were the planning of the code, and the code could have had better encapsulation so that the functions would be split into separate files based on the actions they perform.

We have ensured atomicity of event processing by using the select function call to wait for packets, instead of entering a loop until data arrives. There is a similar situation for everything else in the main function, where no waiting, e.g. for timers, is done inside the main loop so when a condition is met, the associated code is completed instantly, which allows for all operations to be isolated from each other.

Some of the weaknesses of the RIP routing protocol we have identified are that the network cannot support more than 15 hops between routers which means this implementation would not be able to support large networks, when a route is removed many triggered updates are sent quickly to ensure consistency between routers which causes a lot of traffic, and it takes a while for the timeout on a router to trigger so traffic could still be sent through a dead router for a while. Count to infinity is still a problem but it is not as drastic anymore as the maximum metric is 15, but this still takes a while because the current implementation would rely on periodic updates for this as triggered updates are reserved for when a route is invalid.

We performed tests on our code to ensure it worked. One of the tests was to make sure the configuration files would check for errors and raise an exception when there was an error. We tested this by adding a few comment lines, blank lines, and incorrectly formatted lines to a config file while keeping all the correct files needed to complete the configuration. The expected outcome was that the files would still load correctly and all the lines that did not match the correct formatting would be treated as comments. After printing the variables that were stored after reading the config files, we concluded that the actual outcome was the same as the expected outcome. We also tested the configuration files by removing lines

necessary to set up a router. The expected outcome was that whenever a line needed for setting up a router was missing, the program would raise an error specifying what was missing. This was the actual outcome of the tests with the exception being timer values which default to 30-180-120 when not specified or incorrectly specified, a line is also printed stating the timer values were set to default. Through these tests, an acceptable format was made to allow for comments in configuration files.

To test the routing table format, we set a routers routing table to have route entries of it's neighbours designated in the config file, as we had not implemented the distance vector algorithm yet. The expected result of this was for the routing table to have entries for each neighbour that were consistent with the format of the printed routing table. The actual outcome was consistent with the expected outcome

To test the packet format and the sending of packets, we set up two config files to have each other as neighbours as well as a third router, ran them with the RIP script, and printed the messages they received. The expected outcome was for the received packets to be sent in the correct format, which is specified in the RIP document, and for the routing entry of the third router to have the metric specified in the config file while the other routing entry has a metric of 16 as the receiving router is the next hop for this entry and 16 is treated as infinite in this implementation of the RIP protocol. The actual outcome was consistent with the expected outcome.

We also performed tests on the implementation of the routing protocol and the routing table display. We did this by creating a group of config files that would create a network. We then ran the RIP script with these files and compared the output of the table with the expected output. After confirming the outputs were the same, we then shut down one of the routers to see if the router tables would converge to the expected routing tables. Once the tables had converged to the correct values, we then turned back on the router that was shut down, and checked the tables once again converged to the original values, which they did.

Example configuration file for the example network of Figure 1:

router-id 4

input-ports 1028 1031 1038

outputs 1029-4-3 1030-2-5 1039-6-7

timer-values 30-180-120

C:\Users\uberT\Desktop\cosc364-rip-assignment\Config\_file\_reader.py

```
import sys

def readConfig(filename):
    file = open(filename, "r")
    lines = file.readlines()
    #stripping the newline character from each line
    for i in range(len(lines)-1): lines[i] = lines[i].strip()

    #most of the get functions return -1 when there is an error
    routerId = getRouterId(lines)
    if(routerId == -1):
        raise Exception("Router ID not specified or specified incorrectly")

    inputPorts = getInputPorts(lines)
    if(inputPorts == -1):
        raise Exception("Input ports not specified or specified incorrectly")

    outputPorts = getOutputPorts(lines, inputPorts)
    if(outputPorts == -1):
        raise Exception("Output ports not specified or specified incorrectly")

    #no error checking on timer values as it defaults to 30-180-120
    timerValues = getTimerValues(lines)

    file.close()
    return (routerId, inputPorts, outputPorts, timerValues)

def getRouterId(lines):
    #iterate through all lines for a valid parameter line
    for line in lines:
        try:
            line = line.split()
            #check if line is a valid router id then return the router id
            if(line[0] == "router-id" and 1 <= int(line[1]) <= 64000 and len(line) == 2):
                return int(line[1])
        except:
            #line treated as a comment and ignored
            buffer = 0
    #no properly formatted line found, return -1 as indication of error
    return -1

def getInputPorts(lines):
    #iterate through all lines for a valid parameter line
    for line in lines:
        try:
            line = line.split()
            #check if line is indicated as an input port parameter then check the ports are valid and add them to the input ports list
            if(line[0] == "input-ports"):
                inputPortsEstablished = True
```

C:\Users\uberT\Desktop\cosc364-rip-assignment\Config\_file\_reader.py

```
        inputPorts = []
        for i in line[1:]:
            if(1024 <= int(i) <= 64000 and int(i) not in inputPorts):
                inputPorts.append(int(i))
            else:
                inputPortsEstablished = False
                #if all the input ports are valid, return the input ports list
                if inputPortsEstablished: return inputPorts
    except:
        #line treated as a comment and ignored
        buffer = 0
    #no properly formatted line found, return -1 as indication of error
    return -1

def getOutputPorts(lines, inputPorts):
    #iterate through all lines for a valid parameter line
    for line in lines:
        try:
            line = line.split()
            #check if line is indicated as an output port parameter then check if
            the ports are valid and add them to the output ports list
            if(line[0] == "outputs"):
                outputPortsEstablished = True
                outputPorts = []
                for i in line[1:]:
                    outputPort = []
                    i = i.split("-")
                    if(len(i) == 3 and int(i[0]) not in inputPorts and 1024 <= int(i[0]) <= 64000 and 1 <= int(i[1]) <= 15 and 1 <= int(i[2]) <= 64000):
                        #have to iterate through outputPorts to check i isn't in it since it is a 2D list
                        for j in outputPorts:
                            if(int(i[0]) == j[0]):
                                outputPortsEstablished = False
                                outputPort.append(int(i[0]))
                                outputPort.append(int(i[1]))
                                outputPort.append(int(i[2]))
                                outputPorts.append(outputPort)
                        else:
                            outputPortsEstablished = False
                            #if all the output ports are valid, return the output ports list
                            if outputPortsEstablished: return outputPorts
        except:
            #line treated as a comment and ignored
            buffer = 0
    #no properly formatted line found, return -1 as indication of error
    return -1

def getTimerValues(lines):
    #iterate through all lines for a valid parameter line
```

C:\Users\uberT\Desktop\cosc364-rip-assignment\Config\_file\_reader.py

```
    for line in lines:
        try:
            line = line.split()
            #check if line is indicated as a timer values parameter then return the
            #the timer values list if the timer values are valid
            if(line[0] == "timer-values"):
                timerValues = line[1].split("-")
                timerValues[0] = int(timerValues[0])
                timerValues[1] = int(timerValues[1])
                timerValues[2] = int(timerValues[2])
                #check timer ratios are correct
                if(timerValues[1] / timerValues[0] == 6 and timerValues[2] /
                timerValues[0] == 4 and len(timerValues) == 3):
                    return timerValues
            except:
                #line treated as a comment and ignored
                buffer = 0
        #no properly formatted line found, return the default values
        print("Timer values not specified or specified incorrectly, setting to default")
        return [30, 180, 120]

def main():

    routerId, inputPorts, outputPorts, timerValues = readConfig(sys.argv[1])
    print("Router ID:", routerId)
    print("Input Ports:", end=" ")
    for i in inputPorts: print(i, end=", ")
    print()
    print("Output Ports:", end=" ")
    for i in outputPorts: print(i, end=", ")
    print()
    print("Timer Values:", timerValues)

    inputSockets = createSockets(inputPorts)
    updateTimer, timeoutTimer, garbageTimer = timerValues
    incomingQueue = []

    while True:
        read, write, special = select.select(inputSockets, [], [])

        for i in read:
            try:
                addr, data = i.recvfrom(BUFFER_SIZE)
                incomingQueue.append((addr, data))
                print(incomingQueue.pop(0))
            except:
                print("Error recieving a packet")

        if need_periodic_update(next_update, updateTimer):
            # send update
```

C:\Users\uberT\Desktop\cosc364-rip-assignment\Config\_file\_reader.py

```
print("periodic update time")
```

```
if __name__ == '__main__':  
    main()
```



C:\Users\uberT\Desktop\cosc364-rip-assignment\rip.py

```
import sys
import socket
import select
import random
import time
from copy import deepcopy
from Config_file_reader import *

BUFFER_SIZE = 1024

routingTable = {}

def createSockets(input_ports):

    input_sockets = []

    for port in input_ports:
        try:
            # Bind a socket for each port on localhost
            UDP_socket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
            UDP_socket.bind(("localhost", port))
            input_sockets.append(UDP_socket)
        except:
            print("Error starting socket on port " + str(port))
    return input_sockets

def resetUpdateTimer(updateTimer):
    return time.time() + updateTimer

def checkPeriodicUpdate(nextUpdate):
    # Predicate to see if it is time for a periodic update.
    if time.time() >= nextUpdate:
        return True
    else:
        return False

def resetTimeout(routerId, timeoutTimer):
    routingTable[routerId]["timeout"] = time.time() + timeoutTimer

def checkTimeout(routerId, timeoutTimer, garbageTimer):
    # if the garbage flag is true
    if routingTable[routerId]["garbage"]:
        # if timer has gone off
        if time.time() >= routingTable[routerId]["timeout"]:
            # delete the entry
            del(routingTable[routerId])
    else:
        # timer has gone off
```

C:\Users\uberT\Desktop\cosc364-rip-assignment\rip.py

```
        if time.time() >= routingTable[routerId]["timeout"]:
            setGarbage(routerId, garbageTimer)

def createTableEntry(routerId, metric, nextHopId, nextHop, timeoutTimer):

    data = {
        "metric" : metric,
        "nextHop": nextHop,
        "nextHopId": nextHopId,
        "timeout" : time.time() + timeoutTimer,
        "garbage" : False,
        "infiniteRouteFlag" : False
    }

    routingTable[routerId] = data

def printTable():
    print( "  ↵
↵  +-----+-----+-----+-----+-----+-----+-----↵
↵  -----+↵",
        "| Destination | Metric | Next Hop | Next Hop ID | Timeout | Garbage ↵
↵  | Infinite Route Flag |↵",
        ↵
↵  "+-----+-----+-----+-----+-----+-----+-----↵
↵  -----+")

    copiedTable = deepcopy(routingTable)
    for link in routingTable:
        data = routingTable[link]
        timeout = int(data["timeout"] - time.time())
        if data["garbage"]:
            garbage = timeout
            timeout = '-'
        else:
            garbage = '-'
        if not data["infiniteRouteFlag"]:
            infiniteRouteFlag = "Not Set"
        else:
            infiniteRouteFlag = "Set"
        print(" |{0:^13}|{1:^8}|{2:^10}|{3:^13}|{4:^9}|{5:^9}|{6:^21}|".format(↵
↵  link, data["metric"], data["nextHop"], data["nextHopId"], timeout, garbage↵
↵  , infiniteRouteFlag))
    print("  ↵
↵  +-----+-----+-----+-----+-----+-----+-----↵
↵  -----+")

def createUpdatePacket(senderId, destId, routingTable, triggeredUpdate=False):
    packet = bytearray()
    COMMAND = 2
    VERSION = 2
```

C:\Users\uberT\Desktop\cosc364-rip-assignment\rip.py

```
    senderIdBytes = senderId.to_bytes(2, 'big')
    packet.append(COMMAND.to_bytes(1, 'big')[0])
    packet.append(VERSION.to_bytes(1, 'big')[0])
    for i in senderIdBytes: packet.append(i)
    for i in routingTable:
        if not triggeredUpdate:
            createRouteEntry(routingTable, packet, destId, i)
        else:
            if(routingTable[i]["infiniteRouteFlag"]):
                createRouteEntry(routingTable, packet, destId, i)
    return packet

def createRouteEntry(routingTable, packet, destId, routeEntry):
    AFI = 2
    AFIBytes = AFI.to_bytes(2, 'big')
    for i in AFIBytes: packet.append(i)
    for i in range(0, 2): packet.append(0x00)
    routeDestBytes = routeEntry.to_bytes(4, 'big')
    for i in routeDestBytes: packet.append(i)
    for i in range(0, 8): packet.append(0x00)
    if(routingTable[routeEntry]["nextHopId"] == destId):
        metric = 16
        metricBytes = metric.to_bytes(4, 'big')
    else:
        metricBytes = routingTable[routeEntry]["metric"].to_bytes(4, 'big')
    for i in metricBytes: packet.append(i)

def processPacket(senderPort, packet, timeoutTimer, garbageTimer, outputPorts, currentRouterId):
    # discard invalid packets
    if not checkPacket(packet):
        print("discarded an invalid packet")
        return
    entryCount = int((len(packet) - 4) / 20)
    entries = packet[4:]

    senderId = bytearray()
    senderId.append(packet[2])
    senderId.append(packet[3])
    senderId = int.from_bytes(senderId, "big")

    for neighbourPort, neighbourMetric, neighbourRouterId in outputPorts:
        if(senderId == neighbourRouterId):
            senderMetric = neighbourMetric

    if senderId not in routingTable.keys():
        createTableEntry(senderId, senderMetric, senderId, senderPort, timeoutTimer)
    else:
        for neighbourPort, neighbourMetric, neighbourRouterId in outputPorts:
            if senderId == neighbourRouterId and neighbourMetric < routingTable[neighbourRouterId]["metric"]:
```

C:\Users\uberT\Desktop\cosc364-rip-assignment\rip.py

```
5         senderId["metric"]:  
            routingTable[senderId]["metric"] = neighbourMetric  
            routingTable[senderId]["nextHopId"] = neighbourRouterId  
            routingTable[senderId]["nextHop"] = neighbourPort  
        routingTable[senderId]["garbage"] == False  
        resetTimeout(senderId, timeoutTimer)  
  
    for i in range(entryCount):  
        destination = bytearray()  
        metric = bytearray()  
        for j in range(4, 8):  
            destination.append(entries[20 * i + j])  
        for j in range(16, 20):  
            metric.append(entries[20 * i + j])  
  
        destination = int.from_bytes(destination, "big")  
        metric = int.from_bytes(metric, "big")  
  
        distance = senderMetric + metric  
  
        if destination in routingTable.keys():  
  
            if distance < routingTable[destination]["metric"] or routingTable[destination]["nextHopId"] == senderId:  
5                routingTable[destination]["infiniteRouteFlag"] = False  
                routingTable[destination]["metric"] = distance  
                routingTable[destination]["nextHopId"] = senderId  
                routingTable[destination]["nextHop"] = senderPort  
  
            if(distance < 16 and routingTable[destination]["nextHopId"] == senderId):  
5                resetTimeout(destination, timeoutTimer)  
                routingTable[destination]["garbage"] = False  
  
            elif(destination != currentRouterId and not distance > 15):  
5                createTableEntry(destination, distance, senderId, senderPort, timeoutTimer)  
  
            try:  
                if routingTable[destination]["metric"] > 15 and routingTable[destination]["garbage"] == False:  
5                    setGarbage(destination, garbageTimer)  
                    routingTable[destination]["infiniteRouteFlag"] = True  
            except:  
                #doesn't work if destination == currentRouterId as it won't have a routing entry  
5                routing entry  
                buffer = 0  
  
    def checkPacket(packet):  
        entryCount = int((len(packet) - 4) / 20)
```

C:\Users\uberT\Desktop\cosc364-rip-assignment\rip.py

```
    if packet[0] != 2 and packet[1] != 2:
        return False
    entries = packet[4:]
    for i in range(entryCount):
        metric = bytearray()
        for j in range(16, 20):
            metric.append(entries[20 * i + j])
        metric = int.from_bytes(metric, "big")
        if metric < 1 or metric > 16:
            return False

    return True

def setGarbage(routerId, garbageTimer):
    routingTable[routerId]["garbage"] = True
    routingTable[routerId]["timeout"] = time.time() + garbageTimer
    routingTable[routerId]["metric"] = 16

def main():

    routerId, inputPorts, outputPorts, timerValues = readConfig(sys.argv[1])
    """
    print("Input Ports:", end=" ")
    for i in inputPorts: print(i, end=" ",)
    print()
    print("Output Ports:", end=" ")
    for i in outputPorts: print(i, end=" ",)
    print()
    print("Timer Values:", timerValues)
    """

    inputSockets = createSockets(inputPorts)
    outputSocket = [inputSockets[0]]
    updateTimer, timeoutTimer, garbageTimer = timerValues
    nextUpdate = resetUpdateTimer(updateTimer)
    incomingQueue = []

    while True:
        read, write, special = select.select(inputSockets, outputSocket, [])

        for i in read:
            try:
                data, addr = i.recvfrom(BUFFER_SIZE)
                incomingQueue.append((data, addr))
            except:
                # print("Error recieving a packet")
                buffer = 0

        if checkPeriodicUpdate(nextUpdate):
```

C:\Users\uberT\Desktop\cosc364-rip-assignment\rip.py

```
# send update
nextUpdate = resetUpdateTimer(updateTimer)
print(" Router ID:", routerId)
printTable()
j = 0
for i in outputPorts:
    bytesToSend = createUpdatePacket(routerId, i[2], routingTable)
    inputSockets[j].sendto(bytesToSend, ('localhost', i[0]))
    j += 1

for message in incomingQueue:
    data, addr = incomingQueue.pop(0)
    port = addr[1]
    processPacket(port, data, timeoutTimer, garbageTimer, outputPorts, 2,
routerId)

sendTriggeredUpdate = False
for i in routingTable:
    if(routingTable[i]["infiniteRouteFlag"] == True):
        sendTriggeredUpdate = True
if(sendTriggeredUpdate):
    j = 0
    for i in outputPorts:
        bytesToSend = createUpdatePacket(routerId, i[2], routingTable, 2,
sendTriggeredUpdate)
        inputSockets[j].sendto(bytesToSend, ('localhost', i[0]))
        j += 1
    for i in routingTable: routingTable[i]["infiniteRouteFlag"] = False

for entry in deepcopy(routingTable):
    checkTimeout(entry, timeoutTimer, garbageTimer)

if __name__ == '__main__':
    main()
```

# 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:

Tristan Christie

Student ID:

16801546

Signature:

T Christie

Date:

27/04/2021



# 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.

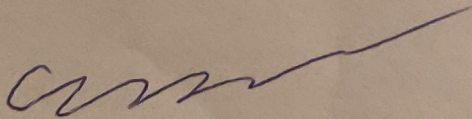
Name:

Clinton Walker

Student ID:

55356003

Signature:



Date:

27/4/21