NWEN302

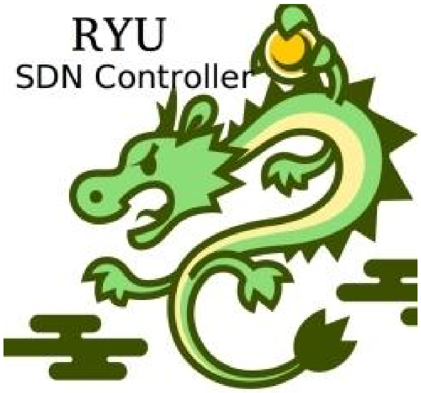
Lab 3

Olivia Fletcher
300534281
fletcholiv

REPORT

Introduction

In this investigation we will be utilizing a tool known as Software-Defined Networking (SDN) where its prime functionality is using software-based controllers or API's to ensure a more efficient network performance. The use of SDN resembles cloud commuting as it enables dynamic configuration and monitoring duties unlike a traditional way of network management.

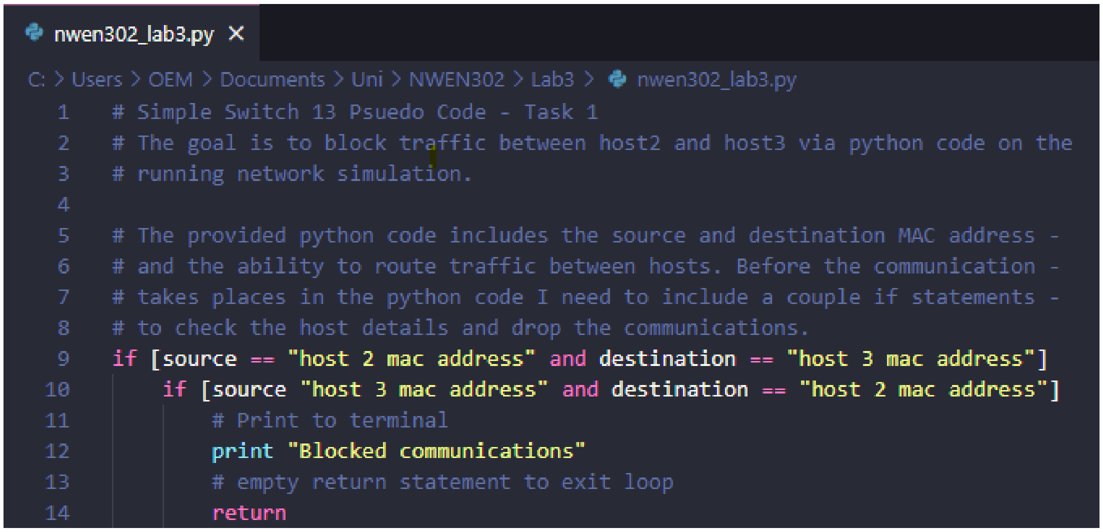


Design

We will be utilizing Python and implementing network simulation through CORE gui. In this investigation we will be learning the concepts behind SDN networks (Software-Defined Networking) and its available operations such as Ryu and OpenFlow. The Ryu will be acting as a controller to the CORE network simulation. Ryu will be extending the use of OpenFlow as a means to implement the Python code to orchestrate the network simulation.
We will be adding to the provided simple_switch13.py code to complete task1 ;

- Block traffic between host 2 and host 3

Due to the source and destination mac addresses being provided I just need to do a simple if statement that checks if the source and/or destination is host2 and/or host 3 and blocks the communications.

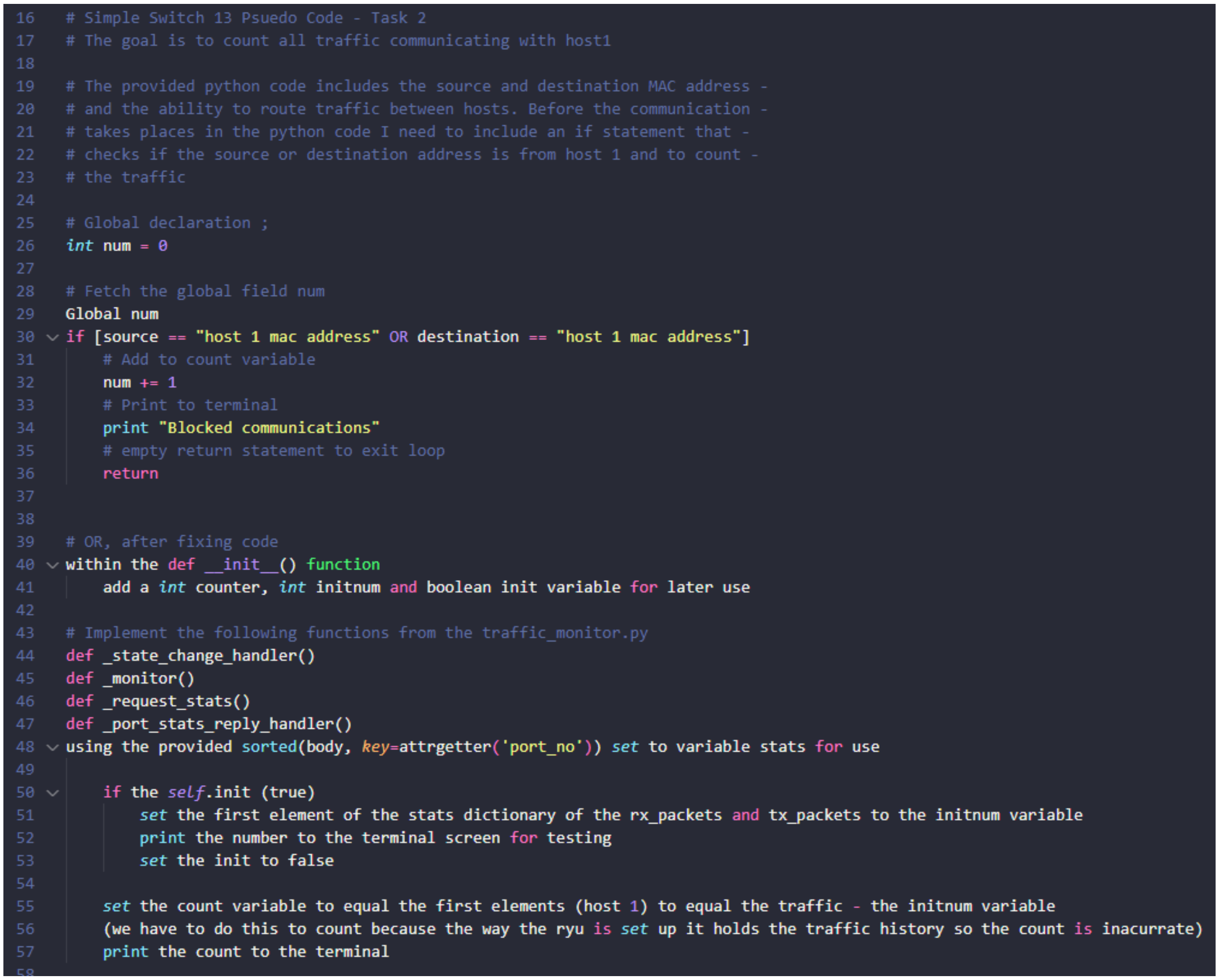

nwen302_lab3.py ×
C: > Users > OEM > Documents > Uni > NWEN302 > Lab3 > nwen302_lab3.py

```
# Simple Switch 13 Psuedo Code - Task 1
# The goal is to block traffic between host2 and host3 via python code on the
# running network simulation.

# The provided python code includes the source and destination MAC address -
# and the ability to route traffic between hosts. Before the communication -
# takes places in the python code I need to include a couple if statements -
# to check the host details and drop the communications.
if [source == "host 2 mac address" and destination == "host 3 mac address"]
    if [source "host 3 mac address" and destination == "host 2 mac address"]
        # Print to terminal
        print "Blocked communications"
        # empty return statement to exit loop
        return
```

To complete task 2;

- Count all traffic communicating with host 1

The first block follows similar to task 1 where the second block of code is after making changes when I encountered some errors/issues (I go into detail later on)

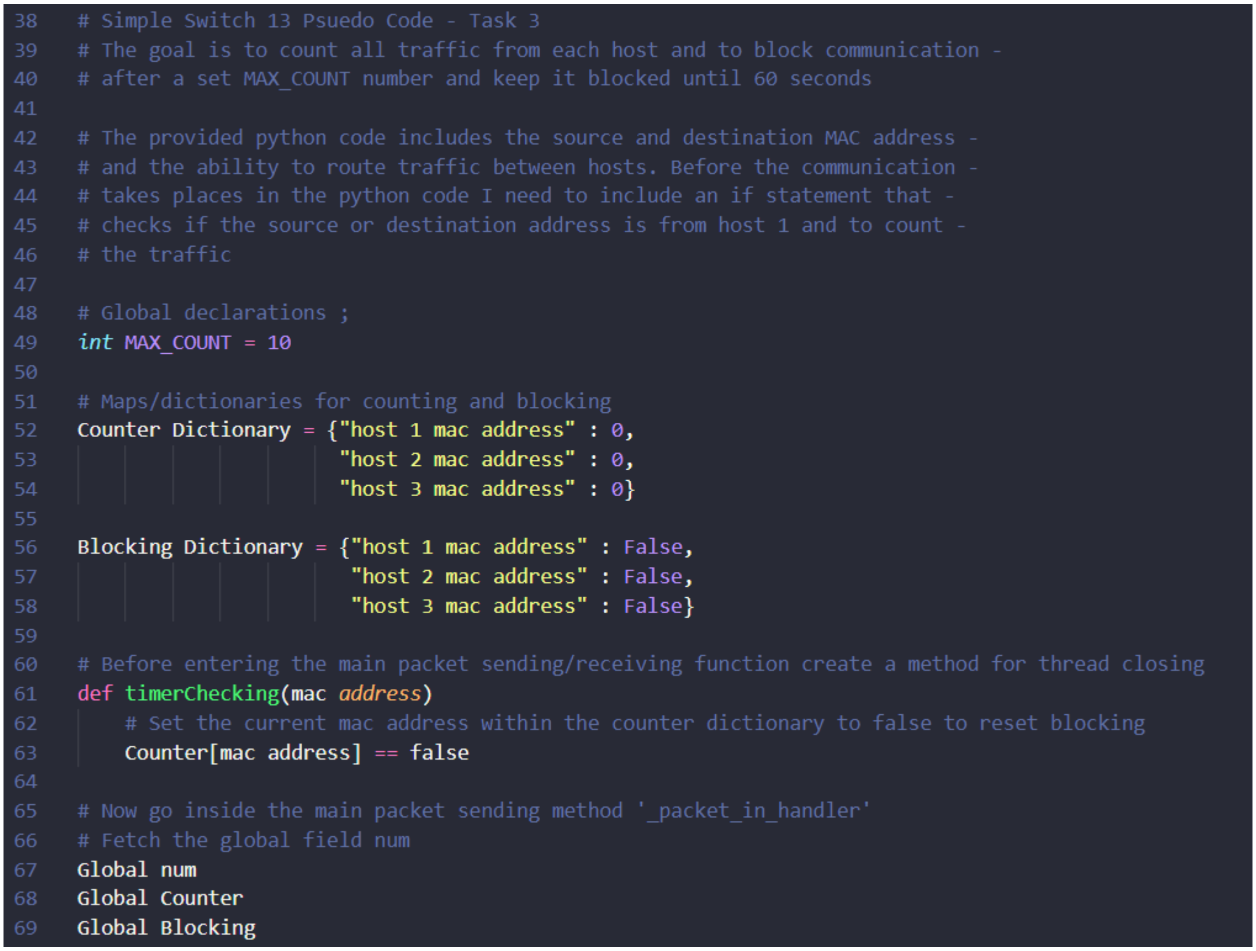


```
# Simple Switch 13 Psuedo Code - Task 2
# The goal is to count all traffic communicating with host1

# The provided python code includes the source and destination MAC address -
# and the ability to route traffic between hosts. Before the communication -
# takes places in the python code I need to include an if statement that -
# checks if the source or destination address is from host 1 and to count -
# the traffic

# Global declaration ;
int num = 0

# Fetch the global field num
Global num
if [source == "host 1 mac address" OR destination == "host 1 mac address"]
    # Add to count variable
    num += 1
    # Print to terminal
    print "Blocked communications"
    # empty return statement to exit loop
    return


# OR, after fixing code
within the def __init__() function
    add a int counter, int initnum and boolean init variable for later use

# Implement the following functions from the traffic_monitor.py
def _state_change_handler()
def _monitor()
def _request_stats()
def _port_stats_reply_handler()
using the provided sorted(body, key=attrgetter('port_no')) set to variable stats for use

    if the self.init (true)
        set the first element of the stats dictionary of the rx_packets and tx_packets to the initnum variable
        print the number to the terminal screen for testing
        set the init to false

    set the count variable to equal the first elements (host 1) to equal the traffic - the initnum variable
    (we have to do this to count because the way the ryu is set up it holds the traffic history so the count is inacurrate)
    print the count to the terminal
```

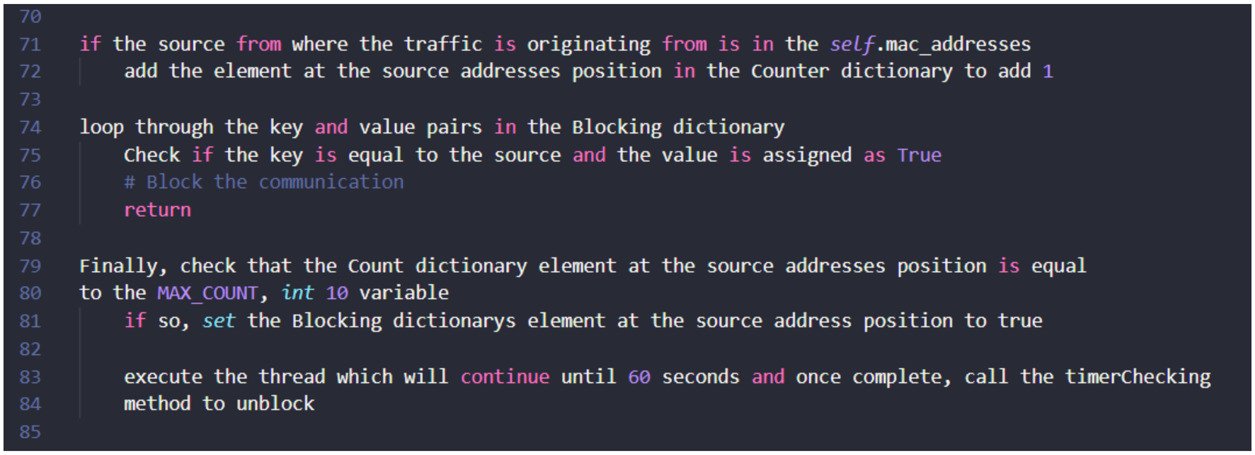
To complete task 3;

- Count traffic communications from each host and once the number has hit a MAX_COUNT to block communications from that host for 60 seconds.

This task we will be combining task 1 and task 2 by implementing a blocking algorithm after a given host's communications count has hit a maximum amount. The below 2 images are for my first iteration of my attempt for task 3.

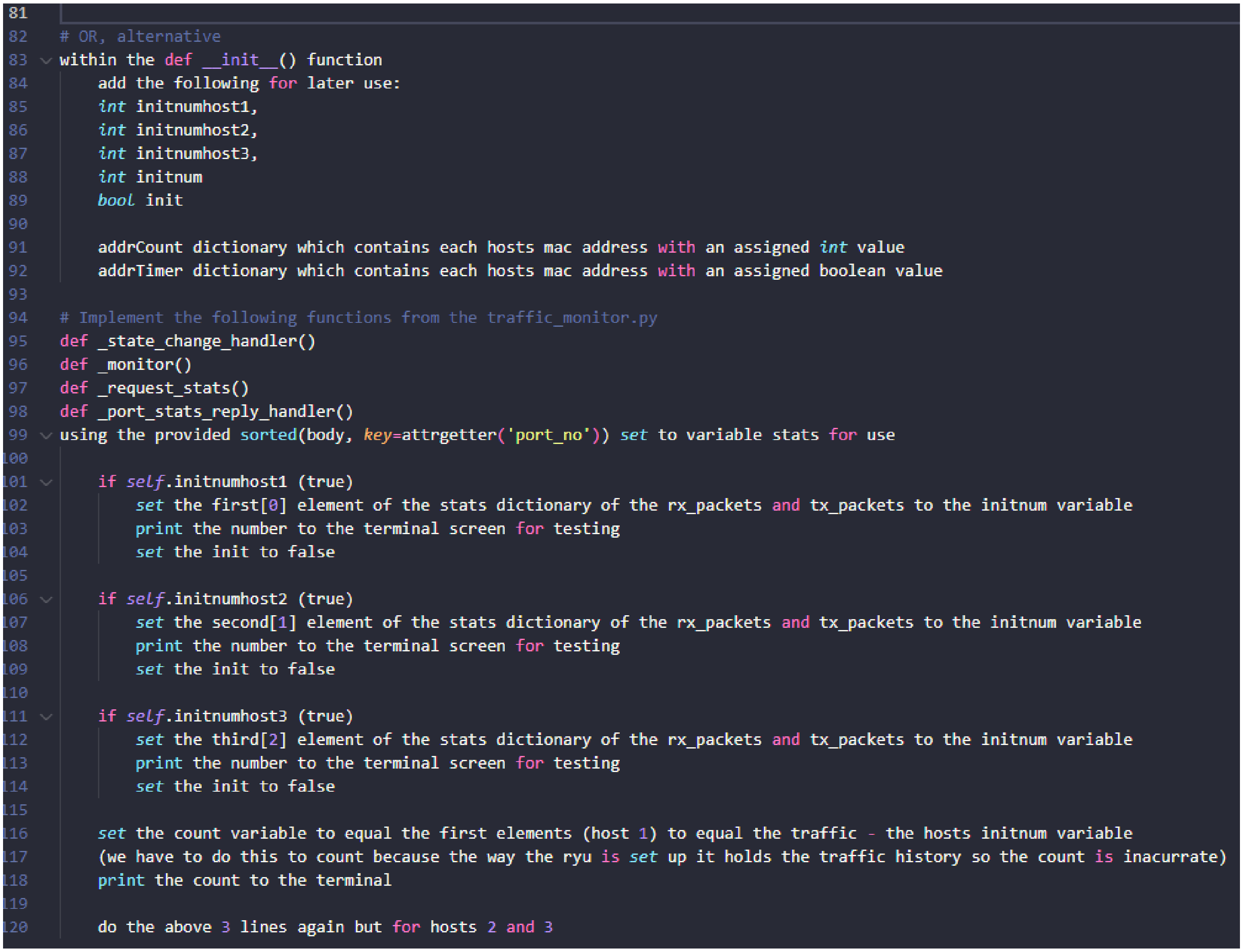


```
# Simple Switch 13 Psuedo Code - Task 3
# The goal is to count all traffic from each host and to block communication -
# after a set MAX_COUNT number and keep it blocked until 60 seconds

# The provided python code includes the source and destination MAC address -
# and the ability to route traffic between hosts. Before the communication -
# takes places in the python code I need to include an if statement that -
# checks if the source or destination address is from host 1 and to count -
# the traffic

# Global declarations ;
int MAX_COUNT = 10

# Maps/dictionaries for counting and blocking
Counter Dictionary = {"host 1 mac address" : 0,
                      "host 2 mac address" : 0,
                      "host 3 mac address" : 0}

Blocking Dictionary = {"host 1 mac address" : False,
                       "host 2 mac address" : False,
                       "host 3 mac address" : False}

# Before entering the main packet sending/receiving function create a method for thread closing
def timerChecking(mac address)
    # Set the current mac address within the counter dictionary to false to reset blocking
    Counter[mac address] == false

# Now go inside the main packet sending method '_packet_in_handler'
# Fetch the global field num
Global num
Global Counter
Global Blocking
```



```

if the source from where the traffic is originating from is in the self.mac_addresses
    add the element at the source addresses position in the Counter dictionary to add 1

loop through the key and value pairs in the Blocking dictionary
    Check if the key is equal to the source and the value is assigned as True
    # Block the communication
    return

Finally, check that the Count dictionary element at the source addresses position is equal
to the MAX_COUNT, int 10 variable
    if so, set the Blocking dictionarys element at the source address position to true

    execute the thread which will continue until 60 seconds and once complete, call the timerChecking
    method to unblock

```

- My second attempt at task 3 after making some changes; (I discuss why later on)

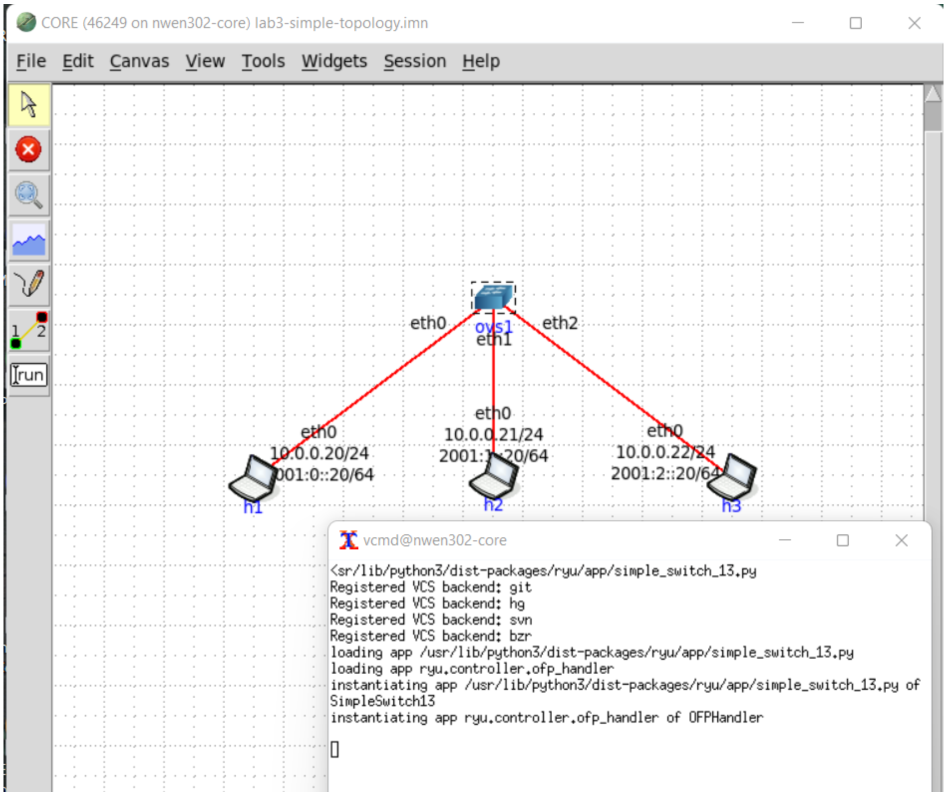


```

# OR, alternative
within the def __init__() function
    add the following for later use:
    int initnumhost1,
    int initnumhost2,
    int initnumhost3,
    int initnum
    bool init

    addrCount dictionary which contains each hosts mac address with an assigned int value
    addrTimer dictionary which contains each hosts mac address with an assigned boolean value

# Implement the following functions from the traffic_monitor.py
def _state_change_handler()
def _monitor()
def _request_stats()
def _port_stats_reply_handler()
using the provided sorted(body, key=attrgetter('port_no')) set to variable stats for use

    if self.initnumhost1 (true)
        set the first[0] element of the stats dictionary of the rx_packets and tx_packets to the initnum variable
        print the number to the terminal screen for testing
        set the init to false

    if self.initnumhost2 (true)
        set the second[1] element of the stats dictionary of the rx_packets and tx_packets to the initnum variable
        print the number to the terminal screen for testing
        set the init to false

    if self.initnumhost3 (true)
        set the third[2] element of the stats dictionary of the rx_packets and tx_packets to the initnum variable
        print the number to the terminal screen for testing
        set the init to false

    set the count variable to equal the first elements (host 1) to equal the traffic - the hosts initnum variable
    (we have to do this to count because the way the ryu is set up it holds the traffic history so the count is inacurrate)
    print the count to the terminal

    do the above 3 lines again but for hosts 2 and 3
```

Procedures

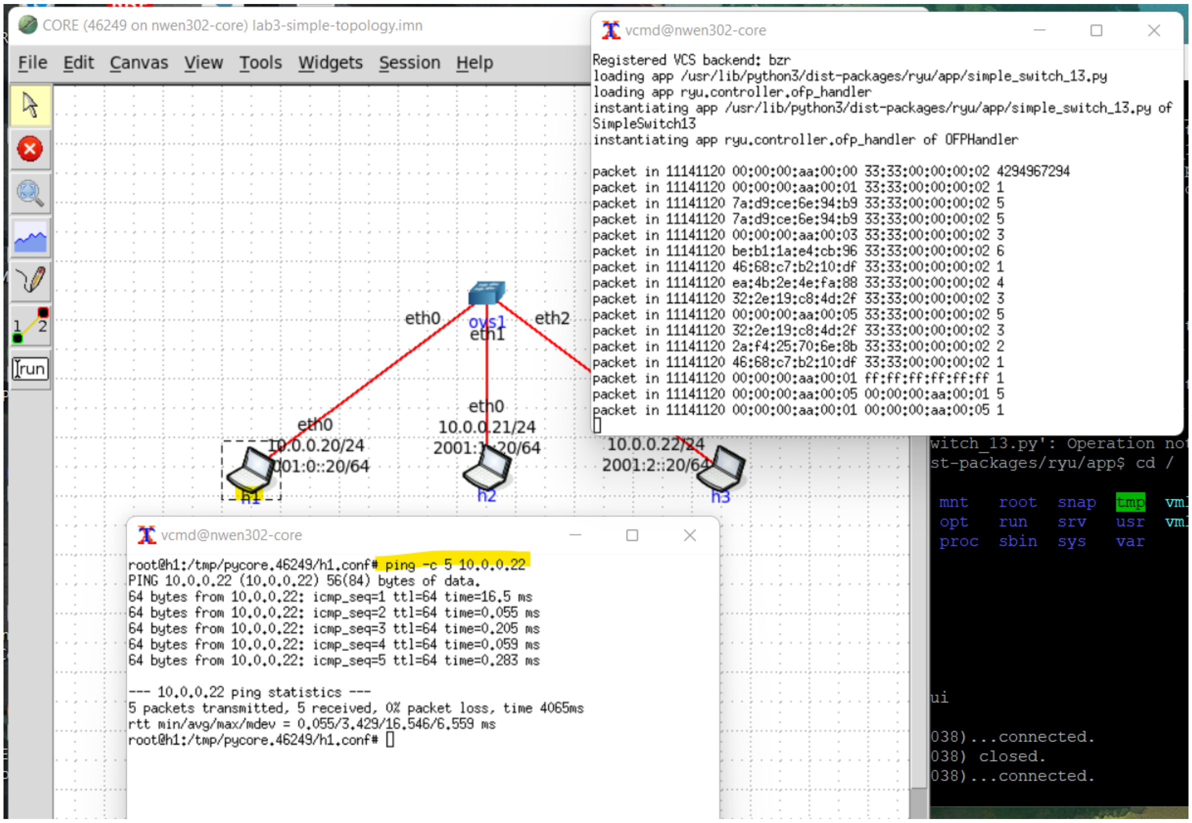
I created a simple network topology which consists of 3 hosts connected to a switch router. The switch is not like a regular layer 2 device but a layer 3 Open vSwitch which will allow for more capabilities such as the use of OpenFlow controller.

Before making any adjustments to the provided python code I have run it through the switches terminal and executed a few commands to test preemptively. Below is a screenshot showing the code running through the switches terminal.



```
<sr/lib/python3/dist-packages/ryu/app/simple_switch_13.py
Registered VCS backend: git
Registered VCS backend: hg
Registered VCS backend: svn
Registered VCS backend: bzr
loading app /usr/lib/python3/dist-packages/ryu/app/simple_switch_13.py
loading app ryu.controller.ofp_handler
instantiating app /usr/lib/python3/dist-packages/ryu/app/simple_switch_13.py of
SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler

▯
```

When pinging from host 1 to host 3 the switches terminal appears as below; showing that it foresees every communication/traffic between hosts.

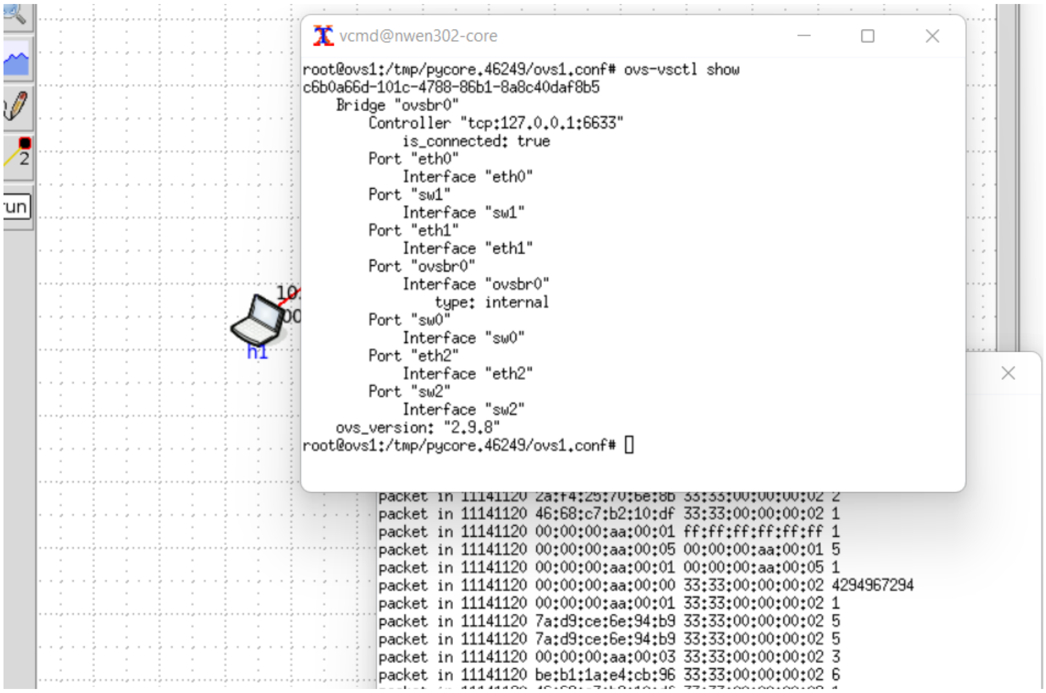


```
Registered VCS backend: bzr
loading app /usr/lib/python3/dist-packages/ryu/app/simple_switch_13.py
loading app ryu.controller.ofp_handler
instantiating app /usr/lib/python3/dist-packages/ryu/app/simple_switch_13.py of
SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler

packet in 11141120 00:00:00:aa:00:00 33:33:00:00:00:02 4294967294
packet in 11141120 00:00:00:aa:00:01 33:33:00:00:00:02 1
packet in 11141120 7a:d9:ce:6e:94:b9 33:33:00:00:00:02 5
packet in 11141120 7a:d9:ce:6e:94:b9 33:33:00:00:00:02 5
packet in 11141120 00:00:00:aa:00:03 33:33:00:00:00:02 3
packet in 11141120 be:b1:1a:e4:cb:96 33:33:00:00:00:02 6
packet in 11141120 46:68:c7:b2:10:df 33:33:00:00:00:02 1
packet in 11141120 ea:4b:2e:4e:fa:88 33:33:00:00:00:02 4
packet in 11141120 32:2e:19:c8:4d:2f 33:33:00:00:00:02 3
packet in 11141120 00:00:00:aa:00:05 33:33:00:00:00:02 5
packet in 11141120 32:2e:19:c8:4d:2f 33:33:00:00:00:02 3
packet in 11141120 2a:f4:25:70:6e:8b 33:33:00:00:00:02 2
packet in 11141120 46:68:c7:b2:10:df 33:33:00:00:00:02 1
packet in 11141120 00:00:00:aa:00:01 ff:ff:ff:ff:ff:ff 1
packet in 11141120 00:00:00:aa:00:05 00:00:00:aa:00:01 5
packet in 11141120 00:00:00:aa:00:01 00:00:00:aa:00:05 1
▯
```

```
witch_13.py': Operation not
st-packages/ryu/app$ cd /

 mnt   root  snap  tmp  vm
 opt   run   srv   usr  vm
 proc  sbin  sys   var

ui

038)...connected.
038) closed.
038)...connected.
```

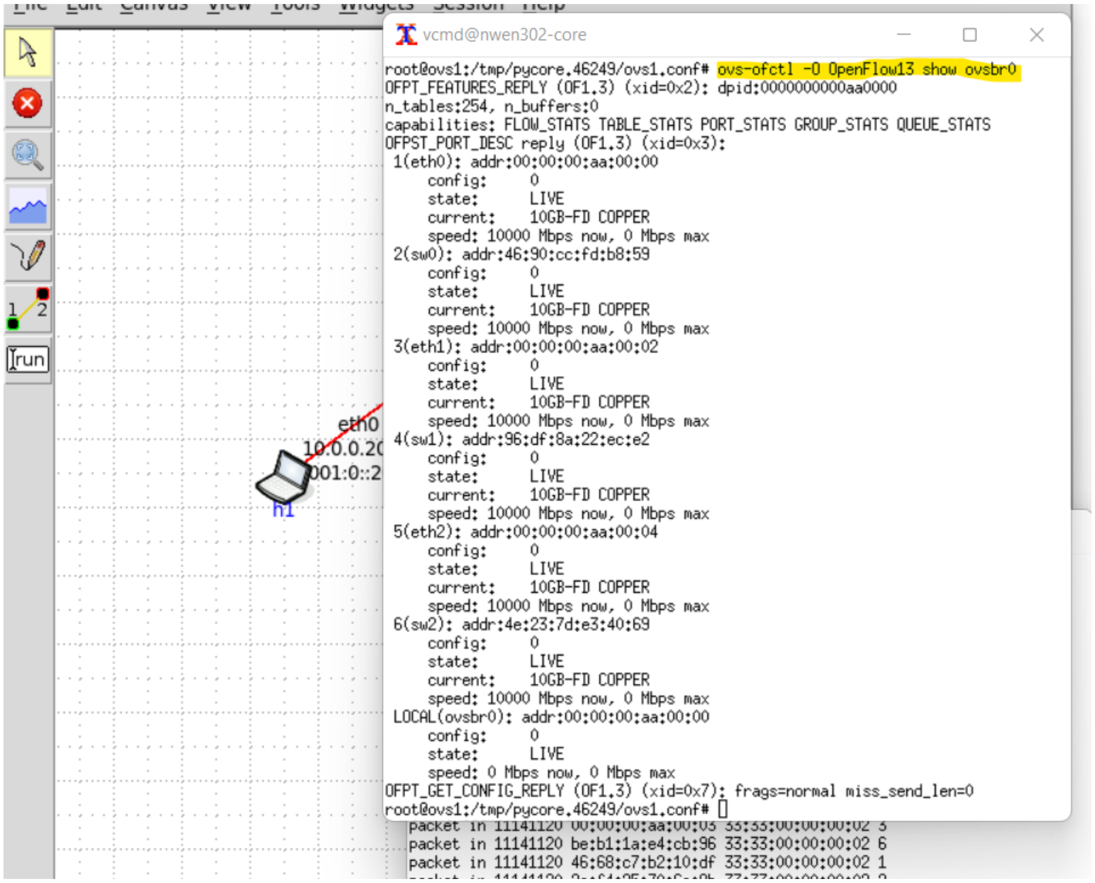
```
root@h1:/tmp/pycore.46249/h1.conf# ping -c 5 10.0.0.22
PING 10.0.0.22 (10.0.0.22) 56(84) bytes of data.
64 bytes from 10.0.0.22: icmp_seq=1 ttl=64 time=16.5 ms
64 bytes from 10.0.0.22: icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from 10.0.0.22: icmp_seq=3 ttl=64 time=0.205 ms
64 bytes from 10.0.0.22: icmp_seq=4 ttl=64 time=0.059 ms
64 bytes from 10.0.0.22: icmp_seq=5 ttl=64 time=0.283 ms

--- 10.0.0.22 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4065ms
rtt min/avg/max/mdev = 0.055/3.429/16.546/6.559 ms
root@h1:/tmp/pycore.46249/h1.conf# ▯
```

The command ‘ovs-vsctl show’ prints an overview of the switch database configuration while ‘ovs-ofctl -O OpenFlow13 show ovsbr0’ shows an overview of the switches OpenFlow configuration.



```
root@ovs1:/tmp/pycore.46249/ovs1.conf# ovs-vsctl show
c6b0a66d-101c-4788-86b1-8a8c40daf8b5
    Bridge "ovsbr0"
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        Port "eth0"
            Interface "eth0"
        Port "sw1"
            Interface "sw1"
        Port "eth1"
            Interface "eth1"
        Port "ovsbr0"
            Interface "ovsbr0"
                type: internal
        Port "sw0"
            Interface "sw0"
        Port "eth2"
            Interface "eth2"
        Port "sw2"
            Interface "sw2"
    ovs_version: "2.9.8"
root@ovs1:/tmp/pycore.46249/ovs1.conf# []
```

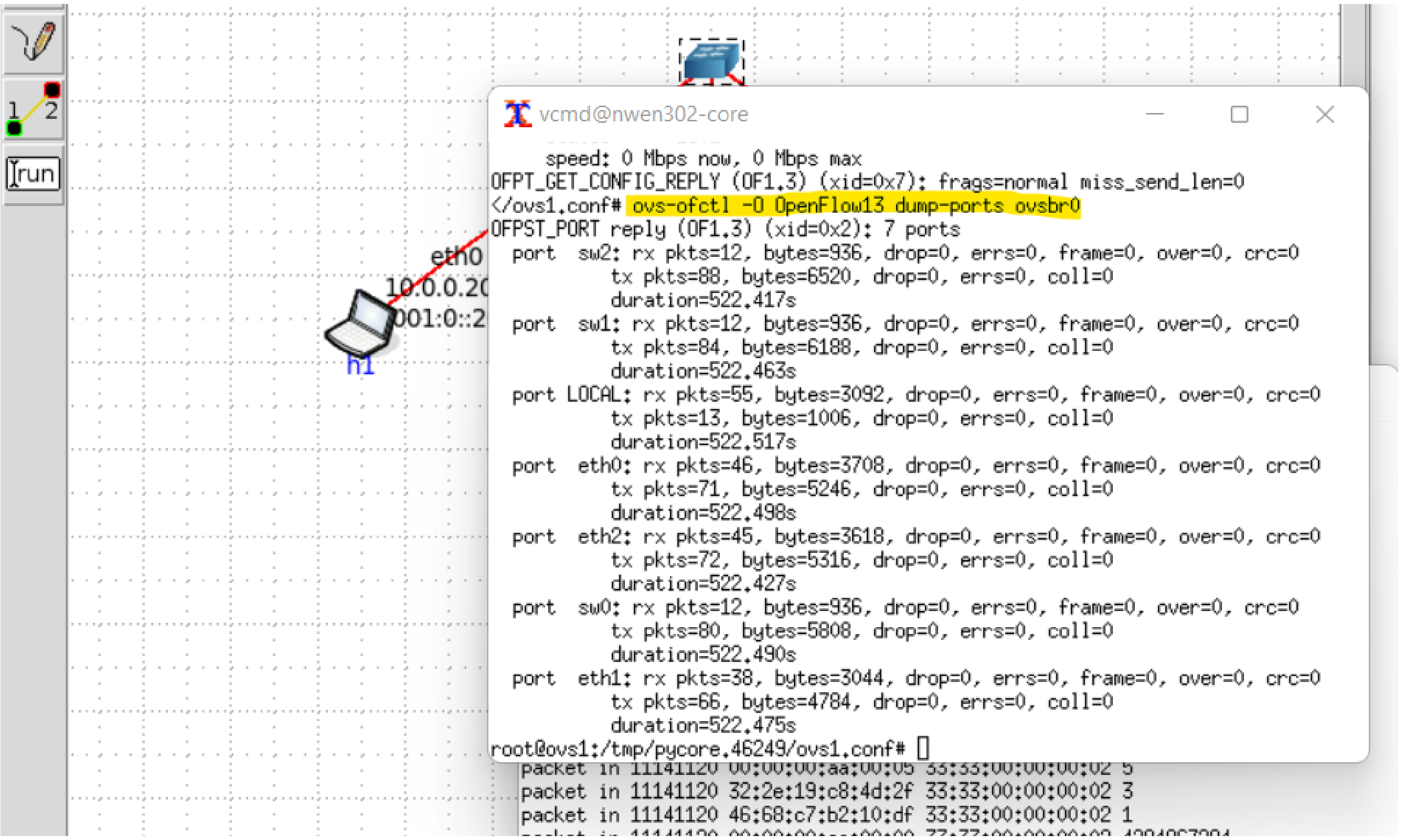
```
packet in 11141120 2a:f4:25:70:6e:8b 33:33:00:00:00:02 2
packet in 11141120 46:68:c7:b2:10:df 33:33:00:00:00:02 1
packet in 11141120 00:00:00:aa:00:01 ff:ff:ff:ff:ff:ff 1
packet in 11141120 00:00:00:aa:00:05 00:00:00:aa:00:01 5
packet in 11141120 00:00:00:aa:00:01 00:00:00:aa:00:05 1
packet in 11141120 00:00:00:aa:00:00 33:33:00:00:00:02 4294967294
packet in 11141120 00:00:00:aa:00:01 33:33:00:00:00:02 1
packet in 11141120 7a:d9:ce:6e:94:b9 33:33:00:00:00:02 5
packet in 11141120 7a:d9:ce:6e:94:b9 33:33:00:00:00:02 5
packet in 11141120 00:00:00:aa:00:03 33:33:00:00:00:02 3
packet in 11141120 be:b1:1a:e4:cb:96 33:33:00:00:00:02 6
```



```
root@ovs1:/tmp/pycore.46249/ovs1.conf# ovs-ofctl -O OpenFlow13 show ovsbr0
OFPT_FEATURES_REPLY (OF1.3) (xid=0x2): dpid:0000000000aa0000
n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS QUEUE_STATS
OFPST_PORT_DESC reply (OF1.3) (xid=0x3):
 1(eth0): addr:00:00:00:aa:00:00
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 2(sw0): addr:46:90:cc:fd:b8:59
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 3(eth1): addr:00:00:00:aa:00:02
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 4(sw1): addr:96:df:8a:22:ec:e2
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 5(eth2): addr:00:00:00:aa:00:04
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 6(sw2): addr:4e:23:7d:e3:40:69
     config:     0
     state:      LIVE
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 LOCAL(ovsbr0): addr:00:00:00:aa:00:00
     config:     0
     state:      LIVE
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.3) (xid=0x7): frags=normal miss_send_len=0
root@ovs1:/tmp/pycore.46249/ovs1.conf# []
```

```
packet in 11141120 00:00:00:aa:00:03 33:33:00:00:00:02 3
packet in 11141120 be:b1:1a:e4:cb:96 33:33:00:00:00:02 6
packet in 11141120 46:68:c7:b2:10:df 33:33:00:00:00:02 1
```

The command ‘ovs-ofctl -O OpenFlow13 dump-ports ovsbr0’ prints the br0 OpenFlow ports statistics and shows detailed information about all the interfaces connected to this bridge, including the speed, state and peer information.



```
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (OF1.3) (xid=0x7): frags=normal miss_send_len=0
</ovs1.conf# ovs-ofctl -O OpenFlow13 dump-ports ovsbr0
OFPST_PORT reply (OF1.3) (xid=0x2): 7 ports
  port  sw2: rx pkts=12, bytes=936, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=88, bytes=6520, drop=0, errs=0, coll=0
           duration=522.417s
  port  sw1: rx pkts=12, bytes=936, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=84, bytes=6188, drop=0, errs=0, coll=0
           duration=522.463s
  port LOCAL: rx pkts=55, bytes=3092, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=13, bytes=1006, drop=0, errs=0, coll=0
           duration=522.517s
  port  eth0: rx pkts=46, bytes=3708, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=71, bytes=5246, drop=0, errs=0, coll=0
           duration=522.498s
  port  eth2: rx pkts=45, bytes=3618, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=72, bytes=5316, drop=0, errs=0, coll=0
           duration=522.427s
  port  sw0: rx pkts=12, bytes=936, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=80, bytes=5808, drop=0, errs=0, coll=0
           duration=522.490s
  port  eth1: rx pkts=38, bytes=3044, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=66, bytes=4784, drop=0, errs=0, coll=0
           duration=522.475s
root@ovs1:/tmp/pycore.46249/ovs1.conf# 
```

```
packet in 11141120 00:00:00:aa:00:05 33:33:00:00:00:02 5
packet in 11141120 32:2e:19:c8:4d:2f 33:33:00:00:00:02 3
packet in 11141120 46:68:c7:b2:10:df 33:33:00:00:00:02 1
```

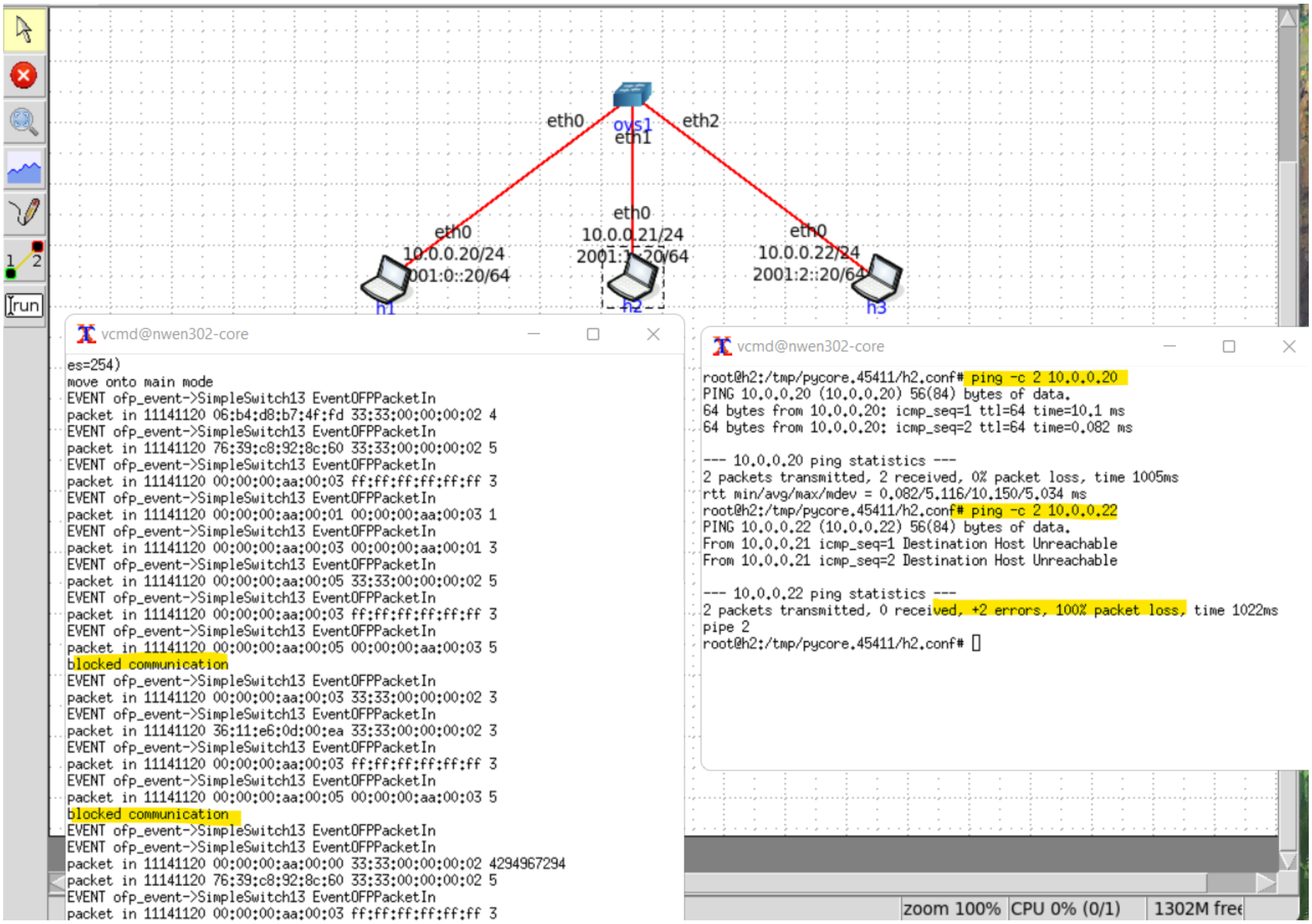
Now that I feel I understand Open vSwitches design and uses I will now be implementing the above pseudo code from the design section into program python code for the switch to run with to complete the given tasks.

KEY TASKS

Task 1

Modify simple_switch_13.py to include logic to **block** traffic between host h2 and host h3. Save the modified file as lab3_task1.py. Explain code and test procedures.

- To implement a blocking function I utilized the functions already provided in the SimpleSwitch13 class to include an if statement (after the code has learnt the src and dst values) that checks if the source or destination mac address is either from/to host 2 or host 3. Within the if statement I have included a print to terminal letting the user know the communication is blocked. If this statement is true after printing it will drop (by implementing an empty return). Lines 96 - 100.
- I got the source/destination MAC address information by inputting 'ifconfig' on each terminal, host 2 and host 3.
- Below is a screenshot of the code running through the switch and the hosts attempting to communicate with each other. (Shows that host 2 is able to ping host 1 but not host 3)



```
es=254)
move onto main mode
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 06:b4:d8:b7:4f:fd 33:33:00:00:00:02 4
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 76:39:c8:92:8c:60 33:33:00:00:00:02 5
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:03 ff:ff:ff:ff:ff:ff 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:01 00:00:00:aa:00:03 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:03 00:00:00:aa:00:01 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:05 33:33:00:00:00:02 5
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:03 ff:ff:ff:ff:ff:ff 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:05 00:00:00:aa:00:03 5
blocked communication
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:03 33:33:00:00:00:02 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 36:11:e6:0d:00:ea 33:33:00:00:00:02 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:03 ff:ff:ff:ff:ff:ff 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:05 00:00:00:aa:00:03 5
blocked communication
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:00 33:33:00:00:00:02 4294967294
packet in 11141120 76:39:c8:92:8c:60 33:33:00:00:00:02 5
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:03 ff:ff:ff:ff:ff:ff 3
```

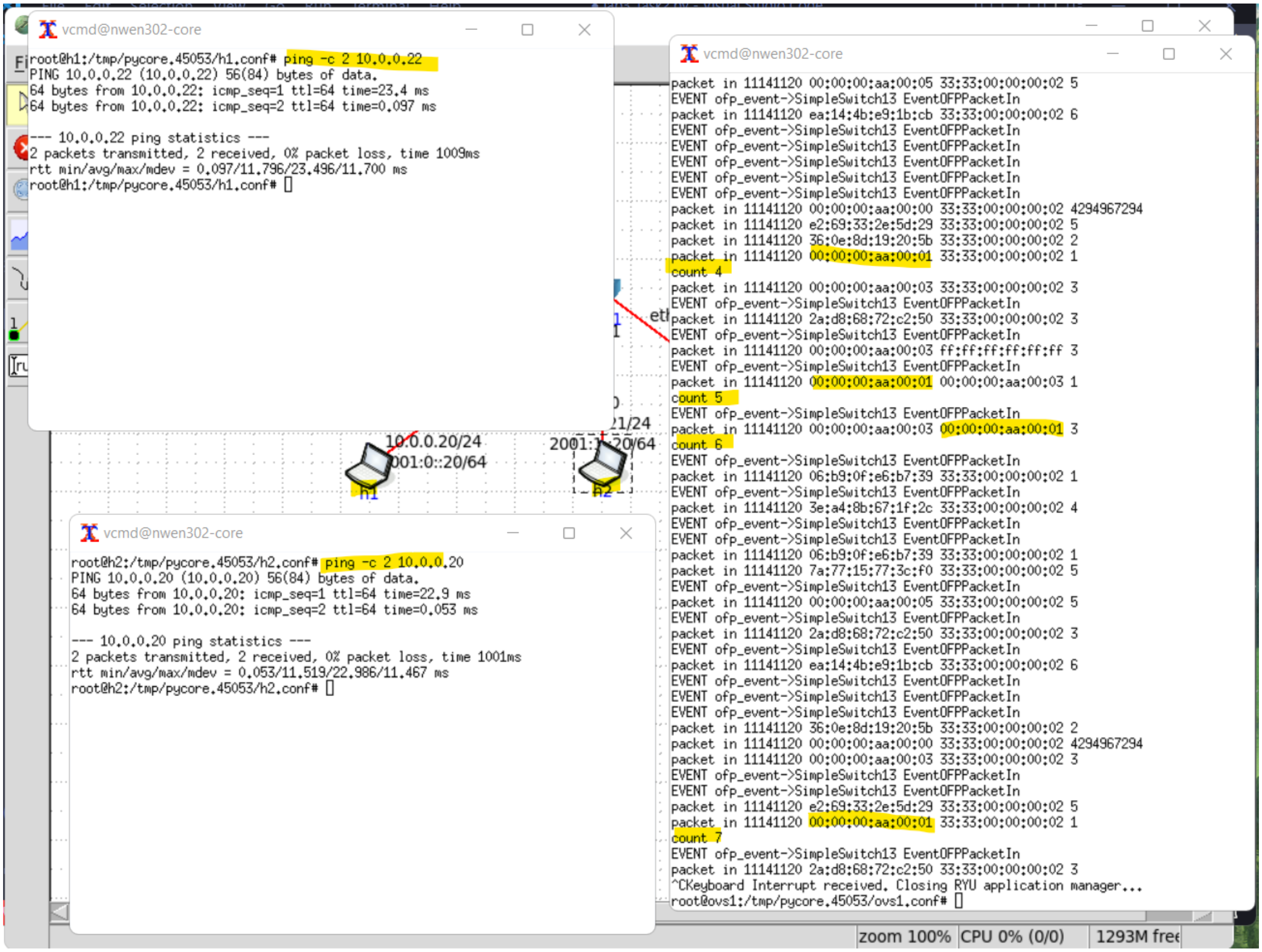
```
root@h2:/tmp/pycore.45411/h2.conf# ping -c 2 10.0.0.20
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.
64 bytes from 10.0.0.20: icmp_seq=1 ttl=64 time=10.1 ms
64 bytes from 10.0.0.20: icmp_seq=2 ttl=64 time=0.082 ms

--- 10.0.0.20 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 0.082/5.116/10.150/5.034 ms
root@h2:/tmp/pycore.45411/h2.conf# ping -c 2 10.0.0.22
PING 10.0.0.22 (10.0.0.22) 56(84) bytes of data.
From 10.0.0.21 icmp_seq=1 Destination Host Unreachable
From 10.0.0.21 icmp_seq=2 Destination Host Unreachable

--- 10.0.0.22 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1022ms
pipe 2
root@h2:/tmp/pycore.45411/h2.conf# 
```

Task 2

Modify simple_switch13.py to count all traffic going to and originating from host h1. Save the modified file as lab3_task2.py. Explain Code and test procedures.

First implementation:

- To be able to count the traffic from a specified host I created a global num variable to use for keeping track of the traffic and updating it's value.
- To be able to access the num variable I just called it before my following if statement within the class itself.
- Similar to task 1, I implemented the src and dst variables and checked if either of them were from host 1 and if so, update the num variable (num + 1) and print the num value.
- Below is a screenshot of the executed code through the VM: (host 1 pinging host 3, host 2 pinging host 1. The switch terminal with the code execution shows that the traffic from host 1 is sufficiently being counted).



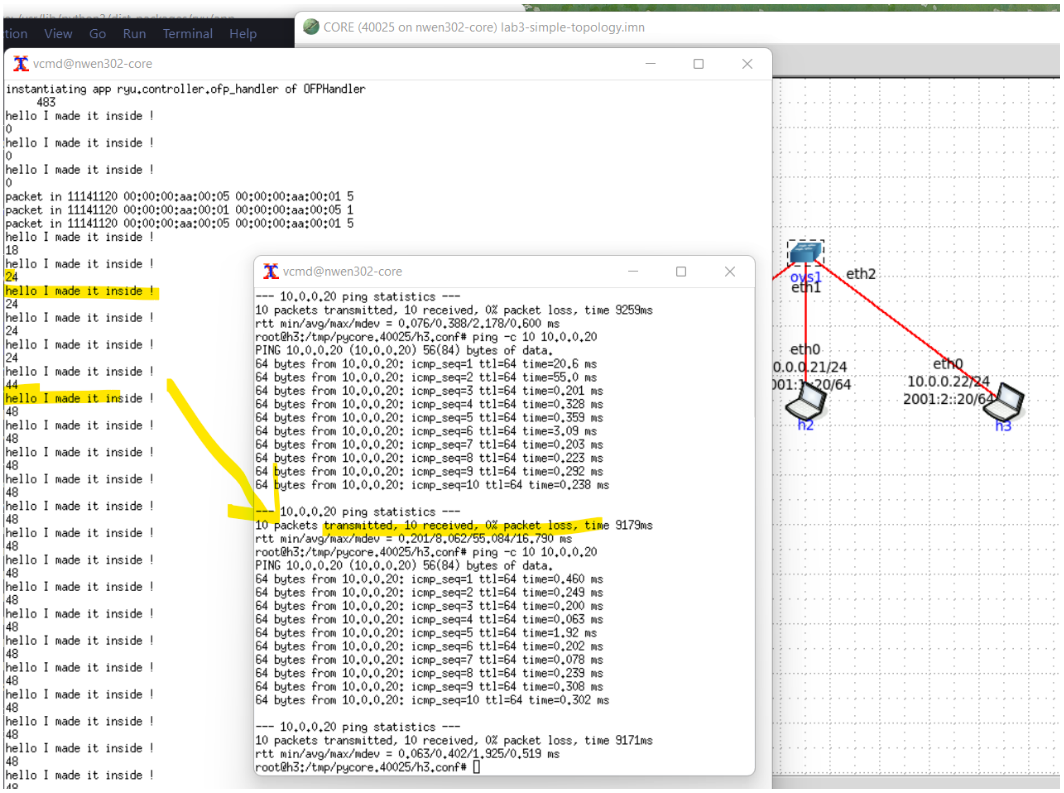
```
root@h1:/tmp/pycore.45053/h1.conf# ping -c 2 10.0.0.22
PING 10.0.0.22 (10.0.0.22) 56(84) bytes of data.
64 bytes from 10.0.0.22: icmp_seq=1 ttl=64 time=23.4 ms
64 bytes from 10.0.0.22: icmp_seq=2 ttl=64 time=0.097 ms

--- 10.0.0.22 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1009ms
rtt min/avg/max/mdev = 0.097/11.796/23.496/11.700 ms
root@h1:/tmp/pycore.45053/h1.conf# 
```

```
root@h2:/tmp/pycore.45053/h2.conf# ping -c 2 10.0.0.20
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.
64 bytes from 10.0.0.20: icmp_seq=1 ttl=64 time=22.9 ms
64 bytes from 10.0.0.20: icmp_seq=2 ttl=64 time=0.053 ms

--- 10.0.0.20 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.053/11.519/22.986/11.467 ms
root@h2:/tmp/pycore.45053/h2.conf# 
```

```
packet in 11141120 00:00:00:aa:00:05 33:33:00:00:00:02 5
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 ea:14:4b:e9:1b:cb 33:33:00:00:00:02 6
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:00 33:33:00:00:00:02 4294967294
packet in 11141120 e2:69:33:2e:5d:29 33:33:00:00:00:02 5
packet in 11141120 36:0e:8d:19:20:5b 33:33:00:00:00:02 2
packet in 11141120 00:00:00:aa:00:01 33:33:00:00:00:02 1
count 4
packet in 11141120 00:00:00:aa:00:03 33:33:00:00:00:02 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 2a:d8:68:72:c2:50 33:33:00:00:00:02 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:03 ff:ff:ff:ff:ff:ff 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:01 00:00:00:aa:00:03 1
count 5
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:03 00:00:00:aa:00:01 3
count 6
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 06:b9:0f:e6:b7:39 33:33:00:00:00:02 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 3e:a4:8b:67:1f:2c 33:33:00:00:00:02 4
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 06:b9:0f:e6:b7:39 33:33:00:00:00:02 1
packet in 11141120 7a:77:15:77:3c:f0 33:33:00:00:00:02 5
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 00:00:00:aa:00:05 33:33:00:00:00:02 5
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 2a:d8:68:72:c2:50 33:33:00:00:00:02 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 ea:14:4b:e9:1b:cb 33:33:00:00:00:02 6
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 36:0e:8d:19:20:5b 33:33:00:00:00:02 2
packet in 11141120 00:00:00:aa:00:00 33:33:00:00:00:02 4294967294
packet in 11141120 00:00:00:aa:00:03 33:33:00:00:00:02 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 e2:69:33:2e:5d:29 33:33:00:00:00:02 5
packet in 11141120 00:00:00:aa:00:01 33:33:00:00:00:02 1
count 7
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11141120 2a:d8:68:72:c2:50 33:33:00:00:00:02 3
^CKeyboard Interrupt received. Closing RYU application manager...
root@ovs1:/tmp/pycore.45053/ovs1.conf# 
```

Second implementation:

- After some debugging I found that the count wasn’t correct and had to implement my code another way. Referencing from the traffic_minotor library (shown within the references section). I had to implement a ‘_state_change_handler()’, ‘_monitor()’, ‘_request_stats(), and a ‘_port_stats_reply_handler’.
- Within the ‘_port_stats_reply_handler(); function I implemented a stats variable which is a dictionary that holds each host's traffic information. By accessing the first element of the stats dictionary (host 1) by stats[0].rx_packets + stats[0].tx_packets it returns the number value of all traffic that has been through host 1. Since this value holds the history of every single packet sent through host 1 the number is quite large as it doesn’t count from our point of beginning the code.
- To debug this I created new variables where one will be counting the total packets and one that will be counting the ‘new’ packets minus the total packets. This will return only the packets that we initialize. From the screenshot below it shows my code running. The value begins at 0 before we ping host 1 which is good, then when we ping 10 packets to host 1 from host 2 the count value goes to 24. We account for 20 packets and the other 4 packets are the background broadcast that the program runs regardless of our code. When I ping 10 packets again the value goes from 24 to 48 which shows we successfully recorded the 20 send/receive packets communicating with host 1, then similar to the last time the Ryu is sending the background broadcast which adds another 4 packets to the count.



```
instantiating app ryu.controller.ofp_handler of OFPHandler
    483
hello I made it inside !
0
hello I made it inside !
0
hello I made it inside !
0
packet in 11141120 00:00:00:aa:00:05 00:00:00:aa:00:01 5
packet in 11141120 00:00:00:aa:00:01 00:00:00:aa:00:05 1
packet in 11141120 00:00:00:aa:00:05 00:00:00:aa:00:01 5
hello I made it inside !
18
hello I made it inside !
24
hello I made it inside !
24
hello I made it inside !
24
hello I made it inside !
24
hello I made it inside !
44
hello I made it inside !
48
hello I made it inside !
48
hello I made it inside !
48
hello I made it inside !
48
hello I made it inside !
48
hello I made it inside !
48
hello I made it inside !
48
hello I made it inside !
48
hello I made it inside !
48
hello I made it inside !
48
hello I made it inside !
48
hello I made it inside !
48
hello I made it inside !
48
hello I made it inside !
```

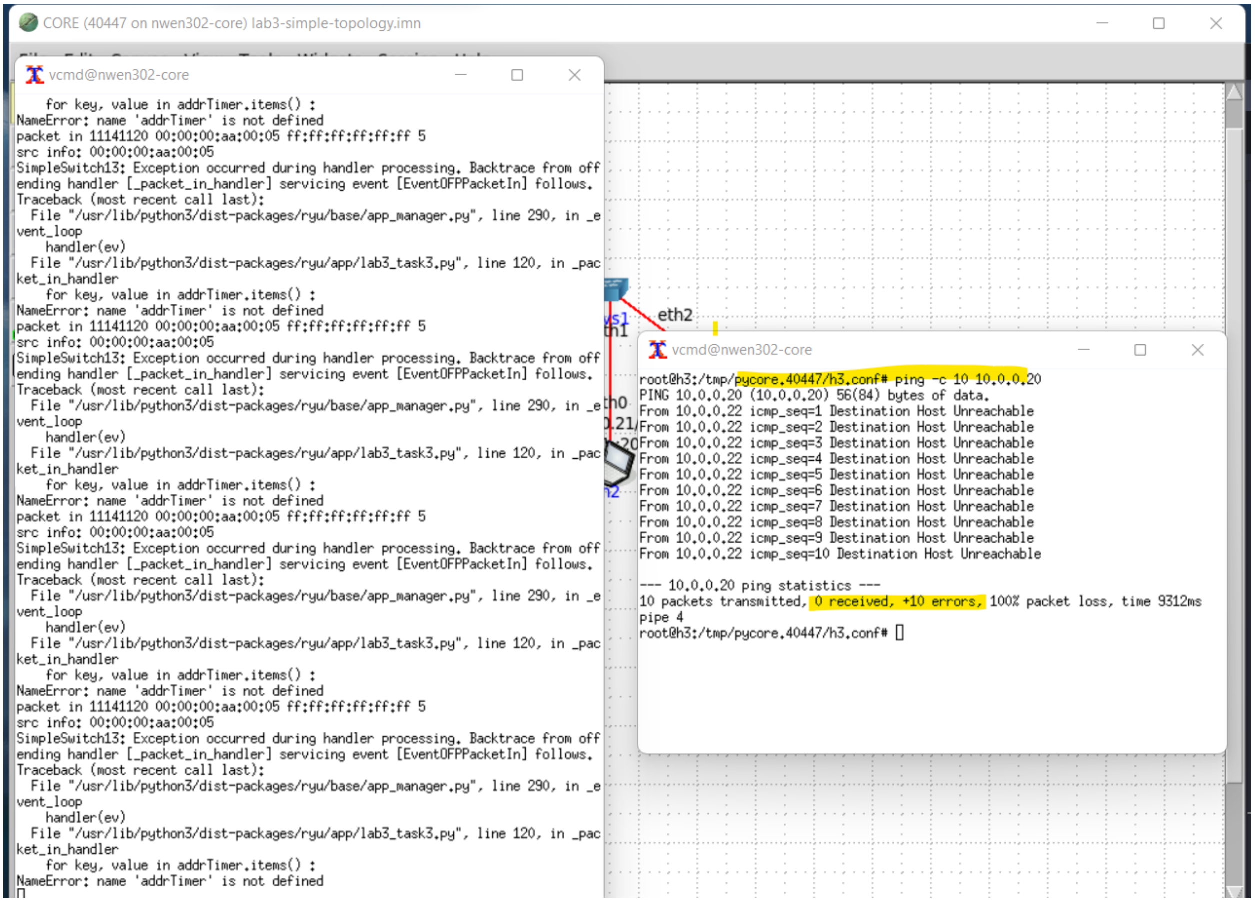
```
--- 10.0.0.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9259ms
rtt min/avg/max/mdev = 0.076/0.388/2.178/0.600 ms
root@h3:/tmp/pycore.40025/h3.conf# ping -c 10 10.0.0.20
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.
64 bytes from 10.0.0.20: icmp_seq=1 ttl=64 time=20.6 ms
64 bytes from 10.0.0.20: icmp_seq=2 ttl=64 time=55.0 ms
64 bytes from 10.0.0.20: icmp_seq=3 ttl=64 time=0.201 ms
64 bytes from 10.0.0.20: icmp_seq=4 ttl=64 time=0.328 ms
64 bytes from 10.0.0.20: icmp_seq=5 ttl=64 time=0.359 ms
64 bytes from 10.0.0.20: icmp_seq=6 ttl=64 time=3.09 ms
64 bytes from 10.0.0.20: icmp_seq=7 ttl=64 time=0.203 ms
64 bytes from 10.0.0.20: icmp_seq=8 ttl=64 time=0.223 ms
64 bytes from 10.0.0.20: icmp_seq=9 ttl=64 time=0.292 ms
64 bytes from 10.0.0.20: icmp_seq=10 ttl=64 time=0.238 ms

--- 10.0.0.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9179ms
rtt min/avg/max/mdev = 0.201/8.062/55.084/16.790 ms
root@h3:/tmp/pycore.40025/h3.conf# ping -c 10 10.0.0.20
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.
64 bytes from 10.0.0.20: icmp_seq=1 ttl=64 time=0.460 ms
64 bytes from 10.0.0.20: icmp_seq=2 ttl=64 time=0.249 ms
64 bytes from 10.0.0.20: icmp_seq=3 ttl=64 time=0.200 ms
64 bytes from 10.0.0.20: icmp_seq=4 ttl=64 time=0.063 ms
64 bytes from 10.0.0.20: icmp_seq=5 ttl=64 time=1.92 ms
64 bytes from 10.0.0.20: icmp_seq=6 ttl=64 time=0.202 ms
64 bytes from 10.0.0.20: icmp_seq=7 ttl=64 time=0.078 ms
64 bytes from 10.0.0.20: icmp_seq=8 ttl=64 time=0.239 ms
64 bytes from 10.0.0.20: icmp_seq=9 ttl=64 time=0.308 ms
64 bytes from 10.0.0.20: icmp_seq=10 ttl=64 time=0.302 ms

--- 10.0.0.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9171ms
rtt min/avg/max/mdev = 0.063/0.402/1.925/0.519 ms
root@h3:/tmp/pycore.40025/h3.conf# 
```

Task 3

Extend simple_switch_13.py to combine Task 1 and Task 2 functionalities. Keep track of all traffic (count the number of packets) originating from each host. If the counter exceeds a specific number, block all the traffic originating from this host for 1 minute. The maximum packet count number should be configured through the MAX_COUNT variable. Save the modified file as lab3_task3.py. Explain Code and test procedures.

First Implementation:

- Outside the class I created two separate 'dictionaries' (similar functioning to maps/arrays in java) that contain the hosts mac addresses. One map/dictionaries value holds an int value for counting and the other dictionaries value holds a boolean value which will be used for the blocking functionality later on.
- Imported the python threading library for opening the threads for each blocked host (each time a host is blocked it needs to be run on a new separate thread to not affect the program root processes).
- Thread function declaration at the beginning of the SimpleSwitch13 class. A simple method named timerChecks which passes in a mac_address and resets the mac_address from the addrTimer dictionary to false (false = unblocked, true = blocked). This gets called when the timer has hit 60 seconds as a means to reset the blocking.
- Added the hosts mac addresses to the __init__ function as a map/dictionary to compare the src info to later on.
- Within the _packet_in_handler method the first thing I did was call the global variables/dictionaries I created and then did a simple if statement which checks the if the source (src) is inside the mac_address list (referring to the dictionary/map within the __init__ function) and when it recognises it add the values count to '+1' on the specific source addresses element.
- For the timer threading function I created a for loop which goes through the keys and values of each mac address within the addrTimer dictionary and checks if the current element's source value is equal to true. If the value returns true, return, as a means to continue blocking the traffic.
- The last if statement is the counter function which checks if the source addresses value is equal to the MAX_COUNT variable (10) and if so, change the source addresses addrTimer value to true and open/start the thread which will count up to 60 seconds before running the timerChecks function which will unblock it.
- The way I have gone about testing this is after running the code through the switch terminal I have pinged from each host 10 packets to check if 1, the count sufficiently goes upwards and 2, when it hits 10 the communications are blocked and the terminal notifies us so.



```
    for key, value in addrTimer.items() :
NameError: name 'addrTimer' is not defined
packet in 11141120 00:00:00:aa:00:05 ff:ff:ff:ff:ff:ff 5
src info: 00:00:00:aa:00:05
SimpleSwitch13: Exception occurred during handler processing. Backtrace from off
ending handler [_packet_in_handler] servicing event [EventOFPPacketIn] follows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _e
vent_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 120, in _pac
ket_in_handler
    for key, value in addrTimer.items() :
NameError: name 'addrTimer' is not defined
packet in 11141120 00:00:00:aa:00:05 ff:ff:ff:ff:ff:ff 5
src info: 00:00:00:aa:00:05
SimpleSwitch13: Exception occurred during handler processing. Backtrace from off
ending handler [_packet_in_handler] servicing event [EventOFPPacketIn] follows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _e
vent_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 120, in _pac
ket_in_handler
    for key, value in addrTimer.items() :
NameError: name 'addrTimer' is not defined
packet in 11141120 00:00:00:aa:00:05 ff:ff:ff:ff:ff:ff 5
src info: 00:00:00:aa:00:05
SimpleSwitch13: Exception occurred during handler processing. Backtrace from off
ending handler [_packet_in_handler] servicing event [EventOFPPacketIn] follows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _e
vent_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 120, in _pac
ket_in_handler
    for key, value in addrTimer.items() :
NameError: name 'addrTimer' is not defined
packet in 11141120 00:00:00:aa:00:05 ff:ff:ff:ff:ff:ff 5
src info: 00:00:00:aa:00:05
SimpleSwitch13: Exception occurred during handler processing. Backtrace from off
ending handler [_packet_in_handler] servicing event [EventOFPPacketIn] follows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _e
vent_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 120, in _pac
ket_in_handler
    for key, value in addrTimer.items() :
NameError: name 'addrTimer' is not defined
```

```
root@h3:/tmp/pycore.40447/h3.conf# ping -c 10 10.0.0.20
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.
From 10.0.0.22 icmp_seq=1 Destination Host Unreachable
From 10.0.0.22 icmp_seq=2 Destination Host Unreachable
From 10.0.0.22 icmp_seq=3 Destination Host Unreachable
From 10.0.0.22 icmp_seq=4 Destination Host Unreachable
From 10.0.0.22 icmp_seq=5 Destination Host Unreachable
From 10.0.0.22 icmp_seq=6 Destination Host Unreachable
From 10.0.0.22 icmp_seq=7 Destination Host Unreachable
From 10.0.0.22 icmp_seq=8 Destination Host Unreachable
From 10.0.0.22 icmp_seq=9 Destination Host Unreachable
From 10.0.0.22 icmp_seq=10 Destination Host Unreachable

--- 10.0.0.20 ping statistics ---
10 packets transmitted, 0 received, +10 errors, 100% packet loss, time 9312ms
pipe 4
root@h3:/tmp/pycore.40447/h3.conf# 
```

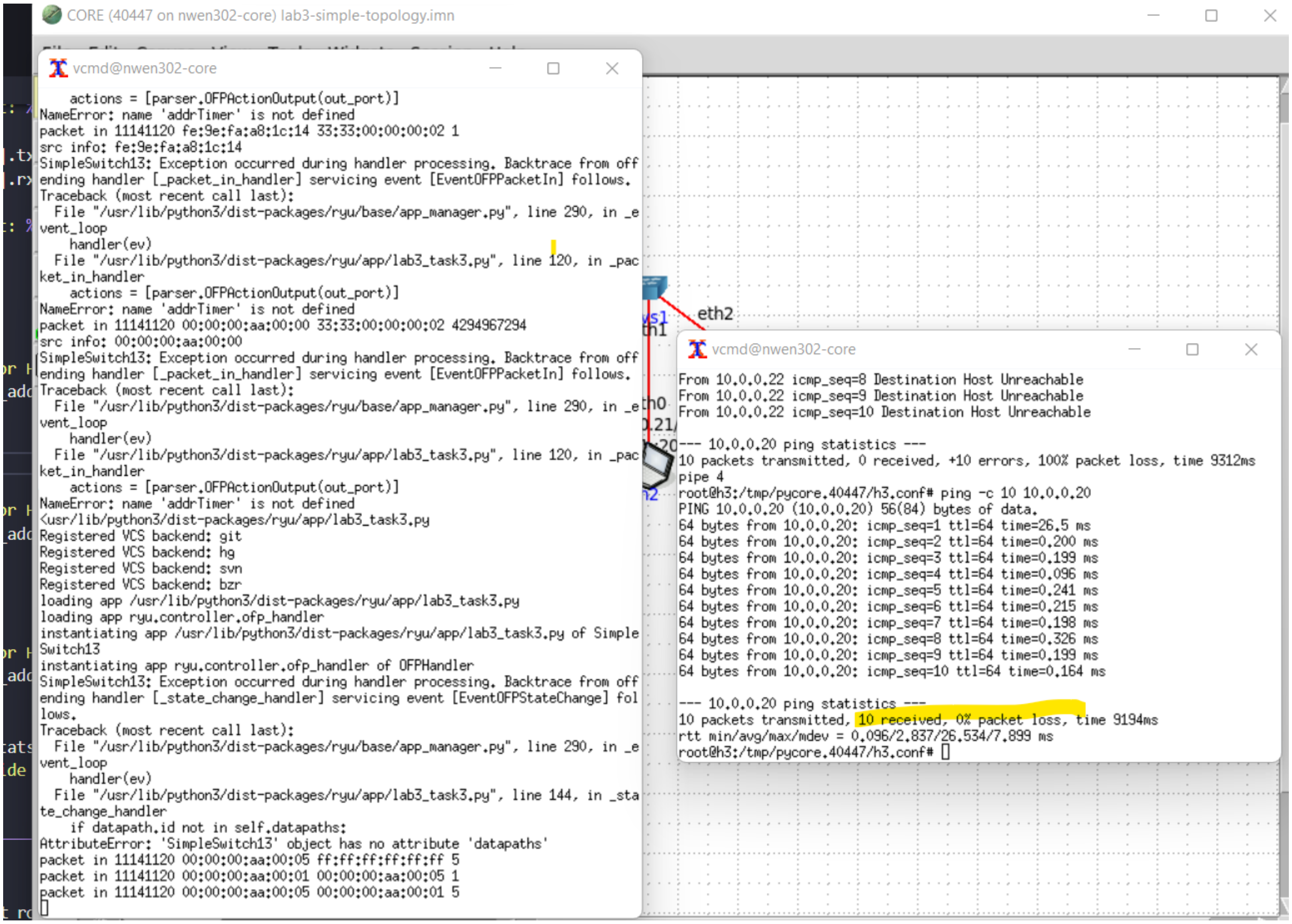
- Above screenshots shows the testing output using the above logic

Second Implementation:

- Similar to the second implementation of task 2 I started with referencing from the traffic_minotor library and implemented a '_state_change_handler()', '_monitor()', '_request_stats(), and a '_port_stats_reply_handler'.
- Within the '_port_stats_reply_handler(); function I implemented a stats variable which is a dictionary that holds each host's traffic information.
- I did the same thing with task 2 where I checked if the stats[0] of rx_packets and tx_packets and then added the values to a count variable except I did so for each host with separate variables.
- Then I changed the hosts elements value within the addrCount variable to increment by 1 as a means to count upwards.
- Then I made a loop to go through the keys and values of the addrCount to assess. When the hosts value within the addrCount dictionary reaches the MAX_COUNT (10) it will then start a timing thread which will block traffic through that host for 60 seconds. After the 60seconds the timerchecks()function will be called which will unblock the traffic.

Note:

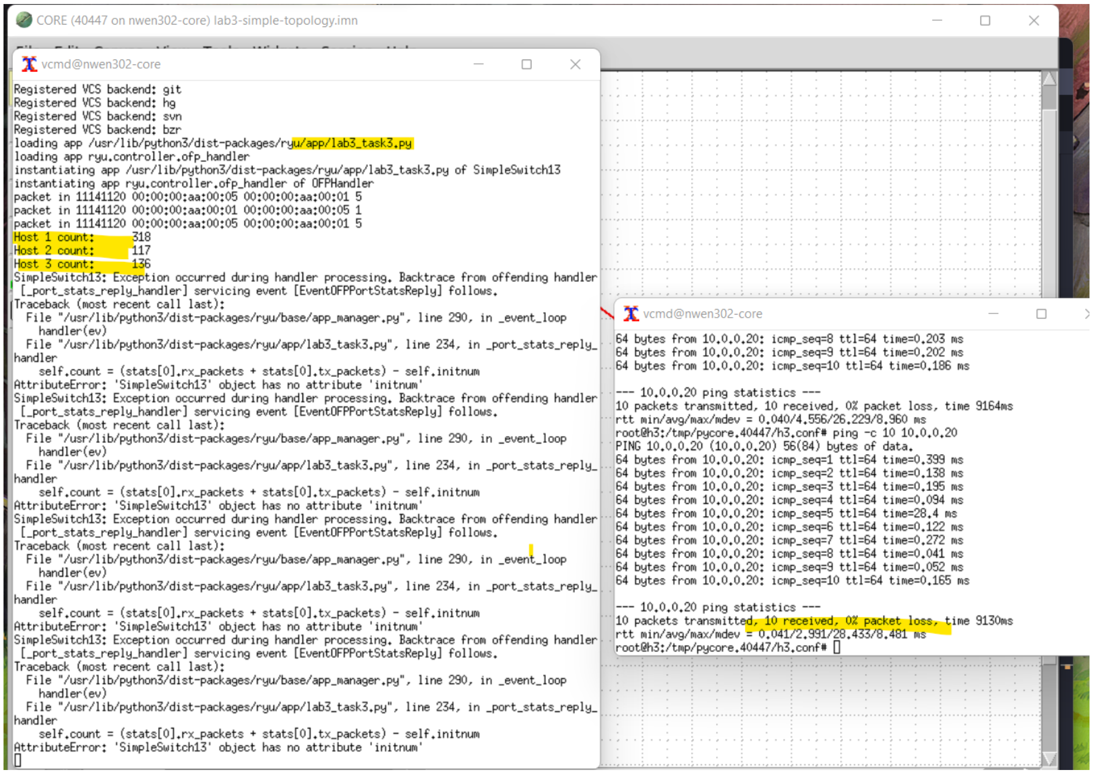
- Through testing I believe I was close but unfortunately did not get the results I had hoped for. The screenshots below show the output I was getting, It appears it was somewhat counting (the number 0 to 26 is from host 3 pinging host 1 10 times, so 20 packets went to and from host 1) but unsure if the blocking is successful as I attempted to ping 10 packets more than once. What should have happened was after host 1's count value hit 10 the manual pings from host 3 would no longer be successful.



```
    actions = [parser.OFPActionOutput(out_port)]
NameError: name 'addrTimer' is not defined
packet in 11141120 fe:9e:fa:a8:1c:14 33:33:00:00:00:02 1
src info: fe:9e:fa:a8:1c:14
SimpleSwitch13: Exception occurred during handler processing. Backtrace from off
ending handler [_packet_in_handler] servicing event [EventOFPPacketIn] follows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _e
vent_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 120, in _pac
ket_in_handler
    actions = [parser.OFPActionOutput(out_port)]
NameError: name 'addrTimer' is not defined
packet in 11141120 00:00:00:aa:00:00 33:33:00:00:00:02 4294967294
src info: 00:00:00:aa:00:00
SimpleSwitch13: Exception occurred during handler processing. Backtrace from off
ending handler [_packet_in_handler] servicing event [EventOFPPacketIn] follows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _e
vent_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 120, in _pac
ket_in_handler
    actions = [parser.OFPActionOutput(out_port)]
NameError: name 'addrTimer' is not defined
<usr/lib/python3/dist-packages/ryu/app/lab3_task3.py
Registered VCS backend: git
Registered VCS backend: hg
Registered VCS backend: svn
Registered VCS backend: bzr
loading app /usr/lib/python3/dist-packages/ryu/app/lab3_task3.py
loading app ryu.controller.ofp_handler
instantiating app /usr/lib/python3/dist-packages/ryu/app/lab3_task3.py of Simple
Switch13
instantiating app ryu.controller.ofp_handler of OFPHandler
SimpleSwitch13: Exception occurred during handler processing. Backtrace from off
ending handler [_state_change_handler] servicing event [EventOFPStateChange] fol
lows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _e
vent_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 144, in _sta
te_change_handler
    if datapath.id not in self.datapaths:
AttributeError: 'SimpleSwitch13' object has no attribute 'datapaths'
packet in 11141120 00:00:00:aa:00:05 ff:ff:ff:ff:ff:ff 5
packet in 11141120 00:00:00:aa:00:01 00:00:00:aa:00:05 1
packet in 11141120 00:00:00:aa:00:05 00:00:00:aa:00:01 5
```

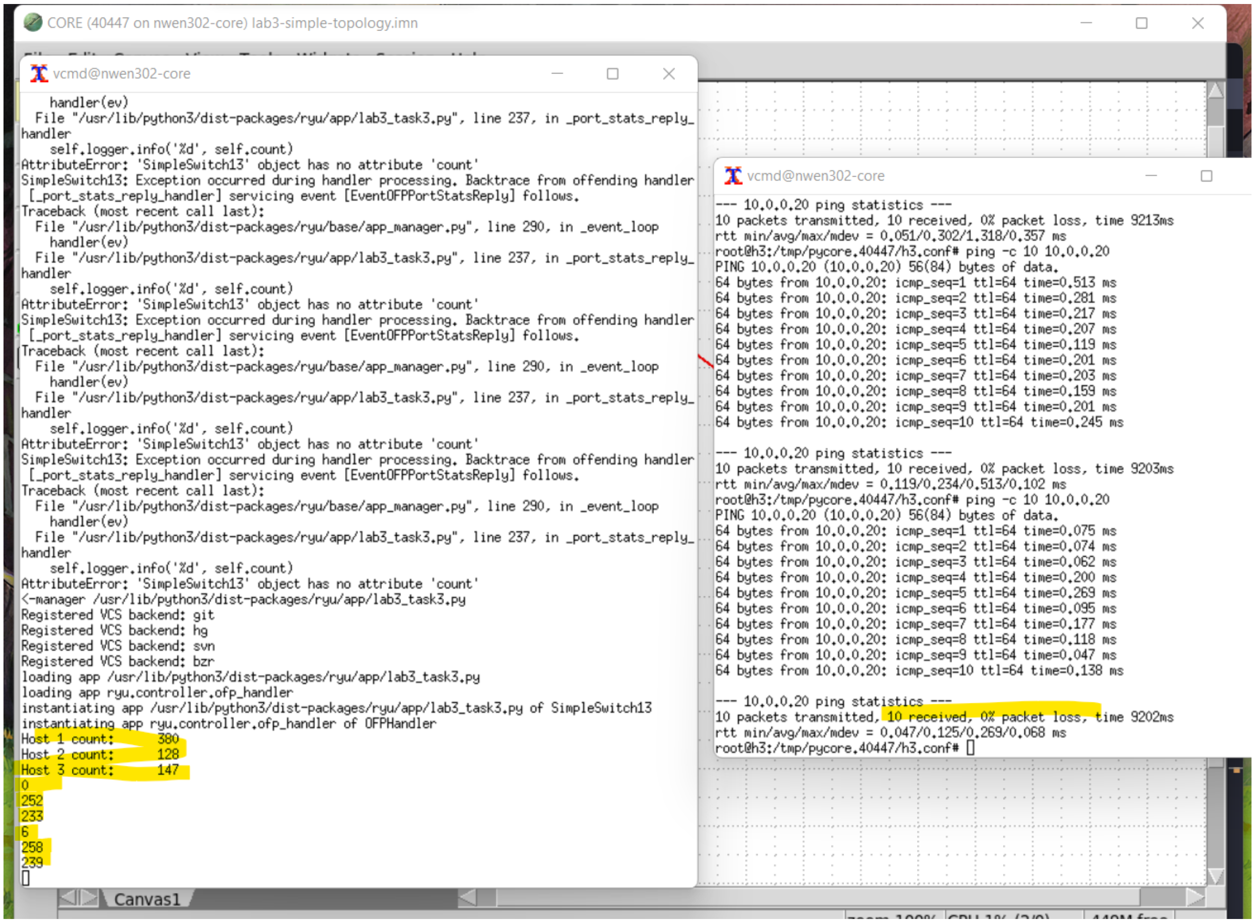
```
From 10.0.0.22 icmp_seq=8 Destination Host Unreachable
From 10.0.0.22 icmp_seq=9 Destination Host Unreachable
From 10.0.0.22 icmp_seq=10 Destination Host Unreachable

--- 10.0.0.20 ping statistics ---
10 packets transmitted, 0 received, +10 errors, 100% packet loss, time 9312ms
pipe 4
root@h3:/tmp/pycore.40447/h3.conf# ping -c 10 10.0.0.20
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.
64 bytes from 10.0.0.20: icmp_seq=1 ttl=64 time=26.5 ms
64 bytes from 10.0.0.20: icmp_seq=2 ttl=64 time=0.200 ms
64 bytes from 10.0.0.20: icmp_seq=3 ttl=64 time=0.199 ms
64 bytes from 10.0.0.20: icmp_seq=4 ttl=64 time=0.096 ms
64 bytes from 10.0.0.20: icmp_seq=5 ttl=64 time=0.241 ms
64 bytes from 10.0.0.20: icmp_seq=6 ttl=64 time=0.215 ms
64 bytes from 10.0.0.20: icmp_seq=7 ttl=64 time=0.198 ms
64 bytes from 10.0.0.20: icmp_seq=8 ttl=64 time=0.326 ms
64 bytes from 10.0.0.20: icmp_seq=9 ttl=64 time=0.199 ms
64 bytes from 10.0.0.20: icmp_seq=10 ttl=64 time=0.164 ms

--- 10.0.0.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9194ms
rtt min/avg/max/mdev = 0.096/2.837/26.534/7.899 ms
root@h3:/tmp/pycore.40447/h3.conf# 
```

- The above screenshots show the terminal output from running my updated task3 code using the above implementation. I believe I was almost there but did not quite execute it.



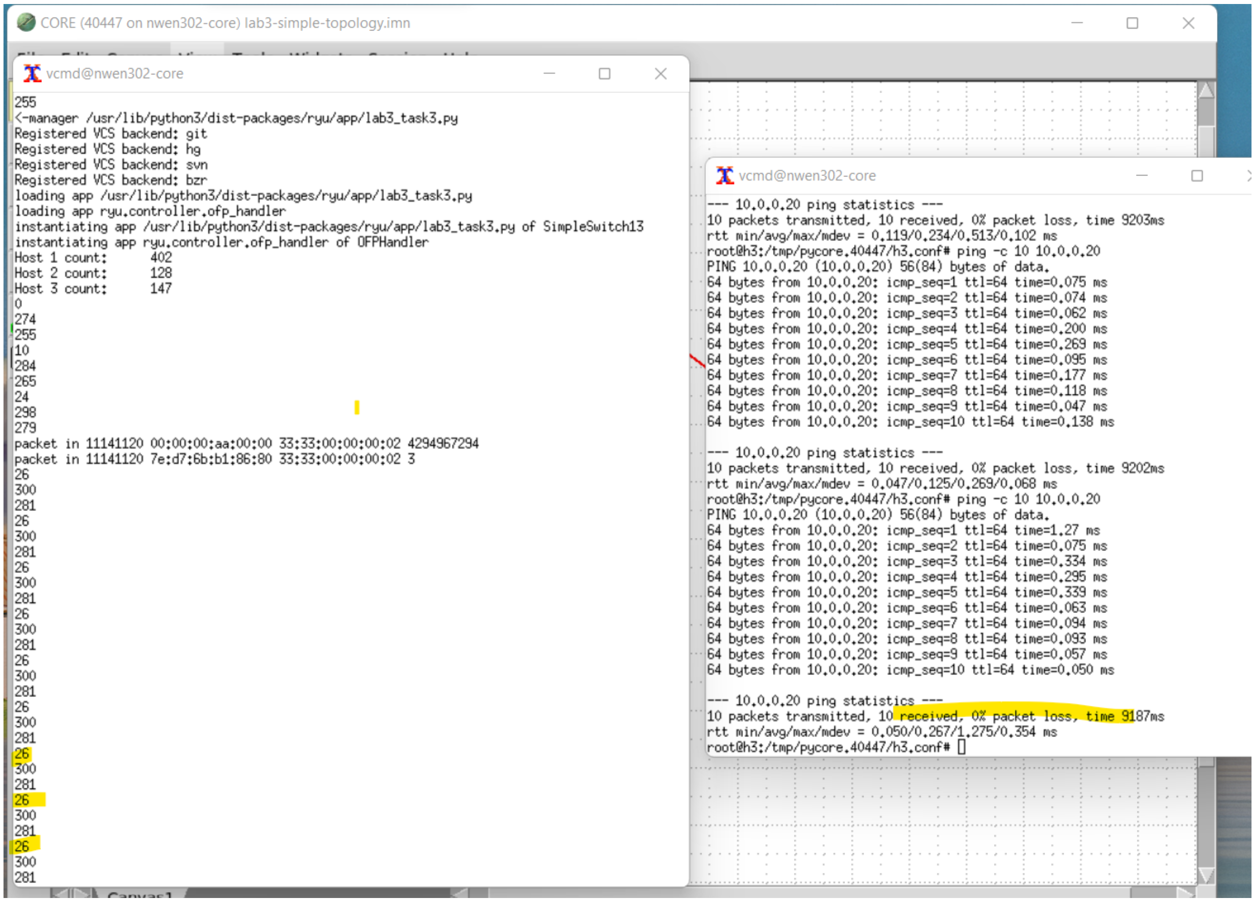
```
Registered VCS backend: git
Registered VCS backend: hg
Registered VCS backend: svn
Registered VCS backend: bzr
loading app /usr/lib/python3/dist-packages/ryu/app/lab3_task3.py
loading app ryu.controller.ofp_handler
instantiating app /usr/lib/python3/dist-packages/ryu/app/lab3_task3.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
packet in 11141120 00:00:00:aa:00:05 00:00:00:aa:00:01 5
packet in 11141120 00:00:00:aa:00:01 00:00:00:aa:00:05 1
packet in 11141120 00:00:00:aa:00:05 00:00:00:aa:00:01 5
Host 1 count:       318
Host 2 count:       117
Host 3 count:       136
SimpleSwitch13: Exception occurred during handler processing. Backtrace from offending handler
 [_port_stats_reply_handler] servicing event [EventOFPPortStatsReply] follows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _event_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 234, in _port_stats_reply_
handler
    self.count = (stats[0].rx_packets + stats[0].tx_packets) - self.initnum
AttributeError: 'SimpleSwitch13' object has no attribute 'initnum'
SimpleSwitch13: Exception occurred during handler processing. Backtrace from offending handler
 [_port_stats_reply_handler] servicing event [EventOFPPortStatsReply] follows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _event_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 234, in _port_stats_reply_
handler
    self.count = (stats[0].rx_packets + stats[0].tx_packets) - self.initnum
AttributeError: 'SimpleSwitch13' object has no attribute 'initnum'
SimpleSwitch13: Exception occurred during handler processing. Backtrace from offending handler
 [_port_stats_reply_handler] servicing event [EventOFPPortStatsReply] follows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _event_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 234, in _port_stats_reply_
handler
    self.count = (stats[0].rx_packets + stats[0].tx_packets) - self.initnum
AttributeError: 'SimpleSwitch13' object has no attribute 'initnum'
SimpleSwitch13: Exception occurred during handler processing. Backtrace from offending handler
 [_port_stats_reply_handler] servicing event [EventOFPPortStatsReply] follows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _event_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 234, in _port_stats_reply_
handler
    self.count = (stats[0].rx_packets + stats[0].tx_packets) - self.initnum
AttributeError: 'SimpleSwitch13' object has no attribute 'initnum'
```

```
64 bytes from 10.0.0.20: icmp_seq=8 ttl=64 time=0.203 ms
64 bytes from 10.0.0.20: icmp_seq=9 ttl=64 time=0.202 ms
64 bytes from 10.0.0.20: icmp_seq=10 ttl=64 time=0.186 ms

--- 10.0.0.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9164ms
rtt min/avg/max/mdev = 0.040/4.556/26.229/8.960 ms
root@h3:/tmp/pycore.40447/h3.conf# ping -c 10 10.0.0.20
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.
64 bytes from 10.0.0.20: icmp_seq=1 ttl=64 time=0.399 ms
64 bytes from 10.0.0.20: icmp_seq=2 ttl=64 time=0.138 ms
64 bytes from 10.0.0.20: icmp_seq=3 ttl=64 time=0.195 ms
64 bytes from 10.0.0.20: icmp_seq=4 ttl=64 time=0.094 ms
64 bytes from 10.0.0.20: icmp_seq=5 ttl=64 time=28.4 ms
64 bytes from 10.0.0.20: icmp_seq=6 ttl=64 time=0.122 ms
64 bytes from 10.0.0.20: icmp_seq=7 ttl=64 time=0.272 ms
64 bytes from 10.0.0.20: icmp_seq=8 ttl=64 time=0.041 ms
64 bytes from 10.0.0.20: icmp_seq=9 ttl=64 time=0.052 ms
64 bytes from 10.0.0.20: icmp_seq=10 ttl=64 time=0.165 ms

--- 10.0.0.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9130ms
rtt min/avg/max/mdev = 0.041/2.991/28.433/8.481 ms
root@h3:/tmp/pycore.40447/h3.conf# 
```



```
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 237, in _port_stats_reply_
handler
    self.logger.info('%d', self.count)
AttributeError: 'SimpleSwitch13' object has no attribute 'count'
SimpleSwitch13: Exception occurred during handler processing. Backtrace from offending handler
 [_port_stats_reply_handler] servicing event [EventOFPPortStatsReply] follows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _event_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 237, in _port_stats_reply_
handler
    self.logger.info('%d', self.count)
AttributeError: 'SimpleSwitch13' object has no attribute 'count'
SimpleSwitch13: Exception occurred during handler processing. Backtrace from offending handler
 [_port_stats_reply_handler] servicing event [EventOFPPortStatsReply] follows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _event_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 237, in _port_stats_reply_
handler
    self.logger.info('%d', self.count)
AttributeError: 'SimpleSwitch13' object has no attribute 'count'
SimpleSwitch13: Exception occurred during handler processing. Backtrace from offending handler
 [_port_stats_reply_handler] servicing event [EventOFPPortStatsReply] follows.
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/ryu/base/app_manager.py", line 290, in _event_loop
    handler(ev)
  File "/usr/lib/python3/dist-packages/ryu/app/lab3_task3.py", line 237, in _port_stats_reply_
handler
    self.logger.info('%d', self.count)
AttributeError: 'SimpleSwitch13' object has no attribute 'count'
<-manager /usr/lib/python3/dist-packages/ryu/app/lab3_task3.py
Registered VCS backend: git
Registered VCS backend: hg
Registered VCS backend: svn
Registered VCS backend: bzr
loading app /usr/lib/python3/dist-packages/ryu/app/lab3_task3.py
loading app ryu.controller.ofp_handler
instantiating app /usr/lib/python3/dist-packages/ryu/app/lab3_task3.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
Host 1 count:       380
Host 2 count:       128
Host 3 count:       147
0
252
233
6
258
239
```

```
--- 10.0.0.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9213ms
rtt min/avg/max/mdev = 0.051/0.302/1.318/0.357 ms
root@h3:/tmp/pycore.40447/h3.conf# ping -c 10 10.0.0.20
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.
64 bytes from 10.0.0.20: icmp_seq=1 ttl=64 time=0.513 ms
64 bytes from 10.0.0.20: icmp_seq=2 ttl=64 time=0.281 ms
64 bytes from 10.0.0.20: icmp_seq=3 ttl=64 time=0.217 ms
64 bytes from 10.0.0.20: icmp_seq=4 ttl=64 time=0.207 ms
64 bytes from 10.0.0.20: icmp_seq=5 ttl=64 time=0.119 ms
64 bytes from 10.0.0.20: icmp_seq=6 ttl=64 time=0.201 ms
64 bytes from 10.0.0.20: icmp_seq=7 ttl=64 time=0.203 ms
64 bytes from 10.0.0.20: icmp_seq=8 ttl=64 time=0.159 ms
64 bytes from 10.0.0.20: icmp_seq=9 ttl=64 time=0.201 ms
64 bytes from 10.0.0.20: icmp_seq=10 ttl=64 time=0.245 ms

--- 10.0.0.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9203ms
rtt min/avg/max/mdev = 0.119/0.234/0.513/0.102 ms
root@h3:/tmp/pycore.40447/h3.conf# ping -c 10 10.0.0.20
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.
64 bytes from 10.0.0.20: icmp_seq=1 ttl=64 time=0.075 ms
64 bytes from 10.0.0.20: icmp_seq=2 ttl=64 time=0.074 ms
64 bytes from 10.0.0.20: icmp_seq=3 ttl=64 time=0.062 ms
64 bytes from 10.0.0.20: icmp_seq=4 ttl=64 time=0.200 ms
64 bytes from 10.0.0.20: icmp_seq=5 ttl=64 time=0.269 ms
64 bytes from 10.0.0.20: icmp_seq=6 ttl=64 time=0.095 ms
64 bytes from 10.0.0.20: icmp_seq=7 ttl=64 time=0.177 ms
64 bytes from 10.0.0.20: icmp_seq=8 ttl=64 time=0.118 ms
64 bytes from 10.0.0.20: icmp_seq=9 ttl=64 time=0.047 ms
64 bytes from 10.0.0.20: icmp_seq=10 ttl=64 time=0.138 ms

--- 10.0.0.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9202ms
rtt min/avg/max/mdev = 0.047/0.125/0.269/0.068 ms
root@h3:/tmp/pycore.40447/h3.conf# 
```



```
255
<-manager /usr/lib/python3/dist-packages/ryu/app/lab3_task3.py
Registered VCS backend: git
Registered VCS backend: hg
Registered VCS backend: svn
Registered VCS backend: bzr
loading app /usr/lib/python3/dist-packages/ryu/app/lab3_task3.py
loading app ryu.controller.ofp_handler
instantiating app /usr/lib/python3/dist-packages/ryu/app/lab3_task3.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
Host 1 count:      402
Host 2 count:      128
Host 3 count:      147
0
274
255
10
284
265
24
298
279
packet in 11141120 00:00:00:aa:00:00 33:33:00:00:00:02 4294967294
packet in 11141120 7e:d7:6b:b1:86:80 33:33:00:00:00:02 3
26
300
281
26
300
281
26
300
281
26
300
281
26
300
281
26
300
281
26
300
281
26
300
281
26
300
281
```

```
--- 10.0.0.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9203ms
rtt min/avg/max/mdev = 0.119/0.234/0.513/0.102 ms
root@h3:/tmp/pycore.40447/h3.conf# ping -c 10 10.0.0.20
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.
64 bytes from 10.0.0.20: icmp_seq=1 ttl=64 time=0.075 ms
64 bytes from 10.0.0.20: icmp_seq=2 ttl=64 time=0.074 ms
64 bytes from 10.0.0.20: icmp_seq=3 ttl=64 time=0.062 ms
64 bytes from 10.0.0.20: icmp_seq=4 ttl=64 time=0.200 ms
64 bytes from 10.0.0.20: icmp_seq=5 ttl=64 time=0.269 ms
64 bytes from 10.0.0.20: icmp_seq=6 ttl=64 time=0.095 ms
64 bytes from 10.0.0.20: icmp_seq=7 ttl=64 time=0.177 ms
64 bytes from 10.0.0.20: icmp_seq=8 ttl=64 time=0.118 ms
64 bytes from 10.0.0.20: icmp_seq=9 ttl=64 time=0.047 ms
64 bytes from 10.0.0.20: icmp_seq=10 ttl=64 time=0.138 ms

--- 10.0.0.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9202ms
rtt min/avg/max/mdev = 0.047/0.125/0.269/0.068 ms
root@h3:/tmp/pycore.40447/h3.conf# ping -c 10 10.0.0.20
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.
64 bytes from 10.0.0.20: icmp_seq=1 ttl=64 time=1.27 ms
64 bytes from 10.0.0.20: icmp_seq=2 ttl=64 time=0.075 ms
64 bytes from 10.0.0.20: icmp_seq=3 ttl=64 time=0.334 ms
64 bytes from 10.0.0.20: icmp_seq=4 ttl=64 time=0.295 ms
64 bytes from 10.0.0.20: icmp_seq=5 ttl=64 time=0.339 ms
64 bytes from 10.0.0.20: icmp_seq=6 ttl=64 time=0.063 ms
64 bytes from 10.0.0.20: icmp_seq=7 ttl=64 time=0.094 ms
64 bytes from 10.0.0.20: icmp_seq=8 ttl=64 time=0.093 ms
64 bytes from 10.0.0.20: icmp_seq=9 ttl=64 time=0.057 ms
64 bytes from 10.0.0.20: icmp_seq=10 ttl=64 time=0.050 ms

--- 10.0.0.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9187ms
rtt min/avg/max/mdev = 0.050/0.267/1.275/0.354 ms
root@h3:/tmp/pycore.40447/h3.conf# 
```

Conclusion

Overall I believe I found and were able to sufficiently utilize the OpenFlow Framework in our pursuit in using python code to emulate a SDN, software-defined network simulation. However, it did take some adjusting to get used to the python format and I had to brief myself on the documentation before properly coding as I did struggle at times with remembering the syntax as it is quite different from c++ or java. The testing methodology I took was after running the code through the switch device I would then ping from hosts to the other depending on the goal, for example, when I executed the code from task 1 I tested its ability to block communications by pinging from host 2 to host 3 and pinging from host 3 to host 2. The results showed that the pings were unsuccessful. The other testing methodology I implemented was the use of print statements within my code loops which print to the switches terminal when executed. For example, for task 2 I needed to count the amount of times traffic passes through host 1 and when pinging from or to host 1 the count variable counts upwards each time shown on the switches terminal. However using my second implementation the initial values look quite large due to Ryu saving the hosts previous traffic history and I needed to account for this. For tasks 2 and 3 I implemented a few methods from the Traffic Monitor Github (seen in the references section) which allowed for me to access the hosts traffic memory which I found quite helpful. Although I didn't get my task 3 running as to how I would've liked I believe my attempt with some more configuring could have worked.

References

Python Tutorial,

- https://docs.python.org/3/tutorial/

Ryu Tutorial & Documentation,

- https://ryu.readthedocs.io/en/latest/writing_ryu_app.html

Ryu Packet Library,

- https://ryu.readthedocs.io/en/latest/library_packet.html

Traffic Monitor Code,

- https://osrg.github.io/ryu-book/en/html/traffic_monitor.html

Appendices

1. Appendix A - Software
   Windows Linux Subsystem, WSL & Ubuntu
   https://ubuntu.com/tutorials/install-ubuntu-on-wsl2-on-windows-11-with-gui-support#1-overview
   WinSCP 5.21.2, Windows 64bit
   https://winscp.net/eng/download.php
   PuTTY 0.77, Windows 64bit
   https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
   VirtualBox VM 6.1.38, Windows 64bit
   https://www.virtualbox.org/wiki/Downloads

2. Appendix B - Configurations
   Ryu SDN Framework
   https://ryu-sdn.org/
   Open vSwitch
   https://www.openvswitch.org/

3. Appendix B - Configurations
   Ryu SDN Framework
   https://ryu-sdn.org/
   Open vSwitch
   https://www.openvswitch.org/

Acknowledgements

- Tutor, Zach Kingsford, with coding and development help
- Professors, Winston Seah and Alvin Valera, for development and general networking knowledge