BÁO CÁO THỰC HÀNH

**Bài thực hành số 04: Lập trình hiện thực và test thử nghiệm một Network Monitor trên mạng SDN/OpenFlow**

**Môn học:** Công nghệ mạng khả lập trình

**Lớp:** NT541.P21.1

**THÀNH VIÊN THỰC HIỆN:**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** |
| 1 | Phạm Thiều Gia Khang | 21520967 |
|  |  |  |
|  |  |  |

|  |
| --- |
| **Điểm tự đánh giá** |
| **9/10** |

**ĐÁNH GIÁ KHÁC:**

|  |  |
| --- | --- |
| Tổng thời gian thực hiện | 6 ngày |
| Phân chia công việc |  |
| Ý kiến *(nếu có)*  + Khó khăn  + Đề xuất, kiến nghị |  |

Phần bên dưới của báo cáo này là báo cáo chi tiết của nhóm thực hiện

MỤC LỤC

[**A.** BÁO CÁO CHI TIẾT 3](#_Toc197808583)

[1. Tạo mạng SDN/OpenFlow với Topology tuỳ ý: 3](#_Toc197808584)

[a. Khởi động Mininet: 3](#_Toc197808585)

[b. Viết chương trình tạo mạng SDN/OpenFlow với topology như hình 1 3](#_Toc197808586)

[c. Test mạng SDN OpenFlow vừa tạo ra: 4](#_Toc197808587)

[2. Viết chương trình để hiện thực một network monitor trên mạng SDN/OpenFlow: 5](#_Toc197808588)

[3. Tiến hành test chương trình với lưu lượng network traffic lớn: 9](#_Toc197808589)

[a. Sw1: Đóng vai trò relay chính cho dòng lưu lượng lớn. 9](#_Toc197808590)

[b. Sw2: Transit node. 10](#_Toc197808591)

[c. Sw3: Transit node. 10](#_Toc197808592)

[d. Sw4: Xử lý hướng ngược của luồng iperf3. 10](#_Toc197808593)

# BÁO CÁO CHI TIẾT

## Tạo mạng SDN/OpenFlow với Topology tuỳ ý:

### Khởi động Mininet:

Khởi tạo môi trường bằng conda và khởi động mininet:

$ conda activate ryuNet

### Viết chương trình tạo mạng SDN/OpenFlow với topology như hình 1

Cấu hình mô hình mạng gồm 4 Switch và 16 host theo hình vẽ bằng thư viện mininet:



Hình 1: Mô hình lab 4

Code:

from mininet.net import Mininet

from mininet.cli import CLI

from mininet.log import setLogLevel

from mininet.topo import Topo

from mininet.node import RemoteController, OVSKernelSwitch

class Lab3\_Topo(Topo):

def build(self):

# Add hosts

hosts = [self.addHost(f'h{i}') for i in range(1, 17)]

# Add switches && use cls=OVSKernelSwitch

switches = [self.addSwitch(f's{i}', cls=OVSKernelSwitch) for i in range(1, 5)]

# Link switches in chain

for i in range(len(switches) - 1):

self.addLink(switches[i], switches[i + 1])

# Link hosts to switches (4 hosts per switch)

for i, host in enumerate(hosts):

self.addLink(host, switches[i // 4])

def run():

topo = Lab3\_Topo()

net = Mininet(topo=topo, controller=None)

net.addController('c0', controller=RemoteController, ip='127.0.0.1', port=6633)

net.start()

CLI(net)

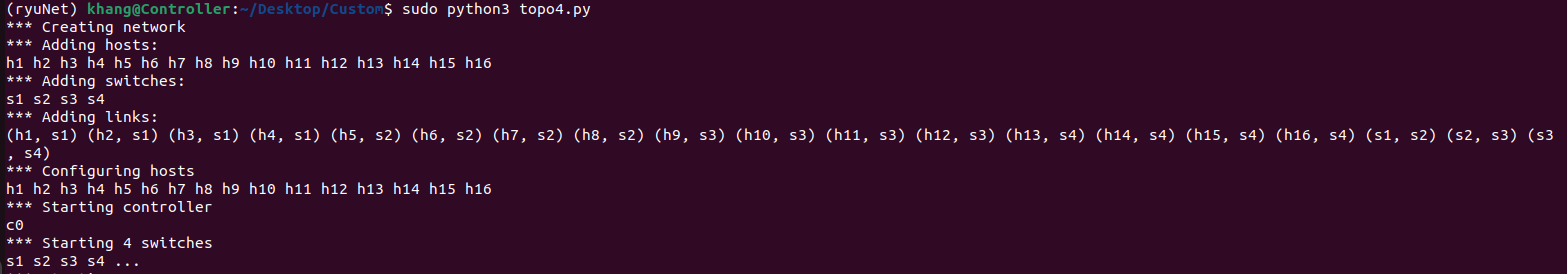
net.stop()

if \_\_name\_\_ == '\_\_main\_\_':

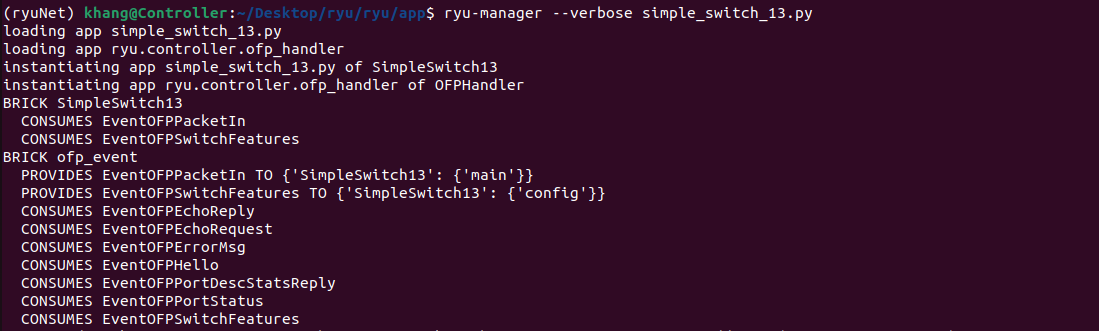
setLogLevel('info')

run()

File cấu hình trên sẽ tạo ra 1 tôp có dạng như Hình trên:

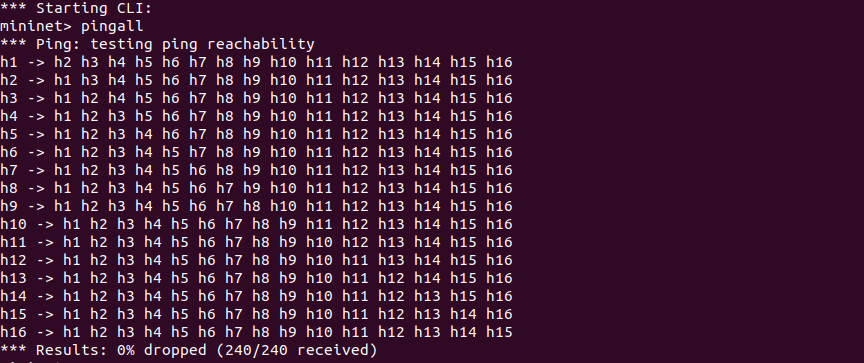


Khởi động Ryu Controller:

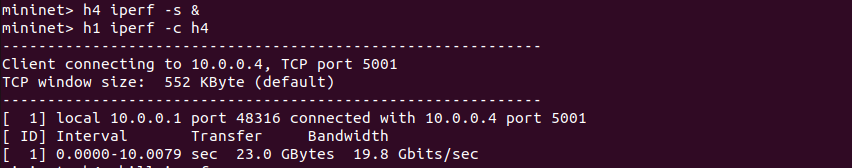


### Test mạng SDN OpenFlow vừa tạo ra:

* Test kết nối bằng lệnh pingall:



* Test hiệu suất bằng iperf giữa host1 và host4, mở 1 iperf server chạy nền trên host4, ping tới từ host1 để kiểm tra:



## Viết chương trình để hiện thực một network monitor trên mạng SDN/OpenFlow:

Tạo một chương trình Learning Switch dựa trên simple\_switch\_13 của Ryu, đăng ký gửi và nhận các thông tin mạng về Flow và Port của các Switch, nội dung code trong file learning\_switch.py đính kèm:

Hàm “\_monitor”: là một luồng chạy liên tục trong nền, được khởi tạo khi ứng dụng bắt đầu. Nhiệm vụ của nó là gửi yêu cầu thống kê đến tất cả các switch đã được đăng ký trong danh sách `self.datapaths`. Sau mỗi lần gửi yêu cầu, nó sẽ tạm dừng trong 30 giây (được điều chỉnh bởi `hub.sleep(30)`) trước khi lặp lại.

def \_monitor(self):

while True:

for dp in list(self.datapaths.values()):

self.\_request\_stats(dp)

hub.sleep(30) # Frequecency

Hàm “\_request\_stats” được gọi bởi “\_monitor” để gửi yêu cầu thống kê đến một switch cụ thể. Nó gửi hai loại yêu cầu:

* “OFPFlowStatsRequest”: Yêu cầu thông tin về các luồng (flows) đang được xử lý bởi switch.
* “OFPPortStatsRequest”: Yêu cầu thông tin về các cổng (ports) của switch, bao gồm số lượng gói tin nhận/gửi và kích thước dữ liệu.

def \_request\_stats(self, datapath):

self.logger.info("Sending stats request to datapath: %016x", datapath.id)

ofproto = datapath.ofproto

parser = datapath.ofproto\_parser

# Request flow stats

req = parser.OFPFlowStatsRequest(datapath)

datapath.send\_msg(req)

# Request port stats

port\_req = parser.OFPPortStatsRequest(datapath, 0, ofproto.OFPP\_ANY)

datapath.send\_msg(port\_req)

Hàm “\_flow\_stats\_reply\_handler”: xử lý phản hồi từ switch khi nhận được thông tin thống kê về các luồng (flows). Nó ghi lại thông tin chi tiết của từng luồng, bao gồm:

* “match”: Điều kiện để nhận diện luồng.
* “duration”: Thời gian luồng đã tồn tại.
* “packets”: Số lượng gói tin đã được xử lý bởi luồng.
* “bytes”: Tổng kích thước dữ liệu của các gói tin trong luồng.

# Print statics flow reply on terminal

@set\_ev\_cls(ofp\_event.EventOFPFlowStatsReply, MAIN\_DISPATCHER)

def \_flow\_stats\_reply\_handler(self, ev):

self.logger.info("Flow Stats Reply from switch %016x", ev.msg.datapath.id)

for stat in ev.msg.body:

self.logger.info(" Flow: match=%s, duration=%ds, packets=%d, bytes=%d",

stat.match, stat.duration\_sec, stat.packet\_count, stat.byte\_count)

Hàm “\_port\_stats\_reply\_handler”: xử lý phản hồi từ switch khi nhận được thông tin thống kê về các cổng (ports). Nó ghi lại thông tin chi tiết của từng cổng, bao gồm:

“port\_no”: Số hiệu của cổng.

* “rx\_packets” và “tx\_packets”: Số lượng gói tin nhận và gửi qua cổng.
* “rx\_bytes” và “tx\_bytes”: Tổng kích thước dữ liệu nhận và gửi qua cổng.

# Print statics port reply on terminal

@set\_ev\_cls(ofp\_event.EventOFPPortStatsReply, MAIN\_DISPATCHER)

def \_port\_stats\_reply\_handler(self, ev):

self.logger.info("Port Stats Reply from switch %016x", ev.msg.datapath.id)

for stat in ev.msg.body:

self.logger.info(" Port %d: rx\_packets=%d tx\_packets=%d rx\_bytes=%d tx\_bytes=%d",

stat.port\_no,

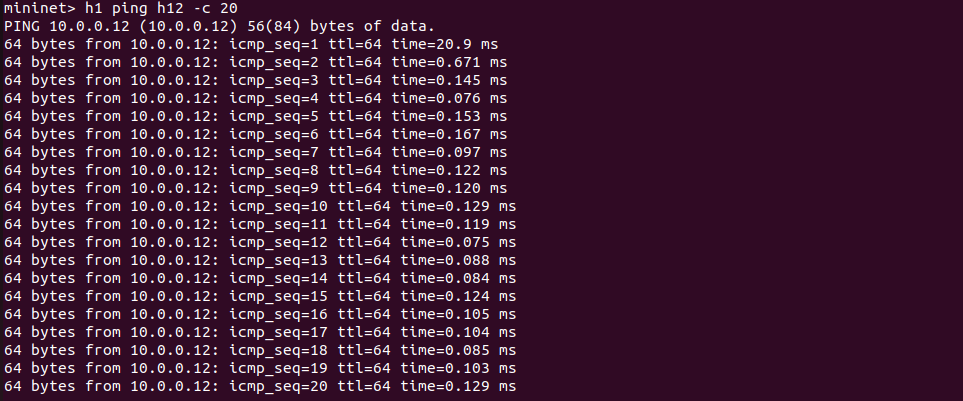
stat.rx\_packets, stat.tx\_packets,

stat.rx\_bytes, stat.tx\_bytes)

Chạy Controller bằng cờ --verbose để in ra toàn bộ gói tin:

$ryu-manager --verbose learning\_switch.py

Kiểm tra hệ thống, thực hiện ping 20 gói giữa 2 host h1 và h12 và kiểm tra Terminal:

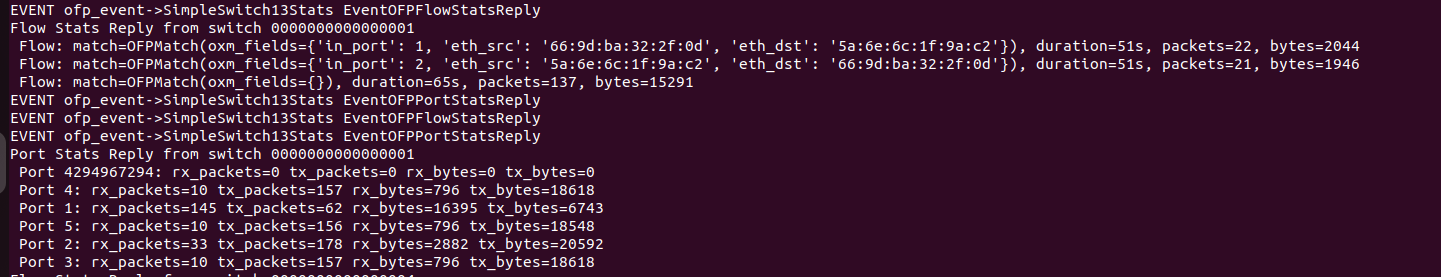


Trên Terminal controller, khi bắt đầu phiên truyền, Controller sẽ gửi các gói Request đến các Switch để đăng ký nhận thông tin mạng (ở đây là thông tin về Flow và Port):



Sau khi các Switch nhận phản hồi sẽ trả về EventOFPFlowStatsReply, Controller sẽ đọc thông tin trong dữ liệu trả về và in ra màn hình:

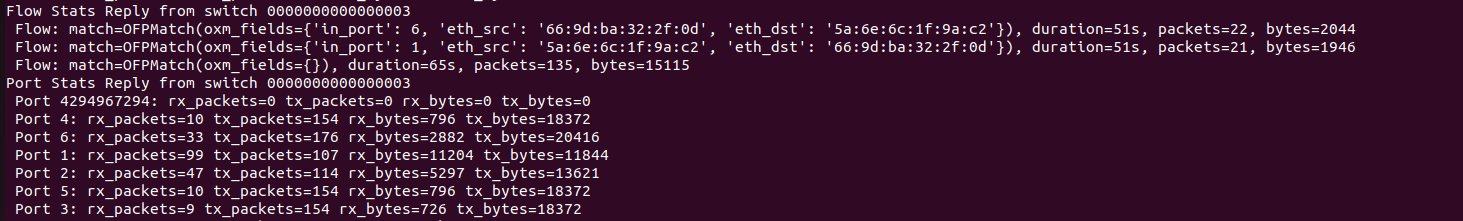
* Sw1:



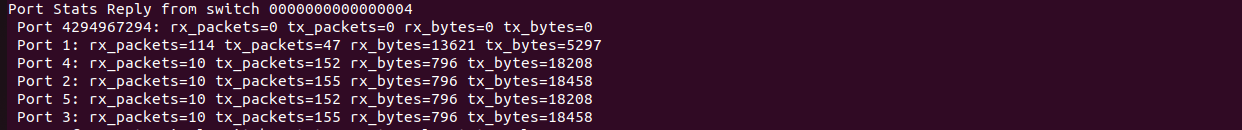
* Sw2:



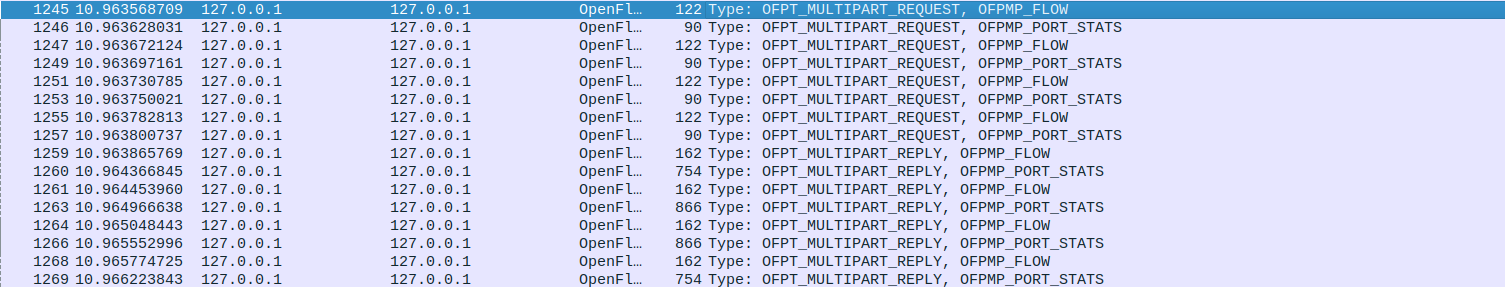
* Sw3:



* Sw4:

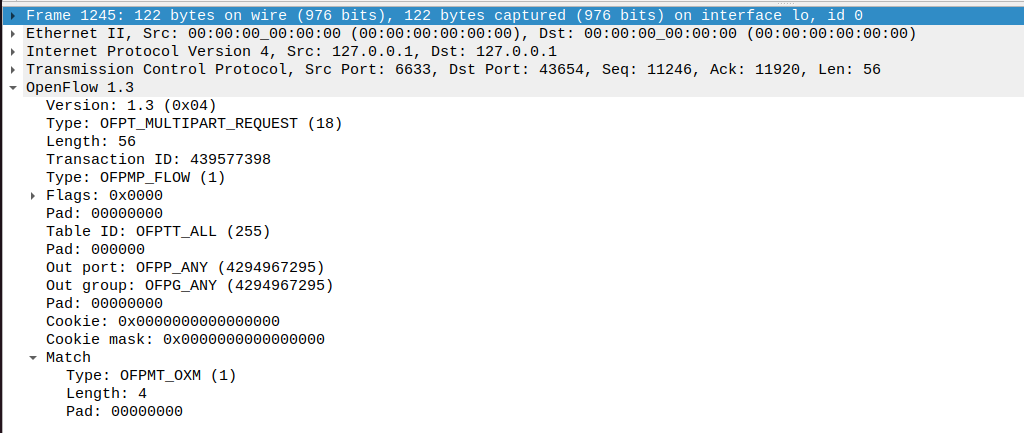


Sử dụng Wireshark để bắt các gói tin Statics, ta sẽ bắt được 2 gói chính đó là OFPT\_MULTIPART\_REQUEST là yêu cầu từ Controller đến các Switch gửi các thông tin mạng, các thông tin này sẽ được biểu thị qua loại gói là OFPTMP\_FLOW (gửi thông tin flow) hay OFPTMP\_PORT\_STATS (gửi thông tin port). Các Switch sẽ trả lời bằng các gói OFPT\_MULTIPART\_REPLY.

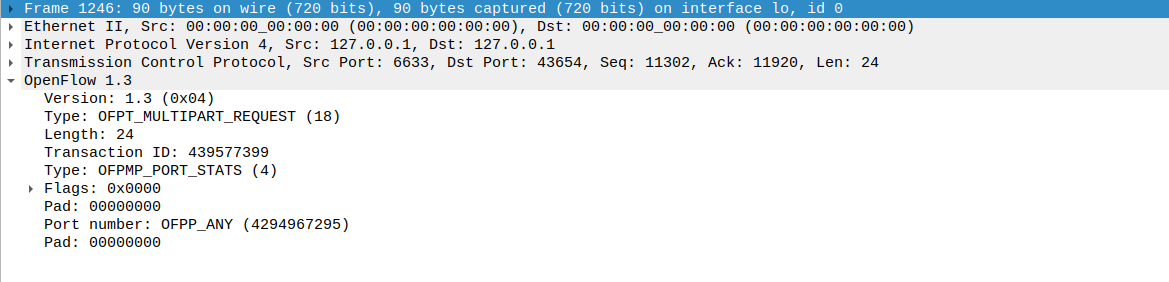


Nội dung các gói tin:

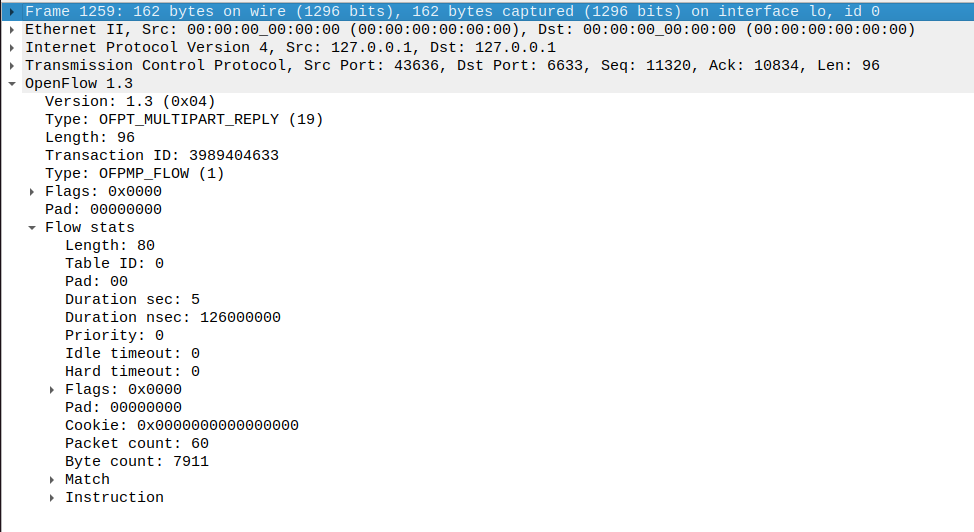
* OFPT\_MULTIPART\_REQUEST, OFPTMP\_FLOW:



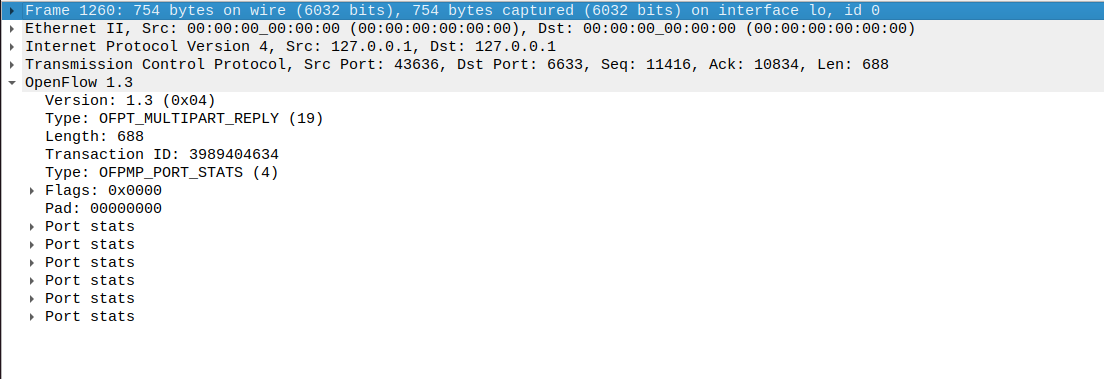
* OFPT\_MULTIPART\_REQUEST, OFPTMP\_PORT\_STATS:



* OFPT\_MULTIPART\_REPLY, OFPTMP\_FLOW:



* OFPT\_MULTIPART\_REPLY, , OFPTMP\_PORT\_STATS:



## Tiến hành test chương trình với lưu lượng network traffic lớn:

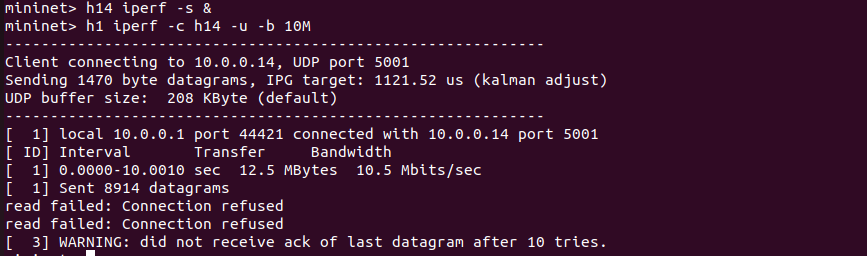
Sử dụng lại mô hình Controller tỏng bài 2, dùng iperf để gửi 1 lướng lớn gói tin từ h1 đến h14 và quan sát Controller.

Mở iperf Server ở h14:

h14 iperf -s &

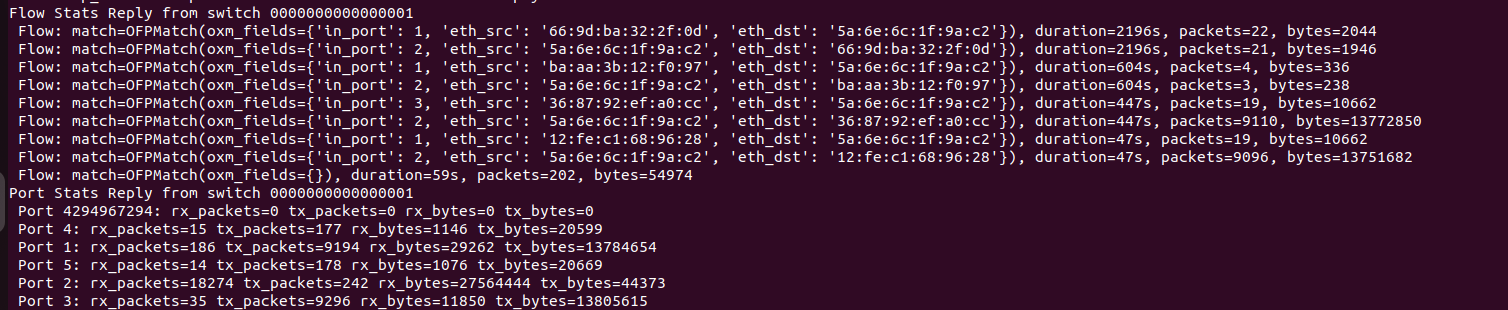
Ping tới từ h1 với giao thức UDP và bang thông 10Mbps:

h1 iperf -c h2 -u -b 10M

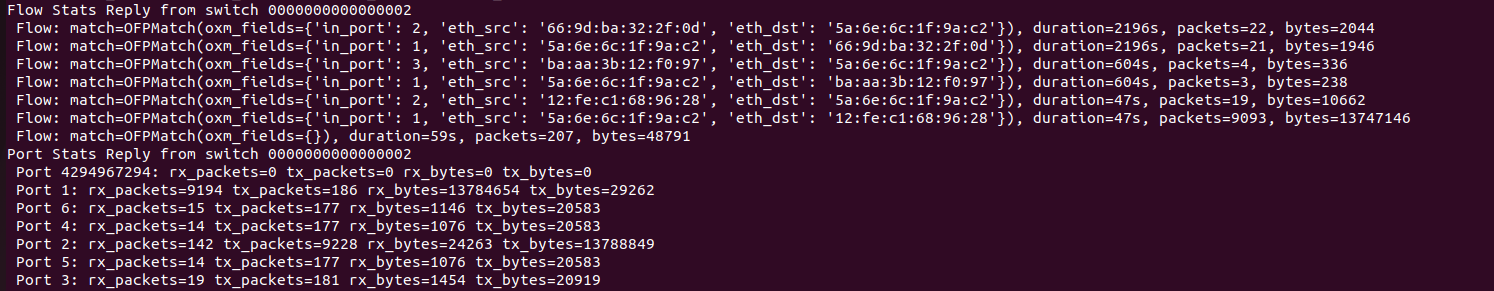


Kiểm tra Controller:

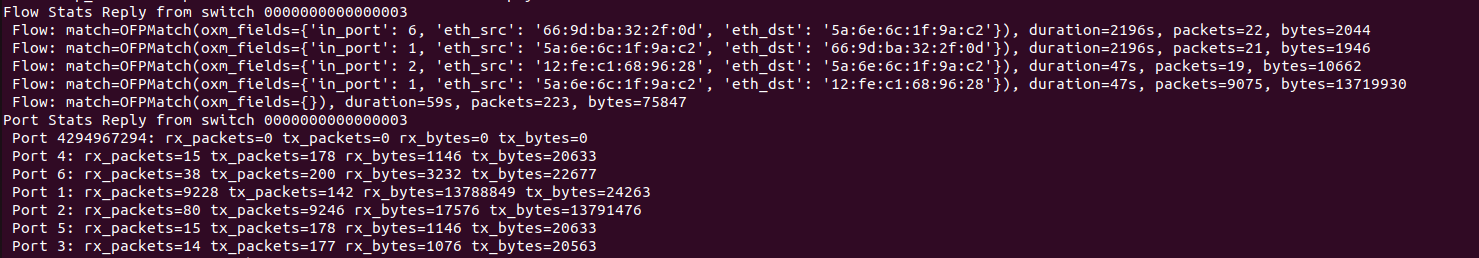
### Sw1: Đóng vai trò relay chính cho dòng lưu lượng lớn.



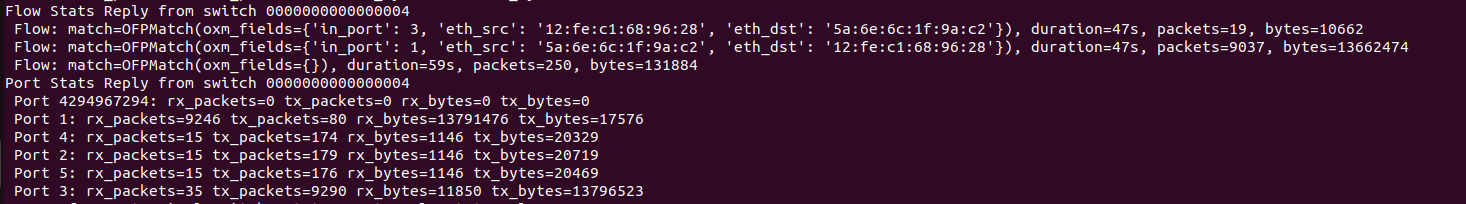
### Sw2: Transit node.



### Sw3: Transit node.



### Sw4: Xử lý hướng ngược của luồng iperf3.



Dựa trên log, hệ thống SDN với bốn switch cho thấy luồng chính giữa các thiết bị với địa chỉ MAC 5a:6e:6c:1f:9a:c2 và 12:fe:c1:68:96:28. Lưu lượng mạng chủ yếu là các gói TCP, đặc biệt là từ luồng iperf3. Các switch 1 và 4 xử lý phần lớn lưu lượng, với Switch 1 đóng vai trò chính trong việc truyền tải dữ liệu từ thiết bị trung tâm đến các thiết bị khác, đặc biệt là qua Port 2. Các switch còn lại (2 và 3) chủ yếu tham gia chuyển tiếp gói giữa các thiết bị. Tổng lưu lượng mạng là khoảng 55 MB với một số luồng tồn tại lâu dài, nhưng chỉ có ít gói dữ liệu phản hồi, cho thấy phần lớn là download từ server iperf3.