BÁO CÁO THỰC HÀNH

Bài thực hành số 03: Lập trình mạng SDN/OpenFlow trong Mininet

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

**Lớp:** NT541. P21.2

**THÀNH VIÊN THỰC HIỆN (Nhóm xx):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** |  | **Điểm tự đánh giá** |
| 1 | Phạm Thiều Gia Khang | 21520967 | **10/10** |

**ĐÁNH GIÁ KHÁC:**

|  |  |
| --- | --- |
| Tổng thời gian thực hiện | 1 tuần |
| 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](#_Toc195566220)

[1. Tạo mạng SDN/OpenFlow với Topology tuỳ ý. 3](#_Toc195566221)

[a. Cài đặt Mininet: 3](#_Toc195566222)

[b. Viết chương trình tạo mạng SDN/OpenFlow với topology như hình 1: 3](#_Toc195566223)

[2. Test mạng SDN/OpenFlow được tạo ra, gồm: test kết nối, test hiệu suất của liên kết giữa hai host bất kỳ trong mạng 4](#_Toc195566224)

[a. Test kết nối: 4](#_Toc195566225)

[b. Kiểm tra hiệu suất : 5](#_Toc195566226)

[3. Mở Wireshark, tiến hành bắt các gói tin OpenFlow trao đổi giữa Controller và các Switch trong 2 trường hợp: 5](#_Toc195566227)

[a. Ping từ H1 đến H4: 5](#_Toc195566228)

[b. Ping từ h1 đến h16: 8](#_Toc195566229)

[4. Cài đặt OpenvSwitch và chạy thử mạng SDN/OpenFlow với OpenvSwitch (homework): 9](#_Toc195566230)

[a. Cài đặt OpenvSwitch: 9](#_Toc195566231)

[b. Chạy thử mạng với ovs: 9](#_Toc195566232)

[B. TÀI LIỆU THAM KHẢO 13](#_Toc195566233)

# BÁO CÁO CHI TIẾT

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

### Cài đặt Mininet:

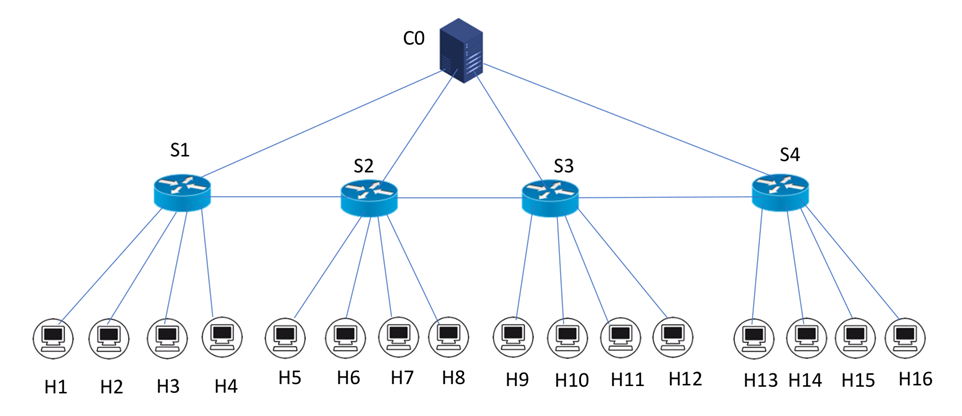
Cài đặt mininet từ Repo Github:

$ git clone https://github.com/mininet/mininet

$ cd mininet

$ ./util/install.sh

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



Mô hình bao gồm 1 controller, 4 Switch và 16 host. Chương trình sẽ được mô phỏng thông qua đoạn code Python sau:

from mininet.net import Mininet

from mininet.node import RemoteController

from mininet.cli import CLI

from mininet.log import setLogLevel

from mininet.topo import Topo

class Lab3\_Topo(Topo):

def build(self):

# Add hosts

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

# Add switches

switches = [self.addSwitch(f's{i}') 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()

Chạy chương trình vừa tạo:



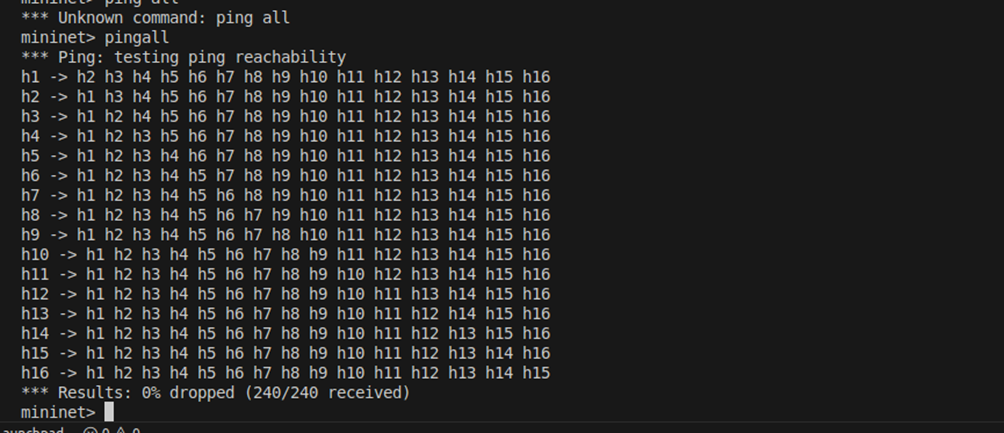
## Test mạng SDN/OpenFlow được tạo ra, gồm: test kết nối, test hiệu suất của liên kết giữa hai host bất kỳ trong mạng

### Test kết nối:

Khởi động Ryu Cotroller:

$ ryu-manager ryu.app.simple\_switch

Sử dụng lên ping all để kiểm tra kết nối:

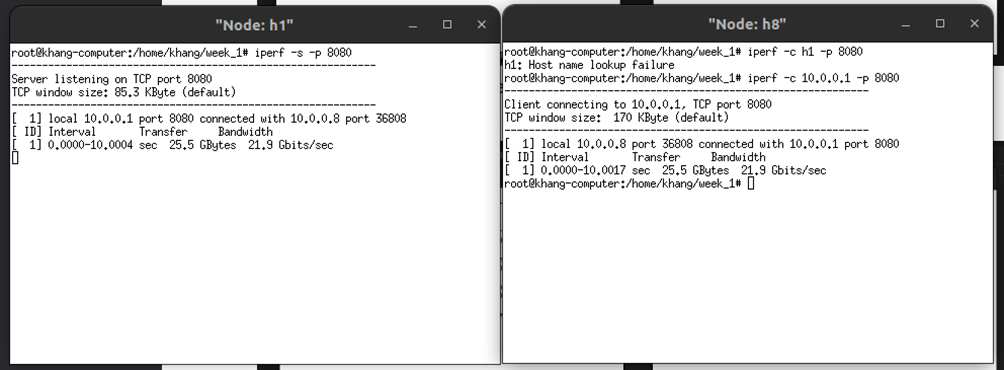


### Kiểm tra hiệu suất :

Kiểm tra hiệu suất giữa H1 và H8, mở CLI của 2 Node bằng lệnh

mininet > xterm h1 h8

Mở iperf Server trên H1 và ping tới từ H8:



=> Hai Host ping thành công và tốc độ mạng khá cao, 21,9 Gbits/s.

## Mở Wireshark, tiến hành bắt các gói tin OpenFlow trao đổi giữa Controller và các Switch trong 2 trường hợp:

Mở wireshark:

$ sudo wireshark

### Ping từ H1 đến H4:

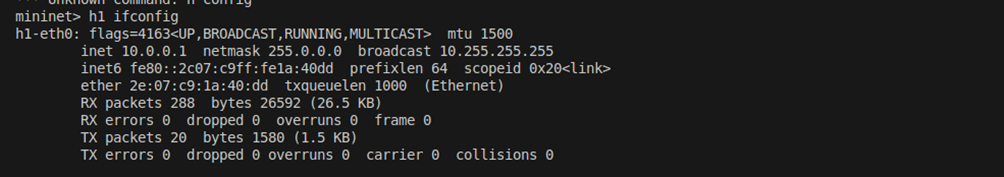
Đầu tiên kiểm tra địa chỉ Mac của h1, h4 và s4 bằng lệnh:

>h1 ifconfig

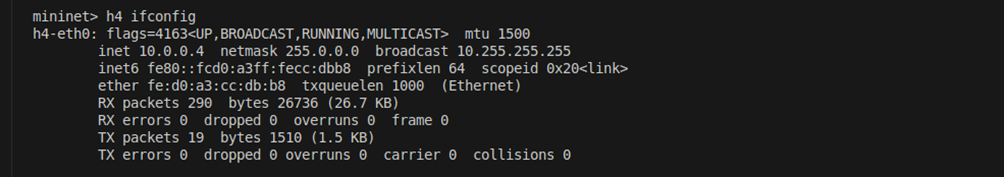
>h4 ifconfig

>s1 ifconfig

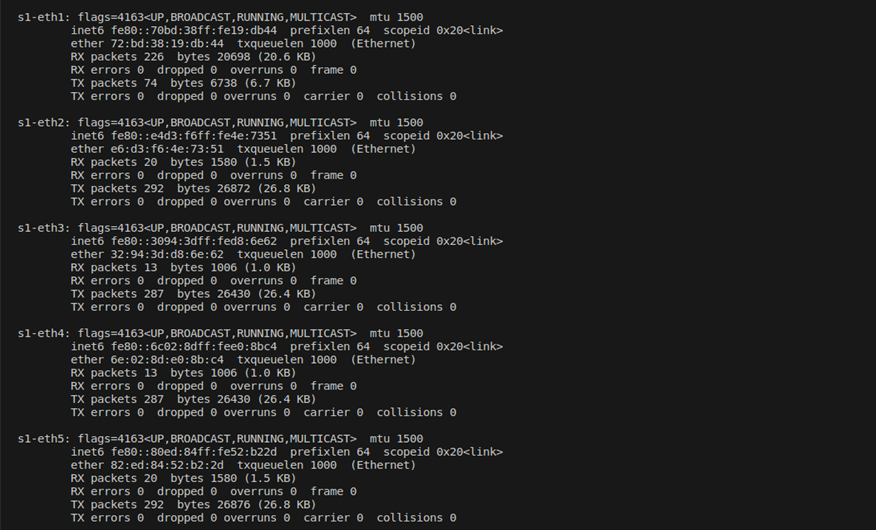
Địa chỉ Mac của h1 là: 2e:07:c9:1a:40:dd



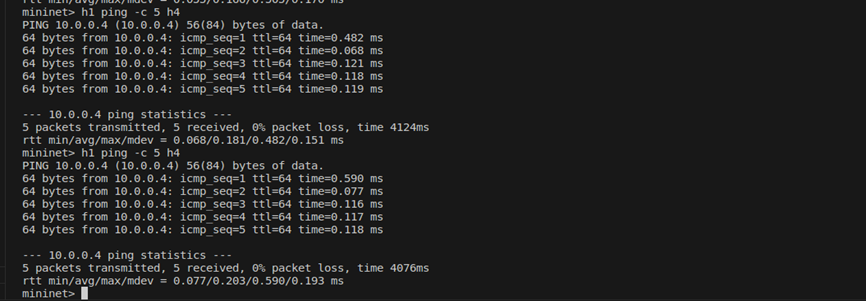
Địa chỉ Mac của h4 là: fe:d0:a3:cc:db:b8



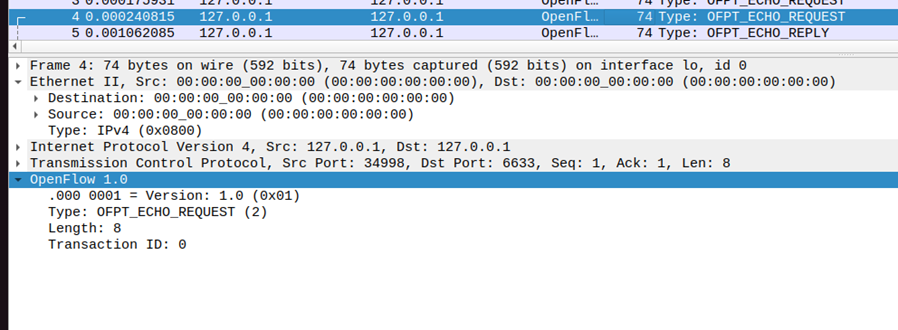
Đối với s1, sẽ có 5 địa chỉ Mac tương ứng với 5 cổng kết nối eth:



Dùng lênh h1 ping -c 5 h4 để gửi 5 gói tin từ h1 đến h4:



Trên wireshark đầu tiên ta sẽ bắt được các gói OFTP\_ECHO\_REQUEST và OFTP\_ECHO\_REPLY, đây là các gói tin mà ryu controller và mininet gửi tuần tự cho nhau để duy trì kết nối.



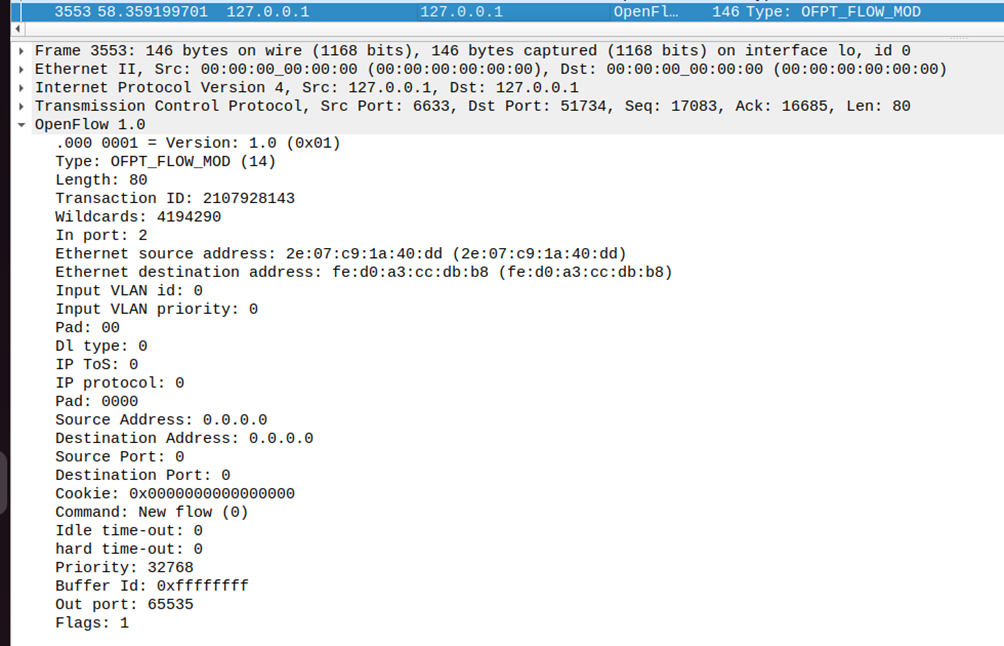
Đầu tiên khi h1 ping tới h4, s1 sẽ nhận được gói tin ARP này nhưng không biết xử lý ra sao, nó sẽ gửi một bản sao của gói đó lên controller dưới dạng OFPT\_PACKET\_IN.Mục đích báo cho controller biết có một gói cần xử lý (do không match rule nào). Cho phép controller ra quyết định xử lý (gửi tiếp, drop, học địa chỉ…). Sau khi controller xử lý PACKET\_IN, nó có thể phản hồi bằng PACKET\_OUT để yêu cầu switch gửi tiếp gói tin đến port đích hoặc cài thêm rule nếu cần.

  
 Tiếp theo Controller gửi gói OFPT\_FLOW\_MOD xuống switch:

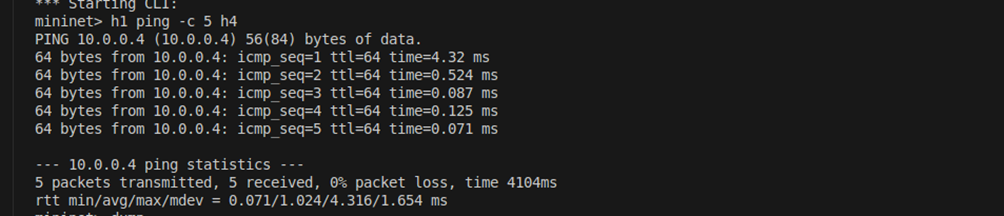
* Match: src IP = A, dst IP = B
* Action: output → port 2

Switch cài rule vào bảng flow.

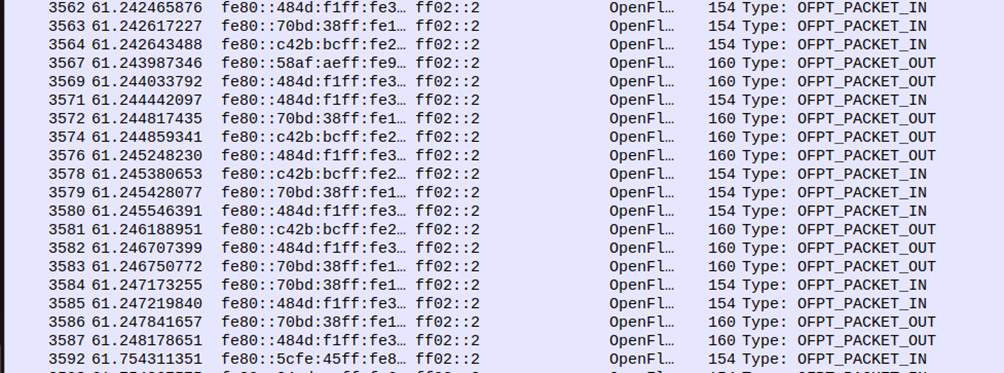
Các gói tiếp theo từ A đến B sẽ được xử lý ngay tại switch, không gửi lên controller nữa.



Sau khi thêm rule thành công, 2 host sẽ trong đổi các gói ICMP như bình thường:



Trên wire shark sẽ hiển thị quá trình gói tin quảng bá đi qua từng cổng của s1:

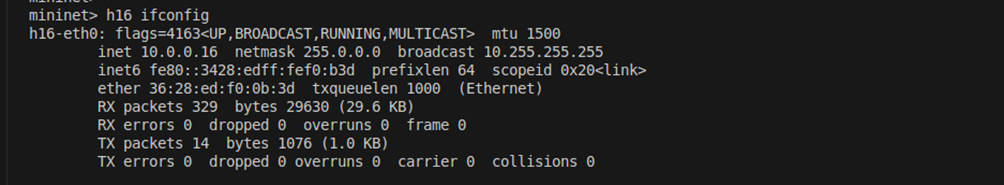


### Ping từ h1 đến h16:

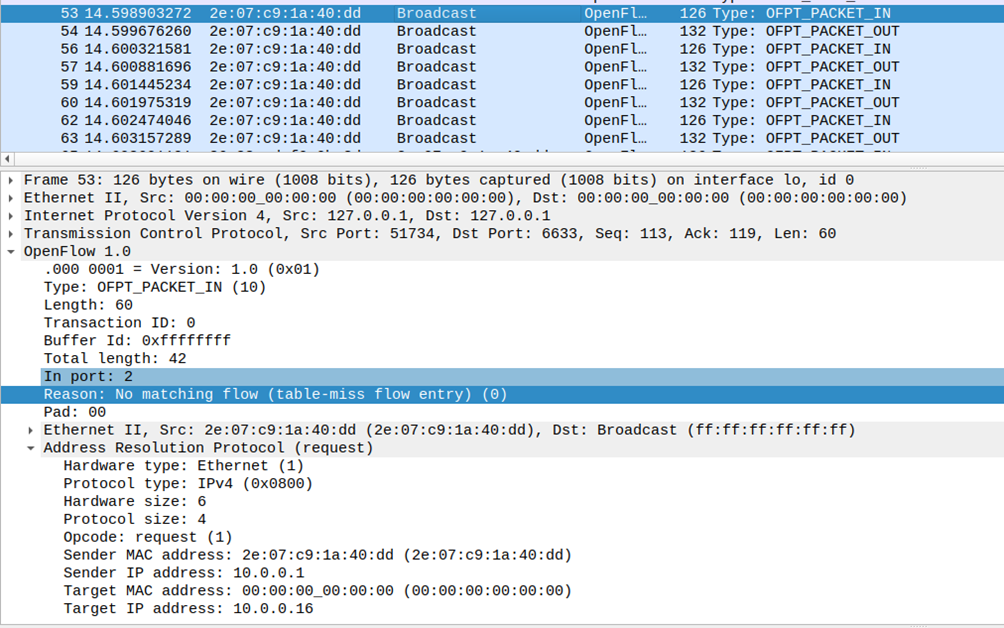
Kiểm tra cổng của h16:

> h16 ifconfig

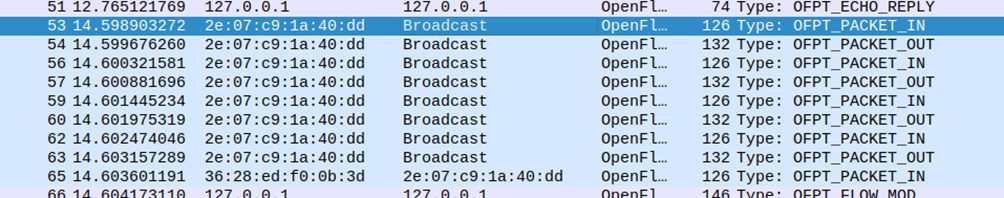
Địa chỉ MAC của h16 là 36:28:ed:f0:0b:3d:

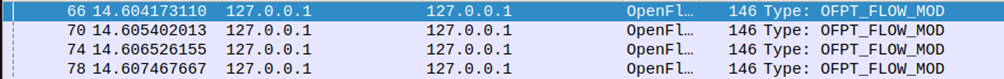


Khi thực hiện ping, s1 cũng sẽ sử dụng các gói OFTP\_PACKET\_IN và OFTP\_PACKET\_OUT để xin cúng cấp thông tin và rule của kết nối cho flow ip table:



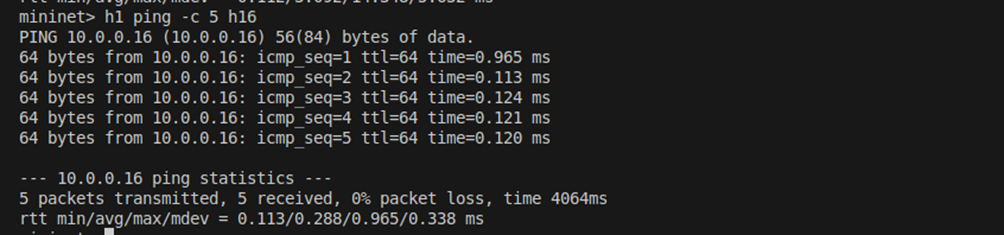
Điểm khác biệt ở đây đó chính là có tới 4 gói PACKET\_IN, 4 gói PACKET\_OUT và 4 FLOW\_MOD:





Các gói định tuyến này được gửi đều cho cả 4 Switch trong mạng để cung cấp thông tin về kết nối giữa h1 và h16. Các gói tiếp theo từ h1 đến h16 sẽ được xử lý ngay tại switch, không gửi lên controller nữa.

Quá trình giao tiếp giữa h1 và h16 sẽ diễn ra bình thường sau khi được phân rule:



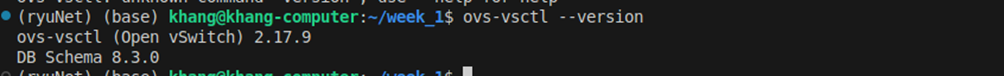
## Cài đặt OpenvSwitch và chạy thử mạng SDN/OpenFlow với OpenvSwitch (homework):

### Cài đặt OpenvSwitch:

Cài đặt OpenvSwitch và kiểm tra cài đặt:

$ sudo apt install openvswitch–switch

$ ovs-vsctl –version



### Chạy thử mạng với ovs:

Chỉnh sửa code tích hợp với OpenvSwitch:

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)

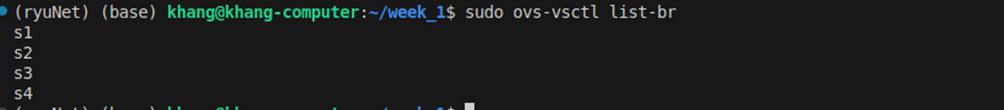
net.stop()

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

setLogLevel('info')

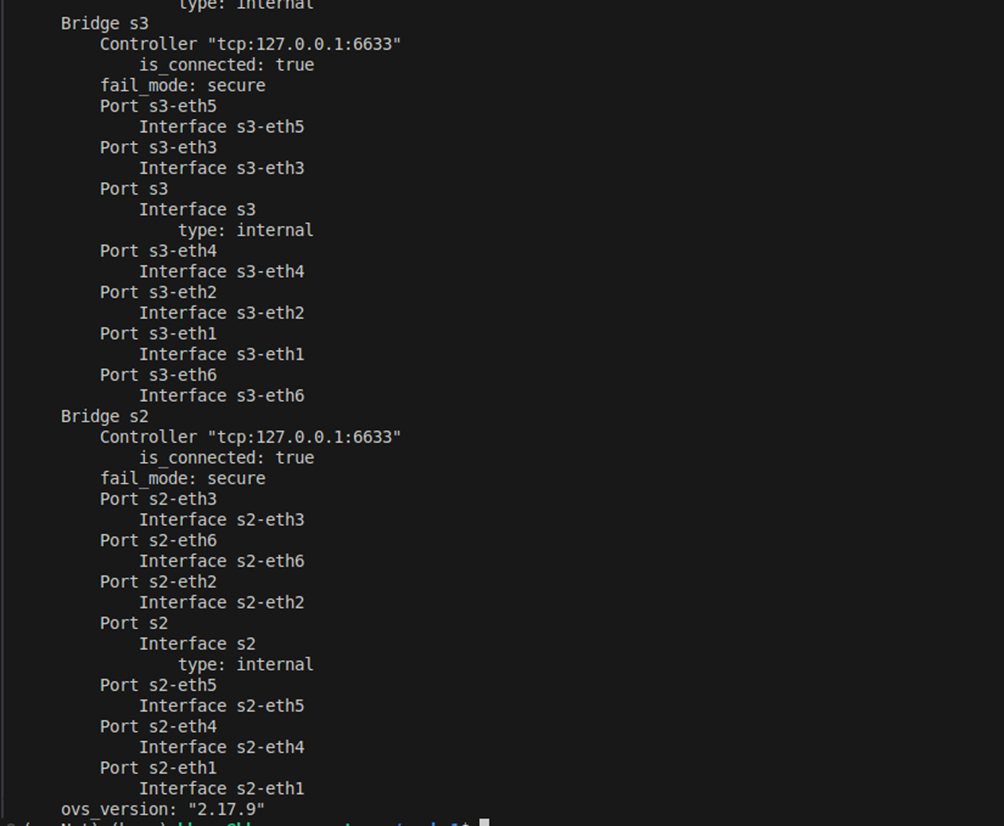
run()

Chạy file cấu hình trên và kiểm tra, ta thấy 4 switch s1, s2, s3 và s4 đã được tạo :

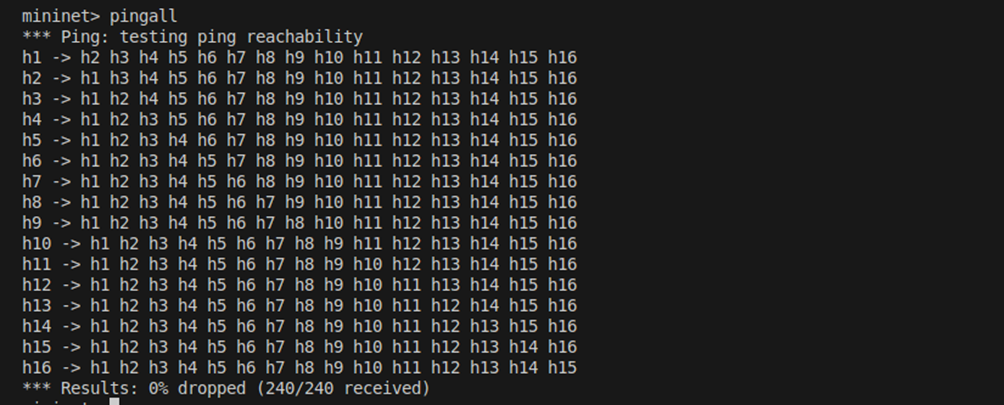


Kiểm tra trạng thái các switchs :

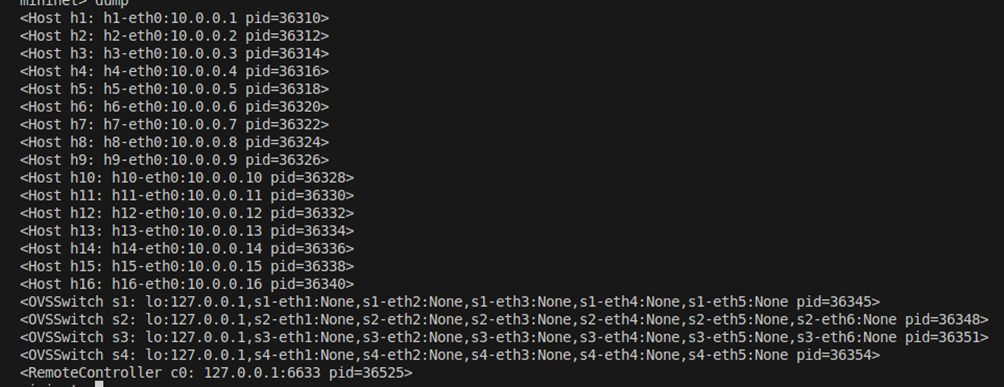




Ta thấy tất cả các switch đều đang hoạt dộng ổn định, tiếp theo sử dụng pingall để kiểm tra hệ thống:



Sử dụng dump để kiểm IP:



# TÀI LIỆU THAM KHẢO

[1] Open vSwitch, “Downloads,” https://www.openvswitch.org/download/ (accessed Apr. 14, 2025).

[2] Mininet, “Mininet: An Instant Virtual Network on your Laptop,” https://mininet.org/ (accessed Apr. 14, 2025).

[3] Ryu SDN Framework, “Home,” https://ryu-sdn.org/ (accessed Apr. 14, 2025).