TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN, ĐHQG-HCM

KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

BÁO CÁO ĐỒ ÁN MÔN HỌC

ĐỀ TÀI: TÌM HIỂU VÀ TRIỂN KHAI MẠNG SDN (SOFTWARE-DEFINED NETWORK)

Môn học: NT132.Q12.ANTT – Quản trị mạng và hệ thống

Giảng viên hướng dẫn: ThS. Đỗ Hoàng Hiển

Thực hiện bởi nhóm 5, bao gồm:

1.	Dương Phước Nhật Nam	23520968	Trưởng nhóm
2.	Nguyễn Gia Luân	23520896	Thành viên
3.	Lương Hoàng Long	23520879	Thành viên
4.	Lê Minh	23520928	Thành viên

Thời gian thực hiện: 27/9/2025 - 21/12/2025

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành nhất đến Thầy Đỗ Hoàng Hiển về sự tận tâm và hướng dẫn chúng em nhiệt tình trong môn học Quản trị mạng và hệ thống. Nhờ sự hỗ trợ của Thầy, chúng em đã không chỉ nắm vững lý thuyết về mạng truyền thống và SDN, mà còn biết cách áp dụng nó vào quá trình thực hiện đồ án. Qua việc làm đồ án, chúng em cũng đã hiểu rõ hơn về cách vận hành của mạng máy tính nói chung và mạng SDN nói riêng, cũng như là củng cố các kỹ năng mềm, gia tăng khả năng giao tiếp và làm việc trong nhóm.

Sự tận tụy và sự hỗ trợ của Thầy đã tạo nên một môi trường học tập tích cực giúp chúng em được đi sâu tìm hiểu, cập nhật những kiến thức và mô hình mạng mới. Chúng em cũng xin bày tỏ lòng biết ơn đối với sự động viên và những lời khuyên quý giá từ Thầy trong quá trình làm đồ án. Thầy đã luôn sẵn lòng lắng nghe và giúp đỡ chúng em vượt qua những khó khăn.

Chúng em xin cam đoan sẽ tiếp tục áp dụng những kiến thức đã học từ Thầy vào công việc và cuộc sống hàng ngày. Sự hướng dẫn và đóng góp của Thầy sẽ mãi là nguồn động lực và định hướng cho chúng em trong tương lai.

Nhóm 5

MỤC LỤC

DANH SÁCH HÌNH

DANH SÁCH BẢNG

TÓM TẮT

Đề tài này tiến hành tìm hiểu về sự khác nhau giữa mạng SDN và mạng truyền thống, tổng quan về mạng SDN, các trường hợp ứng dụng của mạng SDN, kiến trúc tổng quan của mạng SDN, giao thức OpenFlow và cơ chế hoạt động của nó thông qua việc triển khai mạng SDN và điều khiển luồng của các gói tin trong mạng đó.

Chương I. TỔNG QUAN

1. Giới thiệu đề tài

Trước bối cảnh cơ sở hạ tầng mạng ngày càng phức tạp, những đòi hỏi về khả năng quản lý mạng linh hoạt và tự động hóa đang trở nên quan trọng hơn bao giờ hết. Software-Defined Network (SDN) chính là một cách tiếp cận khác với mạng truyền thống, cung cấp các giải pháp cho những hạn chế đang có của mạng truyền thống. Chính vì vậy, đề tài của nhóm sẽ đi từ lý thuyết đến thực hành, từ việc tìm hiểu tổng quan về mạng SDN, sự khác nhau cơ bản giữa mạng truyền thống và SDN, cho đến tìm hiểu về cách hoạt động của mạng SDN thông qua việc triển khai và cấu hình mạng SDN. Từ đó, có thể đề xuất thêm các hướng phát triển của đề tài.

2. Cơ sở lý thuyết

2.1. Mininet

Mininet là một công cụ tạo ra mạng ảo, đồng thời cho phép chúng ta nghiên cứu và phát triển mạng SDN.

2.2. ONOS (Open Network Operating System)

ONOS là công cụ cung cấp mức điều khiển trong mạng SDN. Ngoài ra, ONOS còn cung cấp các ứng dụng quản lý mạng được triển khai, giao diện dòng lệnh (CLI) và giao diện đồ họa (GUI), giúp cho việc quản lý mạng SDN trở nên dễ dàng và trực quan hơn.

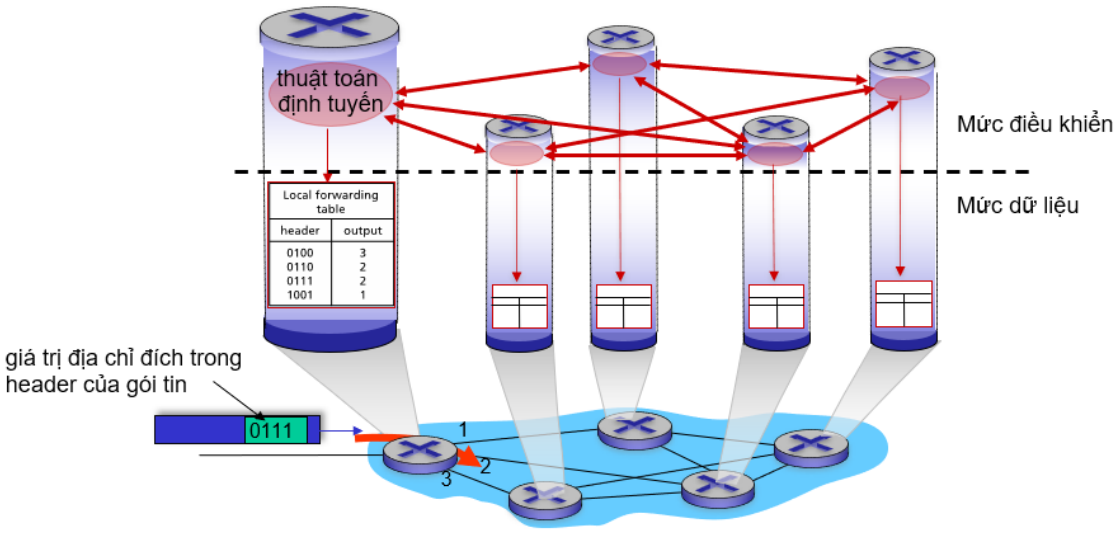
2.3. VirtualBox

VirtualBox là phần mềm ảo hóa, cho phép chạy các hệ điều hành khác nhau bên trong máy chính của chúng ta. Đây cũng chính là phương tiện để chúng ta sử dụng Mininet và ONOS để triển khai và cấu hình mạng SDN. Đề tài sử dụng máy ảo Ubuntu được cung cấp sẵn tại hướng dẫn sử dụng ONOS.

Chương II. LÝ THUYẾT

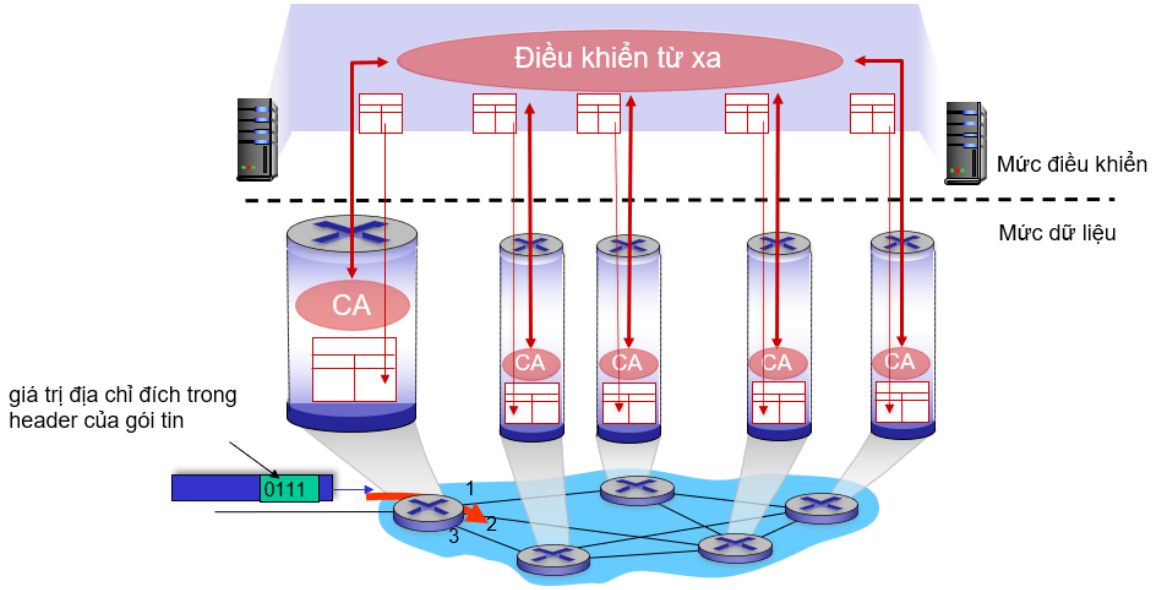
1. Tổng quan về mạng SDN

Trước tiên, chúng ta cần nhìn vào mạng truyền thống. Có thể thấy, các thiết bị mạng đều sẽ có mức điều khiển và mức dữ liệu. Mức điều khiển quyết định xem nên định tuyến và chuyển tiếp các gói tin trong một hệ thống mạng như thế nào, trong khi mức dữ liệu sẽ lưu vào những quyết định mà mức điều khiển đưa ra.



Hình 1: Mạng truyền thống

Tiếp theo, chúng ta sẽ nhìn vào mạng SDN. Các thiết bị mạng bây giờ không còn có cả hai mức điều khiển và mức dữ liệu nữa, mà mức điều khiển sẽ do một bộ điều khiển từ xa quản lý, các thiết bị mạng sẽ được quản lý bởi bộ điều khiển đó, và các thiết bị mạng sẽ chỉ còn đóng vai trò lưu trữ quyết định định tuyến và chuyển tiếp mà bộ điều khiển đưa ra.



Hình 2: Mạng SDN

⇨ Đây chính là cách thiết kế, cũng như là sự khác biệt của mạng truyền thống và SDN. Cụ thể, mạng SDN cung cấp một giải pháp phần mềm, chính là bộ điều khiển, đóng vai trò là mức điều khiển để quản lý mạng, từ đó tập trung sự điều khiển vào một trung tâm nhằm làm việc quản lý mạng trở nên dễ dàng và hiệu quả hơn.

2. Lý do sử dụng mạng SDN

Mạng SDN được thiết kế để giải quyết các nhược điểm trong mạng truyền thống, cụ thể các nhược điểm đó là:

- Cấu hình mạng phải thực hiện thủ công trên từng thiết bị mạng, khó có thể tự động hóa.
- Muốn cấu hình dịch vụ bổ sung như tường lửa, cân bằng tải,… phải tốn chi phí cài đặt thêm phần cứng chuyên dụng.
- Việc mở rộng hệ thống mạng khó khăn, kém linh hoạt.

Mạng SDN có thể giải quyết được các nhược điểm trên như sau:

- Vì có bộ điều khiển làm mức điều khiển cho các thiết bị mạng, ta có thể cấu hình mạng tại bộ điều khiển đó, và thay đổi sẽ được phản ánh trên hệ thống mạng.
- Vì là giải pháp phần mềm, và có thể lập trình được, nên việc cấu hình mạng và tự động hóa đơn giản hơn nhiều.
- Chi phí vận hành, bảo trì và mở rộng thấp hơn.
- Có thể cài đặt dịch vụ cho bộ điều khiển để phân phối cho hệ thống mạng một cách dễ dàng, có thể thấy rõ nhất ở bộ điều khiển ONOS.

3. Kiến trúc của mạng SDN

Kiến trúc mạng SDN có ba lớp: Lớp ứng dụng, lớp điều khiển, và lớp cơ sở hạ tầng. Lớp ứng dụng và lớp điều khiển giao tiếp với nhau qua API Bắc, trong khi lớp điều khiển và lớp cơ sở hạ tầng giao tiếp với nhau qua API Nam.

3.1. Lớp ứng dụng (Application Layer)

Đây là lớp chứa các ứng dụng mà mạng SDN có thể sử dụng như tường lửa, cân bằng tải,… Như đã đề cập, lớp ứng dụng sẽ tương tác với lớp điều khiển để quản lý luồng hoạt động ở lớp cơ sở hạ tầng.

3.2. Lớp điều khiển (Control Layer)

Đây là lớp chứa bộ điều khiển và là trung tâm xử lý luồng hoạt động của các gói tin, giúp các thiết bị mạng có thể chuyển tiếp các gói tin. Là nơi để kiểm tra vận hành và cấu hình mạng.

3.3. Lớp cơ sở hạ tầng (Infrastructure Layer)

Đây là lớp chứa các thiết bị mạng, đóng vai trò chuyển tiếp các gói tin theo chỉ thị từ lớp điều khiển.

3.4. API Bắc (Northbound API)

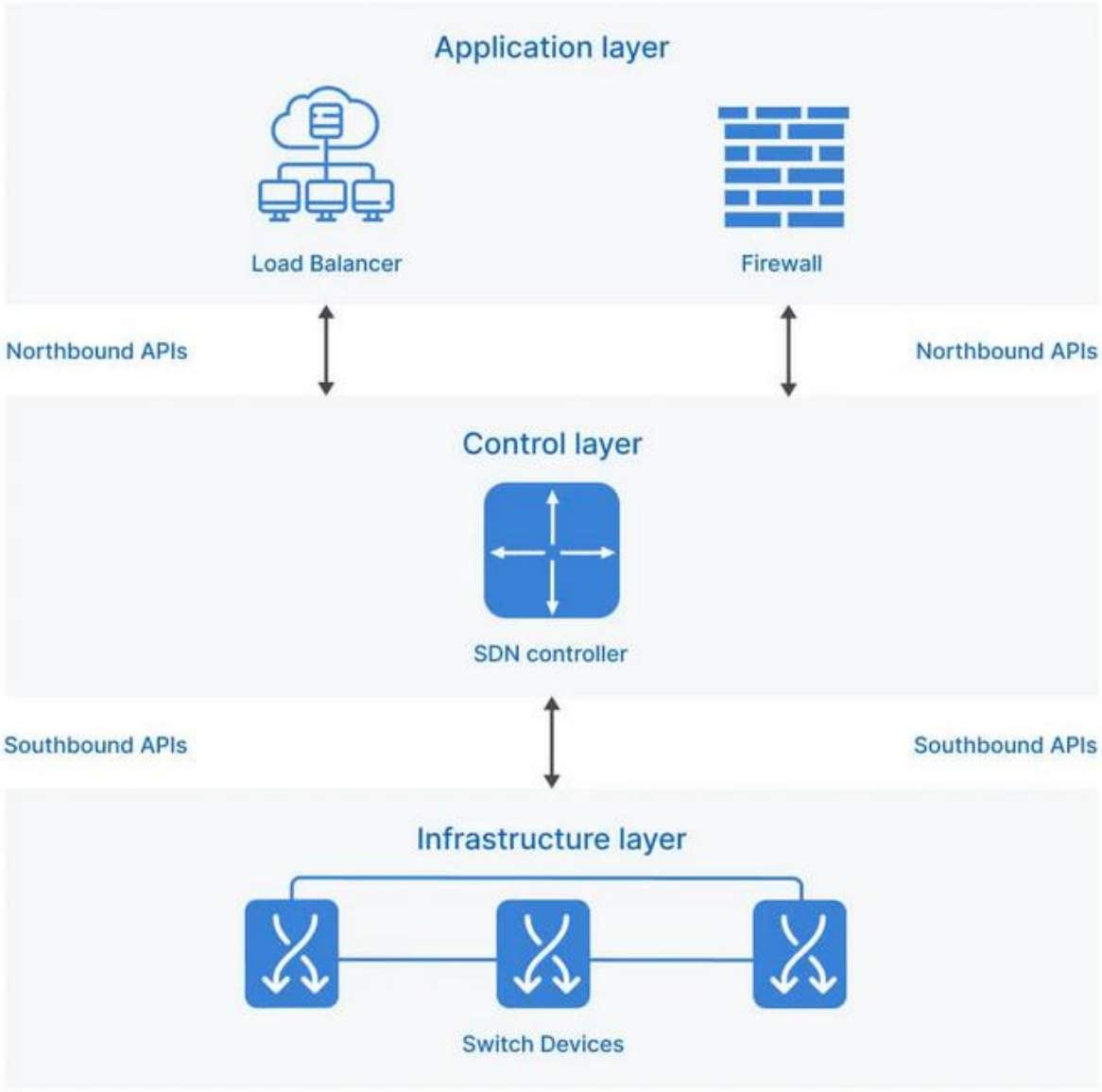
API Bắc là cách để lớp ứng dụng và lớp điều khiển có thể giao tiếp với nhau. Có thể sử dụng API Bắc để yêu cầu trạng thái mạng từ lớp điều khiển, hay là để kiểm soát việc triển khai các dịch vụ vào hệ thống mạng.

Ví dụ, đối với bộ điều khiển ONOS, ta có thể kích hoạt các ứng dụng có sẵn để hỗ trợ chuyển tiếp và định tuyến các gói tin, hay là sử dụng REST API để cài đặt quy tắc (flow rule) thủ công vào bộ điều khiển ONOS, từ đó bộ điều khiển sẽ cài đặt vào hệ thống mạng, nhằm chuyển tiếp, định tuyến, hay là cho phép và chặn lưu thông nhất định.

3.5. API Nam (Southbound API)

API Nam là cách để lớp điều khiển và lớp cơ sở hạ tầng có thể giao tiếp với nhau, cụ thể hơn là giúp lớp điều khiển có thể tương tác và điều khiển các thiết bị mạng.

OpenFlow là một API Nam phổ biến. Lớp điều khiển có thể sử dụng giao thức này để cài đặt flow rule đã được yêu cầu với API Bắc là REST API vào các thiết bị mạng được chỉ định.



Hình 3: Kiến trúc mạng SDN

4. OpenFlow: Switch, Flow table

OpenFlow là giao thức để lớp điều khiển và lớp cơ sở hạ tầng giao tiếp với nhau. OpenFlow là một ví dụ phổ biến nhất về API Nam, là một giao thức cho phép truy cập trực tiếp vào mức dữ liệu của các thiết bị mạng, giúp tầng điều khiển có thể điều khiển và thay đổi dữ liệu trong mức dữ liệu.

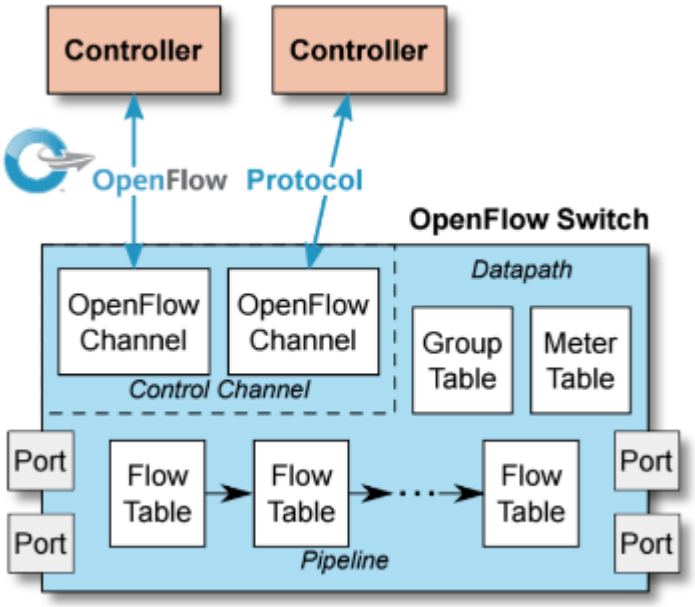
Chi tiết cụ thể hơn về cách vận hành của OpenFlow sẽ được thể hiện qua OpenFlow switch và OpenFlow flow table.

4.1. OpenFlow switch

Trong một OpenFlow switch, sẽ có các kênh để bộ điều khiển có thể sử dụng để giao tiếp và điều khiển. Cụ thể, bộ điều khiển có thể sử dụng OpenFlow để thêm, sửa

hay xóa các flow entry bên trong một flow table xác định. Điều này có thể được thực hiện chủ động, hay là phản hồi khi có một gói tin tới bộ điều khiển.

Trong một OpenFlow switch có thể có nhiều flow table, và các flow table khi được nối với nhau theo thứ tự sẽ được gọi là một pipeline. Điều này cũng chỉ ra rằng, các flow table có thể được sử dụng cho các mục đích khác nhau, cụ thể ở đây, các flow table trước có thể được sử dụng để kiểm tra đầu vào của gói tin, trong khi các flow table sau có thể được sử dụng để xử lý thêm gói tin, và quyết định xem gói tin nên được chuyển tiếp hay là nên bị bỏ đi.



Hình 4: Thành phần chính của OpenFlow switch

Trong các flow table sẽ có chứa các flow entry, mỗi flow entry sẽ làm nhiệm vụ kiểm tra hành động mà một OpenFlow switch sẽ thực hiện khi có một gói tin khớp với điều kiện đầu vào của flow entry đó. Chi tiết về flow entry sẽ được đi vào chi tiết hơn ở dưới.

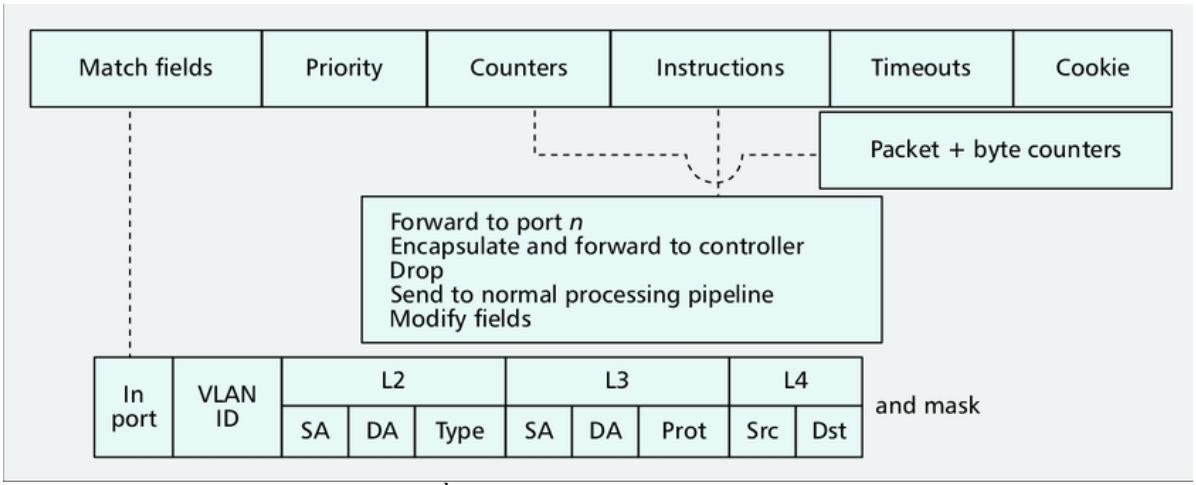
4.2. OpenFlow flow table

Một flow table có thể chứa một hoặc nhiều flow entry. Mỗi flow entry sẽ có các trường cụ thể và chức năng như sau:

- Match field: Trường này quyết định đối tượng cần xử lý là gì. Có thể xác định bằng cổng mà gói tin này tới, địa chỉ MAC nguồn, đích, là địa chỉ IP nguồn, đích hay là VLAN mà gói tin này thuộc về.
- Priority: Trong trường hợp gói tin đi tới có nhiều flow entry thỏa mãn, flow entry có giá trị priority cao nhất sẽ được ưu tiên áp dụng cho gói tin này.
- Counter: Chủ yếu được cập nhật giá trị khi có gói tin sử dụng flow entry này.
- Instruction: Trường này quyết định hành động cần thực hiện cho gói tin sử dụng flow rule đó là gì. Một số hành động bao gồm:

- Chuyển tiếp gói tin qua một cổng của OpenFlow switch.
- Chuyển tiếp gói tin lên bộ điều khiển để xử lý tiếp.
- Bỏ gói tin.
- Đưa gói tin sang flow table khác để xử lý thêm.
- Sửa các trường bên trong gói tin. Ví dụ, khi định tuyến gói tin khác VLAN với nhau, cần thiết phải đổi các trường như địa chỉ MAC nguồn và đích, cũng như là giá trị VLAN.

- Timeout: Thời gian còn lại trước khi flow entry này hết hạn và bị bỏ đi. Có thể tùy chỉnh để flow entry này tồn tại vĩnh viễn.
- Cookie: Trường này có thể được sử dụng bởi bộ điều khiển để lọc các flow entry.



Hình 5: Thành phần chính của flow entry trong một flow table

Cách sử dụng OpenFlow cụ thể như thế nào sẽ được làm rõ thông qua các kịch bản triển khai và cấu hình mạng SDN dưới đây.

Chương III. HIỆN THỰC HÓA ĐỀ TÀI

1. Mạng SDN sử dụng và kịch bản triển khai

Để có thể triển khai và cấu hình mạng SDN, nhóm sẽ sử dụng Mininet để tạo mạng SDN và cấu hình một vài thông số cơ bản, và việc cấu hình mạng SDN sẽ được thực hiện qua bộ điều khiển ONOS, tất cả thực hiện trong môi trường máy ảo Ubuntu.

Đề tài sẽ sử dụng hai mạng SDN để thực hiện các kịch bản.

1.1. Mạng SDN đầu

Mạng SDN đầu: Một switch trung tâm và một switch dự phòng để nối vào ba switch "lá", mỗi switch lá sẽ được nối vào hai máy khách. Tất cả các máy khách sẽ đều thuộc cùng một mạng 10.0.10.0/24, cụ thể:

- Switch trung tâm và switch dự phòng được ký hiệu lần lượt là s0a và s0b, cũng như có ID lần lượt là of:0000000000000001 và of:0000000000000002.
- Switch "lá" từ trái sang phải được ký hiệu lần lượt là sa1, sb1 và sc1, cũng như có ID lần lượt là of:0000000000000003, of:0000000000000004 và of:0000000000000005.
- Switch sa1 nối vào hai máy khách là ha1 và ha2 có IP lần lượt là 10.0.10.11 và 10.0.10.12.
- Switch sb1 nối vào hai máy khách là hb1 và hb2 có IP lần lượt là 10.0.10.21 và 10.0.10.22.
- Switch sc1 nối vào hai máy khách là hc1 và hc2 có IP lần lượt là 10.0.10.31 và 10.0.10.32.

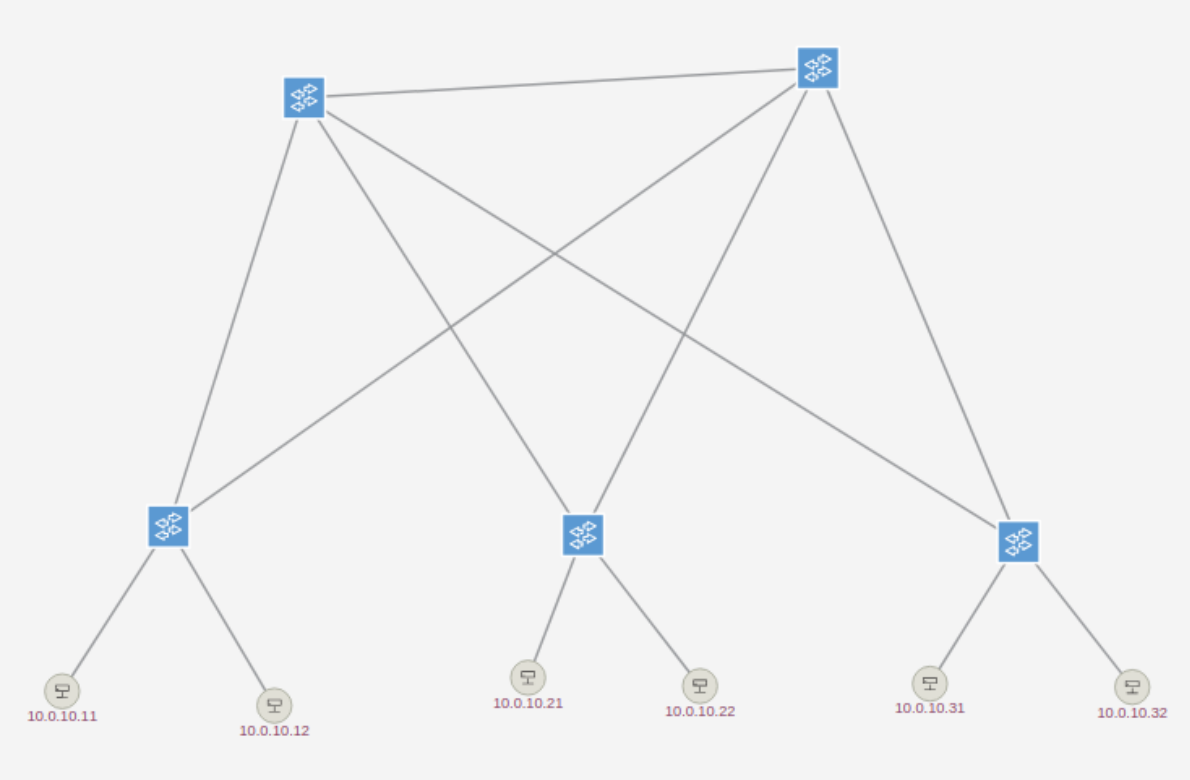
Đoạn mã được sử dụng để triển khai mạng SDN đầu cụ thể như sau:

```
#Importing necessary libraries
from mininet.net import Mininet # For mininet topo
from mininet.node import RemoteController # For connecting to ONOS
from mininet.topo import Topo
from mininet.link import TCLink #Traffic-Control Link, can be used for bandwidth
control and many more
from mininet.cli import CLI
from mininet.log import setLogLevel, info

# Our topology will look like this
"""
      s0a - s0b (s0b will be s0a's replacement, in case s0a is down)
    / | \
   /  |  \
  /   |   \
```

```
  sa1   sb1    sc1
 /  \  /  \   |  \
ha1 ha2 hb1 hb2 hc1 hc2
"""
# Our ONOS controller will connect to all of the switches
# Furthermore, we will define different VLAN for our hosts
# ha1, ha2 will be from 10.0.10.0/24
# hb1, hb2 will be from 10.0.20.0/24
# hc1, hc2 will be from 10.0.30.0/24
class RedundantVlanTopo(Topo):
    def __init__(self, **opts):
        super( RedundantVlanTopo, self ).__init__(**opts)

        # To ensure all five switches show up in ONOS GUI, assign a unique ID when
adding switches
        # Central switches (s0a, s0b)
        s0a = self.addSwitch('s0a', dpid='0000000000000001')
        s0b = self.addSwitch('s0b', dpid='0000000000000002')

        #Leaf switches (sa1, sb1, sc1)
        sa1 = self.addSwitch('sa1', dpid='0000000000000003')
        sb1 = self.addSwitch('sb1', dpid='0000000000000004')
        sc1 = self.addSwitch('sc1', dpid='0000000000000005')

        #Hosts (ha1, ha2, hb1, hb2, hc1, hc2)
        ha1 = self.addHost('ha1', ip = '10.0.10.11/24')
        ha2 = self.addHost('ha2', ip = '10.0.10.12/24')
        hb1 = self.addHost('hb1', ip = '10.0.10.21/24')
        hb2 = self.addHost('hb2', ip = '10.0.10.22/24')
        hc1 = self.addHost('hc1', ip = '10.0.10.31/24')
        hc2 = self.addHost('hc2', ip = '10.0.10.32/24')

        #Main links
        # Central switches links
        self.addLink(s0a, s0b)
        # Central switch to branch switches
        self.addLink(s0a, sa1)
        self.addLink(s0a, sb1)
        self.addLink(s0a, sc1)
        #sa1 to hosts
        self.addLink(sa1, ha1)
        self.addLink(sa1, ha2)
        #sb1 to hosts
        self.addLink(sb1, hb1)
        self.addLink(sb1, hb2)
        #sc1 to hosts
```

```
        self.addLink(sc1, hc1)
        self.addLink(sc1, hc2)
        #Backup links
        self.addLink(s0b, sa1)
        self.addLink(s0b, sb1)
        self.addLink(s0b, sc1)

# This function will be used to run our defined topo
def run():
    setLogLevel('info') # Mostly for debugging
    topo = RedundantVlanTopo()
    net = Mininet(topo=topo,controller=None,link=TCLink)
    #c0 = net.addController('c0', controller = RemoteController, ip = '172.17.0.5', port
= 6653)
    net.start() # Starts it
    info('*** Network is up\n')
    CLI(net) # Opens the terminal for demo
    net.stop() # Stops the topo, if terminal is closed

if __name__ == '__main__':
    run()
topos = { 'TopoWithRedundancy': ( lambda: RedundantVlanTopo() ) }
```

Bảng 1: Đoạn mã để triển mạng SDN đầu



Hình 6: Mạng SDN đầu tiên sau khi triển khai

Kịch bản sẽ được sử dụng cho mạng SDN đầu là:

- Chuyển tiếp giữa các gói tin trong mạng.
- Chặn một máy khách không được phép chủ động liên lạc tới máy khách khác.
- Chỉ cho phép một máy khách được liên lạc tới máy khách chỉ định.
- Kiểm tra việc chuyển tiếp gói tin bằng switch dự phòng.

1.2. Mạng SDN thứ hai

Mạng SDN thứ hai: Các switch "lá" bây giờ sẽ nối vào hai máy khách thuộc mạng khác nhau là 10.0.10.0/24 và 10.0.20.0/24 với VLAN lần lượt là 10 và 20. Mạng này sẽ không còn switch dự phòng để đơn giản hóa việc cấu hình, cụ thể:

- Switch trung tâm được ký hiệu là s0a, và có ID là of:0000000000000001.
- Switch "lá" từ trái sang phải được ký hiệu lần lượt là sa1, sb1 và sc1, cũng như có ID lần lượt là of:0000000000000003, of:0000000000000004 và of:0000000000000005.
- Switch sa1 nối vào hai máy khách là ha1 và ha2 có IP và MAC lần lượt là 10.0.10.11, 00:00:00:00:01:01 và 10.0.20.11, 00:00:00:00:02:01.
- Switch sb1 nối vào hai máy khách là hb1 và hb2 có IP và MAC lần lượt là 10.0.10.12, 00:00:00:00:01:02 và 10.0.20.12, 00:00:00:00:02:02.
- Switch sc1 nối vào hai máy khách là hc1 và hc2 có IP lần lượt là 10.0.10.13, 00:00:00:00:01:03 và 10.0.20.13, 00:00:00:00:02:03.

- Máy khách ha1, hb1, hc1 có cổng mặc định là 10.0.10.1, và máy khách ha2, hb2, hc2 có cổng mặc định là 10.0.20.1.

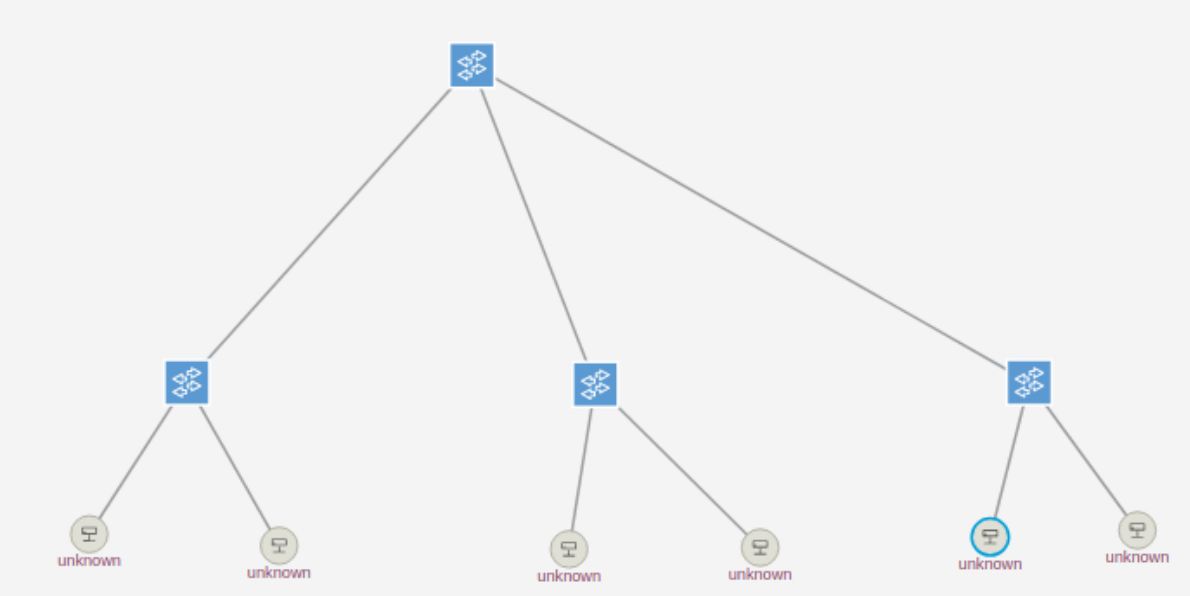
Đoạn mã được sử dụng để triển khai mạng SDN thứ hai cụ thể như sau:

```
#Importing necessary libraries
from mininet.net import Mininet # For mininet topo
from mininet.node import RemoteController # For connecting to ONOS
from mininet.topo import Topo
from mininet.link import TCLink #Traffic-Control Link, can be used for bandwidth
control and many more
from mininet.cli import CLI
from mininet.log import setLogLevel, info

# Our topology will look like this
"""
      s0a
    / |  \
   /  |   \
  /   |    \
 sa1  sb1   sc1
/ \  / \   | \
ha1 ha2 hb1 hb2 hc1 hc2
"""
# Our ONOS controller will connect to all of the switches
# Furthermore, we will define different VLAN for our hosts
# ha1, hb1, hc1 will be from 10.0.10.0/24
# ha2, hb2, hc2 will be from 10.0.20.0/24

ONOS_IP='172.17.0.5'
ONOS_OF_PORT=6653

class VlanRoutingTopo(Topo):
   def __init__(self, **opts):
      super( VlanRoutingTopo, self ).__init__(**opts)

      # To ensure all five switches show up in ONOS GUI, assign a unique ID when
adding switches
      # Central switches (s0a)
      s0a = self.addSwitch('s0a', dpid='0000000000000001')

      #Leaf switches (sa1, sb1, sc1)
      sa1 = self.addSwitch('sa1', dpid='0000000000000003')
      sb1 = self.addSwitch('sb1', dpid='0000000000000004')
      sc1 = self.addSwitch('sc1', dpid='0000000000000005')
```

```
        #Hosts (ha1, ha2, hb1, hb2, hc1, hc2)
        ha1 = self.addHost('ha1', ip = '10.0.10.11/24', mac='00:00:00:00:01:01',
defaultRoute='via 10.0.10.1')
        ha2 = self.addHost('ha2', ip = '10.0.20.11/24', mac='00:00:00:00:02:01',
defaultRoute='via 10.0.20.1')
        hb1 = self.addHost('hb1', ip = '10.0.10.12/24', mac='00:00:00:00:01:02',
defaultRoute='via 10.0.10.1')
        hb2 = self.addHost('hb2', ip = '10.0.20.12/24', mac='00:00:00:00:02:02',
defaultRoute='via 10.0.20.1')
        hc1 = self.addHost('hc1', ip = '10.0.10.13/24', mac='00:00:00:00:01:03',
defaultRoute='via 10.0.10.1')
        hc2 = self.addHost('hc2', ip = '10.0.20.13/24', mac='00:00:00:00:02:03',
defaultRoute='via 10.0.20.1')

        #Main links
        # Central switch to branch switches
        self.addLink(s0a, sa1, port1=2, port2=1)
        self.addLink(s0a, sb1, port1=3, port2=1)
        self.addLink(s0a, sc1, port1=4, port2=1)
        #sa1 to hosts
        self.addLink(sa1, ha1, port1=2, port2=1)
        self.addLink(sa1, ha2, port1=3, port2=1)
        #sb1 to hosts
        self.addLink(sb1, hb1, port1=2, port2=1)
        self.addLink(sb1, hb2, port1=3, port2=1)
        #sc1 to hosts
        self.addLink(sc1, hc1, port1=2, port2=1)
        self.addLink(sc1, hc2, port1=3, port2=1

# This function will be used to run our defined topo
def run():
    setLogLevel('info')
    topo = VlanRoutingTopo()
    onos_ctrl = RemoteController('c0', ip=ONOS_IP, port=ONOS_OF_PORT)
    net = Mininet(topo=topo, controller=onos_ctrl, link=TCLink, autoSetMacs=False)

    net.start()

    info('*** Setting up Static ARP for Gateways\n')
    ha1 = net.get('ha1')
    ha2 = net.get('ha2')
    hb1 = net.get('hb1')
    hb2 = net.get('hb2')
    hc1 = net.get('hc1')
    hc2 = net.get('hc2')
```

```
    # STATIC ARP SO THAT PACKETS CAN GO TO THE LEAF SWITCHES FIRST, USING A DUMMY MAC 00:00:00:00:00:99
    ha1.cmd('arp -s 10.0.10.1 00:00:00:00:00:99')
    ha2.cmd('arp -s 10.0.20.1 00:00:00:00:00:99')
    hb1.cmd('arp -s 10.0.10.1 00:00:00:00:00:99')
    hb2.cmd('arp -s 10.0.20.1 00:00:00:00:00:99')
    hc1.cmd('arp -s 10.0.10.1 00:00:00:00:00:99')
    hc2.cmd('arp -s 10.0.20.1 00:00:00:00:00:99')

    info('*** Network is up\n')
    CLI(net)
    net.stop()

if __name__ == '__main__':
    run()
topos = { 'VlanRouting': ( lambda: VlanRoutingTopo() ) }
```

Bảng 2: Đoạn mã để triển khai mạng SDN thứ hai



Hình 7: Mạng SDN thứ hai sau khi triển khai

Kịch bản sẽ được sử dụng cho mạng SDN thứ hai là định tuyến và chuyển tiếp các gói tin trong mạng.

2. Mạng SDN đầu

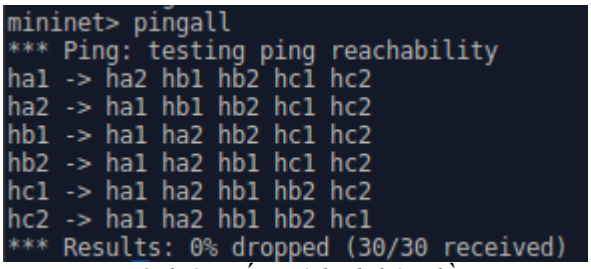
2.1. Chuyển tiếp giữa các gói tin trong mạng

Bộ điều khiển ONOS đã cung cấp sẵn ứng dụng để tiến hành điều khiển các thiết bị mạng chuyển tiếp gói tin đi khi có gói tin được chuyển tiếp lên bộ điều khiển. Để kích hoạt, vào giao diện dòng lệnh của ONOS và sử dụng câu lệnh sau:

```
onos> app activate org.onosproject.fwd
```

Bảng 3: Lệnh kích hoạt ứng dụng chuyển tiếp gói tin

Trên giao diện câu lệnh của Mininet, kiểm tra kết nối bằng lệnh pingall sẽ thấy tất cả các máy khách đều có thể liên lạc tới nhau.



Hình 8: Kết quả kịch bản đầu

2.2. Chặn một máy khách không được chủ động liên lạc tới máy khách khác

Kịch bản này sẽ sử dụng hai máy khách là ha1 và hb1, các gói tin từ máy khách ha1 chủ động gửi đi tới máy khách hb1 sẽ bị bỏ đi, trong khi máy khách hb1 vẫn có thể chủ động liên lạc tới máy khách ha1.

Để thực hiện kịch bản, tiến hành cài đặt một flow rule lên switch "lá" có thể thực hiện những công việc sau:

- Chỉ xử lý khi máy khách ha1 muốn liên lạc với máy khách hb1. Nói cách khác, trường match field cần phải có địa chỉ IP nguồn và đích lần lượt của máy khách ha1 và hb1.
- Trong bối cảnh kịch bản, tiến hành lọc thêm là chỉ xử lý khi máy khách ha1 sử dụng giao thức ICMP Request, tránh việc khi máy khách hb1 cố gắng liên lạc tới máy khách ha1 lại không nhận được phản hồi.
- Không xử lý gói tin khi xác nhận đúng đối tượng, lúc đó gói tin sẽ bị bỏ đi.

Flow rule được cài đặt sẽ như sau:

```
{
 "priority": 50000,
 "timeout": 0,
 "isPermanent": true,
 "deviceId": "of:0000000000000003",
 "treatment": {
  "instructions": []
 },
 "selector": {
  "criteria": [
    { "type": "ETH_TYPE", "ethType": "0x800" },
    { "type": "IPV4_SRC", "ip": "10.0.10.11/32" },
    { "type": "IPV4_DST", "ip": "10.0.10.21/32" },
    { "type": "IP_PROTO", "protocol": "1" },
    { "type": "ICMPV4_TYPE", "icmpType": "8" }
```

```
    ]
  }
}
```

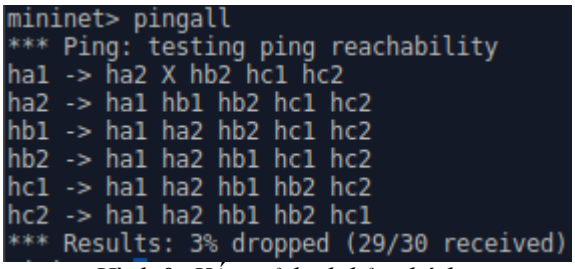
Bảng 4: Flow rule bỏ gói tin đến từ máy khách ha1 đến hb1

Sau khi đã xây dựng flow rule xong, tiến hành đăng tải vào switch "lá" nối tới máy khách ha1 thông qua curl như sau, với of:0000000000000003 là ID của switch cần được cài đặt flow rule, và drop-ha1-to-hb1.json là tên của tệp chứa flow rule đã xây dựng:

```
curl -u onos:rocks -X POST -H "Content-Type: application/json" -d @drop-ha1-to-hb1.json http://172.17.0.5:8181/onos/v1/flows/of:0000000000000003
```

Bảng 5: Lệnh để đăng tải flow rule đã xây dựng

Sau khi cài đặt thành công, kiểm tra kết quả bằng cách sử dụng lệnh pingall trên giao diện dòng lệnh của Mininet. Có thể thấy, liên lạc từ máy khách ha1 đến máy khách hb1 sẽ thất bại.



Hình 9: Kết quả kịch bản thứ hai

2.3. Chỉ cho phép một máy đích được liên lạc tới một máy đích chỉ định

Kịch bản này sẽ sử dụng hai máy khách chính là ha1 và hb2, các gói tin từ máy khách khác ha1 chủ động gửi tới máy khách hb2 sẽ bị bỏ đi, máy khách hb2 vẫn có thể chủ động liên lạc tới các máy khách khác.

Để thực hiện kịch bản, cần thiết phải cài đặt hai flow rule, lần lượt bỏ đi các gói tin khác chủ động gửi tới máy khách hb2, và chấp nhận gói tin đến từ máy khách ha1. Cách xây dựng flow rule bỏ gói tin và chấp nhận gói tin khá tương tự và đã được giải thích ở trên, nhưng cần lưu ý là flow rule chấp nhận gói tin cần phải có giá trị priority cao hơn flow rule bỏ gói tin đi để tránh xung đột hay là gói tin bị xử lý sai, và hành động cho gói tin từ máy khách ha1 là sẽ được chuyển tiếp ra cổng của switch "lá" được kết nối tới máy khách hb2, ở đây sẽ là cổng 3.

Hai flow rule bỏ và cho phép gói tin cụ thể như sau:

```
{
 "priority": 50000,
 "timeout": 0,
 "isPermanent": true,
 "deviceId": "of:0000000000000004",
```

```
  "treatment": {
    "instructions": []
  },
  "selector": {
    "criteria": [
      { "type": "ETH_TYPE", "ethType": "0x800" },
      { "type": "IPV4_DST", "ip": "10.0.10.22/32" },
      { "type": "IP_PROTO", "protocol": "1" },
      { "type": "ICMPV4_TYPE", "icmpType": "8" }
    ]
  }
}
```

Bảng 6: Flow rule bỏ gói tin được gửi chủ động tới máy khách hb2

```
{
  "priority": 51000,
  "timeout": 0,
  "isPermanent": true,
  "deviceId": "of:0000000000000004",
  "treatment": {
    "instructions": [{ "type": "OUTPUT", "port": "3" }]
  },
  "selector": {
    "criteria": [
      { "type": "ETH_TYPE", "ethType": "0x800" },
      { "type": "IPV4_SRC", "ip": "10.0.10.11/32" },
      { "type": "IPV4_DST", "ip": "10.0.10.22/32" },
      { "type": "IP_PROTO", "protocol": "1" },
      { "type": "ICMPV4_TYPE", "icmpType": "8" }
    ]
  }
}
```

Bảng 7: Flow rule chấp nhận gói tin do máy khách ha1 gửi

Tiến hành đăng tải hai flow rule đã xây dựng vào switch “lá” nối tới máy khách hb2, và kiểm tra bằng lệnh pingall trên Mininet. Có thể thấy, tất cả các máy khách ngoại trừ ha1 sẽ không thể chủ động gửi gói tin tới máy khách hb2 được.

```
mininet> pingall
*** Ping: testing ping reachability
ha1 -> ha2 hb1 hb2 hc1 hc2
ha2 -> ha1 hb1 X hc1 hc2
hb1 -> ha1 ha2 X hc1 hc2
hb2 -> ha1 ha2 hb1 hc1 hc2
hc1 -> ha1 ha2 hb1 X hc2
hc2 -> ha1 ha2 hb1 X hc1
*** Results: 13% dropped (26/30 received)
```

Hình 10: Kết quả kịch bản thứ ba

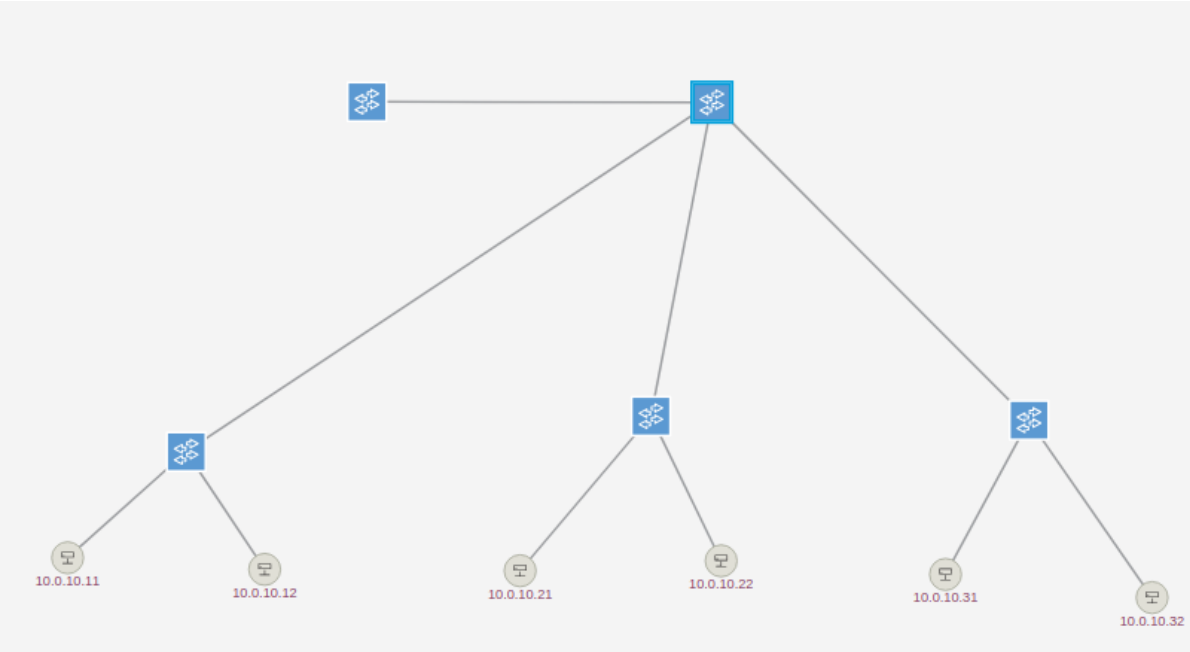
2.4. Kiểm tra việc chuyển tiếp giữa các gói tin bằng switch dự phòng

Để có thể kiểm tra việc chuyển tiếp bằng switch dự phòng, tiến hành ngắt tất cả kết nối giữa các switch "lá" tới switch trung tâm bằng các lệnh trên Mininet sau:

```
mininet> link sa1 s0a down
mininet> link sb1 s0a down
mininet> link sc1 s0a down
```

Bảng 8: Các lệnh ngắt kết nối tới switch chính

Sự thay đổi sẽ được phản ánh trên giao diện đồ họa của ONOS:



Hình 11: Mạng SDN thứ nhất sau khi ngắt kết nối tới switch chính

Kiểm tra kết nối bằng lệnh pingall trên Mininet cho thấy, ứng dụng chuyển tiếp gói tin của ONOS có thể linh hoạt tìm ra đường đi để chuyển tiếp các gói tin đến đích.

```
mininet> link sa1 s0a down
mininet> link sb1 s0a down
mininet> link sc1 s0a down
mininet> pingall
*** Ping: testing ping reachability
ha1 -> ha2 hb1 hb2 hc1 hc2
ha2 -> ha1 hb1 hb2 hc1 hc2
hb1 -> ha1 ha2 hb2 hc1 hc2
hb2 -> ha1 ha2 hb1 hc1 hc2
hc1 -> ha1 ha2 hb1 hb2 hc2
hc2 -> ha1 ha2 hb1 hb2 hc1
*** Results: 0% dropped (30/30 received)
```

Hình 12: Kết quả kịch bản thứ tư

3. Mạng SDN thứ hai

Với kịch bản thứ hai, sẽ chỉ có một kịch bản duy nhất là định tuyến và chuyển tiếp các gói tin từ các máy khách không cùng một mạng. Cụ thể, các switch "lá" sẽ nối tới hai máy khách lần lượt thuộc mạng 10.0.10.0/24 và 10.0.20.0/24.

Vì không thể dựa vào ứng dụng chuyển tiếp như những kịch bản trước, nên với kịch bản này, cần thiết phải cấu hình thủ công các flow rule để các gói tin có thể đến đích. Việc cài đặt có thể chia làm các công việc chính như sau:

- Cấu hình cổng mặc định cho các máy khách tại switch "lá". Cần thiết để định tuyến giữa các máy khách thuộc mạng khác nhau.
- Định nghĩa các hàm xử lý gói tin và flow rule đã xây dựng.
- Định tuyến giữa các máy khách khác mạng, nhưng liên kết trực tiếp cùng một switch "lá".
- Định tuyến và chuyển tiếp giữa các máy khách liên kết trực tiếp khác switch "lá".

Mặc dù bản chất vẫn là cấu hình flow rule thủ công, nhưng để không phải đăng tải từng flow rule, vốn rất mất thời gian, kịch bản sẽ định nghĩa một chương trình để đăng tải tất cả các flow rule cần thiết theo từng bài toán nhỏ trên, và chúng ta sẽ giải thích hướng tiếp cận và giải quyết của các bài toán đó.

Cần lưu ý là các phần được bôi đỏ ở các đoạn mã ở dưới là tham số được truyền vào cho hàm nếu như trong đoạn mã không có đề cập, cũng như là biến toàn cục được định nghĩa ở đầu chương trình, không phải là biến từ thư viện bên ngoài của Python.

3.1. Cấu hình cổng mặc định cho các máy khách tại switch "lá"

Để cho phép định tuyến giữa các máy khách, cần thiết phải cấu hình cổng mặc định, ở đây chính là switch "lá", để thực hiện tiếp các bước chuyển tiếp và định tuyến gói tin.

Để cấu hình cổng mặc định, có thể khai báo cổng mặc định của máy khách khi khởi tạo các máy khách như sau:

```
#Hosts (ha1, ha2, hb1, hb2, hc1, hc2)
ha1 = self.addHost('ha1', ip = '10.0.10.11/24', mac='00:00:00:00:01:01',
defaultRoute='via 10.0.10.1')
ha2 = self.addHost('ha2', ip = '10.0.20.11/24', mac='00:00:00:00:02:01',
defaultRoute='via 10.0.20.1')
hb1 = self.addHost('hb1', ip = '10.0.10.12/24', mac='00:00:00:00:01:02',
defaultRoute='via 10.0.10.1')
hb2 = self.addHost('hb2', ip = '10.0.20.12/24', mac='00:00:00:00:02:02',
defaultRoute='via 10.0.20.1')
hc1 = self.addHost('hc1', ip = '10.0.10.13/24', mac='00:00:00:00:01:03',
defaultRoute='via 10.0.10.1')
hc2 = self.addHost('hc2', ip = '10.0.20.13/24', mac='00:00:00:00:02:03',
defaultRoute='via 10.0.20.1')
```

Bảng 9: Cấu hình IP cổng mặc định các máy khách

Tuy nhiên, nếu chỉ vậy thì chưa đủ. Các máy khách cần phải biết địa chỉ MAC đích tương ứng với IP cổng mặc định là gì để có thể chuyển tiếp gói tin đi. Trong kịch bản này, có thể trực tiếp gán thủ công địa chỉ MAC tương ứng với IP cổng mặc định cho từng máy khách để tránh việc xử lý thông qua bộ điều khiển. Từ đó, các gói tin có thể được chuyển tới các switch "lá". Ở đây, có thể sử dụng địa chỉ MAC ảo bất kỳ cho các switch "lá".

```
ha1.cmd('arp -s 10.0.10.1 00:00:00:00:00:99')
ha2.cmd('arp -s 10.0.20.1 00:00:00:00:00:99')
hb1.cmd('arp -s 10.0.10.1 00:00:00:00:00:99')
hb2.cmd('arp -s 10.0.20.1 00:00:00:00:00:99')
hc1.cmd('arp -s 10.0.10.1 00:00:00:00:00:99')
hc2.cmd('arp -s 10.0.20.1 00:00:00:00:00:99')
```

Bảng 10: Cấu hình ARP thủ công cho IP cổng mặc định

3.2. Định nghĩa một số hàm xử lý gói tin và đăng tải flow rule đã xây dựng

Đầu tiên, cần định nghĩa flow rule để xử lý gói tin khi mới vào switch "lá". Nhiệm vụ của flow rule này chủ yếu là gán vào thẻ VLAN phù hợp, và chuyển tiếp cho flow table tiếp theo xử lý. Ở đây, quy ước là mạng 10.0.10.0/24 sẽ có VLAN 10, mạng 10.0.20.0/24 sẽ có VLAN 20.

```
def provision_ingress_rule(device_id, host_port, vlan_id):
    print(f"Configuring Ingress for Port {host_port} on VLAN {vlan_id}...")
    rule = {
        "priority": 40000, "isPermanent": True, "deviceId": device_id, "tableId": 0,
        "selector": {
            "criteria": [ {"type": "IN_PORT", "port": host_port} ]
        },
        "treatment": { "instructions": [
            {"type": "L2MODIFICATION", "subtype": "VLAN_PUSH"},
```

```
        {"type": "L2MODIFICATION", "subtype": "VLAN_ID", "vlanId":
vlan_id},
        {"type": "TABLE", "tableId": 1}
      ]}
  }
  send_flow(device_id, rule)
```

Bảng 11: Hàm xây dựng flow rule xử lý khi có gói tin vào switch

Ở cuối hàm này có gọi một hàm là send_flow(), hàm này sẽ đóng vai trò đăng tải flow rule đã xây dựng lên switch phù hợp. Hàm send_flow() cũng sẽ được sử dụng cho các hàm xây dựng flow rule khác.

```
def send_flow(device_id, flow_data):
  url = f'http://{ONOS_IP}:{ONOS_PORT}/onos/v1/flows/{device_id}'
  response = requests.post(url, auth=AUTH, data=json.dumps(flow_data),
headers={'Content-Type': 'application/json'})
  if response.status_code not in [200, 201]:
    print(f" [FAIL] {device_id} Error: {response.text}")
```

Bảng 12: Hàm đăng tải flow rule

3.3. Định tuyến giữa các máy khách khác mạng, cùng switch "lá"

Khi đích đến là một máy khách cùng switch "lá", hướng tiếp cận sẽ trở nên đơn giản hơn nhiều, khi mà ta chỉ cần bỏ hẳn thẻ VLAN và chuyển tiếp gói tin sang máy khách đó. Ngoài ra, để cho đúng quy tắc chuyển tiếp, có thể điều chỉnh địa chỉ MAC nguồn thành địa chỉ MAC cổng mặc định đã định nghĩa trước đó.

```
def provision_intra_switch_route(
  name,
  device_id,
  src_vlan,
  dst_ip,
  dst_mac,
  dst_vlan,
  out_port
):
  print(f"Configuring Local Route: {name}...")
  rule = {
    "priority": 40000, "isPermanent": True, "deviceId": device_id, "tableId": 1,
    "selector": {
      "criteria": [
        {"type": "IPV4_DST", "ip": dst_ip},
        {"type": "ETH_TYPE", "ethType": "0x0800"},
        {"type": "VLAN_VID", "vlanId": src_vlan}
      ]
    },
    "treatment": { "instructions": [
```

```
        {"type": "L2MODIFICATION", "subtype": "ETH_DST", "mac": dst_mac},
        {"type": "L2MODIFICATION", "subtype": "ETH_SRC", "mac":
ROUTER_MAC},
        {"type": "L2MODIFICATION", "subtype": "VLAN_ID", "vlanId":
dst_vlan},
        {"type": "L2MODIFICATION", "subtype": "VLAN_POP"},
        {"type": "OUTPUT", "port": out_port}
    ]}
  }
  send_flow(device_id, rule)
```

Bảng 13: Hàm định nghĩa flow rule chuyển tiếp gói tin cùng một switch

3.4. Định tuyến và chuyển tiếp giữa các máy khách khác switch

Về cơ bản, việc chuyển tiếp và định tuyến đều đi qua các bước giống nhau:

- Chuyển tiếp gói tin từ switch "lá" lên switch trung tâm.
- Switch trung tâm đọc thông tin gói tin và quyết định nên chuyển tiếp tới switch "lá" nào.
- Switch "lá" đích nhận gói tin, kiểm tra gói tin có đúng không, bỏ thẻ VLAN và chuyển tiếp về máy đích.

Sự khác nhau nằm ở cách mà switch "lá" chuyển tiếp gói tin lên switch trung tâm như thế nào. Đối với việc chuyển tiếp giữa các máy khách cùng một mạng, không cần thiết phải xử lý gì thêm và chuyển tiếp thẳng lên switch trung tâm. Tuy nhiên, đối với việc định tuyến, cần thiết phải đổi thẻ VLAN để phù hợp với máy khách đích, đổi địa chỉ MAC nguồn vì việc định tuyến phải đi qua cổng mặc định, ở đây cũng là switch "lá" nối tới máy khách nguồn luôn, và đổi địa chỉ MAC đích vì trước đó, địa chỉ MAC đích là cổng mặc định.

```
rule_src = {
    "priority": 40000, "isPermanent": True, "deviceId": src_leaf_id, "tableId": 1,
    "selector": {
      "criteria": [
        {"type": "ETH_DST", "mac": dst_mac},
        {"type": "VLAN_VID", "vlanId": vlan_id}
      ]
    },
    "treatment": { "instructions": [
      {"type": "OUTPUT", "port": src_uplink}
    ]}
  }
```

Bảng 14: Định nghĩa flow rule chuyển tiếp lên switch trung tâm với máy khách cùng mạng

```
rule_src = {
    "priority": 41000,
```

```
        "isPermanent": True, "deviceId": src_leaf_id, "tableId": 1,
        "selector": {
            "criteria": [
                {"type": "ETH_DST", "mac": ROUTER_MAC},
                {"type": "IPV4_DST", "ip": dst_ip},
                {"type": "ETH_TYPE", "ethType": "0x0800"},
                {"type": "VLAN_VID", "vlanId": src_vlan}
            ]
        },
        "treatment": { "instructions": [
            {"type": "L2MODIFICATION", "subtype": "ETH_SRC", "mac":
ROUTER_MAC},
            {"type": "L2MODIFICATION", "subtype": "ETH_DST", "mac": dst_mac},
            {"type": "L2MODIFICATION", "subtype": "VLAN_ID", "vlanId":
dst_vlan},
            {"type": "OUTPUT", "port": src_uplink}
        ]}
    }
```

Bảng 15: Định nghĩa flow rule chuyển tiếp gói tin lên switch trung tâm với máy khách khác mạng

Flow rule sử dụng để chuyển tiếp từ switch trung tâm đến switch "lá" đích, và từ switch "lá" đích tới máy khách đích cho việc chuyển tiếp và định tuyến đều giống nhau.

- Switch trung tâm sẽ lọc switch "lá" đúng và chuyển tiếp tới đó.

```
rule_spine = {
        "priority": 40000, "isPermanent": True, "deviceId": spine_id, "tableId": 0,
        "selector": {
            "criteria": [ {"type": "ETH_DST", "mac": dst_mac} ]
        },
        "treatment": { "instructions": [
            {"type": "OUTPUT", "port": spine_downlink}
        ]}
    }
```

Bảng 16: Định nghĩa flow rule chuyển tiếp tới switch "lá" đích đúng

- Switch "lá" đích sẽ bỏ thẻ VLAN đi và chuyển tiếp tới máy đích đúng.

```
rule_dst = {
        "priority": 40000, "isPermanent": True, "deviceId": dst_leaf_id, "tableId": 0,
        "selector": {
            "criteria": [
                {"type": "ETH_DST", "mac": dst_mac},
                {"type": "VLAN_VID", "vlanId": dst_vlan}
            ]
        },
        "treatment": { "instructions": [
```

```
            {"type": "L2MODIFICATION", "subtype": "VLAN_POP"},
            {"type": "OUTPUT", "port": dst_host_port}
        ]}
    }
```

Bảng 17: Định nghĩa flow rule chuyển tiếp tới máy đích đúng

3.5. Đăng tải flow rule và kết quả

Để tất cả các máy khách có thể liên lạc được với nhau, trong chương trình được soạn sẵn sẽ thực hiện đăng tải các flow rule theo từng cặp máy khách cụ thể (chiều xuôi và ngược) như sau:

- Định tuyến giữa các máy khách khác mạng, cùng switch "lá":
    - Máy khách ha1 và ha2.
    - Máy khách hb1 và hb2.
    - Máy khách hc1 và hc2.

```
provision_intra_switch_route(
        name="ha1 to ha2",
        device_id=DEV_SA1,
        src_vlan=10,
        dst_ip="10.0.20.11/32",
        dst_mac="00:00:00:00:02:01",
        dst_vlan=20,
        out_port=3
    )
```

Bảng 18: Ví dụ gọi hàm định tuyến trong cùng switch "lá"

- Chuyển tiếp giữa các máy khách cùng mạng, khác switch "lá":
    - Máy khách ha1 và hb1.
    - Máy khách ha1 và hc1.
    - Máy khách hb1 và hc1.
    - Máy khách ha2 và hb2.
    - Máy khách ha2 và hc2.
    - Máy khách hb2 và hc2.

```
provision_l2_remote_forwarding(
        name="L2: ha1->hb1",
        src_leaf_id=DEV_SA1,
        dst_leaf_id=DEV_SB1,
        spine_id=DEV_S0A,
        dst_mac=DEV_HB1,
        vlan_id=10,
        src_uplink=1,
        spine_downlink=3,
        dst_host_port=2
```

```
)
```

Bảng 19: Ví dụ gọi hàm chuyển tiếp khác switch "lá"

- Định tuyến giữa các máy khách khác mạng, khác switch "lá":
  - Máy khách ha1 và hb2.
  - Máy khách ha1 và hc2.
  - Máy khách ha2 và hb1.
  - Máy khách ha2 và hc1.
  - Máy khách hb1 và hc2.
  - Máy khách hb2 và hc1.

```
provision_l3_remote_routing(
    name="L3: ha1->hb2",
    src_leaf_id=DEV_SA1,
    dst_leaf_id=DEV_SB1,
    spine_id=DEV_S0A,
    src_vlan=10,
    dst_vlan=20,
    dst_ip="10.0.20.12/32",
    dst_mac=DEV_HB2,
    src_uplink=1,
    spine_downlink=3,
    dst_host_port=3
)
```

Bảng 20: Ví dụ gọi hàm định tuyến khác switch "lá"

Kết quả đăng tải flow rule và thực hiện lệnh pingall trên Mininet để kiểm tra cấu hình cho thấy đã thành công chuyển tiếp và định tuyến tất cả gói tin giữa các máy khách, cùng mạng hay khác mạng bên trong mạng SDN đã triển khai.

```
sdn@onos-tutorial:~/Desktop/Project/vlan-routing$ $PY configure-onos-router.py
Configuring ARP Punt on of:0000000000000003...
Configuring ARP Punt on of:0000000000000004...
Configuring ARP Punt on of:0000000000000005...
Configuring ARP Punt on of:0000000000000001...
Configuring Ingress for Port 2 on VLAN 10...
Configuring Ingress for Port 3 on VLAN 20...
Configuring Ingress for Port 2 on VLAN 10...
Configuring Ingress for Port 3 on VLAN 20...
Configuring Ingress for Port 2 on VLAN 10...
Configuring Ingress for Port 3 on VLAN 20...
Configuring Local Route: ha1 to ha2...
Configuring Local Route: ha2 to ha1...
Configuring Local Route: hb1 to hb2...
Configuring Local Route: hb2 to hb1...
Configuring Local Route: hc1 to hc2...
Configuring Local Route: hc2 to hc1...
--- Provisioning L2 Path (Bridging): L2: ha1->hb1 ---
--- Provisioning L2 Path (Bridging): L2: hb1->ha1 ---
--- Provisioning L2 Path (Bridging): L2: ha1->hc1 ---
--- Provisioning L2 Path (Bridging): L2: hc1->ha1 ---
--- Provisioning L2 Path (Bridging): L2: hb1->hc1 ---
--- Provisioning L2 Path (Bridging): L2: hc1->hb1 ---
--- Provisioning L2 Path (Bridging): L2: ha2->hb2 ---
--- Provisioning L2 Path (Bridging): L2: hb2->ha2 ---
--- Provisioning L2 Path (Bridging): L2: ha2->hc2 ---
--- Provisioning L2 Path (Bridging): L2: hc2->ha2 ---
--- Provisioning L2 Path (Bridging): L2: hb2->hc2 ---
--- Provisioning L2 Path (Bridging): L2: hc2->hb2 ---
--- Provisioning L3 Path (Routing): L3: ha1->hb2 ---
--- Provisioning L3 Path (Routing): L3: hb2->ha1 ---
--- Provisioning L3 Path (Routing): L3: ha1->hc2 ---
--- Provisioning L3 Path (Routing): L3: hc2->ha1 ---
--- Provisioning L3 Path (Routing): L3: ha2->hb1 ---
--- Provisioning L3 Path (Routing): L3: hb1->ha2 ---
--- Provisioning L3 Path (Routing): L3: ha2->hc1 ---
--- Provisioning L3 Path (Routing): L3: hc1->ha2 ---
--- Provisioning L3 Path (Routing): L3: hb1->hc2 ---
--- Provisioning L3 Path (Routing): L3: hc2->hb1 ---
--- Provisioning L3 Path (Routing): L3: hb2->hc1 ---
--- Provisioning L3 Path (Routing): L3: hc2->hb2 ---
```

Hình 13: Kết quả đăng tải flow rule

```
mininet> pingall
*** Ping: testing ping reachability
ha1 -> ha2 hb1 hb2 hc1 hc2
ha2 -> ha1 hb1 hb2 hc1 hc2
hb1 -> ha1 ha2 hb2 hc1 hc2
hb2 -> ha1 ha2 hb1 hc1 hc2
hc1 -> ha1 ha2 hb1 hb2 hc2
hc2 -> ha1 ha2 hb1 hb2 hc1
*** Results: 0% dropped (30/30 received)
```

Hình 14: Kết quả kịch bản thứ năm

Chương IV. KẾT LUẬN

Trong quá trình thực hiện đề tài, nhóm chúng em đã thực hiện được những yêu cầu cơ bản ở phần lý thuyết và thực hành, cũng như thực hiện tìm hiểu thêm một số nội dung bổ sung, cụ thể là giới thiệu và sơ lược về sự khác nhau giữa mạng truyền thống và mạng SDN, và thực hiện thêm kịch bản trong mạng SDN được triển khai, là định tuyến gói tin.

Tuy nhiên, trong quá trình thực hiện kịch bản bị thiếu sót tài liệu hỗ trợ, đồng thời kiến thức về mạng máy tính còn hoàn chỉnh, dẫn đến quá trình tìm hiểu và thực hiện các kịch bản mất thời gian và gặp nhiều khó khăn. Ngoài ra, tuy việc sử dụng máy ảo Ubuntu được cung cấp sẵn trong hướng dẫn của bộ điều khiển ONOS giúp tiết kiệm thời gian cài đặt công cụ ban đầu, việc máy ảo Ubuntu này đã rất cũ dẫn đến sự không tương thích đối với một số tài liệu tham khảo mới hơn của bộ điều khiển ONOS, đặc biệt gây khó khăn trong quá trình thực hiện kịch bản định tuyến gói tin.

Dẫu vậy, trong quá trình thực hiện đề tài, nhóm chúng em đã học hỏi được rất nhiều kiến thức cả trong mạng truyền thống lẫn mạng SDN, và nhóm chúng em mong có thể tiếp tục phát triển đề tài, đặc biệt chính là việc tự phát triển một ứng dụng trên bộ điều khiển ONOS để có thể thực hiện mở rộng quy mô mạng SDN hơn nữa.

NGUỒN THAM KHẢO

1. SDN là gì?
2. Mininet
3. ONOS basic tutorial
4. OpenFlow Theory: Switch, Flow Tables and Protocol
5. Giáo trình môn học Nhập môn mạng máy tính – Chương 4