BÀI THÍ NGHIỆM

2

TẠO CÂY BẮC CẦU TỐI THIỂU

I. MỤC ĐÍCH THÍ NGHIỆM

Bài thí nghiệm này giúp sinh viên hiểu về cơ chế tạo Minimum Spanning Tree (MST), tìm hiểu POX Controller và cách tạo cây trong POX.

II. THẢO LUẬN

Các bộ điều khiển mạng dùng trong mạng OpenFlow gồm có: bộ điều khiển mạng mặc định, POX, POX, SNAC (giao diện Web để quản lý các OpenFlow switch), Beacon (Java). Tuy nhiên bộ điều khiển mạng chính và đáng chú ý nhất là bộ điều khiển mạng POX. POX là bộ điều khiển mở, phát triển bằng ngôn ngữ Python, cấu trúc đơn giản, gọn nhẹ, dễ dàng tạo và thêm các module mới.

Mục đích chính của POX gồm:

- Cung cấp một platform cho phép người lập trình, phát triển mạng triển khai các ý tưởng mới trong lĩnh vực mạng, sử dụng phần cứng thật. Các nhà phát triển có thể điều khiển tất cả các kết nối trong mạng gồm có: forwarding, routing… Ngoài ra POX còn điều khiển cả flow-table trong switch.
- Cung cấp phần mềm quản lý mạng hữu ích cho các tổng đài (operator), gồm có việc quản lý tập trung cho tất cả các switch trong mạng, điều khiển truy nhập của người dùng.

Phương thức hoạt động của POX:

- POX chạy riêng rẽ trên một máy và quản lý việc chuyển tiếp các bản tin giữa nhiều switch khác nhau. Trong quá trình mô phỏng, giả lập POX được chạy trên cùng một máy với đồ hình mạng được tạo ra bởi Mininet.
- POX cung cấp các giao diện lập trình giúp cho nhà phát triển sử dụng dễ dàng lấy được thông tin về sự kiện trong mạng, can thiệp vào lưu lượng, điều khiển các quyết định chuyển mạch của switch và tạo được lưu lượng.

- Khi có flow mới xuất hiện trọng mạng, các gói đầu tiên sẽ được gửi đến bộ điều khiển mạng POX, tại đây có thể thực hiện được: quyết định xem khi nào sẽ chuyển tiếp các gói đi trong mạng, định tuyến cho gói tin, thu thập các thông tin thống kê, chỉnh sửa được gói trong flow đó hoặc có thể xem thêm được về các gói khác trong cùng flow để thu thập được thêm nhiều thông tin.

POX đơn thuần chỉ là một platform, việc điều khiển mạng được thực hiện bởi các phần tử chức năng trong POX gọi là POX component, mỗi component thực thi một chức năng riêng biệt như định tuyến, chuyển mạch, xác thực... Có thể chạy một lúc nhiều POX component với các chức năng điều khiển khác nhau làm cho việc điều khiển và quản lý mạng trở nên hoàn hảo hơn. Các ứng dụng trong bộ điều khiển mạng POX có thể kết hợp với nhận biết các sự kiện trong mạng (network event), can thiệp vào lưu lượng trong mạng, điều khiển định tuyến của các switch và tạo ra lưu lượng.

Các thành phần của POX:

- Các thành phần Stock: POX đi kèm với một số thành phần stock. Một số thành phần cung cấp chức năng cốt lõi, một số cung cấp tính năng tiện lợi, và một số chỉ là ví dụ. Sau đây là một số thành phần stock của POX.

✓ *forwarding.l2_learning*: Thành phần này làm cho OpenFlow switch hoạt động như một thiết bị switch layer 2. Thành phần này học địa chỉ lớp 2 (địa chỉ MAC), các flow được cài đặt được match với các nhiều trường của header càng tốt.

✓ *forwarding.l3_learning*: Thành phần này biến các OpenFlow switch thành thiết bị không hẳn là một router nhưng cũng không phải là một switch layer 2, nó là một switch layer 3. Nó được dùng làm ví dụ khá tốt về việc sử dụng thư viện gói tin của POX để kiểm tra và xây dựng các bản tin ARP request và ARP reply.

✓ *openflow.spanning_tree*: Được sử dụng để khám phá các phần tử để xây dựng đồ hình mạng, xây dựng một spanning tree và sau đó disable các port không nằm trên spanning tree. Kết quả là một topo không bị loop. Chú ý là thành phần này không liên quan tới Spanning Tree Protocol.

✓ *openflow.discovery*: được sử dụng để khám phá topo mạng bằng việc gửi các bản tin LLDP tới các switch.

✓ *web.webcore*: Thành phần webcore khởi động một webserver trong tiến trình POX. Các thành phần khác có thể giao tiếp với nó để cung cấp nội dung tĩnh hoặc động của riêng chúng.

✓ *proto.dhcpd*: Nó đơn giản là một DHCP server. Mặc nó lấy địa chỉ 192.168.0.254 và cấp IP cho các DHCP client trong dải địa chỉ 192.168.0.1 đến 192.168.0.253 với DNS server và Default Gateway chính là DHCP server.

- Phát triển các thành phần tự tạo: Là các thành phần mà ta tự phát triển cho POX. Trong một số trường hợp, chúng ta có thể phải xây dựng một thành phần làm được những gì chúng ta muốn.

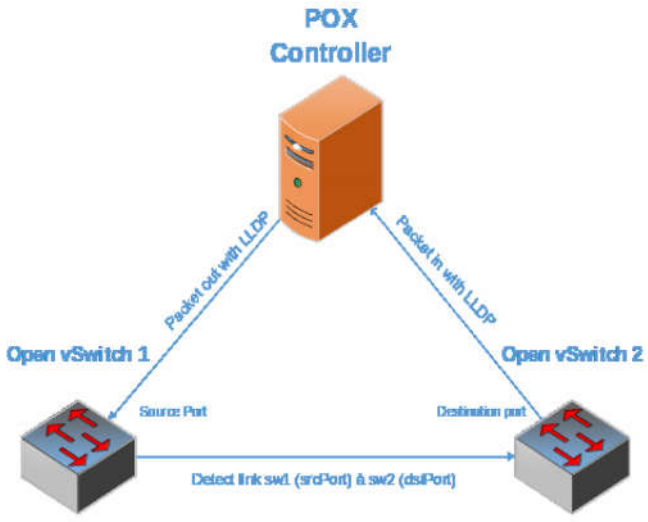
1. Module discovery

POX Controller sử dụng module discovery để phát hiện đồ hình mạng dựa trên cơ chế sau:

a. Các switch khi được bật lên sẽ liên tục sử dụng cơ chế bắt tay ba bước (TCP) để kết nối đến POX. Nếu POX bật, kết nối sẽ được khởi tạo, lúc này POX sẽ nhận biết được tất cả các con switch ở trong mạng cũng như trạng thái của từng switch. Mặc định Mininet sẽ cài đặt địa chỉ Controller mặc định cho switch là IP= localhost và port=6633. Chúng ta hoàn toàn có thể thay đổi bằng cách sử dụng lệnh sau.

```
sudo ovs-vsctl set-controller [controller-address]
```

b. Khi POX nhận được đầy đủ các switch, module discovery sẽ tạo các bản tin LLDP và đóng vào Packet_out gửi xuống các switch, các switch khi nhận được sẽ gửi LLDP ra các port, khi một switch nhận được một gói tin LLDP nó sẽ gửi lên hỏi POX, lúc này POX sẽ nhận được một link. Toàn bộ quá trình được mô tả trong hình dưới:



2. Module spanning_tree

Module spanning_tree giúp tạo cây cho các đồ hình mạng có loop. Cơ chế hoạt động của module này như sau:

a. Lưu tất cả các switch vào một danh sách theo thứ tự DPID

b. Lấy switch đầu tiên làm nút gốc, gọi switch này là **current switch**

c. Tất cả các link từ current switch sẽ được kiểm tra để xác định switch đích. Những link được thêm vào cây spanning_tree có switch đích chưa được xử lý trong lượt này và thỏa mãn

- Chỉ có duy nhất 1 link dẫn đến switch đích
- Có nhiều link dẫn đến switch đích: chọn ra link nhanh nhất, nếu có nhiều link nhanh nhất chọn link có chỉ số bé nhất.

d. Di chuyển tất cả switch đích vừa mới được thêm vào cây lên đầu của danh sách.

e. Làm lại bước c cho đến khi không còn switch nào trong danh sách

III. YÊU CẦU VỀ THIẾT BỊ

Để thực hiện bài thí nghiêm này cần một PC chạy Ubuntu 12.04, đã cài đặt sẵn Mininet và POX Controller.

IV. TRÌNH TỰ THÍ NGHIÊM

Trong bài thí nghiệm, sinh viên sẽ tiến hành tạo topology đơn giản bằng Mininet sau đó chạy POX Controller để điều khiển mạng, cuối cùng thêm các flow-entry vào switch để tạo một firewall đơn giản.

❑ **1.** Tạo một topology đơn giản bằng Mininet

Tạo file topo.py

- Tạo một file trong thư mục home với tên gọi là topo.py: Mở terminal và gõ các lệnh sau

```
cd
gedit topo.py
```

- Trên cửa sổ hiện ra tiến hành code tạo một topology mới như bên dưới.

```
from mininet.topo import Topo
“””
Tao mot mang ao co 4 switch duoc noi voi nhau thanh mot vong Ring
“””
class MyTopo(Topo):
```

```
"Topo gom 4 switch duoc noi voi nhau thanh mot vong Ring."
def __init__(self):
        # Khoi tao topo
        Topo.__init__( self )
        # Tao cac host ao va cac switch ao
        h1 = self.addHost('h1')
    h2 = self.addHost('h2')
    h3 = self.addHost('h3')
    h4 = self.addHost('h4')
    s1 = self.addSwitch('s1')
    s2 = self.addSwitch('s2')
    s3 = self.addSwitch('s3')
    s4 = self.addSwitch('s4')
        # Tao link ao giua cac switch-switch va host-switch
    self.addLink(h1, s1)
    self.addLink(h2, s2)
    self.addLink(h3, s3)
    self.addLink(h4, s4)
    self.addLink(s1, s2)
    self.addLink(s2, s3)
    self.addLink(s3, s4)
    self.addLink(s4, s1)
topos = { 'mytopo': ( lambda: MyTopo() ) }
```

- Một số phương thức được sử dụng

o Class Topo trong packet mininet.topo cung cấp các phương thức để tạo một topo bằng mininet.

o Phương thức addHost được sử dụng để tạo ra một host ảo trong mininet. Trong mininet các host ảo được giả lập từ chính PC mà ta đang chạy mininet.

o Phương thức addSwitch được dùng để tạo ra các switch ảo. Trong mininet các switch ảo là OpenVSwitch.

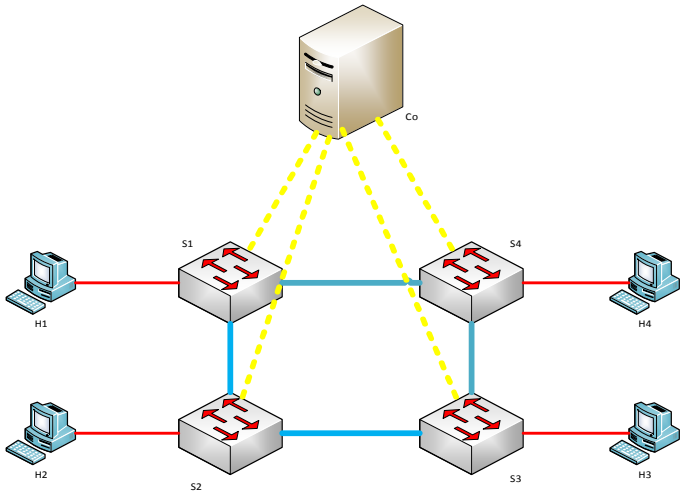
o Phương thức addLink được dùng để tạo link ảo giữa 2 switch hoặc giữa switch với host.

Chạy Mininet với topo tự tạo ở trên

```
sudo mn --custom topo.py --topo mytopo --controller remote
```

Kết quả thu được

```
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s1, s2) (s2, s3) (s3, s4) (s4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet>
```



Hình 1. Topology 4 switch - 4 host

Sau khi tạo topology thành công, sinh viên sẽ tiến hành chạy các module của POX ở một cửa sổ terminal khác.

```
cd pox/
./pox.py openflow.discovery forwarding.l2_learning
```

Lúc này, khi tiến hành pingall, các host không thể liên lạc với nhau. Nguyên nhân là do đồ hình mạng có loop, để các host có thể ping được ta cần chạy thêm module spanning_tree

```
cd pox/
./pox.py openflow.discovery forwarding.l2_learning openflow.spanning_tree
```

Sau khi chạy module spanning_tree, ta có thể ping thành công giữa các host

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

V. KẾT LUẬN

Qua bài thí nghiệm này, sinh viên đã làm quen với POX, một controller rất phổ biến trong SDN, đồng thời cũng hiểu rõ cơ chế tạo cây trong module Spanning_tree của POX.

VI. CÂU HỎI KIỂM TRA

1. Tại sao phải tạo cây MST?

……..
……..
……..
……..