MỞ RỘNG CÁC CHỨC NĂNG GIẢ LẬP HÌNH TRẠNG MẠNG TRONG MININET

Đào Như Ngọc[1], Phạm Quang Dũng[2*]

[1]Trường Khoa học và Kỹ thuật Máy tính, Đại học Chung - Ang, Hàn Quốc
[2] Khoa Công nghệ Thông tin, Học viện Nông nghiệp Việt Nam

Email[*]: pqdung.hau1@gmail.com

Ngày gửi bài: 22.07.2015 Ngày chấp nhận: 03.09.2015

TÓM TẮT

Công nghệ Mạng điều khiển bằng phần mềm (Software Defined Networking) hứa hẹn mở ra một tương lai tươi sáng mới cho mạng IP. Hiện nay, có rất nhiều nghiên cứu đang được tiến hành. Tuy nhiên, chỉ có một số ít framework hỗ trợ sự giả lập và thực hiện để đánh giá kết quả nghiên cứu. Trong số đó, Mininet là một trong những công cụ phổ biến nhất bởi tính mở, miễn phí và hỗ trợ đầy đủ giao thức Openflow phiên bản mới nhất. Mặc định, Mininet giúp tạo một mạng SDN điển hình chạy độc lập cùng với các máy trạm yếu chỉ có các chức năng cơ bản. Trong bài báo này, chúng tôi đóng góp những chức năng mở rộng của chức năng giả lập hình trạng mạng trong Mininet dựa trên Virtualbox. Những chức năng được mở rộng bao gồm: hỗ trợ kết nối Internet, các máy trạm cài hệ điều hành độc lập với Mininet, bộ điều khiển định tuyến chuẩn và log quá trình xử lý luồng dữ liệu tự động. Các chức năng bổ sung này sẽ mang đến sự đơn giản và thuận tiện cho các hoạt động nghiên cứu và đào tạo về công nghệ mạng SDN.

Từ khóa: Mạng điều khiển bằng phần mềm, Mininet, hình trạng mạng.

Function Expansion of Network Topology in Mininet

ABSTRACT

The Software Defined Networking technology promises a bright future to IP network. Many researches have been being conducted. However, until now there are only a few frameworks supporting emulation and implementation to verify the research. Mininet is one of the most popular tools because of the openness, cost effectiveness, and full Openflow support. By default, Mininet helps to create a standalone typical SDN network along with lightweight clients. In this paper, we explore the extended functions of network topology emulation in Mininet based on Virtualbox, such as Internet connection, independent OS clients, standard routing controller, and automatic flow logging. Therefore, it brings about an easier and convenient facility to research and training.

Keywords: Mininet, Openflow, SDN.

1. GIỚI THIỆU

Công nghệ thông tin và truyền thông đưa loài người đến cuộc sống tiện nghi hơn và ngược lại những yêu cầu của loài người thúc đẩy công nghệ phát triển nhanh chóng. Để thỏa mãn những yêu cầu cao về chất lượng dịch vụ, công nghệ mạng phải có khả năng hỗ trợ một cơ sở hạ tầng có tốc độ nhanh, an toàn, linh hoạt và kinh tế. Cho đến nay, Mạng điều khiển bằng phần mềm (SDN) cùng với giao thức Openflow được cho là ứng cử viên sáng giá đảm bảo được các yêu cầu này. Công nghệ SDN và giao thức Openflow đã ảnh hưởng tới tất cả khía cạnh của mạng IP: từ tầng truy cập đến tầng lõi, từ môi trường hộ gia đình đến doanh nghiệp, từ khía cạnh quản lý đến an toàn thông tin. Tuy nhiên, công nghệ SDN hiện vẫn chưa hoàn chỉnh, còn rất nhiều dự án đang tiếp tục được nghiên cứu. Vì vậy, các công cụ đánh giá có vai trò rất quan trọng đối với các nhà nghiên cứu để kiểm tra kết quả trước khi công bố công trình của mình.

Một trong những công cụ phổ biến nhất là Mininet. Đó là một chương trình giả lập có khả năng tạo mạng SDN nhanh chóng. Mininet hỗ trợ tất cả các tác vụ thực tế cần cho hoạt động nghiên cứu, phát triển và học tập. Dựa trên nhân Linux 2.2.26, nó sử dụng sự ảo hóa theo tiến trình để cung cấp các tiến trình độc lập cho các máy trạm, thiết bị chuyển mạch, bộ điều khiển và các liên kết ảo. Mã thật có thể được chạy mà không cần bất kỳ sự thay đổi nào. Mininet hỗ trợ cài đặt dễ dàng toàn bộ thử nghiệm mạng trong môi trường ảo hóa của các công cụ như VMWare hay Virtualbox cho hệ điều hành Mac/Windows/Linux. Tuy nhiên theo mặc định, bộ thư viện minh họa của Mininet chỉ có các tệp tính năng riêng rẽ. Vì vậy, rất khó cho người không chuyên hiểu được và tùy chỉnh theo yêu cầu. Mặt khác, các máy trạm yếu có thể không đủ đáp ứng trong các trường hợp đòi hỏi nhiều tính năng.

Ngoài ra, có hai công cụ đáng kể khác là EstiNet và ns - 3. Estinet là một phần mềm chuyên dụng giả lập và mô phỏng mạng SDN. Nó cho phép người sử dụng không chỉ tạo các hình trạng mạng bằng tính năng kéo thả dễ dàng mà còn đánh giá hiệu năng mạng một cách có hệ thống thông qua giao diện đồ họa. Tuy nhiên, EstiNet là một sản phẩm thương mại và hơn nữa là một ứng dụng không hoàn toàn cung cấp mã nguồn mở, nên khó khăn trong mở rộng các chức năng hoặc sửa mã nguồn. Ngược lại, ns - 3 là một công cụ đã có uy tín trong cộng đồng nghiên cứu một thời gian dài. Dựa trên kiến trúc mô đun hóa, ns - 3 cung cấp mô đun Openflow để mô phỏng mạng SDN bên cạnh các công nghệ mạng khác. Thực tế tại thời điểm hiện tại ns - 3 không được sử dụng rộng rãi vì nó vẫn chỉ dừng lại hỗ trợ giao thức Openflow phiên bản cũ 0.8.9 (phiên bản mới nhất là 1.3.4) và rất khó trong việc lập trình mạng.

Trong bài báo này, chúng tôi xây dựng một số chức năng mở rộng cho việc giả lập hình trạng mạng trong Mininet như: hỗ trợ kết nối Internet, các máy trạm cài hệ điều hành độc lập với Mininet, bộ điều khiển định tuyến chuẩn và lưu vết quá trình xử lý luồng dữ liệu tự động. Tất cả các tính năng được tích hợp chung trên một kiến trúc mạng mô phỏng thống nhất. Mô hình liên kết nối được dựa trên sự hỗ trợ của Virtualbox. Do vậy, kiến trúc này cải thiện ưu điểm và làm giảm những hạn chế của Mininet.

2. CÁC NGHIÊN CỨU LIÊN QUAN

Với nhiều ưu điểm lớn, Mininet là công cụ phổ biến nhất được sử dụng rộng rãi trong nghiên cứu và đào tạo về công nghệ mạng SDN. Để đáp ứng các mục đích khác nhau, Mininet được tùy chỉnh tạo ra nhiều phiên bản phù hợp. Wette et al. đã phát triển MaxiNet bằng cách mở rộng Mininet để có thể cài đặt Mininet chạy song song trên nhiều máy chủ vật lý MaxiNet có thể giả lập các mạng trung tâm dữ liệu rất lớn đòi hỏi băng thông cao và một lượng lớn các host trong kịch bản giả lập. MaxiNet thích hợp với các hoạt động nghiên cứu trong đánh giá hiệu năng mạng trung tâm dữ liệu.

Với mục đích khác, OpenNet được giới thiệu bởi Chan et al., là sự kết hợp giữa Mininet và ns - 3 để giả lập mạng cục bộ không dây điều khiển bằng phần mềm (SDWLAN). Hạn chế của Mininet trong việc hỗ trợ kết nối không dây được bù đắp khi kết hợp với tính năng mô phỏng của ns - 3. Kết quả cung cấp một framework tốt hơn để đánh giá nghiên cứu các giao thức và công nghệ SDWLAN.

Với cách tiếp cận khác, Kim et al. đã triển khai Openflow/SDN trên các bo mạch Raspberry Pi để cung cấp bộ công cụ thực hành có hiệu năng cao hơn với chi phí tiết kiệm. Raspberry Pi là một máy tính mini hỗ trợ một số cổng kết nối cơ bản trên một bo mạch nhỏ, sử dụng hệ điều hành Raspbian (dựa trên Debian core). Tuy nhiên, Raspberry Pi chỉ có tối đa 2 cổng kết nối Ethernet, nó không thể chạy như một bộ chuyển mạch nhiều cổng để thực hiện các chức năng của thiết bị chuyển mạch trên thực tế. Vì vậy, bộ công cụ này có hạn chế với một số trường hợp cụ thể. Hơn nữa, sự ổn định và dễ dàng cài đặt cũng cần được xem xét.

Trong nghiên cứu này, chúng tôi giới thiệu một kiến trúc hợp nhất dựa trên Mininet và

Virtualbox để hỗ trợ nhiều hơn các tính năng giả lập hình trạng mạng. Ưu điểm của nó là có thể dễ dàng cài đặt và thực hiện một hình trạng kết nối đầy đủ trong một máy tính cá nhân cho mục đích nghiên cứu và/hoặc học tập.

3. NHỮNG TÍNH NĂNG MỞ RỘNG

Kiến trúc hợp nhất gồm một máy ảo Mininet và một số máy ảo khách cài hệ điều hành độc lập kết nối với nhau trên nền của framework Virtualbox (Hình 1).

Trong máy ảo Mininet, chúng tôi tạo một hình trạng chuẩn gồm bộ điều khiển, bộ chuyển mạch và một số máy khách yếu (số lượng có thể tùy chỉnh theo yêu cầu, từ 1 đến hơn 4000 máy). Bộ điều khiển được triển khai là RemoteController để sẵn sàng lắng nghe bộ điều khiển POX đã được chúng tôi sửa đổi (sẽ được mô tả ở phần sau).

c0 = net.addController('c0', controller = RemoteController)

Máy ảo Mininet dành riêng một card mạng cho cấu hình Host - only Adapter với mục đích quản trị. Phần còn lại được tích hợp vào Open vSwitch như các cổng của nó. Dựa trên các kết nối này, bộ chuyển mạch có thể giao tiếp với các máy ảo khách cài hệ điều hành độc lập và có thể truy cập Internet.

_intf = Intf('eth1', node = s1)

Từ cửa sổ dòng lệnh Mininet CLI, hình trạng mạng đã sửa đổi có thể được thực hiện bởi lệnh:

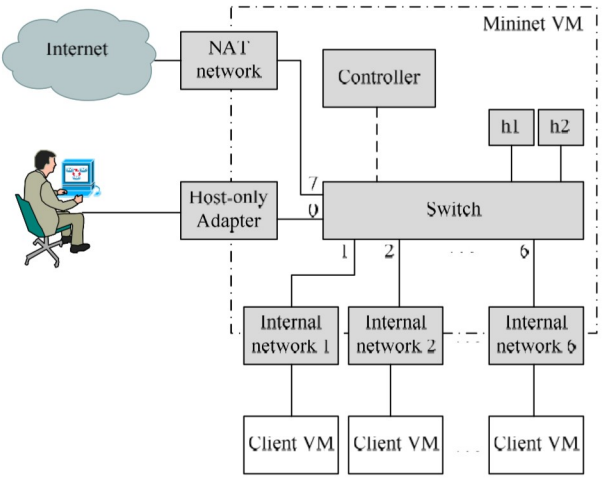
$sudo python modified_topology.py

A. Kết nối Internet

Kết nối truy cập Internet được hỗ trợ bởi cấu hình chức năng mạng NAT của Virtualbox. Adapter đã cấu hình tích hợp trực tiếp vào Open vSwitch, nó hoạt động như một gateway của mạng SDN để liên lạc với bên ngoài. Card mạng NAT có thể cung cấp một số ứng dụng hữu ích như các dịch vụ DHCP, NAT và giao thức IPv6. Để xác nhận các mạng NAT khả dụng, dùng lệnh:

$vboxmanage list natnetworks

Tất cả các máy khách nên thuộc cùng mạng con (subnet) của adapter, các địa chỉ IP được cấu hình tự động bởi dịch vụ DHCP trên máy chủ hoặc được cấu hình thủ công. Mặc định thì lớp địa chỉ IP được sử dụng là trong dải IP riêng.



Hình 1. Kiến trúc kết nối tổng thể dựa trên Virtualbox

B. Các máy khách cài hệ điều hành độc lập

Các máy ảo khách kết nối với Open vSwitch thông qua card mạng của chúng. Các card mạng tích hợp vào bộ chuyển mạch như các cổng. Mỗi máy ảo khách đòi hỏi cấu hình card mạng riêng tương ứng. Các lệnh cần thực hiện như sau:

$vboxmanage modifyvm Mininet – nic1 intnet

$vboxmanage modifyvm Mininet – intnet1 "intnet01"

$vboxmanage modifyvm Mininet – nicpromisc1 allow - all

Vì các máy khách không phụ thuộc vào sự cài đặt hình trạng SDN, chúng có thể được chạy bất kỳ loại hệ điều hành đầy đủ tính năng nào như Windows, Linux, hay thậm chí Mac hoặc Android. Tính năng mở rộng giúp triển khai dễ dàng một mô hình mạng SDN hoàn chỉnh gồm các máy trạm, máy chủ web, máy chủ email, máy chủ DHCP,...

C. Bộ điều khiển định tuyến chuẩn

Về cơ bản, các thiết bị lớp 3 định tuyến các gói tin dựa trên địa chỉ IP nguồn và đích. File cấu hình mặc định của bộ điều khiển *l3_learning* rất phức tạp đối với người mới tìm hiểu vì nó không chỉ xét các địa chỉ lớp 3 mà còn xét tất cả thông tin trong phần tiêu đề (header) của gói tin đến. Vì vậy, chúng tôi xây dựng lại một bộ điều khiển chuyển tiếp lớp 3, chạy như một bộ định tuyến chuẩn để tạo ra các luồng xử lý dữ liệu tường minh hơn với Open vSwitch (Hình 2).

Các gói tin ARP và echo được chuyển tiếp không cần chính sách nào. Các gói tin TCP/IP được theo dõi để tạo các luồng hợp lý trong các bảng định tuyến. Lệnh "$sudo ovs - ofctl dump - flows s1" hiển thị danh sách luồng của chuyển mạch s1 trên cửa sổ lệnh CLI.

Giả sử có *n* máy khách kết nối vào mạng. Ký hiệu số kết nối mà máy trạm i thiết lập tới máy đích *j* là k_{ij}, trong đó i = {1, 2,..., n}.

Số luồng định tuyến được tạo ra bởi bộ điều khiển *l3_learning* được xác định là:

$N_1 = \sum_{i=1}^{n} 2k_{ij}$ (1)

Vì bộ điều khiển định tuyến chuẩn không quan tâm đến thông tin lớp 4, số luồng định tuyến không phụ thuộc vào k_{ij}, do đó:

$N_2 = 2n(2)$

Vì vậy, số luồng định tuyến có thể loại bỏ là:

$N_1 - N_2 = \sum_{i=1}^{n} 2(k_{ij} - 1)$ luồng(3)

Hay

$\frac{N_2}{N_1} = \frac{n}{\sum_{i=1}^{n} k_{ij}}$phần trăm(4)

D. Lưu vết quá trình xử lý luồng dữ liệu tự động

Một trong những chức năng hạn chế quan trọng nhất của Mininet là đánh giá hiệu năng. Mininet chỉ cung cấp một số lệnh để kiểm tra trạng thái của bộ chuyển mạch và bộ điều khiển một cách thủ công. Nó có thể có ích cho học tập nhưng không đủ cho nghiên cứu vì các nhà nghiên cứu muốn nhận thông tin một cách có hệ thống và chính xác.

```
actions = []
actions.append(of.ofp_action_output(port = prt))
msg = of.ofp_flow_mod(command=of.OFPFC_ADD,
                      idle_timeout=FLOW_IDLE_TIMEOUT,
                      hard_timeout=FLOW_HARD_TIMEOUT,
                      buffer_id=event.ofp.buffer_id,
                      actions=actions,)
msg.match.dl_type = 0x800
msg.match.nw_src = srcaddr
msg.match.nw_dst = dstaddr
event.connection.send(msg)
```

Hình 2. Đoạn mã ví dụ của bộ điều khiển định tuyến chuẩn

Trong kiến trúc đề xuất, chúng tôi phát triển một chức năng lưu vết (logging) quá trình xử lý luồng dữ liệu có thể xuất trạng thái chính của các luồng định tuyến trong bộ chuyển mạch và các yêu cầu luồng đến bộ điều khiển một cách định kỳ. Các bản tin yêu cầu thống kê được gửi tới bộ chuyển mạch sau mỗi khoảng thời gian định trước.

```
body = of.ofp_aggregate_stats_request()
body = of.ofp_flow_stats_request()
```

Các hàm xử lý được lập trình để bắt các thông điệp báo cáo thống kê. Dữ liệu thu thập được xử lý để xuất thông tin hữu ích vào tệp tin ghi vết.

```
def _handle_AggregateFlowStatsReceived(self,event)
def _handle_FlowStatsReceived(self,event)
```

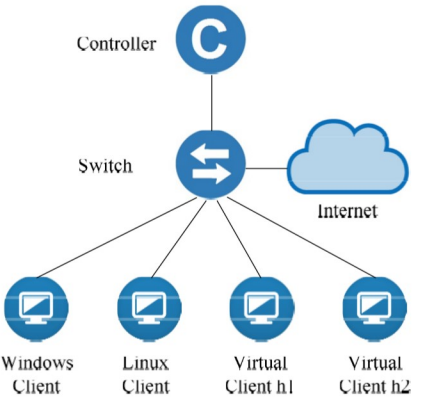
4. CÀI ĐẶT

Chúng tôi cài đặt kiến trúc đề xuất dựa trên Mininet 2.2.0 và framework Virtualbox 4.3.20. Mô hình gồm một máy ảo khách cài Windows XP SP2, một máy ảo khách Ubuntu Desktop 14.10 và hai host ảo h1 và h2. Dải IP là 10.0.10.1/24 với gateway mặc định 10.0.10.1 (Hình 3). Bộ điều khiển dùng chức năng định tuyến chuẩn trên nền POX. Nó cũng xuất thông tin luồng định tuyến vào tệp tin lưu vết một cách tự động. Các lệnh sau được thực hiện để khởi chạy:

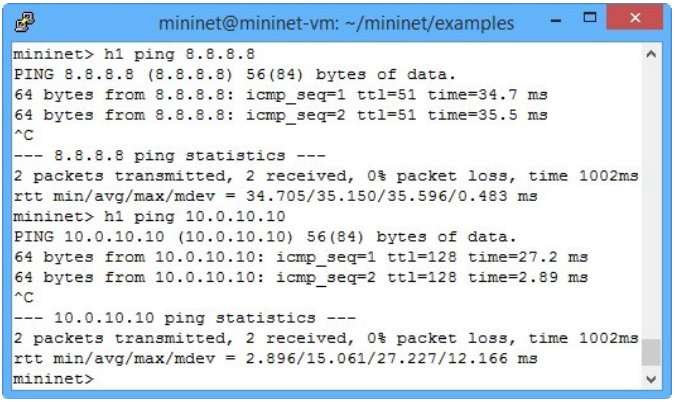
```
$sudo python modified_topology.py
$sudo python pox.py standard_routing
```

Đầu tiên, chúng tôi đánh giá các chức năng kết nối của máy khách yếu. Trong cửa sổ lệnh Mininet, chúng tôi Ping từ h1 đến 10.0.10.10 (địa chỉ IP của máy cài Windows) và 8.8.8.8 (địa chỉ IP của máy chủ DNS Google). Hình 4a mô tả sự thành công khi truy cập LAN và Internet từ h1. Vì độ trễ truyền tin, thời gian đáp ứng của hai gói tin đầu tiên từ 8.8.8.8 là bằng nhau. Tuy nhiên, chúng ta có thể thấy sự khác nhau rõ ràng khi so sánh hai gói tin hồi đáp đầu tiên từ địa chỉ nội bộ 10.0.10.10.

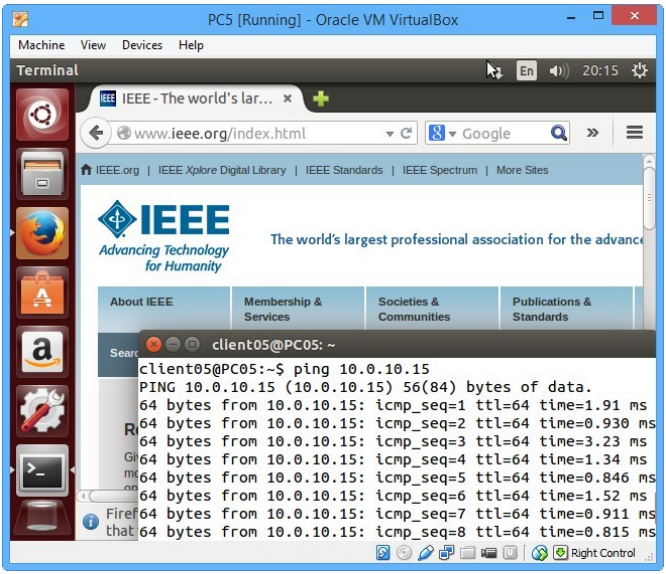
Tiếp theo, chúng tôi kiểm tra các chức năng kết nối của máy khách Ubuntu. Hình 4b cho thấy cửa sổ trình duyệt Firefox với trang chủ IEEE và kết quả Ping thành công đến 10.0.10.15 (địa chỉ IP của máy ảo h2). Không có hạn chế nào về khả năng liên kết nối. Máy khách không chỉ truy cập Internet tự do trên trình duyệt mà còn có thể thực hiện bất kỳ ứng dụng nào theo yêu cầu. Ngược lại, các máy khách yếu của Mininet chỉ cung cấp các chức năng hạn chế cơ bản thậm chí khi nó được khởi động trong chế độ đồ họa xterm GUI.



Hình 3. Mô hình cài đặt



(a). Truy cập LAN và Internet trong máy khách ảo h1



(b). Truy cập LAN và Internet trong máy khách Linux

Hình 4. Liên kết nối đầy đủ trong các máy khách

Kế tiếp, chúng tôi so sánh các luồng định tuyến của bộ điều khiển Mininet *l3_learning* với bộ điều khiển định tuyến chuẩn đề xuất của chúng tôi (Hình 5). Hai tác vụ được thực hiện gồm:

h2 ping h1: h2 gọi dịch vụ ICMP tới h1 bằng lệnh ping.

h2 wget - O - h1: h2 gọi dịch vụ TCP ở cổng 80 tới h1 bằng lệnh wget.

```
 cookie=0x0, duration=2.248s, table=0, n_packets=11, n_bytes=840,
idle_timeout=10, idle_age=2, priority=65535,tcp,in_port=2,vlan_tc
i=0x0000,dl_src=c2:8a:89:2d:22:57,dl_dst=ba:21:69:59:d9:95,nw_src=
10.0.0.2,nw_dst=10.0.0.1,nw_tos=0,tp_src=51254,tp_dst=80
actions=mod_dl_dst:ba:21:69:59:d9:95,output:1
 cookie=0x0, duration=2.246s, table=0, n_packets=11, n_bytes=2259,
idle_timeout=10, idle_age=2, priority=65535,tcp,in_port=1,vlan_t
ci=0x0000,dl_src=ba:21:69:59:d9:95,dl_dst=c2:8a:89:2d:22:57,nw_src
=10.0.0.1,nw_dst=10.0.0.2,nw_tos=0,tp_src=80,tp_dst=51254
actions=mod_dl_dst:c2:8a:89:2d:22:57,output:2
 cookie=0x0, duration=11.664s, table=0, n_packets=9, n_bytes=882,
idle_timeout=10, idle_age=3, priority=65535,icmp,in_port=2,vlan_tc
i=0x0000,dl_src=c2:8a:89:2d:22:57,dl_dst=ba:21:69:59:d9:95,nw_src=
10.0.0.2,nw_dst=10.0.0.1,nw_tos=0,icmp_type=8,icmp_code=0
actions=mod_dl_dst:ba:21:69:59:d9:95,output:1
 cookie=0x0, duration=11.663s, table=0, n_packets=9, n_bytes=882,
idle_timeout=10, idle_age=3, priority=65535,icmp,in_port=1,vlan_tc
i=0x0000,dl_src=ba:21:69:59:d9:95,dl_dst=c2:8a:89:2d:22:57,nw_src=
10.0.0.1,nw_dst=10.0.0.2,nw_tos=0,icmp_type=0,icmp_code=0
actions=mod_dl_dst:c2:8a:89:2d:22:57,output:2
```

(b). Trong bộ điều khiển *l3_learning* của Mininet

```
 cookie=0x0, duration=14.249s, table=0, n_packets=22, n_bytes=1918,
idle_timeout=10, hard_timeout=60, idle_age=7,
ip,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=14.238s, table=0, n_packets=16, n_bytes=2749,
idle_timeout=10, hard_timeout=60, idle_age=7,
ip,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:2
```

(b). Trong bộ điều khiển định tuyến chuẩn của kiến trúc chúng tôi đề xuất

Hình 5. Một ví dụ về các thông tin định tuyến trong các luồng

Trong hình 5a, màn hình hiển thị 4 luồng tương ứng với 2 dịch vụ mà h2 đã gọi. Điều đó có nghĩa là bộ điều khiển *l3_learning* không chỉ xét phần tiêu đề gói tin lớp 3 mà còn xét các thông tin lớp 4 như *tp_src*, *tp_dst*, *tcp protocol*. Ngược lại, bộ điều khiển định tuyến đề xuất chỉ tạo ra 2 luồng dựa trên phần tiêu đề gói tin lớp 3 (Hình 5b). Các trường thông tin chính được xem xét là *nw_src*, *nw_dst* và các *action*.

Cuối cùng, chúng tôi kiểm tra tính năng thống kê luồng định tuyến dựa trên kết quả phân tích dữ liệu được trích xuất trong tệp lưu vết. Trong kịch bản này, chúng tôi ấn định khoảng thời gian báo cáo là 60 giây. Cứ mỗi 60 giây, số gói tin yêu cầu (các yêu cầu luồng) đến bộ điều khiển và số luồng hiện tại trong bộ chuyển mạch được ghi lại (Hình 6). Ví dụ, trong 60 giây đầu tiên tổng số yêu cầu luồng và số luồng hiện tại tương ứng là 191 và 46. Mỗi luồng định tuyến được mô tả rõ ràng chi tiết: các địa chỉ nguồn và đích, trạng thái rỗi và hết hạn, tập các hành động. Dựa trên dữ liệu thu thập được, ta dễ dàng tạo được các đồ thị, biểu đồ trực quan để đánh giá hiệu năng.

```
---------------------------60.02800107--------------------------
The number of request packets from the switch to the controller: 191
The current number of flows: 46
Flow table in switch:
nw_src=222.255.27.22,nw_dst=10.0.10.25,idle_timeout=10,hard_timeout=60,
actions=[<pox.openflow.libopenflow_01.ofp_action_output object at 0xb583f76c>]
nw_src=123.30.51.197,nw_dst=10.0.10.25,idle_timeout=10,hard_timeout=60,
actions=[<pox.openflow.libopenflow_01.ofp_action_output object at 0xb583fccc>]
nw_src=10.0.10.25,nw_dst=74.125.235.164,idle_timeout=10,hard_timeout=60,
actions=[<pox.openflow.libopenflow_01.ofp_action_output object at 0xb583fb2c>]
nw_src=222.255.27.194,nw_dst=10.0.10.25,idle_timeout=10,hard_timeout=60,
actions=[<pox.openflow.libopenflow_01.ofp_action_output object at 0xb584908c>]
nw_src=222.255.27.185,nw_dst=10.0.10.25,idle_timeout=10,hard_timeout=60,
actions=[<pox.openflow.libopenflow_01.ofp_action_output object at 0xb584916c>]
nw_src=74.125.235.164,nw_dst=10.0.10.25,idle_timeout=10,hard_timeout=60,
actions=[<pox.openflow.libopenflow_01.ofp_action_output object at 0xb583f44c>]
```

Hình 6. Một phần của tệp lưu vết

Bảng 1. So sánh các tính năng

Tính năng	Kiến trúc đề xuất	Hình trạng Mininet mặc định
Truy cập Internet	Được tích hợp	Riêng biệt
Tương thích đầy đủ HĐH khách	Có	Không
Bộ điều khiển định tuyến chuẩn	Chuẩn	Phức tạp
Đánh giá hiệu năng	Theo tệp lưu vết	Không
Mô hình hợp nhất	Có	Riêng biệt

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong bài báo này, chúng tôi đã đề xuất một kiến trúc hợp nhất hỗ trợ các khả năng liên kết nối đầy đủ để giả lập mạng SDN. Phần mở rộng cũng bù đắp hạn chế của Mininet trong việc đánh giá hiệu năng. Nó có thể hữu ích cho các nhà nghiên cứu để xem xét ý tưởng SDN chính xác hơn và giúp những người mới tiếp cận hiểu công nghệ dễ dàng hơn mà không tốn kém chi phí. Kiến trúc này có thể được cài đặt dễ dàng, thậm chí trên một máy tính xách tay (Bảng 1).

Nhược điểm chính của kiến trúc đề xuất là không có khả năng giả lập mạng với quy mô lớn. Hơn nữa, bản thân Virtualbox cũng có một số hạn chế về hiệu năng. Vì vậy trong nghiên cứu tiếp theo, bên cạnh việc cải tiến các tính năng kết nối của giải pháp và hỗ trợ nhiều chức năng giả lập hơn, chúng tôi cần quan tâm đến khả năng giả lập mạng quy mô lớn. Chức năng lưu vết tự động cũng nên được nâng cấp để tạo ra đồ thị, biểu đồ trực quan tức thời.

TÀI LIỆU THAM KHẢO

Chan M. - C., C. Chen, J. - X. Huang, T. Kuo, L. - H. Yen, and C. - C. Tseng (2014). OpenNet: A Simulator for Software - Defined Wireless Local Area Network. Proc. of IEEE Wireless Communications and Networking Conference.

Clougherty M.M., C.A. White, H. Viswanathan, và C.L. Kahn (2014). The Role of SDN in IP Network Evolution. Telecommunications Science, Year 2014, Issue 5, Page 1 - 13.

Kim H., J. Kim, and Y. - B. Ko (2014). Developing a Cost - Effective OpenFlow Testbed for Small - Scale Software Defined Networking. Proc. of the 16th International Conference on Advanced Communication Technology.

Kreutz D., F.M.V. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky, and S. Uhlig (2015). Software - Defined Networking: A Comprehensive Survey. Proc. of the IEEE, 103(1): 14 - 76.

Lantz B., B. Heller, and N. McKeown (2010). A Network in a Laptop: Rapid Prototyping for Software - Defined Networks. Proc. of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks.

Mininet Team (2014). Mininet 2.2.0. http: //mininet.org/blog/2014/12/09/announcing - mininet - 2 - 2 - 0, truy cập ngày 01/06/2015.

ns - 3 project (2015). OpenFlow switch support. https: //www.nsnam.org/docs/models/html/openflow - switch.html, truy cập ngày 01/06/2015.

Open Networking Foundation (2015). Software - Defined Networking Definition. https: //www.opennetworking.org/sdn - resources/sdn - definition, truy cập ngày 01/06/2015.

Open Networking Foundation (2014). OpenFlow Switch Specification Version 1.3.4. https: //www.opennetworking.org/images/stories/downloads/sdn - resources/onf - specifications/openflow/openflow - switch - v1.3.4.pdf, truy cập ngày 01/06/2015.

Oracle (2015). Virtualbox documentation. https: //www.virtualbox.org/wiki/Documentation, truy cập ngày 01/06/2015.

Wette P., M. Draxler, A. Schwabe, F. Wallaschek, M.H. Zahraee, and H. Karl (2014). MaxiNet: Distributed Emulation of Software - Defined Networks. Proc. of 2014 IFIP Networking Conference, IEEE, DOI: 10.1109/IFIPNetworking.2014.6857078, p. 1 - 9.

Wang S. - Y., C. - L. Chou, and C. - M. Yang (2013). EstiNet OpenFlow Network Simulator and Emulator. Communications Magazine, IEEE, 51(9): 110 - 117.