**IV.Xây dựng một SDN Framework**

**Giới thiệu**

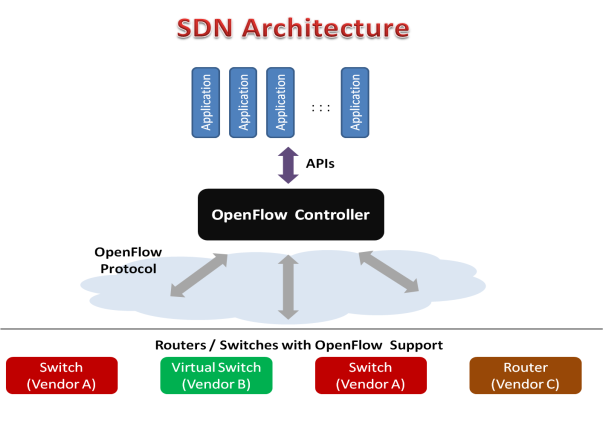
Quay trở lại lịch sử những năm 2007, 2008 khi mà các hãng Nicira/VMware, Nox... và các trường đại học liên quan đã có những bước đầu nghiên cứu về framework SDN. Tại thời điểm này, SDN đã là một công nghệ mạng tiên tiến và phát triển, tuy nhiên vẫn tồn tại một số bất cập về khả năng tương tác giữa các bộ điều khiển (controllers) : cách giao tiếp, tương tác giữa các controller, ngăn chặn ứng dụng từ các nhà tổ chức mạng, các nhà cung cấp các controller hoặc phần mềm thứ ba xây dựng để hoạt động liên tục trên các bộ điều khiển khác nhau. Điều này dẫn đến vấn đề tìm kiểm một giải pháp giải quyết bằng cách xây dựng một SDN Framework dựa trên nền tảng nguồn mở - đảm bảo không những chính xác về ngữ nghĩa mà còn phải chính xác về ngữ nghĩa. Theo đó xuất hiện thêm các thành phần phụ, chẳng hạn như các giao thức (protocols), các protocol tiện ích... Tất cả được chuẩn hóa tại một tiêu chuẩn chung IETF.

**Xây dựng mã trước, sau đó đặt câu hỏi...**

Để công bằng, các nhà cung cấp đã xây dựng về tiếp tục xây dựng mã nguồn và sau đó đặt câu hỏi về giới hạn tương tác. Các bộ điều khiển bắt đầu rời không kiểm tra tiêu chuẩn thực nghiệm môi trường và được xem xét trong triển khai thực tế. Thực tế thì các controller có khả năng thao tác chính xác các loại đối tượng từ nhiều nhà cung cấp (ví dụ: hai bộ điều khiển OpenFlow từ hai nhà cung cấp khác nhau) hoặc trên các miền quản trị (giả định cùng nhà cung cấp nhưng được xây dựng trên hai cơ sở dữ liệu khác khác nhau) là một vấn đề. Ngoài ra việc thiếu một giải pháp dựa trên tiêu chuẩn cho các controllers/agents protocols nghĩa là với sự lựa chọn từ bất kỳ bộ bộ điều khiển yêu cầu kiểm tra chất lượng tương thích với các thiết bị chuyển mạch (switch) triển khai và yêu cầu thử nghiệm tiếp theo nếu các thiết bị chuyển mạch được xem xét sau. Những vấn đề này đã trở lên rõ ràng khi các bộ điều khiển được cân nhắc tích hợp và các hệ thống hỗ trọ hoạt động được các nhà cung cấp sử dụng để cung cấp và kiểm soát mạng lưới của họ. Rào cản cuối cùng của ứng dựng là thiếu tính di động, một kết quả trực tiếp của việc không thành lập một chuẩn SDN northbound API. Đó là những ứng dựng dựa trên bộ điều khiển để tương tác với mạng hoặc mạng dịch vụ, sẽ cần xây dựng dựa trên sự lựa chọn của bộ điều khiển.

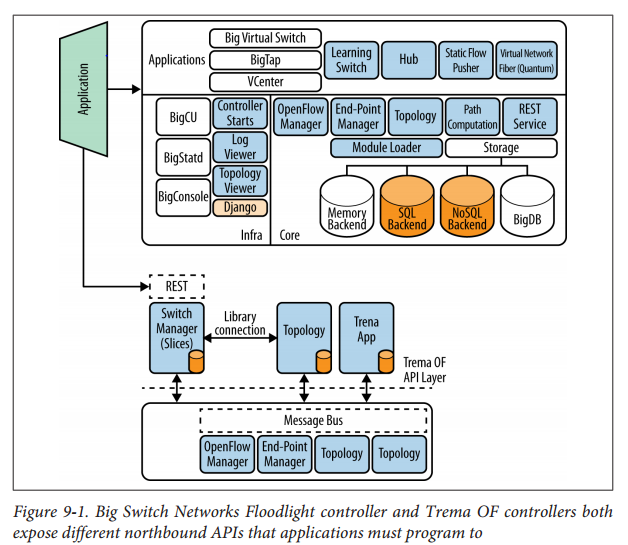
Để đưa ra một số ví dụ về lý do tại sao chuẩn framework cũng như cơ sở dữ liệu trên thực tế thực hiện framework đó là quan trọng, chúng ta hãy xem xét cách quản lý cổng trên thiết bị OpenFlow-enabled switches. Trong những tình huống này, tình thế khó xử có thể nảy sinh liên quan đến quyền sở hữu cổng, chẳng hạn như đặc quyền kiểm soát. Mặc dù điều này tồn tại ngay cả trong một giao thức độc canh southbound, khó khăn này còn trở lên tồi tệ hơn nữa trong trường hợp nhiều bộ điều khiển được ra lệnh.

Ví dụ, trong bộ điều khiển OpenFlow, nếu nhà điều hành chọn nhà cung cấp A (vendor A) bởi vì nó có một ứng dụng orchestration thích hợp để kiểm soát các cổng trong thiết bị chuyển mạch tại trung tâm dữ liệu và sua đó chọn nhà cung cấp B (vendor B) vì thiết bị đó đã yêu cầu một ứng dụng tap ảo, từ đó làm thế nào họ sẽ kiểm soát được bảng chuyển tiếp giữa hai bộ điều khiển khi cả hai đã cố gắng thực hiện chương trình trên cùng một switch? Tồi tệ hơn, vì cả hai nhà cung cấp bộ điều khiển SDN đã nằm trong một tiêu chuẩn hoặc thậm chí sắp xếp trước northbound API, ứng dụng orchestration từ vendor A có thể sẽ là không thể giao tiếp ít nhất là với bộ điều khiển của vendor B. Tương tự như vậy cho ứng dụng tap ảo. Về bản chất sẽ không có API tương tác để vượt trạng thái và phân quyền kiểm soát.



Các vấn đề khác của giao thức độc canh là một vài dịch vụ bị kìm hãm trên bộ điều khiển, đặc biệt trong khu vực topology. Điều này đúng hầu hết với các SDN controller/agent protocol bao gồm topology của lớp trừu trượng (abstract layer) của sự mở mang đơn nhất cung cấp ở lớp 2/MAC/FIB, lớp 3/RIB/LDB hoặc lớp 4+/service. Kết quả là không có cái nhìn toàn cảnh về toàn bộ mạng lưới. Vì vậy điều này rất quan trọng đối với việc ứng dụng mạng, như các trường hợp định tuyến dựa trên máy chủ PCE hoặc phân đoạn, những giải pháp đơn giản sẽ không hoạt động.

Cuối cùng là sự thiếu khả năng tương tác có nghĩa là khả năng hạn chế để bộ liên hợp SDN controller vào thứ gọi là môi trường vận hành mà không cần bổ sung "keo" ứng dụng để phân chia các cổng khác nhau. Ví dụ khả năng khâu một lớp phủ VxLAN được điều khiển bằng hệ thống VMware/Nicira SDN đến một thuê bao VPN VRF tại một cổng trung tâm dữ liệu sẽ yêu cầu một số automa - có thể thông qua một ứng dụng NETCONFbased hoặc cung cấp scripts. Hình 9.1 mô tả một scripts tương tự, theo đó một bộ điều khiển Floodlight và một bộ điều khiển Trema OF, cả 2 đều phơi bày khác biệt giữa các API mà cùng 1 ứng dụng tương tác. Không cung cấp một giao diện nhất quán mà các lập trình viên ứng dụng có thể mã hóa vào, cũng ko cho họ cung cấp cùng một mạng trửu tượng của các thiết bị chuyển mạch bên dưới hoặc các dịch vụ khác, chẳng hạn topology, mà họ trưng bày.



Mặc dù một số môi trường điều khiển hỗ trợ các ngôn ngữ phát triển hiện đại (ví dụ: Python) và các phương pháp mã hóa (Eclipse). Và cũng cung cấp các khung công tác độc quyền mạnh mẽ được hỗ trợ và khuyến khích phát triển ứng dụng. Các môi trường này bao gồm các tài liệu hay về API tới các chương trình như là debug/test tool. Thực tế, hầu hết sự hỗ trợ một vài tổ hợp các thuộc tính. Câu hỏi đặt ra là: làm thế nào các controller thích ứng mới hoặc các dịch vụ bổ sung, gia tăng cái mà mà các bộ điều khiển dịch vụ sẵn có cung cấp, hoặc khả năng lập trình chéo? Sau cùng, nếu thực sự SDN là tất cả ứng dụng thì trả lời câu hỏi này là rất quan trọng. Đi xa hơn 1 bước, một điều nên hỏi nữa là nếu tất cả các ứng dụng, không có một cuộc tranh luận giữa các controller các lựa chọn về sự cởi mở, khả năng tương tác và tính mô đun. Điều này sẽ không phải là không có tiền lệ trong ngành công nghiệp mạng hoặc thậm chí trong không gian giải pháp SDN (ví dụ OpenStack).

Cái nhìn này cho rằng SDN controller không phải là một giao thức đơn giao thức nhưng nên được mở, mô đun, tập trung vào các dịch vụ cung cấp cho các nhà phát triển ứng dụng từ đó dẫn đến định nghĩa của SDN controller như khuôn khổ phát triển ứng dụng mạng. Đó là lý do tại sao các nhà phát triển tin rằng sự kết hợp các yếu tố này sẽ thúc đẩy và kết quả là mục tiêu cuối cùng của việc phát triển nhanh chóng của các ứng dụng tương tác liên tục, nhanh, hiệu suất với mạng và thiết bị mà chúng cung cấp.

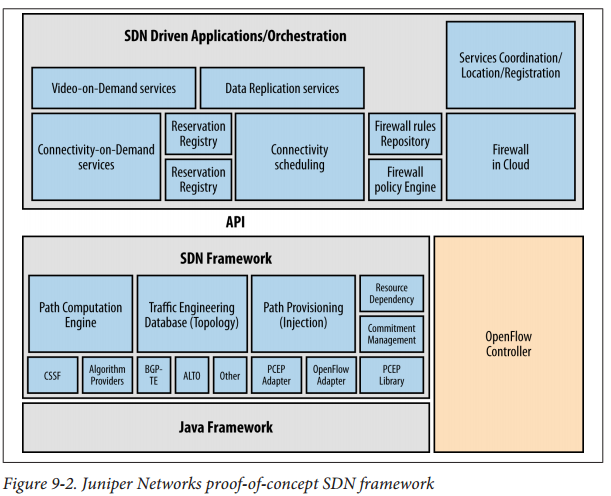
Bởi vì một vài dich vụ điều khiển được hình dung trong quá trình phát triển của khái niệm framework là đòi hỏi về hỗ trợ, tầm nhìn về controller cũng dịch chuyển từ đơn thiết bị hoặc thậm chí là một liên hợp các thiết bị, đến các hệ thống của thiết bị và máy ảo lưu trữ khác thành phần của các dịch vụ đó. Các thành phần này sau đó có thể được liên kết thông qua một thành phần của hệ thống. Từ đó hãy cùng nhau nghiên cứu ý nghĩa của một SDN framework và làm thế nào để phát triển, nhớ rằng khái niệm chính xác framework đã được phát triển cùng với các phần khác của công nghệ SDN được liên kết với nhau.

**THE JUNIPER SDN FRAMEWORK**

Vào đầu năm 2011, trước khi mua lại hệ thống Contrail trong năm 2012, Juniper Networks đã đưa ra ý tưởng rằng SDN nên có nhiều hơn một controller, chứ không phải một framework chỉ hỗ trợ các thành phần dịch vụ - được đặt với nhau trong một hệ thống từ một bộ điều khiển. Quan trọng hơn, mục tiêu của dự án là là minh họa thế nào về hệ thống khi được đặt với nhau đúng cách, cuối cùng có thể hỗ trợ một vòng lặp phản hồi chặt chẽ giữa ứng dụng mạng và mạng. Từ đó yêu cầu lập trình mạng.

Juniper Networks minh họa bằng một Java-bases framework với RESTful API – đáp ứng như một môi trường nguyên mẫu nhanh chóng để thêm vào sự phát triển mới và hữu ích của các ứng dụng. Một giao diện groovy scripting đơn giản phù hợp được cung cấp nhằm nâng cao API bằng cách kết hợp nó với khái niệm SDK. Khái niệm cơ sở là ứng dụng mạng cơ bản thành phần phục vụ của controller truyền thống (ví dụ: topology, tính toán đường dẫn, cung cấp đường dẫn) có thể có nhiều nguồn khác nhau. Các dịch vụ nguồn cung cấp các dịch vụ thông qua mô hình plugin-mở rộng thích ứng trong ứng dụng tương lai. Các dịch vụ ứng dụng tổng quát được tạo ra và sẽ cung cấp northbound APIs – trừu tượng khả năng của sounthbound plugin. Trong trường hợp này, các ứng dụng có thể đệ quy định nghĩa các dịch vụ mới mà các dịch vụ khác có thể dùng và tương tác.

Hệ thống điều khiển không có giao diện lưu trữ và phục vụ nhiều hơn một dịch vụ bus. Dịch vụ tính toán đường dẫn được cung cấp với thuật toán Junos CSPF (sau đó chuyển thành Java applet) đã được sử dụng trong các thiết bị Junos. Thuật toán này truy cập thông qua cơ sở hạ tầng plugin. Dịch vụ cung cấp đường dẫn đã được cung cấp thông qua nhiều giao thức southbound và truy cập thông qua các liên kết plugin (được thể hiện trong hình 9.2). Chúng bao gồm NETCONF (thông qua một NETCONF driver), yếu tố tính toán đường dẫn ( PCE) (thông qua một máy chủ Java bases PCE) và OpenFlow (thông qua một plugin được viết bởi Big Switch Network's Flood Light controller). Dịch vụ topology cũng cung cấp thông qua đa plugin, bao gồm BGP-TE/LS sử dụng Java-bases BGP-TE client, ALTO (cả client và server modules đều có giá trị), file tĩnh cung cấp khả năng thuận tiện, nhanh chóng cấu hình hệ thống. Ngoài ra còn có một trình bổ sung topology plugin cho Openflow controller, nó cung cấp trang bị LLDP-based topology cần thiết cho OpenFlow-base control tại thời điểm đó.



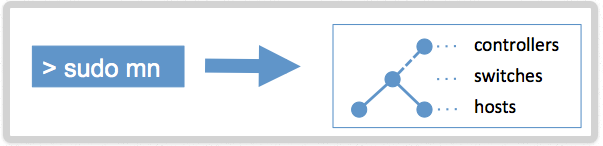
Khái niệm minh chứng ứng dụng mà Juniper đã chứng minh cái gọi là ứng dụng sắp xếp băng thông (BCA) (Được minh họa trong chương 10). Mục đích của BCA là tự mô tả - ứng dụng dựa trên sự đặt trước của băng thông qua một ví dụ hoặc mạng tĩnh tương đối. Những sự đặt trước dựa trên sự đặt trước trong tương lai hoặc ngay lập tức cần thiết hoặc có hoặc không thời gian kết thúc. Đây là một ứng dụng phổ biến trong các mạng nghiên cứu và trong một vài dịch vụ mạng nhà cung cấp Web lớn 2.0.

**V. DEMO**

Trong bài học này chúng tôi sẽ giới thiệu cách thiết lập một mạng OpenFlow được kích hoạt trong Mininet và một bộ điều khiển SDN. Mạng này sẽ chạy cả trên OpenFlow và mạng truyền thống.

**Giới thiệu về Mininet và HPE VAN Controller UI**

Mininet tạo ra một mạng ảo thực tế, chạy nhân thực, chuyển đổi và mã ứng dụng trên một máy duy nhất (VMware, cloud hoặc native) trong vài giây với một lệnh duy nhất:



Bởi vì bạn có thể dễ dàng tương tác với mạng của bạn bằng cách sử dụng Mininet CLI (và API), tùy chỉnh nó, chia sẻ nó với người khác, hoặc triển khai nó trên phần cứng thực, Mininet rất hữu ích cho việc phát triển, giảng dạy và nghiên cứu.

Mininet cũng là một cách tuyệt vời để phát triển, chia sẻ và thử nghiệm với các hệ thống mạng OpenFlow và phần mềm đã xác định.

Tiếp theo là bộ điều khiển HPE VAN SDN: cung cấp một điểm kiểm soát thống nhất trong một mạng có hỗ trợ OpenFlow, đơn giản hoá việc quản lý, cung cấp và dàn xếp và cho phép phân phối một thế hệ mới các dịch vụ mạng dựa trên ứng dụng.

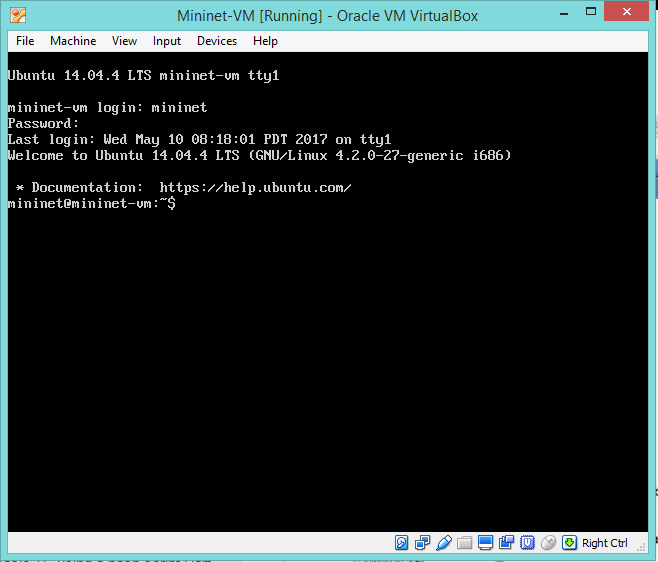
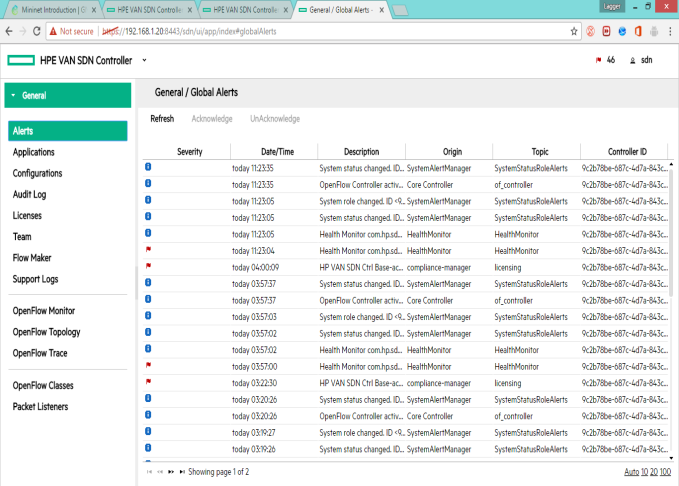
Bộ điều khiển HPE VAN SDN được thiết kế để hoạt động trong nhiều môi trường điện toán khác nhau, bao gồm khuôn viên, trung tâm dữ liệu, nhà cung cấp dịch vụ, đám mây riêng và đám mây công cộng.

Các bạn có thể download 2 phần mềm này về cài đặt trên máy tại 2 đường dẫn sau đây:

* http://mininet.org/download/
* https://marketplace.saas.hpe.com/sdn/content/hpe-van-sdn-controller-ova-free-trial#app\_releases

Ngoài ra chúng ta cần sử dụng virtualbox để tạo máy ảo cài đặt mininet.

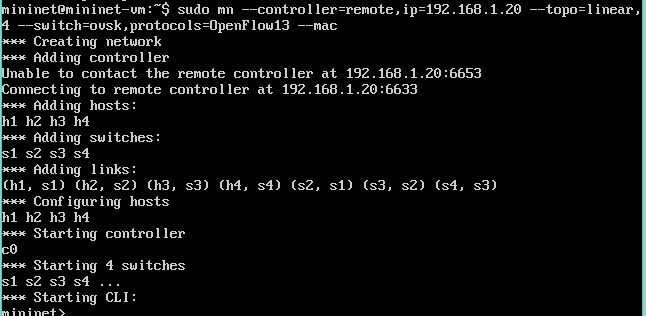
Đây là giao diện khi cài đặt thành công 2 phần mềm:

**Tạo một Topology đơn giản:**

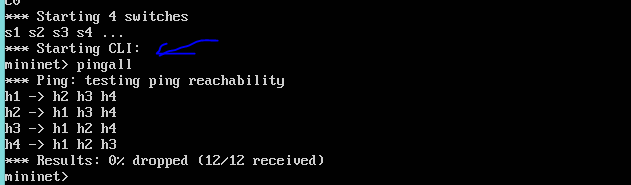
Tạo topology với 4 switch bằng lệnh:

"sudo mn --controller=remote,ip=192.168.1.20 --topo=linear,4 --switch=ovsk,protocols=OpenFlow13 --mac"



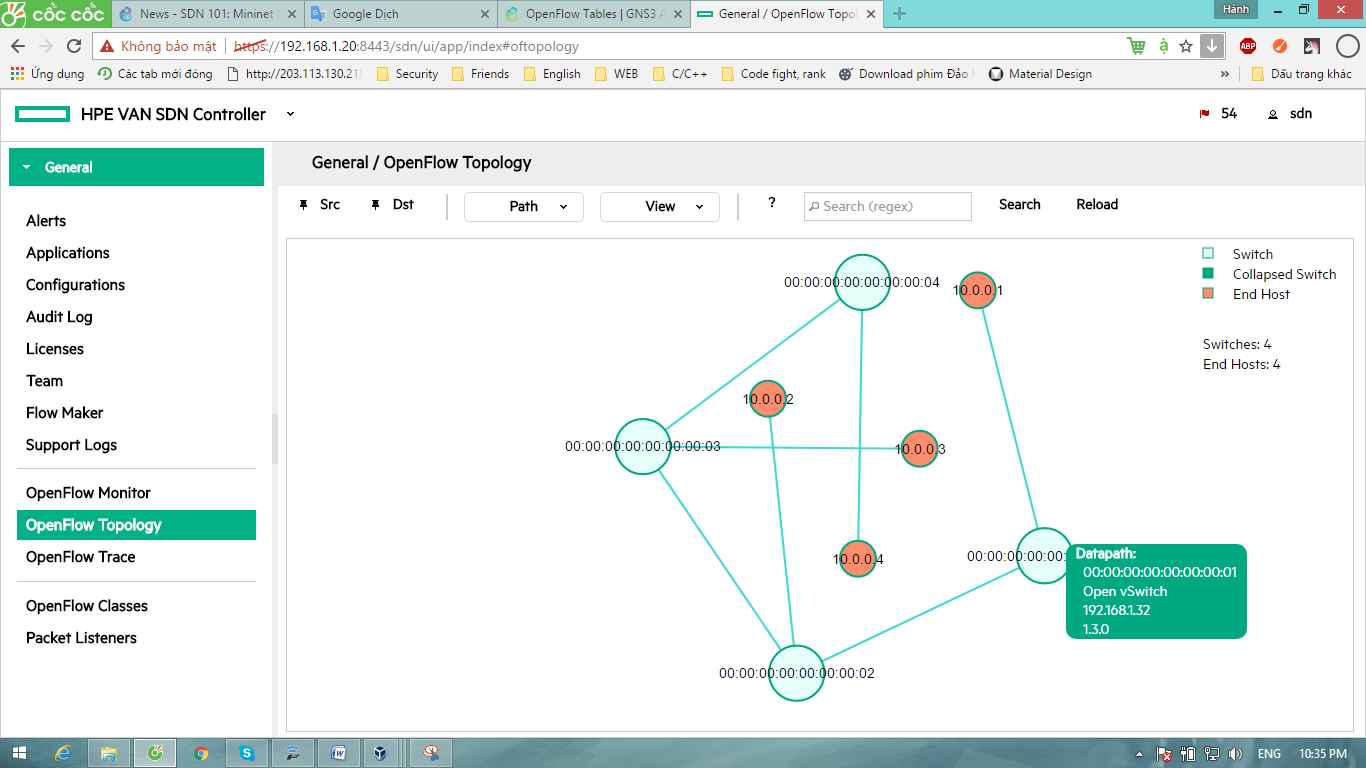
Thông tin các host (h1, h2, h3, h4), các switch (s1, s2, s3, s4), controller (c0) và link kết nối giữa chúng được thể hiện chi tiết.

Kiểm tra tình trạng kết nối mạng của tất cả các host với nhau qua lệnh: "pingall"

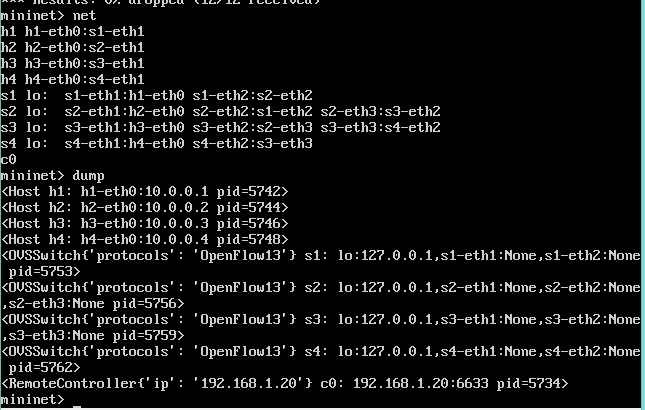


Còn nếu chỉ muốn kiểm tra tình trạng kết nối mạng giữa hai host với nhau ví dụ h1 và h2 thì chúng ta dùng lệnh: "h1 ping h2".

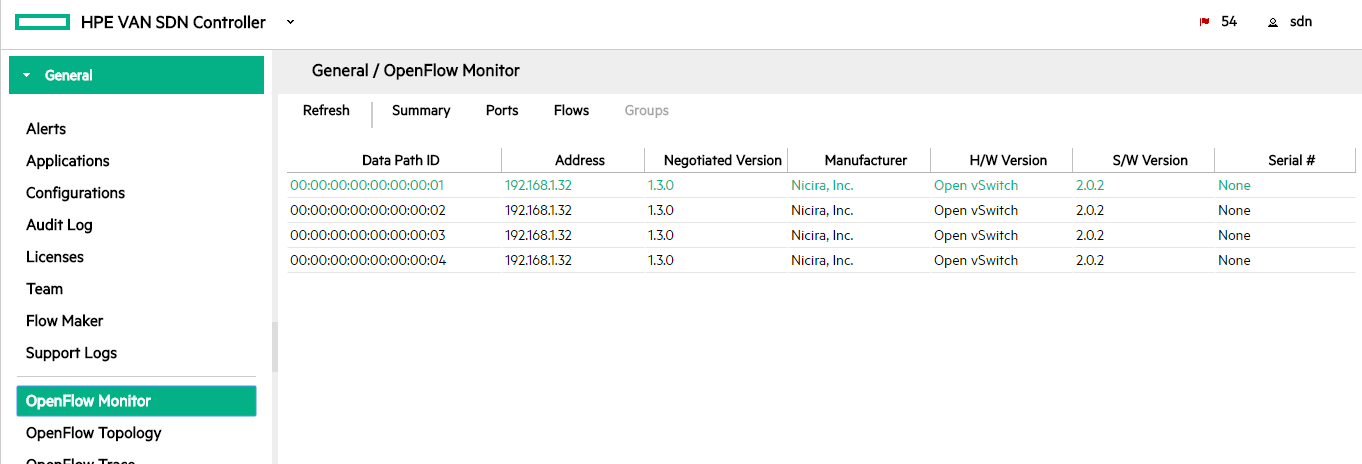
Để có cái nhìn cụ thể về topo mạng này ta vào trình duyệt (Chrome, CocCoc...) truy cập HPE VAN Controller, click vào mục OpenFlow Topology:

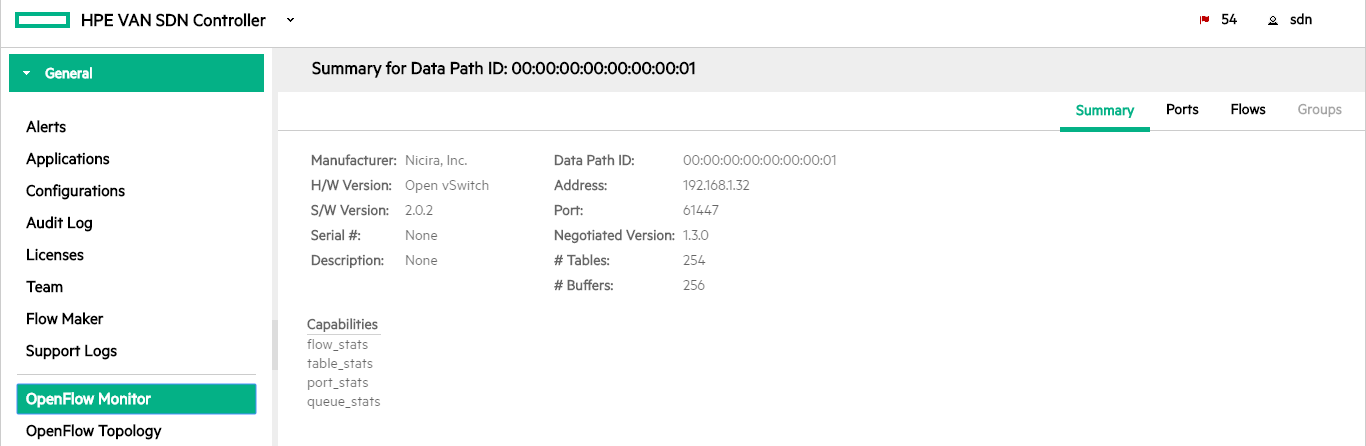


Nhìn vào UI ta có thể thấy được địa chỉ của các host và controller và kết nối giữa chúng. Chúng ta cũng có thể xem chi tiết mọi thứ qua 2 lệnh "net" và "dump".

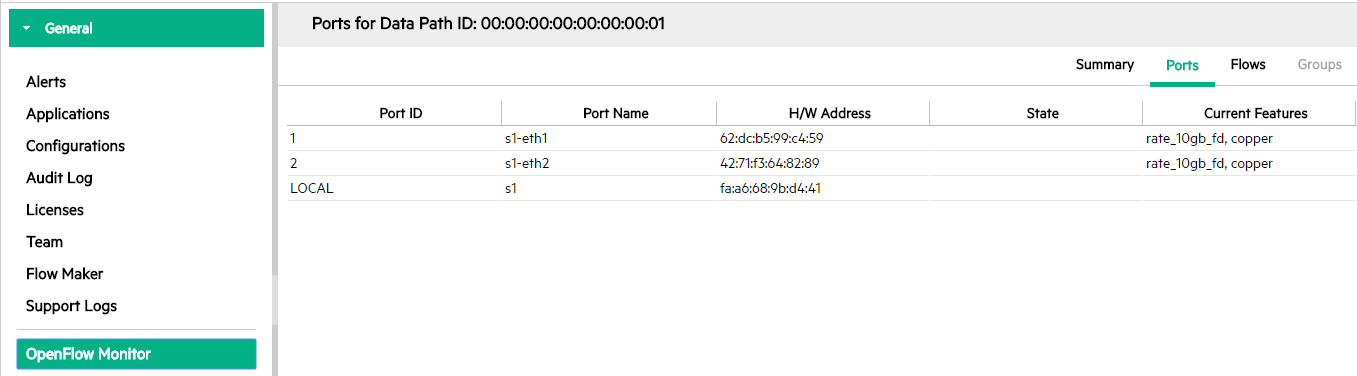


Trong UI vào OpenFlow Moniter chúng ta có được thông tin về OpenFlow của các switch với đầy đủ các chi tiết về OpenFlow Table, Data Path ID, Address, Version, , Serial...

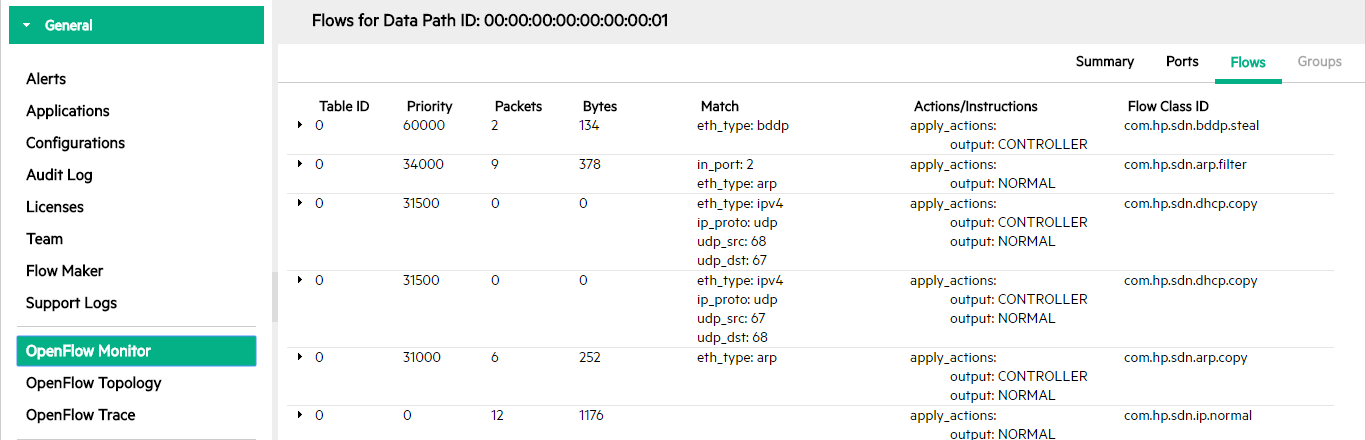




và thông tin về port kết nối:

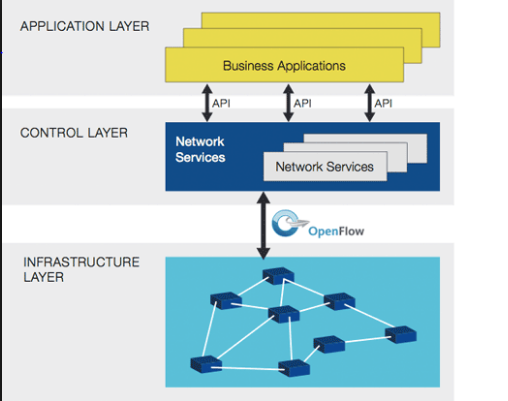


đặc biệt nội dung trong bảng OpenFlow: Id, thứ tự ưu tiên, số bytes, thông tin src và dst... tất cả có tác dụng cập nhật và điều khiển dòng chảy lưu lượng giữa SDN controller và các switch.



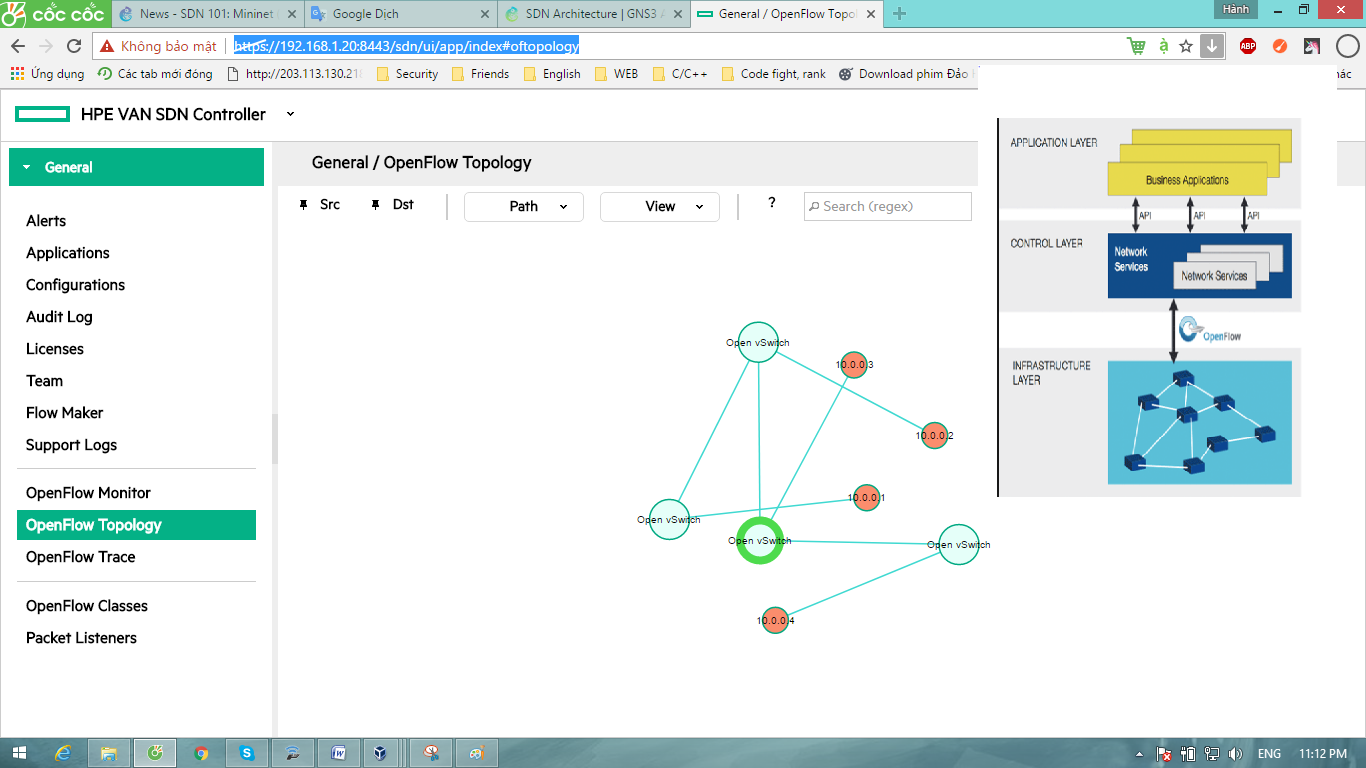
**Kiến trúc SDN**

Về cơ bản, SDN được chia làm ba lớp chính: Lớp ứng dụng (application layer), lớp điều khiển (control layer) và lớp thiết bị (infrastructure layer). Trong môi trường OpenFlow các phần sẽ liên kết với nhau thông qua giao thức hoặc các API.

****

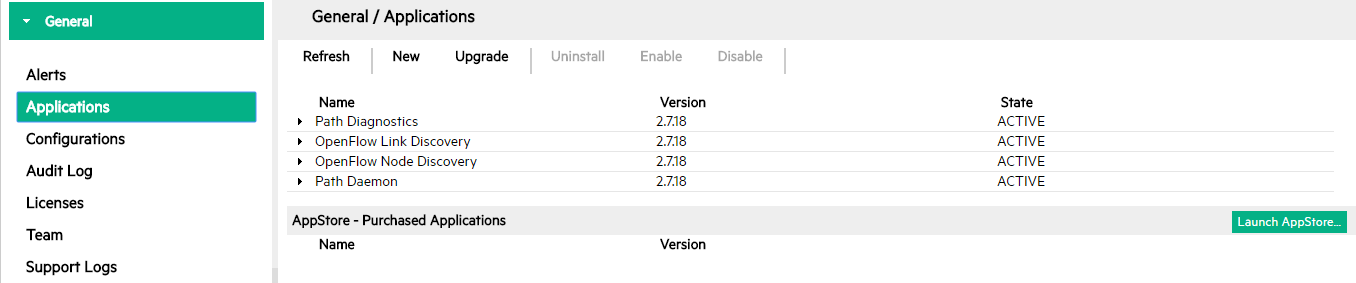
Mô hình kiến trúc SDN

Trong ví dụ này mininet chạy Open vSwitch, hay nói cách khác sử dụng một Open Switch ảo trong Infrastructer layer.



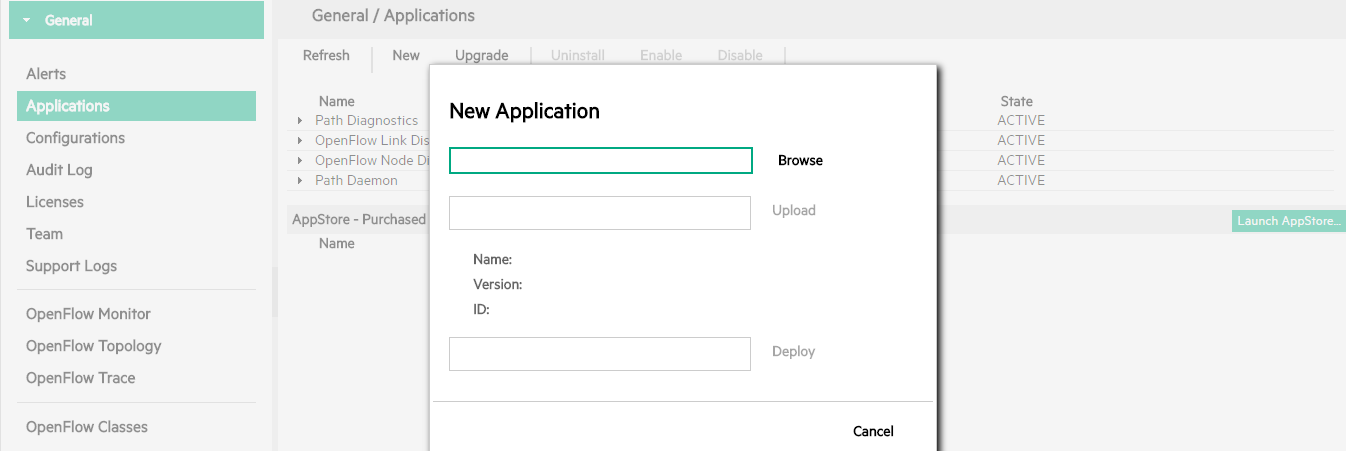
Bên trên Infracture layer là Controller layer, trong ví dụ là remote controller ip.

Tiếp theo xét tới Application layer trong kiến trúc SDN, click mục Application trong giao diện HPE VAN Controller.

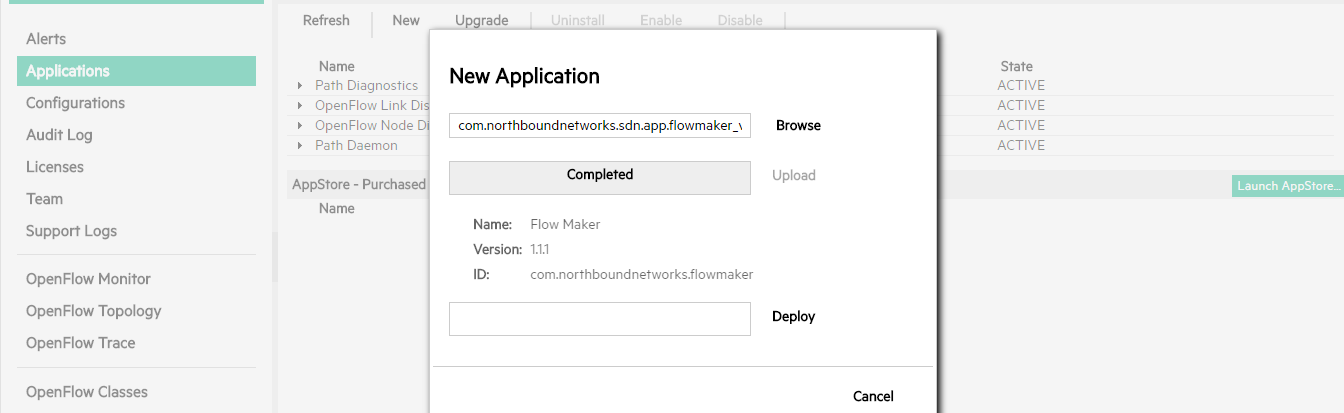


Đây là nơi tạo các ứng dụng được sử dụng trong SDN. Từ các API được cung cấp từ lớp điều khiển, người sử dụng có thể dễ dàng lập trình hoạt động cho hệ thống mạng để tối ưu hoạt động theo một yêu cầu nhất định.

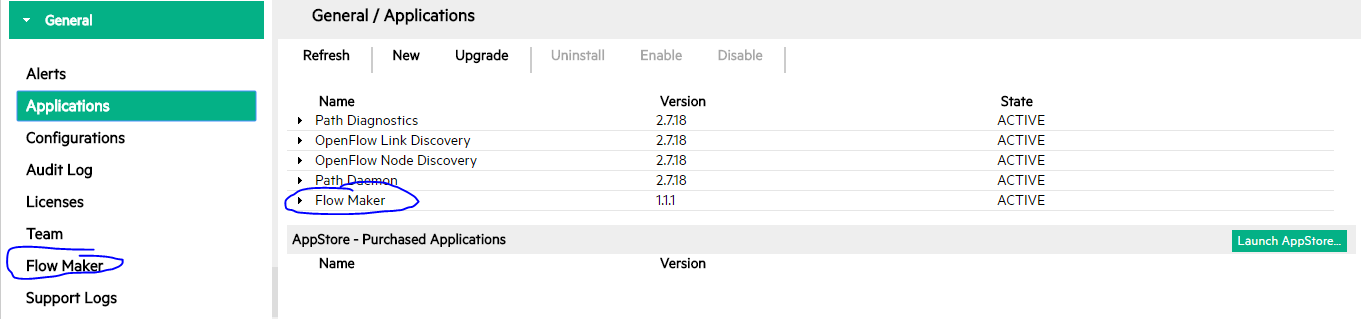
Chúng ta có thể cập nhật, thêm mới hoặc xóa các ứng dụng. Chúng tôi sẽ minh họa việc thêm mới một ứng dụng vào danh sách các ứng dụng có sẵn. Đầu tiên tải 1 ứng dụng cần cài trên AppStore bằng cách click vào mục Lauch AppStore bên phải màn hình, chọn ứng dụng cần thiết về cài. Khi tải ứng dụng về chúng ta tiến hành cài đặt lên controller. Trong trường hợp này chúng tôi đã tải ứng dụng có tên là Flow Maker. Công việc tiếp theo upload file chứa ứng dụng Flow Maker lên HPE VAN Controller, tiến hành bằng cách click "New" trên thanh menu giữa màn hình:



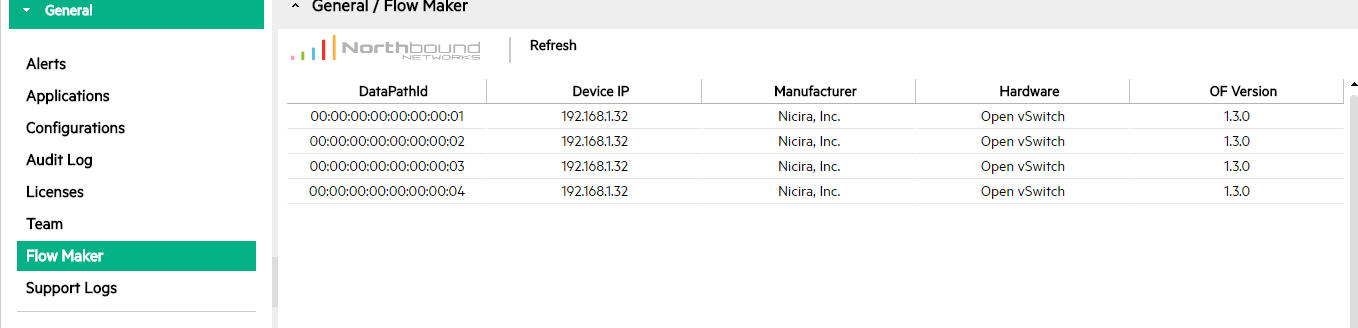
Tiến hành browse ứng dụng tải về trong thư mục máy tính để upload lên.



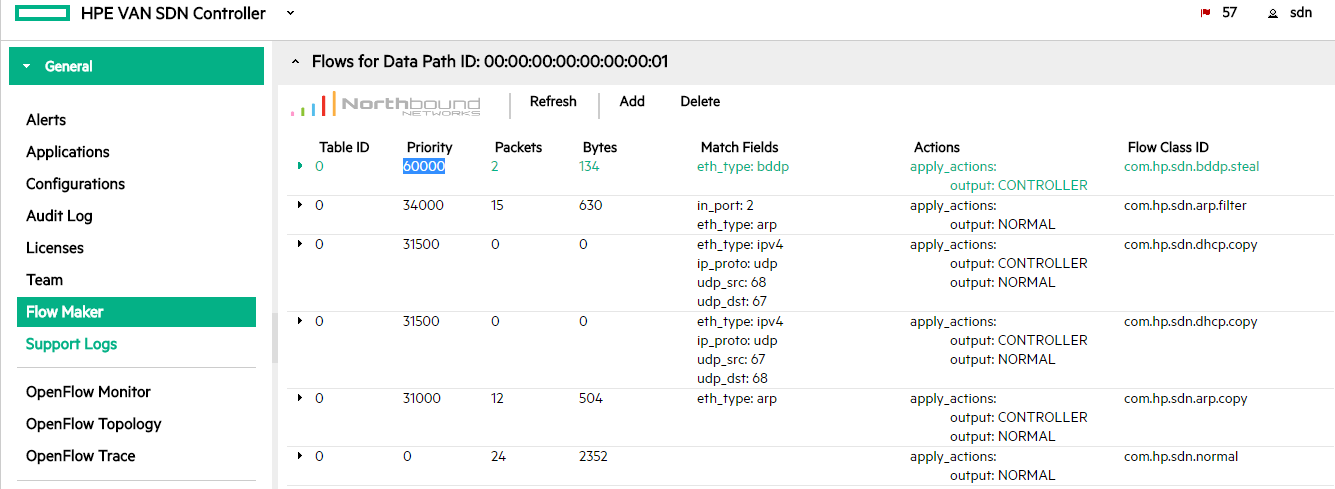
"Completed" có nghĩa tập tin chứa ứng dụng cần cài đã sẵn sàng load lên. Từ đó tiến hành Load lên HPE VAN Controller bằng cách click "Deploy" và kết quả một ứng dụng mới được cài đặt trên controller.



Kiểm tra thử ngay ứng dụng



Hiển thị một danh sách luồng mạng (flow entries) của các switch. Khi click vào 1 luồng mạng ta thấy:



Thực ra ứng dụng này cũng nằm trong sẵn controller rồi. Như bạn đã biết ở những phần demo bên trên trong mục "OpenFlow Monitor->Flows" cũng chính là ứng dụng chúng ta vừa cài thêm.