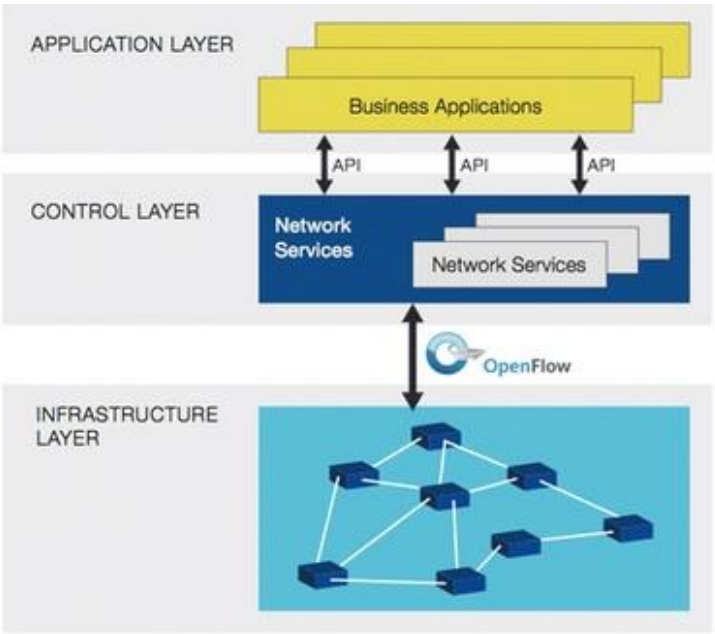
Contents

1. Tổng quan về SDN

Software Defined Network (SDN) là một cấu trúc mới, được thiết kế cho phép hệ thống mạng trở nên linh động và có hiệu quả chi phí hơn. SDN là một khái niệm mang tính lý thuyết, về mặt bản chất, SDN tách riêng các control plane phân tán từ các forwarding plane và đưa các chức năng của control plane vào trong control plane tập trung. Control plane và forwarding plane là 2 dạng tiến trình mà các thiết bị mạng đều thực hiện. Ví dụ, tại cấu trúc mạng truyền thống, nếu ta truy cập vào router, trực hiện các tác vụ như cấu hình các giao thức gateway, etc. thì các hoạt động này đều được thực hiện trên cùng một thiết bị (trên control plane và forwarding plane của router), do đó các nút trong network hoạt động một cách độc lập dựa trên các cấu hình nội bộ tại chính các nút đó. Điều đó có nghĩa, cho dù cấu trúc mạng có linh động, hiệu quả như thế nào thì kết quả của các tác vụ hoạt động trong mạng phải phụ thuộc vào cấu hình của từng nút. Nếu như số lượng nút nhiều (1000, 10000 nodes) thì đồng nghĩa các nhà vận hành mạng phải quản lý toàn bộ (1000, 10000) control plane (process quản lý hầu như toàn bộ hoạt động của thiết bị mạng).

Chính vì những khó khăn trên, SDN ra đời nhằm mục đích chuyển cấu trúc control plane từ phân tán sang tập trung. Control plane tập trung (SDN controller) cho phép chuyển tiếp các quyết định về flow thông qua SDN domain thay vì phải qua từng hop.

Các tập đoàn lớn như Cisco, HP, IBM, VMware đều tự phát triển các SDN controller riêng của họ. Ngoài ra, còn có một số dự án mã nguồn mở về SDN controller như Floodlight hay OpenDayLight. Trong SDN, controller được ví như "bộ não", nó thống kế tất cả thông tin từ các flow switch ở trong mạng, cung cấp cái nhìn tổng thể bên trong của network (trong mạng truyền thống, một switch đơn lẻ sẽ không thể nhận biết toàn cảnh network , sự kết nối giữa các switch với nhau).



Kiến trúc của SDN gồm 3 lớp riêng biệt: lớp ứng dụng, lớp controller, và lớp cơ sở hạ tầng (lớp chuyển tiếp). Trong đó:

- Lớp ứng dụng: là các ứng dụng kinh doanh được triển khai trên mạng, được kết nối tới lớp controller thông qua các API, cung cấp khả năng cho phép lớp ứng dụng lập trình lại (cấu hình lại) mạng (điều chỉnh các tham số trễ, băng thông, định tuyến, …) thông qua lớp controller.
- Lớp controller: là nơi tập trung các bộ controller thực hiện việc controller cấu hình mạng theo các yêu cầu từ lớp ứng dụng và khả năng của mạng. Các bộ controller này có thể là các phần mềm được lập trình. Ngoài ra để truyền thông controller lớp cơ sở hạ tầng, lớp controller sử dụng các cơ chế như OpenFlow, ONOS, ForCES, PCEP, NETCONF, SNMP hoặc thông qua các cơ chế riêng biệt.
- Lớp cơ sở hạ tầng: là các thiết bị mạng thực tế (vật lý hay ảo hóa) thực hiện việc chuyển tiếp gói tin theo sự controller của lớp điều khiển. Một thiết bị mạng có thể hoạt động theo sự controller của nhiều bộ controller khác nhau, điều này giúp tăng cường khả năng ảo hóa của mạng.

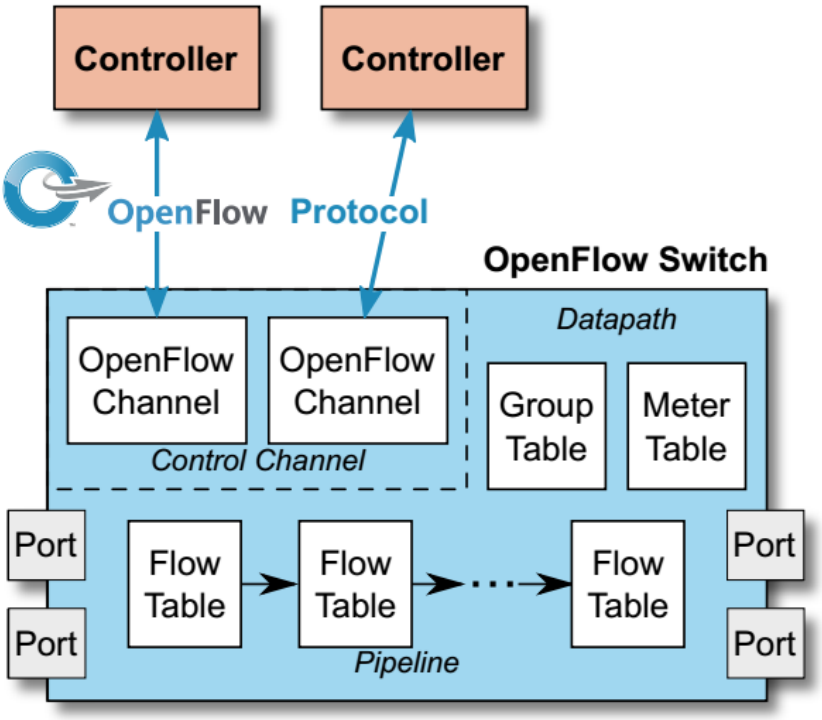
Với kiến trúc như trên, SDN cung cấp các khả năng:

- Lớp controller có thể được lập trình trực tiếp.
- Mạng được điều chỉnh, thay đổi một cách nhanh chóng thông qua việc thay đổi trên lớp controller.
- Mạng được quản lý tập trung do phần controller được tập trung trên lớp controller.
- Cấu hình lớp cơ sở hạ tầng có thể được lập trình trên lớp ứng dụng và truyền đạt xuống các lớp dưới.

Với những tính năng mới, SDN đem lại các lợi ích sau:

- Giảm CapEx: SDN giúp giảm thiểu các yêu cầu mua phần cứng theo mục đích xây dựng các dịch vụ, phần cứng mạng trên cơ sở ASIC, và hỗ trợ mô hình pay-as-yougrow (trả những gì bạn dùng) để loại bỏ lãng phí cho việc dự phòng.
- Giảm OpEx: thông qua các phần tử mạng đã được gia tăng khả năng lập trình, SDN giúp dễ dàng thiết kế, triển khai, quản lý và mở rộng mạng. Khả năng phối hợp và dự phòng tự động không những giảm thời gian quản lý tổng thể, mà còn giảm xắc suất lỗi do con người tới việc tối ưu khả năng và độ tin cậy của dịch vụ.
- Truyền tải nhanh chóng và linh hoạt: giúp các tổ chức triển khai nhanh hơn các ứng dụng, các dịch vụ và cơ sở hạ tầng để nhanh chóng đạt được các mục tiêu kinh doanh.
- Cho phép thay đổi: cho phép các tổ chức tạo mới các kiểu ứng dụng, dịch vụ và mô hình kinh doanh, để có thể tạo ra các luồng doanh thu mới và nhiều giá trị hơn từ mạng.

2. Giới thiệu Switch OpenFlow



2.1 Các thành phần Switch

Switch Logic OpenFlow gồm một hoặc nhiều bảng flow, một bảng group, thực hiện tra cứu và chuyển tiếp gói tin, một hoặc nhiều kênh OpenFlow tới bộ controller bên ngoài. Switch giao tiếp với bộ controller và bộ quản lý controller thông qua giao thức OpenFlow Switch Protocol.

Sử dụng giao thức OpenFlow Switch Protocol, bộ Controller có thể thêm, cập nhật và xóa các mục flow trong các bảng flow. Mỗi bảng flow trong switch chứa một tập các mục flow, mỗi mục flow bao gồm các trường *match fields*, *counters* và một bộ *instruction* để áp dụng cho các gói tin tương ứng.

Việc so khớp bắt đầu tại bảng flow đầu tiên và có thể tiếp tục ở các bảng tiếp theo của quá trình *pipeline*. Mục flow phù hợp với gói tin theo thứ tự ưu tiên. Nếu tìm thấy một mục flow phù hợp, các hướng dẫn trong mục flow đó được thực hiện. Nếu không tìm thấy mục flow phù hợp trong bảng flow, kết quả phụ thuộc vào cấu hình của mục flow *table-miss*. Ví dụ, gói tin có thể được chuyển tiếp đến các bộ controller trên kênh OpenFlow, bị bỏ qua, hoặc có thể tiếp tục tới bảng flow tiếp theo.

Các *instruction* liên quan đến mỗi mục flow chứa các *action* hoặc chỉnh sửa quá trình pipeline. Các *action* trong các *instruction* mô tả việc chuyển tiếp gói, chỉnh sửa gói tin và

xử lý bảng group. Các *instruction* trong quá trình xử lý pipeline cho phép gửi các gói tin tới các bảng tiếp theo để xử lý tiếp và cho phép trao đổi thông tin giữa các bảng dưới dạng metadata. Quá trình xử lý pipeline dừng lại khi bộ *instruction* của mục Flow phù hợp không chỉ định một bảng tiếp theo, tại thời điểm này gói thường được chỉnh sửa và chuyển tiếp.

Các mục flow có thể chuyển tiếp gói tin đến một cổng. Đây thường là một cổng vật lý, nhưng nó cũng có thể là một cổng logic hoặc một cổng reserved. Các cổng reserved chỉ định các tác vụ chuyển tiếp như gửi gói tin đến bộ controller, quảng bá gói tin toàn mạng (*flooding*) hoặc chuyển tiếp bằng các phương pháp khác không phải OpenFlow, chẳng hạn như các switch bình thường, trong khi các cổng logic chỉ định chyển tiếp gói tin tới các nhóm *link aggregation*, *tunnels* hoặc các *loopback interfaces*.

Các action của mục flow cũng có thể chuyển gói tin tới một group. Các group đại diện cho các bộ action quảng bá toàn mạng, cũng như chuyển tiếp phức tạp hơn (ví dụ: *multipath*, *fast reroute* và *link aggregation*). Các group cũng cho phép nhiều mục flow chuyển tiếp tới một thiết bị duy nhất (ví dụ như chuyển tiếp IP đến một next hop).

Bảng group chứa các mục group, mỗi mục group chứa một danh sách các *action bucket* với ý nghĩa cụ thể phụ thuộc vào loại group. Các action trong một hoặc nhiều *bucket* được áp dụng cho các gói được gửi đến group.

2.2 Cổng OpenFlow

Các cổng OpenFlow là các *network interface* để truyền các gói giữa quá trình xử lý OpenFlow và phần còn lại của mạng. Các thiết bị switch OpenFlow kết nối logic với nhau thông qua các cổng OpenFlow, một gói tin chỉ có thể chuyển tiếp từ một bộ switch OpenFlow sang một bộ switch OpenFlow khác thông qua cổng OpenFlow ra trên switch thứ nhất và một cổng OpenFlow vào trên switch thứ hai.

Các gói tin được tiếp nhận trên một *ingress port*, được xử lý bởi OpenFlow pipeline, và có thể chuyển chúng tới một *output port*. *Ingress port* được sử dụng khi so khớp với các gói tin. Quá trình xử lý pipeline quyết định chuyển gói tin trên một cổng đầu ra bằng cách sử dụng *action output*, nó định nghĩa cách gói trở lại mạng.

Bộ switch OpenFlow hỗ trợ ba loại cổng OpenFlow: cổng vật lý, cổng logic và cổng reserved.

Các cổng vật lý OpenFlow được định nghĩa là các cổng tương ứng với một interface phần cứng của switch. Ví dụ, trên một switch Ethernet, các port vật lý là các interface Ethernet.

Các cổng logic OpenFlow là các cổng không tương ứng trực tiếp với một giao diện phần cứng của switch. Cổng logic được sử dụng trong các phương thức khác (ví dụ: *link*

aggregation, *tunnels* hoặc các *loopback interfaces*). Các cổng lôgic có thể đóng gói và chuyển gói tin đến các cổng vật lý khác nhau.

Các cổng reserved OpenFlow chỉ định chuyển tiếp gói tin như gửi đến bộ controller, quảng bá toàn mạng hoặc chuyển tiếp bằng các phương thức khác. Switch không yêu cầu hỗ trợ tất cả các cổng reserved, trừ những cổng được đánh dấu "bắt buộc" bên dưới.

• **ALL** (bắt buộc): Đại diện cho tất cả các cổng mà switch có thể sử dụng để chuyển tiếp một gói cụ thể. Có thể được sử dụng như một cổng đầu ra. Trong trường hợp đó một bản sao của gói bắt đầu quá trình xử lý egress trên tất cả các cổng, ngoại trừ cổng ingress và các cổng được cấu hình OFPPC_NO_FWD.

• **CONTROLLER** (bắt buộc): Biểu diễn kênh controller với bộ controller OpenFlow. Có thể được sử dụng như một cổng vào hoặc là một cổng ra. Khi được sử dụng như một cổng đầu ra, đóng gói gói tin trong bản tin *packet-in* và gửi nó bằng cách sử dụng giao thức OpenFlow. Khi được sử dụng như một cổng vào, điều này xác định một gói có nguồn gốc từ bộ controller.

• **TABLE** (bắt buộc): Biểu thị sự bắt đầu của quá trình pipeline. Cổng này chỉ có giá trị trong action output trong danh sách các action của một thông báo packet-out và gửi gói tin tới bảng Flow đầu tiên để gói có thể được xử lý thông qua pipeline OpenFlow.

• **IN PORT** (bắt buộc): Biểu hiện cổng vào của gói tin. Có thể được sử dụng chỉ như một cổng đầu ra, gửi gói ra qua cổng vào của nó.

• **ANY** (bắt buộc): Giá trị đặc biệt được sử dụng trong một số yêu cầu OpenFlow khi không có cổng nào được chỉ định (ví dụ cổng là wildcarded). Một số yêu cầu OpenFlow chứa một tham chiếu đến một cổng cụ thể mà yêu cầu chỉ áp dụng cho. Sử dụng ANY vì số cổng trong các yêu cầu này cho phép thể hiện yêu cầu đó áp dụng cho bất kỳ và tất cả các cổng. Không thể được sử dụng như một cổng vào hay cổng đầu ra.

• **UNSET** (bắt buộc): Giá trị đặc biệt để xác định rằng cổng đầu ra chưa được đặt trong Action-Set. Chỉ được sử dụng khi cố gắng kết hợp cổng đầu ra trong bộ action sử dụng trường so khớp OXM_OF_ACTSET_OUTPUT. Không thể được sử dụng như một cổng vào hay cổng đầu ra.

• **LOCAL** (không bắt buộc): Biểu diễn ngăn xếp mạng cục bộ và ngăn xếp quản lý. Có thể được sử dụng như một cổng vào hoặc là một cổng đầu ra. Cổng cục bộ cho phép các thực thể từ xa tương tác với switch và các dịch vụ mạng của nó thông qua mạng OpenFlow, chứ không phải qua một mạng controller riêng biệt. Với một bộ các flow thích hợp, nó có thể được sử dụng để thực hiện kết nối bộ controller trong băng (điều này nằm ngoài phạm vi của đặc tả này).

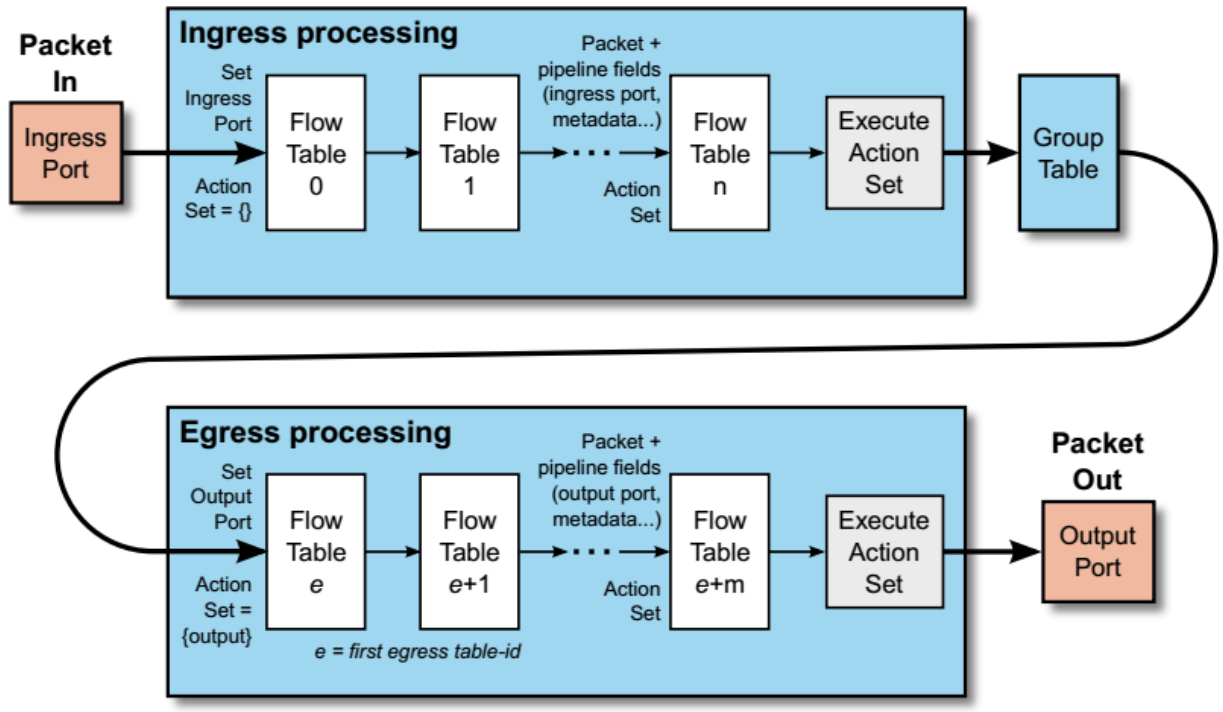
• **NORMAL** (không bắt buộc): Đại diện cho việc chuyển tiếp sử dụng pipeline không phải là OpenFlow truyền thống của switch. Có thể được sử dụng chỉ như là một cổng đầu ra và xử lý các gói tin bằng cách sử dụng pipeline bình thường. Nhìn chung, sẽ nối cầu hoặc định tuyến gói tin, tuy nhiên kết quả thực tế là phụ thuộc vào thực hiện. Nếu switchkhông thể chuyển tiếp các gói tin từ pipeline dẫn OpenFlow đến pipeline bình thường, nó phải chỉ ra rằng nó không hỗ trợ action này.

• **FLOOD** (không bắt buộc): Đại diện cho quảng bá gói tin không sử dụng pipeline, sẽ gửi gói ra tất cả các cổng, trừ cổng gói tin vào và các cổng ở trạng thái OFPPS_BLOCKED. Switch cũng có thể sử dụng VLAN ID hoặc các tiêu chí khác để chọn cổng sử dụng cho quảng bá.

Switch *OpenFlow-only* không hỗ trợ cổng NORMAL và FLOOD, trong khi các bộ switch *OpenFlow-hybrid* có thể hỗ trợ chúng. Chuyển tiếp các gói tin đến cổng FLOOD phụ thuộc vào việc cấu hình switch, trong khi chuyển tiếp sử dụng một *group* loại *all* cho phép bộ controller thực hiện quảng bá linh hoạt hơn.

2.3 Bảng OpenFlow

Phần này mô tả các thành phần của bảng Flow và các group table, cùng với cơ chế kết hợp và xử lý action



2.3.1 Xử lý pipeline

Thiết bị switch hỗ trợ OpenFlow gồm hai loại:

- OpenFlow-only
- OpenFlow-hybrid

Switch OpenFlow-only chỉ hỗ trợ OpenFlow, trong đó tất cả các gói tin được xử lý bằng pipeline OpenFlow và không thể xử lý được theo cách khác.

Switch OpenFlow-hybrid hỗ trợ cả OpenFlow và switch Ethernet bình thường (switch L2 Ethernet, VLAN, định tuyến L3, ACL và QoS).

Pipeline OpenFlow của mỗi switch Logic OpenFlow chứa một hoặc nhiều bảng flow, mỗi bảng flow chứa nhiều mục flow. Việc xử lý pipeline OpenFlow xác định cách các gói tin tương tác với các bảng flow này. Switch OpenFlow có ít nhất một bảng flow ingress.

Các bảng flow của switch OpenFlow được đánh số theo thứ tự bắt đầu từ 0. Quá trình xử lý pipeline diễn ra trong hai giai đoạn, xử lý *ingress* và xử lý *egress*. Việc phân chia hai giai đoạn được chỉ ra bởi bảng flow ra đầu tiên, tất cả các bảng có số thứ tự thấp hơn so với bảng egress đầu tiên phải được sử dụng như các bảng ingress.

Quá trình pipeline luôn bắt đầu với việc xử lý ingress ở bảng flow đầu tiên: gói tin phải phù hợp với mục flow của bảng flow 0. Các bảng flow ingress khác có thể được sử dụng tùy thuộc vào kết quả của việc so khớp trong bảng đầu tiên. Nếu kết quả xử lý ingress là chuyển gói tin tới một cổng đầu ra, switch có thể thực hiện việc xử lý egress trong cổng đầu ra đó.

Quá trình xử lý egress là không bắt buộc, một switch có thể không có bất kỳ bảng egress nào hoặc có thể không được cấu hình để sử dụng chúng. Nếu không có bảng egress hợp lệ được cấu hình như bảng egress đầu tiên, gói tin được xử lý bởi cổng đầu ra. Trong hầu hết các trường hợp, gói tin được chuyển tiếp ra khỏi switch. Nếu một bảng egress hợp lệ được cấu hình, gói tin phải được so khớp với các mục flow của bảng flow đó, và các bảng egress khác có thể được sử dụng tùy thuộc vào kết quả so khớp trong bảng flow đó.

Khi được xử lý bởi một bảng flow, gói tin được so khớp với các mục flow của bảng flow. Nếu một mục flow được tìm thấy, tập instruction trong mục flow đó được thực hiện. Các instruction này có thể trỏ trực tiếp gói tin tới một bảng flow khác, nơi một quá trình tương tự được lặp lại. Một mục flow chỉ có thể trỏ một gói tin đến bảng Flow có số thứ tự lớn hơn số thứ tự của bảng đó. Nếu mục flow phù hợp không trỏ các gói tin đến một bảng flow khác, giai đoạn hiện tại của quá trình xử lý pipeline dừng lại ở bảng này, gói tin được xử lý với bộ action liên quan của nó.

Nếu một gói tin không khớp với một mục flow nào trong một bảng flow, đây là một table-miss. Các action của một table-miss phụ thuộc vào cấu hình. Các instruction trong mục

table-miss có thể xác định cách xử lý các gói tin chưa được so khớp, các tùy chọn bao gồm việc bỏ qua gói tin, chuyển chúng đến một bảng khác hoặc gửi chúng tới bộ controller qua kênh controller thông qua bản tin packet-in.

2.3.2 Bảng flow và mục flow

Một bảng flow bao gồm các mục flow.

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Mỗi mục flow chứa:

- match fields: trường so khớp với các gói tin. Bao gồm cổng vào, tiêu đề gói, và các trường pipeline khác tùy chọn như metadata được chỉ định bởi bảng trước đó.
- priority: thứ tự ưu tiên của mục flow.
- counters: bộ đếm được cập nhật khi các gói tin được so khớp.
- instructions: các hướng dẫn chỉnh sửa bộ action hoặc xử lý pipeline.
- timeouts: thời gian tối đa hoặc thời gian chờ trước khi luồng hết hạn.
- cookie: giá trị dữ liệu được chọn bởi bộ controller. Có thể được sử dụng bởi bộ controller để lọc các mục flow bị ảnh hưởng bởi số liệu thống kê, chỉnh sửa và yêu cầu xóa flow. Trường này không được sử dụng khi xử lý các gói tin.
- flags: cờ thay đổi cách quản lý flow.

Một mục flow được xác định bởi các trường match fields và priority, tức là các trường match fields và priority cùng nhau xác định một mục flow duy nhất trong một bảng flow cụ thể. Các mục flow mà wildcards của tất cả các trường (tất cả các trường bị bỏ qua) và có mức độ ưu tiên bằng 0 được gọi là table-miss.

2.3.3 So khớp

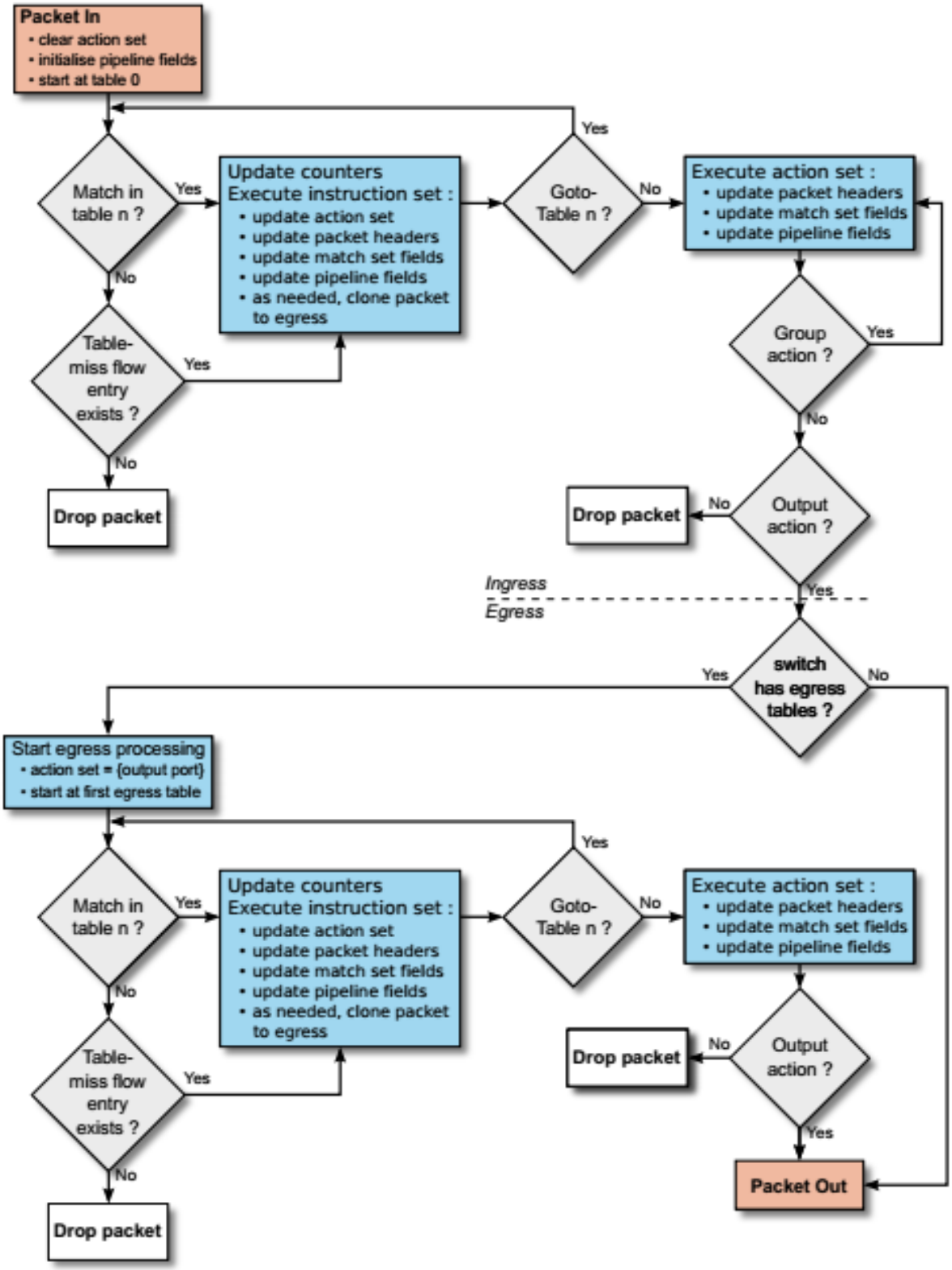
Khi nhận được một gói tin, một Switch OpenFlow thực hiện các chức năng thể hiện trong hình. Switch bắt đầu thực hiện tra cứu trong bảng flow đầu tiên, và dựa trên quá trình xử lý pipeline switch có thể thực hiện tra cứu trong các bảng flow khác.

Các trường tiêu đề được tách ra từ gói tin và trường pipeline được lấy lại. Trường tiêu đề được sử dụng cho tra cứu bảng phụ thuộc vào loại gói tin, và thường bao gồm các trường tiêu đề khác nhau của giao thức, chẳng hạn như địa chỉ nguồn Ethernet hoặc địa chỉ đích IPv4. Ngoài các tiêu đề gói tin, các so khớp cũng có thể được thực hiện đối với cổng nhập, trường siêu dữ liệu (metadata) và các trường pipeline khác. Siêu dữ liệu có thể được dùng để truyền thông tin giữa các bảng trong một switch. Các trường tiêu đề gói tin và trường

pipeline đại diện cho gói trong trạng thái hiện tại của nó. Nếu các action được áp dụng trong một phần trước đó bằng cách sử dụng lệnh Apply-Actions làm thay đổi tiêu đề của gói tin hoặc trường pipeline, những thay đổi này được phản ánh trong các trường tiêu đề gói tin và trường pipeline.

Một gói tin khớp với một mục flow nếu tất cả các trường match fields của nó khớp với các trường tiêu đề và trường pipeline tương ứng của gói tin. Nếu trường match fields bị bỏ qua trong mục flow (nghĩa là giá trị ANY), nó khớp với tất cả các giá trị có thể có trong trường tiêu đề hoặc trường pipeline của gói tin. Nếu trường match fields có mặt và không bao gồm mặt nạ, nó sẽ khớp với trường tiêu đề hoặc trường pipeline tương ứng từ gói nếu nó có cùng giá trị. Nếu switch hỗ trợ bitmask tùy ý trên các trường so khớp cụ thể, các mặt nạ này có thể chỉ rõ chính xác hơn các so khớp, trường match fields được khớp nếu nó có cùng giá trị cho các bit được đặt trong mặt nạ.

Gói tin được so khớp với các mục flow trong bảng flow và chỉ mục flow có priority cao nhất phù hợp được chọn. Phải cập nhật các bộ đếm liên quan đến mục flow đó và thực hiện tập các instruction chứa trong mục flow.


Packet In
• clear action set
• initialise pipeline fields
• start at table 0
Match in table n ?
Yes
No
Update counters
Execute instruction set :
• update action set
• update packet headers
• update match set fields
• update pipeline fields
• as needed, clone packet to egress
Goto-Table n ?
Yes
No
Execute action set :
• update packet headers
• update match set fields
• update pipeline fields
Table-miss flow entry exists ?
Yes
No
Drop packet
Group action ?
Yes
No
Output action ?
No
Drop packet
Ingress
Egress
Yes
switch has egress tables ?
Yes
No
Start egress processing
• action set = {output port}
• start at first egress table
Match in table n ?
Yes
No
Update counters
Execute instruction set :
• update action set
• update packet headers
• update match set fields
• update pipeline fields
• as needed, clone packet to egress
Goto-Table n ?
Yes
No
Execute action set :
• update packet headers
• update match set fields
• update pipeline fields
Table-miss flow entry exists ?
Yes
No
Drop packet
Output action ?
No
Drop packet
Yes
Packet Out

2.3.4 Table-miss

Mọi bảng flow phải hỗ trợ một mục table-miss. Mục table-miss chỉ định cách xử lý các gói tin không khớp với bất cứ mục flow khác trong bảng.

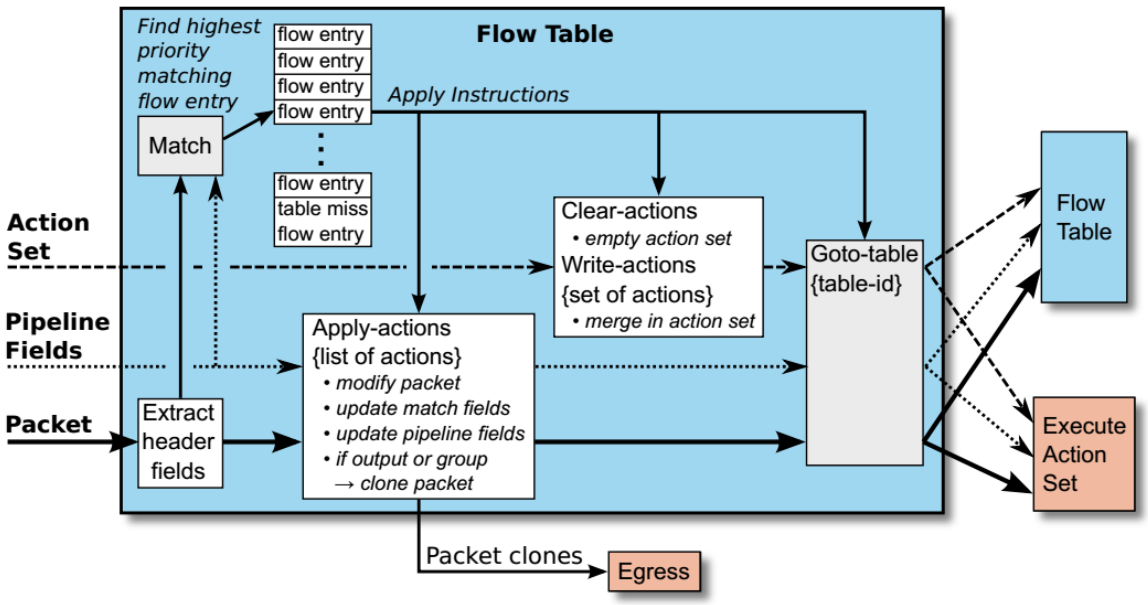
Mục table-miss được xác định bởi các trường match fields và priority của nó. Table-miss sử dụng wildcard cho tất cả các trường match fields (tất cả các trường bị bỏ qua) và có mức ưu tiên thấp nhất (0).

Mục table-miss trong bảng thực hiện theo hầu hết các cách như bất kỳ mục flow nào khác: nó không tồn tại mặc định trong một bảng flow, bộ controller có thể thêm hoặc xoá nó bất cứ lúc nào, nó có thể hết hạn, nó có thể bị trục xuất và nó có thể được đọc bởi bộ controller.

Nếu mục table-miss không tồn tại, theo mặc định các gói tin không khớp với các mục flow nào sẽ bị bỏ qua.

2.3.5 Instruction

Mỗi mục flow chứa một tập hợp các instruction được thực hiện khi một gói tin khớp với mục flow. Các hướng dẫn này dẫn đến thay đổi gói tin, thiết lập action và / hoặc xử lý pipeline.



Switch không bắt buộc phải hỗ trợ tất cả các loại instruction, trừ những loại được đánh dấu là "instruction bắt buộc" dưới đây:

- **Apply-Actions** *action* (instruction không bắt buộc): Áp dụng các action cụ thể ngay lập tức mà không có bất kỳ thay đổi nào đối với bộ action. Hướng dẫn này có thể được sử dụng để chỉnh sửa gói giữa hai bảng hoặc để thực hiện nhiều action cùng loại. Các action được chỉ định như một danh sách các action.
- **Clear-Action** (instruction bắt buộc): Xóa tất cả các action trong bộ action ngay lập tức.
- **Write-Actions** *action* (instruction bắt buộc): Hợp nhất (các) action cụ thể vào bộ action hiện tại. Nếu một action của một loại nào đó tồn tại trong tập hiện tại, hãy ghi đè nó, nếu không thì thêm nó.
- **Write-Metadata** *metadata / mask* (instruction không bắt buộc): Viết giá trị siêu dữ liệu được che dấu vào trường metadata. Mặt nạ xác định các bit của đăng ký siêu dữ liệu phải được chỉnh sửa.
- **Stat-Trigger** *stat thresholds* (instruction không bắt buộc): Tạo một sự kiện cho bộ controller nếu một số thống kê flow vượt qua một trong các giá trị ngưỡng stat.
- **Goto-Table** *next-table-id* (instruction bắt buộc): Cho biết bảng kế tiếp trong quá trinh xử lý pipeline. ID bảng tiếp theo phải lớn hơn ID bảng hiện tại.

Tập instruction liên kết với mục flow chứa tối đa một instruction cho từng loại. Các instruction của bộ thực hiện theo thứ tự được chỉ định bởi danh sách trên. Trong thực tế, các ràng buộc duy nhất là lệnh Clear-Actions được thực thi trước lệnh Write-Actions, các action-Apply được thực hiện trước khi Write-Metadata và Goto-Table được thực hiện lần cuối.

Một switch phải từ chối một mục flow nếu nó không thể thực hiện các instruction hoặc một phần của các instruction liên quan đến mục flow đó. Trong trường hợp này, switch phải trả lại thông báo lỗi liên quan đến vấn đề.

2.3.6 Bộ Action

Một bộ action được kết hợp với mỗi gói. Bộ này mặc định trống. Mục flow có thể chỉnh sửa bộ action bằng cách sử dụng lệnh Write-Action hoặc Clear-Action liên quan đến một so khớp cụ thể. Bộ action được chuyển giữa các bảng flow. Khi bộ instruction của mục flow không chứa lệnh Goto-Table, quá trình xử lý pipeline dừng lại và các action trong bộ action của gói được thực hiện.

Bộ action chứa tối đa một action của mỗi loại. Các action *set-field* được xác định bởi các trường trường đích của chúng, do đó bộ action chứa tối đa một action *set-field* cho từng loại trường (có thể đặt nhiều trường, nhưng mỗi trường chỉ có thể được đặt một lần). Khi

action của một loại cụ thể được thêm vào trong bộ action, nếu một action cùng loại tồn tại, nó sẽ bị ghi đè bởi action sau này. Nếu cần nhiều action cùng loại, ví dụ: đẩy nhiều nhãn MPLS hoặc popping nhãn MPLS, nên sử dụng lệnh Apply-Actions.

Action trong bộ action được áp dụng theo thứ tự được chỉ định bên dưới, bất kể thứ tự chúng được thêm vào tập hợp.

1. **copy TTL inwards**: áp dụng các action sao chép TTL nội bộ vào gói tin
2. **pop**: áp dụng tất cả các hoạt động của thẻ vào gói tin
3. **push-MPLS**: áp dụng thao tác đẩy thẻ MPLS đến gói tin
4. **push-PBB**: áp dụng thao tác push thẻ PBB vào gói tin
5. **push-VLAN**: áp dụng các thao tác đẩy thẻ VLAN vào gói tin
6. **copy TTL outwards**: áp dụng bản sao TTL ra bên ngoài action vào gói tin
7. **decrement TTL**: áp dụng giảm TTL đến gói tin
8. **set**: áp dụng tất cả các action thiết lập trường vào gói tin
9. **qos**: áp dụng tất cả các action QoS, chẳng hạn như *meter* và *set-queue* cho gói tin
10. **group**: nếu một action nhóm được chỉ định, áp dụng các action của (các) nhóm nhóm có liên quan theo thứ tự được chỉ định bởi danh sách này
11. **output**: nếu không có *action group*, chuyển tiếp gói tin trên cổng được xác định bởi action output

Action output trong tập action được thực hiện cuối cùng. Nếu cả action output và action group được chỉ định trong tập hợp action, action output sẽ bị bỏ qua và action group được ưu tiên. Nếu không có action output và không có action group đã được chỉ định trong một tập action, gói tin sẽ bị bỏ qua. Nếu không có action group được chỉ định và action output tham chiếu đến một cổng không tồn tại, gói tin sẽ bị bỏ qua.

Action output được xử lý khác nhau về ingress và egress. Khi bộ action ingress chứa một action output hoặc một action group chuyển tiếp gói tin tới một cổng, gói tin phải bắt đầu xử lý egress trên cổng đó. Nếu action output tham chiếu đến cổng reserved ALL, một bản sao (clone) của gói tin bắt đầu xử lý egress cho mỗi cổng có liên quan. Khi bộ hành trình egress chứa action output, gói tin rời khỏi quá trình egress và được xử lý bởi cổng, và trong hầu hết các trường hợp nó được chuyển tiếp ra khỏi switch.

2.3.7 Danh sách các action

Instruction Apply-Actions và thông báo Packet-out bao gồm một danh sách các action. Các action của một danh sách các action được thực hiện theo thứ tự được chỉ định bởi danh sách, và được áp dụng ngay cho gói.

Việc thực hiện một danh sách các action bắt đầu với action đầu tiên trong danh sách và mỗi action được thực hiện trên gói theo thứ tự. Hiệu quả của những action này là tích lũy, nếu

danh sách các action có chứa hai action Push VLAN, hai tiêu đề VLAN được thêm vào gói tin.

Nếu danh sách các action có chứa một action output, một bản sao (clone) của gói tin được chuyển tiếp trong trạng thái hiện tại đến cổng mong muốn, nơi nó bắt đầu xử lý egress. Nếu action đầu ra tham chiếu đến cổngt reserved ALL, một bản sao của gói bắt đầu xử lý egress cho mỗi cổng có liên quan. Nếu action đầu ra tham chiếu đến một cổng không tồn tại, thì bản sao của gói sẽ bị bỏ qua. Nếu danh sách các action có chứa một action group, một bản sao của gói trong trạng thái hiện tại được xử lý bởi các group có liên quan.

Sau khi thực hiện danh sách các action trong một instruction Apply-Actions, thực hiện pipeline vẫn tiếp tục trên các gói đã chỉnh sửa. Bộ action của gói không thay đổi bởi việc thực hiện danh sách các action.

2.3.8 Action

Switch không bắt buộc để hỗ trợ tất cả các loại action, chỉ trừ những loại action được đánh dấu là "action bắt buộc" bên dưới.

- **Output** *port_no* (action bắt buộc): Xuất chuyển gói tin tới một cổng OpenFlow đã chỉ định, nơi nó bắt đầu xử lý egress. Thiết bị switch OpenFlow phải hỗ trợ chuyển tiếp tới các cổng vật lý, các cổng logic được định nghĩa và các cổng reserved được yêu cầu.
- **Group** *group_id* (action bắt buộc): Xử lý gói thông qua các group được chỉ định. Giải thích chính xác phụ thuộc vào loại nhóm.
- **Drop** (action bắt buộc): Không có action đại diện rõ ràng. Thay vào đó, gói tin mà tập action không có action đầu ra và không có action nhóm phải được bỏ qua. Kết quả này có thể xuất phát từ các tập lệnh rỗng hoặc giỏ action rỗng trong quá trình xử lý pipeline hoặc sau khi thực hiện lệnh Clear-Actions.
- **Set-Queue** *queue_id* (action không bắt buộc): Action set-queue đặt id hàng đợi cho một gói tin. Khi gói được chuyển tiếp đến một cổng sử dụng action đầu ra, id hàng đợi xác định hàng đợi nào được gắn vào cổng này được sử dụng để lên kế hoạch và chuyển tiếp gói tin. Hành vi chuyển tiếp được quyết định bởi cấu hình của hàng đợi và được sử dụng để cung cấp hỗ trợ Quality-of-Service (QoS) cơ bản.
- **Meter** *meter_id* (action không bắt buộc): Chuyển trực tiếp gói tới meter được chỉ định. Theo kết quả đo, gói tin có thể bị bỏ qua (tùy thuộc vào cấu hình và trạng thái meter). Nếu switch hỗ trợ meter, nó phải hỗ trợ action này như là action đầu tiên trong một danh sách các action, để tương thích với các phiên bản trước của đặc tả.
- **Push-Tag/Pop-Tag** *ethertype* (action không bắt buộc): Các switch có thể hỗ trợ khả năng push / pop tags. Các thẻ được đẩy sau luôn phải được chèn vào dưới dạng thẻ ngoài cùng ở vị trí ngoài cùng bên phải của thẻ đó. Khi nhiều action đẩy được thêm vào bộ action của gói, chúng áp dụng cho gói theo thứ tự được xác định bởi các quy tắc

action, MPLS đầu tiên, sau đó là PBB, cuối cùng là VLAN. Khi nhiều action đẩy được bao gồm trong một danh sách các action, chúng áp dụng cho gói tin theo thứ tự liệt kê.

- **Set-Field** *field_type value* (action không bắt buộc): Các action Set-Field khác nhau được xác định theo loại trường và chỉnh sửa các giá trị của các trường tiêu đề tương ứng trong gói tin.
- **Copy-Field** *src_field_type dst_field_type* (action không bắt buộc): Tác vụ Copy-Field có thể sao chép dữ liệu giữa bất kỳ trường tiêu đề gói tin hoặc pipeline nào.
- **Change-TTL** *ttl* (action không bắt buộc): Các action Change-TTL khác nhau làm thay đổi giá trị của IPv4 TTL, IPv6 Hop Limit hoặc MPLS TTL trong gói tin.

2.3.9 Bảng group

Một group table bao gồm các mục group. Khi một mục flow trỏ đến một group cho phép OpenFlow đại diện cho các phương pháp chuyển tiếp bổ sung.

Group Identifier	Group Type	Counters	Action Buckets

Mỗi mục của group được xác định bởi group identifter của nó và chứa:

- **group identifter**: một số nguyên 32 bit không dấu duy nhất xác định group trên switch OpenFlow.
- **group type**: để xác định loại nhóm.
- **counters**: bộ đếm cập nhật khi các gói được xử lý bởi một nhóm.
- **action buckets**: danh sách các giỏ action theo thứ tự, trong đó mỗi giỏ action chứa một tập hợp các action để thực hiện và các tham số liên quan. Các action trong một cái giỏ luôn được áp dụng như một bộ action.

Mục group có thể không có hoặc có nhiều giỏ action, ngoại trừ group indirect, là loại luôn luôn có một giỏ action.

Giỏ thường chứa các action chỉnh sửa gói tin và action output chuyển tiếp đến một cổng. Giỏ cũng có thể bao gồm một action group mà gọi một nhóm khác nếu switch hỗ trợ chuỗi nhóm, trong trường hợp này gói tin tiếp tục xử lý trong group được gọi. Một nhóm mà không có action nào hợp lệ, một group mà không có đầu ra hoặc action group sẽ bỏ qua bản sao của gói tin đi kèm với group đó.

2.3.10 Bảng meter

Một bảng đo bao gồm các mục meter, định nghĩa cho mỗi meter. Mỗi luồng flow cho phép OpenFlow thực hiện việc hạn chế tốc độ, một thao tác QoS đơn giản sẽ hạn chế một tập hợp các luồng đến một băng thông đã chọn. Các meter hoàn toàn độc lập với các hàng đợi per-port, tuy nhiên trong nhiều trường hợp, hai tính năng này có thể được kết hợp để thực hiện các công việc phức tạp để bảo tồn các khuôn khổ QoS, chẳng hạn như DiffServ.

Một meter đo tốc độ các gói được gán cho nó và cho phép controller điều khiển tốc độ của các gói đó. Meter được gắn trực tiếp vào các mục flow (trái với hàng đợi được gắn vào cổng). Nếu switch hỗ trợ nó, bất kỳ mục flow nào cũng có thể chỉ định một action meter trong một danh sách các action: meter đo và kiểm soát tỷ lệ tổng của tất cả các mục Flow mà nó được gắn vào.

Các thành phần chính của một mục meter trong bảng meter:

Meter Identifier	Meter Bands	Counters

Mỗi mục meter được xác định bởi meter identifter của nó và chứa:

- **meter identifter**: một số nguyên 32 bit không dấu xác định duy nhất một meter
- **meter bands**: một danh sách các dải meter không có thứ tự, trong đó mỗi băng đo được xác định tỷ lệ băng tần và cách xử lý gói tin
- **counters**: cập nhật khi các gói tin được xử lý bởi một meter

Các flow khác nhau trong cùng một bảng Flow có thể sử dụng cùng một meter, meter khác hoặc không có meter. Bằng cách sử dụng các meter khác nhau trong một bảng flow, tập hợp các mục flow có thể được đo độc lập. Các gói tin có thể đi qua nhiều meter khi sử dụng các meter trong các bảng kế tiếp, tại mỗi bảng flow có thể nhập trực tiếp nó vào một meter.

3 Tổng quan về OpenFlow Switch Protocol

Kênh OpenFlow là kênh kết nối mỗi Switch OpenFlow với một bộ controller OpenFlow. Thông qua kênh này, bộ controller cấu hình và quản lý switch, nhận các sự kiện từ switch, gửi các gói tin ra khỏi switch. Kênh controller của switch có thể hỗ trợ một kênh OpenFlow với một bộ controller đơn, hoặc nhiều kênh OpenFlow cho phép nhiều bộ controller chia sẻ quản lý switch. Kênh OpenFlow thường được mã hóa bằng TLS, nhưng có thể chạy trực tiếp qua TCP.

Giao thức OpenFlow Switch Protocol hỗ trợ ba loại bản tin:

- *controller-to-switch*
- *asynchronous* - không đồng bộ
- *symmetric* - đối xứng

Mỗi loại có nhiều loại phụ. Bản tin Controller-to-switch được tạo bởi bộ controller và được sử dụng để trực tiếp quản lý hoặc kiểm tra trạng thái của switch. Các thông điệp không đồng bộ được khởi tạo bởi switch và được sử dụng để cập nhật cho bộ controller về các sự kiện mạng và thay đổi trạng thái switch. Các thông điệp đối xứng được tạo bởi switch hoặc bộ controller và được gửi mà không có sự báo trước.

3.1 Các loại bản tin

❖ Bản tin *controller to switch* được khởi tạo bởi bộ controller và có thể yêu cầu phản hồi từ switch:

- **Features**: Bộ controller có thể yêu cầu định danh switch và các tính năng cơ bản của nó bằng cách gửi một yêu cầu features. Switch phải trả lời bằng một trả lời về danh tính và các khả năng cơ bản của switch. Điều này thường được thực hiện khi thiết lập kênh OpenFlow.
- **Conftguration**: bộ controller có thể thiết lập và truy vấn các tham số cấu hình trong switch. Switch chỉ trả lời một truy vấn từ bộ controller.
- **Modify-State**: được gửi bởi bộ controller để quản lý trạng thái trên các switch. Mục đích chính của chúng là bổ sung, xóa và sửa đổi các mục flow hoặc group, chèn hoặc xóa giỏ action của group trong các bảng OpenFlow và để thiết lập các thuộc tính cổng switch.
- **Read-State**: bộ controller sử dụng để thu thập các thông tin khác nhau từ bộ switch, chẳng hạn như cấu hình hiện tại, thống kê và khả năng.
- **Packet-out**: được bộ controller sử dụng để gửi các gói tin ra khỏi một cổng quy định trên switch, và để chuyển tiếp các gói tin nhận được thông qua bản tin Packet-in. Thông báo gói tin phải chứa một gói đầy đủ hoặc một bộ đệm ID tham chiếu đến

một gói được lưu trữ trong switch. Bản tin cũng chứa một danh sách các action được áp dụng theo thứ tự chỉ định, một danh sách các action trống sẽ bỏ qua gói tin.

- **Barrier**: bản tin barrier request / reply được bộ controller sử dụng để đảm bảo gói tin đã được xử lý hoặc nhận thông báo cho các hoạt động đã hoàn thành.
- **Role-Request**: được bộ controller sử dụng để thiết lập vai trò của kênh OpenFlow của nó, thiết lập Controller ID, hoặc truy vấn các thông số này. Điều này hầu như hữu ích khi switch kết nối với nhiều bộ controller.
- **Asynchronous-Conftguration**: bản tin được bộ controller sử dụng để thiết lập một bộ lọc bổ sung cho các bản tin không đồng bộ mà nó muốn nhận được trên kênh OpenFlow, hoặc để truy cập bộ filter. Điều này hầu như hữu ích khi switch kết nối với nhiều bộ controller và thường được thực hiện khi thiết lập kênh OpenFlow.

❖ *Bản tin không đồng bộ*: switch gửi các bản tin không đồng bộ đến các bộ controller để biểu thị một gói tin đến hoặc sự thay đổi trạng thái. Các loại bản tin không đồng bộ chính được mô tả dưới đây:

- **Packet-in**: Chuyển việc xử lý gói tin đến bộ controller. Đối với tất cả các gói được chuyển tiếp tới cổng reserved CONROLROLER sử dụng mục flow hoặc mục table-miss.
- **Flow-Removed**: Thông báo cho bộ controller về việc loại bỏ một mục flow từ một bảng flow.
- **Port-status**: Thông báo cho bộ controller sự thay đổi trạng thái cổng. Các sự kiện này bao gồm sự thay đổi trong các sự kiện cấu hình cổng và các sự kiện thay đổi trạng thái cổng.
- **Role-status**: Thông báo cho bộ controller về sự thay đổi vai trò của nó. Khi một bộ controller mới tự chọn làm chủ, switch sẽ gửi thông báo trạng thái đến bộ controller cũ.
- **Controller-Status**: Thông báo cho bộ controller khi tình trạng của một kênh OpenFlow thay đổi. Điều này có thể hỗ trợ quá trình thiết lập kênh dự phòng nếu bộ controller mất khả năng giao tiếp với nhau.
- **Flow-monitor**: Thông báo cho bộ controller sự thay đổi trong một bảng flow.

❖ *Bản tin đối xứng* được gửi đi theo cả hai hướng:

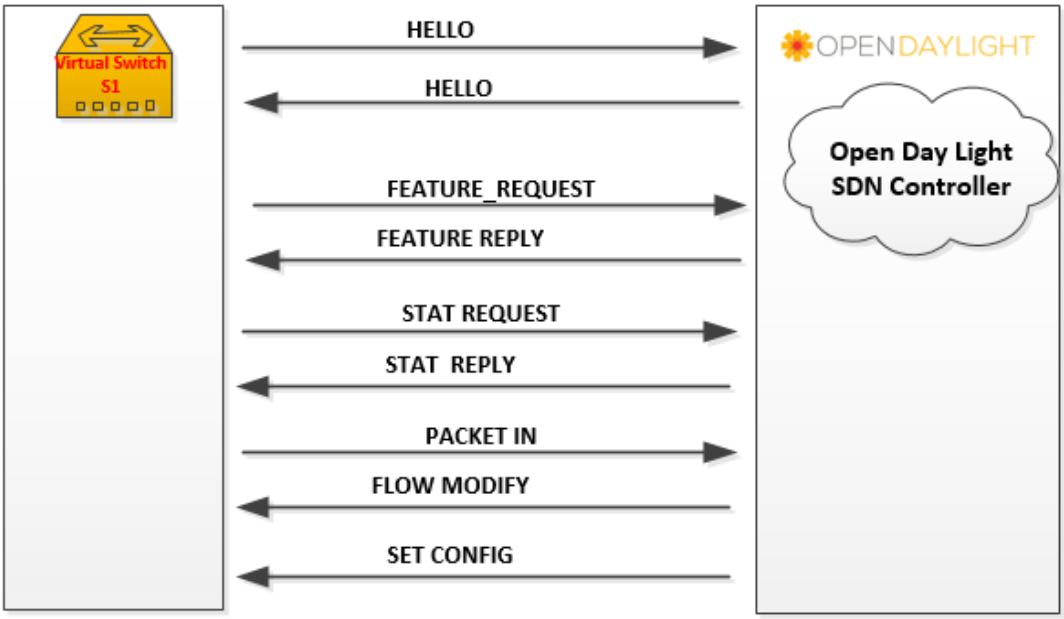
- **Hello**: các bản tin được trao đổi giữa switch và controller khi khởi động kết nối.
- **Echo**: có thể được gửi từ switch hoặc bộ controller, và phải trả về một phản hồi echo. Chúng chủ yếu được sử dụng để xác minh tính liên tục của bộ controller và switch, cũng có thể được sử dụng để đo độ trễ hoặc băng thông của nó.
- **Error**: Các thông báo lỗi được sử dụng bởi switch hoặc bộ controller để thông báo các sự cố cho phía bên kia của kết nối. Chúng chủ yếu được sử dụng bởi switch để chỉ ra sự thất bại của một yêu cầu được tạo bởi bộ controller.

- **Experimenter**: Thông điệp Experimenter cung cấp một cách chuẩn cho các switch OpenFlow cung cấp các chức năng bổ sung. Đây là một khu vực dàn cho các tính năng mới của các phiên bản OpenFlow trong tương lai.

3.2 Thiết lập kết nối

Switch phải có khả năng giao tiếp với một bộ controller qua kết nối URI do người dùng định cấu hình, sử dụng cổng người dùng chỉ định hoặc cổng OpenFlow mặc định 6653. Nếu switch sử dụng URI để kết nối tới bộ controller, switch sẽ khởi tạo một kết nối TLS hoặc TCP tiêu chuẩn tới bộ controller theo các trường của kết nối URI.

Khi thiết lập một kết nối OpenFlow lần đầu tiên, mỗi bên của kết nối phải gửi một thông điệp OFPT_HELLO với trường phiên bản OpenFlow cao nhất mà nó hỗ trợ. Thông báo Hello này có thể bao gồm một số thành phần OpenFlow để giúp thiết lập kết nối. Khi nhận được thông báo này, người nhận phải tính toán phiên bản giao thức chuyển mạch OpenFlow sẽ được sử dụng. Nếu cả hai bản tin Hello đã gửi và bản tin Hello đã nhận chứa một thành phần hello OFPHET_VERSIONBITMAP và nếu các bitmap đó có một số bit thông thường, phiên bản thương lượng phải là phiên bản cao nhất được đặt trong cả hai bitmap. Nếu không, phiên bản thương lượng phải là số phiên bản nhỏ hơn đã được gửi và số phiên bản đã nhận được trong các trường phiên bản.



Nếu phiên bản thương lượng được hỗ trợ bởi người nhận, thì kết nối sẽ tiến hành. Nếu không, người nhận phải trả lời thông báo lỗi Hello Failed, và sau đó kết thúc kết nối.

Sau khi switch và bộ controller trao đổi bản tin OFPT_HELLO và đàm phán thành công một phiên bản chung, thiết lập kết nối được thực hiện và thông điệp OpenFlow tiêu chuẩn có thể được trao đổi qua kết nối. Một trong những điều đầu tiên mà bộ controller nên làm là gửi một thông báo OFPT_FEATURES_REQUEST để nhận được Datapath ID của switch.

3.3 Bảo trì kết nối

Bảo trì kết nối OpenFlow hầu hết được thực hiện bởi các cơ chế kết nối TLS hoặc TCP, điều này đảm bảo rằng các kết nối OpenFlow được hỗ trợ rộng rãi. Các cơ chế chính để phát hiện sự gián đoạn kết nối và chấm dứt kết nối là TCP timeouts và TLS session timeouts (khi có).

Mỗi kết nối phải được duy trì riêng biệt, nếu kết nối với một bộ controller hay bộ switch cụ thể bị chấm dứt hoặc bị hỏng, điều này sẽ không làm cho các kết nối với các bộ controller hoặc switch khác bị chấm dứt. Các kết nối phụ phải được chấm dứt khi kết nối chính tương ứng được chấm dứt hoặc bị hỏng, mặt khác nếu một kết nối phụ bị ngắt hoặc hỏng sẽ không làm ảnh hưởng đến kết nối chính hoặc các kết nối phụ.

Khi kết nối chấm dứt do điều kiện mạng hoặc thời gian chờ, nếu một switch hoặc bộ controller là người khởi tạo kết nối, nó sẽ cố gắng kết nối lại với bên kia cho đến khi kết nối mới được thiết lập hoặc cho đến khi Connection URI của bên kia được gỡ bỏ khỏi cấu hình của nó.

Bộ controller hoặc bộ switch có thể chấm dứt kết nối khi nhận được thông báo lỗi OpenFlow, ví dụ nếu không có phiên bản OpenFlow chung hoặc nếu bên thứ ba không hỗ trợ một tính năng quan trọng.

3.4 Ngắt kết nối

Trong trường hợp một switch bị mất liên lạc với bộ controller, nó phải gửi thông báo *controller status* đến tất cả các bộ controller kết nối còn lại (nếu có) cho biết vai trò và tình trạng kênh điều khiển của bộ controller bị ảnh hưởng. Nếu bộ switch mất liên lạc với nhiều bộ controller, nó phải gửi nhiều thông báo *controller status*, một cho mỗi bộ controller bị ảnh hưởng. Điều này cho phép các bộ controller còn lại thực hiện hành động ngay cả khi chúng mất liên lạc với nhau. Khi kênh OpenFlow được thiết lập lại, switch phải gửi thông báo cập nhật *controller status* đến tất cả bộ controller.

Khi kênh OpenFlow được thiết lập, mục flow hiện tại trong các bảng flow tại thời điểm đó được bảo toàn và hoạt động của OpenFlow trở lại bình thường. Nếu muốn, bộ controller có tùy chọn đọc tất cả các dòng chảy với một yêu cầu thống kê flow, để đồng bộ lại trạng thái với trạng thái switch. Ngoài ra, bộ controller sau đó có tùy chọn xóa tất cả các mục dòng chảy với một yêu cầu mod-flow, để bắt đầu từ một trạng thái sạch trên switch.

4. Cài đặt thử nghiệm

4.1 Công cụ sử dụng

❖ *Mininet Emulator*

Mininet là một công cụ giả lập mạng, bao gồm tập hợp các hosts đầu cuối, các switches, routers và các liên kết trên một Linux kernel. Mininet sử dụng công nghệ ảo hóa (ở mức đơn giản) để tạo nên hệ thống mạng hoàn chỉnh, chạy chung trên cùng một kernel, hệ thống và user code.

Các host ảo, switch, liên kết và các controller trên mininet là các thực thể thực sự, được giả lập dưới dạng phần mềm thay vì phần cứng. Một host mininet có thể thực hiện ssh vào đó, chạy bất kì phần mềm nào đã cài trên hệ thống linux (môi trường mà mininet đang chạy). Các phần mềm này có thể gửi gói tin thông các ethernet interface của mininet với tốc độ liên kết và trễ đặt trước.

Mininet cho phép tạo topo mạng nhanh chóng, tùy chỉnh được topo mạng, chạy được các phần mềm thực sự như web servers, TCP monitoring, Wireshark; tùy chỉnh được việc chuyển tiếp gói tin. Mininet cũng dễ dàng sử dụng và không yêu cầu cấu hình đặc biệt gì về phần cứng để chạy: mininet có thể cài trên laptop, server, VM, cloud (linux).

❖ *Controller OpenDayLight*

OpenDaylight là phần mềm mã nguồn mở dành cho Software Defined Networking (SDN) sử dụng giao thức mở cung cấp khả năng kiểm soát tập trung, có khả năng lập trình được và theo dõi các thiết bị mạng. Giống như nhiều SDN Controllers khác, OpenDaylight hỗ trợ OpenFlow, cũng như cung cấp các giải pháp mạng khác sẵn sàng để cài đặt khi có yêu cầu. OpenDaylight cung cấp giao diện cho phép kết nối các thiết bị mạng nhanh chóng và thông minh để tối ưu hiệu năng mạng.

OpenDaylight Controller cung cấp northbound APIs, được sử dụng bởi các ứng dụng. Các ứng dụng này sử dụng controller để thu thập thông tin về mạng, chạy các thuật toán để kiểm soát, phân tích, sau đó sử dụng OpenDaylight Controller tạo các rules mới cho mạng. OpenDaylight Controller viết bằng ngôn ngữ Java, có nghĩa là có thể sử dụng OpenDaylight Controller trên bất kì môi trường nào hỗ trợ Java. Tuy nhiên để đạt hiệu năng tốt nhất, OpenDaylight nên chạy trên môi trường Linux hỗ trợ JVM tối thiểu 1.7.

❖ *Postman*

Postman là 1 ứng dụng REST Client, dùng để thực hiện test, gửi các request, API. Postman cũng có thể dùng để test các request tới server, xem kết quả trả về từ server mà không cần sử dụng browser hoặc app.

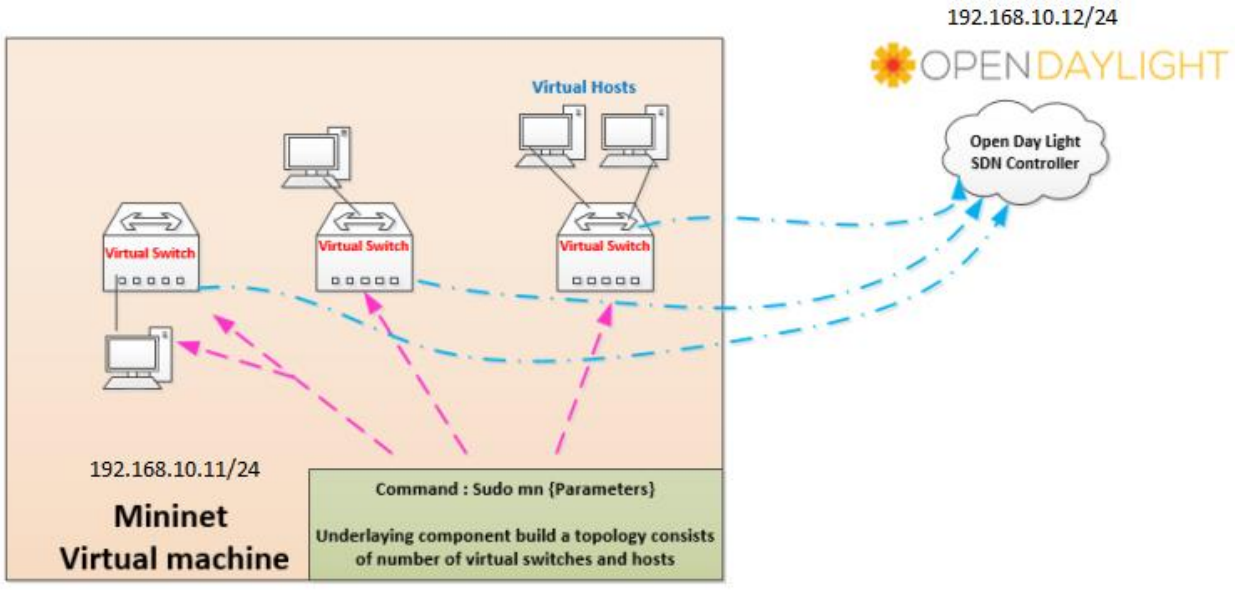
- Cho phép gửi HTTP Request với các method GET, POST, PUT, DELETE.
- Cho phép post dữ liệu dưới dạng form (key-value), text, json
- Hiện kết quả trả về dạng text, hình ảnh, XML, JSON
- Hỗ trợ authorization (Oauth1, 2)
- Cho phép thay đổi header của các request

4.2 Kết quả thử nghiệm

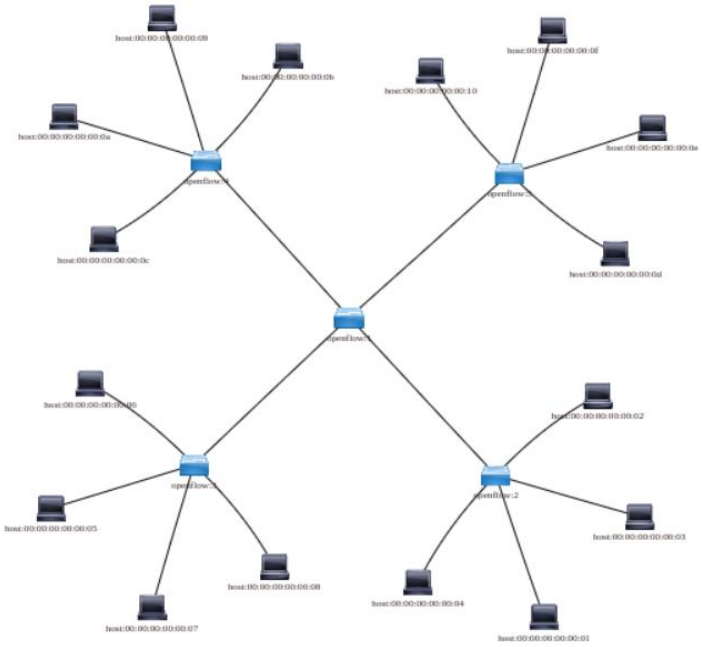
- Controller OpendayLight phiên bản 6.0.1 (cacbon) cài trên máy Ubuntu 16.04 (x64) địa chỉ ip: 192.168.10.12/24

```
tuan@tuan-virtual-machine: ~/distribution-karaf-0.6.1-Carbon
tuan@tuan-virtual-machine:~$ cd distribution-karaf-0.6.1-Carbon/
tuan@tuan-virtual-machine:~/distribution-karaf-0.6.1-Carbon$ ./bin/karaf
Apache Karaf starting up. Press Enter to open the shell now...
100% [========================================================================]

Karaf started in 40s. Bundle stats: 334 active, 334 total

    ________                                ________                .__  .__       .__     __
    \_____  \ ______  ____   ____ \_____  \ ______ ___.__.|  | |__| ____ |  |___/  |_
     /   |   \\____ \_/ __ \ /    \ |    |  \\__  \<   |  ||  | |  |/ ___\|  |  \   __\
    /    |    \  |_> >  ___/|   |  \|    `   \/ __ \\___  ||  |_|  / /_/  >   Y  \  |
    \_______  /   __/ \___  >___|  /_______  (____  / ____||____/__\___  /|___|  /__|
            \/|__|        \/     \/        \/     \/\/                /_____/      \/


Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

opendaylight-user@root>
```

- Mininet Emulator địa chỉ ip: 192.168.10.11/24



❖ Topo mạng hình cây tạo trên mininet gồm 5 switch và 16 host



❖ Kết quả ping giữa các host:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12 h13 h14 h15 h16
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12 h13 h14 h15 h16
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12 h13 h14 h15 h16
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12 h13 h14 h15 h16
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12 h13 h14 h15 h16
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h13 h14 h15 h16
h13 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h14 h15 h16
h14 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h15 h16
h15 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h16
h16 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
*** Results: 0% dropped (240/240 received)
mininet>
```

Tài liệu tham khảo:

[1] Open Networking Foundation. OpenFlow Switch Specification (version 1.5.1). 3-2015

[2] Reza Toghraee. Learning OpenDayLight. 5-2017

[3] Website: https://www.opennetworking.org/software-defined-standards/overview/

[4] Website: http://mininet.org/walkthrough/

[4] Website: https://vietstack.wordpress.com/2015/03/25/tong-quan-ve-sdn-va-nfv/

[3] Website: http://www.brianlinkletter.com/using-the-opendaylight-sdn-controller-with-the-mininet-network-emulator/