RESTful API trong Ryu Controller

Tóm tắt

Báo cáo này trình bày một phương pháp triển khai RESTful API trong Ryu Controller, nhằm tăng tính bảo mật, linh hoạt và mở rộng của mạng được định nghĩa bằng phần mềm (SDN). RESTful API cung cấp cho các quản trị viên một cách tiện lợi để quản lý các Open vSwitch bằng cách sử dụng các phương thức HTTP như GET, POST, PUT, DELETE để thao tác với các luật chuyển tiếp và chính sách cũng như cấu hình lại mạng theo nhu cầu của họ. Để triển khai RESTful API, báo cáo này đề xuất sử dụng REST Linkage, một bộ phận quan trọng trong Ryu Controller, và môi trường Mininet để cấu hình lại các ứng dụng được cài đặt sẵn trong Ryu. Kết quả thử nghiệm cho thấy phương pháp đề xuất có tính dễ sử dụng, tính tương thích cao, tính linh hoạt và tiết kiệm thời gian cho các ứng dụng quản lý và điều khiển mạng SDN. Bên cạnh đó, việc triển khai RESTful API cũng giúp tăng cường tính bảo mật của mạng SDN bằng cách giảm thiểu sự can thiệp của bên thứ ba vào hệ thống.

Từ khóa: RESTful API, Ryu Controller, HTTP, REST Linkage, Mininet.

Mục Lục

Danh sách các hình ảnh và bảng biểu

1. Giới thiệu

1.1. Mục tiêu nghiên cứu

Mục tiêu nghiên cứu triển khai RESTful API trong Ryu Controller:

1. Đề xuất phương pháp sử dụng ứng dụng REST Linkage trong Ryu Controller giúp quản lý và điều khiển mạng SDN.
2. So sánh phương pháp được đề xuất với các ứng dụng hiện có để quản lý và điều khiển mạng SDN.
3. Đánh giá hiệu quả của phương pháp trong ứng dụng thực tiễn.

Nhìn chung, các mục tiêu nghiên cứu nhằm cung cấp một giải pháp thiết thực và hiệu quả để cải thiện tính linh hoạt, độ tương thích và tính bảo mật của mạng SDN qua việc quản lý các chính sách và luật chuyển tiếp.

1.2. Phương pháp nghiên cứu

Phương pháp chính để sử dụng RESTful API trong Ryu Controller trong nghiên cứu của chúng tôi là sử dụng giao thức HTTP gửi yêu cầu đến URL của tài nguyên switch. Các bước sau đây đã được thực hiện:

1. Cấu hình và tạo yêu cầu RESTful: RESTful API được cung cấp bởi một số ứng dụng trong Ryu Controller, chẳng hạn như REST API Switch và REST API Topology. Cấu hình các ứng dụng này để sử dụng RESTful API. Sau đó tạo yêu cầu RESTful API bằng cách sử dụng các phương thức HTTP như GET, POST, PUT và DELETE để thao tác với các tài nguyên mạng như Switch, Port, Link, Topology, Host, Flow entry, Meter entry, Group entry, Table entry và Queue entry. Các yêu cầu này được gửi đến địa chỉ URL của tài nguyên mạng mà người dùng muốn truy cập.

2. Xử lý yêu cầu RESTful: Ryu Controller xử lý các yêu cầu RESTful API và trả về kết quả tương ứng. Kết quả này có thể là dữ liệu JSON hoặc XML, tùy thuộc vào cách người dùng yêu cầu.
3. Phân tích kết quả: Phân tích kết quả trả về từ RESTful API để hiểu cấu trúc và thông tin của tài nguyên mạng đang truy cập. Kết quả này có thể được sử dụng để tùy chỉnh và cấu hình lại các tài nguyên mạng theo nhu cầu của quản trị viên.

1.3. Đóng góp của nghiên cứu

Nghiên cứu triển khai RESTful API trong Ryu Controller đóng góp cho lĩnh vực quản lý mạng SDN theo nhiều cách. Thứ nhất, đây là nghiên cứu đầu tiên triển khai RESTful API để quản lý các tài nguyên mạng như Switch, Port, Link, Topology, Host, Flow entry, Meter entry, Group entry, Table entry và Queue entry trong một mô hình mạng SDN. Thứ hai, nó giải quyết một số thách thức trong việc quản lý mạng SDN, ví dụ như tính linh hoạt, khả năng mở rộng và tính đáng tin cậy. Nghiên cứu cung cấp một giải pháp cụ thể và hiệu quả để triển khai RESTful API trong Ryu Controller. Tiếp theo, nghiên cứu cung cấp các khuyến nghị để triển khai thực tế giải pháp này trong các mạng SDN trong thế giới thực. Nhìn chung, nghiên cứu góp phần nâng cao kiến thức và cung cấp một giải pháp thiết thực và hiệu quả để quản lý các tài nguyên mạng trong mạng SDN bằng RESTful API.

2. Chi tiết nghiên cứu

2.1. Các nghiên cứu liên quan

Trong những năm gần đây, sự phát triển của công nghệ mạng đã tạo ra nhiều cơ hội và thách thức cho các nhà nghiên cứu và kỹ sư trong việc thiết kế và triển khai các hệ thống mạng mới. Trong số đó, SDN (Software Defined Networking) dường như đang trở thành một kỹ thuật quan trọng để giải quyết các vấn đề liên quan đến quản lý và điều khiển mạng.

RESTful API là một phương pháp được sử dụng để thiết kế các ứng dụng web và dịch vụ web. RESTful API cung cấp một cách tiếp cận đơn giản và hiệu quả để truyền tải dữ

liệu giữa các ứng dụng khác nhau. Trong mạng SDN, RESTful API được sử dụng để cung cấp một giao diện đơn giản và linh hoạt cho việc quản lý và điều khiển mạng.

Các nghiên cứu trước đây đã chứng minh rằng việc sử dụng RESTful API trong mạng SDN có thể giúp cải thiện hiệu suất và tăng tính linh hoạt của hệ thống. Nhiều ứng dụng và dịch vụ đã được xây dựng trên nền tảng RESTful API để quản lý và điều khiển các mạng SDN. Một số nghiên cứu cụ thể đã tập trung vào việc sử dụng RESTful API trong mạng SDN sử dụng Ryu Controller.

Bài báo [2] tạo ra các REST API cho phép người dùng tùy ý chọn đường dẫn lưu lượng mà dữ liệu đến và đi qua các switch phải tuân theo. Và họ cũng tạo ra một GUI (Graphical User Interface) Web giúp đảm bảo những tương tác thích hợp với Controller.

Bài báo [3] đề xuất việc ứng dụng REST API trong Cloud Computing (điện toán đám mây) bằng cách kết hợp với mạng SDN. Cụ thể đề xuất một khung quản lý ứng dụng an toàn dựa trên kiểm soát truy cập API REST, có tên là SEAPP, giúp quản lý chi tiết các quyền của ứng dụng và mã hóa các lệnh gọi API REST để bảo vệ chống lại các cuộc tấn công độc hại, tránh việc các ứng dụng độc hại lạm dụng API để khởi chạy các cuộc tấn công thù địch, gây ra các mối đe dọa nghiêm trọng cho mạng. Đồng thời SEAP cũng là một kiến trúc logic nhẹ giữa mặt phẳng ứng dụng và mặt phẳng điều khiển, hỗ trợ triển khai và cấu hình lại nhanh chóng trong thời gian chạy.

Bài báo [4] sử dụng Mininet để đánh giá chất lượng dịch vụ (Quality of Service) trong một mạng SDN có Ryu-controller thông qua băng thông (bandwidth), thông lượng (throughput) sử dụng iperf.

Ngoài ra, REST API cũng được ứng dụng trong việc bảo mật đối với mạng SDN. Bài báo [5] giới thiệu một framework gọi là RECHECKER có thể tìm thấy các lỗ hổng bảo mật của các dịch vụ RESTful trong bộ điều khiển SDN. Họ đã tìm thấy bốn loại lỗi đối với ba bộ điều khiển nguồn mở: ONOS, FloodLight, và Ryu. Các lỗi này được tìm ra bằng cách thực hiện những yêu cầu HTTP không đúng định dạng và chờ thông báo phản hồi, sau đó

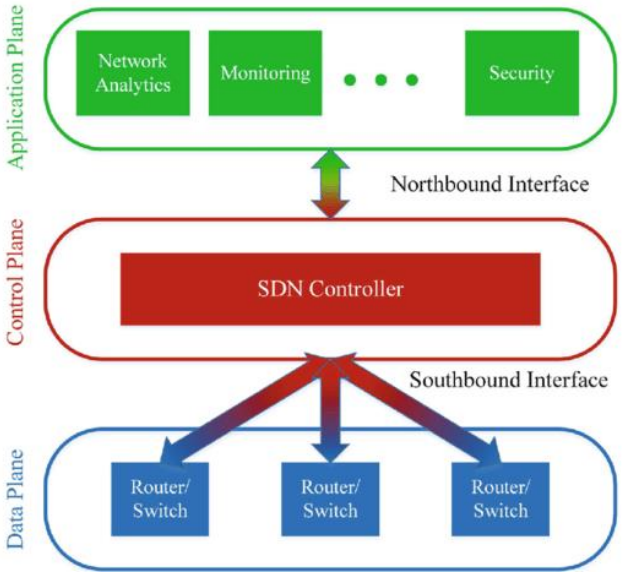
RECHECKER sẽ kiểm tra và so sánh các flow rule được cài đặt với các flow rule yêu cầu để tìm ra sự khác biệt.

2.2. Tổng quan SDN

2.1.1. SDN

SDN (Software Defined Networking) là một phương pháp đổi mới trong lĩnh vực mạng máy tính, mang lại tính linh hoạt, tự động hóa, khả năng lập trình và khả năng quản lý tập trung cho hạ tầng mạng. Trong một mạng truyền thống, các thiết bị mạng như bộ chuyển mạch và bộ định tuyến sử dụng các giao thức phức tạp để giao tiếp với nhau, gây khó khăn cho việc quản lý và mở rộng mạng. Với SDN, mặt phẳng điều khiển (control plane) được tách ra khỏi mặt phẳng dữ liệu (data plane), cho phép điều khiển và quản lý mạng tập trung. Điều này sẽ cho phép các quản trị viên mạng dễ dàng hơn trong việc định cấu hình và quản lý mạng của họ thông qua phần mềm.

Trong kiến trúc SDN, mạng thường bao gồm ba thành phần chính: mặt phẳng dữ liệu (data plane), mặt phẳng điều khiển (control plane) và lớp ứng dụng (application layer). Mặt phẳng dữ liệu bao gồm các thiết bị mạng như bộ chuyển mạch và bộ định tuyến chuyển tiếp lưu lượng. Mặt phẳng điều khiển được quản lý bởi bộ điều khiển tập trung, chịu trách nhiệm đưa ra quyết định về cách chuyển tiếp lưu lượng dựa trên chính sách và điều kiện mạng. Lớp ứng dụng bao gồm các ứng dụng phần mềm sử dụng tính chất có thể lập trình của SDN để xác định và thực thi các chính sách mạng.



Hình 1. Kiến trúc SDN

SDN có nhiều lợi ích cho các doanh nghiệp và tổ chức, bao gồm tăng tính linh hoạt và khả năng mở rộng của mạng, cải thiện an ninh mạng và giảm chi phí quản lý mạng, etc. Một trong những ưu điểm chính của SDN là khả năng cung cấp cho quản trị viên mạng cái nhìn tổng thể về toàn bộ mạng, cho phép quản lý và kiểm soát tập trung. Điều này cho phép ta dễ dàng định cấu hình và tối ưu hóa luồng lưu lượng mạng, thực hiện các chính sách bảo mật và phản ứng nhanh chóng với các yêu cầu mạng đang thay đổi. SDN cũng cho phép tự động hóa mạng, giúp giảm tính chất phức tạp và tốn thời gian của các tác vụ quản lý mạng truyền thống.

Tuy nhiên, việc triển khai SDN cũng đưa ra những thách thức, chẳng hạn như nhu cầu về kỹ năng của các lập trình viên mạng, khả năng tương tác giữa các giải pháp SDN khác nhau, sự đảm bảo an ninh và quyền riêng tư. Tuy nhiên, SDN có thể tiếp tục đạt được mục tiêu lớn hơn khi cộng đồng, các nhà cung cấp phát triển cách xây dựng các mạng linh hoạt hơn, có thể mở rộng và lập trình để theo kịp nhu cầu phát triển của mạng hiện đại.

Tóm lại, SDN là một cách tiếp cận mang tính biến đổi đối với mạng cung cấp khả năng kiểm soát tập trung, khả năng lập trình và tự động hóa cho các mạng. Nó có khả năng cách mạng hóa mạng được thiết kế, quản lý và vận hành, mang lại nhiều lợi ích cho các tổ

chức về tính linh hoạt, khả năng mở rộng và hiệu quả. Khi SDN tiếp tục phát triển và trưởng thành, nó được kỳ vọng sẽ đóng một vai trò quan trọng trong việc định hình tương lai của mạng.

2.1.2. OpenFlow

OpenFlow là một giao thức nguồn mở cho phép giao tiếp giữa mặt phẳng chuyển tiếp của bộ chuyển mạch mạng và bộ điều khiển quản lý hành vi tổng thể của mạng. Giao thức được Open Networking Foundation (ONF) phát triển để tạo điều kiện cho Software-Defined Networking (SDN), cho phép quản trị viên mạng kiểm soát và quản lý mạng của họ thông qua các ứng dụng phần mềm.

OpenFlow hoạt động bằng cách tách mặt phẳng điều khiển khỏi mặt phẳng dữ liệu trong các thiết bị mạng. Mặt phẳng điều khiển, chịu trách nhiệm đưa ra các quyết định về cách định tuyến dữ liệu, được chuyển đến bộ điều khiển tập trung giao tiếp với các bộ chuyển mạch và bộ định tuyến trong mạng thông qua giao thức OpenFlow. Mặt phẳng dữ liệu, chịu trách nhiệm chuyển tiếp các gói, vẫn nằm trên chính các thiết bị mạng. Bằng cách tập trung kiểm soát theo cách này, OpenFlow cho phép quản trị viên mạng có cái nhìn toàn diện hơn về mạng của họ và quản lý mạng hiệu quả hơn.

Một ưu điểm khác của OpenFlow là tính linh hoạt của nó. Giao thức được thiết kế để mở và có thể mở rộng, cho phép phát triển các ứng dụng và tính năng mới. Điều này giúp dễ dàng tùy chỉnh hành vi mạng dựa trên các nhu cầu cụ thể và cho phép tạo ra các kiến trúc mạng phức tạp và sáng tạo hơn.

Và một trong những điểm mạnh không thể thiếu nữa của OpenFlow Protocol là khả năng ảo hóa mạng. Với khả năng này, các quản trị viên mạng có thể quản lý một mạng tách biệt phần cứng thông qua phần mềm điều khiển. Khi sử dụng ảo hóa mạng, các quản trị viên mạng có thể dễ dàng tạo ra các mạng ảo (virtual network) với các tài nguyên mạng được chia sẻ, tạo ra một môi trường mạng linh hoạt hơn và tiết kiệm chi phí.

Ngoài ra, ảo hoá mạng cũng cho phép các quản trị viên mạng dễ dàng triển khai các chính sách mạng (network policies) để đáp ứng các yêu cầu đặc biệt của người dùng và các ứng dụng. Các chính sách mạng có thể được thiết lập tại mức độ mạng ảo, giúp tăng cường tính linh hoạt và hiệu quả của mạng.

Nhìn chung, OpenFlow đã được chứng minh là một công cụ mạnh mẽ dành cho quản trị viên mạng, cho phép họ quản lý mạng của mình hiệu quả, linh hoạt và an toàn hơn. Sự phát triển và áp dụng liên tục của nó có thể sẽ đóng một vai trò quan trọng trong tương lai của mạng.

Với sự hỗ trợ của OpenFlow, mạng SDN có thể cung cấp tính linh hoạt cao hơn, giảm thiểu thiết bị mạng cần thiết và giúp quản lý tài nguyên mạng hiệu quả hơn. Phiên bản đầu tiên của giao thức OpenFlow là phiên bản OpenFlow 1.0, ra đời vào tháng 12 năm 2009, sau đó lần lượt ra đời các phiên bản 1.2, 1.3, 1.4, etc. Ở trong bài báo cáo này, chúng tôi sử dụng phiên bản 1.3

2.1.3. Mininet

Mininet là một công cụ mô phỏng mạng mã nguồn mở được sử dụng để thử nghiệm và phát triển các ứng dụng SDN (Software-Defined Networking). Mininet cho phép người dùng tạo ra một mạng ảo với các switch, host, controller và liên kết mạng ảo.

Trong mininet, các host chạy phần mềm mạng Linux tiêu chuẩn, còn các thiết bị chuyển mạch (switch) hỗ trợ OpenFlow để cho phép định tuyến tùy chỉnh linh hoạt và tạo mạng SDN. Mininet được sử dụng rất rộng rãi cho các mục đích nghiên cứu, phát triển, tạo mẫu và thử nghiệm.

Một số tính năng của Mininet bao gồm:

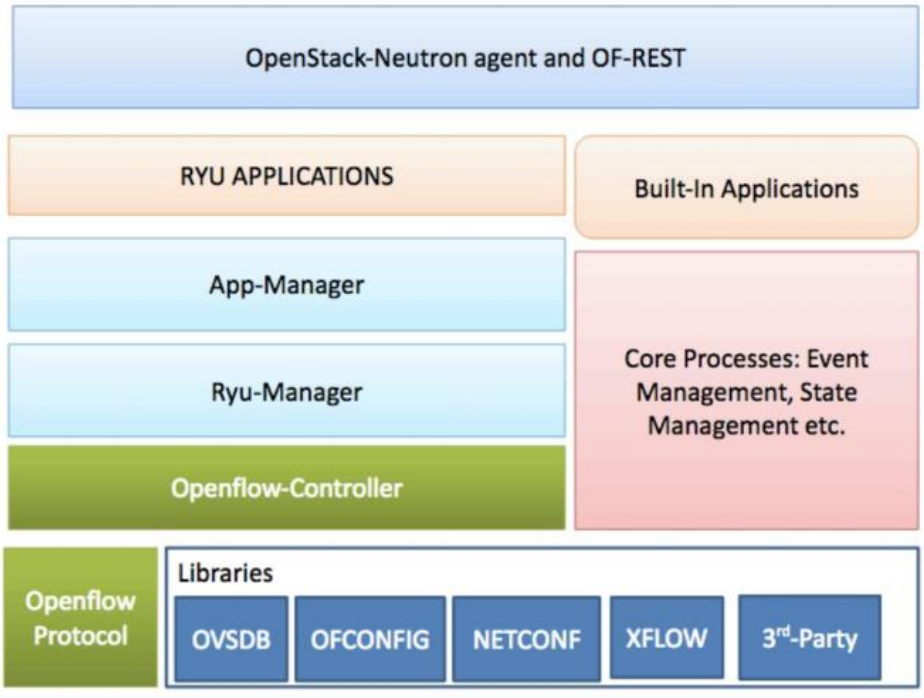
- Hỗ trợ nhiều giao thức mạng như Ethernet, TCP/IP, UDP và ICMP.
- Cho phép người dùng chạy các ứng dụng điều khiển mạng như Ryu, POX, Floodlight và OpenDaylight.

- Hỗ trợ tạo và chạy các topo mạng phức tạp với nhiều switch và host.
- Có thể thực hiện các thử nghiệm mạng như kiểm tra hiệu suất, thực hiện các tấn công mạng và phát triển các ứng dụng SDN.
- Mininet được viết bằng Python và hoạt động trên các nền tảng Linux.

2.1.4. Ryu Controller

Ryu là một controller mã nguồn mở được sử dụng trong mạng SDN, được hỗ trợ bởi NTT và được viết hoàn toàn bằng python. Ryu hỗ trợ nhiều giao thức như OpenFlow, NETCONF, OF-config. Trong bài báo này, chúng tôi sử dụng Ryu Controller thay vì những bộ điều khiển khác bởi Ryu controller là một OpenFlow Controller mã nguồn mở, có thể được tùy chỉnh và triển khai dễ dàng trên nhiều nền tảng khác nhau. Việc triển khai Ryu controller cũng đơn giản hơn so với một số OpenFlow Controller khác. Ryu controller được viết bằng Python và hỗ trợ đa nền tảng, cho phép chạy trên nhiều hệ điều hành khác nhau.

Dưới đây chúng tôi trình bày sơ qua về kiến trúc của Ryu:



Hình 2. Kiến trúc Ryu

- Ryu libraries: Đối với các southbound protocols, Ryu hỗ trợ OF-Config, Open vSwitch Database Management Protocol (OVSDB), NETCONF và XFlow và vẫn còn những thứ khác từ bên thứ ba. Ví dụ: XFlow chủ yếu được sử dụng để thực hiện các loại phép đo khác nhau liên quan đến lưu lượng đi qua mạng. Ngoài ra, Ryu có một thư viện gói, mà chúng ta có thể sử dụng để phân tích cú pháp và xây dựng các gói của giao thức mong muốn (VLAN, MPLS và GRE)
- OpenFlow Protocol: giúp giải mã và mã hóa các giao thức
- Openflow controller: là nguồn chính của các sự kiện nội bộ trong kiến trúc Ryu
- Ryu-manager: là tệp thực thi chính, tạo ra quy trình lắng nghe các địa chỉ IP đã chỉ định và trên cổng được chỉ định và cho phép kết nối các switch
- App-manager: là nơi các ứng dụng điều khiển mạng được phát triển bằng Python. Ryu App cung cấp các API và thư viện cho phép lập trình viên tương tác với các thành phần khác của Ryu controller, bao gồm cả OpenFlow switch và các giao thức điều khiển mạng khác
- Ryu-application: là một module của Ryu Controller, mà các lập trình viên có thể viết mã để tương tác với các thành phần khác trong Ryu Controller, bao gồm OpenFlow switch, giao thức điều khiển mạng khác và REST API.Các ứng dụng điều khiển mạng được phát triển bằng Ryu Application có thể cung cấp các chức năng như định tuyến lưu lượng mạng (network routing), chuyển tiếp gói tin (packet forwarding), phân phối tải (load balancing), theo dõi mạng (network monitoring) và bảo mật mạng (network security)

Ryu controller cung cấp một REST API để cho phép các ứng dụng trên SDN giao tiếp với các thành phần khác trong mạng thông qua các yêu cầu REST. REST API của Ryu controller là một giao diện đơn giản và dễ sử dụng để các ứng dụng trên SDN có thể truy cập vào các chức năng của mạng. Ngoài ra, Ryu còn có một cộng đồng lớn của các nhà phát triển và người dùng trên toàn thế giới, vì vậy nó được hỗ trợ và phát triển liên tục.

2.1.5. RESTful API

REST (Representational State Transfer) là một kiểu kiến trúc phần mềm được sử dụng trong mạng SDN (Software-Defined Networking) để quản lý các thiết bị mạng. RESTful API là một phương thức cung cấp khả năng giao tiếp giữa các ứng dụng và các thiết bị mạng thông qua HTTP (Hypertext Transfer Protocol).

Trong mạng SDN, RESTful API được sử dụng để tạo liên kết (linkage) giữa các thiết bị mạng. RESTful API cho phép các thiết bị mạng được quản lý từ xa và tạo các liên kết giữa chúng thông qua các yêu cầu HTTP.

Các liên kết này cho phép các thiết bị mạng trao đổi thông tin và thực hiện các chức năng như cấu hình mạng, quản lý lưu lượng mạng và phân tích dữ liệu mạng. RESTful API đơn giản và dễ sử dụng, giúp các nhà quản trị mạng dễ dàng tìm hiểu và triển khai các ứng dụng SDN phức tạp.

2.3. REST Linkage

REST Linkage ở trong Ryu là một bộ phận quan trọng của Ryu Controller, cung cấp các API RESTful để quản lý, cấu hình và giám sát mạng SDN. REST Linkage bao gồm các file chức năng như:

- File “rest_conf_switch.py”: cung cấp API để cấu hình switch mạng SDN, bao gồm các cấu hình luật chuyển tiếp, định tuyến và các thông số khác của switch.
- File “rest_qos.py”: cung cấp API để quản lý chất lượng dịch vụ (QoS) trên switch mạng SDN, bao gồm cấu hình băng thông, độ trễ và độ ưu tiên của các luồng mạng trên switch.
- File "rest_topology.py": cung cấp API để truy vấn và giám sát thông tin về topoogy mạng SDN, bao gồm các thiết bị mạng, liên kết và các đường đi giữa các thiết bị.
- File "wsgi.py": cung cấp các chức năng WSGI (Web Server Gateway Interface) để giao tiếp giữa API RESTful và các ứng dụng web.

- File "utils.py": cung cấp các hàm tiện ích để xử lý các yêu cầu HTTP và truy vấn dữ liệu từ cơ sở dữ liệu của Ryu.
- File “ofctl_rest” là một ứng dụng của REST Linkage, cung cấp các API để thực hiện các hoạt động kiểm soát trên bảng dẫn đường (flow table) của switch, bao gồm thêm, sửa và xóa các luật chuyển tiếp. Ứng dụng này cho phép người dùng tương tác với switch thông qua các yêu cầu HTTP.
- File “"simple_switch_rest" là một ứng dụng của REST Linkage, cung cấp các API để cấu hình switch và thực hiện các hoạt động kiểm soát trên switch, bao gồm thêm, sửa và xóa các luật chuyển tiếp. Ứng dụng này cung cấp một cách dễ dàng để kiểm tra và khám phá tính năng của switch mạng SDN bằng cách sử dụng các yêu cầu HTTP.

Ở phần tiếp theo, chúng tôi sẽ triển khai hai ứng dụng chính của REST Linkage là “ofctl_rest” và “simple_switch_rest”. Cả "ofctl_rest" và "simple_switch_rest" đều sử dụng WSGI để cung cấp các API RESTful và tương tác với các ứng dụng web khác. Chúng đều là các thành phần quan trọng của REST Linkage và giúp tăng tính linh hoạt và khả năng tương tác của mạng SDN được triển khai bằng Ryu. Tuy nhiên điểm khác nhau của chúng nằm ở trong chức năng. "simple_switch_rest" được thiết kế như một ứng dụng đơn giản và dễ sử dụng để kiểm tra và thử nghiệm Ryu. "ofctl_rest" là một ứng dụng nâng cao hơn "simple_switch_rest" về khả năng kiểm soát luồng gói tin. Ngoài việc thêm, sửa, xóa các mục đỉa chỉ MAC trong bảng chuyển tiếp của switch như trong “simple_switch_rest”, “ofctl_rest” cho phép làm điều đó với các flow entry giúp người dùng điều khiển luồng gói tin đi qua mạng bằng cách chỉ định các mục quyết định luồng cần được cài đặt trên switch. "ofctl_rest" là một ứng dụng nâng cao hơn "simple_switch_rest" về khả năng kiểm soát luồng gói tin.

2.3.1. ofctl_rest

Trong Ryu, ofctl_rest là một ứng dụng web REST API được tích hợp sẵn, cung cấp các chức năng điều khiển và giám sát tổng thể cho các thiết bị mạng SDN. Nó cho phép các ứng dụng có khả năng truy cập vào mạng thông qua HTTP REST. Cụ thể hơn, "ofctl_rest"

là một công cụ dòng lệnh được cung cấp bởi thư viện "ryu" trong OpenFlow. Công cụ này cho phép người dùng truy cập và quản lý các switch mạng OpenFlow thông qua các API RESTful. Điều này có nghĩa là người dùng có thể sử dụng các yêu cầu HTTP như GET, POST, PUT và DELETE để tương tác với switch mạng, thay vì sử dụng các lệnh dòng lệnh truyền thống như "ovs-ofctl" hay "dpctl". Các API RESTful của "ofctl_rest" cho phép người dùng có thể thực hiện các hoạt động như hiển thị trạng thái của switch, thêm hoặc xóa các luật chuyển tiếp, hoặc điều chỉnh các cấu hình switch mạng. Việc sử dụng "ofctl_rest" có thể giúp giảm độ phức tạp của việc quản lý switch mạng OpenFlow, đồng thời tăng tính linh hoạt và khả năng mở rộng của hệ thống mạng.

“ofctcl_rest” cung cấp một số API để hiện thị danh sách các switch được kết nối đến controller. Các API được cung cấp bao gồm:

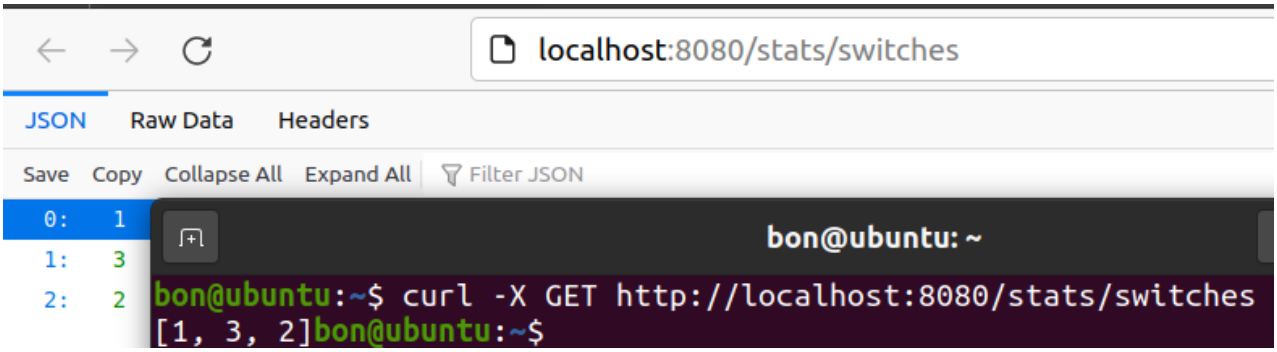
1. /stats/switches: Trả về danh sách các switch được kết nối đến controller.
2. /stats/flow/{switch_id}: Truy xuất các flow entry từ switch có ID là switch_id.
3. /stats/port/{switch_id}: Truy xuất thông tin về các port của switch có ID là switch_id.
4. /stats/aggregate/{switch_id}: Truy xuất thông tin tổng hợp về traffic flow của switch có ID là switch_id.
5. /stats/desc/{switch_id}: Truy xuất thông tin mô tả cho switch có ID là switch_id.

Các API này giúp người dùng kiểm tra và thử nghiệm các chức năng của REST API trong Ryu. Ngoài ra, ofctl_rest cũng cung cấp một số lệnh hiệu chỉnh trong quá trình chạy, cho phép người dùng tùy chỉnh các thông số để phù hợp với nhu cầu sử dụng của mình. Các lệnh chức năng này bao gồm:

1. Lấy thông tin về switch OpenFlow bao gồm các flow entries, group entries, meter entries, port statistics, switch statistics và các thông tin khác.
2. Thêm hoặc xóa flow entries, group entries hoặc meter entries từ switch OpenFlow.
3. Lấy thông tin về các switch OpenFlow khác nhau trong hệ thống.

4. Thực hiện các thao tác trên switch OpenFlow, bao gồm kích hoạt hoặc ngưng hoạt động các port trên switch.

Ví dụ, để truy xuất thông tin của tất cả switch kết nối đến controller, phương pháp dùng đến là URL và GET /start/switches. Tin nhắn trả về sẽ cho Datapath ID của các switch. Ở đây chúng tôi tạo một topo trong môi trường mininet với 3 switch kết nối trực tiếp với nhau và thực thi lệnh curl đến API. Kết quả nhận được sẽ trả về dưới dạng JSON



Hình 3. Truy xuất thông tin của các switch trong ofctl_rest

Đối với những lệnh cập nhật thông tin của switch phức tạp hơn, cần thêm nội dung yêu cầu vào lệnh curl. Ví dụ để thêm flow entry từ port 1 của host 1 đến port 2 của host 2 vào switch, câu lệnh sử dụng là:

$ curl -X POST -d '{"dpid": 1, "cookie": 1, "cookie_mask": 1, "table_id": 0, "idle_timeout": 30, "hard_timeout": 30, "priority": 11111, "flags": 1, "match": {"in_port":1}, "actions": [{"type":"OUTPUT", "port": 2}]}' http://localhost:8080/stats/flowentry/add

Người dùng có thể sử dụng "ofctl_rest" để cấu hình các luật bảo mật trên switch mạng OpenFlow để bảo vệ mạng SDN. Các luật bảo mật được cấu hình trên switch mạng OpenFlow có thể giúp ngăn chặn các cuộc tấn công mạng bằng cách kiểm soát lưu lượng mạng. Ví dụ, người dùng có thể sử dụng "ofctl_rest" để tạo các luật chuyển tiếp để chặn các gói tin độc hại hoặc từ chối dịch vụ (DoS) bằng cách giới hạn lưu lượng mạng đến các đích hợp lệ. Giới hạn số lượng yêu cầu được gửi đến trong một khoảng thời gian nhất định

bằng cách cấu hình lại file ofctl_rest.conf. Ở đây để giới hạn số lượng yêu cầu được gửi đến trong một khoảng thời gian 5 giây, có thể sử dụng các giá trị sau:

[REQUEST_LIMIT]

enable = True

limit_count = 10

limit_sec = 5

Trong đó, enable = True cho phép tính năng giới hạn yêu cầu, limit_count = 10 giới hạn số lượng yêu cầu tối đa là 10 yêu cầu, và limit_sec = 5 giới hạn thời gian là 5 giây. Nếu người dùng gửi quá nhiều yêu cầu trong khoảng thời gian này, ofctl_rest sẽ từ chối yêu cầu và trả về mã lỗi "429 Too Many Requests".

“ofctl_rest” còn dùng để bảo vệ mạng SDN khỏi các cuộc tấn công bằng cách thực hiện các thao tác bảo mật như xác thực người dùng và quản lý quyền truy cập. Ví dụ như quản lý quyền truy cập: Các luật điều khiển có thể được xác định bằng cách sử dụng các trường trong các gói tin mạng, chẳng hạn như địa chỉ MAC, địa chỉ IP, số cổng, hoặc thậm chí là các trường tùy chỉnh khác. Đầu tiên cần xác định luật bảo mật rồi sử dụng công cụ HTTP để gửi các yêu cầu HTTP đến “ofctl_rest” để tạo hoặc chỉnh sửa các luật bảo mật. Sử dụng các yêu cầu HTTP POST hoặc PUT để tạo yêu cầu chỉnh sửa các luật bảo mật. Ngoài việc sử dụng công cụ HTTP, các luật điều khiển cũng có thể được định nghĩa bằng cách sử dụng các ngôn ngữ lập trình như Python hoặc Java, hoặc sử dụng các công cụ cấu hình đồ họa như Ryu SDN Framework hoặc Floodlight Controller. Các công cụ này cung cấp các API cho phép tạo, sửa đổi và xóa các luật điều khiển trên switch. Để tạo một luật điều khiển đơn giản chỉ cho phép các địa chỉ MAC cụ thể truy cập vào switch có thể sử dụng mã Python như sau:

```
from ryu.ofproto import ofproto_v1_0
from ryu.ofproto import ofproto_v1_0_parser

class MyController(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_0.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(MyController, self).__init__(*args, **kwargs)

    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def packet_in_handler(self, ev):
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser

        # Define the MAC address you want to allow
        mac_to_allow = '00:11:22:33:44:55'

        # Define the actions to take if the MAC address is allowed
        actions = [parser.OFPActionOutput(ofproto.OFPP_FLOOD)]

        # Define the match criteria for the flow entry
        match = parser.OFPMatch(dl_dst=mac_to_allow)

        # Create the flow entry to allow the MAC address
        inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]
        mod = parser.OFPFlowMod(datapath=datapath, priority=1, match=match, instructions=inst)
        datapath.send_msg(mod)
```

Hình 4. Mã cho phép địa chỉ MAC cụ thể truy cập vào switch

Ở đây chúng tôi cho phép địa chỉ MAC “00:11:22:33:44:55” được phép truy cập, hành động được cho phép là đẩy gói tin đến tất cả các cổng trên switch và chỉ định rằng chỉ có các gói tin có địa chỉ trên mới được cho phép

Ngoài ra, ofctl_rest còn có thể cấu hình để xác thực người dùng, giám sát lưu lượng, tài nguyên và lỗi để giảm thiểu các rủi ro bảo mật và các cuộc tấn công mạng trên switch cũng như các sự cố mạng bằng các cách như trên. Nhìn chung, ofctl_rest là một công cụ quan trọng trong SDN Ryu, cung cấp một cách tiếp cận đơn giản và linh hoạt để điều khiển mạng SDN

2.3.2. simple_switch_rest_13

"simple_switch_rest_13" là một trong những ứng dụng mẫu được cung cấp bởi Ryu để giúp người dùng nhanh chóng bắt đầu với các dự án SDN và thử nghiệm các tính năng của Ryu. Nó cung cấp một switch đơn giản được triển khai trên giao thức OpenFlow 1.3 và

hỗ trợ RESTful API để cấu hình và điều khiển switch. Switch được triển khai trong "simple_switch_rest_13" hỗ trợ các tính năng chính của OpenFlow 1.3 như chuyển tiếp gói tin dựa trên các luật chuyển tiếp, giao tiếp với controller thông qua các message type như PacketIn, PacketOut, FlowMod, và các message type khác. Ứng dụng cũng cung cấp các API RESTful cho phép người dùng thêm, sửa và xóa các mục quyết định luồng (flow entry) và các mục địa chỉ MAC trong bảng chuyển tiếp của switch.

Trước tiên, xây dựng một topo trên mininet với 1 switch và 3 host bằng cầu lệnh:

$ sudo mn --topo single,3 --mac --switch ovsk,protocols=OpenFlow13 --controller remote -x

Sau đó chạy ứng dụng trong file /ryu/app. Trong thông báo tại thời điểm bắt đầu, có một dòng cho biết “wsgi starting up on http://0.0.0.0:8080/” và điều này cho biết rằng máy chủ Web đã khởi động tại cổng số 8080. Khi phát lệnh ping từ h1 sang h2 có thể thấy Packet-In đến Ryu xuất hiện 3 lần. Vì khi packet được gửi từ h1 sang h2, Khi packet được gửi từ host a sang host b, 1 tin nhắn packet in được gửi và địa chỉ MAC h1 sẽ được ghi nhớ vào port 1. Vì port kết nối với h2 chưa được tìm hiểu, nên packets chuyển sang sẽ thực hiện flood, packets sẽ được nhận bởi cả h2 và h3. Khi packets được gửi về từ h2 cho h1, khi đó địa chỉ MAC của h2 đã được tìm hiểu và thêm vào bảng địa chỉ MAC và địa chỉ MAC của h1 đã được tìm hiểu từ trước, nên packets sẽ được chuyển thẳng vào port 1.

Thực thi API REST để lấy bảng MAC của switching hub. Lần này, sử dụng lệnh curl để gọi REST API. Kết quả trả về có thể thấy rằng host 1 và host 2 đã được học trong bảng MAC khi gọi lệnh: curl -X GET http://127.0.0.1:8080/simpleswitch/mactable/0000000000000001

Hình 5. Dữ liệu đã được học trong bảng MAC

"simple_switch_rest_13" không phải là một giải pháp hoàn chỉnh để giải quyết các vấn đề mạng, mà là một ứng dụng minh họa để hiểu cách sử dụng chức năng tham chiếu hoặc cập nhật bảng địa chỉ MAC làm tài liệu để giải thích cách thêm API REST.

2.4. Các ứng dụng

Khác với hai ứng dụng trên cho phép quản trị viên tương tác với switch OpenFlow trên mạng SDN, các ứng dụng dưới đây sẽ đi cụ thể hơn vào từng phần trong quản lý mạng.

2.4.1. Firewall

Bằng cách cung cấp một API cho phép các ứng dụng khác truy cập và quản lý các tài nguyên trên mạng, có thể sử dụng RESTful API để quản lý các chính sách bảo mật trên firewall, bao gồm cấu hình quy tắc chặn, cho phép hoặc giám sát lưu lượng mạng, các chính sách liên quan đến quản lý người dùng và các chức năng bảo mật khác.

Cụ thể hơn, việc sử dụng RESTful API để quản lý các chính sách bảo mật trên firewall bao gồm:

- Cấu hình các quy tắc chặn: Cấu hình các quy tắc chặn trên firewall, bao gồm thiết lập các điều kiện để xác định lưu lượng mạng được chặn và cấu hình các hành động được thực hiện trên các gói tin bị chặn.

- Quản lý lưu lượng mạng: Giám sát và kiểm soát lưu lượng mạng trên firewall, bao gồm theo dõi các kết nối mạng và các gói tin được chuyển tiếp qua firewall, và cấu hình các chính sách liên quan đến quản lý lưu lượng mạng.
- Quản lý người dùng: Quản lý người dùng trên firewall, bao gồm cấu hình các chính sách liên quan đến quản lý quyền truy cập và xác thực người dùng.
- Các chức năng bảo mật khác: Quản lý các chức năng bảo mật khác trên firewall, bao gồm cấu hình các chính sách liên quan đến quản lý bảo mật và các cơ chế phòng chống tấn công mạng.

Đồng thời triển khai một topo mạng trong môi trường mininet và khởi động ứng dụng "rest_firewall", khi kết nối thành công Ryu Controller với router sẽ hiện lên thông báo "[FW] [INFO] dpip=000000000001: Join as firewall." Có nghĩa là tường lửa đã được đặt ở trạng thái vô hiệu hóa cắt dứt mọi liên lạc. Vì các quy tắc về quyền thiết lập không được thiết lập nên kết nối sẽ bị chặn và các gói bị chặn sẽ được xuất ra nhật ký.

Hình 6. Nhật ký của rest_firewall

Để thêm một quy tắc cho phép kết nối giữa hai host, cần thêm quy tắc cho cả hai host. Sử dụng POST với các thông tin là địa chỉ source, địa chỉ destination và giao thức:

curl -X POST -d '{"nw_src": "10.0.0.1/32", "nw_dst": "10.0.0.2/32", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001

curl -X POST -d '{"nw_src": "10.0.0.2/32", "nw_dst": "10.0.0.1/32", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001

Các quy tắc đã thêm sẽ được đăng ký trong switch dưới dạng các flow entry.

```
bon@ubuntu:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows s1
[sudo] password for bon:
 cookie=0x0, duration=597.525s, table=0, n_packets=2, n_bytes=84, priority=65534,arp actions=NORMAL
 cookie=0x4, duration=122.792s, table=0, n_packets=0, n_bytes=0, priority=1,icmp,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=NORMAL
 cookie=0x5, duration=115.603s, table=0, n_packets=0, n_bytes=0, priority=1,icmp,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=NORMAL
 cookie=0x0, duration=597.525s, table=0, n_packets=8, n_bytes=588, priority=0 actions=CONTROLLER:128
```

Hình 7. Các quy tắc được học dưới dạng các flow entry

Ở trên chúng tôi đã cho phép host 1 và host 2 giao tiếp với nhau qua giao thức ICMP. Ngoài ra, chúng tôi tiếp tục thêm một quy tắc và cho phép tất cả các gói Ipv4, bao gồm cả ping giữa host 2 và host 3. Priotiry cũng có thể được đặt thành quy tắc. Thêm quy tắc chặn ping (ICMP) giữa h3 và h2. Đặt giá trị lớn hơn 1, là giá trị ưu tiên mặc định. Cuối cùng, xác định các quy tắc đã được thiết lập:

curl http://localhost:8080/firewall/rules/0000000000000001


```
0:
    switch_id:              "0000000000000001"
  access_control_list:
    0:
      rules:
        0:
            rule_id:        1
            priority:       1
            dl_type:        "IPv4"
            nw_src:         "10.0.0.1"
            nw_dst:         "10.0.0.2"
            nw_proto:       "ICMP"
            actions:        "ALLOW"
        1:
            rule_id:        2
            priority:       1
            dl_type:        "IPv4"
            nw_src:         "10.0.0.2"
            nw_dst:         "10.0.0.1"
            nw_proto:       "ICMP"
            actions:        "ALLOW"
        2:
            rule_id:        5
            priority:       10
            dl_type:        "IPv4"
            nw_src:         "10.0.0.2"
            nw_dst:         "10.0.0.3"
            nw_proto:       "ICMP"
            actions:        "DENY"
        3:
            rule_id:        6
            priority:       10
            dl_type:        "IPv4"
            nw_src:         "10.0.0.3"
            nw_dst:         "10.0.0.2"
            nw_proto:       "ICMP"
            actions:        "DENY"
        4:
            rule_id:        3
            priority:       1
            dl_type:        "IPv4"
            nw_src:         "10.0.0.2"
            nw_dst:         "10.0.0.3"
            actions:        "ALLOW"
        5:
            rule_id:        4
            priority:       1
            dl_type:        "IPv4"
            nw_src:         "10.0.0.3"
            nw_dst:         "10.0.0.2"
            actions:        "ALLOW"
```


Hình 8. Các quy tắc đã được thiết lập trong URL của firewall

Kể từ đây, tin nhắn ping từ host 1 đến host 2 sẽ được thực hiện. Tuy nhiên, các gói khác ping sẽ bị từ chối và xuất ra nhật ký. Còn đối với giao tiếp giữa host 2 và host 3, các gói không phải ping có thể giao tiếp được với nhau.

Ngoài việc thêm các quy tắc cho phép các host giao tiếp với nhau, cũng có thể sử dụng "rest_firewall" để xóa quy tắc và bặt, tắt việc nhận trạng thái đầu ra nhật ký của các switch.

2.4.2. Router

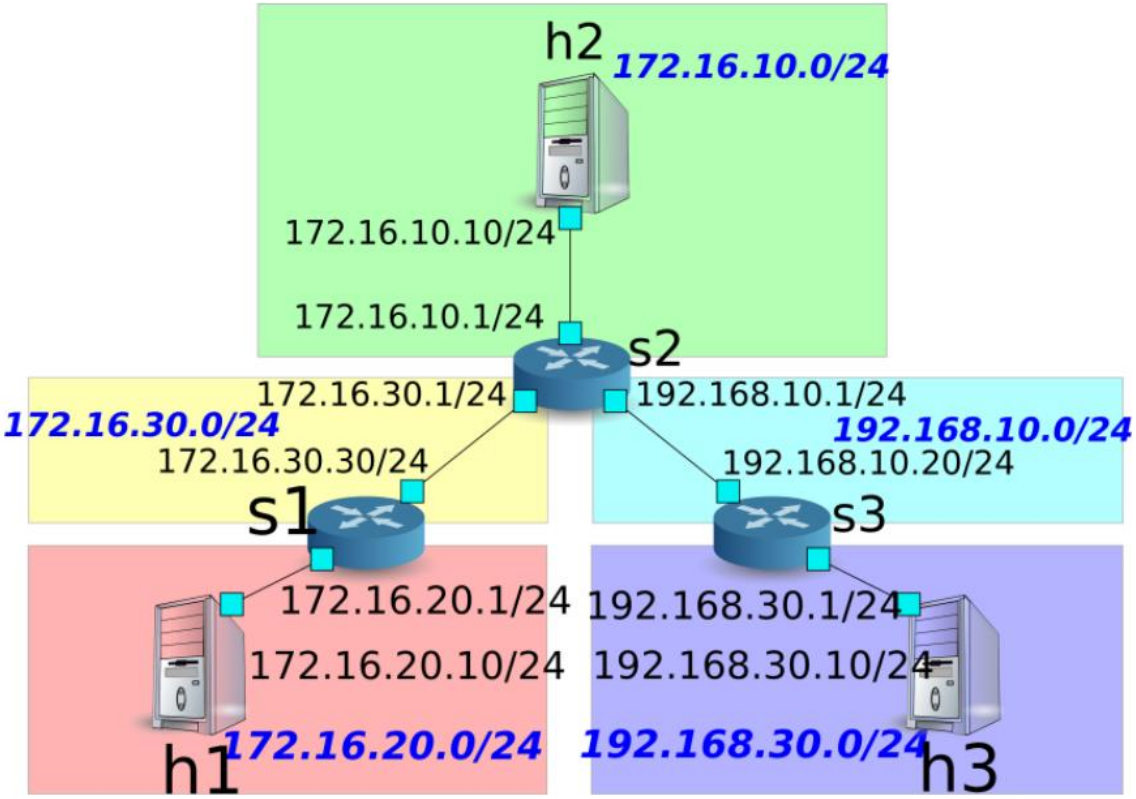
RESTful API có thể được sử dụng để quản lý các chính sách định tuyến trên router, bao gồm cấu hình các quy tắc định tuyến, theo dõi và giám sát lưu lượng mạng, quản lý các kết nối mạng và các chức năng liên quan đến quản lý mạng. Việc sử dụng RESTful API để quản lý các chính sách định tuyến trên REST Router trong Ryu có thể được thực hiện bằng các bước sau:

- Xác định các API endpoint: REST Router cung cấp một số API endpoint để quản lý các chính sách định tuyến trên router. Để sử dụng RESTful API trên REST Router, cần xác định các API endpoint tương ứng với các chức năng quản lý định tuyến, ví dụ: "/router/{router_id}/route", "/router/{router_id}/gateway", "/router/{router_id}/interface", ...
- Gửi yêu cầu HTTP: Sau khi xác định các API endpoint, có thể gửi các yêu cầu HTTP tương ứng để truy cập và quản lý các tài nguyên trên router. Các yêu cầu HTTP bao gồm GET, POST, PUT và DELETE, tùy thuộc vào chức năng quản lý định tuyến mà bạn muốn thực hiện.
- Xử lý phản hồi HTTP: Sau khi gửi yêu cầu HTTP, REST Router sẽ trả về một phản hồi HTTP chứa thông tin về tài nguyên được truy cập hoặc kết quả của các chức năng quản lý định tuyến đã được thực hiện. Xử lý phản hồi HTTP để hiển thị thông tin hoặc lưu trữ dữ liệu để sử dụng cho các mục đích khác.

Ví dụ, để lấy danh sách các định tuyến trên một router cụ thể, gửi một yêu cầu GET đến API endpoint "/router/{router_id}/route" và REST Router sẽ trả về danh sách các định

tuyến hiện có trên router. Tương tự, để cấu hình một định tuyến mới, gửi một yêu cầu POST đến API endpoint "/router/{router_id}/route" với các thông tin cấu hình định tuyến được gửi kèm theo trong body của yêu cầu HTTP.

Trong trường hợp single-tenant. triển khai một topo với ba switch tuyến tính, mỗi switch có một host trong môi trường mininet và đồng thời chạy ứng dụng “rest_router”. Xóa địa chỉ IP được gán tự động trên mỗi host và đặt một địa chỉ mới. Gán các địa chỉ mới cho host 1 là 172.16.20.10/24, host 2 là 172.16.10.10/24 và host 3 là 192.168.30.10/24. Tiếp tục gán các địa chỉ mới cho switch 1 là 172.16.30.30/24 và 172.16.20.1/24, switch 2 là 172.16.30.1/24, 172.16.10.1/24 và 192.168.10.1/24 và switch 3 là 192.168.10.20/24 và 192.168.30.1/24. Sau khi gán các địa chỉ, topo có được sẽ như hình:



Hình 9. Cấu trúc mạng sau khi cấu hình lại router

Với cấu hình định tuyến mới, có thể tiếp tục điều chỉnh các luật theo nhu cầu của quản trị viên. Trong trường hợp multi-tenant, cấu hình tương tự cho các VLAN sẽ phức tạp hơn.

2.4.3. QoS

Trong mạng SDN, RestAPI được sử dụng để cấu hình và quản lý các chính sách QoS cho các dịch vụ và ứng dụng trên mạng. RestAPI cho phép các ứng dụng và dịch vụ trên mạng giao tiếp với bộ điều khiển SDN để yêu cầu cấu hình các chính sách QoS cho các dịch vụ và ứng dụng khác nhau, giúp đảm bảo chất lượng dịch vụ và hiệu suất của mạng. Đồng thời, RestAPI cũng cho phép các ứng dụng và dịch vụ trên mạng truy cập và theo dõi thông tin về tình trạng mạng và các chính sách QoS hiện tại. Điều này cung cấp cho các ứng dụng và dịch vụ trên mạng một cách để tùy chỉnh các yêu cầu và thực hiện các điều chỉnh cho các chính sách QoS.

Ví dụ, một ứng dụng video trên mạng có thể yêu cầu băng thông cao hơn để đảm bảo chất lượng video tốt hơn. Sử dụng RestAPI, ứng dụng video có thể gửi yêu cầu tới bộ điều khiển SDN để cấu hình các chính sách QoS cho dịch vụ video của mình. Bộ điều khiển SDN sẽ sử dụng thông tin này để điều khiển các thiết bị mạng để cấu hình băng thông và độ ưu tiên cho dịch vụ video. Dưới đây, chúng tôi sẽ thực hiện cấu hình băng thông của các host trong mạng SDN.

Trong triển khai mô hình per-flow QoS, chúng tôi tạo 1 topo mạng SDN bằng mininet với 1 controller, 1 switch và 2 host (địa chỉ Mac của các host được cài đặt tùy ý). Sau đó chúng tôi khởi chạy ứng dụng rest_qos, qos_simple_switch_13 và rest_conf_switch, khi router và Ryu controller kết nối thành công, sẽ hiện lên thông báo “[QoS][INFO] dpid=0000000000000001: Join qos switch”, điều này có nghĩa hệ thống đã sẵn sàng để người dùng cấu hình các chính sách QoS cho mạng.

Bằng câu lệnh curl “-X PUT -d '"tcp:127.0.0.1:6632"'http://localhost:8080/v1.0/conf/switches/0000000000000001/ovsdb_addr”, chúng tôi đã đặt ra 1 Queue có quy tắc như sau vào switch:

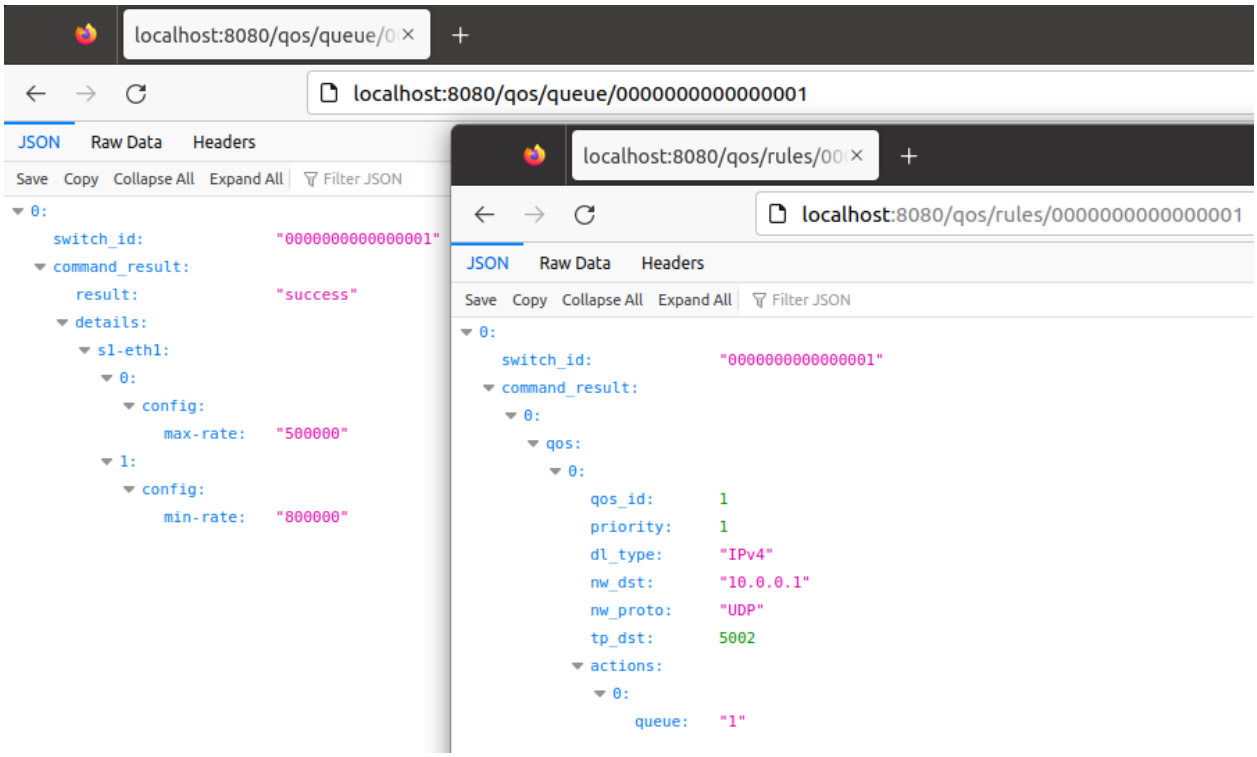
- QueueID 0
    - Max_rate: 500Kbps
- QueueID 1

- Max_rate: 1Mbps
- Min_rate: 800kbps

Tiếp tục, chúng tôi cài Flow entry sau vào switch bằng câu lệnh “curl -X POST -d '{"match": {"nw_dst": "10.0.0.1", "nw_proto": "UDP", "tp_dst": "5002"}, "actions": {"queue": "1"}}' http://localhost:8080/qos/rules/0000000000000001”

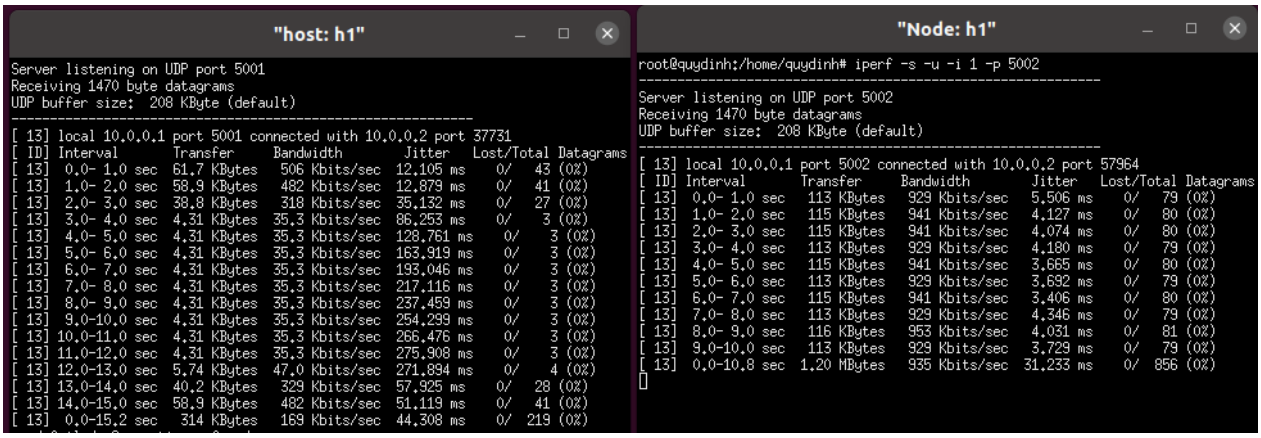
- Priority: 1
- Destination address: 10.0.0.1
- Destination port: 5002
- Protocol: UDP
- Queue ID: 1
- QoS ID: 1

Lưu ý rằng quy tắc này được cài đặt với cổng 5002. Kiểm tra các quy tắc vừa được cài đặt:



Hình 10. URL trả về các quy tắc đã học của QoS per-flow

Sau đó chúng tôi tiến hành ping từ host h2 đến host h1, các gói tin được gửi đi là UDP, có cùng băng thông 1Mbps tới 2 cổng 5001 và 5002 của host 1, kết quả là:



Hình 11. UDP traffic khi tạo kết nối giữa hai host được cấu hình trong QoS per-flow

Có thể thấy tại host h1, dù cả 2 cổng đều nhận được UDP traffic với băng thông 1Mbps, nhưng tại cổng 5001, băng thông của các gói tin đều ở dưới 500Kbps, còn tại host h2, băng thông được giới hạn trong khoảng từ 800Kbps đến 1Mbps. Như vậy, các quy tắc mà chúng tôi cài đặt trong switch đã được áp dụng thành công.

Ở trên là triển khai QoS trên mỗi luồng (per-flow QoS), tuy nhiên, khi các luồng tăng lên, các flow entries được đặt cho mỗi switch kết nối với controller cũng tăng lên, vì vậy per-flow QoS không còn phù hợp. Để khắc phục vấn đề trên, DiffServ là một phương pháp để quản lý chất lượng dịch vụ (Quality of Service, hay QoS) trong mạng. Mô hình DiffServ cho phép các gói tin được đánh dấu với một số lượng dịch vụ khác nhau (như cấp độ ưu tiên, độ trễ, độ rộng băng thông) để đảm bảo rằng chúng được xử lý và vận chuyển một cách hiệu quả hơn. Trong mô hình DiffServ, các gói tin được đánh dấu với mã DiffServ (DS) trong tiêu đề IP. Mỗi mã DS đại diện cho một lớp dịch vụ khác nhau trong mạng, ví dụ như dịch vụ ưu tiên cao (Premium), dịch vụ ưu tiên trung bình (Assured Forwarding) hoặc dịch vụ mặc định (Default). Các Router và thiết bị chuyển mạch trong mạng có thể sử dụng các mã DS để quyết định cách xử lý các gói tin. Ví dụ, các gói tin được đánh dấu với

mã DS ưu tiên cao có thể được xử lý trước các gói tin đánh dấu với mã DS ưu tiên thấp hơn. Điều này giúp tăng cường chất lượng dịch vụ và giảm độ trễ trong mạng.

Ngoài ra, Một phương pháp triển khai RESTful API trong Ryu Controller là Meter Table cũng được giới thiệu trong OpenFlow 1.3, cho phép sử dụng việc kiểm soát lưu lượng trong cơ chế OpenFlow.

2.4.4. Các ứng dụng khác

Ngoài các ứng dụng như firewall, router và QoS, REST Linkage trong Ryu Controller còn có thể sử dụng để triển khai nhiều ứng dụng khác trong mạng SDN như Load Balancing. Bằng cách cấu hình các switch trong mạng SDN để chuyển tiếp các yêu cầu từ người dùng đến các máy chủ khác nhau theo một phương pháp như Round-robin, Least Connections hoặc Source IP Hash để tăng hiệu suất và khả năng chịu tải của các dịch vụ của mạng. Round-robin là một phương pháp load balancing phổ biến, trong đó các yêu cầu từ người dùng được phân phối đều qua các máy chủ khác nhau. Bằng cách sử dụng REST Linkage, có thể triển khai Round-robin cấu hình các switch trong mạng SDN để chuyển tiếp các yêu cầu từ người dùng đến các máy chủ khác nhau theo một thứ tự nhất định. Hoặc sử dụng Least Connections, trong đó yêu cầu được chuyển tiếp đến máy chủ với số kết nối hiện tại ít nhất. Triển khai Least Connections bằng cách cấu hình các switch trong mạng SDN để chuyển tiếp các yêu cầu từ người dùng đến các máy chủ có số kết nối hiện tại ít nhất. Source IP Hash là một phương pháp load balancing khác, trong đó yêu cầu từ một địa chỉ IP nhất định được chuyển tiếp đến một máy chủ cụ thể. Có thể triển khai Source IP Hash qua cấu hình các switch trong mạng SDN để chuyển tiếp các yêu cầu từ người dùng đến các máy chủ dựa trên địa chỉ IP nguồn của yêu cầu.

Virtual Private Network (VPN) cũng sử dụng REST Linkage để triển khai VPN trong mạng SDN. Bằng cách cấu hình các switch trong mạng SDN để chuyển tiếp các gói tin của VPN đến đúng đích, giúp quản trị viên có thể tạo ra một mạng riêng ảo và bảo mật cho các dịch vụ. Với VPN, các thiết bị có thể kết nối với nhau thông qua một kênh riêng tư, giúp bảo vệ thông tin và dữ liệu trước các mối đe dọa bên ngoài. Để triển khai VPN trong mạng

SDN, có thể sử dụng REST Linkage để cấu hình các switch trong mạng SDN để xử lý các gói tin của VPN. Qua việc cấu hình các luồng chuyển tiếp trên switch giúp chỉ định các đích đến cho các gói tin VPN của bạn. Điều này có nghĩa là các gói tin VPN sẽ được chuyển tiếp đến đích đến của chúng, thay vì bị chuyển đến các thiết bị khác trên mạng. Ngoài ra, REST Linkage cũng cung cấp các API endpoint để quản lý và cấu hình các thiết bị VPN như các máy chủ VPN và các kết nối VPN. Bằng cách sử dụng các API endpoint như "/vpn/server" hoặc "/vpn/connection", quản trị viên có thể tạo ra các thiết bị VPN và kết nối chúng để hình thành một mạng riêng ảo an toàn.

Triển khai tự động hóa trong mạng SDN sử dụng RESTful API cũng đã được triển khai nhiều dự án. Nó cung cấp khả năng tạo ra các luồng chuyển tiếp và nhóm chuyển tiếp tự động trên switch để tối ưu hóa mạng, qua đó giảm thiểu thời gian và chi phí của quá trình quản lý, đồng thời tăng cường hiệu suất và tính sẵn sàng của mạng. Cụ thể, để triển khai tự động hóa mạng trong mạng SDN, cần sử dụng REST Linkage để cấu hình các luồng chuyển tiếp tự động trên switch. Bằng cách sử dụng các API endpoint như "/switch/{switch_id}/flow" và "/switch/{switch_id}/group". Các luồng chuyển tiếp tự động được tạo ra bằng cách sử dụng các quy tắc và điều kiện cấu hình trên switch. Ví dụ, quản trị viên có thể tạo ra các luồng chuyển tiếp tự động để xử lý các yêu cầu từ các ứng dụng hoặc các thiết bị khác trên mạng. Các nhóm chuyển tiếp tự động được sử dụng để tối ưu hóa việc tạo các luồng chuyển tiếp trên switch bằng cách sử dụng các quy tắc và điều kiện để nhóm các luồng chuyển tiếp liên quan đến nhau.

3. Đánh giá

Qua nghiên cứu trên có thể thấy việc sử dụng RESTful API và REST Linkage là một lựa chọn phổ biến để quản lý và điều khiển mạng SDN. Điểm mạnh của các công nghệ này là tính dễ sử dụng và hiệu quả trong việc thao tác và cấu hình các tài nguyên mạng. Thay vì sử dụng các công cụ quản lý mạng phức tạp, RESTful API và REST Linkage cho phép người dùng thao tác với mạng SDN một cách đơn giản và trực quan hơn.

So với các công nghệ khác trong mạng SDN, RESTful API và REST Linkage có thể được so sánh với SNMP (Simple Network Management Protocol). SNMP là một giao thức quản lý mạng tiêu chuẩn được sử dụng rộng rãi trong các mạng truyền thống. Tuy nhiên, SNMP có những hạn chế về tính linh hoạt và hiệu quả so với RESTful API và REST Linkage. Ví dụ, để cấu hình một tài nguyên mạng bằng SNMP, người dùng cần phải tìm kiếm và tương tác với nhiều mục lục khác nhau, trong khi RESTful API và REST Linkage cho phép các tài nguyên mạng được quản lý và điều khiển một cách trực quan hơn.

Một ví dụ khác để so sánh với RESTful API và REST Linkage trong Ryu Controller là sử dụng CLI (Command Line Interface) để quản lý và điều khiển mạng SDN. CLI là một công cụ phổ biến cho phép người dùng thao tác với hệ thống mạng bằng cách sử dụng lệnh dòng lệnh. Tuy nhiên, CLI có những hạn chế về tính trực quan và khả năng mở rộng so với RESTful API và REST Linkage. Ví dụ, để cấu hình một tài nguyên mạng bằng CLI, người dùng cần phải nhập các lệnh dòng lệnh phức tạp và tìm kiếm thông tin trong nhiều mục lục khác nhau. Điều này có thể làm tăng thời gian và công sức đối với người dùng. Trong khi đó, RESTful API và REST Linkage cho phép người dùng thao tác với các tài nguyên mạng một cách trực quan hơn, thông qua các yêu cầu HTTP đơn giản và dễ đọc. Ngoài ra, việc triển khai RESTful API và REST Linkage cũng đảm bảo tính khả năng mở rộng cao hơn so với CLI. Các ứng dụng khác trong mạng SDN có thể được tích hợp với Ryu Controller thông qua các giao thức RESTful, tạo ra một hệ thống mạng SDN linh hoạt và hiệu quả. Trong khi đó, việc triển khai các tác vụ quản lý và điều khiển mạng bằng CLI có thể đòi hỏi nhiều công sức và thời gian hơn.

So với các ứng dụng thao tác có giao diện web, RESTful API và REST Linkage trong Ryu Controller cung cấp tính linh hoạt và khả năng mở rộng cao hơn để quản lý và điều khiển mạng SDN. Các ứng dụng thao tác có giao diện web thường có giới hạn về tính tương thích và khả năng mở rộng, do phụ thuộc vào các trình duyệt và các giao thức web phức tạp. Trong khi đó, RESTful API và REST Linkage sử dụng các yêu cầu HTTP đơn giản để truy cập và quản lý các tài nguyên mạng, không phụ thuộc vào các trình duyệt hoặc giao thức web cụ thể nào. Hơn nữa, RESTful cung cấp một giao diện thao tác trực quan hơn so với các ứng dụng thao tác có giao diện web. Người dùng có thể sử dụng các yêu cầu HTTP

đơn giản để truy cập và cập nhật các tài nguyên mạng, thay vì phải điều hướng giữa các trang web và nhập thông tin vào các biểu mẫu phức tạp. Điều này giúp tiết kiệm thời gian và giảm thiểu sự phức tạp khi quản lý và điều khiển mạng SDN.

Việc sử dụng RESTful API và REST Linkage trong Ryu Controller đòi hỏi người dùng phải có kiến thức về HTTP và các phương thức RESTful để hiểu cách các yêu cầu HTTP được sử dụng để truy cập và cập nhật các tài nguyên mạng. Ngoài ra, để triển khai các công nghệ này thành công, người dùng cũng cần có kiến thức về lập trình và mạng để hiểu các khái niệm cơ bản về mạng SDN và cách thức hoạt động của Ryu Controller. Việc đào tạo và triển khai các công nghệ này có thể làm tăng thời gian và chi phí đào tạo và triển khai hệ thống. Tuy nhiên, với sự đầu tư này, người dùng có thể tận dụng tính linh hoạt và khả năng mở rộng của RESTful API và REST Linkage để quản lý và điều khiển mạng SDN một cách hiệu quả.

Trong tổng quan, việc sử dụng RESTful API trong Ryu Controller là một lựa chọn tốt để quản lý và điều khiển mạng SDN. Các công nghệ này đem lại tính dễ sử dụng, linh hoạt và khả năng mở rộng cao, tạo ra một hệ thống mạng SDN mạnh mẽ và hiệu quả. Tuy nhiên, việc triển khai các công nghệ này cũng đòi hỏi người dùng có kiến thức và kinh nghiệm trong lập trình và mạng.

4. Kết luận

Trong bài báo nghiên cứu này, chúng tôi đã tìm hiểu về ứng dụng của RESTful API trong mạng SDN với sử dụng Ryu Controller, đặc biệt là ứng dụng REST linkage, router, QoS, firewall. Kết quả nghiên cứu cho thấy, RESTful API là một cách tiếp cận hiệu quả để quản lý và điều khiển các chức năng này trong mạng SDN.

Với REST linkage, chúng tôi đã triển khai một ứng dụng RESTful API để kết nối các thiết bị mạng với nhau. Việc sử dụng RESTful API giúp tương tác với các thiết bị mạng một cách dễ dàng và linh hoạt hơn, đồng thời cung cấp cho người dùng một giao diện đơn giản để quản lý kết nối giữa các thiết bị. Trong ứng dụng router, chúng tôi đã triển khai một

ứng dụng RESTful API để điều khiển và quản lý các thiết bị định tuyến trong mạng SDN. Việc sử dụng RESTful API giúp tối ưu hóa việc quản lý định tuyến, đồng thời giảm thiểu chi phí và tăng tính linh hoạt cho mạng. Với ứng dụng firewall, chúng tôi đã triển khai một ứng dụng RESTful API để quản lý tường lửa trong mạng SDN. Việc sử dụng RESTful API giúp tối ưu hóa quản lý tường lửa, đồng thời cung cấp cho người dùng một giao diện đơn giản để quản lý các quy tắc tường lửa. Cuối cùng, với ứng dụng QoS, chúng tôi đã triển khai một ứng dụng RESTful API để quản lý chất lượng dịch vụ trong mạng SDN. Việc sử dụng RESTful API giúp tăng tính linh hoạt và khả năng tùy chỉnh của mạng, đồng thời cung cấp cho người dùng một giao diện đơn giản để quản lý QoS.

Tóm lại, việc sử dụng RESTful API trong mạng SDN với sử dụng Ryu Controller là một cách tiếp cận hiệu quả để quản lý và điều khiển các chức năng trong mạng. Việc triển khai các ứng dụng RESTful API cho các chức năng như linkage, router, QoS, firewall giúp tối ưu hóa quản lý mạng, đồng thời cung cấp cho người dùng một giao diện đơn giản và dễ sử dụng để quản lý mạng SDN. Tuy nhiên, việc triển khai RESTful API cần được xây dựng và quản lý một cách cẩn thận để đảm bảo sự an toàn và bảo mật của mạng.

5. Tham khảo

[1] Li Li; Wu Chou; Wei Zhou; Min Luo (2016), “Design Patterns and Extensibility of REST API for Networking Applications”, IEEE Transactions on Network and Service Management (Volume: 13, Issue: 1, March 2016).

[2] David Donato (2016), “Analysis and development of virtual network functions integrated in SDN architectures for remote control applications of automatic machines”.

[3] Tao Hu, Zhen Zhang, Peng Yi a, Dong Liang, Ziyong Li, Quan Ren, Yuxiang Hu, Julong Lan, “A secure application management framework based on REST API access control in SDN-enabled cloud environment”, Journal of Parallel and Distributed Computing, Volume 147, January 2021.

[4] Himanshi Babbar, Shalli Rani (2020), “Performance Evaluation of QoS metrics in Software Defined Networking using Ryu Controller”.

[5] Seungwon Woo, Seungsoo Lee, Jinwoo Kim và Seungwon Shin (2018), “RE-CHECKER: Towards Secure RESTful Service in Software-Defined Networking”, IEEE NFV-SDN 2018.

6. Phụ lục