NGHIÊN CỨU VÀ TRIỂN KHAI CƠ CHẾ ĐỊNH TUYẾN LIÊN MIỀN ĐỂ NÂNG CAO KHẢ NĂNG QUẢN LÝ VÀ VẬN HÀNH TRONG MẠNG SDN

Trần Bá Thiện[*], Đinh Trường Duy[#], Lê Trần Đức[*,+]
[*]Trường Đại Học Bách Khoa - Đại Học Đà Nẵng, Đà Nẵng, Việt Nam
[#]Học Viện Công Nghệ Bưu Chính Viễn Thông, Hà Nội, Việt Nam
[+]Université du Québec à Trois-Rivières, Canada

***Tóm tắt*—** Trong hai thập kỷ gần đây, sự gia tăng kích thước của mạng Internet đã khiến công tác quản lý và vận hành mạng trở nên phức tạp hơn đối với các kỹ sư quản trị mạng. Để giải quyết những vấn đề này, mạng điều khiển bằng phần mềm (Software-Defined Networking – SDN) đã được phát triển. Tuy nhiên, kiến trúc SDN ban đầu chỉ sử dụng một bộ điều khiển duy nhất, dẫn đến khó khăn trong việc kiểm soát toàn bộ mạng và định tuyến giữa các miền SDN khi kích thước mạng tăng lên. Vì vậy, bài báo này đề xuất một giải pháp định tuyến liên miền phân tán, trong đó mỗi miền SDN được quản lý bởi một bộ điều khiển riêng biệt, và các bộ điều khiển của từng miền sẽ trao đổi thông tin trực tiếp với nhau thông qua giao thức Westbound và Eastbound để chia sẻ cấu trúc mạng (topology) và thông tin QoS. Giải pháp này cũng bao gồm một cơ chế định tuyến liên miền SDN dựa trên topology toàn cục và thông tin QoS, hỗ trợ việc tìm đường đi ngắn nhất cho các gói tin trong nội bộ mỗi miền và giữa các miền.

***Từ khóa*—** SDN; SDN controller; định tuyến liên miền; QoS

I. GIỚI THIỆU

Với sự tiến bộ nhanh chóng của công nghệ thông tin, Internet đã trở thành một nền tảng quan trọng cho việc trao đổi thông tin nhanh chóng và dễ dàng. Thực tế, trong hai thập kỷ qua, mạng Internet toàn cầu đã đáp ứng được nhu cầu ngày càng tăng về lưu lượng truy cập và sự giám sát chặt chẽ từ các tổ chức và người tiêu dùng. Tuy nhiên, sự gia tăng của kích thước mạng Internet đã dẫn đến sự tăng cường về số lượng thiết bị phần cứng như switch và router, tạo ra thách thức đối với quản lý và vận hành mạng. Để giải quyết những thách thức này, mạng điều khiển bởi phần mềm (SDN) [1,2] đã được phát triển.

SDN là một cách tiếp cận mới trong mạng máy tính, nơi một bộ điều khiển phần mềm hoặc API mã nguồn mở, như OpenStack [3] hoặc Cloudstack [4], được sử dụng để điều khiển việc truyền dữ liệu trong mạng thay vì dựa vào thiết bị phần cứng như switch và router [5]. Tuy nhiên, khi mạng SDN trở nên phức tạp hơn với nhiều miền (Domain), một bộ điều khiển không đủ để kiểm soát toàn bộ các miền, và việc định tuyến giữa các miền trở thành một vấn đề cần giải quyết.

Các giải pháp trước đây đã được đề xuất để giải quyết vấn đề định tuyến liên miền trong mạng SDN. Trong số đó, phương pháp phân cấp, nơi một bộ điều khiển trung tâm phối hợp với các bộ điều khiển của mỗi miền, đã được sử dụng rộng rãi [6]. Tuy nhiên, phương pháp này gặp hạn chế trong việc mở rộng mạng và độ tin cậy khi bộ điều khiển trung tâm gặp sự cố.

Chính vì vậy, bài báo này đề xuất một giải pháp định tuyến liên miền phân tán mới, nơi mỗi miền SDN được điều khiển bởi một bộ điều khiển riêng và các bộ điều khiển trao đổi thông tin trực tiếp với nhau thông qua các giao thức Westbound và Eastbound. Giải pháp này không chỉ giải quyết các hạn chế của các giải pháp trước đây, mà còn tối ưu hóa hiệu suất mạng và đảm bảo độ tin cậy cao hơn trong môi trường liên miền.

Phần II của bài báo sẽ đánh giá tổng quan các giải pháp hỗ trợ định tuyến liên miền hiện có trong mạng SDN. Phần III mô tả triển khai giải pháp đề xuất cho định tuyến liên miền trong mạng SDN. Cuối cùng, phần kết luận của bài báo được đề cập trong phần IV.

II. TỔNG QUAN CÁC GIẢI PHÁP HỖ TRỢ ĐỊNH TUYẾN LIÊN MIỀN HIỆN CÓ TRONG MẠNG SDN

Trong một mạng quy mô lớn với nhiều miền SDN (Hình 1), mục tiêu của định tuyến liên miền là xác định đường đi ngắn nhất cho gói tin từ một máy chủ (host) thuộc miền SDN này sang một máy chủ thuộc miền SDN khác.

Trong thiết kế ban đầu của SDN (Hình 2), toàn bộ mạng SDN chỉ sử dụng một bộ điều khiển duy nhất để điều khiển tất cả các OpenFlow switch bên dưới. Trong quá trình hoạt động của mạng SDN, các OpenFlow switch sẽ thường xuyên gửi các sự kiện lên bộ điều khiển, giúp bộ điều khiển nắm bắt được các đường truyền dữ liệu bên dưới cũng như việc chuyển tiếp gói tin của toàn bộ mạng. Tuy nhiên, khi số lượng switch tăng lên, số lượng sự kiện được gửi lên bộ điều khiển cũng tăng đáng kể, dẫn đến việc bộ điều khiển

Tác giả liên hệ: Đinh Trường Duy,
Email: duydt@ptit.edu.vn
Đến tòa soạn: 5/2023, chỉnh sửa: 6/2023, chấp nhận đăng: 7/2023.

Controller dễ bị quá tải và làm giảm khả năng mở rộng của mạng.

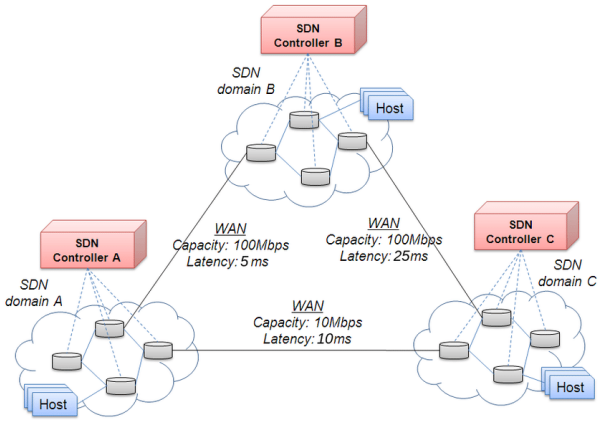
Có nhiều giải pháp đã được đề xuất để giải quyết vấn đề này, mỗi giải pháp đều có những ưu điểm và hạn chế riêng. Trong phần này, chúng tôi sẽ tổng quan một số giải pháp tiêu biểu.

Trước khi đi sâu vào chi tiết từng giải pháp, chúng tôi sẽ giới thiệu một đánh giá tổng quát của các giải pháp như sau:

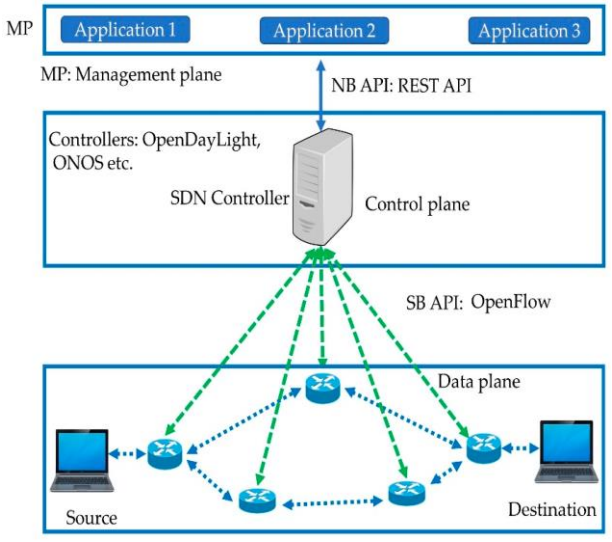
Giải pháp HyperFlow [7], tập trung vào việc tối ưu hóa quá trình chuyển mạch. Mặc dù giải pháp này đem lại hiệu quả trong việc cải thiện hiệu suất mạng, nó không đề cập đến vấn đề quản lý hiệu suất mạng trong môi trường liên miền.

Giải pháp WheelFS [8], giải quyết vấn đề trên bằng cách sử dụng một cấu trúc mạng phân tán. Tuy nhiên, giải pháp này chưa tối ưu hóa hiệu suất mạng.

Giải pháp Root Controller [9], cố gắng giải quyết cả hai vấn đề trên nhưng phụ thuộc nhiều vào cấu trúc hạ tầng mạng hiện có.



Hình 1. Mạng SDN với nhiều miền là A, B và C



Hình 2. Kiến trúc của mạng SDN ban đầu

Phân tích các giải pháp hiện có

Để giảm tải cho tầng Control Plane trong bối cảnh số lượng thiết bị tăng lên, cần có một phương pháp để xử lý các sự kiện được gửi từ switch lên bộ điều khiển. Các tác giả Soheil Hassas Yeganeh và Yashar Ganjali [6] đã thiết kế và tạo ra một Control Plane phân tán có tên là Kandoo [6] để giải quyết vấn đề này. Kandoo bao gồm các bộ điều khiển được chia thành hai cấp độ: Local Controller và Root Controller.

Local Controller sẽ thực thi các chương trình ở mức cục bộ và trực tiếp điều khiển các OpenFlow switch cũng như xử lý các sự kiện được gửi từ chúng. Các Local Controller không kết nối trực tiếp với nhau, và chúng chỉ nắm quyền kiểm soát các switch mà chúng quản lý. Root Controller sẽ thực thi chương trình để điều khiển toàn bộ mạng (Hình 3).

Các Local Controller có thể điều khiển một hoặc nhiều OpenFlow switch. Khi số lượng switch tăng lên, chỉ cần tăng số lượng Local Controller, và chúng sẽ xử lý các sự kiện từ các switch thay cho Root Controller, giúp giảm tải công việc cho Root Controller.

Trong Kandoo, các Controller có vai trò tương tự như OpenFlow Controller thông thường, nhưng chúng được mở rộng để xác định các yêu cầu từ các chương trình. Khi Root Controller cần cài đặt các luồng mới lên các OpenFlow switch, nó sẽ ủy quyền cho Local Controller để thực hiện công việc này.

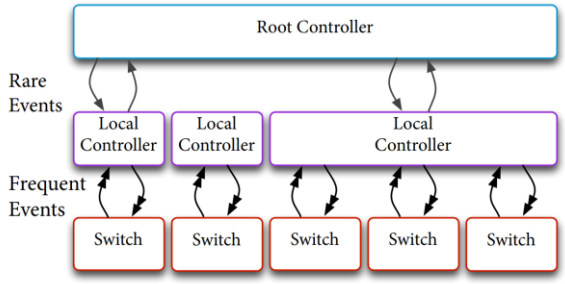
Các chương trình điều khiển hoạt động bằng cách sử dụng các trừu tượng được cung cấp bởi Controller, và chúng không cần biết về kiến trúc bên trong của Kandoo (Hình 4). Thực chất, các chương trình điều khiển này (Controller) là chương trình OpenFlow và chúng có thể gửi tin nhắn OpenFlow cũng như lắng nghe các sự kiện. Ngoài ra, chúng còn có khả năng tạo ra các sự kiện nội bộ trong mạng cho các chương trình khác. Các chương trình này cũng có thể gửi và nhận sự kiện từ các chương trình khác trong mạng.

Nhờ thiết kế này, Kandoo giúp giảm tải cho tầng Control Plane và tăng khả năng mở rộng của mạng SDN, đồng thời duy trì tính linh hoạt và khả năng điều khiển của OpenFlow Controller thông thường.

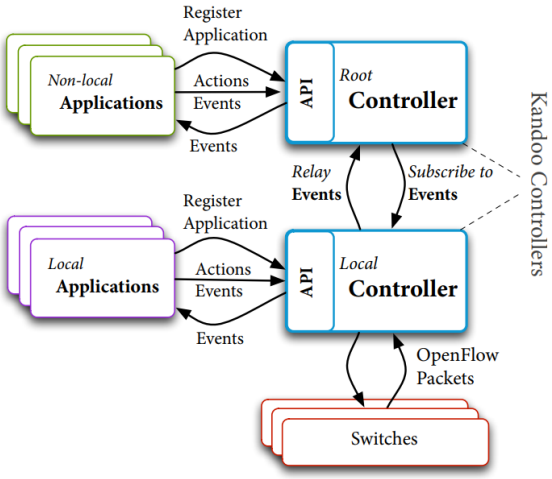
Dù giải pháp Kandoo đã giúp giảm tải cho Root Controller và tăng khả năng mở rộng của mạng SDN, nhưng vẫn tồn tại một vấn đề khi Root Controller gặp trục trặc sẽ ảnh hưởng đến toàn bộ mạng. Để giải quyết vấn đề này, Amin Tootoonchian đã đề xuất HyperFlow, một Control Plane phân tán được sử dụng với giao thức OpenFlow, cho phép các kỹ sư quản trị mạng triển khai nhiều Controller trong mạng của họ [7].

Trong HyperFlow, tất cả các Controller chia sẻ thông tin với nhau để nắm bắt được topology của toàn bộ mạng. Mỗi Controller sẽ điều khiển các OpenFlow switch do nó quản lý, giúp giảm thiểu thời gian thiết lập luồng dữ liệu trên mỗi switch. Các ứng dụng chạy trên HyperFlow Controller chịu trách nhiệm đồng bộ hóa kiến trúc mạng toàn cục của các Controller, và các Controller sẽ giao tiếp với nhau thông qua hệ thống pub/sub tin nhắn.

Mạng SDN sử dụng HyperFlow bao gồm các thành phần: OpenFlow Switch, NOX Controller chạy các chương trình HyperFlow, và một hệ thống pub/sub sự kiện để các Controller giao tiếp với nhau. Mỗi switch sẽ kết nối đến Controller gần nó nhất, và khi một Controller gặp lỗi, các switch do nó điều khiển sẽ được cấu hình lại để kết nối với Controller hoạt động gần đó.

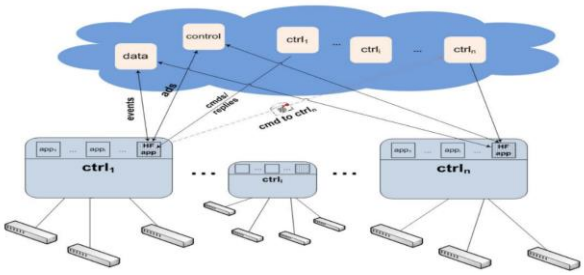


Hình 3. Kiến trúc tổng quan của Kandoo [6]



Hình 4. Kiến trúc chi tiết của Kandoo [6]

Nhờ giải pháp HyperFlow, mạng SDN không chỉ giảm tải cho Controller mà còn tăng độ ổn định, giảm độ trễ và tăng khả năng mở rộng khi quy mô mạng tăng lên.



Hình 5. Kiến trúc của HyperFlow [7]

Trong hệ thống HyperFlow, để các Controller biết được topology của toàn bộ mạng, các chương trình HyperFlow trên mỗi Controller gửi các sự kiện thông báo về sự thay đổi trạng thái của mạng lên hệ thống pub/sub (Hình 5). Các Controller khác sau đó trả lời các sự kiện đó để xây dựng lại trạng thái mạng mới. Hệ thống pub/sub lưu trữ các sự kiện được các Controller Publish lên và phải được lưu trữ theo đúng thứ tự publish.

Controller của mỗi site phải cập nhật thông tin từ các Controller kế cận để tránh tắc nghẽn kết nối giữa các site. Hệ thống pub/sub được cài đặt bằng WheelFS [8], một hệ thống tệp phân tán được thiết kế để cung cấp khả năng lưu trữ trên diện rộng linh hoạt cho các ứng dụng phân tán.

Mỗi Controller sẽ subscribe 3 kênh trong hệ thống pub/sub, bao gồm kênh dữ liệu, kênh điều khiển và kênh riêng của nó. Tất cả các Controller trong mạng được cấp quyền để publish vào tất cả các kênh và subscribe 3 kênh trên. Các chương trình HyperFlow sẽ publish các sự kiện ứng dụng và mạng cục bộ đã chọn lên kênh dữ liệu. Các sự kiện và lệnh OpenFlow nhắm mục tiêu đến một Controller cụ thể được publish đến kênh tương ứng của Controller đó.

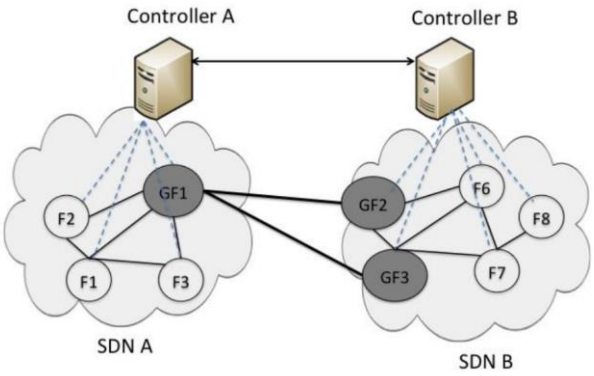
Mỗi Controller trong HyperFlow phải thông báo định kỳ tình trạng của nó lên kênh điều khiển để các Controller khác biết. Trong trường hợp một Controller gặp sự cố, các Controller khác cũng sẽ sớm biết được và có thể thực hiện các biện pháp khắc phục hoặc điều chỉnh mạng để đảm bảo hoạt động ổn định của toàn bộ hệ thống.

Khi mạng được phân vùng, WheelFS trên mỗi phân vùng tiếp tục hoạt động độc lập. Controller trên mỗi phân vùng không còn nhận được thông báo từ các Controller của các phân vùng khác và cho rằng chúng đã bị lỗi. Khi các phân vùng kết nối lại, các nút WheelFS trong cả hai phân vùng sẽ đồng bộ hóa lại. Do đó, Controller nhận được thông báo về tất cả các sự kiện xảy ra trong phân vùng khác trong khi chúng đã bị ngắt kết nối, và tất cả các Controller sẽ biết được topology của toàn bộ mạng.

Tuy nhiên, giải pháp này vẫn có nhược điểm là khả năng mở rộng bị hạn chế. Khi mạng có nhiều miền với nhiều Controller, số lượng sự kiện được publish vào hệ thống pub/sub rất lớn, dễ làm cho hệ thống pub/sub bị quá tải.

Một kiến trúc quản lý liên miền phân tán mới, mở rộng, cho phép SDN Controller với các chức năng đa miền để định tuyến liên miền và đưa ra các quyết định quản lý luồng mà không cần sự ủy quyền của Root Controller, đã được đề xuất trong [9]. Trong giải pháp này, sẽ có nhiều miền SDN, mỗi miền sẽ có ít nhất một Controller và nhiều switch được kết nối với nhau để tạo thành tầng data plane. Một switch kết nối đến switch ở miền SDN khác được gọi là switch biên. Một switch thuộc một miền SDN và chỉ kết nối đến các switch thuộc miền đó được gọi là các switch nội bộ.

Kiến trúc này giúp giải quyết vấn đề mở rộng của mạng SDN, giảm thiểu độ trễ, và tăng khả năng chịu đựng lỗi. Tuy nhiên, để đảm bảo hiệu quả cao và ổn định trong việc triển khai, các kỹ sư mạng cần xem xét và lựa chọn phương án tối ưu cho hệ thống của họ, cũng như tiếp tục nghiên cứu và đổi mới để đáp ứng nhu cầu không ngừng phát triển của công nghệ mạng hiện đại.



Hình 6. Kiến trúc SDN nhiều miền phân tán [9]

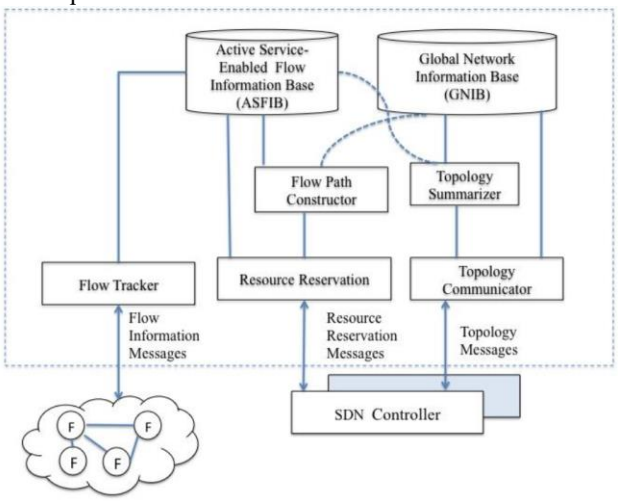
Ở Hình 6, Miền SDN A bao gồm các switch nội bộ là F1, F2, F3 và switch biên là GF1. Miền SDN B bao gồm

các switch nội bộ là F6, F7, F8 và các switch biên là GF2 và GF3. Các đường dẫn GF1 – GF2 và GF1 – GF3 là các đường dẫn liên miền, các đường dẫn F1 – F2 hoặc F7 – F8 là đường dẫn nội miền. 2 Controller của 2 miền A và B sẽ kết nối với nhau để trao đổi thông tin [12].

Các Controller phải chia sẻ các thông tin quan trọng cho nhau như thông tin về topology, thông tin định tuyến nên mạng kết nối giữa các Controller phải có tính bảo mật cao và tình sẵn sàng cao giống mạng truyền thống. Trong giải pháp này, Mỗi SDN Controller sẽ chia sẻ với nhau thông tin tóm tắt của topology mà nó quản lý, để Mỗi Controller biết được topology khái quát của toàn bộ mạng, các switch biên chỉ biết các switch biên thuộc miền SDN khác mà nó kết nối đến [9].

Các Controller không nhất thiết phải biết topology chi tiết của các miền khác, mà chúng chỉ cần chia sẻ thông tin topology tóm tắt để mỗi Controller biết topology toàn cục khái quát và các switch biên mỗi miền để có thể định tuyến liên miền. Ngoài ra các Controller còn cần phải biết thêm thông tin về khả năng dịch vụ của các miền khác dọc theo các đường đi của luồng dịch vụ được kích hoạt.

Để các Controller có thể chia sẻ cho nhau thông tin về các topology tóm tắt trong đó có thông tin về các đường dẫn đi qua miền mà nó quản lý thì mỗi Controller phải được trang bị chức năng đa miền cụ thể, có thể được triển khai bên trong SDN Controller hoặc là các ứng dụng bên ngoài. Các Controller phải duy trì cơ sở dữ liệu tài nguyên mạng toàn cục để nắm bắt topology toàn cục và thông tin luồng hoạt động. Controller với chức năng đa miền phải duy trì ba cơ sở dữ liệu là cơ sở dữ liệu chứa thông tin về topology do nó quản lý và thông tin về dịch vụ cụ thể liên quan đến miền riêng của Controller (DIB) [9]. Cơ sở dữ liệu thứ hai (GNIB) [9] chứa thông tin mới nhất về topology của toàn bộ mạng và các thông tin về dịch vụ được thu thập từ các Controller khác. DIB và GNIB là 2 cơ sở dữ liệu quan trọng chứa thông tin cần thiết để định tuyến liên miền SDN. Cơ sở dữ liệu thứ 3 là (ASFIB) [9] , là nơi chứa thông tin về tất cả các đường đi đang hoạt động mà gói tin bên ngoài có thể đi qua của miền đó.



Hình 7. SDN Controller với chức năng đa miền [9]

Topology Summarizer là module xác định topology tóm tắt của miền SDN mà Controller đó đang quản lý để chia sẻ thông tin đó với Controller khác. Topology Communicator là module nhận về và xử lý thông tin topology tóm tắt từ miền SDN khác và sau đó gởi kết quả cho GNIB. Flow Path Constructor là module sử dụng thuật toán để tính toán đường đi tối ưu nhất liên miền, module này phải được cung cấp thông tin QoS trước khi tính đường đi tối ưu nhất để định tuyến liên miền. Flow Tracker là module theo dõi đường đi trong miền SDN để đảm bảo đường đi đó sẵn sàng cho việc định tuyến liên miền, nó sẽ đánh giá chất lượng của các đường đi nội miền. Resource Reservation là module đảm bảo băng thông cho đường đi tối ưu được tính toán bởi Flow Path Constructor, việc đảm bảo băng thông phải được thực hiện trên các Controller của các miền mà đường đi ngắn nhất đi qua sử dụng giao thức RSVP.

Như vậy, việc phân tán Control Plane trong mạng SDN đã được thực hiện để giảm tải cho Controller và tăng khả năng mở rộng. Cụ thể: Giải pháp Kandoo sử dụng 2 cấp độ Controller để giảm tải cho Root Controller và tăng khả năng mở rộng. Tuy nhiên nó vẫn bị hạn chế về khả năng mở rộng khi quy mô mạng lớn. HyperFlow sử dụng nhiều Controller thay thế nhau điều khiển mạng. Các Controller giao tiếp với nhau thông qua hệ thống pub/sub để biết topology của mạng. Nó giúp tăng độ ổn định, giảm độ trễ và mở rộng mạng. Ngoài ra, Kiến trúc SDN nhiều miền phân tán được đề xuất. Mỗi miền có ít nhất 1 Controller và nhiều switch. Các Controller kết nối với nhau và trao đổi thông tin topology tóm tắt của miền. Hơn nữa, SDN Controller được trang bị các chức năng đa miền để định tuyến liên miền. Nó có các cơ sở dữ liệu về topology của miền cũng như khả năng dịch vụ. Tóm lại, sau khi nghiên cứu các giải pháp hỗ trợ định tuyến liên miền hiện có trong mạng SDN đã cho thấy phân tán Control Plane và sử dụng nhiều Controller trong mạng SDN giúp giảm tải cho riêng một Controller, tăng mở rộng và độ ổn định cho toàn bộ mạng.

III. GIẢI PHÁP ĐỀ XUẤT CHO ĐỊNH TUYẾN LIÊN MIỀN TRONG SDN

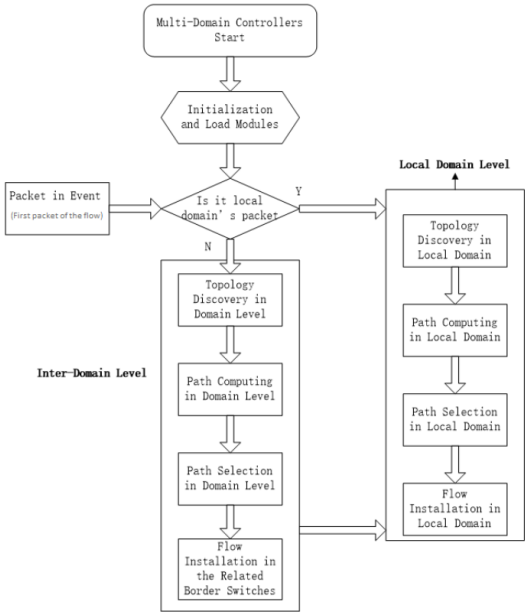
A. Thiết kế giải pháp định tuyến liên miền trong mạng SDN

Bài báo đề xuất giải pháp trong đó các Controller của mỗi miền sẽ trao đổi thông tin trực tiếp với nhau để chia sẻ thông tin topology và QoS mà không cần thông qua Root Controller. Việc định tuyến sẽ bao gồm 2 cấp độ là định tuyến nội miền SDN và định tuyến liên miền SDN. Định tuyến liên miền sẽ được thực hiện bằng cách cài đặt nguyên tắc chuyển tiếp gói tin theo địa chỉ IP giữa các miền dựa trên topology của toàn bộ mạng. Định tuyến nội miền sẽ được thực hiện bởi Controller của miền SDN đó.

Giải pháp đề xuất sẽ bao gồm 3 phần là :

- Xác định Topology của toàn bộ mạng. Việc xác định Topology sẽ bao gồm 2 phần là xác định Topology nội bộ mỗi miền SDN và xác định Topology toàn cục liên miền SDN. Mỗi miền SDN là một subnet với không gian địa chỉ IP duy nhất.
- Tính đường đi ngắn nhất liên miền. Giải pháp sẽ tính đường đi ngắn nhất từ máy chủ ở miền SDN này đến máy chủ ở miền SDN khác.

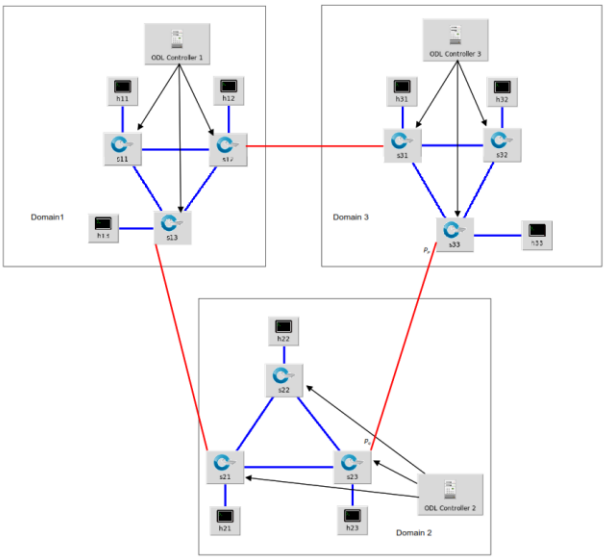
- Cài đặt kết quả tính đường đi ngắn nhất. Controller sẽ cài đặt kết quả tính đường đi ngắn nhất lên các switch nằm trên đường đi ngắn nhất đó.



Hình 8. Tổng quan về các bước trong giải pháp định tuyến liên miền SDN

B. *Xác định Topology của toàn bộ mạng*

Controller của mỗi miền SDN sẽ biết được thông tin chi tiết về Topology của miền mà nó quản lý. Như trong hình 9, mỗi Controller có thể lấy được thông tin Topology nội miền bằng cách dùng giao thức LLDP. Tuy nhiên, các Controller không thể nhận ra đường dẫn kết nối giữa các miền, vì vậy các Controller phải kết nối với nhau để trao đổi thông tin Topology của miền SDN mà chúng quản lý.

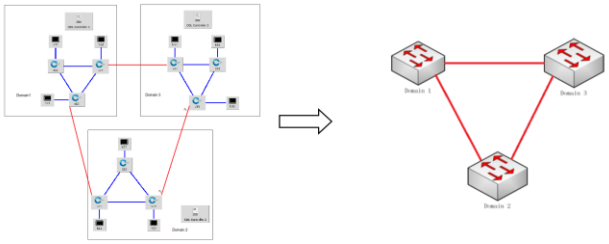


Hình 9. Topology liên miền

Các miền, switch, host trong mạng sẽ được mô tả như sau:

- Miền thứ i trong mạng sẽ được ký hiệu là D_i, mạng toàn cục sẽ bao gồm tập các miền D_1, D_2 , D_3,…, D_i,…, D_n
- $S_{i,n}$ là switch n trong miền D_i, S_i là tập tất cả các switch trong miền D_i.
- $H_{i,n}$ là host n trong miền D_i
- P_i là cổng i của switch hoặc host
- L_i là tập tất cả các đường dẫn của miền D_i, đường dẫn từ cổng p của $S_{i,n}$ đến cổng q của $S_{j,m}$ được ký hiệu là $[S_{i,n} : P_p, S_{j,m} : P_q]$. Nếu i = j thì đường dẫn trên là đường dẫn nội miền D_i, ngược lại thì đường dẫn trên là đường dẫn liên miền i và j trong đó $S_{i,n}$ là switch nguồn và $S_{j,m}$ là switch đích
- Đường dẫn giữa cổng p của switch $S_{i,n}$ và cổng q của host $H_{i,m}$ trong miền D_i được ký hiệu là $[S_{i,n} : P_p, H_{i,m} : P_q]$

Nếu miền i có đường dẫn $[S_{i,n} : P_p, S_{j,m} : P_q]$ và miền j cũng có đường dẫn tương ứng là $[S_{j,m} : P_q, S_{i,n} : P_p]$, thì 2 miền đó kết nối với nhau. Trong giải pháp này, đường dẫn kết nối giữa các miền sẽ được Controller học thông qua giao thức BGP. Sau khi biết được đường dẫn kết nối giữa các miền, các Controller sẽ biết được Topology của toàn bộ mạng ở cấp độ miền (Hình 10).

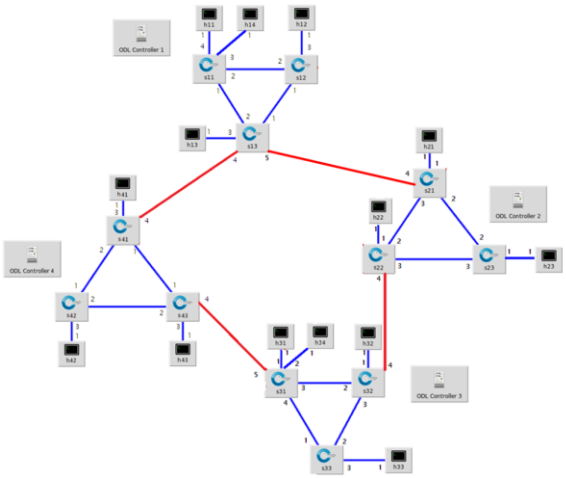


Hình 10. Topology toàn cục chi tiết (bên trái) và Topology ở cấp độ miền (bên phải)

Thuật toán sau đây sẽ được cài đặt trên Controller của mỗi miền để xác định Topology toàn cục ở cấp độ miền

1: Call the northbound API of current local domain D_k to get the local topology
2: Add all local switches into the local switches set S_k, and add all nodes and links L_k into local domain topology graph G
3: for D_i in Domains do
4: if D_i is not D_k then
5: Acquire the topology information (S_i and link set L_i) of the domain D_i by west/east protocol
6: Add D_i into domain level topology graph domainGraph as a node
7: if $S_{i,n}$ not in =S_k then
8: add $S_{i,n}$ into the peer switches list – peerSwList
9: add the links connected to $S_{i,n}$ into external links – externalLinkList
10: end if
11: end if
12: end for
13: for $S_{i,n}$ in peerSwList do
14: for D_k in Domains do
15: if ∃[S_k,m, $S_{i,n}$] ∈ externalLinkList and peer Domain D_i's Link [$S_{i,n}$, $S_{k,m}$] ∈ L_i then
16: identify Si,n , Sk,m and as border Switches and link [$S_{i,n}$, $S_{k,m}$] as an external link
17: record the port number of the two border switches
18: add a link[D_i, D_k] into domainGraph
19: end if
20: end for

21: end for
22: for D_i in Domains do
23: for D_j in Domains, j>=i+1 do
24: if D_i is not D_k and D_j is not D_k then
25: if at ∃ $[S_{i,n}, S_{j,m}] \in L_i$ and ∃ $[S_{j,m}, S_{i,n}] \in L_j$ then
26: add link[D_i, D_j] into domainGraph
27: end if
28: end if
29: end for
30: end for



Hình 11. Topology chi tiết toàn cục

Giả sử giờ ta có Topology của toàn bộ mạng như hình 11, trong đó:

Miền D_1:

- Tập các switch là S_1 : {s11, s12, s13}
- Tập các host là H_1 : {h11, h12, h13, h14}
- Tập các đường dẫn là L_1 : {[s11:1, s13:2], [s11:2, s12:2], [s11:3, h14:1], [s11:4, h11:1], [s12:1, s13:1], [s12:3, h11:1],[s13:3, h13:1], [s13:3, h13:2], [s13:4, s41:4], [s13:5, s21:4]}

Miền D_2 :

- Tập các switch S_2 : {s21, s22, s23}
- Tập các host H_2 : { h21, h22, h23}
- Tập các đường dẫn L_2 : {[s21:1, h21:1], [s21:2, s23:2], [s21:3, s22:2], [s21:4, s13:5], [s22:1, h22:1], [s22:2, s21:3], [s22:3, s23:3],[s22:4, s32:4], [s23:1, h23:1], [s23:2, s21:2] , [s23:3, s22:3]}

Miền D_3 :

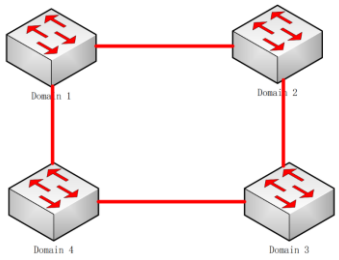
- Tập các switch S_3 : { s31, s32, s33}
- Tập các host H_3 : { h31, h32, h33, h34}
- Tập các đường dẫn L_3 : {[s31:1, h31:1], [s31:2, h34:1], [s31:3, s32:2], [s31:4, s33:1], [s31:5, s43:4], [s32:1, h32:1],[s32:3, s33:2], [s32:4, s22:4], [s33:3, h33:1] }

Miền D_4 :

- Tập các switch S_4 : { s41, s42, s43}
- Tập các host H_4 : { h41, h42, h43}
- Tập các đường dẫn L_4 : {[s41:1, s43:1], [s41:2, s42:1], [s41:3, h41:1], [s41:4, s13:4], [s42:2, s43:2], [s42:3, h42:1],[s43:3, h43:1], [s43:4, s31:5]}

Controller D_1 sẽ tiến hành kiểm tra tập đường dẫn L_1 và tập switch S_1, sau đó Controller D_1 sẽ biết được 2 đường dẫn [s13:4, s41:4] và [s13:5, s21:4] sẽ dẫn đến miền SDN khác. Sau đó, Controller D_1 sẽ trao đổi thông tin Topology với các Controller khác và nhận thông tin Topology từ chúng.

Từ miền 2 : [s21:4, s13:5], [s22:4, s32:4]
Từ miền 3 : [s31:5, s43:4], [s32:4, s22:4]
Từ miền 4 : [s41:4, s13:4], [s43:4, s31:5]



Hình 12. Topology toàn cục ở cấp độ miền

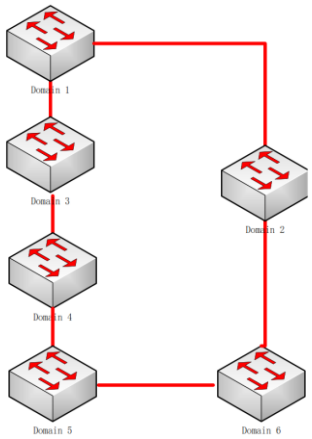
Sau khi chia sẻ thông tin Topology với nhau, các Controller sẽ xác định được Topology của toàn bộ mạng ở cấp độ miền như hình 12.

C. Xác định Topology của toàn bộ mạng

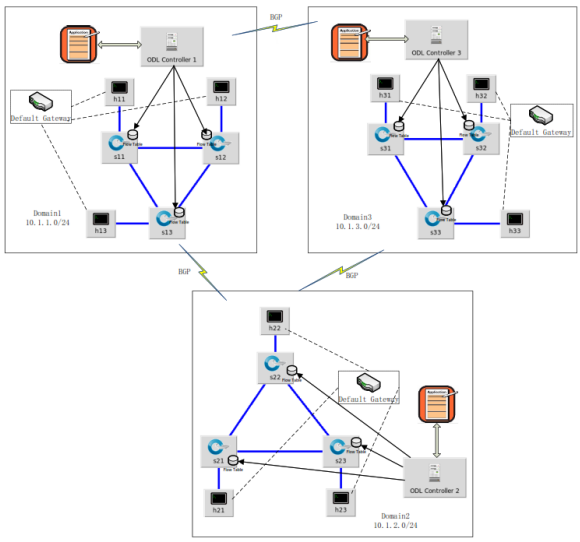
Dựa trên Topology toàn cục cấp độ miền, Mỗi Controller có thể tính đường đi ngắn nhất liên miền giữa host nguồn và host đích. Giải pháp này sẽ sử dụng bước nhảy giữa các miền để làm weight metric để tính toán đường đi ngắn nhất. Thuật toán được mô tả như sau:

1: graph G ← Local Topology Information
2: domainGraph ← Global Topology Discovery
3: srcDomain ← the Domain of the source host
4: dstDomain ← the Domain of the destination host
5: shortestPaths ← get all shortest paths（domainGraph, srcDomain, dstDomain）
6: for shortestpath in shortestPaths do
7: if current Domain is the end node of the shortestpath then
8: return shortestpath
9: else
10: identify the border switch and port along the shortestpath
11: weight metric of the shortestpath ← call the path computing method of the peer domain along the shortest path
12: end if
13: end for

14: shortestpath ← the path which has the least weight metric of all the shortestpath
15: return shortestpath



Hình 13. Topology toàn cục cấp độ miền với 6 miền



Hình 14. Giao tiếp giữa các miền

Ta sẽ sử dụng hình 12 để làm ví dụ, trong Topology toàn cục, ta có tập miền là {D_1, D_2, D_3, D_4}, tập đường dẫn là {[D_1, D_2], [D_1, D_4], [D_2, D_3], [D_3, D_4]}. Giờ ta muốn tính đường đi ngắn nhất liên miền từ h11 đến h31 sử dụng weight metric, Miền của h11 là D_1, Miền của h31 là D_3, vì vậy nên ta có 2 đường đi ngắn nhất là {D_1, D_2, D_3} và {D_1, D_4, D_3} vì weight metric của 2 đường đi này bằng nhau và đều bằng 2.

Một ví dụ khác là Topology ở hình 13 , ta có tập miền là {D_1, D_2, D_3, D_4, D_5, D_6}. Tập đường dẫn là {[D_1, D_2], [D_1, D_3], [D_3, D_4], [D_4, D_5], [D_5, D_6], [D_2, D_6]}. Có 2 đường đi từ D_1 đến D_6 . Một đường là {D_1, D_3, D_4, D_5, D_6} có weight metric là 4, đường khác là {D_1, D_2, D_6}, vì vậy đường đi ngắn nhất là {D_1, D_2, D_6} vì weight metric của nó là 2 nhỏ hơn đường đi kia.

D. Cài đặt kết quả tính đường đi ngắn nhất liên miền

Qua 2 bước trên, ta đã có Topology toàn cục ở cấp độ miền và tính toán được đường đi ngắn nhất liên miền. Khi Controller của miền nguồn nhận gói tin đầu tiên được gởi bởi host nguồn đến host đích ở miền khác, Controller đó cần cài đặt luồng dữ liệu liên miền lên các switch biên dọc theo đường đi ngắn nhất liên miền. Controller thông thường sẽ đưa ra các quyết định chuyển tiếp gói tin dựa trên địa chỉ MAC đích. Nhưng khi chuyển tiếp các gói tin liên miền thì quyết định chuyển tiếp gói tin dựa trên địa chỉ IP đích. Controller cần phải phân biệt được việc chuyển tiếp gói tin nội miền và liên miền.

Giải pháp này sử dụng một địa chỉ là địa chỉ MAC cổng ảo và địa chỉ IP tương ứng. Địa chỉ IP cổng ảo sẽ được cấu hình trên mỗi host như là địa chỉ IP của default gateway. Việc so sánh tương ứng giữa địa chỉ MAC cổng ảo và địa chỉ IP cổng ảo cũng được cấu hình trong bảng ARP của host. Khi một host muốn gửi một gói tin đến một host nằm trong miền khác, nó sẽ chèn địa chỉ MAC cổng ảo vào header của gói tin. Khi Controller thấy địa chỉ MAC của cổng ảo trong một gói tin, nó biết rằng nó đang xử lý một gói tin liên miền và kiểm tra IP header của gói tin để xác định đường dẫn liên miền ngắn nhất.

Qua 2 bước đầu tiên là xác định Topology toàn cục cấp độ miền và tính đường đi ngắn nhất liên miền, ta đã có đường đi ngắn nhất ở cấp độ miền và switch tại biên của đường đi trên mỗi miền. Vì vậy, Controller của miền nguồn cần cài đặt luồng lên các switch biên nằm trên đường đi ngắn nhất của miền nguồn, ngoài ra nó còn phải xác định đường đi ngắn nhất giữa switch nguồn và switch biên được chọn. Controller của miền chuyển tiếp cũng phải xác định đường đi ngắn nhất giữa 2 switch biên và Controller của miền đích phải xác định đường đi ngắn nhất giữa switch biên và switch đích. Các Controller trên cài đặt luồng phải khớp với địa chỉ MAC và địa chỉ IP dọc theo đường đi ngắn nhất nội miền.

Thuật toán cài đặt luồng như sau:

1: Input: domain shortest path, source host, destination host
2: for domain in domain shortest path do
3: if domain is source domain then
4: identify the border switch and port to downstream peer domain in shortest path
5: push the forward and reverse flows to the border switch
6: detemines the intra-domain shortest path between the source switch and the chosen border switch
7: push flows along the intra-domain shortest path
8: else if domain is destination domain then
9: identify the border switch and port to upstream peer domain in shortest path
10: push the forward and reverse flows the border switch
11: determines the intra-domain shortest path between the chosen border switch and the destination switch
12: push flows along the intra-domain shortest path
13: else
14: identify the border switches and ports' number which are linked downstream and upstream domain in the transit domain
15: push the forward and reverse flows to downstream domain

16: push the forward and reverse flows to upstream domain
17: determines the intra-domain shortest path between the two chosen border switches
18: push flows along the intra-domain shortest path
19: end for

Ta sẽ lấy ví dụ Topology như Hình 14 với host nguồn và đích là h11 và h31, ta có đường đi ngắn nhất liên miền là {D_1, D_2, D_3}. Dựa trên đường đi ngắn nhất, Controller của miền nguồn có thể xác định switch biên là s13 và cổng của nó là 5 được kết nối với miền D_2. Chính vì thế Controller của miền nguồn sẽ cài đặt luồng đi gói tin vào flow table của s13 để điều hướng đường đi gói tin đến miền D_2, và nhận gói tin ngược từ D_2 về D_1 (bảng 1).

Bảng 1. Luồng đi của gói tin qua s13 của miền D_1

Đường đi gói tin	Hành động	Miền:Switch
IP Src(h11)-> IP Dst(h31) Gói tin đi vào ở cổng 2	Gói tin đi ra ở cổng 5	D_1:s13
IP Src(h31)-> IP Dst(h11) Gói tin đi vào ở cổng 5	Gói tin đi ra ở cổng 2	D_1:s13

Khi Controller D_1 nhận được tin nhắn chuyển tiếp, nó sẽ tìm địa chỉ MAC đích của gói tin đó là địa chỉ MAC cổng ảo (ví dụ 00:00:00:00:00:64). Controller sau đó sẽ tìm đường dẫn ngắn nhất nội miền giữa switch nguồn là s11 và switch biên được chọn là s13 – [s11, s13] và cài đặt luồng dữ liệu lên đường đi ngắn nhất nội miền (bảng 2).

Bảng 2. Luồng đi gói tin bên trong miền D_1

Đường đi gói tin	Hành động	Miền:Switch
IP Src(h11)-> IP Dst(h31) Mac Dst(00:00:00:00:00:64)	Đi ra ở cổng 1	D_1:s11

Khi gói tin đã đi đến miền chuyển tiếp D_2, thông qua switch biên s21, Controller D_2 xác định gói tin này từ miền khác đến vì nó đến từ đường dẫn bên ngoài. Controller D_2 sẽ xử lý IP header và xác định switch biên để gói tin đi đến miền đích, s22. Vì vậy, Controller D_2 sẽ cài đặt luồng chuyển tiếp gói tin vào flow table của switch biên s22 để chuyển tiếp gói tin ở cổng 4 đến miền D_3 và cũng để nhận gói tin ngược lại từ miền D_3. Tương tự, luồng chuyển tiếp cũng được cài đặt lên switch biên s21 (bảng 3).

Bảng 3. Luồng đi gói tin qua switch biên của miền D_2

Đường đi gói tin	Hành động	Miền:Switch
IP Src(h11)-> IP Dst(h31) Vào cổng 4	Ra ở cổng 3	D_2:s21
IP Src(h31)-> IP Dst(h11) Vào cổng 3	Ra ở cổng 4	D_2:s21
IP Src(h11)-> IP Dst(h31) Vào cổng 2	Ra ở cổng 4	D_2:s22
IP Src(h31)-> IP Dst(h11) Vào cổng 4	Ra ở cổng 2	D_2:s22

Khi gói tin đi đến miền đích D_3 thông qua switch biên s32, Controller D_3 sẽ cài đặt luồng dữ liệu vào flow table của switch biên s32 để điều hướng luồng đi gói tin đến h31 và h11 (bảng 4).

Bảng 4. Luồng đi dữ liệu của các switch biên

Đường đi gói tin	Hành động	Miền:Switch
IP Src(h11)-> IP Dst(h31) Vào cổng 2	Ra ở cổng 5	D_1:s13
IP Src(h31)-> IP Dst(h11) Vào cổng 5	Ra ở cổng 2	D_1:s13
IP Src(h11)-> IP Dst(h31) Vào ở cổng 4	Ra ở cổng 3	D_2:s21
IP Src(h31)-> IP Dst(h11) Vào ở cổng 3	Ra ở cổng 4	D_2:s21
IP Src(h11)-> IP Dst(h31) Vào ở cổng 2	Ra ở cổng 4	D_2:s22
IP Src(h31)-> IP Dst(h11) Vào ở cổng 4	Ra ở cổng 2	D_2:s22
IP Src(h11)-> IP Dst(h31) Vào ở cổng 4	Ra ở cổng 2	D_3:s32
IP Src(h31)-> IP Dst(h11) Vào ở cổng 2	Ra ở cổng 4	D_3:s32

E. Các công cụ

Các công cụ để triển khai giải pháp bao gồm :

- Ryu là một SDN framework mã nguồn mở gồm các thành phần để triển khai Controller cho mạng SDN [10],
- Mininet là một trình giả lập mạng được dùng để tạo ra một mạng gồm các máy chủ ảo, switch ảo và các

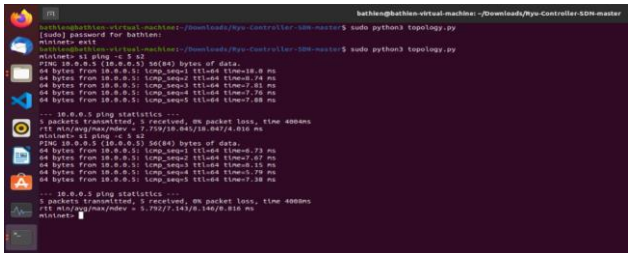
đường dẫn kết nối giữa chúng. Các switch trong mininet có hỗ trợ OpenFlow để định tuyến tùy chỉnh linh hoạt [11]

- Toàn bộ giải pháp sẽ được cài đặt bằng ngôn ngữ lập trình Python [12]. Python là một ngôn ngữ lập trình thông dịch và hướng đối tượng có rất nhiều thư viện hỗ trợ cho việc lập trình mạng nên nếu sử dụng Python để cài đặt và cấu hình cho giải pháp như tạo Topology,… sẽ thuận tiện hơn là sử dụng ngôn ngữ lập trình khác.

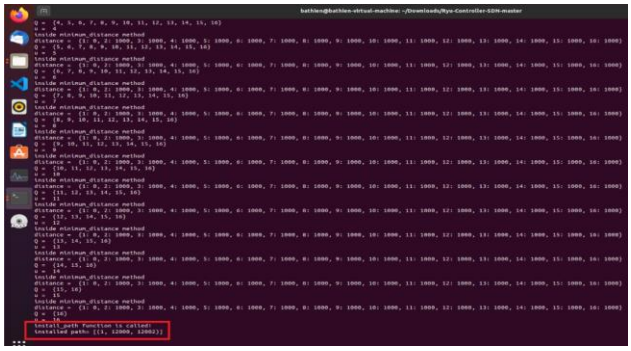
Giải pháp sẽ bao gồm 2 miền SDN được cài đặt trên 2 máy ảo VMware, mỗi máy ảo sẽ là 1 miền SDN.

F. Kết quả triển khai đề xuất

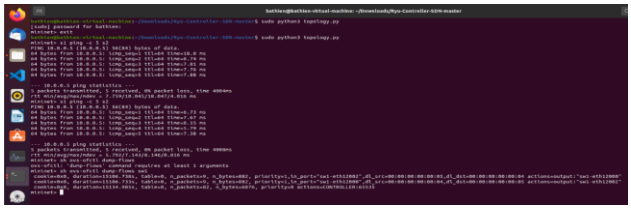
Ta sẽ kiểm tra việc định tuyến trong miền 1 bằng cách ping từ host s1 sang host s2. Sau khi ping xong, ta sẽ kiểm tra Flow Table trên switch 1 là switch kết nối với s1 và s2 (Hình 15-17). Ta có thể thấy Controller có thể xác định được switch 1 nằm trên đường đi ngắn nhất từ s1 sang s2 nên Controller đã cài đặt luồng dữ liệu lên switch 1. Nhìn vào hình 17, ta có thể thấy đường đi của gói tin, địa chỉ MAC của host nguồn và địa chỉ MAC của host đích, gói tin đi vào cổng nào và đi ra cổng nào của switch 1.



Hình 15. Kết quả ping từ host s1 sang host s2 của miền 1



Hình 16. Đường đi ngắn nhất được cài đặt



Hình 17. Flow Table trên switch 1 của miền SDN 1

Trên thực tế, để đánh giá một cách toàn diện giải pháp đề xuất chúng ta cần thực hiện so sánh hiệu suất định tuyến của giải pháp này với các giải pháp khác dựa trên các tham số hiệu năng định tuyến như thời gian truy cập, tỷ lệ lỗi, và độ trễ. Tuy nhiên, việc so sánh này khó thực hiện được với một số lý do sau:

- Các giải pháp đã được nhắc đến (trong phần II) sử dụng các loại SDN controllers khác nhau. Một số nghiên cứu không chỉ ra loại SDN controllers được sử dụng.
- Đa phần các giải pháp đều có sự hỗ trợ rất lớn về phần cứng và cơ sở hạ tầng. Điều này dẫn đến khả năng thực hiện so sánh tính hiệu quả giữa các giải pháp là rất thấp vì khó đáp ứng chuẩn phần cứng chung.

Các nghiên cứu tiếp theo của chúng tôi sẽ tập trung hoàn thiện và khắc phục các nhược điểm kể trên.

IV. KẾT LUẬN

Trong bài báo này, chúng tôi đã thành công trong việc đề xuất và triển khai một giải pháp định tuyến liên miền trong mạng SDN. Bằng cách sử dụng Ryu Controller, Mininet và Python, giải pháp đã được cài đặt và kiểm tra trên môi trường giả lập. Kết quả cho thấy Controller có thể xác định đường đi ngắn nhất giữa hai host trong miền SDN và cài đặt luồng dữ liệu trên các switch nằm trên đường đi đó.

Mặc dù đã đạt được kết quả nhất định, bài báo vẫn còn những hạn chế. Chúng tôi chưa áp dụng IPv6 cho mạng SDN và chưa triển khai cơ chế để xác thực gói tin khi gói tin đi từ miền này sang miền khác. Tuy nhiên, chúng tôi đề xuất một số hướng phát triển trong tương lai như triển khai IPv6 cho mạng SDN, so sánh các thuật toán định tuyến, triển khai cơ chế xác thực gói tin khi gói tin đi từ miền này sang miền khác và triển khai thêm việc cân bằng tải trong mạng.

TÀI LIỆU THAM KHẢO

[1] T. D. Nadeau and K. Gray, SDN: Software Defined Networks, Sebastopol: O'Reilly Media Inc, 2013.

[2] K. Benzekki, A. El Fergougui, & A. Elbelrhiti Elalaoui, "Software-defined networking (SDN): a survey," Security and communication networks, vol. 9, no 18, pp. 5803-5833, 2016.

[3] O. Foundation, "Open Source Cloud Computing Infrastructure - OpenStack," [Online]. Available: https://www.openstack.org/. [Accessed 23 March 2023].

[4] T. A. S. Foundation, "Apache CloudStack: Open Source Cloud Computing," 2020. [Online]. Available: https://cloudstack.apache.org/. [Accessed 23 March 2023].

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69-74, April 2008.

[6] S. H. Yeganeh and Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications," 2012.

[7] A. Tootoonchian and Y. Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow," Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, p. 3, 2010.

[8] J. Stribling, Y. Sovran, I. Zhang, X. Pretzer, J. Li, M. F. Kaashoek and R. Morris, "Flexible, Wide-Area Storage

for Distributed Systems with WheelFS," Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, 2009.
[9] S. Civanlar, E. Lokman, B. Kaytaz and A. M. Tekalp, "Distributed Management of Service-Enabled Flow-Paths Across Multiple SDN Domains," IEEE European Conf. on Networks and Communications (EuCNC), pp. 360-364, 2015.
[10] "Ryu SDN Framework," 2017. [Online]. Available: https://ryu-sdn.org/. [Accessed 19 March 2023].
[11] "Mininet - An Instant Virtual Network on your Laptop (or other PC)," 2022. [Online]. Available: http://mininet.org/. [Accessed 19 March 2023].
[12] P. S. Foundation, "The Python Tutorial," 25 March 2023. [Online]. Available: https://docs.python.org/3/tutorial/. [Accessed 26 March 2023].

RESEARCH AND IMPLEMENT DISTRIBUTED INTER-DOMAIN ROUTING MECHANISM IN SOFTWARE DEFINED NETWORKING (SDN)

Abstract: Over the past two decades, the increase in the size of the Internet has made it more difficult for network administrators to manage and operate the network. Therefore, Software-Defined Networking (SDN) was created to solve the above problems. However, the early SDN architecture had only one single controller, so as the size of the SDN network increased with multiple SDN domains, one controller could not control the entire network. In addition, the routing between domains was also a problem that needed to be solved when the network had many domains. Therefore, the goal of this paper is to propose a distributed inter-domain routing solution, in which each SDN domain will be controlled by a controller and the controller of each domain will exchange information directly with each other via Westbound and Eastbound protocols to share topology and QoS information. Additionally, the solution also includes a mechanism for SDN inter-domain routing based on global topology and QoS information to assist in finding the shortest path for packets within each domain and between domains.

Keywords— Software-Defined Networking; SDN; Controller; Domain; inter-domain routing; Westbound and Eastbound protocols; topology; QoS

Kỹ sư Trần Bá Thiện nhận bằng kỹ sư ngành công nghệ thông tin tại Trường Đại học Bách Khoa - Đại học Đà Nẵng, Việt Nam vào năm 2019. Ông là học viên cao học từ năm 2020 và hiện là kỹ sư phần mềm tại Công ty cổ phần tư vấn DataHouse Asia. Các lĩnh vực hứng thú nghiên cứu hiện nay của ông bao gồm: Định tuyến trong mạng SDN, các giải pháp về cân bằng tải trong mạng SDN, phát hiện mã độc và bảo mật trong mạng SDN

TS. Đinh Trường Duy nhận bằng cử nhân năm 2014 tại Đại học Viễn thông St. Petersburg, Nga. Ông nhận bằng thạc sĩ và tiến sĩ lần lượt vào năm 2016 và 2021 tại Đại học Viễn thông St. Petersburg, Nga. Ông hiện là giảng viên tại Khoa An Toàn Thông Tin, Học viện Công nghệ Bưu chính Viễn thông. Các hướng nghiên cứu hiện nay của ông bao gồm: phát hiện tấn công, xâm nhập, phát hiện mã độc, bảo mật và an toàn các mạng thế hệ mới WSN, FANET, VANET, 5G, IoT, blockchain.

TS. Lê Trần Đức tốt nghiệp từ Đại học Viễn thông St. Petersburg mang tên Giáo sư M.A. Bonch-Bruevich, Liên Bang Nga vào năm 2014 và nhận bằng tiến sĩ tại Đại học Hàng hải và Vận tải Quốc nội Admiral Makarov, St. Petersburg, Liên Bang Nga, vào năm 2018. Hiện tại, TS. Lê Trần Đức đang thực hiện nghiên cứu sau tiến sĩ tại Viện nghiên cứu SME (InRPME) tại Université Québec à Trois-Rivières, Québec, Canada. Lĩnh vực nghiên cứu bao gồm phân tích bảo mật, bảo mật mạng và phân tích mã độc.