RESEARCH ARTICLE

PHÁT HIỆN BẤT THƯỜNG SỚM TRONG MẠNG QUẢN LÝ BẰNG PHẦN MỀM

Trần Mạnh Hà[1], Nguyễn Anh Tuấn[1], Lê Thanh Sơn[2], Phạm Nguyễn Thế Anh[1]

[1] Trường Đại học Ngoại ngữ -Tin học TP.HCM
[2] Trường Đại học Quốc tế - Đại học Quốc gia TP.HCM

hatm@huflit.edu.vn, tuanna@huflit.edu.vn, ltson@hcmiu.edu.vn, theanh.phamnguyen1979@gmail.com

TÓM TẮT— Phát hiện bất thường sớm trong hệ thống mạng là một trong những chức năng quan trọng của quản lý lỗi mạng. Khi hệ thống mạng phát triển lớn về qui mô và quản trị, phức tạp về kiến trúc và chức năng, đồng thời độ co giãn lớn, vấn đề phát hiện bất thường càng trở nên thách thức và khó giải quyết. Nghiên cứu này tập trung giải quyết vấn đề phát hiện bất thường trên mạng quản lý bằng phần mềm (software defined network hay mạng SDN) là một trong những hệ thống mạng mới nổi có đầy đủ đặc tính nêu trên. Giải pháp đề xuất kết hợp giám sát và thu thập dữ liệu sự kiện từ thiết bị chuyển tiếp và áp dụng kĩ thuật máy học vào dữ liệu sự kiện để phát hiện bất thường. Giải pháp được tích hợp vào bộ điều khiển của mạng SDN cho phép thu thập, phân tích sự kiện và cảnh báo bất thường cho người quản trị hệ thống thông qua ứng dụng. Đánh giá giải pháp bao gồm mở rộng chức năng bộ điều khiển sử dụng công cụ mã nguồn mở Ryu và thử nghiệm kĩ thuật phân loại rừng ngẫu nhiên trên tập dữ liệu sự kiện thu thập từ hệ thống Spark thực tế.

Từ khóa— Giám sát và phát hiện bất thường, Kĩ thuật rừng ngẫu nhiên, Bộ điều khiển Ryu, Mạng quản lý bằng phần mềm, Dữ liệu sư kiện Spark

I. GIỚI THIỆU

Mạng quản lý bằng phần mềm (Software Defined Network hay mạng SDN) [1. 2] là một hướng tiếp cận công nghệ mới trong mạng máy tính, tách biệt thành phần điều khiển và dữ liệu của các thiết chuyển tiếp thành tầng điều khiển và tầng dữ liệu nhằm cung cấp và quản lý các dịch vụ mạng hiệu quả đồng thời tăng cường ảo hóa chức năng mạng. Hệ điều hành mạng (NOS) được tích hợp trên tầng điều khiển để giao tiếp với các thiết bị chuyển tiếp ở tầng dữ liệu. Giao thức OpenFlow [3] cho phép hai tầng điều khiển và dữ liệu giao tiếp với nhau. Mạng SDN tích hợp OpenFlow vào các bộ điều khiển và thiết bị chuyển tiếp, như Open vSwitch [4, 5]. Các bộ điều khiển duy trì các thiết bị chuyển tiếp thông qua việc thu thập các thông tin lưu thông trên mạng, các thông báo sự kiện hay các lỗi kết nối mạng.

Mạng SDN là một mô hình hệ thống mạng được nghiên cứu kĩ càng, cung cấp nhiều lợi ích cho các tổ chức và doanh nghiệp. Mạng SDN giảm chi phí hoạt động, tối ưu hóa tài nguyên máy tính và tăng cường hoạt động liên tục của doanh nghiệp một cách hiệu quả bất kể khủng hoảng về nhân lực, dự án và tài chính, đặc biệt là đối với doanh nghiệp chuyên về công nghệ. Tuy nhiên, mạng SDN cũng gây ra một số vấn đề, như là vấn đề quá tải của mạng điều khiển xảy ra khi một số lượng lớn các thiết bị chuyển tiếp kết nối, dẫn đến các bộ điều khiển trở thành điểm yếu tập trung đáp ứng yêu cầu của các thiết bị chuyển tiếp; vấn đề phụ thuộc nặng nề vào các quyết định của bộ điều khiển và ứng dụng đối với sự cố bất thường hay vấn đề thiếu giải pháp thông minh của bộ điều khiển đối với sự kiện cảnh báo và lỗi ảnh hưởng đến hiệu quả vận hành của thiết bị chuyển tiếp và hệ thống mạng, v.v. Nhiều giải pháp đề xuất bao gồm giải pháp điều khiển phân tán bao gồm nhiều bộ điều khiển cùng quản lý thiết bị chuyển tiếp nhưng khó khăn trong việc kiểm soát trạng thái đồng bộ giữa các bộ điều khiển hay giải pháp tăng cường giao thức chuẩn OpenFlow hỗ trợ phát hiện và xử lý lỗi kết hợp chính sách xác định lỗi của bộ điều khiển và ứng dụng nhằm giảm tối thiểu lỗi không xác định.

Tuy nhiên, đối với mạng SDN lớn và phức tạp với số lượng lớn thiết bị chuyển tiếp, bộ điều khiển và ứng dụng trên mỗi tầng, sự cố bất thường, lỗi không xác định hay không dự báo trước, thậm chí tấn công mạng, là vấn đề không thể tránh khỏi và thường ảnh hưởng nghiêm trọng đến dịch vụ mạng. Nghiên cứu này trình bày giải pháp phát hiện bất thường sớm bằng phân tích dữ liệu sự kiện thông báo (gọi tắt là sự kiện) để giải quyết vấn đề nêu trên. Các bộ điều khiển ở tầng điều khiển thu thập sự kiện từ các thiết bị chuyển tiếp ở tầng dữ liệu, áp dụng kĩ thuật phân loại rừng ngẫu nhiên [6] để phân tích dữ liệu sự kiện, phân loại và phát hiện bất thường, đồng thời gửi cảnh báo cho người quản trị hệ thống thông qua ứng dụng ở tầng quản lý. Vì vậy, nghiên cứu này tập trung vào 3 đóng góp chính:

(1) Thiết kế kiến trúc bộ điều khiển mạng SDN tích hợp phát hiện bất thường.

(2) Áp dụng kĩ thuật rừng ngẫu nhiên trên dữ liệu sự kiện để phát hiện bất thường.

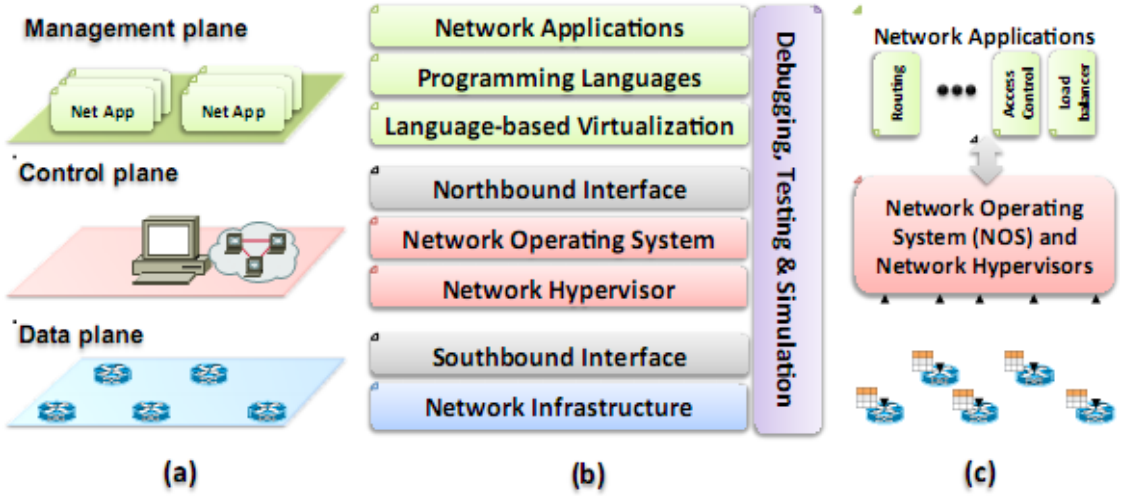
(3) Thử nghiệm và đánh giá giải pháp trên môi trường mạng SDN mô phỏng.

Phần còn lại của bài báo được trình bày như sau: phần tiếp theo giới thiệu mạng SDN, các điểm nổi bật của SDN so với mạng truyền thống, giao thức chuẩn OpenFlow và các giải pháp giám sát, phát hiện và xử lý bất thường hiện có trên mạng này. Phần III trình bày giải pháp phát hiện bất thường sớm trên mạng SDN bao gồm thiết kế

kiến trúc bộ điều khiển với bộ phận thu thập dữ liệu sự kiện và bộ phận phát hiện bất thường dựa trên kĩ thuật rừng ngẫu nhiên. Một số thử nghiệm trong Phần IV báo cáo đánh giá sơ bộ về hiệu suất và khả năng triển khai giải pháp đề xuất trên môi trường mô phỏng mạng SDN trước khi bài báo kết luận trong Phần V.

II. MẠNG SDN VÀ VẤN ĐỀ PHÁT HIỆN BẤT THƯỜNG

Một kiến trúc SDN bao gồm 3 tầng tách biệt như trong Hình 1. Mỗi tầng bao gồm các thành phần chịu trách nhiệm cho hoạt động nhất định [7]. Tầng dữ liệu chịu trách nhiệm thiết lập bộ đệm, lập lịch cho gói dữ liệu, sửa đổi thông tin tiêu đề gói dữ liệu và chuyển tiếp. Bất cứ khi nào một gói dữ liệu đến, tầng dữ liệu sẽ kiểm tra thông tin tiêu đề, so sánh với thông tin đã lưu trong bảng chuyển tiếp để sửa đổi trường tiêu đề nếu cần thiết và chuyển tiếp gói dữ liệu mà không ảnh hưởng đến hai tầng điều khiển và quản lý. Tuy nhiên, không phải tất cả các gói đều có thể được xử lý theo cách đó, đơn giản vì thông tin của họ đôi khi chưa được lưu trữ trong bảng hoặc vì thuộc về một giao thức điều khiển phải được xử lý bởi tầng điều khiển. Tầng điều khiển tham gia vào nhiều hoạt động. Vai trò chính của nó là lưu trữ thông tin mới trong bảng chuyển tiếp để tầng dữ liệu có thể xử lý một lượng lớn lưu lượng mạng mà không phụ thuộc vào tầng điều khiển. Tầng điều khiển có thể phải xử lý một số giao thức điều khiển khác nhau ảnh hưởng đến bảng chuyển tiếp, tùy thuộc vào bộ chuyển đổi mạng và cấu hình của bộ chuyển đổi. Những kiểm soát này các giao thức cũng chịu trách nhiệm quản lý cấu trúc liên kết hoạt động của mạng. Các tác vụ này khá phức tạp nên yêu cầu phải sử dụng các chip xử lý kết hợp với phần mềm đi kèm trong tầng điều khiển. Tầng quản lý là nơi quản trị viên cấu hình và giám sát hoạt động của thiết bị chuyển đổi mạng. Một số hoạt động bao gồm trích xuất thông tin hoặc sửa đổi dữ liệu của tầng dữ liệu và tầng điều khiển. Các quản trị viên sử dụng một các hệ thống quản lý mạng để giao tiếp với tầng quản lý.



Hình 1. Mạng quản lý bằng phần mềm (a) Các tầng, (b) Các thành phần, và (c) Mô hình hệ thống [8]

Mỗi tầng trong cấu trúc mạng SDN sở hữu các thành phần như hạ tầng mạng (Network Interface), giao diện bắc (Northbound Interface), giao diện nam (Southbound Interface), hệ điều hành mạng (NOS), ứng dụng mạng (Network Application), v.v. Mỗi thành phần có một chức năng nhất định, ví dụ, hệ điều hành mạng tương tác với các thiết bị chuyển tiếp trong hạ tầng mạng để điều phối lưu lượng mạng hoặc phát hiện lỗi mạng thông qua giao diện nam trong tầng dữ liệu như trong Hình 1. Do đó, mạng SDN nhằm mục đích phân chia các hoạt động mạng theo cách sau [9]:

(1) Chuyển tiếp, lọc và ưu tiên: Các chức năng chuyển tiếp được giữ nguyên bên trong thiết bị. Các tính năng dựa trên bộ lọc danh sách kiểm soát truy cập (ACL) và ưu tiên lưu lượng cũng vẫn còn trên thiết bị.

(2) Điều khiển: Thiết bị được đơn giản hóa bằng cách loại bỏ hoàn toàn phần điều khiển và chức năng này bây giờ được tích hợp vào một bộ điều khiển trung tâm. Điều này cung cấp cho tầng điều khiển một cái nhìn toàn diện về cấu trúc mạng cũng như cho phép tầng điều khiển quyền quyết định tác vụ chuyển tiếp và định tuyến một cách tốt nhất. Các lập trình viên từ đó cũng có thể can thiệp và làm việc trực tiếp với bộ điều khiển. Bộ điều khiển không được nhúng cũng như không kết hợp chặt chẽ với phần cứng.

(3) Ứng dụng: Trên tầng điều khiển là các ứng dụng mạng cung cấp các chức năng thực hiện cấp cao hơn. Nhiệm vụ của các ứng dụng thường là các tác vụ liên quan đến quản lý, kiểm soát chuyển tiếp và phân phối gói trong mạng một cách hiệu quả nhất.

GIAO THỨC CHUẨN OPENFLOW

OpenFlow là một thành phần rất quan trọng của SDN và có thể coi OpenFlow là khởi nguồn của kiến trúc SDN, được sử dụng cho tác vụ trao đổi thông tin giữa các bộ điều khiển SDN và thiết bị chuyển đổi. OpenFlow ra đời tại Đại học Stanford trong một dự án nghiên cứu, sau đó trở thành dự án phát triển của hai công ty khởi nghiệp Nicira và Big Switch Networks. Nicira sau này được mua lại bởi VMWare và hiện là một phần của dòng sản phẩm mạng ảo hóa của VMWare. Big Switch Networks tham gia một số dự án mã nguồn mở dựa trên OpenFlow như Floodlight Controller, Switch Light OS cho các dòng thiết bị chuyển đổi, và Switch Light vSwitch cho các dòng thiết bị chuyển đổi mạng ảo hóa.

OpenFlow qui định việc giao tiếp giữa các thiết bị chuyển đổi mạng và bộ điều khiển. Khi thiết bị chuyển đổi mạng khởi động, nó sẽ đăng ký với bộ điều khiển. Thông thường, địa chỉ IP của bộ điều khiển sẽ được cung cấp trong quá trình khởi động của thiết bị chuyển đổi mạng hoặc thông qua mạng quản lý riêng, kế đến thiết bị chuyển đổi sẽ thiết lập kênh liên lạc với bộ điều khiển. Kết thúc quá trình khởi động, thiết bị chuyển đổi mạng sẽ giao tiếp với bộ điều khiển thông qua một số loại thông điệp chính sau:

(1) Đối xứng (Symmetric): Các thông điệp này được gửi bởi cả thiết bị chuyển đổi mạng lẫn bộ điều khiển. Thường là các thông điệp chào (hello), các thông điệp dội lại (echo request/reply), hoặc một số loại thông điệp thử nghiệm có thể mở rộng sau này.

(2) Thông điệp bộ điều khiển đến thiết bị chuyển đổi (controller-to-switch): Các thông điệp này được tạo ra bởi bộ điều khiển để quản lý các thiết bị chuyển đổi hoặc truy vấn trạng thái các thiết bị bao gồm các truy vấn tính năng, cấu hình thiết bị, thu thập số liệu để thống kê hay các thông báo tác vụ hoàn thành. Ngoài ra, bộ điều khiển có thể khởi tạo các thông báo trạng thái sửa đổi để thay đổi các luồng và hay đổi nhóm cho các thiết bị chuyển đổi.

(3) Không đồng bộ (Asynchronous): Thiết bị chuyển đổi mạng gửi thông điệp không đồng bộ đến bộ điều khiển khi có gói tin đến, thay đổi trạng thái hoặc lỗi. Khi gói tin đến và thiết bị chuyển đổi không có mục nào trong bảng luồng, thông điệp PakcetIn sẽ được gửi đến bộ điều khiển. Bộ điều khiển sẽ trả lời với một thông điệp packet-out. Thiết bị chuyển đổi mạng cũng sẽ gửi lỗi hay các thay đổi trạng thái dưới dạng các thông điệp bất đồng bộ.

Bảng 1. Các bộ điều khiển SDN mã nguồn mở

Tên sản phẩm	Tác giả	Bản quyền	Ngôn ngữ	Người dùng
NOX	ICSI	GPL	C++	Nghiên cứu, điều hành
POX	ICSI	GPL	Python	Nghiên cứu
Beacon	Stanford University	BSD	Java	Nghiên cứu
Floodlight	Big Switch Networks	Apache	Java	Nghiên cứu, phát triển
Ryu	NTT Communication	Apache2	Java	Nghiên cứu, phát triển
ONOS	The Linux Foundation	Apache2	Java	Nghiên cứu, điều hành
OpenDaylight	OpenDaylight	EPL	_	Phát triển
Jaxon	University of Tsukuba	GPLv3	Java	Nghiên cứu
MulSDN	Kulcloud	GPLv2	C	Điều hành
Trema	NEC	GPLv2	Ruby, C	Nghiên cứu

ỨNG DỤNG VÀ BỘ ĐIỀU KHIỂN SDN MÃ NGUỒN MỞ

Nghiên cứu ban đầu về việc tích hợp SDN và OpenStack với khả năng tương tác [10] đã phát triển giao diện cho phép quản lý các bộ điều khiển SDN sử dụng nền tảng Ryu. Giao diện có khả năng hiển thị hàng triệu máy ảo và thiết bị chuyển mạch. Một số hệ thống giao diện đơn giản cũng được phát triển để hiển thị và tương tác với nhiều tác vụ khác nhau như: hiển thị cấu trúc mạng trên trình duyệt web, trạng thái đường truyên và ứng dụng tường lửa đơn giản. Các ứng dụng này cũng có thể kết hợp với nền tảng FlowVisor [11] để đảm bảo sự tách biệt giữa các ứng dụng. Một số ứng dụng còn cho phép các nhà phát triển tạo ra một kiến trúc mạng như mong muốn kèm theo các ứng dụng phụ trợ. Một trình tối ưu hóa sẽ giúp chuyển đổi các chương trình thành các tác vụ có thể cấu hình trên hệ thống. Gần đây cũng có một số công trình cho phép triển khai các giao diện điều khiển trên điện toán đám mây để cung cấp các ứng dụng mạng thời gian thực, tạo ra một môi trường linh hoạt cho các nhà cung

cấp thiết bị mạng. Một số phiên bản SDN mã nguồn mở và thương mại. Bảng 1 và 2 liệt kê một số bộ điều khiển và ứng dụng SDN phổ biến cùng với các tính năng cơ bản.

Bảng 2. Các ứng dụng SDN mã nguồn mở

Tên sản phẩm	Tác giả	Bản quyền	Ngôn ngữ	Người dùng
Routeflow	CPqD (Brazil)	-	-	Nghiên cứu, phát triển
Quagga	Quagga Routing Project	GPL	C	Nghiên cứu, phát triển
Avior	Marist College	MIT	Java	Nghiên cứu
OSCARS	Energy Services Networks	New BSD	Java	Nghiên cứu
The BIRD	CERN	GPL	-	Nghiên cứu
FlowScale	nCNTRE	Apache2	Java	Nghiên cứu
Frenetic	Princeton University	GPL	Python	Nghiên cứu
FortNOX	SRI International	-	-	Nghiên cứu
FRESCO	SRI International	-	-	Nghiên cứu

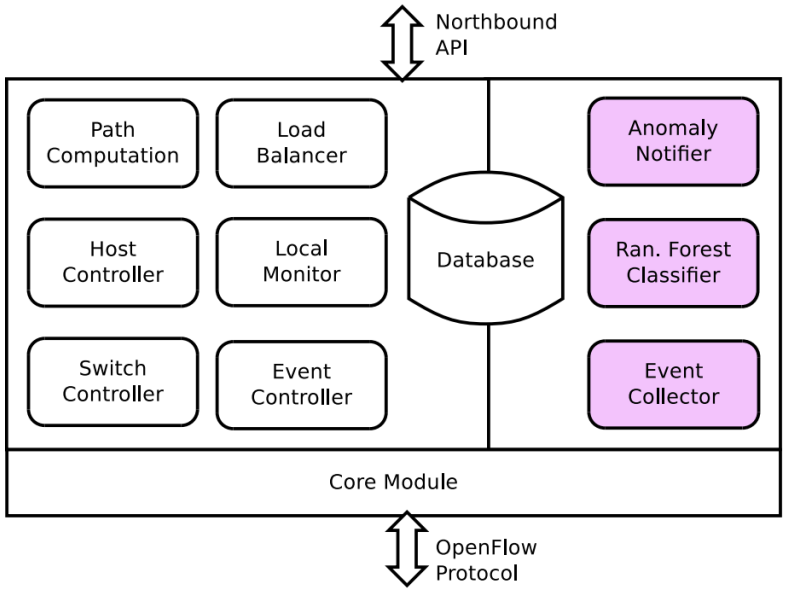
PHÁT HIỆN BẤT THƯỜNG TRÊN MẠNG SDN

Nhiều nghiên cứu và giải pháp đã được đề xuất và thử nghiệm để giải quyết vấn đề phát hiện bất thường hoặc lỗi trên hệ thống mạng sử dụng dữ liệu sự kiện hệ thống. Nghiên cứu của Alspaugh và đồng nghiệp [12] trình bày một số kinh nghiệm phân tích sự kiện bao gồm lọc, tái định dạng và tóm lược dữ liệu sự kiện. Họ sử dụng cách đặc tả trạng thái máy để phân tích tuần tự sự kiện đặc trưng và phân cụm các kiểu chuyển đổi trạng thái nhằm hỗ trợ ra quyết định. Nghiên cứu của Kotani và Okabe [13] đề xuất cơ chế phát hiện lỗi nhanh ở bộ chuyển mạch và bộ điều khiển thông qua OpenFlow. Cơ chế này gửi thông báo kiểm tra trạng thái để phát hiện nhanh nếu thông báo không thể chuyển đến phía bên kia, sau đó bộ điều khiển chia sẻ thông báo nhận được từ bộ chuyển mạch cho tất cả các bộ điều khiển một cách đồng bộ, hoặc bộ điều khiển có liên quan đến đường truyền không khả dụng khi thông báo không thể chuyển. Tác giả của nghiên cứu [14] đề xuất giải pháp phân tích sự kiện để phát hiện bất thường. Giải pháp áp dụng 6 kĩ thuật máy học trên tập dữ liệu chứa 16 triệu sự kiện và 365 ngàn sự cố bất thường được xác định. Tác giả của nghiên cứu [15] cũng cung cấp giải pháp phân cụm để nhận dạng bất thường nghiêm trọng trên hệ thống mạng. Giải pháp áp dụng thuật toán phân cụm theo lớp trên chuỗi sự kiện để xác định mối tương quan giữa các cụm sự kiện với các dịch vụ hệ thống. Dự án LogPAI [16] gần đây xây dựng môi trường trí tuệ nhân tạo mã nguồn mở để hỗ trợ phân tích dữ liệu sự kiện tự động. Dự án thu thập số lượng lớn và đa dạng dữ liệu sự kiện từ hệ thống phân tán, máy chủ và siêu máy tính thực tế trên Loghub [17], thu hút nhiều hoạt động nghiên cứu từ các chuyên gia. Nghiên cứu của Tran và đồng nghiệp [18] đề xuất giải pháp đánh giá tự động mức độ nghiêm trọng của sự kiện cảnh bảo và lỗi. Giải pháp sử dụng kĩ thuật phân loại cây quyết định và rừng ngẫu nhiên trên tập dữ liệu báo cáo lỗi và thuộc tính để xây dựng bộ phân loại. Nghiên cứu của Mohammed và đồng nghiệp [19] đề xuất giải pháp máy học cho vấn đề tự động phát hiện trạng thái hệ thống mạng và định vị lỗi mạng. Giải pháp sử dụng cây quyết định và tối ưu hóa tăng cường để phát hiện các vấn đề về nghẽn hoặc lỗi mạng. Tác giả của nghiên cứu [20] áp dụng trí tuệ nhân tạo để phát hiện sớm các tấn công trong mạng SDN. Nghiên cứu đề xuất giải pháp kiến trúc đa lớp phân cấp để phát hiện và phản ứng chống lại các tấn công khi phát hiện các gói tin độc hại trong mạng SDN. Các thử nghiệm trên môi trường mô phỏng SDN sử dụng một số kĩ thuật cây quyết định, rừng ngẫu nhiên, học sâu, v.v. có hiệu quả cao đối với hình thức tấn công từ chối dịch vụ phân tán.

III. GIẢI PHÁP PHÁT HIỆN BẤT THƯỜNG

A. THIẾT KẾ KIẾN TRÚC BỘ ĐIỀU KHIỂN

Hình 2 minh họa kiến trúc bộ điều khiển kết hợp giải pháp phát hiện bất thường. Bộ điều khiển giữ nguyên các chức năng và bổ sung bộ phận thu thập dữ liệu sự kiện (event collector), bộ phận phân loại rừng ngẫu nhiên (random forest classifier) và bộ phận cảnh báo bất thường (anomaly notifier).



Hình 2. Kiến trúc bộ điều khiển kết hợp giải pháp phát hiện bất thường

Bộ phận thu thập dữ liệu sự kiện kết hợp chặt chẽ với các bộ phận điều khiển khác, đặc biệt là bộ phận điều khiển sự kiện, để lưu trữ sự kiện vào cơ sở dữ liệu để dùng huấn luyện mô hình phân loại, chuyển đến bộ phận phân loại để đánh giá nguy cơ hoặc phát hiện bất thường. Bộ phận phân loại chứa mô hình phân loại dựa trên kĩ thuật máy học, trong nghiên cứu này sử dụng kĩ thuật rừng ngẫu nhiên, và việc huấn luyện mô hình phân loại rừng ngẫu nhiên xảy ra ở tầng quản lý trên các ứng dụng, dữ liệu sự kiện lớn thu thập và máy tính có năng lực tính toán mạnh. Bộ phận phân loại đánh giá các sự kiện và chuyển nguy cơ hoặc bất thường đến bộ phận cảnh báo bất thường. Bộ phận cảnh báo bất thường thông báo cho người quản trị hệ thống thông qua ứng dụng ở tầng quản lý đồng thời lưu trữ sự cố bất thường.

B. KĨ THUẬT RỪNG NGẪU NHIÊN

Kỹ thuật phân loại rừng ngẫu nhiên [6] chia dữ liệu thành các nhóm dựa trên sự đồng thuận của các cây quyết định. Kỹ thuật học có hướng dẫn này sử dụng tập dữ liệu huấn luyện gồm các báo cáo lỗi để tạo ra nhiều cây quyết định tối ưu ở giai đoạn huấn luyện, sau đó khai thác các cây quyết định để đánh giá tập dữ liệu thực tế. Giải thuật tạo cây quyết định từ trên xuống, tức là từ nút gốc đến nút lá, đóng vai trò quan trọng trong kỹ thuật kết hợp dựa vào cây quyết định này. Cho tập dữ liệu Y=[y_1, ...,y_M] gồm M báo cáo lỗi; mỗi báo cáo lỗi được biểu diễn bởi 1 vector y_i=[y_{i1},...,y_{ik}], trong đó k chỉ rõ số lượng thuộc tính trích xuất từ báo cáo lỗi; σ là ngưỡng chọn cây quyết định tối ưu. Kỹ thuật này tạo ra rừng ngẫu nhiên gồm R cây quyết định tối ưu. Thuật toán 1 trình bày các bước để tạo rừng ngẫu nhiên từ các báo cáo lỗi.

Thuật toán bắt đầu với việc chia ngẫu nhiên tập dữ liệu Y thành tập kiểm tra E và tập huấn luyện T theo tỉ lệ 25% và 75% tương ứng (1). Tập huấn luyện T được dùng để tạo cây quyết định (2). Quá trình phát triển cây quyết định bao gồm 3 bước. Bước đầu tiên xác định việc rẻ nhánh tốt nhất đối với từng thuộc tính (3). Bước này sử dụng các luật rẻ nhánh dựa trên tính toán entropy để xác định rẻ nhánh tốt nhất trong tất cả rẻ nhánh có thể của từng thuộc tính, dẫn đến chia thành 2 tập giá trị của thuộc tính. Mỗi rẻ nhánh phụ thuộc vào giá trị của chỉ 1 thuộc tính và rẻ nhánh tốt nhất cực đại hóa tiêu chí rẻ nhánh được định nghĩa. Bước thứ hai xác định rẻ nhánh tốt nhất của nút trong số các rẻ nhánh được xác định từ bước đầu tiên (4). Rẻ nhánh tốt nhất cũng cực đại hóa tiêu chí rẻ nhánh được định nghĩa. Bước cuối cùng rẻ nhánh cho nút sử dụng rẻ nhánh tốt nhất đã xác định từ bước thứ hai (6). Quá trình này lặp lại bước đầu tiên cho đến khi một trong các luật dừng thỏa mãn. Sau khi tạo cây quyết định, giải thuật sử dụng tập kiểm tra E để đánh giá cây và so sánh kết quả đánh giá với σ. Cây quyết định có kết quả đánh giá tốt hơn ngưỡng được thêm vào rừng ngẫu nhiên R (8). Giải thuật tiếp tục chia Y và tạo nhiều cây quyết định tối ưu cho đến khi đủ số lượng (9).

Thuật toán 1: Tạo mô hình phân loại rừng ngẫu nhiên từ dữ liệu báo cáo lỗi
Nhập Tập dữ liệu Y và ngưỡng σ chọn cây quyết định
Xuất Rừng ngẫu nhiên R cây quyết định tối ưu
Bắt đầu

(1) Chia tập dữ liệu Y ngẫu nhiên thành tập kiểm tra E và tập huấn luyện T
(2) Dùng T để tạo cây quyết định
(3) Xác định việc rẻ nhánh tốt nhất đối với từng thuộc tính
(4) Xác định nút đối với rẻ nhánh tốt nhất
(5) Dừng nếu không còn nút được tạo ra
(6) Rẻ nhánh cho nút sử dụng rẻ nhánh tốt nhất đã xác định của nút
(7) Lặp lại bước 3
(8) Đánh giá cây quyết định với E và σ, sau đó cập nhật R
(9) Dừng nếu không còn cây quyết định được tạo ra
(10) Lặp lại bước 1

Kết thúc

C. DỮ LIỆU SỰ KIỆN VÀ THUỘC TÍNH

Một thông báo sự kiện có nhiều thuộc tính: *date* (ngày) và *time* (giờ) theo định dạng yy/MM/dd HH: mm:ss; *severity* (độ ảnh hưởng) mức độ ảnh hưởng đến hệ thống; *component* (thành phần) và *class* (chức năng) tạo ra thông báo; *content* (nội dung) trình bày thông tin chi tiết về sự kiện. Thuộc tính severity nhận các giá trị sau:

(1) FATAL: Lỗi được hiển thị trên bảng điều khiển trạng thái và có thể dừng ứng dụng hay hệ thống.
(2) ERROR: Lỗi thời gian chạy hoặc tình huống không mong đợi được hiển thị trên bảng điều khiển trạng thái và có thể tiếp tục chạy ứng dụng hay hệ thống.
(3) WARN: Tình huống không mong muốn hoặc không mong đợi được hiển thị trên bảng điều khiển trạng thái và có khả năng gây ra các sự cố nguy hiểm
(4) INFO: Thông báo cập nhật về tiến trình ứng dụng hay hệ thống được hiển thị trên bảng điều khiển trạng thái.
(5) DEBUG: Thông tin chi tiết của một sự kiện để gỡ rối ứng dụng hay hệ thống chỉ được ghi vào nhật ký.
(6) TRACE: Thông tin chi tiết hơn DEBUG để giúp gỡ rối ứng dụng hay hệ thống chỉ được ghi vào nhật ký.

Bảng 3. Các thuộc tính trích xuất của sự kiện thông báo

Thuộc tính	Mô tả	Kiểu dữ liệu
datetime	Ngày và giờ tạo sự kiện nhật kí	Thời gian, cơ bản
severity	Mức độ ảnh hưởng của sự kiện nhật kí	Liệt kê, cơ bản
component	Thành phần phần mềm tạo sự kiện nhật kí	Liệt kê, dẫn xuất
class	Chức năng tạo sự kiện nhật kí	Liệt kê, dẫn xuất
keyword	Danh sách từ khóa mô tả sự kiện nhật kí	Văn bản, dẫn xuất
category	Phân loại sự kiện nhật kí	Liệt kê, dẫn xuất
repetition	Thông báo nhắc sự kiện nhật kí	Liệt kê, dẫn xuất

Có nhiều sự kiện lặp lại với tần suất xuất hiện khác nhau. Thuộc tính datetime được hợp nhất lại và thuộc tính repetition được thêm vào để giảm bớt số lượng lớn sự kiện lặp. Thuộc tính repetition nhận giá trị: none (không lặp), intermittent (không thường xuyên), high (thường xuyên). Thuộc tính severity chỉ nhận giá trị: FATAL, ERROR và WARN. Thuộc tính component và class được tách ra từ thành phần hệ thống và chức năng. Thuộc tính keyword chứa 1 tập từ khóa hoặc nhóm từ khóa quan trọng trích xuất từ nội dung thông báo bằng phương pháp tf x idf. Thuộc tính category được xác định dựa trên thuộc tính keyword, ví dụ: Memory, Disk, Cache, IO, Process, v.v. Bảng 3 trình bày danh sách thuộc tính trích xuất từ sự kiện thông báo. Kiểu dữ liệu nguyên thủy giữ nguyên thuộc tính và giá trị từ hệ thống, trong khi kiểu dữ liệu dẫn xuất được tạo ra hoặc thay đổi từ thuộc tính hoặc giá trị ban đầu.

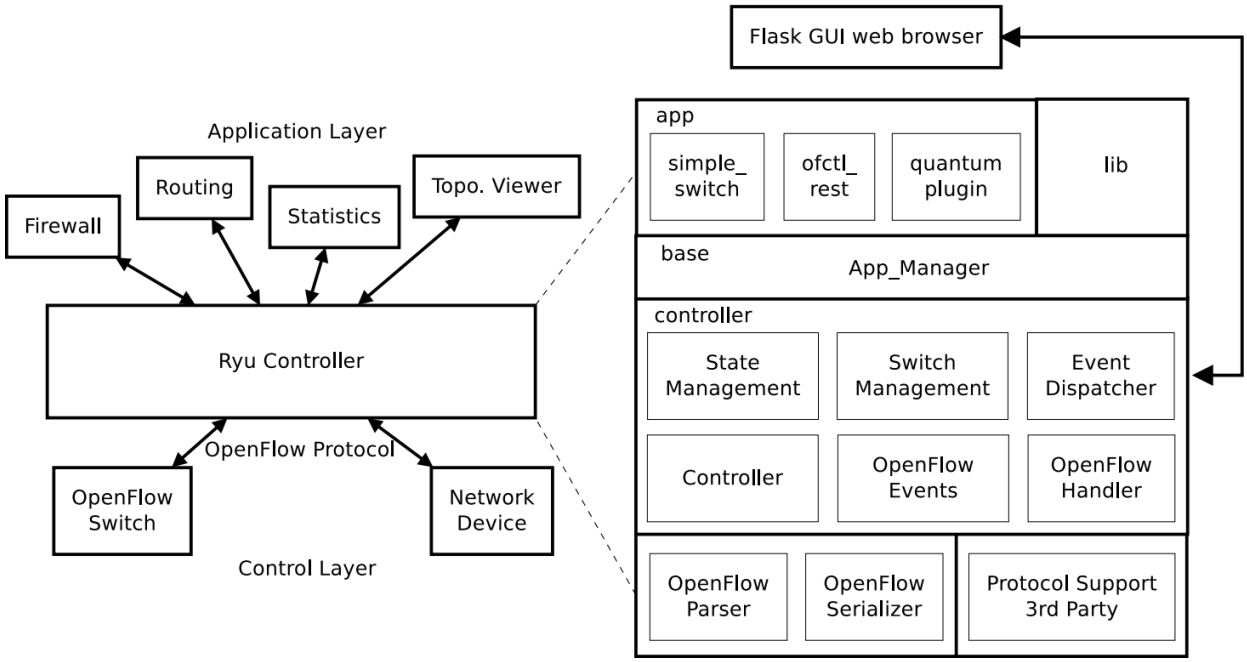
IV. THỬ NGHIỆM VÀ ĐÁNH GIÁ GIẢI PHÁP

A. MÔ PHỎNG MÔI TRƯỜNG THỬ NGHIỆM

Để đáp ứng yêu cầu môi trường thử nghiệm, các chức năng mạng bao gồm thiết bị đầu cuối, kết nối, bộ định tuyến, v.v. được mô phỏng bởi Mininet. Các chức năng SDN được hiện thực sử dụng bộ điều khiển mã nguồn mở Ryu kết hợp thư viện và hệ thống quản lý sự kiện của Python để thu thập dữ liệu sự kiện từ các thiết bị chuyển mạch ảo Open vSwitch

1. PHẦN MỀM RYU

Ryu [21] là một nền tảng của SDN, là thành phần cung cấp nền tảng thực thi cho các bộ điều khiển trong kiến trúc mạng SDN. Đây là một bộ mã nguồn mở, hỗ trợ các hàm API và thư viện cung hỗ trợ tất cả các phiên bản hiện thời của giao thức OpenFlow. Trong ứng dụng của nhóm, Ryu được thử nghiệm với các thiết bị chuyển mạch có hỗ trợ OpenFlow và được chỉnh sửa để hoạt động với các thiết bị Open vSwitch. Ryu có một kiến trúc mạnh mẽ và phức tạp nhưng lại được cộng đồng người dùng hỗ trợ đông đảo, nên rất phù hợp cho các nhà phát triển nghiên cứu và áp dụng. Trong nghiên cứu này, Ryu là một nền tảng rất thích hợp để triển khai cho hệ thống các bộ điều khiển trên mạng SDN



Hình 3. Các thành phần của bộ điều khiển Ryu

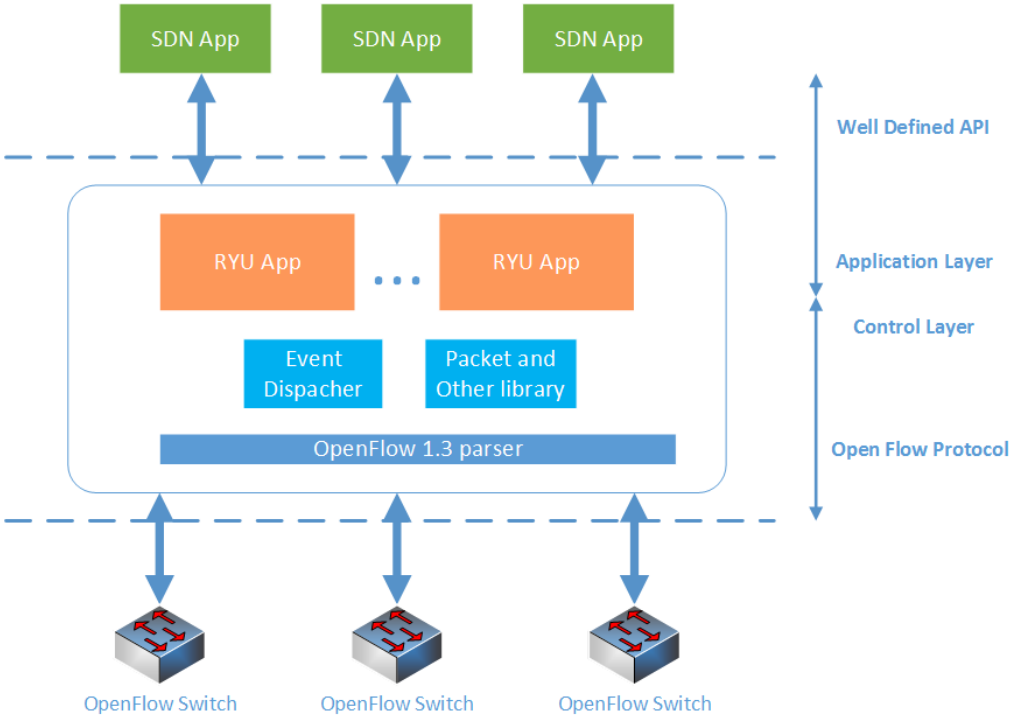
Ryu được xây dựng dựa trên kiến trúc "Sự kiện" và sử dụng "Trình điều khiển sự kiện" để tạo và kiểm soát các sự kiện từ các thông điệp OFP. Ryu sử dụng hai luồng để truyền và nhận các gói đi và đến từ các thiết bị chuyển mạch. Dựa trên thông tin các gói nhận được, luồng nhận sẽ chịu trách nhiệm lưu trữ các gói này trong hàng đợi bằng cách đọc các dữ liệu thô chứa trong thông điệp OFP. Trình điều khiển sẽ gọi ứng dụng để phân tích gói đồng thời tạo ra sự kiện tương ứng. Nền tảng Ryu có sẵn và duy trì các hàng đợi sự kiện, các ứng dụng chạy trên nền Ryu luôn ở chế độ chờ và sẽ được kích hoạt bởi trình điều khiển kiện để thực thi khi có sự kiện thích hợp. Trình điều khiển sự kiện có vai trò rất quan trọng, nó phân tách nguồn sự kiện và gọi các phương thức của Bộ điều phối sự kiện để xây dựng các đối tượng thực thi, từ đó gửi các sự kiện đến hàng đợi sự kiện của các ứng dụng Ryu.

2. RYU REST API KẾT HỢP GIAO DIỆN BẮC (NORTHBOUND INTERFACE)

REST API là một cách tiếp cận để thiết lập giao tiếp sử dụng trong việc phát triển các dịch vụ Web, thường sử dụng giao thức HTTP. REST cung cấp một số lượng nhất định các tác vụ cho tương tác hiệu quả giữa các ứng dụng và người dùng. Tính linh hoạt được đảm bảo bằng cách gán các định danh tài nguyên chung (URLs). API dịch vụ web tuân thủ các ràng buộc kiến trúc REST được gọi là RESTful APIs. Trong trường hợp của Ryu, một REST API thực hiện giao tiếp giữa lớp ứng dụng và lớp điều khiển. Giao diện hàm cung cấp giao tiếp này được biết đến trong SDN là giao diện bắc.

Lớp ControllerBase của Ryu là lớp cơ sở được sử dụng để tạo API do người dùng. Nó xử lý các kết nối HTTP đến thông qua Giao diện máy chủ Web (WSGI). WSGI là một đặc tả cho các giao diện đơn giản giữa các máy chủ web và các ứng dụng web Python. Một lớp mở rộng từ ControllerBase có thể xử lý các yêu cầu HTTP đến từ giao diện WSGI. Các yêu cầu được lọc theo URL và phương thức được yêu cầu. Ryu quyết định phương thức nào được gọi. Để chỉ ra URL và phương thức nào phù hợp, các phương thức lớp phải được mô tả bằng bộ mô tả đường dẫn.

Liên kết của một ứng dụng Ryu với giao diện web được thực hiện bên ngoài lớp RyuApp. Như đã giải thích ở trên, một lớp điều khiển phải được tạo để xử lý các yêu cầu và sau đó phản hồi. Để liên kết bộ điều khiển với một ứng dụng, một đối tượng WSGI được sử dụng. Đối tượng này được tạo bởi Ryu và được lưu trữ. Đối tượng này cho phép đăng ký bộ điều khiển cho API ứng dụng tương ứng. Bộ điều khiển đã đăng ký sẽ nhận được các yêu cầu đến cùng với tham chiếu đến một đối tượng RyuApp.



Hình 4. Mạng SDN trên nền tảng Ryu [21]

3. MẠNG MÔ PHỎNG MININET

Mininet [22] là trình giả lập mạng được viết bằng Python và C, cho phép tạo ra các máy chủ, thiết bị chuyển mạch và các liên kết ảo. Mininet hỗ trợ nhiều thiết bị chuyển mạch dựa trên OpenFlow, mô hình mạng tùy chỉnh đồng thời cung cấp bộ lệnh dựa trên Python để lập trình. Các thành phần ảo dùng trong mô phỏng được tạo nên thông qua việc sử dụng các cấu thành có sẵn của hệ điều hành Linux và các dịch vụ mạng của hệ điều hành. Điều này cho phép triển khai mô hình giả lập lên hệ thống thực tế một cách dễ dàng. Mạng Mininet bao gồm:

(1) Thiết bị đầu cuối tách biệt: cho phép tạo ra các thiết bị đầu cuối (host) với không gian làm việc tách biệt có đầy đủ các cổng kết nối, giao tiếp và bảng định tuyến
(2) Mô phỏng liên kết mạng: Trình quản lý lưu lượng của Linux cho phép cấu hình tốc độ của từng liên kết ảo theo đúng cấu hình mong muốn. Mỗi thiết bị đầu cuối có các giao diện Ethernet ảo riêng.
(3) Mô phỏng thiết bị chuyển mạch: Thiết bị chuyển mạch mặc định của Linux hoặc Open vSwitch chạy trong chế độ nhân được sử dụng. Các bộ chuyển mạch và bộ định tuyến có thể chạy trực tiếp trên nhân Linux hoặc tách biệt hẳn trong mạng ảo đã tạo ra.

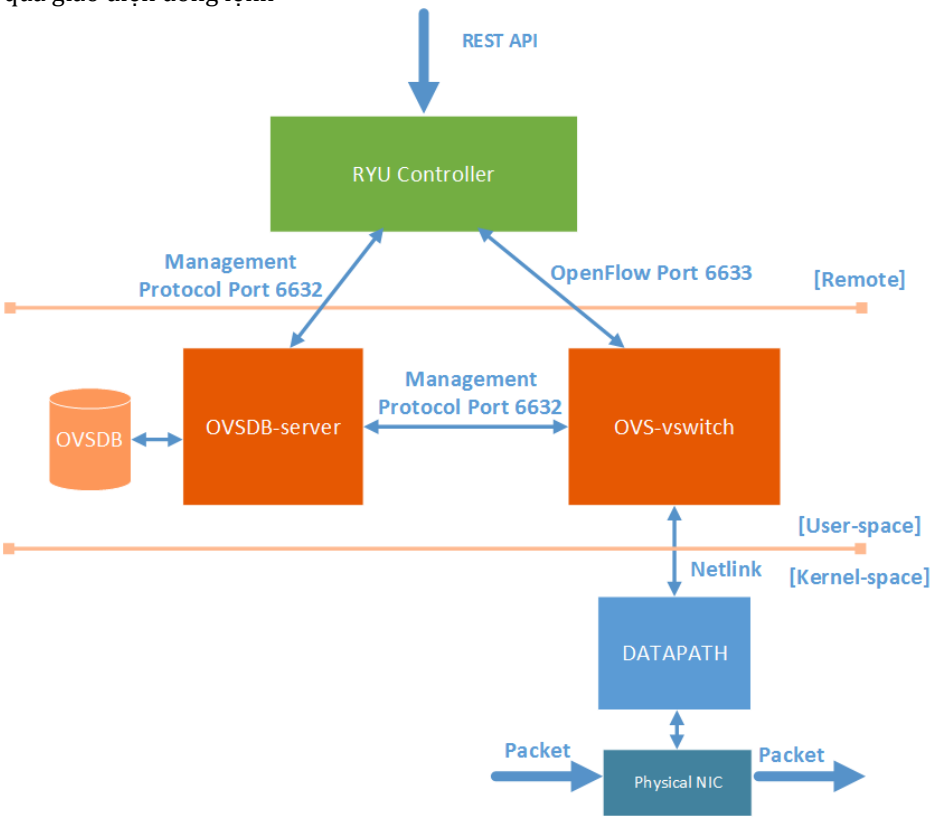
4. THIẾT BỊ OPEN VSWITCH

Open vSwitch (OvS) [4, 5] là một thiết bị chuyển mạch ảo đa lớp mã nguồn mở. Open vSwitch cung cấp nền tảng hỗ trợ các cổng giao tiếp, trình quản lý, các hàm chuyển mạch, tất cả đều có khả năng mở rộng và tùy biến thông qua lập trình. Có ba thành phần chính trong Open vSwitch: máy chủ cơ sở dữ liệu (ovsdb-server), trình nền (ovs-vswitchd), và dữ liệu nhân hệ điều hành:

(1) Bộ điều khiển được trình bày trong trường hợp này là bộ điều khiển Ryu. Tất cả các thành phần của Open vSwitch đều có thể cấu hình từ xa.
(2) ovsdb-server là một máy chủ cơ sở dữ liệu thu gọn thông qua đó ovs-vswitchd truy vấn để có được cấu hình.

(3) ovs-vswitchd là một trình nền thực hiện việc chuyển mạch, kết hợp với dịch vụ của nhân hệ điều hành để điều khiển chuyển mạch dựa trên luồng dữ liệu.

Open vSwitch cung cấp khả năng cấu hình thông qua giao diện dòng lệnh. Thông qua giao diện này, quản trị viên có thể can thiệp trực tiếp dữ liệu các bảng chuyển mạch luồng có trên thiết bị. Dữ liệu có thể được xem, xóa hay thay đổi thông qua giao diện dòng lệnh



Hình 5. Kiến trúc Open vSwitch [4]

B. KẾT QUẢ PHÂN LOẠI

Dữ liệu sự kiện thu thập từ môi trường thử nghiệm mô phỏng thiếu đa dạng và giới hạn sự cố bất thường. Để đánh giá hiệu quả phân loại sự kiện, tập dữ liệu sử dụng được thu thập từ hệ thống mạng thực tế có độ tương tự cao.

Dữ liệu thử nghiệm từ dự án Loghub [17] cung cấp 1 bộ sưu tập lớn và miễn phí dữ liệu sự kiện từ nhiều hệ thống khác nhau. Cụ thể, dữ liệu sự kiện hệ thống Spark gồm các tập tin sự kiện được thu thập từ 32 máy chủ tại một trường đại học. Dữ liệu sự kiện được thu thập ở cấp độ máy chủ mà không chỉnh sửa hay đánh nhãn, và có thể liên quan các sự cố bất thường do tình trạng sửa chữa của 3 máy chủ trong hệ thống. Bảng 4 trình bày thông tin về dữ liệu sự kiện này.

Bảng 4. Thông tin về dữ liệu sự kiện hệ thống Spark

Số lượng tập tin sự kiện	3,852
Kích thước tập tin sự kiện (GB)	2.75
Tổng số sự kiện	27,410,336
Số lượng sự kiện INFO	27,389,482
Số lượng sự kiện WARN	9,595
Số lượng sự kiện ERROR	11,259
Số lượng sự kiện FATAL	0

Phần hiện thực sử dụng ngôn ngữ lập trình *python* và nhiều thư viện mã nguồn mở *sklearn*, *pandas*, *numpy*, ... để lọc và xử lý dữ liệu sự kiện đồng thời phát triển kỹ thuật phân loại rừng ngẫu nhiên. Kỹ thuật này chỉ chấp nhận dữ liệu kiểu số, liệt kê, liên tục, vì vậy, các thuộc tính cần được trích xuất và chuyển đổi. Tuy nhiên, các thông báo sự kiện có các thuộc tính văn bản, như là tên, mô tả, ... thường ẩn chứa thông tin quan trọng và cần được xử lý trước khi áp dụng. Thuật toán 2 phân tích các từ khóa quan trọng dựa trên trọng số từ dữ liệu sự kiện. Giải thuật bắt đầu với việc tải tập từ khóa ban đầu (1), thực hiện một số bước loại bỏ từ khóa ít quan trọng, trùng lặp, dư thừa (2, 3, 4), và tạo ra tập từ khóa chọn lọc. Giải thuật tiếp tục sử dụng kỹ thuật *tf x idf* (term frequency-inverse document frequency) để tính toán trọng số cho tập từ khóa chọn lọc (5) và xuất tập kết quả các từ khóa quan trọng với trọng số cao (6).

Thuật toán 2: Chọn lọc từ khóa quan trọng với trọng số cao
Nhập Tập từ khóa ban đầu (tên, mô tả, ...)
Xuất Tập từ khóa quan trọng với trọng số cao
Bắt đầu
(1) Tải tập từ khóa ban đầu
(2) Loại bỏ từ khóa thừa, lặp và ít quan trọng
(3) Loại bỏ từ đa dạng với gốc từ
(4) Loại bỏ các từ vô nghĩa, kí hiệu đặc biệt bằng biểu thức chính quy
(5) Xử lý *tf x idf* trên tập từ khóa sau khi loại bỏ
(6) Chọn lọc từ khóa quan trọng với trọng số cao
Kết thúc

Sau khi xử lý dữ liệu văn bản, áp dụng thuật toán tạo rừng ngẫu nhiên dựa trên tập dữ liệu huấn luyện với các thuộc tính và mức độ ảnh hưởng: FATAL, ERROR và WARN để tạo bộ phân loại rừng ngẫu nhiên gồm nhiều cây quyết định tối ưu. Bộ phân loại sau đó được dùng để đánh giá trên tập dữ liệu kiểm thử với 2 nhãn: bình thường và bất thường và sử dụng 3 tiêu chuẩn precision (độ chính xác), recall (độ thu hồi) và f1-score (điểm f1) để đánh giá hiệu quả của kỹ thuật phân loại rừng ngẫu nhiên trên tập dữ liệu sự kiện theo định nghĩa sau:

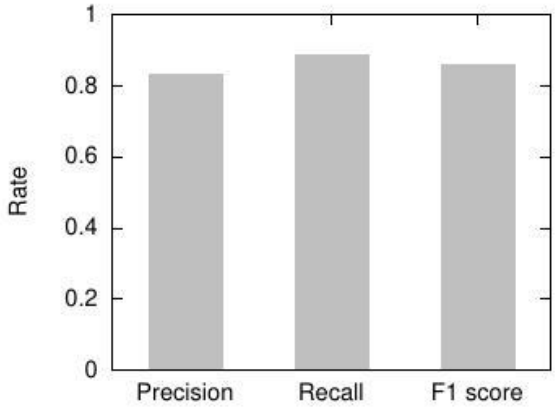
Precision = Số lượng bất thường đúng được phát hiện/Số lượng bất thường đúng hoặc sai được phát hiện

Recall = Số lượng bất thường đúng được phát hiện/Tổng số lượng bất thường được phát hiện

F1score = (2 * Precision * Recall)/(Precision + Recall)

Sự kiện ERROR và một số sự kiện WARN liên quan đến bất thường hệ thống rõ ràng được xác định là bất thường đúng và được dùng để tính toán các tiêu chuẩn. Hình~\ref{rforestres} báo cáo hiệu quả của bộ phân loại rừng ngẫu nhiên trên tập dữ liệu sự kiện với 2 nhãn: bình thường và bất thường.

Tỉ lệ precision và recall của bộ phân loại rừng ngẫu nhiên trên tập sự kiện khá cao ở mức 0.83 và 0.89 tương ứng vì phát hiện đúng hầu hết sự kiện ERROR và FATAL, nhưng nhầm lẫn khi phát hiện sự kiện WARN không thực sự liên quan bất thường hoặc lỗi hệ thống. Bộ phân loại cũng cung cấp một số kết quả đáng chú ý về việc phát hiện đúng một số sự kiện WARN liên quan đến bất thường hoặc lỗi thực sự. Tuy nhiên, tỉ lệ precision bị ảnh hưởng bởi một số sự kiện ERROR chứa thông tin khá chung gây nhầm lẫn cho bộ phân loại. Tỉ lệ f1 score ở mức 0.85 có thể chưa giúp người quản trị hệ thống phát hiện sớm bất thường nhưng cảnh báo nguy cơ và giảm sự kiện thông báo sai trên hệ thống.



Hình 6. Tỉ lệ precision, recall và f1 score của bộ phân loại rừng ngẫu nhiên

V. KẾT LUẬN

Nghiên cứu trong bài báo đề xuất giải pháp phát hiện bất thường sớm trên mạng SDN. Sự cố bất thường, lỗi không xác định hay không dự báo trước, thậm chí tấn công mạng, xảy ra trên tầng dữ liệu chứa số lượng lớn thiết bị chuyển tiếp thường khó tránh khỏi và làm gián đoạn nghiêm trọng dịch vụ mạng. Để giải quyết vấn đề này, các bộ điều khiển ở tầng điều khiển thu thập sự kiện từ các thiết bị chuyển tiếp, áp dụng kĩ thuật phân loại rừng ngẫu nhiên để phân tích và phát hiện sự cố bất thường, từ đó gửi cảnh báo cho người quản trị hệ thống thông qua ứng dụng ở tầng quản lý. Giải pháp đề xuất bao gồm bổ sung thiết kế kiến trúc bộ điều khiển SDN, xây dựng bộ phân loại rừng ngẫu nhiên và thử nghiệm trên môi trường SDN mô phỏng với sự hỗ trợ của công cụ và thư viện mã nguồn mở như: Ryu, Mininet, Open vSwitch, sklearn, pandas, numpy, v.v. Bộ điều khiển Ryu mã nguồn mở phù hợp cho việc nghiên cứu và mở rộng chức năng thử nghiệm đồng thời sử dụng ngôn ngữ python và hệ thống sự kiện python, trong khi Mininet và Open vSwitch mô phỏng môi trường mạng SDN. Tuy nhiên, việc thử nghiệm trên môi trường mô phỏng không thu thập được dữ liệu sự kiện đa dạng với các sự cố bất thường thực tế. Đánh giá hiệu quả bộ phân loại rừng ngẫu nhiên dựa trên tập sự kiện thu thập từ hệ thống Spark thực tế và kết quả ban đầu có thể giúp cảnh báo nguy cơ trên hệ thống mạng SDN với tỉ lệ precision, recall và f1 score lần lượt là 0.83, 0.89 và 0.85.

VI. LỜI CẢM ƠN

Nghiên cứu này được tài trợ bởi Trường Đại học Ngoại ngữ-Tin học TPHCM theo đề tài mã số H2022-02.

VII. TÀI LIỆU THAM KHẢO

[1] N. Feamster, J. Rexford, and E. Zegura. The Road to SDN. Queue–Large-Scale Implementations, 11(12):20:20–20:40, December 2013.
[2] M. Boucadair and C. Jacquenet. Software-Defined Networking: A Perspective from within a Service Provider Environment. RFC 7149, March 2014.
[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. SIGCOMM Comput. Commun. Rev., 38(2):69–74, March 2008.
[4] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado. The design and implementation of open vswitch. In Proc. 12th USENIX Conference on Networked Systems Design and Implementation (NSDI'15), pages 117–130, Berkeley, CA, USA, 2015. USENIX Association.
[5] Open vSwitch Project. Last access in July 2022 at https://www.openvswitch.org/, 2016.
[6] L. Breiman. Random Forests. Machine Learning, 45(1):5–32, 2001.
[7] P. Goransson, C. Black, and T. Culver. Software Defined Networks: A Comprehensive Approach. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2016.
[8] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-Defined Networking: A Comprehensive Survey. Proc. IEEE, 103(1):14–76, Jan 2015.
[9] R. Narayanan, S. Kotha, G. Lin, A. Khan, S. Rizvi, W. Javed, H. Khan, and S. A. Khayam. Macroflows and microflows: Enabling rapid network innovation through a split SDN data plane. In European Workshop on Software Defined Networking (EWSDN 2012), pages 79–84, 2012.
[10] S. C. Chen and R. H. Hwang. A Scalable Integrated SDN and OpenStack Management System. In 2016 IEEE International Conference on Computer and Information Technology (CIT'16), pages 532–537, 2016.
[11] R. Sherwood, G. Gibb, K. Yap, M. Casado, N. Mckeown, and G. Parulkar. Flowvisor: A network virtualization layer. Technical report, 2009.
[12] S. Alspaugh, B. Chen, J. Lin, A. Ganapathi, M. A. Hearst, and R. Katz. Analyzing log analysis: An empirical study of user log mining. In Proceedings of the 28th USENIX Conference on Large Installation System Administration, LISA'14, pages 53–68, USA, 2014. USENIX Association.
[13] A. Kotani and Y. Okabe. Fast Failure Detection of OpenFlow Channels. In Proceedings of the Asian Internet Engineering Conference, AINTEC '15, pages 32–39, New York, NY, USA, 2015. ACM.
[14] S. He, J. Zhu, P. He, and M. R. Lyu. Experience report: System log analysis for anomaly detection. In Proceedings of the 27th IEEE International Symposium on Software Reliability Engineering (ISSRE'16), pages 207–218. IEEE, 2016.
[15] S. He, Q. Lin, J. G. Lou, H. Zhang, M. R. Lyu, and D. Zhang. Identifying impactful service system problems via log analysis. In Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE'18, pages 60–70, New York, NY, USA, 2018. ACM.

[16] M. R. Lyu, J. Zhu, P. He, S. He, and J. Liu. The logpai project. http://www.logpai.com/, 2019.
[17] S. He, J. Zhu, P. He, and M. R. Lyu. Loghub: A large collection of system log datasets towards automated log analytics. ArXiv, abs/2008.06448, 2020.
[18] H. M. Tran, S. T. Le, V. S. Nguyen, and P. T. Ho. An Analysis of Software Bug Reports Using Machine Learning Techniques. SN Comput. Sci., 1(1):4:1–4:11, 2020.
[19] A. R. Mohammed, S. A. Mohammed, D. Côté, and S. Shirmohammadi. Machine Learning-Based Network Status Detection and Fault Localization. IEEE Transactions on Instrumentation and Measurement, 70:1–10, 2021.
[20] H. Chuang, F. Liu, and C. Tsai. Early Detection of Abnormal Attacks in Software-Defined Networking Using Machine Learning Approaches. Symmetry, 14(6), 2022.
[21] Ryu Component-based Software Defined Networking Framework. Last access in July 2022 at https://github.com/faucetsdn/ryu, 2017.
[22] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In Proc. 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-IX), pages 19:1–19:6, New York, NY, USA, 2010. ACM

EARLY FAULT DETECTION IN SOFTWARE DEFINED NETWORKS

Tran Manh Ha, Nguyen Anh Tuan, Le Thanh Son, Pham Nguyen The Anh

ABSTRACT— Detecting faults early in network and communication systems is one of the important functionalities of fault management. As these systems become large in size and scalability, complex in architecture and design, and dynamic in operation and control, detecting faults become even more difficult and challenging. This study presents a method for solving this problem in emerging and promising software defined networks (SDN) with the above characteristics. This method collects event log data from switches and uses machine learning techniques to detect faults. The method is associated with the SDN controller to monitor, analyze and notify faults to the system operator through applications. Evaluating the method includes the extensibility of the controller using the Ryu open-source tool and several experiments on the Spark event log dataset using the Random Forest technique.

Tran Manh Ha is associate professor of computer science at HCMC University of Foreign Languages–Information Technology. He obtained his master degree in computer science in 2004 from University of Birmingham, United Kingdom and his doctoral degree in computer science in 2009 from Jacobs University Bremen, Germany. His research interests include communication networks, distributed systems, network management, information retrieval and machine learning.

Nguyen Anh Tuan is senior lecturer of computer science at HCMC University of Foreign Languages–Information Technology. He obtained his master degree in computer science in 2004 from University of Natural Sciences–HCMC Vietnam National University and his doctoral degree in computer science in 2012 from La Trobe University, Australia. His research interests include mobile computing, context-aware computing and information security.

Le Thanh Son received his master degree in computer science in 2006 from Korea Advanced Institute of Science and Technology (KAIST), Korea. He is working as senior lecturer at School of Computer Science and Engineering, International University–HCMC Vietnam National University. His current research interests focus on data science, web technology, information retrieval, mobile computing and computer networks.

Pham Nguyen The Anh is master student of information technology at HCMC University of Foreign Languages–Information Technology. He obtained his bachelor of information technology in 2001 from University of Technology- HCMC Vietnam National University. He currently works for HCMC University of Food Industry. His research interests include network security, network management and fault data analytics.