**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN, ĐHQG-HCM**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**



**BÁO CÁO ĐỒ ÁN MÔN HỌC**

ĐỀ TÀI: TRIỂN KHAI MẠNG SDN VỚI RYU

**Môn học:** NT132.P12.ANTT – QUẢN TRỊ MẠNG VÀ HỆ THỐNG

**Giảng viên hướng dẫn: ĐỖ HOÀNG HIỂN**

**Thực hiện bởi nhóm 15, bao gồm:**

1. Lê Triệu Phú 22521100 Trưởng nhóm
2. Trần Thiện Mạnh 22520853 Thành viên
3. Bùi Châu Thống 22521428 Thành viên

**Thời gian thực hiện:** 01/10/2024 – 30/11/2024

# MỤC LỤC

[MỤC LỤC 2](#_Toc186327022)

[LỜI NÓI ĐẦU 5](#_Toc186327023)

[CHƯƠNG I. GIỚI THIỆU 6](#_Toc186327024)

[1.1. Nguồn gốc của SDN và Ryu 6](#_Toc186327025)

[1.2. Sự phát triển 7](#_Toc186327026)

[Chương II. LÝ THUYẾT VỀ MẠNG SDN 8](#_Toc186327027)

[2.1. Khái niệm và kiến trúc của SDN 8](#_Toc186327028)

[2.1.1. Khái niệm 8](#_Toc186327029)

[2.1.2. Kiến trúc 8](#_Toc186327030)

[2.2. Ưu điểm của SDN 10](#_Toc186327031)

[2.2.1. Tổng quan ưu điểm 10](#_Toc186327032)

[2.2.2. SDN và mạng truyền thống 10](#_Toc186327033)

[2.3. Giao thức OpenFlow 11](#_Toc186327034)

[2.3.1. Định nghĩa 11](#_Toc186327035)

[2.3.2. Kiến trúc 12](#_Toc186327036)

[2.3.3. Nguyên lý hoạt động của OpenFlow: 12](#_Toc186327037)

[2.4. Ứng dụng của SDN 13](#_Toc186327038)

[CHƯƠNG III.CÁC CÔNG CỤ HỖ TRỢ TRIỂN KHAI SDN 14](#_Toc186327039)

[3.1. Mininet 14](#_Toc186327040)

[3.1.1. Cách hoạt động 14](#_Toc186327041)

[3.1.2. Ứng dụng của Mininet 14](#_Toc186327042)

[3.2. Ryu – Bộ điều khiển SDN mã nguồn mở 14](#_Toc186327043)

[3.2.1. Định nghĩa 14](#_Toc186327044)

[3.2.2. Kiến trúc của RYU 15](#_Toc186327045)

[3.2.3. Ứng dụng thực tiễn của RYU 15](#_Toc186327046)

[3.2.4. Giao thức OpenFlow trong Ryu 16](#_Toc186327047)

[3.3. Open vSwitch(OVS) 16](#_Toc186327048)

[CHƯƠNG IV. MỘT VÀI ỨNG DỤNG CỦA SDN VÀ RYU 17](#_Toc186327049)

[4.1. Quản lý mạng trung tâm dữ liệu (Data Center Network Management) 17](#_Toc186327050)

[4.2. Mạng doanh nghiệp (Enterprise Networks) 18](#_Toc186327051)

[4.3. Mạng viễn thông (Telecommunication Networks) 18](#_Toc186327052)

[4.4. Internet of Things (IoT) 18](#_Toc186327053)

[CHƯƠNG V. KẾT LUẬN 19](#_Toc186327054)

[CHƯƠNG VI. KỊCH BẢN TRIỂN KHAI 19](#_Toc186327055)

[6.1. Kịch bản I: 19](#_Toc186327056)

[6.2. Kịch bản II: 20](#_Toc186327057)

[6.3. Kịch bản III: 21](#_Toc186327058)

[Chương VII. HƯỚNG DẪN SETUP 23](#_Toc186327059)

[7.1. Cài đặt mininet: 23](#_Toc186327060)

[7.2. Cài đặt Ryu: 23](#_Toc186327061)

[CHƯƠNG VIII. CÁC BƯỚC THỰC HIỆN 23](#_Toc186327062)

[8.0. Tổng quát 23](#_Toc186327063)

[8.1. Kịch bản I: 28](#_Toc186327064)

[8.1.1. Run RYU 28](#_Toc186327065)

[8.1.2. Run topology 1 29](#_Toc186327066)

[8.1.3. POST ip getway 29](#_Toc186327067)

[8.1.4. Check topology 29](#_Toc186327068)

[8.2. Kịch bản II: 29](#_Toc186327069)

[8.2.1. Run RYU 29](#_Toc186327070)

[8.2.2. Run topology 2 29](#_Toc186327071)

[8.2.3. POST ip getway 29](#_Toc186327072)

[8.2.4. Check topology 29](#_Toc186327073)

[8.3. Kịch bản III: 32](#_Toc186327074)

[8.3.1. Run RYU 32](#_Toc186327075)

[8.3.2. Run topology 3 32](#_Toc186327076)

[8.3.3. Check topology 32](#_Toc186327077)

[CHƯƠNG IX. PHÂN TÍCH FLOW KỊCH BẢN 32](#_Toc186327078)

[9.1. Kịch bản I 32](#_Toc186327079)

[9.2. Kịch bản II 37](#_Toc186327080)

[9.3. Kịch bản III 42](#_Toc186327081)

[9.3.1 Bảng flow 1 44](#_Toc186327082)

[9.3.2. Bảng flow 2 50](#_Toc186327083)

[KẾT QUẢ THỰC THI 50](#_Toc186327084)

[NGUỒN THAM KHẢO 51](#_Toc186327085)

# LỜI NÓI ĐẦU

Báo cáo này được thực hiện trong khuôn khổ môn học Quản trị mạng và hệ thống, nhằm tổng hợp và phân tích những kiến thức đã học, đồng thời ứng dụng vào thực tiễn qua một dự án cụ thể. Nội dung báo cáo sẽ tập trung vào chủ đề triển khai Mạng SDN cùng các công cụ hỗ trợ.

Thông qua báo cáo này, nhóm chúng em hy vọng sẽ làm rõ các khía cạnh lý thuyết và thực hành đã triển khai, đánh giá hiệu quả của các giải pháp và đề xuất các cải tiến cho hệ thống. Bên cạnh đó, đây cũng là cơ hội để nhóm rèn luyện kỹ năng làm việc nhóm, giải quyết vấn đề và tư duy hệ thống trong lĩnh vực quản trị mạng.

Chúng em xin chân thành cảm ơn thầy/cô đã tận tình giảng dạy và hướng dẫn, tạo điều kiện cho chúng em thực hiện báo cáo này. Chúng em rất mong nhận được sự góp ý để hoàn thiện hơn về kiến thức và kỹ năng trong lĩnh vực này.

# CHƯƠNG I. GIỚI THIỆU

Mạng điều khiển bằng phần mềm (SDN - Software-Defined Networking) là một công nghệ mạng mới giúp tách biệt lớp điều khiển khỏi lớp dữ liệu. SDN cho phép quản trị viên mạng linh hoạt trong việc cấu hình và quản lý mạng thông qua phần mềm, thay vì phải can thiệp vào phần cứng như các mạng truyền thống. Điều này đặc biệt quan trọng trong việc cải thiện hiệu suất, bảo mật và khả năng mở rộng mạng.



## 1.1. Nguồn gốc của SDN và Ryu

* **SDN:**
* **Khái niệm ban đầu**: SDN bắt nguồn từ các nghiên cứu tại **Đại học Stanford** và **Đại học UC Berkeley** vào khoảng **năm 2005** trong dự án **Ethane**, với mục tiêu tạo ra một hệ thống mạng đơn giản và có thể quản lý tập trung.
* **Cột mốc quan trọng**: **OpenFlow**, một giao thức giúp điều khiển mạng qua phần mềm, được giới thiệu vào **năm 2008** bởi nhóm nghiên cứu tại Đại học Stanford. OpenFlow được coi là nền tảng đầu tiên của SDN.
* **Ryu:**
* Ryu là một trong những framework SDN mã nguồn mở phổ biến nhất, được phát triển bởi NTT Communications từ năm 2012. Với sự hỗ trợ mạnh mẽ cho OpenFlow và các giao thức SDN khác, Ryu trở thành công cụ mạnh mẽ để nghiên cứu và triển khai mạng SDN hiện đại.



## 1.2. Sự phát triển

* **2008**:
* **OpenFlow** được công bố, tạo tiền đề cho SDN.
* OpenFlow tách biệt **Control Plane** (quản lý) và **Data Plane** (truyền dữ liệu), cho phép quản lý mạng tập trung qua các bộ điều khiển (controllers).
* **2011**:
* Tổ chức phi lợi nhuận **Open Networking Foundation (ONF)** được thành lập để thúc đẩy và chuẩn hóa SDN.
* Các thành viên sáng lập gồm các công ty lớn như **Google, Microsoft, Facebook, Cisco, Juniper, HP**, và **IBM**.
* **2012-2014**:
* Các công ty công nghệ lớn bắt đầu triển khai SDN trong các trung tâm dữ liệu:
* **Google** sử dụng SDN để tối ưu hóa lưu lượng mạng giữa các trung tâm dữ liệu.
* **VMware** phát triển **NSX**, một nền tảng SDN tập trung vào ảo hóa mạng.
* **Cisco** phát hành giải pháp **Application Centric Infrastructure (ACI)**.
* **2015 trở đi**:
* SDN mở rộng từ các trung tâm dữ liệu sang mạng diện rộng (WAN) và mạng doanh nghiệp.
* Các giao thức và kiến trúc mới ra đời như **SD-WAN**, giúp quản lý mạng diện rộng dựa trên phần mềm.
* SDN ra đời từ năm 2008 với giao thức OpenFlow, được thúc đẩy bởi các tổ chức như Open Networking Foundation (ONF). Qua thời gian, nó phát triển mạnh mẽ nhờ các công ty lớn như Google, Cisco, VMware và Juniper, trở thành một giải pháp mạng hiện đại cho Data Center, WAN, và mạng doanh nghiệp.

# Chương II. LÝ THUYẾT VỀ MẠNG SDN

## 2.1. Khái niệm và kiến trúc của SDN

### 2.1.1. Khái niệm

* SDN (Software-Defined Networking) là một kiến trúc có khả năng trừu tượng hóa các layer khác nhau của mạng, giúp mạng trở nên linh hoạt hơn. Mục tiêu chính của SDN là cải thiện khả năng kiểm soát mạng bằng cách cho phép các doanh nghiệp và nhà cung cấp dịch vụ đáp ứng nhanh chóng khi các nhu cầu kinh doanh thay đổi.

### 2.1.2. Kiến trúc

* SDN là một mô hình mạng tách biệt rõ ràng hai chức năng chính: **lớp điều khiển** (Control Plane) và **lớp dữ liệu** (Data Plane). Kiến trúc mạng SDN bao gồm ba thành phần chính:
* **Lớp ứng dụng (Application Layer)**: Bao gồm các ứng dụng SDN có thể được phát triển để kiểm soát và tối ưu hóa mạng, ví dụ như điều khiển lưu lượng, bảo mật, và quản lý chất lượng dịch vụ.
* **Lớp điều khiển (Control Layer)**: Bộ điều khiển SDN như ONOS sẽ điều phối và quản lý toàn bộ hoạt động của mạng, ra quyết định dựa trên thông tin từ các thiết bị mạng.
* **Lớp hạ tầng (Infrastructure Layer)**: Bao gồm các thiết bị phần cứng như switch và router, nhưng không cần phải tự quyết định về điều phối lưu lượng, mọi việc đều được quyết định bởi lớp điều khiển.

A diagram of a network application

Description automatically generated

* **Ba mặt phẳng trong kiến trúc SDN**
* **Mặt phẳng quản lý (Management Plane)**:
  + Là lớp trên cùng, chứa các ứng dụng mạng và công cụ hỗ trợ.
  + Chịu trách nhiệm ra quyết định trên toàn mạng, tối ưu hóa, và thực thi các chính sách.
  + Ví dụ: Các ứng dụng mạng như định tuyến (routing), cân bằng tải (load balancing), và kiểm soát truy cập (access control).
* **Mặt phẳng điều khiển (Control Plane)**:
  + Là cầu nối giữa mặt phẳng quản lý và mặt phẳng dữ liệu.
  + Bao gồm Hệ điều hành mạng (Network Operating System - NOS) và các hypervisor mạng.
  + Chức năng: Xử lý luồng dữ liệu, cung cấp giao diện phía bắc (Northbound Interface) để giao tiếp với mặt phẳng quản lý và giao diện phía nam (Southbound Interface) để kết nối với cơ sở hạ tầng mạng.
* **Mặt phẳng dữ liệu (Data Plane)**:
  + Là lớp dưới cùng, chịu trách nhiệm xử lý trực tiếp các gói dữ liệu.
  + Bao gồm cơ sở hạ tầng mạng như bộ định tuyến (router), switch và thiết bị phần cứng khác.

A diagram of a network

Description automatically generated

* + - **Mô hình hệ thống tổng quan:**
* Các ứng dụng mạng (Network Applications) như định tuyến, cân bằng tải, kiểm soát truy cập được triển khai trong mặt phẳng quản lý.
* Các ứng dụng này tương tác với NOS và hypervisor ở mặt phẳng điều khiển để đưa ra chỉ dẫn cho cơ sở hạ tầng mạng ở mặt phẳng dữ liệu.
* Việc mô phỏng, kiểm thử và gỡ lỗi giúp tối ưu hóa và cải thiện hiệu suất mạng trong hệ thống tổng thể.

## 2.2. Ưu điểm của SDN

### 2.2.1. Tổng quan ưu điểm

* **Linh hoạt**: Quản trị viên mạng có thể cấu hình mạng theo nhu cầu mà không cần phải thay đổi phần cứng.
* **Tiết kiệm chi phí**: SDN sử dụng các thiết bị phần cứng tiêu chuẩn (commodity hardware), giảm sự phụ thuộc vào thiết bị chuyên dụng đắt tiền.
* **Tăng cường bảo mật**: Các chính sách bảo mật được triển khai tập trung qua bộ điều khiển SDN, giúp giám sát và quản lý bảo mật dễ dàng hơn.
* **Khả năng mở rộng**: SDN có thể mở rộng linh hoạt theo nhu cầu của các tổ chức và doanh nghiệp mà không cần thay đổi cấu trúc hạ tầng.

### 2.2.2. SDN và mạng truyền thống

|  |  |  |
| --- | --- | --- |
|  | Mạng truyền thống | Mạng SDN |
| Kiến trúc | Control Plane và Data Plane tích hợp trên từng thiết bị | Phân tách Control Plane và Data Plane, điều khiển tập trung qua SDN Controller. |
| Quản lý | Quản lý thủ công từng thiết bị, cấu hình cục bộ | Quản lý tập trung qua SDN Controller, hỗ trợ tự động hóa và API để lập trình mạng |
| Khả năng mở rộng | Hạn chế, tăng thiết bị làm tăng độ phức tạp | Dễ mở rộng, thêm thiết bị mà không làm răng độ phức tạp, dễ quản lý |
| Hiệu suất | Phụ thuộc vào từng thiết bị, dễ gặp hạn chế khi xử lý lưu lượng lớn | Tối ưu hóa hiệu suất nhờ kiểm soát và quản lý tập trung |
| Khả năng thích nghi | Khó thay đổi, phù thuộc vào phần cứng của nhà sản xuất | Dễ dàng thay đổi và nâng cấp qua phần mềm, hỗ trợ đổi mới nhanh |
| Ứng dụng | Phù hợp với hệ thống mạng ổn định, ít thay đổi, môi trường nhỏ | Thích hợp cho mạng lớn, trung tâm dữ liệu, hệ thống cloud, IoT và các ứng dụng hiện đại |
| Chi Phí | Chi phí ban đầu nhỏ, chi phí bảo hành và vận hành cao do cần nhân lực quản lý thủ công | Chi phí ban đầu cao, nhưng tiết kiệm chi phí vận hành nhờ tự động hóa và tối ưu hơn so với mạng truyền thống |
| Khả năng tự động hóa | Hạn chế, yêu cầu can thiệp thủ công | Cao, hỗ trợ tự động hóa cấu hình, kiểm soát và giám sát |
| Bảo mật | Quản lý bảo mật phân tán, dễ bỏ sót điểm yếu | Bảo mật tập trung, đẽ dàng phát hiện và xử lý các mối đe dọa |

## 2.3. Giao thức OpenFlow

### 2.3.1. Định nghĩa

* **OpenFlow** là một trong những giao thức quan trọng nhất trong mô hình mạng SDN (Software-Defined Networking). Nó đóng vai trò là cầu nối giữa **lớp điều khiển** (Control Plane) và **lớp dữ liệu** (Data Plane) trong kiến trúc SDN. Cụ thể, OpenFlow cho phép bộ điều khiển SDN giao tiếp trực tiếp với các thiết bị chuyển mạch (switch) hoặc các thiết bị mạng khác ở lớp hạ tầng (Infrastructure Layer) bằng cách ra quyết định về việc xử lý các gói tin.

### 2.3.2. Kiến trúc

* Một thiết bị OpenFlow bao gồm ít nhất 3 thành phần: Bảng luồng (Flow table), Kênh an toàn (Secure Chanel) và Giao thức Openflow (OpenFlow Protocol).
* Flow Table: một liên kết hành động với mỗi luồng, giúp thiết bị xử lý các luồng.
* Secure Channel: kênh kết nối thiết bị tới bộ điều khiển (controller), cho phép các lệnh và các gói tin được gửi giữa controller và thiết bị.
* OpenFlow Protocol: giao thức cung cấp phương thức tiêu chuẩn mở cho một controller truyền thông với thiết bị.

### 2.3.3. Nguyên lý hoạt động của OpenFlow:

* **Bộ điều khiển SDN**: Khi một gói tin đến một switch, nếu switch không có quy tắc để xử lý gói tin đó, nó sẽ gửi gói tin này lên bộ điều khiển SDN qua giao thức OpenFlow.
* **Ra quyết định**: Bộ điều khiển SDN sau đó sẽ quyết định cách xử lý gói tin (ví dụ: gửi tiếp, chặn, hoặc điều chỉnh luồng), và gửi quy tắc trở lại switch. Các quy tắc này sẽ được lưu lại trong bảng luồng (flow table) của switch.
* **Switch**: Sau khi nhận được các quy tắc từ bộ điều khiển, switch sẽ áp dụng các quy tắc này để xử lý các gói tin tương lai.
* **Bộ các phần mềm và phần cứng hỗ trợ SDN và OverFlow:**

## 2.4. Ứng dụng của SDN

A diagram of software defined networking

Description automatically generated

* DevOps – phương pháp tiếp cận dựa trên SDN có thể giúp tự động hóa các bản cập nhật và triển khai ứng dụng, từ đó tạo điều kiện thuận lợi cho DevOps. Cụ thể, SDN giúp tự động hóa các thành phần có sở hạ tầng khi các ứng dụng và nền tảng của DevOps được triển khai.
* Campus network (mạng campus) – Các mạng campus thường khó để quản lý, đặc biệt là khi đang có nhu cầu về việc thống nhất mạng WiFi và Ethernet. SDN controller có thể giúp cung cấp khả năng quản lý tập trung và tự động hóa cho mạng campus. Từ đó cải thiện bảo mật và chất lượng dịch vụ ở cấp độ ứng dụng trên toàn mạng.
* Mạng nhà cung cấp dịch vụ – SDN giúp các nhà cung cấp dịch vụ đơn giản hóa và tự động hóa việc cung cấp mạng để quản lý và kiểm soát dịch vụ và end-to-end.
* Bảo mật trung tâm dữ liệu – SDN hỗ trợ bảo vệ tập trung hơn và đơn giản hóa việc quản trị firewall. Nói chung, một doanh nghiệp phụ thuộc vào tương lửa để bảo mật trung tâm dữ liệu của nó. Tuy nhiên, một trung tâm có thể tạo ra một hệ thống tường lửa phân tán bằng cách thêm các tường lửa ảo để bảo vệ các máy ảo. Lớp bảo mật bổ sung này giúp ngăn vi phạm giữa các máy ảo. Ngoài ra, kiểm soát tập trung và tự động hóa SDN cho phép admin có thể xem, sửa đổi và kiểm soát hoạt động mạng để giảm nguy cơ vi phạm.

# CHƯƠNG III.CÁC CÔNG CỤ HỖ TRỢ TRIỂN KHAI SDN

## 3.1. Mininet

* Mininet là một công cụ mô phỏng mạng SDN được sử dụng rộng rãi để tạo ra một môi trường mạng ảo với các thiết bị chuyển mạch và bộ điều khiển SDN. Nó giúp dễ dàng mô phỏng và kiểm tra các kịch bản SDN trước khi triển khai thực tế.

### 3.1.1. Cách hoạt động

* Mininet sử dụng công nghệ mạng ảo để giả lập các thiết bị như switch, router, và các host. Thông qua Mininet, bạn có thể tạo ra một mạng ảo đầy đủ với các thiết bị này, sau đó sử dụng chúng để mô phỏng các kịch bản mạng khác nhau như điều khiển lưu lượng, bảo mật, hoặc tối ưu hóa hiệu suất.
* Các thiết bị trong Mininet bao gồm:
* **Switch ảo**: Thường là các Open vSwitch (OVS), hỗ trợ giao thức OpenFlow và giao tiếp với các bộ điều khiển SDN.
* **Host ảo**: Các máy ảo đại diện cho các thiết bị đầu cuối như máy tính, điện thoại, hoặc máy chủ.
* **Bộ điều khiển SDN ảo**: Mininet hỗ trợ tích hợp với nhiều bộ điều khiển SDN như ONOS, POX, OpenDaylight.

### 3.1.2. Ứng dụng của Mininet

* **Nghiên cứu SDN**: mô phỏng các kịch bản SDN và thử nghiệm các giải pháp quản lý lưu lượng, bảo mật, hoặc tối ưu hóa mạng.
* **Kiểm thử**: Trước khi triển khai một mạng SDN thực tế, sử dụng Mininet để kiểm tra và tối ưu hóa cấu hình mạng.

## 3.2. Ryu – Bộ điều khiển SDN mã nguồn mở

### 3.2.1. Định nghĩa

* Ryu là một bộ điều khiển SDN mã nguồn mở, được phát triển để cung cấp một nền tảng dễ sử dụng, linh hoạt và mạnh mẽ. Bộ điều khiển này hỗ trợ nhiều giao thức SDN như OpenFlow, NETCONF, và OF-Config. Ryu cung cấp một bộ API Python phong phú, cho phép các nhà phát triển dễ dàng xây dựng các ứng dụng SDN để điều khiển và quản lý mạng.

### 3.2.2. Kiến trúc của RYU

A screenshot of a computer

Description automatically generated

* Kiến trúc của Ryu rất mô-đun và linh hoạt, cho phép người dùng bổ sung hoặc điều chỉnh các tính năng theo nhu cầu cụ thể. Các thành phần chính của kiến trúc Ryu bao gồm:
* **Core (Lõi)**: Đây là phần chính của Ryu, bao gồm các thành phần cơ bản như bộ điều khiển chính, các chức năng giao tiếp với switch OpenFlow, và các thành phần quản lý trạng thái của hệ thống.
* **API**: Ryu cung cấp một bộ API phong phú được viết bằng Python, giúp các nhà phát triển dễ dàng tương tác với mạng SDN thông qua bộ điều khiển. Các API này có thể được sử dụng để phát triển các ứng dụng SDN với các tính năng như quản lý lưu lượng, bảo mật, hoặc tối ưu hóa mạng.
* **Driver giao thức**: Ryu hỗ trợ nhiều giao thức điều khiển mạng như OpenFlow (phiên bản 1.0 đến 1.5), NETCONF, BGP, LISP, OF-Config, OVSDB. Điều này cho phép Ryu làm việc với nhiều loại thiết bị và môi trường mạng khác nhau.

### 3.2.3. Ứng dụng thực tiễn của RYU

* **Quản lý lưu lượng mạng:** Ryu có thể được sử dụng để phát triển các ứng dụng SDN giúp tối ưu hóa lưu lượng mạng, đảm bảo lưu lượng quan trọng được ưu tiên.
* **Bảo mật mạng:** Với Ryu, các ứng dụng bảo mật mạng có thể được phát triển để phát hiện và ngăn chặn các cuộc tấn công mạng theo thời gian thực.
* **Quản lý năng lượng:** Ryu có thể được tích hợp vào các mạng doanh nghiệp để phát triển các giải pháp tiết kiệm năng lượng, tắt các thiết bị không cần thiết khi không sử dụng.

### 3.2.4. Giao thức OpenFlow trong Ryu

* OpenFlow là giao thức phổ biến nhất mà Ryu hỗ trợ và cũng là giao thức tiêu chuẩn trong các mạng SDN. Với OpenFlow, Ryu có thể ra quyết định về việc điều khiển luồng dữ liệu giữa các switch trong mạng. Khi một gói tin đến một switch và switch không biết cách xử lý, nó sẽ gửi yêu cầu lên bộ điều khiển (tức Ryu), và Ryu sẽ quyết định cách xử lý gói tin đó, ví dụ: truyền tiếp, hủy bỏ, hoặc định tuyến lại.
* Trong mạng SDN với Ryu, quá trình điều khiển luồng dữ liệu qua OpenFlow diễn ra như sau:
* Switch OpenFlow gửi yêu cầu Packet-In lên Ryu.
* Ryu xử lý yêu cầu và quyết định dựa trên các chính sách mà ứng dụng đã lập trình.
* Ryu gửi lệnh Packet-Out về switch để hướng dẫn nó xử lý gói tin (chẳng hạn như gửi tới một cổng nhất định).
* Các quyết định này được lưu trong bảng luồng (flow table) của switch, giúp switch xử lý các gói tin tiếp theo một cách tự động.

## 3.3. Open vSwitch(OVS)

* **Open vSwitch (OvS)** 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**:
* **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.
* **ovsdb-server** là một máy chủ cơ sở dữ liệu thu gọn, thông qua đó **ovs-vswitchd** truy vấn để có được cấu hình.
* **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 vào 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.

A diagram of a computer network

Description automatically generated

# CHƯƠNG IV. MỘT VÀI ỨNG DỤNG CỦA SDN VÀ RYU

## 4.1. Quản lý mạng trung tâm dữ liệu (Data Center Network Management)

* **SDN**:
* Tăng hiệu quả sử dụng tài nguyên trong trung tâm dữ liệu bằng cách quản lý tập trung luồng dữ liệu giữa các máy chủ.
* Dễ dàng cấu hình và điều chỉnh mạng động theo nhu cầu của ứng dụng hoặc người dùng.
* Giảm thiểu chi phí bằng cách tự động hóa việc quản lý và tối ưu hóa luồng dữ liệu.
* **Ryu**:
* Quản lý kết nối và định tuyến giữa các máy chủ trong trung tâm dữ liệu.
* Phát triển ứng dụng tự động hóa điều khiển switch và thiết bị mạng bằng giao thức OpenFlow.

## 4.2. Mạng doanh nghiệp (Enterprise Networks)

* **SDN**:
* Cung cấp khả năng kiểm soát chính sách mạng tập trung, như kiểm soát truy cập, định tuyến ưu tiên (QoS).
* Quản lý băng thông mạng hiệu quả hơn cho các ứng dụng doanh nghiệp.
* Đơn giản hóa việc triển khai các dịch vụ mạng mới.
* **Ryu**:
* Cung cấp công cụ tùy chỉnh cho mạng doanh nghiệp thông qua các ứng dụng như tường lửa (firewall) và thống kê lưu lượng.
* Dễ dàng tích hợp với hệ thống bảo mật và quản lý tập trung của doanh nghiệp.

## 4.3. Mạng viễn thông (Telecommunication Networks)

* **SDN**:
* SDN giúp triển khai mạng 5G hiệu quả hơn bằng cách quản lý động các tài nguyên mạng.
* Hỗ trợ Network Slicing, cho phép phân chia tài nguyên mạng thành các phần logic phục vụ nhiều dịch vụ khác nhau.
* **Ryu**:
* Phát triển các ứng dụng cụ thể như quản lý băng thông, tối ưu hóa luồng dữ liệu.
* Giám sát và phân tích lưu lượng trong thời gian thực để tăng hiệu suất.

## 4.4. Internet of Things (IoT)

* **SDN**:
* Quản lý các kết nối IoT khổng lồ bằng cách tối ưu hóa và điều phối lưu lượng giữa các thiết bị IoT.
* Cung cấp khả năng định tuyến thông minh cho các thiết bị IoT theo thời gian thực.
* **Ryu**:
* Cung cấp môi trường phát triển ứng dụng mạng IoT, như định tuyến dữ liệu từ cảm biến hoặc các thiết bị thông minh.
* Tích hợp với giao thức OpenFlow để điều khiển lưu lượng đến các cổng IoT.
  + - Ngoài ra còn một số ứng dụng như là: Bảo mật mạng (Network Security), Mạng ảo hóa (Network Virtualization), Quản lý giao thông mạng (Traffic Engineering), Quản lý mạng thành phố thông minh (Smart City),…

# CHƯƠNG V. KẾT LUẬN

Mạng điều khiển bằng phần mềm (SDN) mang đến nhiều lợi ích về tính linh hoạt, quản lý tập trung và khả năng mở rộng. Bộ điều khiển SDN Ryu, với khả năng hỗ trợ nhiều giao thức và API Python dễ sử dụng, là một lựa chọn lý tưởng cho các nhà nghiên cứu, nhà phát triển và doanh nghiệp. Sự kết hợp giữa Ryu và các công cụ hỗ trợ như Mininet giúp kiểm thử và triển khai các giải pháp SDN một cách dễ dàng và hiệu quả.

# CHƯƠNG VI. KỊCH BẢN TRIỂN KHAI

## 6.1. Kịch bản I:

* Host: 4 hosts
* Switch: 2 switches
* Router:  1 router
* Controller: 1 controller

A diagram of computer network

Description automatically generated

* Kịch bản có 2 lớp mạng :
* 10.0.10.0/24: PC2 và PC3
* 10.0.20.0/24: PC4 và PC5
* Kịch bản triển khai:
* Các host cùng lớp mạng sẽ liên kết với cùng một switch
* Cho hai lớp mạng giao tiếp được với nhau
* Cả hai lớp mạng đều được phép ping qua nhau

## 6.2. Kịch bản II:

* Host: 7 hosts
* Switch: 2 switches
* Router:  1 router
* Controller: 1 controller

A diagram of computer network

Description automatically generated

* Kịch bản có 2 lớp mạng :
* 10.0.0.0/24: PC2 và PC3
* 100.0.0.0/8: PC4, PC5, PC6, PC7 và PC8
* Kịch bản triển khai:
* Các host cùng lớp mạng sẽ liên kết với cùng một switch
* Cho hai lớp mạng giao tiếp được với nhau
* Lớp mạng 10.0.0.0/24 được phép ping qua lớp mạng 100.0.0.0/8
* Lớp mạng 100.0.0.0/8 không được phép ping qua lớp mạng 10.0.0.0/24

## 6.3. Kịch bản III:

* Host: 4 hosts
* Switch: 2 switches
* Router:  2 router
* Controller: 1 controller

A computer network diagram with a computer and a computer server

Description automatically generated with medium confidence

* Kịch bản có 2 lớp mạng :
* 192.168.1.0/24: PC2 và PC3
* 192.168.2.0/24: PC4 và PC5
* Kịch bản triển khai:
* Các host cùng lớp mạng sẽ liên kết khác switch
* Các host khác lớp mạng sẽ liên kết cùng switch
* Cho hai lớp mạng giao tiếp được với nhau
* Cả hai lớp mạng đều được phép ping qua nhau

# Chương VII. HƯỚNG DẪN SETUP

## 7.1. Cài đặt mininet:

* Install ubuntu 20.04: https://releases.ubuntu.com/focal/ubuntu-20.04.6-desktop-amd64.iso
* Install mininet: sudo apt install mininet
* Check version: mn --version

## 7.2. Cài đặt Ryu:

* install tool: sudo apt-get install python3-pip
* install ryu: sudo pip3 install ryu
* install more: pip3 install eventlet==0.30.2
* check version: ryu-manager --version

# CHƯƠNG VIII. CÁC BƯỚC THỰC HIỆN

## 8.0. Tổng quát

* Trước khi thực hiện các kich bản ta phải thiết lập hàm **Packet\_in\_handler**, hàm này có chức năng tương tự như Routing Table của Cisco nhưng có nhiều tính năng hơn, vì nó xử lý các gói tin mà switch gửi đến controller khi không có luồng (flow) nào trong bảng luồng (flow table) của switch khớp với gói tin
* Đây là một phần quan trọng trong kiến trúc điều khiển tập trung của SDN.

A computer screen shot of a program

Description automatically generated  
A screenshot of a computer program

Description automatically generated

* **Phân tích và trích xuất thông tin gói tin**

A close-up of a computer code

Description automatically generated

* Tạo một đối tượng pkt từ dữ liệu gói tin trong thông báo (msg.data).
* Lưu các giao thức của gói tin vào header\_list (một dictionary) với tên giao thức làm khóa.
* Phân loại gói tin dựa trên giao thức (ARP, ICMP, TCP, UDP...).
* **Kiểm tra VLAN ID**

A close-up of a computer code

Description automatically generated

* Nếu gói tin thuộc một VLAN, lấy VLAN ID từ trường VLAN trong gói tin.
* Nếu không, đặt vlan\_id là mặc định (VLANID\_NONE).
* Xác định VLAN để phân biệt các miền mạng trong xử lý gói tin
* **Gửi gói tin đến handler của VLAN**

A computer screen shot of a computer code

Description automatically generated

* Nếu vlan\_id đã được cấu hình trong router, gửi gói tin đến handler tương ứng của VLAN.
* Nếu không, ghi log và bỏ gói tin.
* Đảm bảo chỉ xử lý các gói tin thuộc VLAN hợp lệ
* Trích xuất thông tin từ gói tin
* **datapath**:
* Lấy datapath (switch) mà gói tin được gửi từ đó.
* Đây là đối tượng đại diện cho switch trong mạng SDN.
* **ofproto và parser**:
* Lấy thông tin giao thức OpenFlow và bộ phân tích giao thức (parser) từ switch.
* Được dùng để gửi các lệnh OpenFlow từ controller đến switch.
* **Phân tích gói tin**:
* pkt = packet.Packet(msg.data):
* Tạo một đối tượng gói tin (pkt) từ dữ liệu gói tin được gửi bởi switch.
* Lấy các giao thức cụ thể trong gói tin:
* eth: Giao thức Ethernet (lớp 2).
* pkt\_arp: Giao thức ARP.
* pkt\_icmp: Giao thức ICMP.
* pkt\_ip: Giao thức IPv4.
* pkt\_tcp: Giao thức TCP.
* pkt\_udp: Giao thức UDP.

* **Xử lý các giao thức cụ thể**

1. **ARP**

A close-up of a computer screen

Description automatically generated

*Hoạt động:*

* Kiểm tra xem gói tin có phải là ARP không (pkt\_arp).
* Trích xuất địa chỉ IP nguồn (srcip) và đích (dstip) từ gói ARP.
* Ghi log thông tin về gói ARP, bao gồm:
* Địa chỉ IP nguồn (src\_ip).
* Địa chỉ IP đích (dst\_ip).
* Opcode (Request/Reply).

*Vai trò:*

* Giám sát các gói ARP trong mạng.
* Có thể mở rộng logic để xử lý các gói ARP như trả lời ARP Request.

1. **ICMP**

A close-up of a computer screen

Description automatically generated

Hoạt động:

* Kiểm tra xem gói tin có phải là ICMP không (pkt\_icmp).
* Trích xuất địa chỉ IP nguồn (srcip) và đích (dstip) từ gói tin IPv4.
* Ghi log thông tin về gói ICMP, bao gồm:
* Địa chỉ nguồn và đích.
* Loại giao thức (ICMP - inet.IPPROTO\_ICMP).

Vai trò:

* Giám sát các gói tin kiểm tra kết nối (ping).
* Có thể mở rộng logic để chặn hoặc trả lời các gói ICMP

1. **TCP**

A computer screen shot of a computer code

Description automatically generated

*Hoạt động:*

* Kiểm tra xem gói tin có phải là TCP không (pkt\_tcp).
* Trích xuất địa chỉ IP nguồn (srcip) và đích (dstip) từ gói tin IPv4.
* Trích xuất thông tin TCP:
* Cổng nguồn (t.src\_port).
* Cổng đích (t.dst\_port).
* Các cờ TCP (t.bits) như SYN, ACK, FIN...
* Ghi log đầy đủ thông tin gói TCP.

*Vai trò:*

* Giám sát lưu lượng TCP, hữu ích cho phát hiện lưu lượng không mong muốn (như SYN flood).
* Có thể mở rộng để tạo các quy tắc tường lửa chặn lưu lượng TCP.

1. **UDP**

*Hoạt động:*

* Kiểm tra xem gói tin có phải là UDP không (pkt\_udp).
* Trích xuất địa chỉ IP nguồn (srcip), đích (dstip), cổng nguồn và cổng đích.
* Định nghĩa các hành động (actions) để xử lý gói UDP:
* Cập nhật các trường IP và UDP (nguồn, đích).
* Chuyển tiếp gói tin qua cổng (out\_port).

*Vai trò:*

* Xử lý các gói UDP, chẳng hạn như chuyển tiếp lưu lượng DNS hoặc VoIP.
* Có thể dùng để thêm quy tắc luồng (flow rule) vào switch, giúp tối ưu xử lý các gói tương tự trong tương lai.

## 8.1. Kịch bản I:

### 8.1.1. Run RYU

* Tab 1: sudo ryu-manager simple\_switch\_13\_custom.py rest\_router\_FW\_custom.py /home/mininet/Desktop/DoAn\_QTM/flowmanager/flowmanager.py --observe-links --ofp-tcp-listen-port 6633 --wsapi-port 8085

### 8.1.2. Run topology 1

* Tab 2: sudo python2 topo1.py

### 8.1.3. POST ip getway

* Tab 3:
* curl -X POST -d '{"address": "10.0.10.1/24"}' http://localhost:8085/router/0000000000000021
* curl -X POST -d '{"address": "10.0.20.1/24"}' http://localhost:8085/router/0000000000000021

### 8.1.4. Check topology

* Tab 2: pingall
* Web interface: <http://localhost:8085/home/flows.html>

## 8.2. Kịch bản II:

### 8.2.1. Run RYU

* Tab 1: sudo ryu-manager simple\_switch\_13\_custom.py rest\_router\_FW\_custom.py /home/mininet/Desktop/DoAn\_QTM/flowmanager/flowmanager.py --observe-links --ofp-tcp-listen-port 6633 --wsapi-port 8085

### 8.2.2. Run topology 2

* Tab 2: sudo python2 topo2.py

### 8.2.3. POST ip getway

* Tab 3:
* curl -X POST -d '{"address": "10.0.0.1/24"}' http://localhost:8085/router/0000000000000021
* curl -X POST -d '{"address": "100.0.0.1/8"}' http://localhost:8085/router/0000000000000021

### 8.2.4. Check topology

* Tab 2: pingall
* Web interface: <http://localhost:8085/home/flows.html>

Trong kịch bản hai bổ sung code có chức năng tường lửa chặn ICMP và TCP SYN vào file gốc Router.py

A screenshot of a computer

Description automatically generated

* + - **Mục đích chính của đoạn code:**
* **Đăng ký một switch (datapath) làm router**: Chỉ áp dụng nếu switch có datapath ID là 0x21.
* **Thiết lập các quy tắc tường lửa (firewall rules)**:
* Chặn **ICMP Echo Request** (Ping) từ dải địa chỉ IP 100.0.0.1/8.
* Chặn **TCP SYN** (khởi tạo kết nối TCP) từ dải địa chỉ IP 100.0.0.1/8.
* **Lưu danh sách router đã đăng ký** để quản lý.
* **Kiểm tra datapath ID**

A black and white text

Description automatically generated

* Chỉ tiếp tục nếu datapath ID của switch là 0x21.
* Nếu không, thoát khỏi hàm mà không thực hiện gì.
* **Thiết lập các quy tắc tường lửa**

**# Quy tắc 1: Khởi tạo tường lửa và chặn ICMP Echo Request**

A screenshot of a computer

Description automatically generated

* Match với gói tin ICMP Echo Request (ping) từ nguồn 100.0.0.1/8.
* Drop gói tin (không thực hiện bất kỳ hành động nào).
* Ngăn chặn lưu lượng ICMP (ping) từ dải địa chỉ không mong muốn.

**# Quy tắc 2: Chặn TCP SYN**

A computer screen shot of a computer code

Description automatically generated

* Match với gói tin TCP có cờ SYN từ nguồn 100.0.0.1/8
* Ngăn chặn lưu lượng TCP SYN không mong muốn từ địa chỉ nguồn cụ thể
* **Đăng ký Router**

A screen shot of a computer code

Description automatically generated

* Tạo một đối tượng Router liên kết với switch (datapath).
* Thêm router vào danh sách cls.\_ROUTER\_LIST để quản lý.
* Ghi log thông báo router đã được thêm thành công.
* Duy trì trạng thái của router trong hệ thống
* **Vai trò của hàm**
* Tường lửa: Ngăn chặn lưu lượng không mong muốn.
* Quản lý router: Ghi nhận và quản lý các switch được cấu hình làm router.
* Tích hợp với SDN Controller: Gửi các quy tắc OpenFlow đến switch thông qua Ryu Controller.

## 8.3. Kịch bản III:

### 8.3.1. Run RYU

* Tab 1: sudo ryu-manager ryu-two-router-vlan.py --observe-links --ofp-tcp-listen-port 6633

### 8.3.2. Run topology 3

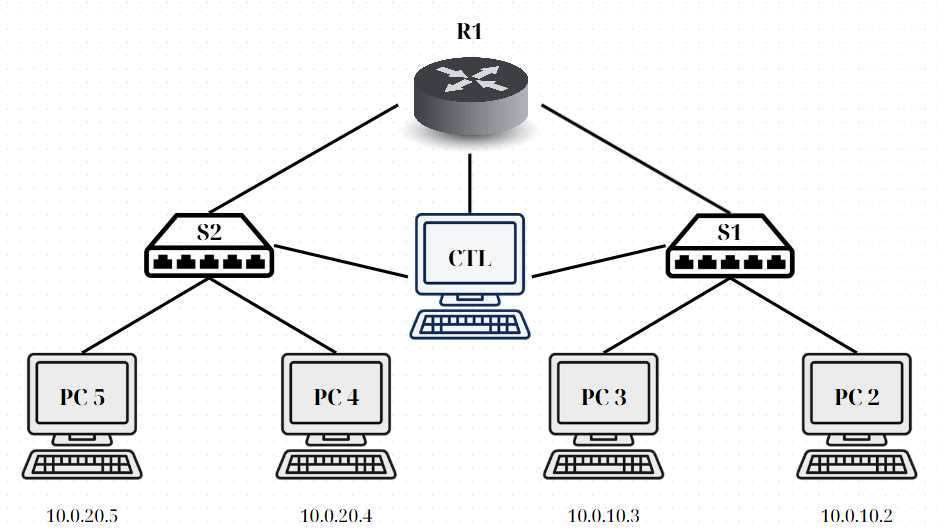
* Tab 2: sudo python2 topo3.py

### 8.3.3. Check topology

* Tab 2: pingall

# CHƯƠNG IX. PHÂN TÍCH FLOW KỊCH BẢN

## 9.1. Kịch bản I



1. **S1**



* Gửi các gói LLDP hoặc các gói liên quan đến quản lý mạng lên controller để xử lý.



* eth\_src = 00:00:00:00:00:03 là PC3
* eth\_dst = 00:00:00:00:00:02 là PC2
* in\_port = 2
* out\_port = 1
* gói tin đi từ PC3 qua PC2 sẽ đi vào Port 2 của S1 và ra Port 1 của S1



* eth\_src = 00:00:00:00:00:02 là PC2
* eth\_dst = 00:00:00:00:00:03 là PC3
* in\_port = 1
* out\_port = 2
* gói tin đi từ PC2 qua PC3 sẽ đi vào Port 1 của S1 và ra Port 2 của S1



* eth\_src = 32:57:5c:8a:02:46 là gateway
* eth\_dst = 00:00:00:00:00:02 là PC2
* in\_port = 3
* out\_port = 1
* các gói tin của lớp mạng khác muốn đi đến PC2 thì phải thông qua gateway, nên gói tin sẽ đi qua gateway vào từ Port 3 của S1 và Port 1 của S1 để đến PC2



* eth\_src = 00:00:00:00:00:02 là PC2
* eth\_dst = 32:57:5c:8a:02:46 là gateway
* in\_port = 1
* out\_port = 3
* các gói tin đi từ PC2 đi qua lớp mạng khác phải thông qua gate ,nên gói tin sẽ đi lên gateway đặt ở router bằng cách đi vào Port 1 của S1 và ra ở Port 3 của S1



* eth\_src = 32:57:5c:8a:02:46 là gateway
* eth\_dst = 00:00:00:00:00:03 là PC3
* in\_port = 3
* out\_port = 2
* các gói tin của lớp mạng khác muốn đi đến PC3 thì phải thông qua gateway, nên gói tin sẽ đi qua gateway vào từ Port 3 của S1 và Port 2 của S1 để đến PC3

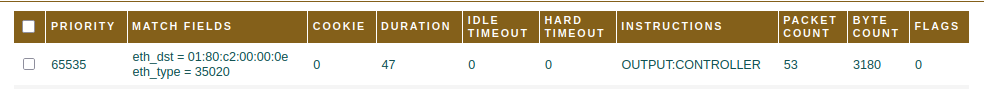


* eth\_src = 00:00:00:00:00:03 là PC3
* eth\_dst = 32:57:5c:8a:02:46 là gateway
* in\_port = 2
* out\_port = 3
* các gói tin đi từ PC3 đi qua lớp mạng khác phải thông qua gate ,nên gói tin sẽ đi lên gateway đặt ở router bằng cách đi vào Port 2 của S1 và ra ở Port 3 của S1

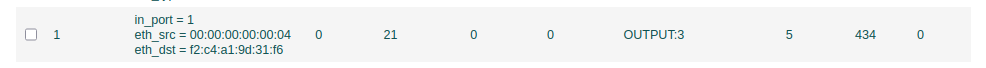


* Các gói tin không thỏa các flows trên sẽ được chuyển lên CONTROLLER

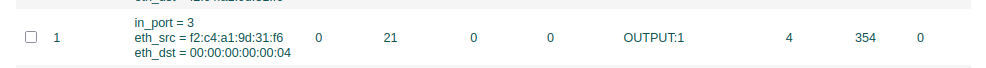
1. **S2**



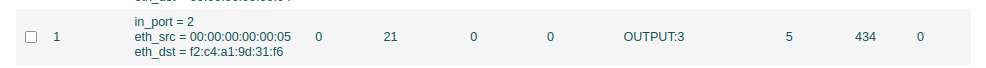
* Gửi các gói LLDP hoặc các gói liên quan đến quản lý mạng lên controller để xử lý.



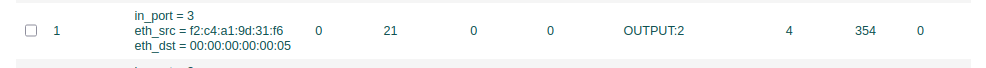
* eth\_src = 00:00:00:00:00:04 là PC4
* eth\_dst = f2:c4:a1:9d:31:f6 là gateway
* in\_port = 1
* out\_port = 3
* các gói tin từ PC4 muốn đi ra lớp mạng khác sẽ đi đến gateway được đặt trên router, nên gói tin sẽ đi vào Port 1 của S2 và ra ở Port 3 của S2 để đến gateway



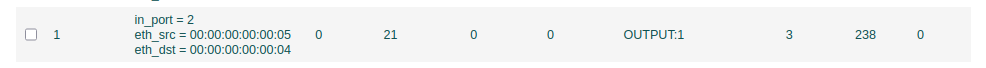
* eth\_src = f2:c4:a1:9d:31:f6 là gateway
* eth\_dst = 00:00:00:00:00:04 là PC4
* in\_port = 3
* out\_port = 1
* các gói tin muốn đi vào PC4 từ lớp mạng khác thì phải đi qua gateway của lớp mạng của PC4 ,vì vậy gói tin sẽ đi từ gateway đến PC4 thông qua đầu vào là Port 3 của S2 và đầu ra là Port 1 của S2



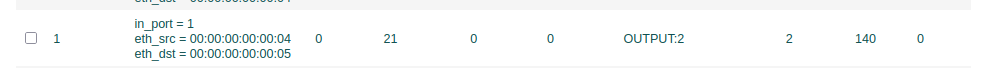
* eth\_src = 00:00:00:00:00:05 là PC5
* eth\_dst = f2:c4:a1:9d:31:f6 là gateway
* in\_port = 2
* out\_port = 3
* các gói tin từ PC5 muốn đi ra lớp mạng khác sẽ đi đến gateway được đặt trên router, nên gói tin sẽ đi vào Port 2 của S2 và ra ở Port 3 của S2 để đến gateway



* eth\_src = f2:c4:a1:9d:31:f6 là gateway
* eth\_dst = 00:00:00:00:00:05 là PC5
* in\_port = 3
* out\_port = 2
* các gói tin muốn đi vào PC5 từ lớp mạng khác thì phải đi qua gateway của lớp mạng của PC5 ,vì vậy gói tin sẽ đi từ gateway đến PC5 thông qua đầu vào là Port 3 của S2 và đầu ra là Port 2 của S2



* eth\_src = 00:00:00:00:00:05 là PC5
* eth\_dst = 00:00:00:00:00:04 là PC4
* in\_port = 2
* out\_port = 1
* gói tin đi từ PC5 sang PC4 sẽ vào Port 2 của S2 và ra ở Port 1 của S2



* eth\_src = 00:00:00:00:00:04 là PC4
* eth\_dst = 00:00:00:00:00:05 là PC5
* in\_port = 1
* out\_port = 2
* gói tin đi từ PC4 sang PC5 sẽ vào Port 1 của S2 và ra ở Port 2 của S2



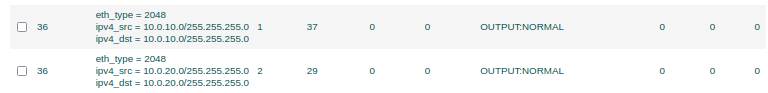
* các gói tin không thỏa các flows trên sẽ được chuyển đến CONTROLLER

1. **R1**

A screenshot of a computer

Description automatically generated

* Các gói tin LLDP và các gói tin đó đích là 10.0.10.1 và 10.0.20.1 ( 2 gateway của 2 lớp mạng) sẽ được chuyển đến CONTROLLER



* Các gói tin nằm cùng một lớp mạng thì giao tiếp bình thường



* Các gói tin có đích là 10.0.10.2 tức là PC2 :

+ DEC\_NW\_TTL: Giảm TTL đi 1 (để thể hiện gói tin đã đi qua router).

+ Thay đổi địa chỉ MAC nguồn thành địa chỉ MAC của gateway của lớp mạng 10.0.10.0/24

+ Thay đổi địa chỉ MAC đích thành địa chỉ MAC của PC2

+ Đầu ra là Port 1 của R1



* Các gói tin có đích là 10.0.10.3 tức là PC3 :

+ DEC\_NW\_TTL: Giảm TTL đi 1 (để thể hiện gói tin đã đi qua router).

+ Thay đổi địa chỉ MAC nguồn thành địa chỉ MAC của gateway của lớp mạng 10.0.10.0/24

+ Thay đổi địa chỉ MAC đích thành địa chỉ MAC của PC3

+ Đầu ra là Port 1 của R1



* Các gói tin có đích là 10.0.20.5 tức là PC5 :

+ DEC\_NW\_TTL: Giảm TTL đi 1 (để thể hiện gói tin đã đi qua router).

+ Thay đổi địa chỉ MAC nguồn thành địa chỉ MAC của gateway của lớp mạng 10.0.20.0/24

+ Thay đổi địa chỉ MAC đích thành địa chỉ MAC của PC5

+ Đầu ra là Port 2 của R1



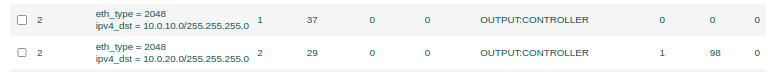
* Các gói tin có đích là 10.0.20.4 tức là PC4 :

+ DEC\_NW\_TTL: Giảm TTL đi 1 (để thể hiện gói tin đã đi qua router).

+ Thay đổi địa chỉ MAC nguồn thành địa chỉ MAC của gateway của lớp mạng 10.0.20.0/24

+ Thay đổi địa chỉ MAC đích thành địa chỉ MAC của PC4

+ Đầu ra là Port 2 của R1



* Gửi các gói tin broadcast hoặc multicast tới controller để xử lý (thường là ARP hoặc DHCP).

## 9.2. Kịch bản II

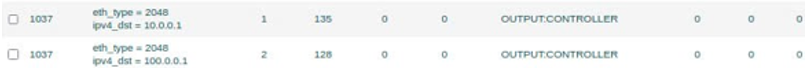
A screen shot of a computer

Description automatically generated

* Priority 65535 (Gói LLDP hoặc quản lý mạng):

****

* **eth\_dst:** 01:80:c2:00:00:0e, **eth\_type:** 35020.
* **Mô tả:** Gửi các gói tin LLDP hoặc liên quan đến quản lý mạng lên controller.
* **OUTPUT:CONTROLLER**: Tất cả gói tin thuộc loại quản lý mạng được chuyển trực tiếp tới controller để xử lý
* Priority 1037 (Quản lý địa chỉ cụ thể):



* **eth\_type:** 2048, **ipv4\_dst:** 10.0.0.1 .
  + **Mô tả:** Gói tin đến 10.0.0.1 sẽ được chuyển tiếp lên controller để xử lý.
  + **OUTPUT:CONTROLLER**: Controller cần xử lý các gói tin có đích đến là 10.0.0.1 để điều khiển.
* **eth\_type:** 2048, **ipv4\_dst:** 100.0.0.1.
  + **Mô tả:** Gói tin đến 100.0.0.1 sẽ được chuyển tiếp lên controller để xử lý.
  + **Hành động:** OUTPUT:CONTROLLER: Controller sẽ xử lý gói tin đến địa chỉ này.
* Priority 1001 và 1000 (Chặn lưu lượng):

A group of dots on a white surface

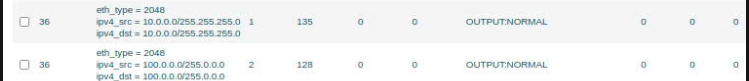
Description automatically generated

**Priority 1001 (Chặn TCP SYN):**

* **Match fields:**
  + **eth\_type:** 2048 (IPv4).
  + **ipv4\_src:** 100.0.0.0/255.0.0.0.
  + **ip\_proto:** 6 (TCP).
  + **tcp\_flags:** 2 (SYN flag).
* **Mô tả:** DROP Chặn các gói TCP SYN từ lớp mạng 100.0.0.0/8 (khởi tạo kết nối TCP).
* **Ý nghĩa:** Ngăn chặn việc khởi tạo kết nối TCP trái phép từ lớp mạng này.

**Priority 1000 (Chặn ICMP Echo Request - Ping):**

* **Match fields:**
  + **eth\_type:** 2048 (IPv4).
  + **ipv4\_src:** 100.0.0.0/255.0.0.0.
  + **ip\_proto:** 1 (ICMP).
  + **icmp4\_type:** 8 (ICMP echo request).
* **Mô tả:** DROP Chặn các gói ICMP (ping) từ lớp mạng 100.0.0.0/8.
* **Ý nghĩa:** Ngăn các thiết bị trong lớp mạng 100.0.0.0/8 thực hiện lệnh ping đến các mạng khác.
* Priority 36 (Giao tiếp giữa các lớp mạng):



* Cho phép gói tin từ lớp mạng 10.0.0.0/8 đến lớp mạng 100.0.0.0/8 và ngược lại.
* Priority 35 (Luồng cụ thể cho từng PC):

1. Luồng dành cho PC2 (IPv4 đích: 10.0.0.2):



* **Match fields:**
  + **eth\_type:** 2048 (IPv4).
  + **ipv4\_dst:** 10.0.0.2.
* **Hành động:**
  + DEC\_NW\_TTL: Giảm TTL của gói tin.
  + SET\_FIELD:

+ Đặt địa chỉ MAC nguồn thành 52:0b:fa:05:81.

+ Đặt địa chỉ MAC đích thành 00:00:00:00:00:02.

* + OUTPUT:1: Gửi gói tin qua port 1.
* **Mô tả:** Gói tin đến PC2 được định tuyến qua port 1

1. Luồng dành cho PC3 (IPv4 đích: 10.0.0.3):



* **Match fields:**
  + **eth\_type:** 2048 (IPv4).
  + **ipv4\_dst:** 10.0.0.3.
* **Hành động:** Tương tự như PC2, nhưng địa chỉ MAC đích là 00:00:00:00:00:03.
* **Mô tả:** Gói tin đến PC3 được định tuyến qua port 1.

1. Luồng dành cho PC4 (IPv4 đích: 100.0.0.4):



* **Match fields:**
  + **eth\_type:** 2048 (IPv4).
  + **ipv4\_dst:** 100.0.0.4.
  + DEC\_NW\_TTL: Giảm TTL của gói tin.
  + SET\_FIELD:

+ Đặt địa chỉ MAC nguồn thành 5a:14:7e:ac:11:30.

+ Đặt địa chỉ MAC đích thành 00:00:00:00:00:04.

* + OUTPUT:1: Gửi gói tin qua port 2.
* **Mô tả:** Gói tin đến PC4 được định tuyến qua port 2.

1. Luồng dành cho PC5 (IPv4 đích: 100.0.0.5):



* **Match fields:**
  + **eth\_type:** 2048 (IPv4).
  + **ipv4\_dst:** 100.0.0.5.
* **Hành động:** Tương tự như PC4, nhưng địa chỉ MAC đích là 00:00:00:00:00:05.
* **Mô tả:** Gói tin đến PC5 được định tuyến qua port 2.

1. Các luồng còn lại:

A number on a white background

Description automatically generated

* Hành động: tương tự những cái trên

## 9.3. Kịch bản III

Trên kịch bản III dùng 2 switch khác nhau để kết nối các máy có các địa chỉ mạng khác nhau. Sau khi thực hiện chạy kịch bản III. Thu được hai bảng log ghi lại flow của hai switch S1, S2 và Router R1, R2 như sau

**Đặc điểm và tính năng của bảng Flow 1,2:**

* **Flow của Switch (S1 hoặc S2)**:
* Flow liên quan đến **broadcast/multicast** (Priority 65535).
* Flow xử lý giao tiếp **nội bộ trong cùng mạng con** (Priority 36, OUTPUT:NORMAL).
* Flow gửi gói tin đến controller để học (Priority 1037).
* **Flow của Router (R1 hoặc R2)**:
* Flow liên quan đến **định tuyến liên mạng** (Priority 35):
* Có các hành động DEC\_NW\_TTL và SET\_FIELD, đặc trưng của router.
* Xử lý các gói tin cần định tuyến giữa các mạng khác nhau (ví dụ: 192.168.1.x đến 192.168.2.x).

**Chức năng của từng flow chính**

|  |  |  |  |
| --- | --- | --- | --- |
| Priority | Match Fields | Instructions | Chức năng |
| 65535 | eth\_dst = 01:80:c2:00:00:0e  eth\_type = 35020 | Output: Controller | Xử lý các gói tin multicast,/broadcast(như ARP Request, LLDP) và gửi đến controller để phần tích |
| 1037 | eth\_type = 2048  ipv4\_dst = 192.168.1.1 hoặc 192.168.2.1 | Output: Controller | Gửi các gói tin đến controller khi không có flow phù hợp để xử lý; controller sẽ học và cài đặt thêm flow động |
| 36 | eth\_type = 2048  ipv4\_src/dst cùng mạng con (ví dụ: 192.168.1.x, 192.168.2.x) | Output: normal | Xử lý các gói tin nội bộ trong cùng mạng con (cùng switch) mà không cần qua router. |
| 35 | eth\_type = 2048  ipv4\_dst thuộc mạng con khác thông qua router (R1, R2) | DEC\_NW\_TTL  SET\_FIELD  OUTPUT | Định tuyến gói tin giữa các mạng con qua router hoặc Layer 3 Switch. TTL giảm và địa chỉ MAC được thay đổi phù hợp. |
| 2 | eth\_type = 2048  ipv4\_dst = 192.168.1.255 hoặc 192.168.2.255 | Output: Controller | Gửi các gói tin broadcast hoặc multicast đến controller để xử lý, thường dùng cho ARP Request hoặc DHCP. |
| 0 | ANY | Output: Controller | Flow mặc định, xử lý tất cả các gói tin không khớp với bất kỳ flow nào khác. |

### 9.3.1 Bảng flow 1

A screenshot of a computer

Description automatically generated

**PHÂN TÍCH TƯỜNG LUỒNG:**



* **Priority**: 65535 (cao nhất), xử lý trước tất cả các flow khác.
* **Match Fields**:
  + eth\_dst = 01:80:c2:00:00:0e: Địa chỉ multicast dành cho quản lý mạng (LLDP, ARP).
  + eth\_type = 35020: Loại gói tin LLDP.
* **Instructions**: OUTPUT:CONTROLLER - Gửi gói tin khớp đến controller SDN để xử lý.
* **Timeouts**:
  + Không có Idle và Hard Timeout, flow luôn tồn tại.
* **Statistics**: Đã xử lý 47 gói tin (2820 byte).

**Chức năng chính:**

* Gửi các gói tin multicast/broadcast (LLDP, ARP) đến controller để giám sát mạng.
* Phát hiện topology và trạng thái liên kết vật lý giữa các thiết bị mạng.
* Quản lý và học mạng qua controller SDN.

A group of numbers on a white background

Description automatically generated

* **Priority**: 1037 (ưu tiên trung bình cao, dùng cho học mạng).
* **Match Fields**:
  + eth\_type = 2048: Gói tin IPv4.
  + ipv4\_dst: Địa chỉ IP đích là:
    - 192.168.1.1
    - 192.168.2.1
    - 10.10.10.1
* **Instructions**: OUTPUT:CONTROLLER - Gửi gói tin đến controller để xử lý.

**Chức năng chính**:

* **Học mạng (Dynamic Learning)**:
  + Các gói tin gửi đến những địa chỉ IP này sẽ được chuyển đến controller vì chưa có flow cụ thể để xử lý.
  + Controller sử dụng các gói tin này để cài đặt thêm các flow phù hợp trong tương lai.
* **Xử lý gói tin không rõ đích**:
  + Khi switch không có thông tin về địa chỉ đích, flow này đảm bảo rằng gói tin được gửi lên controller thay vì bị bỏ qua.
* **Ứng dụng**:
  + Các địa chỉ IP đích (gateway hoặc thiết bị mạng khác) cần được controller theo dõi và quản lý để xây dựng hoặc cập nhật bảng flow.

A white background with blue numbers

Description automatically generated

* **Priority**: 36 (ưu tiên trung bình thấp, dùng cho giao tiếp nội bộ).
* **Match Fields**:
  + eth\_type = 2048: Gói tin IPv4.
  + ipv4\_src và ipv4\_dst thuộc cùng một mạng con:
    - 192.168.1.0/255.255.255.0
    - 192.168.2.0/255.255.255.0
    - 10.10.10.0/255.255.255.0
* **Instructions**: OUTPUT:NORMAL - Chuyển tiếp gói tin theo cách truyền thống (nội bộ switch).

**Chức năng chính**:

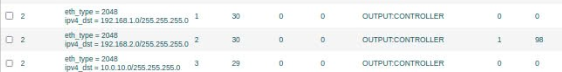
* **Xử lý giao tiếp nội bộ trong mạng con**:
  + Các gói tin có địa chỉ nguồn và đích thuộc cùng một mạng con (cùng dải subnet) sẽ được xử lý nội bộ trong switch mà không cần qua router.
* **Tối ưu hóa lưu lượng mạng nội bộ**:
  + Loại bỏ sự phụ thuộc vào controller hoặc router cho các gói tin nội bộ.
  + Đảm bảo hiệu năng cao khi xử lý lưu lượng trong cùng switch.
* **Thống kê**:
  + **Packet Count = 8, Byte Count = 784** (dòng 1 và 2): Gói tin đã được chuyển tiếp nội bộ.
  + **Packet Count = 0** (dòng 3): Hiện tại chưa có lưu lượng xử lý cho mạng 10.10.10.0.



* **Priority**: 35 (ưu tiên trung bình cao, dành cho định tuyến liên mạng).
* **Match Fields**:
  + eth\_type = 2048: Gói tin IPv4.
  + ipv4\_dst: Các địa chỉ IP thuộc mạng khác:
    - 192.168.1.1
    - 192.168.2.1
    - 10.10.10.2
    - 192.168.2.3, 192.168.2.2
* **Instructions**:
  + DEC\_NW\_TTL: Giảm giá trị TTL của gói tin.
  + SET\_FIELD: Thay đổi địa chỉ MAC nguồn (eth\_src) và đích (eth\_dst) để định tuyến chính xác.
  + OUTPUT: Chuyển tiếp gói tin đến cổng cụ thể (OUTPUT1, OUTPUT4...).

**Chức năng chính**:

* **Định tuyến liên mạng (Inter-network Routing)**:
  + Xử lý gói tin có địa chỉ đích nằm ngoài mạng nội bộ, yêu cầu định tuyến qua router hoặc switch Layer 3.
* **Cập nhật thông tin gói tin khi định tuyến**:
  + Giảm giá trị TTL (DEC\_NW\_TTL) để ngăn vòng lặp mạng.
  + Thay đổi địa chỉ MAC nguồn/đích (SET\_FIELD) để phù hợp với giao tiếp giữa các mạng.
* **Thống kê**:
  + **Packet Count và Byte Count**:
    - Một số dòng flow đã xử lý gói tin, ví dụ:
      * Dòng ipv4\_dst = 192.168.2.3 có **Packet Count = 294**, **Byte Count = 392**.
      * Dòng ipv4\_dst = 192.168.2.2 có **Packet Count = 882**, **Byte Count = lớn**.
    - Các dòng khác hiện tại chưa xử lý (Packet Count = 0).



* **Priority**: 2 (ưu tiên thấp, dùng cho broadcast/multicast hoặc gói tin đặc biệt).
* **Match Fields**:
  + eth\_type = 2048: Gói tin IPv4.
  + ipv4\_dst:
    - 192.168.1.0/255.255.255.0
    - 192.168.2.0/255.255.255.0
    - 10.10.10.0/255.255.255.0
* **Instructions**: OUTPUT:CONTROLLER - Gửi gói tin đến controller để xử lý.

**Chức năng chính**:

* **Xử lý gói tin broadcast/multicast**:
  + Gói tin được gửi đến controller vì không có flow cụ thể để xử lý.
  + Dành cho các trường hợp như ARP Request, DHCP, hoặc các gói tin yêu cầu xử lý đặc biệt.
* **Thống kê**:
  + **Packet Count và Byte Count**:
    - Một số dòng đã xử lý lưu lượng, ví dụ:
      * ipv4\_dst = 192.168.2.0/255.255.255.0 có **Packet Count = 1**, **Byte Count = 98**.
    - Các dòng khác hiện chưa xử lý (Packet Count = 0).
* **Ứng dụng**:
  + Flow này thường được sử dụng trên **switch (S1, S2)** để chuyển tiếp gói tin đặc biệt lên controller nhằm phân tích và xử lý.



* **Priority**: 1 (ưu tiên thấp).  **(Flow 1)**
* **Match Fields**:
  + **eth\_type = 2054:** Gói tin ARP.
* **Instructions**: OUTPUT:CONTROLLER - Gửi gói tin đến controller để xử lý.
* **Statistics**:
  + **Packet Count = 16**: Flow đã xử lý 16 gói tin ARP.
  + **Byte Count = 702**: Tổng số byte xử lý là 702.

**Chức năng chính**

* Xử lý gói tin **ARP Request/Reply**.
* Gửi ARP đến controller để quản lý bảng địa chỉ ARP hoặc phân tích thêm.
* **Priority**: 1 (ưu tiên thấp hơn các flow quản lý khác). **(Flow 1.1)**
* **Match Fields**:
  + **eth\_type = 2048:** Gói tin IPv4.
  + **65536**: Giá trị đặc biệt có thể chỉ định tất cả các gói tin chưa được phân loại.
* **Instructions**:
  + DEC\_NW\_TTL: Giảm giá trị TTL (đặc trưng của định tuyến).
  + SET\_FIELD: Thay đổi địa chỉ MAC nguồn (eth\_src) và đích (eth\_dst) để chuyển gói tin đúng nơi.
  + OUTPUT: Chuyển tiếp gói tin qua cổng OUTPUT1.
* **Statistics**:
  + Hiện tại chưa có gói tin nào được xử lý (Packet Count = 0).

**Chức năng chính:**

* Flow này chuẩn bị định tuyến các gói tin IPv4 chưa được xử lý hoặc chưa khớp với các flow cụ thể khác.
* Đảm bảo chuyển tiếp lưu lượng liên mạng qua các router hoặc Layer 3 Switch.



* **Priority**: 0 (thấp nhất, dùng cho flow mặc định).
* **Match Fields**:
  + ANY: Áp dụng cho tất cả các gói tin không khớp với bất kỳ flow nào khác.
* **Instructions**:
  + OUTPUT:NORMAL: Chuyển tiếp gói tin theo cách truyền thống (giống switch không quản lý SDN).
* **Statistics**:
  + **Packet Count = 0**: Hiện tại chưa có gói tin nào được xử lý bởi flow này.
  + **Byte Count = 0**: Chưa có lưu lượng dữ liệu nào đi qua.

**Chức năng chính**

* **Flow mặc định (Fallback Flow)**:
  + Flow này đảm bảo rằng tất cả các gói tin không khớp với bất kỳ flow cụ thể nào vẫn được xử lý.
  + Giúp tránh mất gói tin do thiếu flow phù hợp.
* **Xử lý theo cách truyền thống**:
  + Gói tin được chuyển tiếp qua switch mà không cần điều khiển từ controller, tương tự hoạt động của switch không sử dụng SDN.

### 9.3.2. Bảng flow 2

**Chức năng và nội dung flow tương tự như Flow 1 bên trên**

A screenshot of a computer

Description automatically generated

# KẾT QUẢ THỰC THI

Báo cáo chúng em triển khai thành công các kịch bản SDN với sự hỗ trợ của Ryu Controller, đảm bảo điều khiển lưu lượng mạng qua lại được giữa các VLAN và thấy tường lửa được thiết lập chặn được các goi ICMP và TCP SYN chạy thành công giúp tăng cường bảo mật và cải thiện khả năng kiểm soát lưu lượng. Tuy nhiên, chúng em chỉ thiết lập ở mức cấu hình hệ thống đơn giản, hệ thống cần được mở rộng và tối ưu hóa hơn nữa để thực hiện nhiều chức năng hơn và làm việc hiệu xuất hơn trong các mô hình mạng lớn và phức tạp hơn.

# NGUỒN THAM KHẢO

1. <https://mininet.org/> (Trang web mininet.org)
2. <https://book.ryu-sdn.org/en/html/> (Trang web về Ryu)
3. <https://en.wikipedia.org/wiki/Software-defined_networking> (Wikipedia về kiến thức tổng quan Ryu
4. Giáo trình môn học Quản Trị mạng và hệ thống Trường đại học Công nghệ thông tin.
5. T. D. Nadeau and K. Gray, SDN: Software Defined Networks, Sebastopol: O’Reilly Media Inc, 2013.
6. K. Benzekki, A. El Fergougui, & A. Elbelrhiti Elalaoui, “Software‐defined networking (SDN): a survey,” Security and communication networks.
7. O. Foundation, “Open Source Cloud Computing Infrastructure - OpenStack,” [Online]. Available: https://www.openstack.org/. [Accessed 23 March 2023].
8. T. A. S. Foundation, “Apache CloudStack: Open Source Cloud Computing,” 2020. [Online]. Available: https://cloudstack.apache.org/. [Accessed 23 March 2023].

BÀI BÁO CÁO KẾT THÚC