TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Xây dựng ứng dụng định tuyến đa miền hướng dịch vụ trong mạng SDN phân tán

NGỌ VIỆT HOÀNG
hoang.nv183748@sis.hust.edu.vn

Ngành: Công nghệ thông tin
Chuyên ngành: Kỹ thuật máy tính

Giảng viên hướng dẫn: TS. Trần Hải Anh _____

Chữ kí GVHD

Khoa: Kỹ thuật máy tính

Trường: Công nghệ thông tin và Truyền thông

HÀ NỘI, 03/2023

LỜI CẢM ƠN

Để hoàn thành đồ án này, em xin bày tỏ lòng biết ơn chân thành và sâu sắc nhất tới TS Trần Hải Anh, người đã định hướng đề tài và chỉ bảo chu đáo, theo sát em trong quá trình thực hiện đồ án.

Em xin cảm ơn TS Tống Văn Vạn và Ths Hoàng Nam Thắng và các bạn sinh viên thuộc trung tâm CNTT - Trường Đại học Xây Dựng Hà Nội đã tạo điều kiện và quan tâm giúp đỡ để em có thể hoàn thành được đồ án của mình.

Em xin cảm ơn Ban giám hiệu trường cùng toàn thể các thầy cô giáo trong trường Công nghệ thông tin và Truyền thông - Đại học Bách Khoa Hà Nội đã cung cấp, truyền đạt cho em những kiến thức quý báu trong quá trình học tập để em hoàn thành đồ án.

Cuối cùng, em xin chân thành cảm ơn gia đình, bạn bè những người đã giúp đỡ động viên em trong quá trình học tập.

Em xin chân thành cảm ơn!

TÓM TẮT NỘI DUNG ĐỒ ÁN

Sự tiến bộ của mô hình mạng mới nổi, mạng điều khiển phần mềm (SDN), đã đảm bảo chất lượng và yêu cầu đa dạng của các dịch vụ mạng nhờ kiến trúc tập trung và được định nghĩa bởi phần mềm. SDN không chỉ giúp cấu hình chính sách mạng cho kỹ thuật điều khiển lưu lượng mà còn mang lại tiện lợi cho việc lấy trạng thái mạng. Lưu lượng của nhiều dịch vụ khác nhau được truyền qua mạng, trong khi mỗi dịch vụ có thể đòi hỏi các chỉ số mạng khác nhau, chẳng hạn như độ trễ thấp hoặc tỷ lệ mất gói thấp. Chính sách chất lượng dịch vụ tương ứng phải được thực thi để đáp ứng các yêu cầu của các dịch vụ khác nhau cũng như đảm bảo sự cân bằng của việc sử dụng băng thông là rất quan trọng.

Trong nghiên cứu này, em xin đề xuất ứng dụng định tuyến đa miền hướng dịch vụ để tìm ra đường đi tối ưu cho mỗi dịch vụ trong mạng SDN đa miền với sự hỗ trợ của module phân loại dịch vụ sử dụng mô hình Học liên kết (Federated learning).

Ứng dụng lấy trạng thái mạng và loại dịch vụ làm giá trị đầu vào rồi từ đó cập nhật lại trọng số của các cạnh trên đồ thị mạng giúp các service tìm đường đi tốt nhất cho mình dựa trên thuật toán Dijstra. Các giao thức tùy chỉnh được thiết kế để lấy trạng thái mạng, và một mô hình phân loại lưu lượng dựa trên Học sâu cũng được tích hợp để xác định các dịch vụ mạng.

Kết quả thực nghiệm cho thấy RED-STAR có thể áp dụng cho môi trường mạng động, đạt giá trị phần thưởng trung bình cao nhất là 1,8579 và tỷ lệ sử dụng băng thông tối đa trung bình thấp nhất là 0,3601 so với tất cả các chế độ phân phối đường trong một kịch bản thực tế.

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ	**Ý nghĩa**
API	Giao diện lập trình ứng dụng (Application Programming Interface)
EUD	Phát triển ứng dụng người dùng cuối(End-User Development)
GWT	Công cụ lập trình Javascript bằng Java của Google (Google Web Toolkit)
HTML	Ngôn ngữ đánh dấu siêu văn bản (HyperText Markup Language)
IaaS	Dịch vụ hạ tầng

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

Chương 1 có độ dài từ 3 đến 6 trang với các nội dung sau đây

1.1 Đặt vấn đề

Trong mạng SDN (Software-Defined Networking), định tuyến đa miền là một chủ đề được quan tâm đến bởi các nhà phát triển mạng. Định tuyến đa miền cho phép các đường dẫn mạng được chia thành nhiều miền riêng biệt, mỗi miền có thể được quản lý và cấu hình độc lập với các miền khác. Điều này mang lại nhiều lợi ích cho mạng, bao gồm tăng cường tính tin cậy, hiệu suất và tính khả dụng.

Việc định tuyến giữa các miền là một thách thức lớn, các miền trong mạng SDN thường có các yêu cầu dịch vụ khác nhau và có thể có nhiều nhà cung cấp dịch vụ khác nhau đang hoạt động trong một miền mạng. Việc đảm bảo độ trễ thấp, băng thông đủ và đảm bảo chất lượng cho từng loại dịch vụ cho từng loại dịch vụ khác nhau là rất quan trọng. Tuy nhiên hiện nay cơ chế định tuyến đa miền hướng dịch vụ chưa được xem xét dẫn đến việc định tuyến đường đi không phù hợp cho các loại dịch vụ khác nhau trong mạng. Với cơ chế định tuyến hiện tại, tại một một thời điểm nhất định, các dịch vụ khác nhau có thể bị định tuyến trên cùng một đường và dẫn đến tắc nghẽn mạng, trong khi đó mỗi dịch vụ khác nhau lại yêu cầu các tham số QoS khác nhau để đảm bảo chất lượng dịch vụ, từ đó dẫn đến việc định tuyến không hiệu quả.

Từ đó tôi đề xuất một giải pháp về định tuyến đa miền hướng dịch vụ trọng mạng SDN, cụ thể trong nghiên cứu này đề cập đến ba dịch vụ phổ biến trên internet hiện nay là File Transfer, VoIP và Video Streaming; từ đó giúp cải thiện hiệu suất mạng đồng thời đảm bảo chất lượng dịch vụ của từng loại dịch vụ.

1.2 Các giải pháp hiện tại và hạn chế

Hiện nay, có nhiều giải pháp được đưa ra để giải quyết vấn đề định tuyến đa miền và định tuyến hướng dịch vụ trong mạng SDN.

Li và cộng sự [1] nhấn mạnh rằng các quản trị viên mạng đang gặp phải thách thức cung cấp dịch vụ tốt hơn với sự tăng trưởng nhanh chóng của 5G và dịch vụ đa phương tiện yêu cầu nhiều về chất lượng mạng (ví dụ: độ trễ thấp, độ tin cậy cao, v.v.). Tuy nhiên, các quản trị viên mạng không biết ứng dụng nào đang hoạt động trên mạng của họ, do đó họ không thể có được một cái nhìn toàn cầu để quản lý mạng một cách hiệu quả. Do đó, nhiều nghiên cứu tập trung vào định tuyến hướng dịch vụ (ví dụ: MPLS hướng dịch vụ, v.v.).

Peng và cộng sự [2] đề xuất một framework mạng hướng dịch vụ áp dụng

Segment Routing (SR) để đáp ứng yêu cầu SLA của họ. Khung mạng này xác định đặc tính dịch vụ thông qua cơ chế kiểm tra gói tin và sau đó chuyển tiếp gói tin vào các đường dẫn tương ứng. Tuy nhiên, các cơ chế SR này xác định các đường dẫn định tuyến thông qua sự can thiệp của con người, điều này không hiệu quả với sự thay đổi chưa từng có của môi trường mạng. Trái lại, cơ chế SR được đề xuất xác định các đường dẫn định tuyến bằng cách sử dụng RL để thích nghi với mạng động. Hơn nữa, việc phân loại lưu lượng bằng cơ chế kiểm tra gói tin không hiệu quả do lưu lượng mạng được mã hóa hiện nay. Do đó, cơ chế định tuyến hướng dịch vụ cần có một phương pháp phân loại lưu lượng để phân loại lưu lượng mạng được mã hóa và xác định lớp dịch vụ.

Trong [3] A. Chooprateep và cộng sự đề xuất Video Path Selection Algorithm (VPSA). Thuật toán đưa ra cơ chế route selection dựa trên một controller thu thập băng thông trên toàn mạng để truy cập video. Trong [4] Al-Jawad và cộng sự đề xuất intelligent QoS management framework được gọi là LearnQoS trên các ứng dụng đa phương tiện dựa trên SDN nhằm đảm bảo chất lượng video ổn định. LearnQoS được triển khai quản lý mạng để đảm bảo các yêu cầu QoS thông qua học tăng cường (gọi là PBNM). Trong [5] Kai-Cheng Chiu giới thiệu phương pháp định tuyến đa đường dựa trên nền tảng học tăng cường (Reinforcement Learning) và tập trung vào việc cải thiện chất lượng dịch vụ (Quality of Service) trong mạng. Ở bài báo này, tác giả tập trung vào việc phân tích trạng thái mạng và đưa ra các đường đi tối ưu với các dịch vụ khác nhau. Tuy nhiên các nghiên cứu trên mới được kiểm nghiệm trên một mạng SDN đơn lẻ và với quy mô cỡ nhỏ thay vì sử dụng mạng SDN phân tán cỡ lớn.

1.3 Mục tiêu và định hướng giải pháp

Mục tiêu chính của đồ án là thiết kế một giải thuật định tuyến đa miền dựa trên mô hình học máy đã có sẵn giúp cải thiện hiệu năng mạng và hiệu năng của các dịch vụ khác nhau trong mạng. Giải pháp tôi đề xuất trong đồ án này sẽ giải quyết hạn chế của vấn đề định tuyến cho nhiều dịch vụ khác nhau trong mạng bằng một thuật toán định tuyến mới, có khả năng tìm đường đi tối ưu cho các loại dịch vụ khác nhau. Ngoài ra giải pháp tôi đề xuất trong đồ án này cũng được triển khai trên nhiều mạng SDN không đồng nhất và có ở ngoài thực tế, từ đó có thể đưa ra những kết quả thực nghiệm chính xác hơn.

1.4 Đóng góp của đồ án

Đồ án này có 2 đóng góp chính như sau:

1. Đồ án đề xuất và triển khai một module có nhiệm vụ bắt gói tin, xử lý gói tin, đưa vào module phân loại dịch vụ để tìm ra loại dịch vụ đang được sử dụng.

2. Đồ án đề xuất và triển khai giải thuật và các hàm mục tiêu mới ứng với mỗi dịch vụ dựa trên tình trạng mạng trên toàn bộ các miền thỏa mãn yêu cầu về QoS ứng với mỗi dịch vụ

1.5 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau.

Chương 2 trình bày về các lý thuyết nền tảng được sử dụng trong đồ án bao gồm: SDN, SINA, Federated Learning và kiến trúc mạng Knowledge-defined Heterogeneous

Chương 3 khảo sát về sự ảnh hưởng của các tham số QoS đến chất lượng các dịch vụ và đề xuất ra các hàm mục tiêu ứng với mỗi dịch vụ để tối ưu hóa đường đi

Chương 4 sẽ đánh giá về hiệu quả của các hàm mục tiêu dựa vào hai tham số là link utilization và respone time.

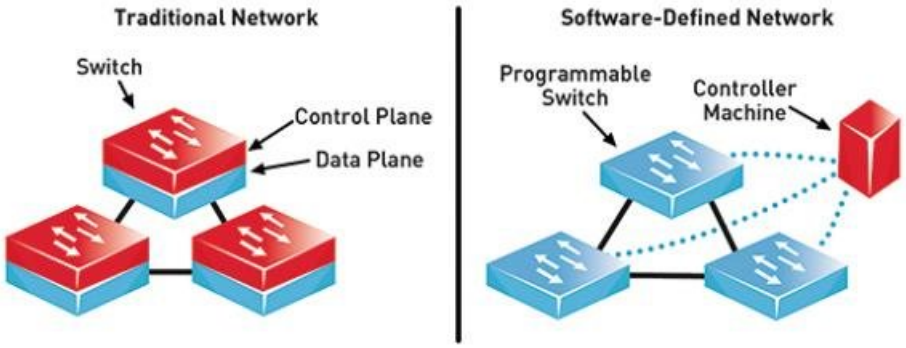
Chương 5 sẽ đưa ra kết luận cho kết quả đạt được trong đồ án và định hướng phát triển cho bài toán trong tương lai.

CHƯƠNG 2. NỀN TẢNG LÝ THUYẾT

2.1 Software-defined Networking (SDN)

2.1.1 Khái niệm

Mạng SDN (Software-Defined Networking) là một kiến trúc mạng linh hoạt, được thiết kế để tách biệt các lớp phần cứng và phần mềm trong mạng, cho phép quản trị viên mạng điều khiển và cấu hình mạng tập trung hơn. Kiến trúc này phân tách phần điều khiển mạng (Control Plane) và Forwarding Plane (hay Data Plane - chức năng vận chuyển dữ liệu), điều này cho phép việc điều khiển mạng trở nên có thể lập trình được dễ dàng và cơ sở hạ tầng mạng độc lập với các ứng dụng và dịch vụ mạng". Sau đây là hình ảnh so sánh giữa mạng truyền thống và mạng SDN:



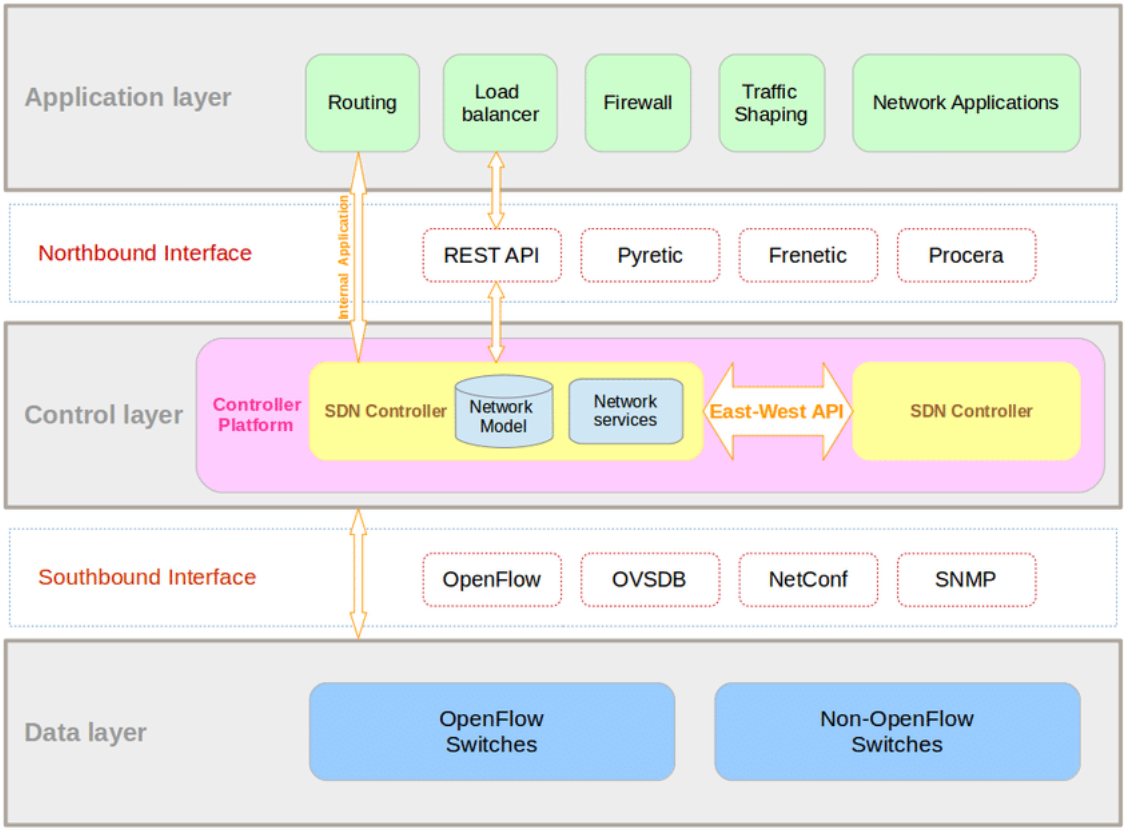
Hình 2.1: So sánh giữa mạng truyền thống và mạng SDN

Trong thiết bị mạng truyền thống, phần điều khiển (Control Plane) đều được tích hợp với phần vận chuyển dữ liệu (Data Plane) trong cùng một thiết bị mạng. Trong mạng SDN, phần điều khiển được tách riêng khỏi thiết bị mạng và được xử lý tập trung tại một phần mềm được gọi là controller SDN. Về thu thập và xử lý các thông tin: Đối với mạng truyền thống, các thiết bị mạng thực hiện độc lập việc nhận thông tin các thành phần trong mạng cũng như định tuyến việc gửi gói tin. Với hệ thống mạng lớn thì thời gian hội tụ mạng (thời gian các node trong mạng hiểu biết về thông tin toàn bộ hoặc một phần mạng) mất rất nhiều thời gian. Trong mạng SDN, phần này được tập trung xử lý ở controller, giúp SDN kiểm soát thông minh các tài nguyên mạng phía dưới. Về khả năng lập trình mạng, mạng truyền thống không thể được lập trình bởi các ứng dụng tự phát triển. Các thiết bị mạng phải được cấu hình một cách riêng lẻ và thủ công. Trong khi đối với mạng SDN, mạng có thể lập trình bởi các ứng dụng, bộ điều khiển SDN có thể tương tác đến tất cả các thiết bị trong mạng thông qua bộ API mở Về khả năng lập trình mạng, mạng truyền thống không thể được lập trình bởi các ứng dụng tự phát triển. Các thiết bị

mạng phải được cấu hình một cách riêng lẻ và thủ công. Trong khi đối với mạng SDN, mạng có thể lập trình bởi các ứng dụng, bộ điều khiển SDN có thể tương tác đến tất cả các thiết bị trong mạng thông qua bộ API mở.

2.1.2 Kiến trúc

Kiến trúc mạng SDN gồm ba tầng chính: tầng ứng dụng (Application plane), tầng điều khiển (Control plane) và tầng dữ liệu (Data plane)



Hình 2.2: Kiến trúc tổng quan của SDN

Lớp ứng dụng gồm các ứng dụng hoặc chức năng mạng mà các tổ chức muốn sử dụng, có thể gồm các hệ thống phát hiện xâm nhập, cân bằng tải hoặc tường lửa. Trong một mạng truyền thống sẽ sử dụng các thiết bị chuyên dụng, như tường lửa hoặc cân bằng tải, SDN sẽ thay thế thiết bị đó bằng một ứng dụng sử dụng bộ điều khiển để quản lý "hành vi" của data plane. Lớp điều khiển đại diện cho phần mềm điều khiển SDN tập trung hoạt động như bộ não của SDN. Tầng điều khiển nằm trên một máy chủ và có khả năng quản lý các chính sách và luồng dữ liệu trên toàn bộ hệ thống mạng. Giữa lớp ứng dụng và lớp điều khiển là các API cầu bắc (Northbound) chịu trách nhiệm giao tiếp giữa hai lớp. Tương tự, giao tiếp giữa lớp điều khiển và lớp hạ tầng do các API cầu nam (southbound) đảm nhiệm. Nhắc đến API cầu nam ta nhắc tới OpenFlow. OpenFlow là một tiêu chuẩn mở cho giao thức

truyền thông được ONF đưa ra năm 2012, cho phép lớp điều khiển tương tác với lớp hạ tầng. Cần lưu ý rằng OpenFlow không phải là giao thức duy nhất có sẵn hoặc đang được phát triển cho SDN. Lớp cơ sở hạ tầng được tạo thành từ các thiết bị vật lý trong mạng. Bao gồm các thiết bị vật lý (phần cứng) hoặc (và) các phần mềm chuyển mạch (phần mềm).

2.1.3 Đặc điểm

Các đặc điểm của mạng SDN bao gồm:

- Tách rời lớp điều khiển và lớp chuyển mạch: trong mạng SDN, lớp điều khiển và lớp chuyển mạch được tách rời nhau. Lớp điều khiển được quản lý bởi một bộ điều khiển trung tâm (SDN Controller) để điều khiển và quản lý lớp chuyển mạch.
- Trung tâm hóa quản lý mạng: SDN Controller trung tâm hóa quản lý mạng, cho phép quản lý mạng một cách trung tâm, tập trung hơn và nhanh chóng hơn.
- Tính linh hoạt và dễ dàng cấu hình: Mạng SDN cho phép cấu hình linh hoạt, có thể điều chỉnh tốc độ và dịch vụ mạng một cách nhanh chóng và đáp ứng các yêu cầu của ứng dụng một cách tốt hơn.
- Tính độc lập phần cứng: Mạng SDN giúp tách biệt phần cứng và phần mềm trong mạng, giúp đơn giản hóa quản lý mạng và cho phép sử dụng các thiết bị mạng của nhiều nhà sản xuất khác nhau.
- Tính mở và tiên tiến: Mạng SDN hỗ trợ các giao thức mở, cho phép tích hợp với các công nghệ mới nhất để cải thiện hiệu suất và tính bảo mật của mạng.

2.1.4 Ứng dụng

Mạng SDN (Software-Defined Networking) đã được triển khai và sử dụng rộng rãi trong thực tế, đặc biệt trong các doanh nghiệp, tổ chức và công ty lớn. Các ứng dụng của mạng SDN có thể được chia thành các lĩnh vực khác nhau, bao gồm truyền thông, điện toán đám mây, các ứng dụng IoT, hệ thống viễn thông và các hệ thống quản lý mạng.

Trong lĩnh vực truyền thông, mạng SDN được sử dụng để cải thiện khả năng chịu tải và tăng cường hiệu suất của mạng. Ví dụ, một công ty sản xuất nội dung có thể sử dụng mạng SDN để tối ưu hóa lưu lượng video và cải thiện trải nghiệm người dùng. Một hệ thống SDN cung cấp khả năng tối ưu hóa đường truyền dữ liệu, đảm bảo rằng dữ liệu được chuyển tới người dùng một cách nhanh chóng và đáp ứng các yêu cầu của ứng dụng.

Các ứng dụng IoT là một lĩnh vực khác mà mạng SDN được sử dụng. Với số lượng thiết bị IoT ngày càng tăng, SDN giúp quản lý và định tuyến lưu lượng dữ liệu đám mây, cung cấp dịch vụ mạng phù hợp với các yêu cầu của các thiết bị IoT và đáp ứng nhu cầu truyền thông giữa các thiết bị IoT.

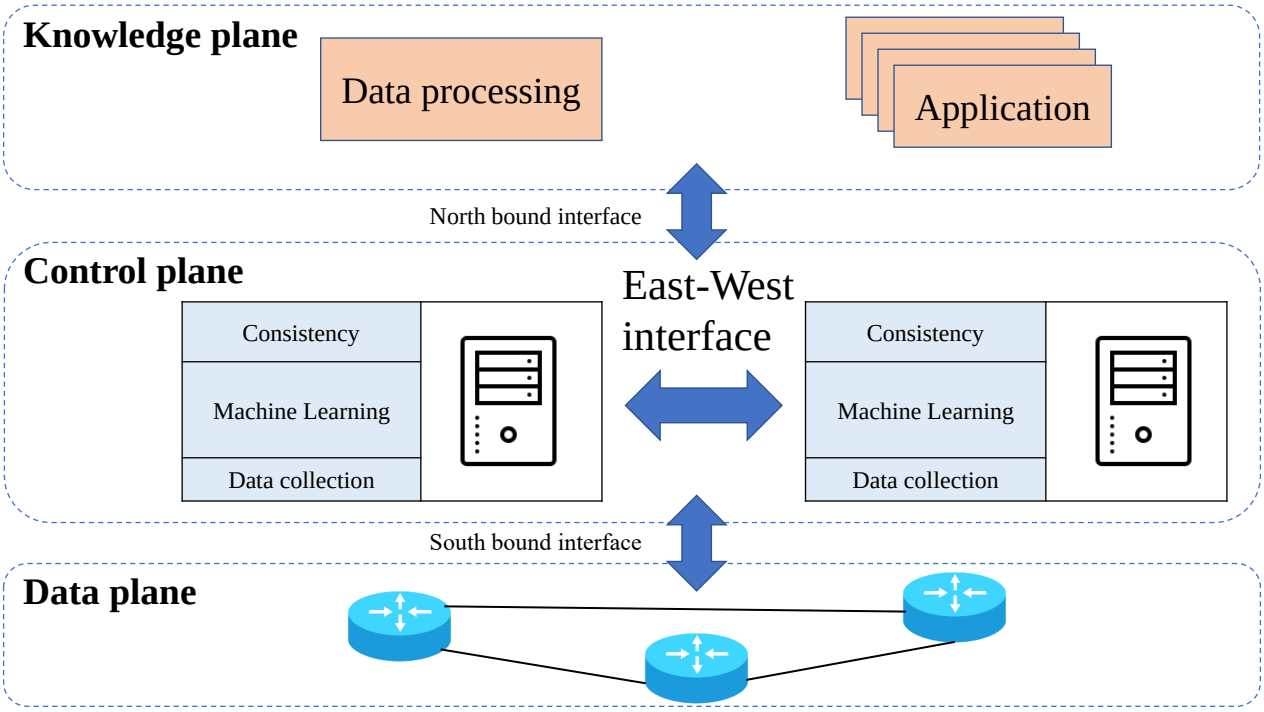
Các doanh nghiệp và tổ chức cũng sử dụng mạng SDN để quản lý mạng. Mạng SDN cung cấp khả năng quản lý trung tâm và tập trung hơn, giúp các quản trị viên mạng dễ dàng cấu hình và giám sát hệ thống mạng. Nó cũng cung cấp khả năng theo dõi và phát hiện các cuộc tấn công mạng, giúp các tổ chức bảo vệ hệ thống mạng của họ.

Trong lĩnh vực hệ thống viễn thông, mạng SDN giúp tối ưu hóa hoạt động của các mạng di động và mạng cố định. Nhờ vào sự linh hoạt và dễ dàng điều khiển của mạng SDN, các nhà cung cấp dịch vụ viễn thông có thể tùy chỉnh mạng của mình theo nhu cầu và đáp ứng nhu cầu sử dụng của khách hàng một cách nhanh chóng và hiệu quả hơn. Ngoài ra, mạng SDN còn có khả năng quản lý lưu lượng mạng một cách thông minh và đáp ứng các yêu cầu của ứng dụng đa dạng, từ video trực tuyến đến các ứng dụng đòi hỏi độ trễ thấp như trò chơi trực tuyến. Mạng SDN cũng đang được sử dụng trong các hệ thống viễn thông mới như mạng 5G. Mạng 5G đòi hỏi một hạ tầng mạng linh hoạt và có thể tùy biến để đáp ứng các yêu cầu đa dạng từ các ứng dụng khác nhau. Mạng SDN cho phép các nhà cung cấp dịch vụ viễn thông cung cấp một mạng 5G hiệu quả hơn, đáp ứng các yêu cầu đa dạng từ khách hàng và tối ưu hóa hoạt động của mạng một cách nhanh chóng và linh hoạt.

Với những ứng dụng đa dạng và tiềm năng trong thực tế, SDN đang trở thành một công nghệ quan trọng trong các hệ thống mạng hiện đại. Các công ty và tổ chức đang tích cực triển khai SDN để tăng tính linh hoạt, tính bảo mật và tối ưu hóa hiệu suất của hệ thống mạng của mình.

2.2 Knowledge-defined Heterogeneous Network (KHN)

Tổng quan kiến trúc mạng KHN được mô tả trong hình 3.2. Gồm 3 lớp: Data Layer, Control layer và Knowledge layer. Với North Bound API và South Bound API là cầu nối giao tiếp giữa các lớp.



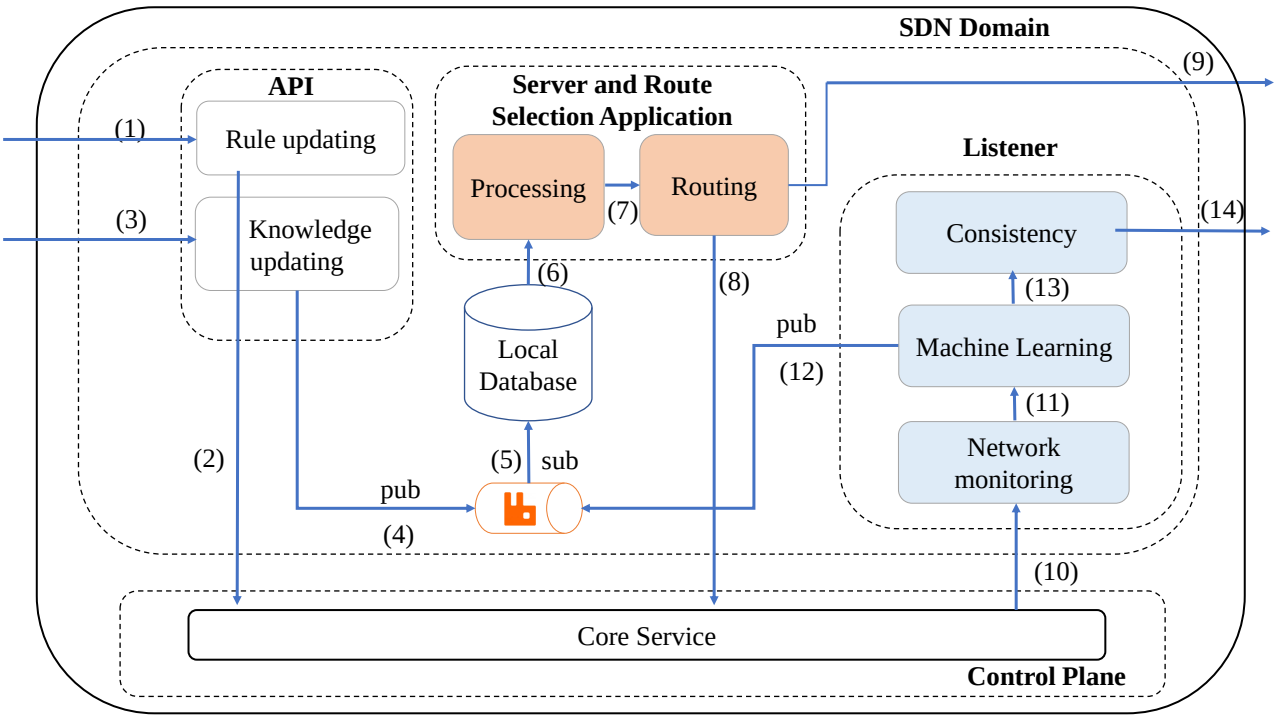
Hình 2.3: Knowledge-defined heterogeneous network architecture.

Đầu tiên, Data layer là mặt phẳng chứa các thiết bị chuyển mạch vật lý và các thiết bị mạng khác. Data layer có trách nghiệm lắng nghe điều hướng từ controller layer và tiến hành vận chuyển các gói tin. Thứ hai, Control layer là một mặt phẳng chứa nhiều SDN controllers phân tán không đồng nhất. ONOS là một hệ điều hành mạng mã nguồn mở viết bằng java được phát triển bởi ON.Lab (Open Networking Lab) và cộng đồng các nhà phát triển toàn cầu. ONOS cung cấp các chức năng quản lý, điều khiển và triển khai mạng tập trung và phân tán, cung cấp một cách tiếp cận khác biệt đối với các mạng hiện đại. Các controllers có thể giao tiếp với nhau thông qua một cầu nối giao tiếp East-West interface và consistency mechanism để đồng bộ trạng thái của chúng [6]. Thông qua cầu nối này, các controllers có khả năng lan truyền dữ liệu cho nhau sau khi thu thập dữ liệu trên chính mạng của nó (data colllection). Tuy nhiên, bảo mật và độ tin cậy của dữ liệu là hai vấn đề phát sinh khi ta tăng cường khả năng trao đổi thông tin trong một hệ thống phân tán. Đối với hệ thống có rất nhiều luồng hoạt động trong cùng 1 thời điểm như SDN thì rất khó để biết xem thông tin trạng thái của mạng có bị "nghe lén" hay không. Chính vì thế để cải thiện khả năng bảo mật, chúng tôi không lan truyền toàn bộ thông tin trạng thái của mạng, mà dữ liệu đó được đưa qua mô hình ML để trích xuất tri thức. Sau đó, cơ chế consistency sẽ đảm bảo các controllers đều nhận biết được tri thức từ nhiều vùng mạng khác nhau.

Để làm rõ hơn về cơ chế đồng bộ tri thức, sau đây chúng tôi sẽ mô tả về cầu nối East-West interface và consistency mechanism. Đầu tiên, cầu nối này còn gọi là

"SDN-inter network application (SINA)". SINA gồm 2 thành phần chính như sau: Listener component và API component. Ban đầu, Listener component có trách nhiệm lắng nghe các sự kiện từ mặt phẳng liên kết dữ liệu nhằm thực hiện các hành động phù hợp ngay sau những sự kiện đó và thông báo về sự thay đổi tới các Mạng SDN khác. Về API component, thành phần này được xây dựng một nền tảng REST API cho phép các SDN khác có thể gọi tới để cập nhật dữ liệu. Các API này được xây dựng dựa trên các East/West API. Dựa trên khả năng giao tiếp đa nền tảng của SINA, chúng tôi triển khai thêm giải thuật có khả năng thống nhất dữ liệu của nhiều mạng SDN phân tán. Thuật toán này có khả năng thống nhất dữ liệu một cách thích nghi (adaptive consistency) dựa theo trạng thái mạng tại thời điểm hiện tại. Thuật toán được dựa trên cơ chế quorum replication và reinforcement-learning. Nói một cách ngắn gọi, ta gọi N, N_r, N_w lần lượt là số lượng controllers, read-quorum value và write-quorurm value. Ví dụ, khi có sự thay đổi mạng của 1 SDN, controller thuộc SDN này sẽ forward dữ liệu của nó tới N_w controllers khác. Tương tự như vậy, khi cần tổng hợp dữ liệu toàn mạng, 1 controller có khả năng đọc dữ liệu từ N_R controllers khác. Rõ ràng nếu $N_w + N_r > N$, tức là tổng số bản ghi và đọc luôn lớn hơn số controllers, ta có thể luôn đảm bảo consistency của mạng, đây còn gọi là strong consistency model. Tuy nhiên, để duy trì mô hình này, ta sẽ phải đánh đổi hiệu năng của mạng do chi phí đồng bộ lớn. Chính vì vậy, giải thuật Reinforcement-learning[xxx] được chúng tôi sử dụng, có khả năng lựa chọn bộ giá trị cho N_w và N_r một cách thích nghi dựa theo trạng thái của mạng. N_w và N_r sẽ được RL algorithm chọn sao cho cho thỏa mãn $N_w + N_r < N$ nhưng vẫn cân bằng được tính consistency và giảm chi phí đồng bộ trạng thái giữa các mạng.

Cuối cùng, là tầng Knowledge layer, tại tầng này các tri thức của bộ mạng phân tán sau khi được đồng nhất sẽ được tổng hợp và tích hợp vào các ứng dụng cụ thể (data processing). Trong bài báo này chúng tôi sẽ xem xét ứng dụng "server and route selection", để làm rõ các thành phần của ứng dụng, hình 2.4 mô tả chi tiết các luồng dữ liệu và các modules được tích hợp trong mỗi SDN inter-domains.

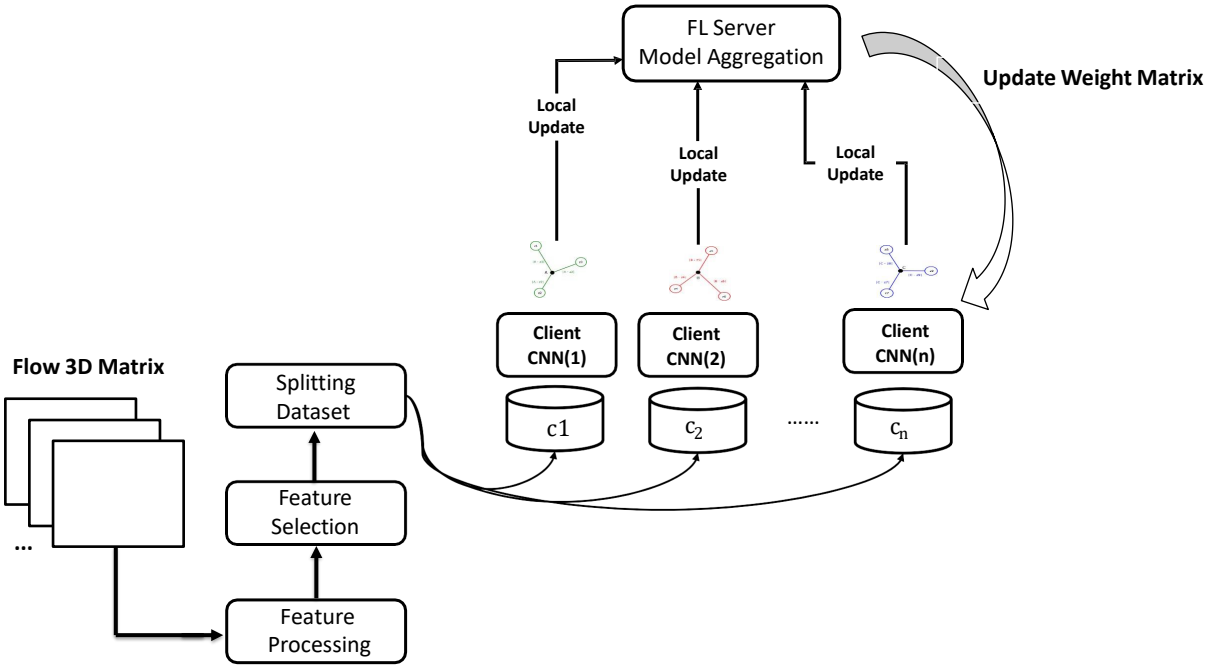


Hình 2.4: Knowledge-defined heterogeneous network architecture.

Trong thành phần API bao gồm 2 modules "Update rule" và "Topology state updating, Link cost" được xây dựng trên nền tảng REST API cho phép các SDN khác có thể gọi tới để cập nhật dữ liệu. SDN domain nhận các chính sách và quy tắc luồng từ controller khác thông qua (flow 1). Tiếp theo đó, giao thức OpenFlow (flow 2) sẽ gửi các rule đến data plane và các thiết bị chuyển mạch sẽ xử lý gói tin dựa trên những rule này. Thành phần "Topology state updating, Link cost" được sử dụng để cập nhật thông tin topology từ các SDN domain khác (flow 3) giúp đảm bảo tính nhất quán trong toàn bộ các SDN domain. Với hệ thống xử lý mạng phân tán cỡ lớn, dữ liệu từ các SDN domain khác sẽ được lắng nghe liên tục với khối lượng lớn, để tránh mất mát thông tin khi xử lý chúng tôi sử dụng cơ chế pub/sub và xử lý dữ liệu nhận được một cách tuần tự. Khi dữ liệu được gửi tới, chúng tôi sẽ đẩy vào một Queue (flow 4), khi có yêu cầu xử lý, dữ liệu sẽ được lấy từ Queue cập nhật vào local network state database (flow 5) - nơi lưu trữ thông tin của toàn bộ mạng. Để tối ưu các tham số QoS, dữ liệu trọng số của các links được lấy từ local database (flow 6) sẽ giúp quyết định đường, quy tắc luồng cho thành phần Routing (flow7). Giống như flow 2, các quy tắc sẽ được gửi tới mặt phẳng liên kết dữ liệu bằng giao thức OpenFlow (flow 8) và gửi tới các SDN Domain khác quản lý các switches nằm trên đường đi đó (flow 14). Trong thành phần Listener, Network monitoring module sẽ tiến hành lắng nghe những sự kiện của mạng local từ Core Service của Control Plane (flow 10). Trong khoảng 3s, module này sẽ thu thập các tham số QoS trên từng cạnh của mạng local. Các tham số này đóng vai

trò là những đặc trưng của mô hình ML huấn luyện dự đoán Link cost (flow 11), mô hình được chúng tôi mô tả trong phần xxx. Đầu ra của module "Link cost prediction" được chúng tôi đẩy vào local database (flow 9) và chuyển tới module "Consistency" (flow 12) để thực hiện truyền bá cho các SDN domain khác thông qua East/West Interface (là một giao diện cho phép các SDN domains khác nhau có khả năng trao đổi thông tin với nhau thông qua các API) (flow 13). Việc truyền bá link cost cho các SDN domains khác thể hiện tính bảo mật, tránh làm lộ các thông tin của một SDN domain trong quá trình lan truyền. Việc sử dụng ML, ta có thể trích xuất được quan hệ từ lượng dữ liệu lớn trong các SDN domains. Sau đó, ta có thể dùng tri thức này để làm đầu vào cho các giải thuật server selection routing để đưa ra quyết định tốt hơn, giúp giá trị của các tham số QoS trên links ở ngưỡng phù hợp trong tương lai.

2.3 Học liên kết (Federated Learning)



Hình 2.5: Kiến trúc tổng quan Federating Learning cho bài toán phân loại dịch vụ.

Như đã mô tả ở Hình 3.2 và Hình 2.4, module Machine Learning đóng vai trò quan trọng để khai phá tri thức từ tập dữ liệu trong mạng. Trong đồ án này, tôi sẽ ứng dụng mô hình mạng tích chập sâu, Deep Convolutional Network (D-CNN) để học các đặc tính của các gói tin, từ đó phân loại được dịch vụ mà người dùng đang sử dụng. Từ dịch vụ được phân loại, module định tuyến Routing sẽ cập nhập lại đường đi tối ưu cho từng dịch vụ yêu cầu. Ngoài ra, để tăng tốc huấn luyện mô hình Machine Learning với tài nguyên phần cứng hạn chế, tôi sẽ triển khai kỹ thuật học phân tán (Federated learning) để huấn luyện mô hình CNN với bộ dữ liệu

lớn. Trong các phần tiếp theo, tôi sẽ lần lượt mô tả lí thuyết và cách hoạt động của Federated Learning.

2.3.1 Cơ sở lý thuyết kỹ thuật học liên kết (Federated Learning)

Trong phần này, tôi sẽ mô hình hóa tóm gọn kỹ thuật FL. Ta xét N bộ tính toán. Các bộ tính toán này $\{C_1, C_2, ..., C_N\}$ có thể là bộ tính toán của từng nhà cung cấp dịch vụ hay đơn giản chỉ là các servers tính toán trên miền cục bộ của nó. Vì trong đồ án này, mô hình mạng tôi sử dụng là SDN phân tán, nên mỗi controller SDN cục bộ sẽ có nhiệm vụ quản lý và huấn luyện mô hình D-CNN trên từng miền dữ liệu $\{D_1, D_2, ..., D_N\}$

Trong mô hình huấn luyện tập trung (centralized Machine Learning model), mô hình Machine Learning sẽ suy diễn bằng việc **gộp tất cả bản ghi dữ liệu**:

$$D = D_1 \cup D_2 \cup D_N = \bigcup_{i=1}^{N} D_i \tag{2.1}$$

Kết quả đầu ra sẽ đươc một model, ký hiệu là M_{sum}. Tuy nhiên, trong mô hình FL, quá trình học sẽ bao gồm từng miền controllers sẽ **cùng nhau** huấn luyện một model, ký hiệu là M_{fed}. Ngoài ra, từng miền tính toán C_i không cần nhất thiết phải chia sẻ bản ghi dữ liệu D_i của chính nó cho các miền khác. Điều này, sẽ giúp tăng cường bảo mật dữ liệu khi ngày nay các nhà cung cấp dịch vụ có xu hướng không chia sẻ dữ liệu của họ một cách công khai.

Mục tiêu của mô hình FL M_{fed} là đạt được độ tới độ chính xác trong việc phân loại, ký hiệu là Acc_{fed} sao cho xấp xỉ được độ chính xác của mô hình tập trung M_{sum}, ký hiệu là Acc_{sum}. Hay nói cách khác, nếu ta gọi Δ là một số thực không âm, thì ta coi độ mất mát khi huấn luyện của mô hình FL M_{fed} là:

$$\|Acc_{fed} - Acc_{sum}\| < \Delta \tag{2.2}$$

2.3.2 Ứng dụng kỹ thuật học liên kết (Federated Learning) cho bài toán phân loại dịch vụ

Nếu ta gọi ma trận D_i là từng bộ dữ liệu của SDN controller thứ i, trong đó từng hàng của D_i được coi là một mẫu dữ liệu, từng cột của D_i được coi là một đặc trưng dữ liệu. Giả sử M là số lượng mẫu, N là số lượng đặc trưng, ta coi $X \in R^{M \times N}$ là chiều không gian của bộ dữ liệu D và $X_i \in R^{M \times 1}$ là vector cột của một đặc trưng

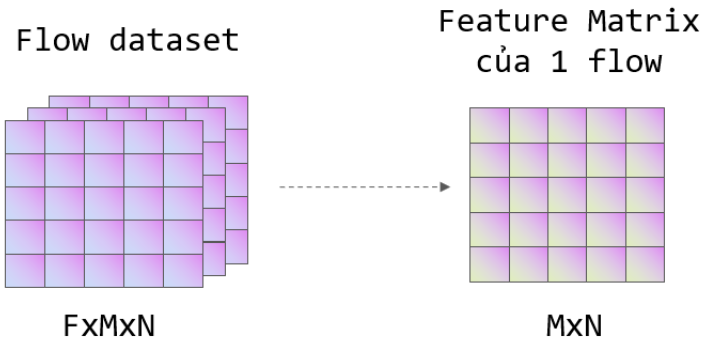
dữ liệu, ID_i là id của một mẫu thứ i. Trong FL, ta có thể sử dụng chiến lược chia tập dữ liệu D theo chiều dọc (Vertical splitiing) cho các clients C_i [xxx], tức là chia dữ liệu sao cho $X_i = X_j$, $ID_i \neq ID_j$,$\forall D_i, D_j$, $i \neq j$. Hình 2.6 minh họa cách chia dữ liệu theo chiều dọc trong FL.

	Client C1				Client C2		
	X_1	X_2	X_3		X_1	X_2	X_3
ID_1				ID_4			
ID_2				ID_5			
ID_3				ID_6			

Hình 2.6: Minh họa chia tập dữ liệu cho nhiều Clients theo chiều dọc trong Federated Learning.

	Flow i			
	Byte_1	Byte_2	...	Byte_N
Packet_1				
Packet_2				
...				
Packet_M				

Hình 2.7: Minh họa ma trận đặc trưng của mỗi flow.



Hình 2.8: Minh họa tensor 3 chiều đầu vào.

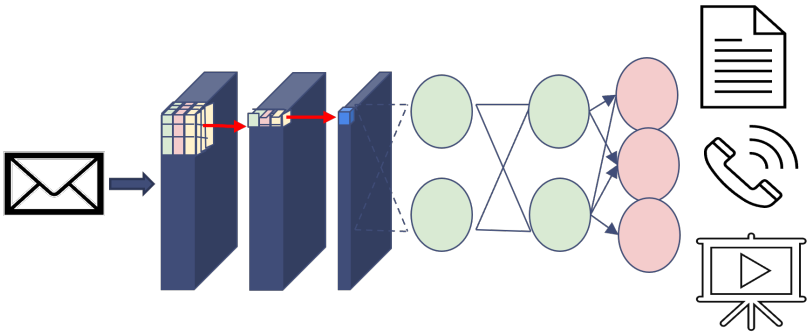
a, Thu thập dữ liệu

Trong việc thu thập dữ liệu nhằm mục đích phân loại dịch vụ, có 2 phương pháp thu thập dữ liệu phổ biến đó là Flow-based và Packet-based. Đối phương pháp Flow-based, ta có thể biểu diễn mỗi hàng của ma trận D là một flow, với các cột $X_1, ...X_{N-1}$ là đặc trưng của flow đó: thời gian flow đó được yêu cầu, thời gian sống của flow, số lượng packet trong flow, hay số bytes được truyền trên giây. Cột thứ X_N sẽ là nhãn dịch vụ của flow.

Tuy nhiên, trong đồ án này, tôi sẽ sử dụng phương pháp phổ biến hơn đó là Packet-based. Với việc sử dụng phương pháp này, số lượng chiều của dữ tập dữ liệu

sẽ lớn hơn bởi vì bởi vì tôi sẽ trích xuất từng byte trên một packet và coi từng byte này tương đương với 1 đặc trưng. Hình 2.7 minh họa ma trận đặc trưng của mỗi flow, với số hàng là số gói tin thuộc flow, và số cột là số byte của gói tín tin đó. tôi sẽ trích xuất mỗi packet 256 byte, tương đương với 256 cột. Nếu packet nào không đủ byte, thì tôi sẽ thêm padding, tức là thêm giá trị 0 cho các cột trống. Lưu ý rằng, mỗi flow này đều có nhãn là dịch vụ được yêu cầu.

Nếu biểu diễn mỗi một flow có ma trận đặc trưng như hình 2.7, ta có thể ghép nhiều flow lại thành một tensor 3 chiều như hình 2.8. Trong đó: F là số lượng flows, M là số lượng packet thuộc một flow và N số byte thuộc từng packet.



Hình 2.9: Minh họa mô hình mạng tích chập sâu trong việc phân loại các gói tin.

Việc biểu diễn các packets đầu vào dưới dạng tensor như này rất có lợi bởi vì thực chất kiểu dữ liệu của chúng rất tương đồng với dữ liệu dạng ảnh, với các giá trị byte từ 0 đến 255. Vì vậy, tôi có thể tận dụng mô hình mạng tích chập học sâu D-CNN, một mô hình phổ biến trong xử lý ảnh để học các đặc trưng của packet, từ đó phân loại được dịch vụ. Sau khi phân loại được mỗi packet trong một flow thuộc dịch vụ gì, tôi sẽ quyết định xem flow đó thuộc nhãn nào bằng cách tính tần suất xem số lượng packet được gãn nhãn nào nhiều nhất.

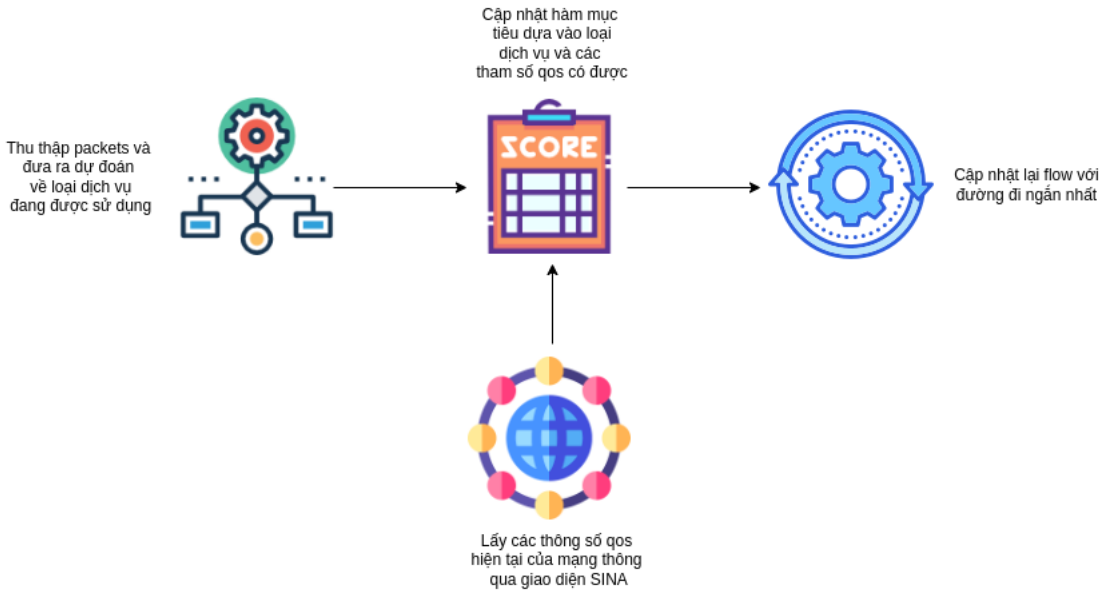
b, Kiến trúc tổng quan phương pháp học liên kết (Federated Learning) cho bài toán phân loại dịch vụ

Trong kiến trúc FL như được mô tả ở hình 2.5, đầu vào là một 3D tensor, với mỗi ma trận 2D đại diện cho một ma trận flow chưa các đặc trưng của packet thuộc flow đó. Đầu vào 3D tensor này sẽ được qua bước tiền xử lý như: loại bỏ nhiễu, loại bỏ missing values, chuẩn hóa về 0-1 với phương pháp Min-Max scaling. Sau đó, dữ liệu được làm sạch sẽ được chia cho N máy clients khác nhau, mỗi client sẽ thực hiện việc huấn luyện với mô hình D-CNN như ở hình 2.9 với dữ liệu của chính nó, sau đó cập nhật trọng số mô hình lên server. Server tiến hành tính toán và các trọng số được đẩy lên và cập nhật lại cho các clients. Quá trình này được lặp lại cho tới khi kết thúc số lần lặp ở mỗi mô hình D-CNN thuộc clients.

CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT

3.1 Tổng quan giải pháp

Để tìm ra được đường đi tối ưu nhất cho mỗi dịch vụ dựa vào các thông số QoS, hệ thống được thiết kế gồm 3 bước chính như hình 3.10. Trong bước thứ nhất, module phân loại sẽ bắt các gói tin đang lưu thông trong mạng, xử lý các gói tin để biến đổi thành ma trận đặc trưng của gói tin rồi đưa vào mô hình học liên kết đã có để dự đoán ra kiểu dịch vụ đang được sử dụng. Bước thứ hai, sau khi dự đoán dịch vụ đang được sử dụng, module phân loại dịch vụ sẽ gửi một API đến ứng dụng định tuyến để ứng dụng cập nhật lại trọng số của đồ thị mạng dựa trên hàm mục tiêu của dịch vụ vừa dự đoán được. Bước thứ ba, sau khi trọng số của đồ thị đã được cập nhật dựa trên dịch vụ được dự đoán, ứng dụng định tuyến sẽ tìm ra đường đi ngắn nhất từ client đến server và gửi yêu cầu cập nhật lại đến SDN controller để cập nhật chính sách đường đi cho luồng dữ liệu.



Hình 3.1: Tổng quan giải pháp

3.2 Ảnh hưởng của các tham số QoS đến các dịch vụ File Transfer, VoIP và Video Streaming

Mất gói tin, độ trễ và sử dụng liên kết có thể ảnh hưởng đáng kể đến các dịch vụ trên mạng internet và có thể dẫn đến hiệu suất giảm, độ tin cậy giảm. Packet loss đề cập đến việc mất các gói dữ liệu đang được truyền trên mạng. Điều này có thể xảy ra do tắc nghẽn, lỗi hoặc các vấn đề khác trên mạng. Delay đề cập đến thời gian mà các gói dữ liệu mất để được truyền từ một điểm đến điểm khác trên mạng. Độ trễ này có thể do các yếu tố như tắc nghẽn mạng, khoảng cách giữa các điểm

kết nối mạng và thời gian xử lý cần thiết của các thiết bị mạng. Link utilization đề cập đến lượng băng thông mạng đang được sử dụng bởi các gói dữ liệu tại bất kỳ thời điểm nào.

3.2.1 File Transfer

Trong dịch vụ truyền tải tập tin, mất gói tin, độ trễ và sử dụng liên kết có thể ảnh hưởng đến tốc độ truyền tải và độ tin cậy của quá trình truyền tải. Khi mất gói tin xảy ra, tốc độ truyền tải sẽ bị giảm và có thể làm gián đoạn quá trình truyền tải tập tin. Việc mất gói tin có thể xảy ra do đường truyền không ổn định hoặc do sử dụng băng thông quá tải. Độ trễ cũng có thể làm giảm tốc độ truyền tải tập tin và gây ra sự chậm trễ trong quá trình tải về. Điều này đặc biệt đáng chú ý trong việc tải về các tệp lớn hoặc khi truyền tải thông tin qua các khoảng cách xa. Độ trễ có thể được gây ra bởi sự cạnh tranh về băng thông và sử dụng liên kết cao giữa các người dùng khác nhau. Ngoài ra, sử dụng liên kết cao cũng có thể dẫn đến tình trạng cạnh tranh băng thông giữa các người dùng và làm giảm tốc độ truyền tải. Việc sử dụng liên kết quá tải có thể làm giảm tốc độ truyền tải và dẫn đến tình trạng giật, gây ra sự gián đoạn trong quá trình truyền tải tập tin.

Dựa theo nghiên cứu của Md. Iftekhar Hussain [7] cho thấy, hai thông số quan trọng nhất đối với dịch vụ File Transfer là link utilization và packet loss rate.

3.2.2 VoIP

Trong dịch vụ VoIP, mất gói tin, độ trễ và sử dụng liên kết đều có tác động đáng kể đến chất lượng của cuộc gọi. Khi mất gói tin xảy ra, âm thanh truyền tải có thể bị lệch và nhiễu, làm gián đoạn quá trình truyền tải tin nhắn giữa người dùng. Việc mất gói tin có thể xảy ra do đường truyền không ổn định hoặc do sử dụng băng thông quá tải. Độ trễ cũng có thể dẫn đến khoảng thời gian giữa khi người dùng nói và khi người nhận nghe được tin nhắn. Điều này có thể làm giảm tính hiệu quả của cuộc gọi và gây khó khăn trong việc giao tiếp. Độ trễ có thể được gây ra bởi sự cạnh tranh về băng thông và sử dụng liên kết cao giữa các người dùng khác nhau. Ngoài ra, sử dụng liên kết cao cũng có thể dẫn đến tình trạng cạnh tranh băng thông giữa các người dùng và làm giảm chất lượng cuộc gọi. Việc sử dụng liên kết quá tải có thể làm giảm tốc độ truyền tải và dẫn đến tình trạng giật, gây ra sự gián đoạn trong cuộc gọi.

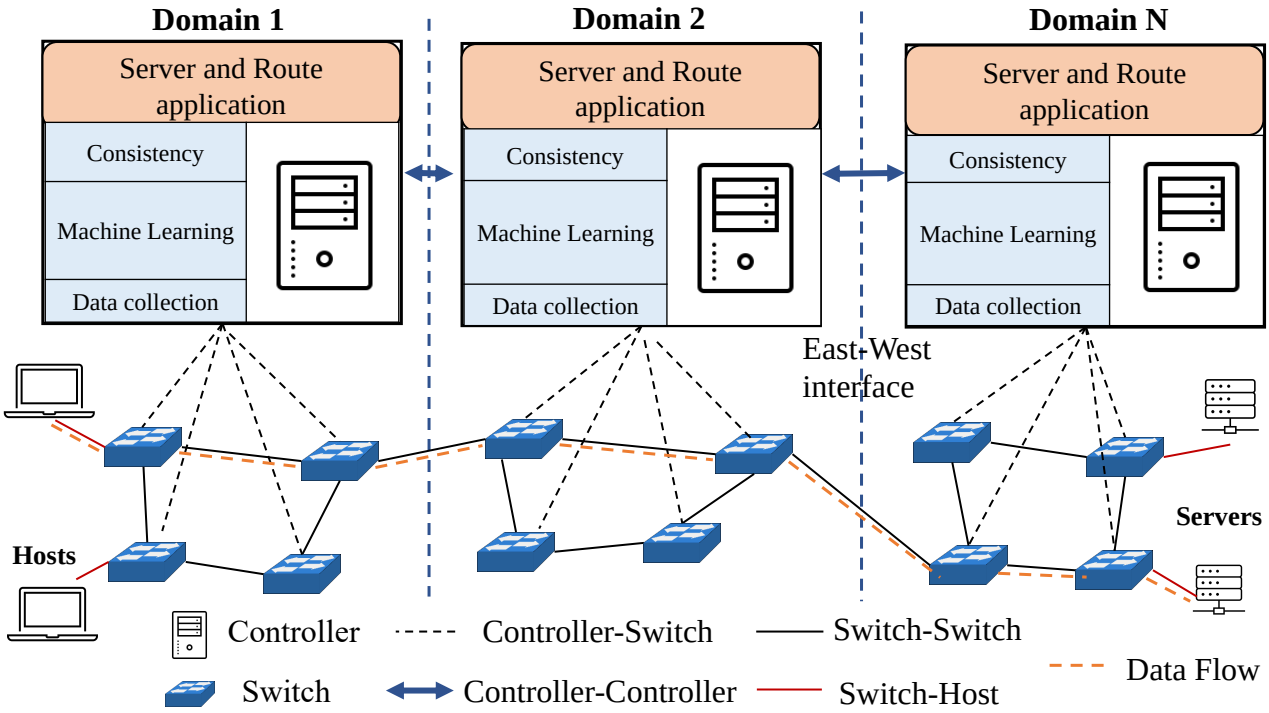
Vì vậy, để đảm bảo chất lượng cuộc gọi VoIP, các nhà cung cấp dịch vụ phải đảm bảo sự ổn định của đường truyền và đảm bảo rằng băng thông được phân phối đúng đắn. Theo khảo sát của tối đối với hai nhà cung cấp dịch vụ VoIP là Nexttivia [8] và GoogleHangout [9] thì hai thông số được họ quan tâm nhất là delay và link utilization.

3.2.3 Video Streaming

Dịch vụ Video Streaming là một trong những dịch vụ trực tuyến phổ biến nhất hiện nay và nó phụ thuộc rất nhiều vào mạng internet. Khi sử dụng dịch vụ Video Streaming, mất gói tin, độ trễ và sử dụng liên kết có thể ảnh hưởng đến trải nghiệm của người dùng. Khi mất gói tin xảy ra, video có thể bị gián đoạn và gây ảnh hưởng đến trải nghiệm xem video. Điều này thường xảy ra khi đường truyền không ổn định hoặc khi có quá nhiều người dùng cùng sử dụng cùng một đường truyền. Khi sử dụng đường truyền không ổn định, độ trễ có thể tăng lên và video có thể bị chậm hoặc bị gián đoạn trong quá trình phát. Sử dụng liên kết quá tải cũng có thể làm giảm tốc độ phát video và dẫn đến tình trạng giật. Để đảm bảo trải nghiệm tốt nhất cho người dùng, các nhà cung cấp dịch vụ Video Streaming cần đảm bảo rằng đường truyền được phân phối đúng đắn và đảm bảo rằng băng thông được phân phối một cách hợp lý giữa các người dùng. Người dùng cũng nên sử dụng kết nối mạng ổn định và đủ lớn để đảm bảo tốc độ phát video tốt nhất có thể. Bên cạnh đó, độ trễ cũng ảnh hưởng đến trải nghiệm xem video trên Video Streaming. Nếu độ trễ quá cao, video có thể bị chậm hoặc bị gián đoạn trong quá trình phát. Do đó, các nhà cung cấp dịch vụ Video Streaming cần đảm bảo rằng độ trễ được giảm thiểu và giữ ở mức thấp nhất có thể để đảm bảo trải nghiệm xem video tốt nhất cho người dùng.

Theo khảo sát của tôi đối với Cisco [10] và Onos [11] thì thông số quan trọng nhất đối với dịch vụ video streaming là delay, trong khi đó packet loss và link utilization cũng giữ vai trò quan trọng trong trong việc đảm bảo trải nghiệm người dùng.

3.3 Mô hình hóa bài toán



Hình 3.2: Knowledge-defined heterogeneous network architecture.

Trong kiến trúc mạng inter-SDN domains, ta gọi:

- $N = \{n_i | i = 1, 2, ..., n\}$ là tập n domains mạng,
- $C = \{c_i | i = 1, 2, ..., n\}$ là tập n controllers,
- $DB = \{db_i | i = 1, 2, ..., n\}$ là tập n database.

Mỗi controller c_i sẽ quản lý một domain mạng ni bao gồm các switches. Từng c_i có khả năng điều khiển các switches trong miền mạng n_i của nó để chuyển gói tin, thu thập dữ liệu và lưu vào database db_i.

Biểu diễn topology toàn cục bởi một đồ thị trọng số, có hướng $G = (V, E)$. Trong đó:

- Topology toàn cục G là hợp của các miền mạng: $G = n1 \cap n2 \cap ... \cap ni$
- Các miền mạng n_i nối với n_j thông qua cầu (bi, bj) và mỗi miền mạng được coi là một đồ thị nhỏ.
- V là tập các switches (nodes).
- E là tập các cạnh (links). Các cạnh này có thể bị deactivate và activate trong tương lai – tức là cấu hình của topology sẽ thay đổi real time.
- Trọng số của các links là một mapping $w : E \to R$. Trọng số này sẽ thay đổi

liên tục (do vậy việc thiết kế thuật toán cần tính real time để có thể thay đổi đường đi liên tục).

- Tham số mạng được thu thập trên mỗi link bao gồm delay, packet loss, link utilization và overhead. Các tham số này trên topo sẽ được cập nhật liên tục sau mỗi 3s.

Input: Host h, đỉnh $v_i \in V$ và cạnh $e_j \in E$ với $0 \leq i \leq m$ và $0 \leq j \leq n$.
Output: server s tương ứng với đỉnh cuối cùng trên đường đi được chọn và đường đi từ host h đến server s là $(v1, v2, \ldots, v_i, \ldots, v_p)$ sao cho $(v_i, v_i + 1) \in E$. Ngoài ra, mỗi đường đi tới server sẽ nhận được kết quả phản hồi của server gọi là server response time (RT).

Có 2 mục tiêu chính của bài toán:

- Cần thời gian phải hồi RT nhanh khi một host có yêu cầu lựa chọn và định tuyến tới server:

$$\text{Minimize } \frac{1}{N} * \sum_{k=1}^{N} RT_k$$

- Tối ưu hoá các trạng thái mạng delay (d), packet loss (l) và link overhead (lo) trên các flow trong thời gian dài (long-term performance):

$$\text{Minimize } \frac{1}{N} * \sum_{k=1}^{N} deplay_k$$

$$\text{Minimize } \frac{1}{N} * \prod_{k=1}^{N} loss_k$$

Trong đó các tham số mạng có các ràng buộc sau:

$$\sum X_{ij}^{k} d_{ij} \leq \delta_k$$

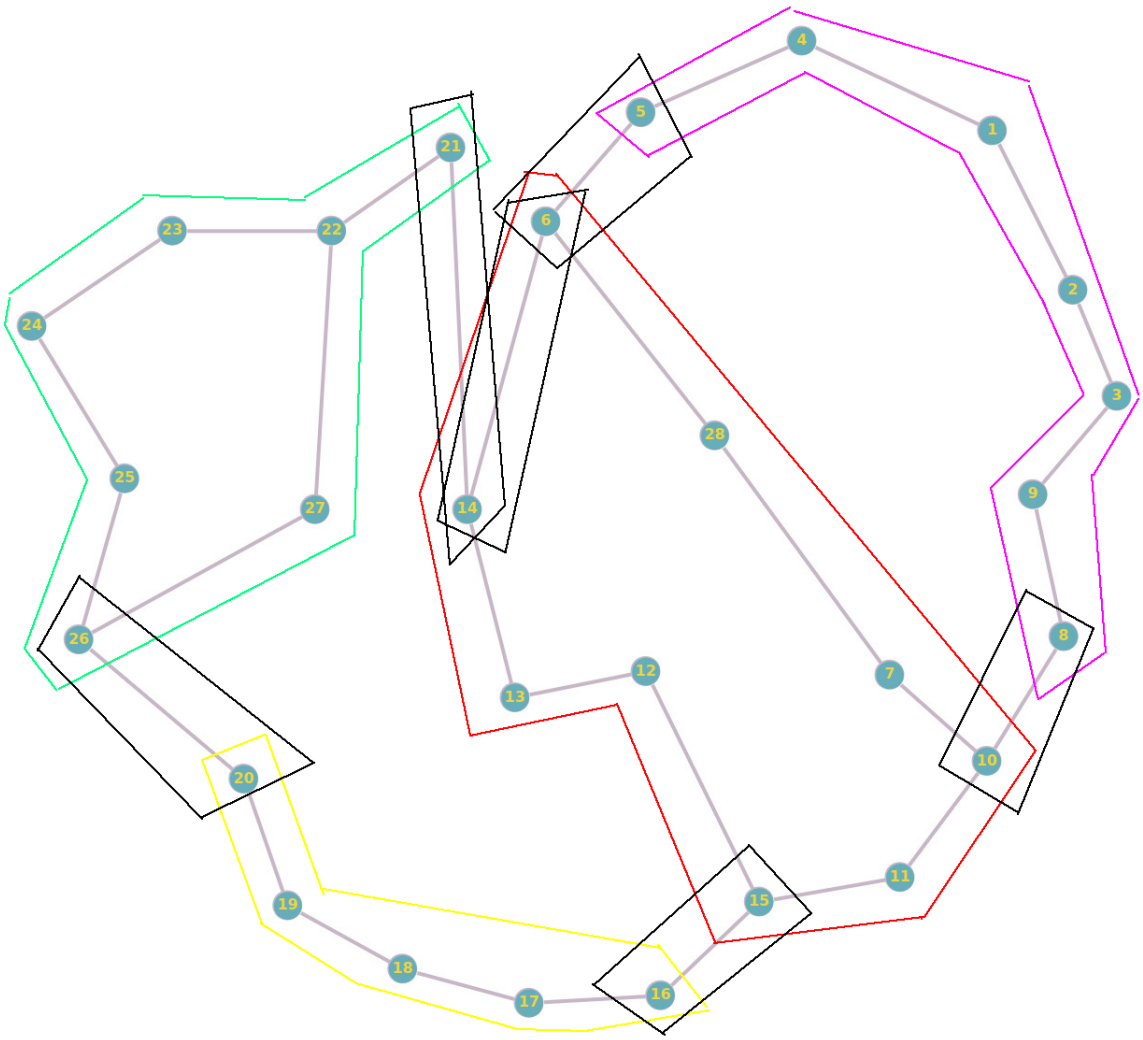
$$\prod X_{ij}^{k} l_{ij} \leq L_k$$

Tham số	**Mô tả**
m	Tổng số đỉnh trong topo
n	Tổng số cạnh trong topo
RT_k	Thời gian phản hồi tính từ lúc gói tin được gửi cho đến khi nhận được gói tin phản hồi từ server của flow k
N	Tổng số yêu cầu từ các hosts gửi tới controllers để tìm đường đi từ các hosts tới đích (tổng số flow controllers cần xử lí).
d_{ij}	Delay của link giữa node i và j.
l_{ij}	Packet loss của link giữa node i và j.
L_k	Packet loss tối đa cho flow k.
δ_k	Delay tối đa cho flow k.
X_{ij}^{k}	Nếu flow k sử dụng link(i, j) thì X=0 và nếu không sử dụng thì X=1.

3.3.1 Phân cụm các SDN domain

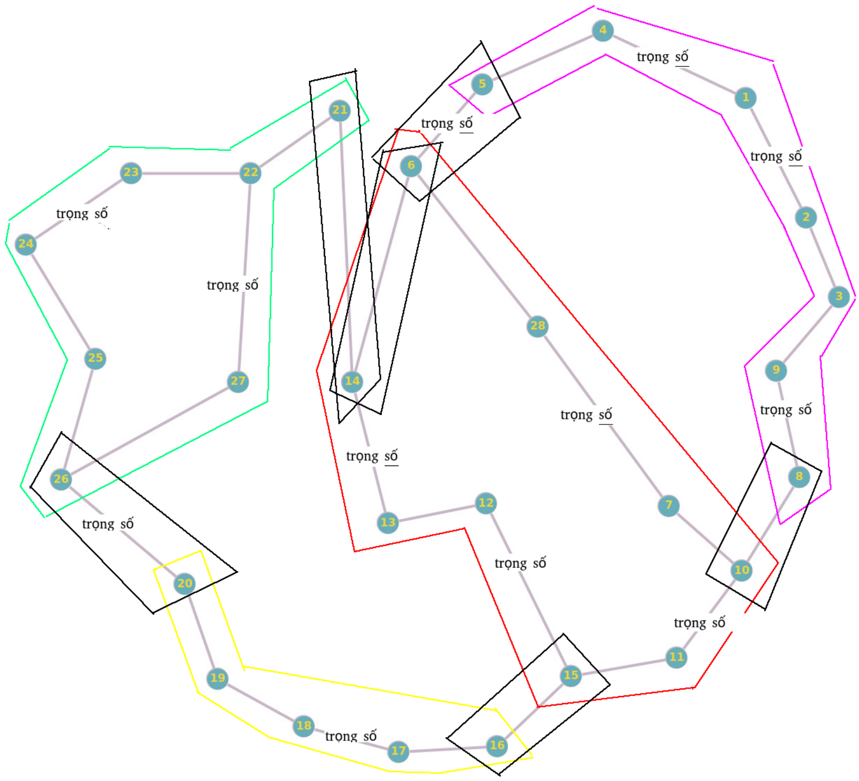
Bước 1: Sử dụng topologyzoo để xây dựng cấu hình mạng (global topology).

Bước 2: Chia global topology thành các sub topology cho 5 SDN domains (1 SDN domain được quản lý bởi 1 SDN controller). Trong đồ án này, tôi sử dụng kiểu controller là ONOS controller.



Hình 3.3: Sơ đồ sau khi chia các switch cho các SDN domains

Bước 3: Tiến hành link discovery, trạng thái hiện tại của link sẽ được tính trung bình trong khoảng 3 giây khám phá và cập nhật lại. Các trạng thái như: delay, packet loss, link utilization Sau khi thu thập trạng thái được tôi sử dụng hàm mục tiêu để tính real-time link cost ở mục 3.3.4 để từ trạng thái của mạng xác định được giá trị trọng số trên từng link và lưu vào local database của mỗi SDN domain.

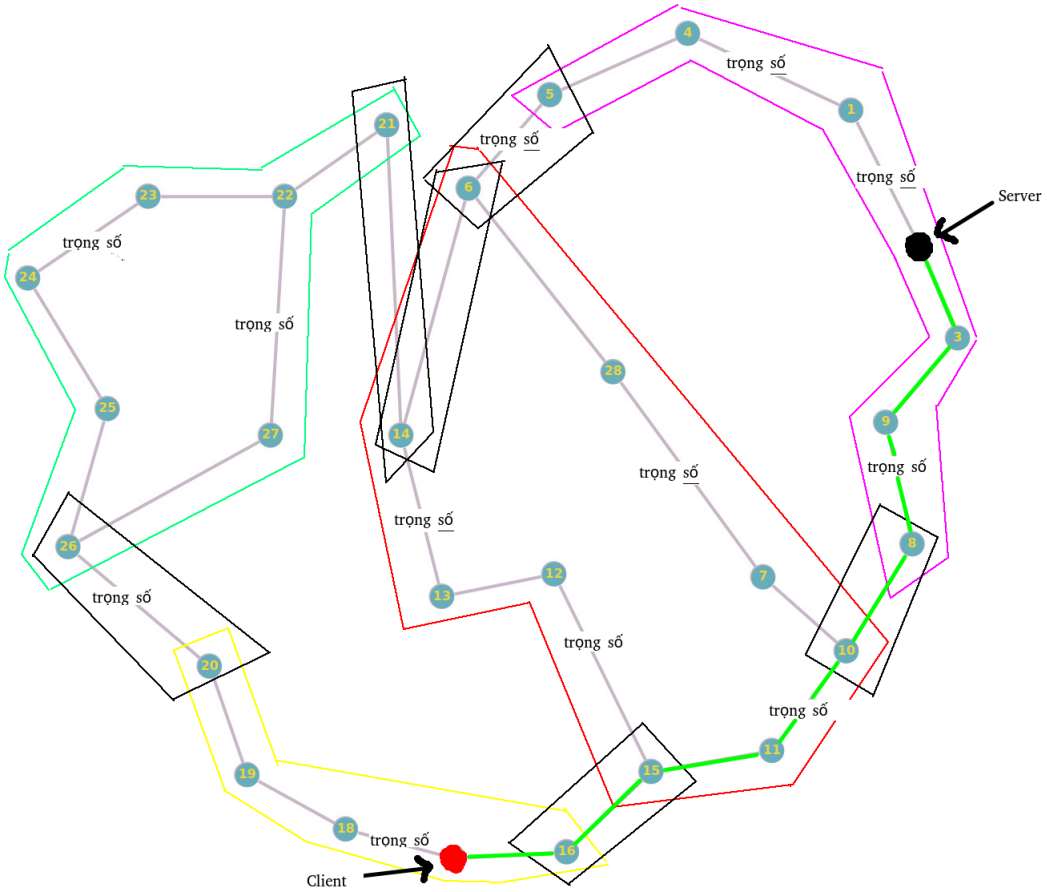


Hình 3.4: Sau 3s, các trọng số của link được khám phá sẽ thay đổi

Bước 4: Chia sẻ tri thức (trọng số học được) cho các SDN domains khác thông qua kết nối cầu đông-tây.

Bước 5: Khi có một host (đóng vai trò như một thiết bị của người dùng) muốn kết nối tới server. Ứng dụng sẽ sử dụng thuật toán Dijkstra với trọng số là tri thức sẽ được tổng hợp từ các miền mạng để tìm đường tới tất cả các server và lựa chọn server phù hợp cho host đó.

Bước 6: Khi có được đường đi phù hợp, sẽ cấu hình flow rule xuống dưới các switch để định tuyến.



Hình 3.5: Đường màu xanh là đường đi được chọn

3.3.2 Thuật toán định tuyến

Để ứng dụng định tuyến đa miền hướng dịch vụ của tôi hoạt động hiệu quả, việc lựa chọn các tham số để tính toán chi phí đường đi là yếu tố quyết định. Ở bài toán này, ta cần tối ưu lưu lượng trên từng link cũng như phân chia hợp lý tải cho các server nên hàm chi phí sẽ bao gồm hai thành phần là chi phí đường đi (PC) và chi phí server (SC). Với giải thuật định tuyến cốt lõi là thuật toán Dijiktra chi phí đường đi chính là tổng trọng số của các cạnh của đường định tuyến được chọn. Trong đó, trọng số của mỗi cạnh LC sẽ được tính toán liên tục real-time dựa trên các tham số QoS nhận được của từng link trong từng SDN domains và loại dịch vụ đang được sử dụng theo mô tả ở phần 3.3.4. Tiếp theo chúng tôi sẽ mô tả về các hàm chi phí bao gồm Path Cost và Server Cost. Sau khi tính được link cost và lan truyền link cost tới các SDN domains khác, khi có yêu cầu gửi gói tin từ client tới server, chi phí của quãng đường đi (path cost) tính dựa trên thông số của các link cost:

$$PC = \sum_{k=1}^{N} LC_i$$

Ngoài ra, server cost (SC) là lưu lượng mà server đang sử dụng hiện tại được tính bằng lưu lượng server đang truyền chia cho băng thông của server. Vậy, tổng chi phí (total cost) bao gồm hai thành phần là path cost và server cost được tính như sau:

$$TotalCost = s * SC + p * PC$$

Với tổng s và p bằng 1 đại diện cho mức độ quan trọng của SC và PC, được chúng tôi điều chỉnh để tối ưu với từng bài toán riêng. Total Cost sẽ được tính dựa trên đường đi tới tất cả các server và chọn server có total cost thấp nhất.

Khi có 1 yêu cầu từ một host kết nối đến một server. Với mỗi loại dịch vụ khác nhau, từ loại service mà Module phân loại dịch vụ dự đoán ra, hệ thống sẽ tính ra được "Link Cost" dựa vào các hàm mục tiêu. Trên mỗi miền mạng cũng đồng thời được tính toán giá trị "Server Cost". Sau đó ứng dụng được cài đặt trên SDN sẽ đọc dữ liệu từ Database local d. Database này sẽ được đồng bộ dữ liệu với n database trên n SDN domains khác nhau (dựa trên cơ chế consistency ở phần 2.2). Với mỗi server d thuộc tập Server hiện có, hệ thống sẽ tìm tất cả đường đi và tổng chi phí ứng với mỗi đường đi đó. Sau đó hệ thống sẽ tìm server và đường đi đến nó có "total cost" nhỏ nhất. Cuối cùng đường đi đó sẽ được thêm vào từng controller tương ứng.

3.3.3 Chuẩn hóa số liệu

Trong các chỉ số liên quan đến chất lượng được sử dụng để xác định chi phí liên kết, băng thông có giá trị càng cao thì chất lượng dịch vụ càng cao. Độ trễ và tỷ lệ mất gói tin có giá trị càng cao thì chất lượng dịch vụ càng thấp. Ngoài ra, mỗi chỉ số đều có đơn vị khác nhau. Đơn vị độ trễ là giây, đơn vị băng thông là bps và tỷ lệ mất gói tin được biểu diễn dưới dạng số thập phân đại diện cho phần trăm. Để biểu diễn tổng trọng số của các chỉ số độc lập, giá trị của các chỉ số đó cần được chuẩn hóa để đưa về cùng một tỷ lệ chuẩn. Chúng ta sử dụng phương pháp feature scaling [12] để chuẩn hóa phạm vi của các chỉ số độc lập này. Phương pháp này tái tổ chức phạm vi của tất cả các giá trị và đưa chúng về trong phạm vi [0, 1]. Các công thức chuẩn hóa cho từng chỉ số được trình bày trong các phương trình (1) đến (3). Để đặt một ranh giới cho mỗi biến, phạm vi tối đa và tối thiểu có thể có giá trị cố định hoặc điều chỉnh động dựa trên thông tin về topology mạng tại bất kỳ thời điểm nào.

$$lu_{ij}^{'} = \frac{lu_{max} - lu_{ij}}{lu_{max} - lu_{min}}, lu_{min} \leq lu_{ij} \leq lu_{max} \tag{3.1}$$

Trong đó lu_{ij} là link utilization đo được trên cạnh i, j và $lu_{ij}^{'}$ là giá trị link utilization sau khi chuẩn hóa. lu_{max} và lu_{min} là các giá trị lớn nhất và nhỏ nhất của link utilization trong database ở thời điểm lưu data mới vào data mới vào database.

$$pl_{ij}^{'} = \frac{pl_{max} - pl_{ij}}{b_{max} - pl_{min}}, pl_{min} \leq pl_{ij} \leq pl_{max} \tag{3.2}$$

Trong đó pl_{ij} là giá trị của tỉ lệ mất gói tin đo được và $pl_{ij}^{'}$ là giá trị của tỉ lệ mất gói tin sau khi chuẩn hóa. pl_{max} và pl_{min} là các giá trị lớn nhất và nhỏ nhất của packet loss rate trong database ở thời điểm lưu data mới vào data mới vào database.

$$d_{ij}^{'} = \frac{d_{max} - d_{ij}}{d_{max} - d_{min}}, d_{min} \leq d_{ij} \leq d_{max} \tag{3.3}$$

Trong đó d_{ij} là giá trị của độ trễ đo được và $d_{ij}^{'}$ là giá trị của độ trễ sau khi chuẩn hóa. d_{max} và d_{min} là các giá trị lớn nhất và nhỏ nhất của link utilization trong database ở thời điểm lưu data mới vào data mới vào database.

3.3.4 Hàm mục tiêu

Trọng số của mỗi liên kết trong mô hình của tôi được biểu diễn dưới dạng tổng có trọng số của băng thông kết nối hiện có, tỷ lệ mất gói tin và độ trễ, theo công thức (2), trong đó Cij biểu thị chi phí của kết nối (i, j), các chỉ số bij, plij và dij thể hiện băng thông kết nối hiện có, tỷ lệ mất gói tin và độ trễ trên kết nối (i, j), tất cả được tính động dựa trên tình trạng mạng hiện tại được giám sát bởi bộ điều khiển.

$$C_{ij} = \alpha \times lu_{ij} + \beta \times pl_{ij} + \gamma \times d_{ij} \tag{3.4}$$

Hệ số α, β và γ có mối quan hệ được biểu thị trong (3) như là các yếu tố tỉ lệ để gán ưu tiên cho mỗi chỉ số khác nhau. Liên quan đến ứng dụng của chúng ta, phân loại dựa trên độ nhạy cảm của lưu lượng với độ trễ và băng thông, chúng ta có thể định nghĩa trọng số khác nhau cho mỗi chỉ số trong từng lớp ứng dụng cụ thể. Module phân loại có thể cung cấp đầu vào động để phân biệt các lớp ứng dụng và áp dụng kết quả trong cơ chế ưu tiên lưu lượng.

$$\alpha + \beta + \gamma = 1 \tag{3.5}$$

Dựa theo khảo sát ở phần 3.2 và theo tác giả Kai-Cheng Chiu [5], tôi xác định được các tham số α, β, γ cho từng service như sau:

- Đối với File Transfer, packet loss cần phải thấp để tệp tin truyền được chính xác không có lỗi nào, deplay có thể cao miễn là trong giới hạn chấp nhận được, link utilization cần tối ưu hóa để truyền tệp càng nhanh càng tốt.

$$C_{ij} = 0.3 \times lu_{ij} + 0.6 \times pl_{ij} + 0.1 \times d_{ij}$$

- Đối với VoIP, delay phải thấp để cuộc nói chuyện diễn ra tự nhiên và không có độ trễ đáng chú ý, link utilization cần được tối ưu để truyền dữ liệu thoại càng nhanh càng tốt, packet loss nên thấp để cuộc hội thoại được truyền chính xác mà không có từ nào bị cắt xén.

$$C_{ij} = 0.3 \times lu_{ij} + 0.1 \times pl_{ij} + 0.6 \times d_{ij}$$

- Đối với Video Streaming, packet loss có thể cao, miễn là không làm ảnh hưởng đáng kể đến chất lượng video, delay phải thấp để đảm bảo phát lại mượt mà và không bị đệm âm thanh, hình ảnh bị giật, link utilization cần được tối ưu để truyền phát với tốc độ bit không đổi.

$$C_{ij} = 0.25 \times lu_{ij} + 0.25 \times pl_{ij} + 0.5 \times d_{ij}$$

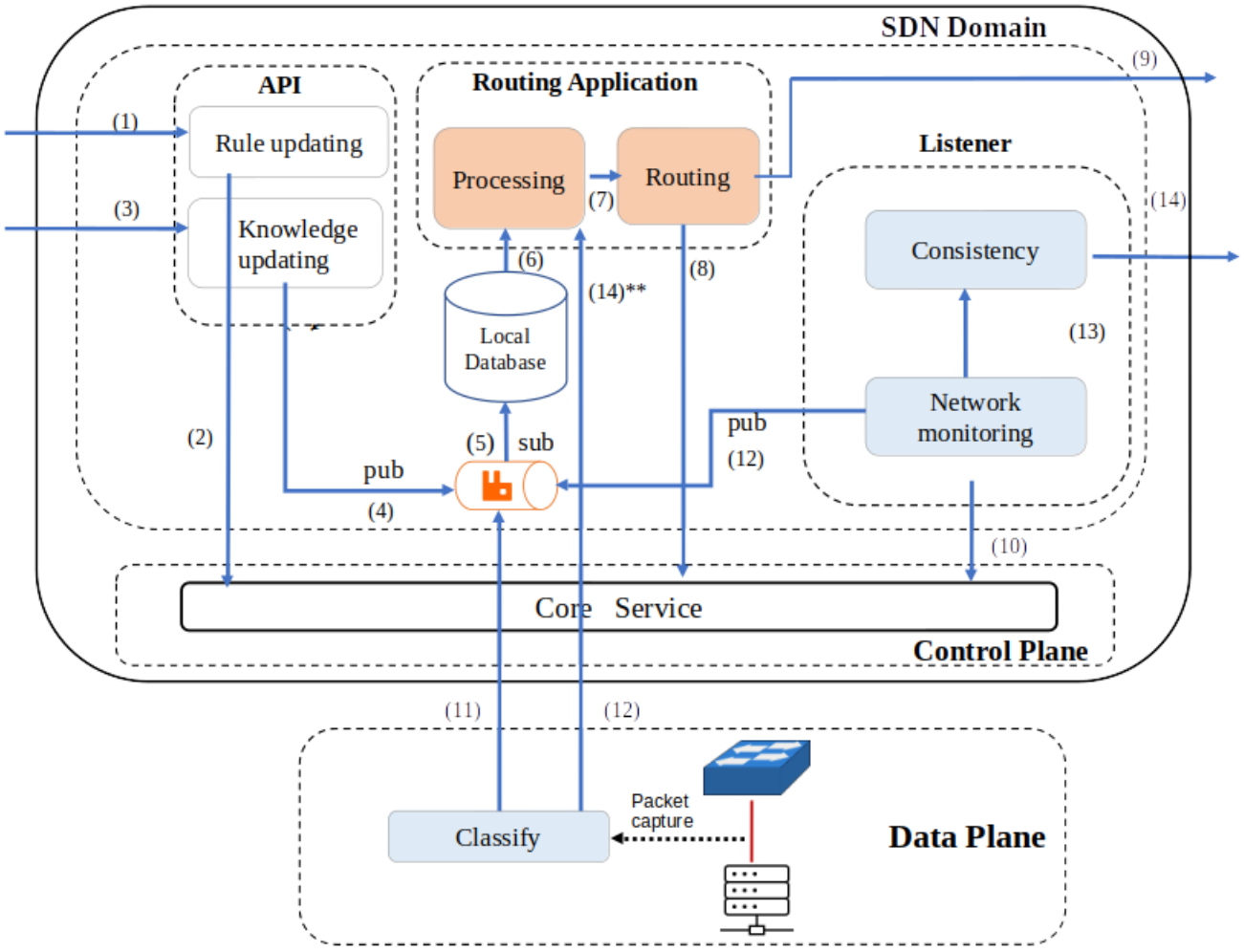
Bảng 3.1: Trọng số của các tham số Qos với mỗi dịch vụ

Tên dịch vụ	link utilization	packet loss rate	deplay
File Transfer	0.3	0.6	0.1
VoIP	0.3	0.1	0.6
Video Streaming	0.25	0.25	0.25

3.4 Thiết kế hệ thống

3.4.1 Kiến trúc hệ thống

Hệ thống tổng thể là một mô hình SDN, có thể được chia thành ba phần: lớp điều khiển, lớp dữ liệu và lớp kiến thức. Lớp dữ liệu đảm nhận việc chuyển tiếp lưu lượng và là nơi triển khai module phân loại dịch vụ; lớp điều khiển đảm nhận việc quản lý lưu lượng và các chính sách mạng ; lớp kiến thức là nơi chính để triển khai các mô-đun tùy chỉnh, đó là các thành phần cốt lõi của kiến trúc. Chi tiết của cả ba phần được mô tả trong 3.6.



Hình 3.6: Routing Application Architecture

Lớp dữ liệu bao gồm các OFS, chuyển tiếp lưu lượng giữa máy chủ và máy khách theo các quy tắc triển khai trong bảng luồng. Bảng luồng lưu trữ các lệnh được gửi bởi bộ điều khiển và OFS chuyển tiếp gói tin hoặc sửa đổi các trường tiêu đề theo các quy tắc. OFS liên lạc với lệnh điều khiển qua các kênh OpenFlow, cho dù triển khai luồng hoặc báo cáo thống kê. Đối với lớp dữ liệu, tôi đã thiết kế module Classify. module này có nhiệm vụ capture gói tin khi nó đi qua switch. Đối với mỗi flow, tôi xử lý dữ liệu và lấy ra 20 packets của mỗi flow, với mỗi packet tôi lấy ra 256 byte đầu tiên của packet để đưa vào mô hình học liên kết để dự đoán ra loại service. Sau khi dự đoán được loại service, module sẽ gửi một API đến Routing Application trên lớp kiến thức để có thể cập nhật lại đường đi. Đối với lớp kiến thức, tôi có chỉnh sửa lại phần Processing data của Routing application để có thể cập nhật lại đường đi của flow đúng với kiểu service đang chạy.

Tuy nhiên, một gói dữ liệu thô được bắt không phải là dạng lý tưởng để xử lý và phát triển trong mô hình học liên kết được sử dụg. Ví dụ, gói dữ liệu được sử dụng trong công việc này ở định dạng PCAP hoặc PCAPNG. Một gói dữ liệu thô bao

gồm các thông tin không cần thiết cho việc phân loại, chẳng hạn như tổng số giao thức TCP/UDP/ICMP, vv. Trong phần này, tôi sẽ giới thiệu các thủ tục tiền xử lý gói dữ liệu thô và bộ dữ liệu thô để tạo ra data chuẩn có thể đưa vào mô hình học liên kết.

a, Tiền xử lý PACKET BYTE VECTOR

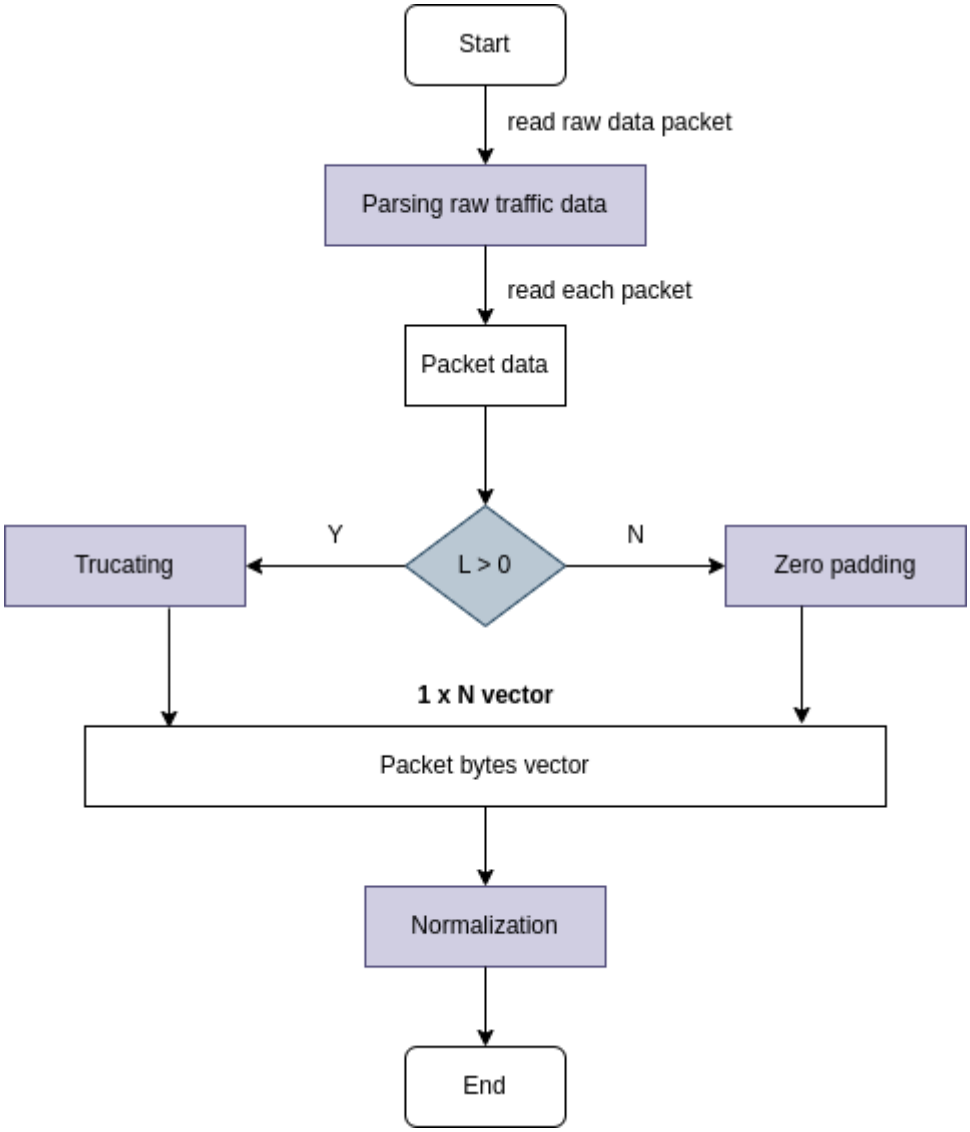
Việc tiền xử lý một gói dữ liệu thô có ba bước, bao gồm parsing, truncating/padding và normalization. Tổng quan về quy trình tiền xử lý được hiển thị trong Hình 3.7. Một gói dữ liệu thô được xử lý từng byte, tương tự như một pixel của một hình ảnh có thể dễ dàng được đưa vào một bộ phân loại dựa trên học liên kết. Parsing là để loại bỏ tiêu đề Ethernet của một gói dữ liệu thô. Thông tin lớp liên kết dữ liệu như địa chỉ MAC, loại khung, v.v., không hữu ích trong việc phân loại gói tin. Quá trình phân tích cú pháp giảm kích thước đầu vào của một gói tin. Hơn nữa, một số nhiễu được lọc trong quá trình này để đạt được độ chính xác phân loại tốt hơn.

Truncating/zero-padding là để cố định kích thước của mỗi đầu vào gói dữ liệu cho bộ phân loại. Các đầu vào cần có cùng kích thước cho các bộ phân loại gói tin dựa trên học sâu được đề xuất. Cụ thể, xác định n là kích thước đầu vào mục tiêu cho DataNet, trong đó 0 < n 1500. Đơn vị truyền tải tối đa có kích thước là 1500 byte. Một gói tin đầu vào sẽ được cắt hoặc được thêm số 0 tùy thuộc vào độ dài của nó so với n.

Đầu vào sau khi được truncating and zero-padding được xác định là một vector byte gói tin (PBV). Ví dụ, PBV thứ i được miêu tả như sau:

$$X_i = \{x_{i1}, x_{i2}, x_{i3}, ...x_{in}\}$$

Trong đó xij biểu thị byte thứ j của gói tin Xi. Sau đó, mỗi PBV được chuẩn hóa về khoảng [0, 1] để tăng tốc độ phân loại. Để đơn giản, chúng ta giả định Xi là kết quả được chuẩn hóa của PBV thứ i. Phân loại một gói dữ liệu được xử lý bằng cách sử dụng PBV đã được chuẩn hóa.

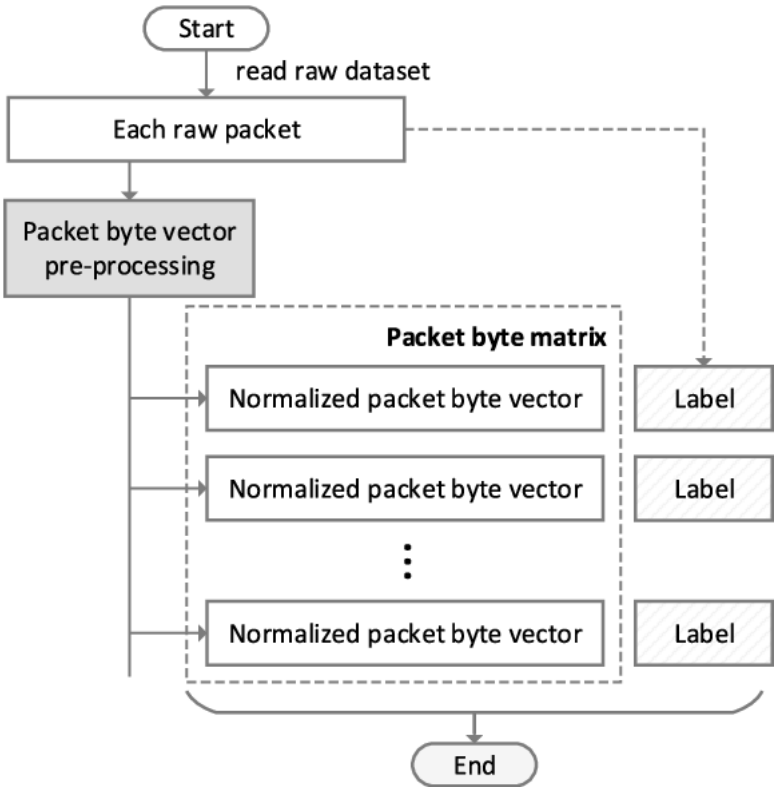


Hình 3.7: Raw data packet pre-processing

b, Tiền xử lý PACKET BYTE MATRIX

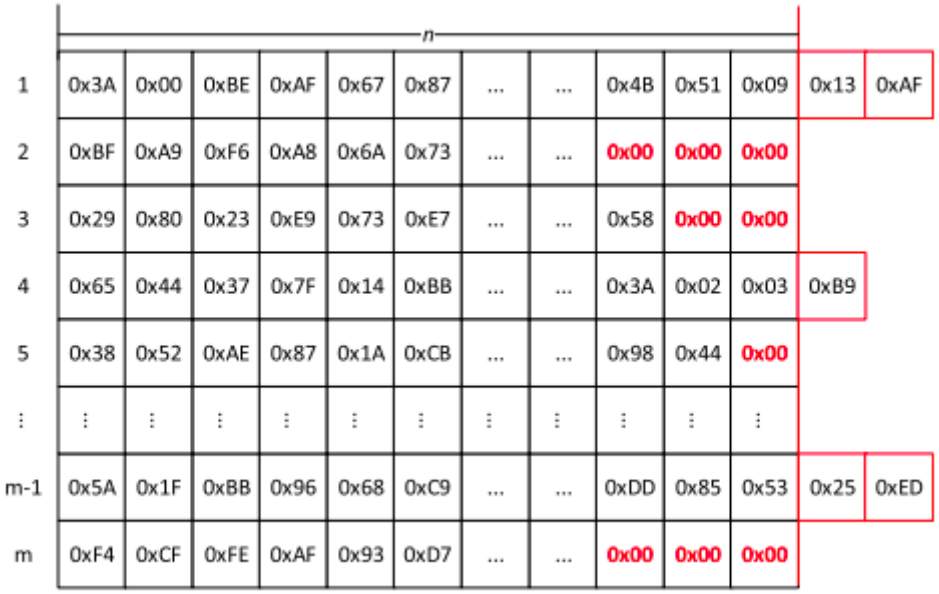
Để tạo ra DataNet, một bộ dữ liệu chứa các gói dữ liệu thô đã được gán nhãn được xử lý theo các bước như được hiển thị trong Hình 3.8. Bộ dữ liệu thô được xử lý thành ma trận byte gói tin (PBM) X như sau:

$$X = \{X_1^T, X_2^T, ..., X_N^T\}$$



Hình 3.8: Preprocessing the training dataset.

trong đó T là hàm chuyển vị, và m là số lượng gói dữ liệu thô trong bộ dữ liệu. Hình 3.9 minh họa một PBM.



Hình 3.9: Preprocessing the training dataset.

Mỗi PBV Xi được liên kết với một nhãn Li, ví dụ, Youtube, File Transfer, VoIP, vv. Sau khi tiền xử lý, bộ dữ liệu thô được tạo thành từ một PBM và một vector

nhãn, được mô tả như sau:

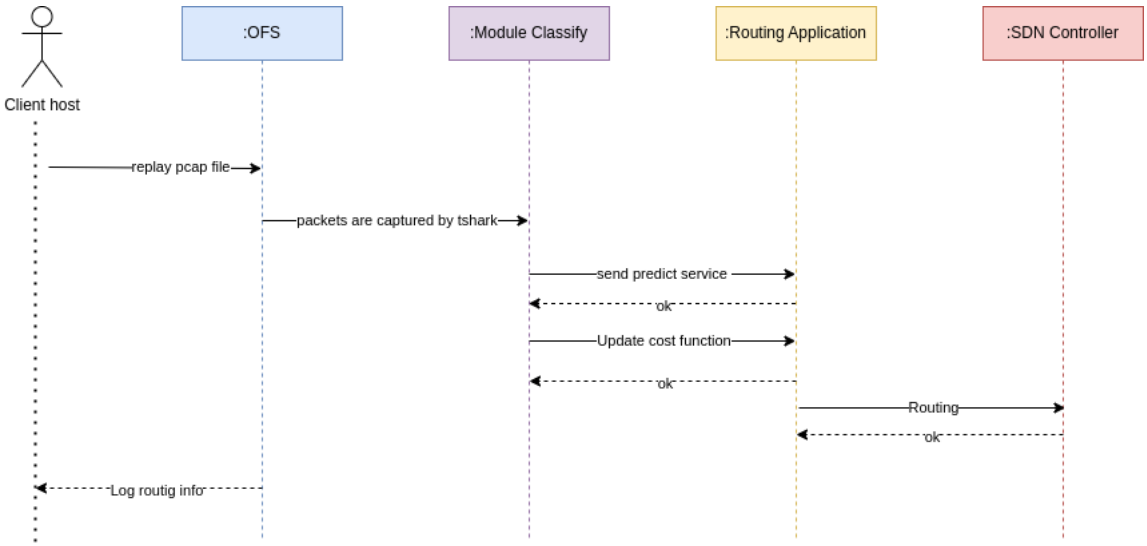
$$X = [X_1, X_2, \ldots, X_m]^T, \quad X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]^T \tag{3.6}$$

Một ví dụ về PBM đã được gán nhãn được hiển thị trong Bảng 1. Vì kích thước dữ liệu thay đổi ngay cả đối với cùng một lớp ứng dụng, một kích thước cố định sẽ làm cho việc bảo trì và cập nhật dễ dàng hơn khi thêm các ứng dụng mới vào hệ thống. Mà không mất tính tổng quát, số 256 được chọn cho phần còn lại của công việc này.

No	Label	Packet Byte Vector	Input size n	Original size
1	File Transfer		256	42
2	VoIP		256	108
3	Video Streaming		256	1480

3.4.2 Biểu đồ trình tự

Biểu đồ trình tự của Module SDN-base Application-Aware for Routing được thể hiện trong hình sau:



Hình 3.10: Routing Application sequence diagram

Quy trình hoạt động của module được tóm tắt như sau: Mỗi khi các host client thuộc Control Plane gửi data đến các host server, module phân loại dịch vụ sẽ sử dụng tshark để bắt gói tin và đưa vào mô hình học máy đã được huấn luyện để tìm ra loại dịch vụ đang được sử dụng. Ở thời điểm này, do vẫn chưa tìm ra được đường đi tối ưu với kiểu dịch vụ nên data sẽ được chuyển với đường đi có trọng số mặc định ban đầu trên mỗi cạnh là 10^{-7}. Sau khi dự đoán được loại dịch vụ, module phân loại dịch vụ sẽ gửi một API đến Routing Application để cập nhật hàm mục

tiêu và tìm ra đường đi tối ưu cho dịch vụ. Sau khi tìm ra được đường đi tối ưu thì Routing Application sẽ gửi API đến SDN controller để cập nhật lại bộ quy tắc định tuyến cho luồng dữ liệu.

CHƯƠNG 4. ĐÁNH GIÁ THỰC NGHIỆM

4.1 Các tham số đánh giá

Để đánh giá hiệu năng của mô hình đề xuất so với mô hình không sử dụng dịch vụ phân loại và hàm chi phí, trong phần này tôi sẽ lần lượt đo hiệu năng mạng dựa trên chỉ số sử dụng của link (link utilization) và độ mất mát gói tin (packet loss rate) đối với từng dịch vụ Youtube, chia sẻ File và VoIP. Ngoài ra, tôi cũng đánh giá thêm chỉ số phản hồi của server (server response time) nhằm đánh giá độ trễ của từng dịch vụ sau khi phản hồi tới người dùng dựa của mô hình đề xuất.

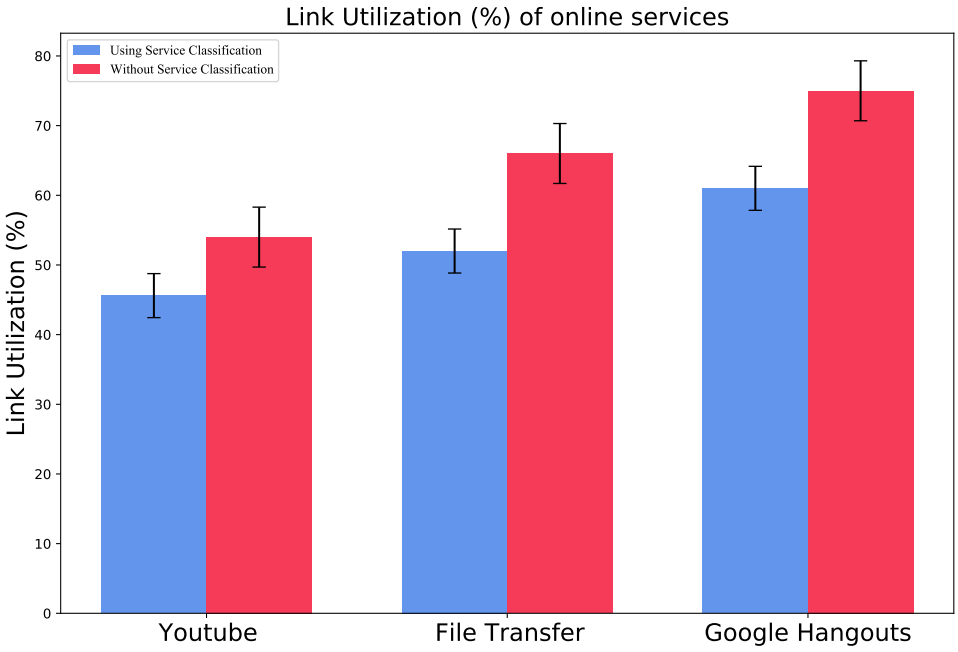
4.2 Phương pháp thí nghiệm

Trong phần này, triển khai hệ thống phân tán gồm 1 máy master và 4 máy slave. Máy master có cấu hình Dell PowerEdge R840, 32 GB RAM. Máy này sẽ được sử dụng đề cài đặt ứng dụng "Routing Application" và chạy thuật toán phân loại và điều hướng luồng dịch vụ. 4 máy slave còn lại sử dụng SDN controller ONOS version 2.3.1 [13] dùng để quản lý, thu thập và trao đổi dữ liệu giữa các miền mạng. Công cụ mô phỏng mạng được tôi sử dụng là Mininet [14]. Mạng topology được tôi sử dụng là Darkstrand [15]. Đây là một mạng trong thực tế bao gồm 27 switch và 28 servers.

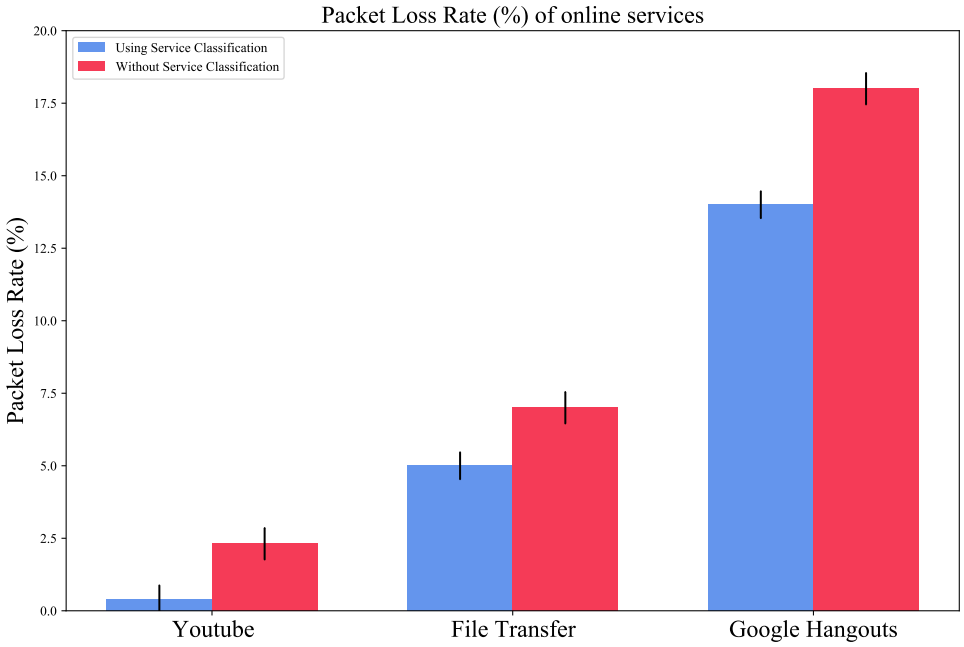
Để đánh giá khả năng mở rộng của mạng inter-SDN domains cũng như khả năng hỗ trợ các SDN giao tiếp của cầu nối SINA (như đã mô tả ở phần 2.2). Mạng Darkstrand này sẽ được chia thành 4 mạng con, với mỗi mạng con được quản lý bởi 1 SDN controller. Để đánh giá tính hiệu quả của các hàm mục tiêu đã đề xuất, tôi thiết lập mạng dynamic network. Trong dynamic network, sau khoảng random time RT, trạng thái của links sẽ được thay đổi với một xác suất theo mô hình trong bài báo [16]. Traffic generation tool được tôi sử dụng là Tcpreplay. Công cụ này sẽ giúp sinh ra các file PCAP (Packet Capture - một dạng tệp tin được sử dụng để lưu trữ dữ liệu bắt gói tin mạng) được lưu trên máy local. Các file PCAP này có kích cỡ trong khoảng [20, 80] Mb. Với mỗi kết quả thực nghiệm, tôi sẽ lặp lại khoảng 10 lần với độ tin cậy là 95%.

Bảng 4.1: Bảng cấu hình các tham số thực nghiệm trên mạng

Dynamic params	Value
Link	Random Delay [25, 50, 75, 100, 125] ms
Switch	Random Loss [0.1, 1, 2, 5] %



Hình 4.1: Mức độ sử dụng trung bình của links.



Hình 4.2: Tỷ lệ trung bình mất mát gói tin.

4.3 Phân tích hiệu năng mạng

Trong phần này, tôi sẽ tiến hành so sánh mô hình đề xuất với mô hình không có hàm chi phí đối với các dịch vụ Youtube, truyền File và VoIP dựa trên 2 thông số đo hiệu năng mạng là mức độ sử dụng của link và độ mất mát của gói tin.

4.3.1 Phân tích mức độ sử dụng trên link đối với các dịch vụ

Trong biểu đồ thực nghiệm 4.1, ban đầu tôi thiết lập ngưỡng link tắc nghẽn khi đối với dịch vụ Youtube là 60%, File Transfer là 65% và Google Hangouts là 70% theo bài báo [17]. Khi mức độ sử dụng link trở nên quá tải và vượt qua ngưỡng, điều này có thể ảnh hưởng tới chất của từng lượng dịch vụ. Việc tắc nghẽn và sử dụng quá quả nhiều băng thông có thể kéo theo độ trễ và mất mát gói tin cao trên mạng.

Ta dễ dàng thấy rằng, với mỗi dịch vụ được thực nghiệm, việc điều hướng dịch vụ với module phân loại và hàm chi phí đều cho kết quả mức độ sử dụng link tương đối tốt và không vượt ngưỡng như việc điều hướng dịch vụ mặc định.

Cụ thể hơn, đối với dịch vụ Youtube, mô hình đề xuất của tôi đã cải thiện gần 10% mức độ sử dụng lưu lượng của link so với cơ chế điều hướng mặc định (đạt khoảng 56%). Đối với dịch vụ File Transfer, mô hình đề xuất cũng đã cải thiện thấp hơn khoảng 10% mức độ sử dụng của link so với cơ chế điều hướng mặc định (đạt xấp xỉ 64%). Đối với dịch vụ Google Hangouts, mô hình đề xuất chỉ sử dụng khoảng gần 60% mức độ của link, tốt hơn gần 15% so với việc điều hướng mặc định.

Kết quả này, có thể lý giải rằng với phương pháp không có hàm chi phí, hệ thống chỉ định tuyến mạng trên cùng một đường được coi là ngắn nhất trong hầu hết thời gian phát dịch vụ liên tục. Ngược lại, với mô hình đề xuất đã xác định được luồng của dịch vụ thực sự đang cần yêu cầu và tiền hành phân chia tài nguyên cân bằng trên toàn mạng.

4.3.2 Phân tích tỷ lệ mất mát gói tin đối với các dịch vụ

Trong biểu đồ thực nghiệm 4.2, tôi tiến hành đánh giá tỷ lệ trung bình mất của gói tin đối với từng dịch vụ.

Đối với các dịch vụ trực tuyến, mất mát gói tin có thể ảnh hưởng tới các dịch vụ này khi mà chúng luôn đòi hỏi tốc độ truyền tải cao như video trên Youtube. Theo hình 4.2 với dịch vụ Youtube, mô hình đề xuất với module phân loại dịch vụ và hàm chi phí gần như ghi nhận không có tỷ lệ mất mát gói tin so với xấp xỉ 2.5% mất mát của cơ chế điều hướng mặc định.

Ngoài ra, mất mát gói tin cũng có thể ảnh hưởng đến quá trình truyền file trong

các dịch vụ truyền file trực tuyến và có thể làm mất mát dữ liệu trong quá trình truyền file, khiến file bị hỏng hoặc không hoàn chỉnh. Do trong case thực nghiệm này, việc truyền file giữa client-server rất lớn liên tục nên tỷ lệ mất mát gói tin trung bình trên toàn thời gian của mô hình đề xuất là xấp xỉ 5%. Tuy nhiên, chỉ số này vẫn tốt hơn rất nhiều so với cơ chế điều hướng mặc định, đạt tỷ lệ mất mát gần 7.5%.

Đối với dịch vụ Google Hangouts, việc điều hướng mặc định ghi nhận kết quả trung bình mất mát gói tin vượt ngưỡng 17%, cao hơn nhiều so với mô hình đề xuất (chỉ xấp xỉ 13%),

Hiện tượng này có thê được giải thích rằng, với module phân loại dịch vụ và điều hướng với hàm chi phí, mô hình đề xuất sẽ luôn ưu tiên các gói tin của từng dịch vụ được truyền tải trên links đáp ứng được chất lượng của dịch vụ đó (delay, packet losss, etc.). Trong khi đó với cơ chế điều hướng mặc định, thuật toán luôn đánh các gói tin có độ ưu tiên như nhau, từ đó dẫn tới nhiều luồng dịch vụ được truyền tải tới cùng một link và hậu quả tắc nghẽn mạng và khiến tỷ lệ mất mát gói tin cao.

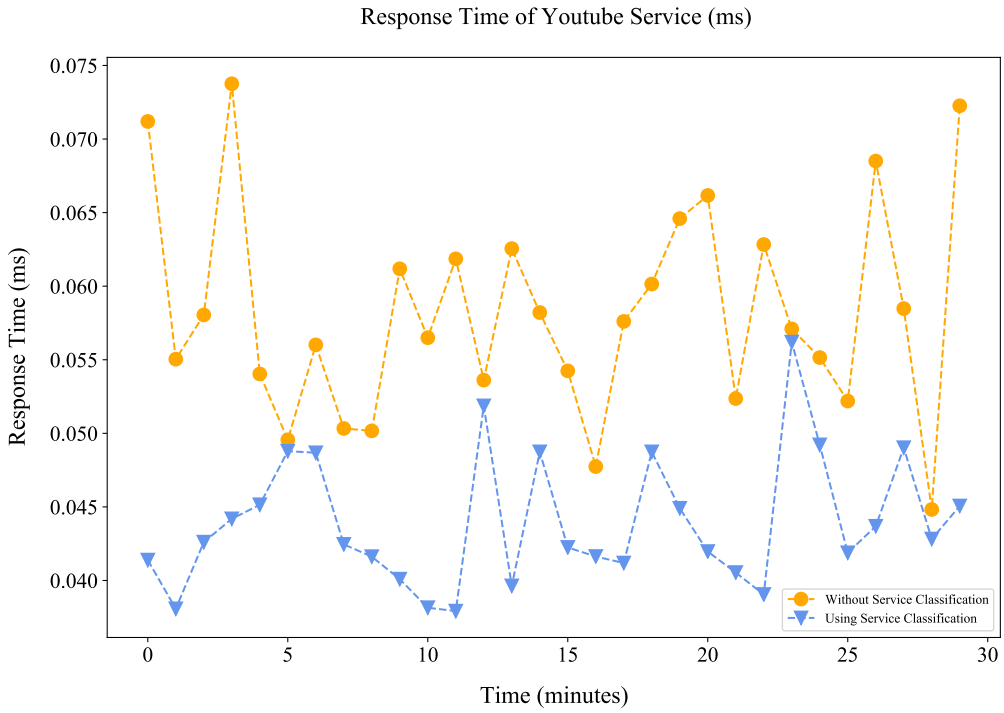
4.4 Phân tích thời gian phản hồi của server

Trong phần này, tôi sẽ tiến hành so sánh mô hình đề xuất với mô hình không có hàm chi phí đối với các dịch vụ Youtube, truyền File và VoIP dựa trên thông số thời gian phản hồi của server. Theo bài báo [xxx], thời gian phản hồi của server là một trong những chỉ số quan trọng nhất trong bài toán định tuyến tới server trong mạng cho các dịch vụ như: chơi game trực tuyến, truyền video hay chia sẻ file. Nếu thời gian phản hồi của server quá lâu, có thể ảnh hướng tới trải nghiệm của người dùng và thậm chí người dùng sẽ bỏ qua luôn việc sử dụng dịch vụ do phải chờ đợi.

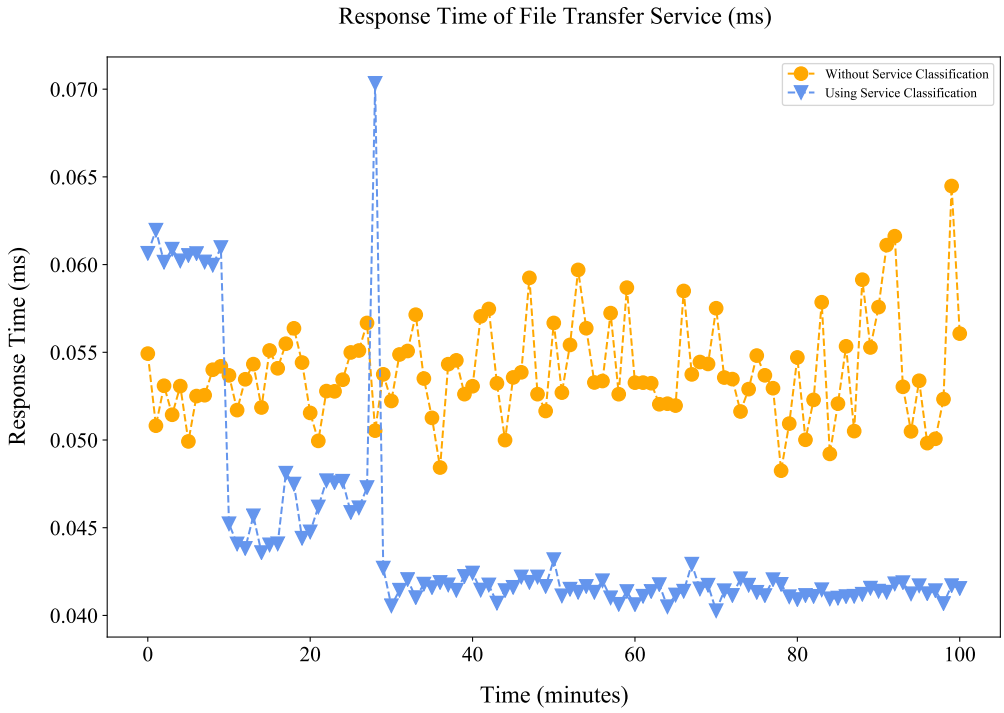
4.4.1 Phân tích thời gian phản hồi của server cho dịch vụ Youtube

Trong biểu đồ 4.3, tôi tiến hành so sánh thời gian phản hồi trung bình của server đối với dịch vụ Youtube. Ta dễ dàng nhận thấy rằng, từ thời điểm ban đầu cho tới kết thúc thời điểm so sánh, thời gian phản hồi của server đối với mô hình đề xuất luôn nhanh hơn nhiều so với việc định tuyến không sử dụng module phân loại dịch vụ và hàm chi phí.

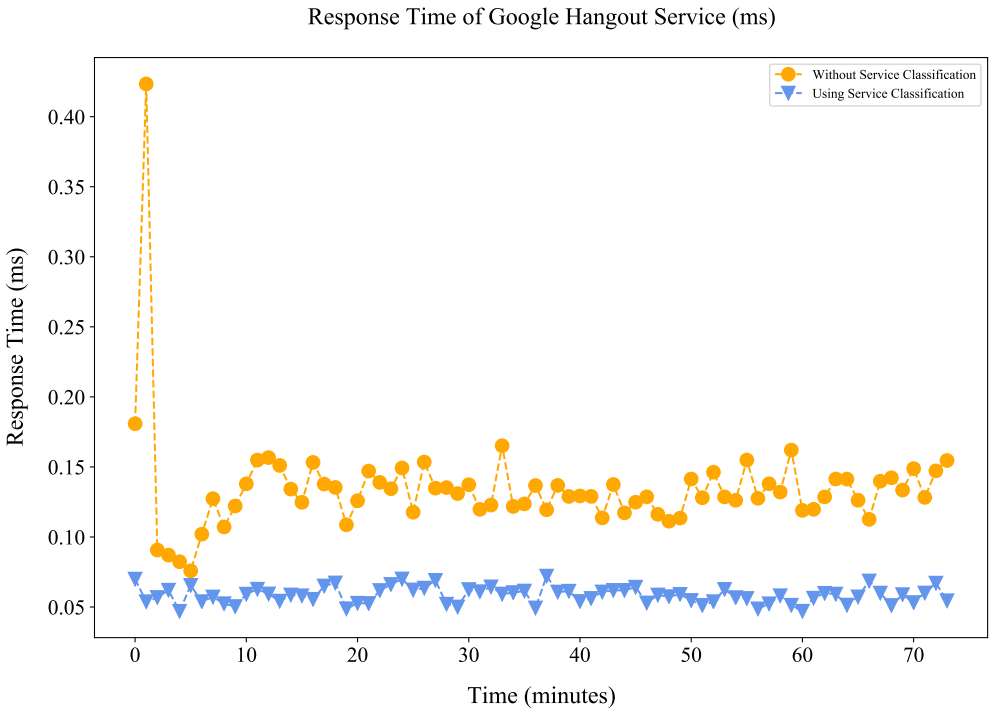
Tuy nhiên, ta vẫn còn thấy còn nhiều giá trị nhiễu của thời gian phản hồi server ở cả 2 phương pháp được so sánh. Ví dụ, ở phương pháp đề xuất, chỉ số phản hồi server có thể bị nhiễu đỉnh điểm vào khoảng trên 0.058 ms, tuy nhiên hầu như xu hướng của chỉ số này ở mô hình đề xuất đều dao động ổn định trong ngưỡng [0.04, 0.045] ms và thấp nhất là 0.030 ms.



Hình 4.3: Thời gian phản hồi của server (ms) - dịch vụ Youtube.



Hình 4.4: Thời gian phản hồi của server (ms) - dịch vụ File Transfer.



Hình 4.5: Thời gian phản hồi của server (ms) - dịch vụ Google Hangouts.

4.4.2 Phân tích thời gian phản hồi của server cho dịch vụ File Transfer

Trong biểu đồ 4.4, tôi tiến hành so sánh thời gian phản hồi trung bình của server đối với dịch vụ File Transfer. Tại thời điểm ban đầu (khoảng 15 phút đầu tiên), server response time của mô hình đề xuất khá cao, vượt ngưỡng 0.060 ms so với cơ chế điều hướng mặc định, dao động từ 0.050 đến 0.055 ms. Tuy nhiên, xu hướng này của cơ chế mặc định được duy trì đến hết thời gian so sánh.

Mặc dù sau 15 phút đầu, mô hình đề xuất ghi nhận server response time giảm tương đối đáng kể xuống gần 0.045 ms, tuy nhiên chỉ số này vẫn chưa được ổn định với đỉnh điểm tăng vọt lên 0.070 ms ở khoảng phút thứ 30. Chỉ sau gần 35 phút, server response time đã dần dần hội tụ và chỉ dao động khoảng 0.042 ms và luôn thấp hơn nhiều so với cơ chế điều hướng mặc định đến kết thúc thời gian so sánh.

Hiện tượng này có thể được giải thích rằng, mô hình đề xuất mất khoảng thời gian đầu để khám phá các giá trị link cost của links, cộng thêm với độ trễ trong việc dự đoán dịch vụ của các packet được gửi đi. Từ đó mô hình đề xuất dựa vào hàm chi phí cho từng dịch vụ để phân tải lưu lượng đến links đang ít được sử dụng, nên có tốc độ phản hồi nhanh hơn thay vì phải đợi số lượng lớn các gói tin di chuyển lần lượt tại một link cố định.

4.4.3 Phân tích thời gian phản hồi của server cho dịch vụ Google Hangouts

Trong biểu đồ 4.5, tôi tiến hành so sánh xu hướng thời gian phản hồi trung bình của server đối với dịch vụ Google Hangouts.

Theo biểu đồ 4.5, ta dễ dàng nhận thấy trong suốt thời gian so sánh, mô hình đề xuất với hàm chi phí và module phân loại dịch ghi nhận server response time dao động ổn định trong ngưỡng 0.05 ms. Ngược lại, với cơ chế định tuyến mặc định, hệ thống mạng ghi nhận chỉ số server response time hầu như luôn cao gần gấp đôi so với mô hình đề xuất, xấp xỉ trên ngưỡng 0.12 ms.

Điều đáng chú ý trong case thực nghiệm này là dịch vụ Google Hangouts ghi nhận độ trễ server response time lớn nhất so với hai dịch vụ Youtube và File Transfer. Điều này có thể được lý giải một phần rằng, vị trí đặt server của dịch vụ Google Hangouts trong case thực nghiệm xa hơn nhiều so với 2 dịch vụ còn lại và một phần ảnh hưởng tới chỉ số này. Bảng 4.2 tóm gọn hiệu năng của thuật toán đề xuất so với cơ chế điều hướng dịch vụ mặc định đối với server response time.

Bảng 4.2: So sánh tỷ lệ cải thiện server response time của cơ chế điều hướng với module phân loại và hàm chi phí so với cơ chế điều hướng mặc định

Service	**Cơ chế thuật toán**	**Performance**	
		Response Time (ms)	**Improvement**
Youtube	Đề xuất	0.045	25%
	Mặc định	0.06	
File Transfer	Đề xuất	0.04	38%
	Mặc định	0.065	
Google Hangouts	Đề xuất	0.06	53%
	Mặc định	0.13	

CHƯƠNG 5. KẾT LUẬN

5.1 Kết luận

Sự ra đời của hệ thống SDN đã thay đổi kiến trúc của một mạng thông thường bằng cách phân tách Mặt phẳng điều khiển khỏi Mặt phẳng dữ liệu, đồng thời tập trung các chức năng điều khiển của mạng trong một thực thể phần mềm được gọi là SDN Controller. Điều này cho phép người quản trị viên có thể quản lý và cấu hình mạng một cách dễ dàng. Đồng thời, để đáp ứng được với tính mở rộng của mạng thì kiến trúc phân tán gồm nhiều các SDN cũng ra đời, và đây được hứa hẹn sẽ là một giải pháp tuyệt vời để thay thế cho kiến trúc mạng truyền thống. Tuy nhiên bên cạnh những lợi ích mà nó mang lại thì kiến trúc này cũng cũng mở ra rất nhiều các thách thức cần được giải quyết.

Vì vậy, trong đồ án lần này, tôi đã giải quyết được hai vấn đề trong việc triển khai các dịch vụ mạng trên mạng SDN phân tán. Thứ nhất, đối với mỗi dịch vụ của từng nhà cung cấp khác nhau đều yêu cầu thông số đảm bảm bảo chất lượng khác nhau. Chính vì vậy, tôi đã tận dụng kỹ thuật học liên kết (Federated Learning) kết hợp mô hình mạng tích chấp học sâu (Deep Convolutional Neural Network) để triển khai được một mô hình học sâu có khả năng phân loại được từng luồng dịch vụ của các nhà cung cấp dựa vào các gói tin được truyền đi trên mạng. Từ đó, doanh nghiệp và những kỹ sư quản lý mạng có thể nhận biết và điều chỉnh chính sách của tôi để phù hợp với chất lượng của từng dịch vụ.

Thứ hai, để đảm bảo được hiệu suất của mạng và tận dụng được tối đa tài nguyên mạng khi điều hướng các dịch vụ trên đa miền SDN, tôi đã đề xuất các hàm mục tiêu tối ưu nhất định dựa trên từng dịch vụ, từ đó triển khai được một hệ thống định tuyến tự động thông minh theo yêu cầu dịch vụ.

Kết quả thực nghiệm dựa trên giải thuật và hàm mục tiêu đề xuất cho thấy hiệu suất truyền tin tới người dùng của hệ thống ở mức tương đối tốt nhưng vẫn đảm bảo được tỷ lệ phản hồi của mỗi server. Cụ thể hơn, hệ thống điều hướng hướng dịch vụ đã cải thiện chỉ số sử dụng lưu lượng trên link đối với các dịch vụ Youtube, File Transfer và Google Hangouts xấp xỉ lần lượt là 10%, 15%, 14%. Hệ thống cũng đã ghi nhận giảm tỷ lệ mất mát gói tin với các dịch vụ Youtube, File Transfer và Google Hangouts lần lượt khoảng 2%, 2.5%, và 3.5%. Cuối cùng, hệ thống điều hướng dịch vụ đề xuất cải thiện chỉ số phản hồi của server đối với các dịch vụ Youtube, File Transfer và Google Hangouts lần lượt là 25%, 38% và 53%. Rõ ràng, việc cải thiện hiệu năng, tài nguyên mạng và dịch vụ đã đạt được sẽ góp phần đảm bảo và nâng cao trải nghiệm người dùng đối với các dịch vụ trực tuyến.

5.2 Các vấn đề đã đạt được

Các vấn đề mà tôi đã đạt được trong đồ án bao gồm:

- Nắm bắt được kiến trúc và cơ chế hoạt động của mạng SDN phân tán.
- Nắm bắt được kiến trúc, cơ chế hoạt động và cách triển khai ứng dụng SINA (SDN inter-domains application) để kết nối trao đổi dữ liệu một cách nhất quán giữa các bộ điều khiển SDN.
- Tận dụng được kỹ thuật học liên kết và mạng tích chập học sâu nhằm xây dựng được một mô hình học sâu có khả năng nhận biết và phân loại dịch vụ.
- Đề xuất và thiết kế được hàm mục tiêu và giải thuật định tuyến nhằm điều hướng đa dịch vụ và đảm bảo chất lượng, hiệu năng mạng.
- Thiết kế và tích hợp được module phân loại dịch vụ và định tuyến dựa trên nền tảng REST API vào hệ thống mạng SDN phân tán.

5.3 Hướng phát triển trong tương lai

Trong đồ án hiện tại, tôi vẫn còn ba mặt hạn chế cần được giải quyết sau đây. Thứ nhất, hiện nay tôi mới triển khai hệ thống đề xuất trên tảng SDN ONOS. Để triển khai hệ thống này trên các bộ điều khiển SDN khác như: Opendaylight và Faucet thì cần thời gian nghiên cứu thêm. Thứ hai, do hạn chế về mặt thời gian, nên hệ thống phân loại và định tuyến hiện tại chưa triển khai thực nghiệm được trên mạng cỡ lớn 18 miền nhằm kiểm chứng hiệu năng của hệ thống đề xuất. Thứ ba, hệ thống đề xuất này mới chỉ được thử nghiệm trong môi trường phòng nghiên cứu chứ chưa được cài đặt trong hệ thống mạng thực tế. Tuy nhiên, những mặt hạn chế này sẽ là cơ hội đề mở ra hướng nghiên cứu mới, phát triển và hoàn thiển sản phẩm.

TÀI LIỆU THAM KHẢO

[1] D. V. C. X. P. L. Z. Q. K. E. S. P. Z. Li S. Peng **and** Guichard, "Problem statement and use cases confidence interval is computed at the 95% confidence level of application-aware networking (apn)," March 2023. **url**: https://datatracker.ietf.org/doc/html/draft-li-apn-problem-statement-usecases-01.

[2] S. Peng, J. Mao, R. Hu **and** Z. Li, "Demo abstract: Apn6: Application-aware ipv6 networking," **in***IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* IEEE, 2020, **pages** 1330–1331.

[3] A. Chooprateep **and** Y. Somchit, "Video path selection for traffic engineering in sdn," **in***2019 11th International Conference on Information Technology and Electrical Engineering (ICITEE)* IEEE, 2019, **pages** 1–6.

[4] A. Al-Jawad, P. Shah, O. Gemikonakli **and** R. Trestian, "Learnqos: A learning approach for optimizing qos over multimedia-based sdns," **in***2018 IEEE international symposium on broadband multimedia systems and broadcasting (BMSB)* IEEE, 2018, **pages** 1–6.

[5] C. Kai-Cheng, L. Chien-Chang **and** C. Li-Der, "Reinforcement learning-based service-oriented dynamic multipath routing in sdn," *Wireless Communications & Mobile Computing (Online)*, **jourvol** 2022, 2022.

[6] N.-T. Hoang, H.-N. Nguyen, H.-A. Tran **and** S. Souihi, "A novel adaptive east–west interface for a heterogeneous and distributed sdn network," *Electronics*, **jourvol** 11, **number** 7, **page** 975, 2022.

[7] I Hussain **and** N Ahmed, "A performance analysis of e-learning over wifi-based long distance networks," *Journal of Wireless Networking and Communications*, **jourvol** 6, **number** 4, **pages** 85–93, 2016.

[8] Nextivia, "What is voip qos how can it improve your call quality?," March 2023. **url**: https://www.nextiva.com/blog/voip-qos.html.

[9] G. W. Admin, "Optimize your network for voice," March 2023. **url**: https://support.google.com/a/answer/9288756?hl=en#zippy=%2Cpacket-inspectionprotocol-analyzers%2Cwi-fi-best-practices%2Cvoice-ip-address-range%2Cproxy-best-practices%2Ccreate-a-cloud-friendly-network.

[10] Cisco, "Video quality of service (qos) tutorial," March 2023. **url**: https://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-video/212134-Video-Quality-of-Service-QOS-Tutorial.html#_ftn4.

[11] Onos, "Apps and use cases," March 2023. **url**: https://wiki.onosproject.org/display/ONOS/BroadCtrl.

[12] G. Joel, *Data science from scratch: First principles with python*, 2015.

[13] P. Berde, M. Gerola, J. Hart **andothers**, "Onos: Towards an open, distributed sdn os," **in***Proceedings of the third workshop on Hot topics in software defined networking* 2014, **pages** 1–6.

[14] R. Jawaharan, P. M. Mohan, T. Das **and** M. Gurusamy, "Empirical evaluation of sdn controllers using mininet/wireshark and comparison with cbench," **in***2018 27th international conference on computer communication and networks (icccn)* IEEE, 2018, **pages** 1–2.

[15] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden **and** M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, **jourvol** 29, **number** 9, **pages** 1765–1775, 2011. DOI: 10.1109/JSAC.2011.111002.

[16] N.-T. Hoang, C.-S. Duong, T.-L.-T. Nguyen, V. Tong **and** H. A. Tran, "Knowledge-defined heterogeneous network: Use-case of qos-based server and route selection in large-scale network," **in***Proceedings of the 11th International Symposium on Information and Communication Technology* 2022, **pages** 150–157.

[17] A. Meshinchi, R. Al Mallah **and** A. Quintero, "Status-aware and sla-aware qos routing model for sdn transport network,"

PHỤ LỤC