HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

VŨ THANH TÙNG

ĐÁNH GIÁ HIỆU NĂNG CÁC GIẢI PHÁP ĐẢM BẢO CHẤT LƯỢNG DỊCH VỤ TRONG MẠNG IoT ĐỊNH NGHĨA BẰNG PHẦN MỀM

LUẬN VĂN THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI – 2020

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

VŨ THANH TÙNG

ĐÁNH GIÁ HIỆU NĂNG CÁC GIẢI PHÁP ĐẢM BẢO CHẤT LƯỢNG DỊCH VỤ TRONG MẠNG IoT ĐỊNH NGHĨA BẰNG PHẦN MỀM

Chuyên ngành: Kỹ thuật viễn thông

Mã số: 8. 52. 02. 08

LUẬN VĂN THẠC SĨ KỸ THUẬT

(*Theo định hướng ứng dụng*)

NGƯỜI HƯỚNG DẪN KHOA HỌC:

TS. LÊ HẢI CHÂU

HÀ NỘI – 2020

LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của riêng tôi.

Các số liệu, kết quả nêu trong luận văn là trung thực và chưa từng công bố trong bất kỳ công trình nào khác.

Hà Nội, tháng 05 năm 2020

Tác giả luận văn

Vũ Thanh Tùng

LỜI CẢM ƠN

Tôi xin cảm ơn gia đình, người thân đã luôn bên cạnh tôi và là nguồn động lực lớn lao để tôi làm việc và học tập.

Tôi xin được gửi lời cảm ơn chân thành tới TS. Lê Hải Châu – đã luôn hướng dẫn tận tình trong quá trình làm luận văn.

Đồng thời cũng xin gửi lời cảm ơn tới bạn bè và đồng nghiệp đã động viên, hỗ trợ để tôi có thể hoàn thành luận văn này.

Hà Nội, ngày 15 tháng 05 năm 2020

Vũ Thanh Tùng

MỤC LỤC

BẢNG CÁC THUẬT NGỮ VIẾT TẮT

THUẬT NGỮ	NGHĨA TIẾNG ANH	NGHĨA TIẾNG VIỆT
API	Application Programming Interface	Giao diện lập trình ứng dụng
DIFFSERV	Differentiated Services	Dịch vụ phân biệt
INTSERV	Integrated Services	Dịch vụ tích hợp
IoT	Internet Of Things	Vạn vật kết nối
IoT-A	IoT Architecture	Kiến trúc IoT
IP	Internet Protocol	Giao thức mạng internet
ISI	Inter-Symbol Interference	Nhiễu liên ký hiệu
NFV	Network Feature Virtualization	Ảo hóa chức năng mạng
NOS	Network Operating System	Ảo hóa chức năng mạng
OVSDB	OpenvSwitch Database	Cơ sở dữ liệu OpenvSwitch
QoS	Quality of Service	Chất lượng dịch vụ
SDN	Software Defined Networking	Mạng định nghĩa bằng phần
SLA	Service Level Agreement	Thỏa thuận mức dịch vụ
SOA	Service-Oriented Architecture	Kiến trúc hướng dịch vụ
SFB	Synthenis Filter Bank	Hệ bộ lọc tổng hợp

TCP	Transmission Control Protocol	Giao thức điều khiển truyền dẫn
TLS	Transport Layer Security	Bảo mật tầng truyền tải
UDP	User Datagram Protocol	Giao thức dữ liệu người dùng
WAN	Wide Area Network	Mạng diện rộng

DANH MỤC HÌNH VẼ

DANH MỤC CÁC BẢNG

LỜI NÓI ĐẦU

Công nghệ Internet vạn vật (IoT) đang phát triển mạnh mẽ cả về số lượng thiết bị, loại hình dịch vụ, công nghệ kết nối và cả dải yêu cầu đa tạp về băng thông cũng như chất lượng dịch vụ. Công nghệ IoT cho thấy nhiều triển vọng phát triển và khả năng ứng dụng mới nhưng cũng đồng thời tạo ra nhiều áp lực trong việc nâng cấp, cải tiến cũng như chỉ ra nhiều hạn chế và vấn đề khó khăn trong hạ tầng mạng thông tin hiện tại. Do vậy, các công nghệ mạng và thiết bị mạng mới đang được quan tâm đầu tư nghiên cứu và phát triển nhằm đáp ứng các yêu cầu của IoT. Với các ưu điểm trong quản lý, điều khiển và lập trình tài nguyên linh hoạt cùng khả năng triển khai, nâng cấp hạ tầng và dịch vụ linh hoạt với chi phí hiệu quả, giải pháp ứng dụng công nghệ mạng định nghĩa bằng phần mềm (SDN) trong hạ tầng truyền thông IoT (SD-IoT) đang dần trở thành giải pháp hứa hẹn cho truyền thông Internet tương lai.

Với mục tiêu nghiên cứu, tìm hiểu và tiến đến nắm bắt và làm chủ công nghệ IoT và SD-IoT, nội dung luận văn này tập trung nghiên cứu, khảo sát và đánh giá hiệu năng của các giải pháp đảm bảo chất lượng dịch vụ trong các mạng IoT định nghĩa bằng phần mềm. Trong luận văn, giải pháp đảm bảo QoS điển hình của mạng SD-IoT là mô hình phân biệt dịch vụ (DiffServ) được triển khai theo các kịch bản mô phỏng nhằm đánh giá và so sánh hiệu năng đối với các loại hình lưu lượng cơ bản của hệ thống. Nội dung luận văn được tổ chức thành 03 chương như sau:

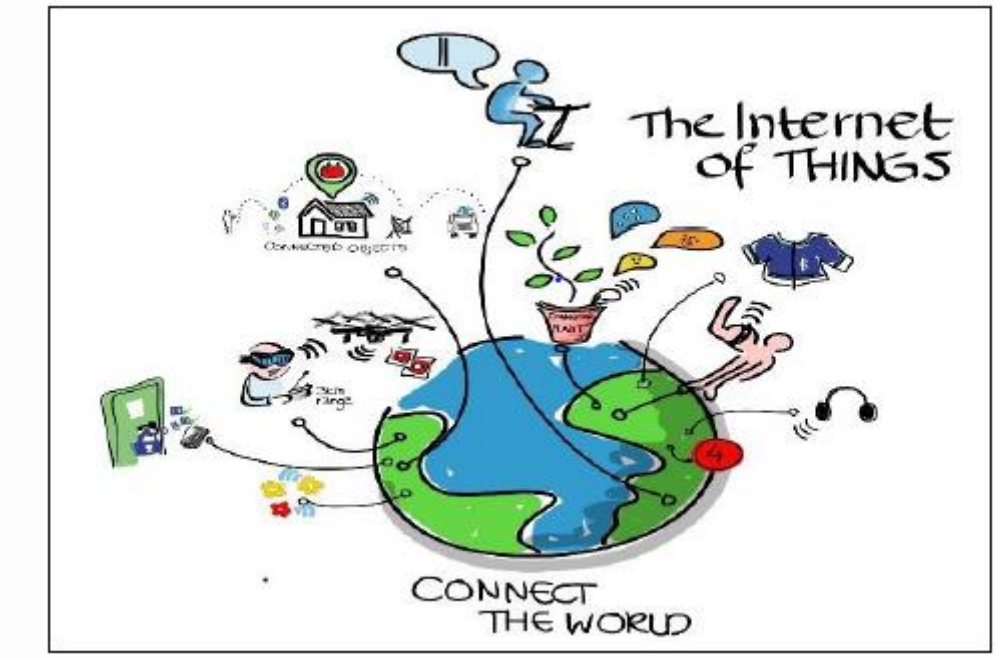
- *Chương 1- Tổng quan về công nghệ IoT và công nghệ mạng định nghĩa bằng phần mềm:* Giới thiệu khái quát về các công nghệ IoT, kiến trúc chức năng và các thành phần hệ thống IoT, đồng thời trình bày tổng quan về công nghệ SDN và khả năng áp dụng trong các hệ thống IoT.
- *Chương 2- Các giải pháp đảm bảo chất lượng dịch vụ trong mạng IoT định nghĩa bằng phần mềm:* Giới thiệu về các giải pháp đảm bảo chất lượng dịch vụ, các thành phần, tính chất đặc trưng và các tham số QoS (như băng thông, độ trễ, biến thiên trễ, độ tin cậy, mất gói).

- *Chương 3- Triển khai mô phỏng và đánh giá hiệu năng giải pháp đảm bảo QoS:* Trình bày kịch bản triển khai mô phỏng giải pháp đảm bảo chất lượng dịch vụ trong hệ thống SD-IoT và đánh giá hiệu năng giải pháp đảm bảo chất lượng dịch vụ.

CHƯƠNG 1: TỔNG QUAN VỀ CÔNG NGHỆ IoT VÀ CÔNG NGHỆ MẠNG ĐỊNH NGHĨA BẰNG PHẦN MỀM

1.1 Tổng quan về công nghệ IoT

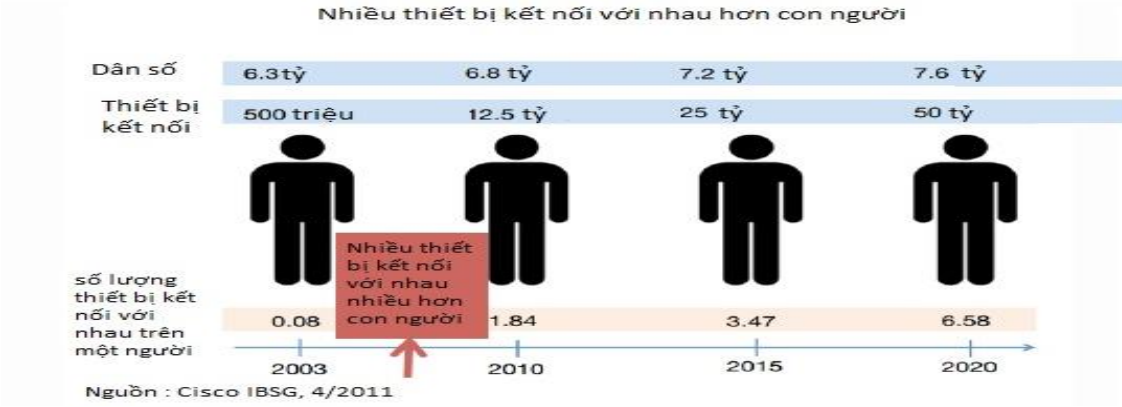
Internet of Things (IoT) là thuật ngữ dùng để chỉ các đối tượng có thể được nhận biết cũng như sự tồn tại của chúng trong một kiến trúc mang tính kết nối. Đây là một viễn cảnh trong đó mọi vật, mọi con vật hoặc con người được cung cấp các định danh và khả năng tự động truyền tải dữ liệu qua một mạng lưới mà không cần sự tương tác giữa con người-với-con người hoặc con người-với-máy tính. IoT tiến hoá từ sự hội tụ của các công nghệ không dây, hệ thống vi cơ điện tử (MEMS) và Internet. Cụm từ này được đưa ra bởi Kevin Ashton vào năm 1999. Ông là một nhà khoa học đã sáng lập ra Trung tâm Auto-ID ở đại học MIT [1-2].



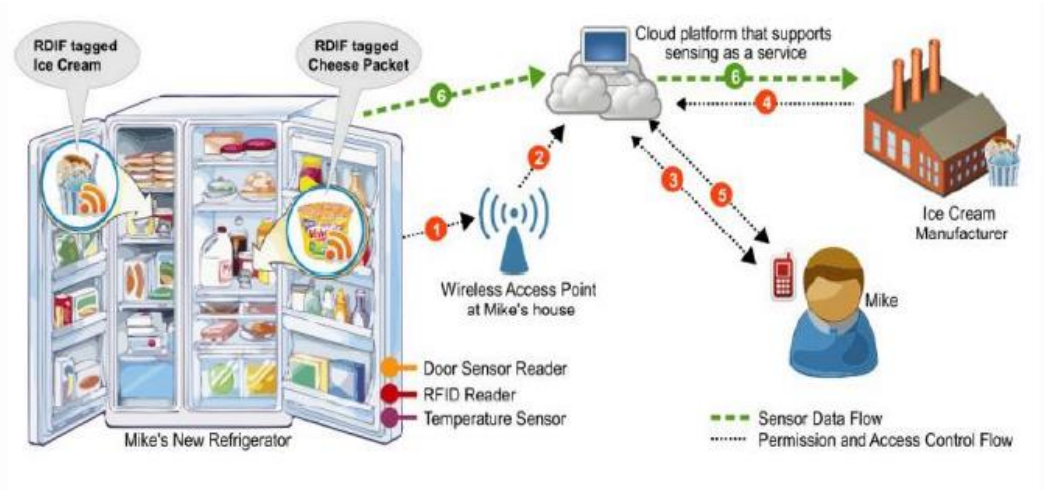
Hình 1.1 Internet kết nối vạn vật

"Thing" - sự vật - trong Internet of Things, có thể là một trang trại động vật với bộ tiếp sóng chip sinh học, một chiếc xe ô tô tích hợp các cảm biến để cảnh báo lái xe khi lốp quá non, hoặc bất kỳ đồ vật nào do tự nhiên sinh ra hoặc do con người

sản xuất ra mà có thể được gán với một địa chỉ IP và được cung cấp khả năng truyền tải dữ liệu qua mạng lưới.



Hình 1.2 Sự gia tăng nhanh chóng của giao tiếp máy-máy



Hình 1.3 Ứng dụng tủ lạnh trong IoT

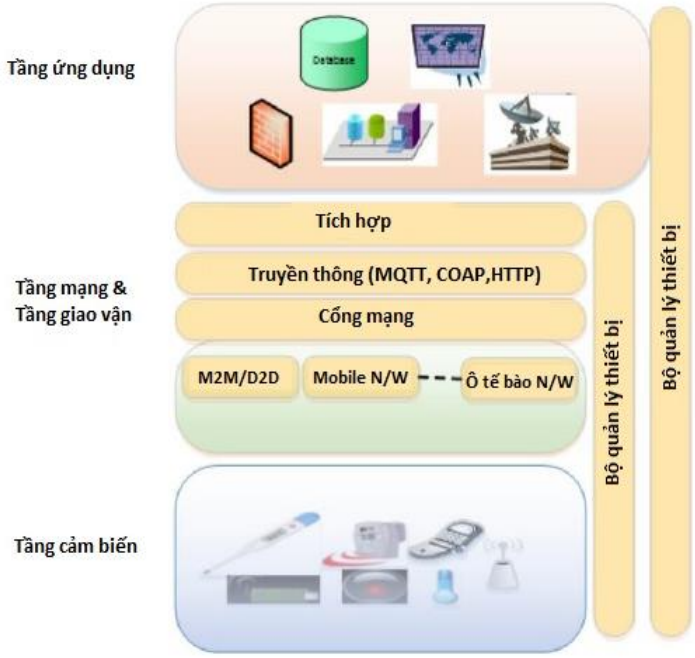
IoT phải có 2 thuộc tính: một là đó phải là một ứng dụng internet. Hai là, nó phải lấy được thông tin của vật chủ. Một ví dụ điển hình cho IoT là tủ lạnh thông minh, nó có thể là một chiếc tủ lạnh bình thường nhưng có gắn thêm các cảm biến bên trong giúp kiểm tra được số lượng các loại thực phẩm có trong tủ lạnh, cảm biến nhiệt độ, cảm biến phát hiện mở cửa,…và các thông tin này được đưa lên internet. Với một danh mục thực phẩm được thiết lập trước bởi người dùng, khi mà một trong các loại thực phẩm đó sắp hết thì nó sẽ thông báo ngay cho chủ nhân nó biết rằng cần phải bổ sung gấp, thậm chí nếu các loại sản phẩm được gắn mã ID thì

nó sẽ tự động trực tiếp gửi thông báo cần nhập hàng đến siêu thị và nhân viên siêu thị sẽ gửi loại thực phẩm đó đến tận nhà.

1.2 Yêu cầu của IoT đối với hạ tầng và các thiết bị truyền thông

Đối với bất kỳ mạng truyền thông nào, kiến trúc phân lớp đảm bảo tính linh hoạt và khả năng thiết lập các dịch vụ mới trong mạng, kiến trúc IoT theo kiến trúc phân lớp. Do IoT bao gồm nhiều lĩnh vực khác nhau, kiến trúc và các thành phần của IoT không hội tụ nhưng kiến trúc IoT thành công nhất là IoT-A. Nhiều mô hình kiến trúc IoT khác cũng có trong thị trường nhưng phổ biến nhất là kiến trúc bốn lớp (như minh họa trên Hình 1.4) [3]:

- Tầng cảm biến: Tầng cảm biến là lớp đối tượng vật lý bao gồm cảm biến, thiết bị truyền động, RFID, thiết bị di động, động cơ, Bluetooth, ... Tầng này thu thập dữ liệu từ môi trường và truyền trên rìa của mạng, nghĩa là gateway hoặc sink.
- Tầng mạng: Tầng này chịu trách nhiệm truyền dữ liệu từ vật thể tới gateway/biên của mạng để tiếp tục xử lý thông tin thu thập được. Các công nghệ truyền khác nhau góp phần tạo ra sự không đồng nhất của IoT như ZigBee, Bluetooth, Wi-Fi, ...
- Tầng ứng dụng: Tầng này xử lý ứng dụng / dịch vụ theo yêu cầu người sử dụng thao tác thông tin thu thập được từ lớp nhận thức và xử lý trong hệ thống xử lý.
- Tầng middleware: Các thiết bị IoT khác nhau trong một miền có thể khác nhau nhưng các thiết bị có thể tương tác với một thiết bị tương thích/giống nhau. Tầng này dịch thông điệp của một thông tin dịch vụ mà không quan tâm đến chi tiết phần cứng. Lớp Middleware được kết hợp với quản lý dịch vụ, giải quyết và đặt tên cho dịch vụ được yêu cầu.



Hình 1.4 Kiến trúc IoT-A

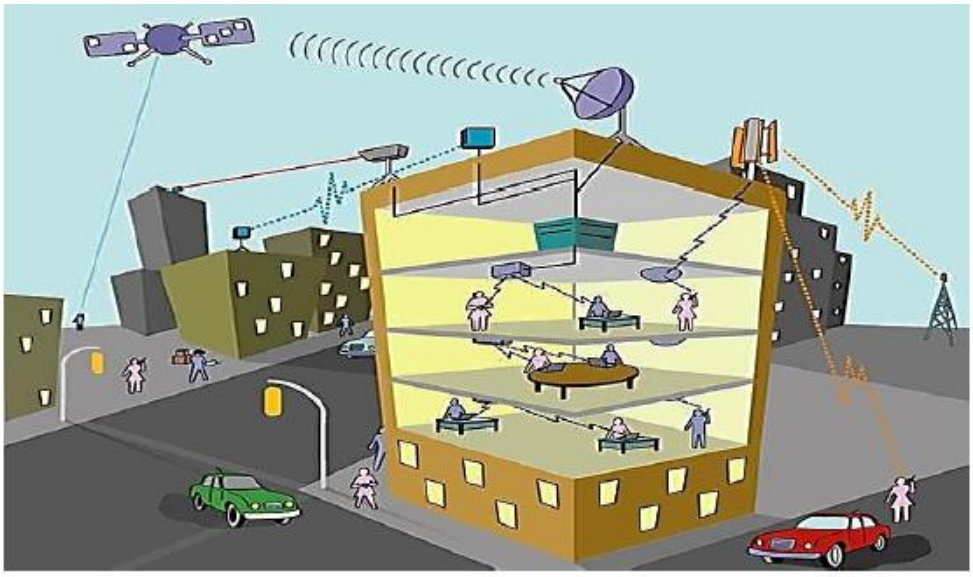
Bên cạnh các lớp chính này, có nhiều thành phần đóng vai trò quan trọng trong việc thu thập, xử lý và quản lý thông tin của IoT. Thành phần edge chịu trách nhiệm cung cấp thông tin qua Internet. Các dịch vụ này có thể là dịch vụ tên miền, mạng phân phối nội dung, tường lửa, cân bằng tải,... Các thành phần dịch vụ Analytics hướng dẫn và tự động hóa quá trình phân tích, phát hiện và hình dung dữ liệu. Các dịch vụ quản lý quy trình giúp quản lý luồng công việc xử lý thông tin và kết nối thiết bị với các dịch vụ tương ứng. Dịch vụ nhận dạng thiết bị xác định người dùng đăng ký dịch vụ trên thiết bị. Dịch vụ xác thực cho phép xác thực người dùng đăng ký với dịch vụ liên quan của nó. Kiến trúc hướng dịch vụ (SOA) giúp cung cấp kiến trúc trừu tượng từ các chi tiết cơ bản và cung cấp các dịch vụ cần thiết. Lựa chọn kết nối Internet cho IoT thay đổi đáng kể tùy thuộc vào từng mục đích và thiết bị cụ thể, ví dụ trong hệ thống mạng liền kề có các yêu cầu khác nhau về hiệu năng và mức độ dịch vụ giữa các hệ thống quản lý và đám mây dữ liệu. Các lựa chọn kết nối điển hình sẵn có ở các khu vực đô thị khác với các lựa chọn được cung cấp bởi các nhà cung cấp dịch vụ ở khu vực nông thôn và chất lượng, độ tin cậy và lưu lượng sẽ có sự phân biệt rõ rệt. Vì vậy, một công nghệ truyền thông thích hợp cho kết nối IoT mang tính thực tế và khả thi về mặt tài chính đối với các dự án

yêu cầu đòi hỏi hiệu năng cao của liên kết giữa các trạm từ xa, trụ sở chính và các giao diện của trung tâm dữ liệu cần xem xét các vấn đề về độ khả dụng, băng thông, chi phí, khả năng quản lý, QoS, SLA và độ tin cậy [4-5].

1.2.1 Các yêu cầu của truyền thông IoT

a) Các phương thức truyền dữ liệu phù hợp

Để kết nối trên diện rộng và khoảng cách xa, thực chất các thiết bị trong thế giới Internet of Things sẽ phải tận dụng rất nhiều kênh truyền tải dữ liệu không dây khác nhau [6]. Trong đó sẽ bao gồm cả mạng điện thoại di động, mạng vệ tinh, một số công nghệ mới như Weightless, LPWAN. Lưu ý khái niệm kênh truyền tải này khác với các "giao thức" giúp các thiết bị nói chuyện được với nhau đã nêu ở trên. Bạn đọc có thể tưởng tượng mỗi kênh truyền sóng giống như một loại cáp (cáp đồng – cáp quang – cáp Ethernet dùng trong gia đình) khác nhau để dễ hình dung. Điều quan trọng là mỗi công nghệ phát sóng sẽ có lợi thế riêng tùy vào hoàn cảnh sử dụng, vì vậy người ta sẽ phải sử dụng kết hợp chúng.



Hình 1.5 Các phương thức truyền thông đa dạng trong IoT

b) Sự cần thiết của Hub và Gateway

Tuy mô hình mạng dạng lưới đã nói ở trên có rất nhiều lợi thế. Nhưng rào cản lớn nhất là sự khác biệt giữa các kênh truyền dữ liệu và các bộ giao thức. Ví dụ

như việc các thiết bị được thiết kế để giao tiếp với nhau bằng ZigBee sẽ gặp khó khăn khi cần trao đổi với dữ liệu với "ngôn ngữ" Z-Wave. Các thiết bị chỉ có thể phát sóng radio tầm gần sẽ khó mà kết nối được với mạng vệ tinh hay trạm phát sóng của mạng điện thoại di động. Vì vậy thực tế trong thế giới mạng Internet of Things ta vẫn cần các thiết bị trung gian, tương tự các hub hay gateway trong hệ thống mạng dây hiện nay – và đây chính là một trong những bước mấu chốt để trào lưu IoT bùng nổ. Theo nhiều dự đoán, các chức năng này sẽ dần được tích hợp thẳng vào các router phổ thông, các thiết bị đầu cuối mà ISP hay hãng truyền hình cung cấp cho người dùng, hay thậm chí là tồn tại trong các sản phẩm như Google Chromecast.

c) Công suất thiết bị

Các tiêu chí hình thức chính của thiết bị khi triển khai các kết nối IoT là phải giá thành thấp, mỏng, nhẹ…và như vậy phần năng lượng nuôi thiết bị cũng sẽ trở nên nhỏ gọn lại, năng lượng tích trữ cũng sẽ trở nên ít đi. Do đó đòi hỏi thiết bị phải tiêu tốn một công suất cực nhỏ (Ultra Low Power) để sử dụng nguồn năng lượng có hạn đó. Bên cạnh đó yêu cầu có những giao thức truyền thông không dây gọn nhẹ hơn, đơn giản hơn, đòi hỏi ít công suất hơn (Low Energy Wireless Technologies) như Zigbee, BLE (Bluetooth low energy), ANT/ANT+, NIKE+,..

1.2.2 Yêu cầu đối với thiết bị truyền thông trong IoT

a) Chi phí thấp

Chi phí luôn luôn quan trọng trong thiết kế IoT và khi xem xét công nghệ WAN công suất thấp [6]. Lý do cơ bản là khi xem xét các ứng dụng truy nhập từ xa, các nhà thiết kế sẽ dự đoán một yêu cầu cho hàng trăm thiết bị đầu cuối, cảm biến, bộ truyền động. Ví dụ như khi chúng ta xem xét một dự án thành phố IoT sáng kiến cho quản lý giao thông thông minh, hàng triệu thiết bị đầu cuối có thể được triển khai trên một diện rộng. Do đó, nhà thiết kế phải xem xét vốn và chi phí hoạt động của một công nghệ WAN công suất thấp khi nhúng vào thiết bị.

b) Tiêu thụ ít năng lượng

Một điểm quan trọng nữa trong thiết là mức độ tiêu thụ năng lượng của công nghệ WAN. Cũng như chi phí, thiết kế sẽ yêu cầu rất nhiều các nút đầu cuối và mỗi một nút sẽ có yêu cầu nguồn năng lượng riêng. Vấn đề với truyền thông không dây vô tuyến là phải mất rất nhiều năng lượng để truyền tải một tín hiệu và hầu hết các nút đầu cuối được hỗ trợ bởi các nguồn pin có tuổi thọ ngắn. Ví dụ như việc chúng ta tưởng tượng đến viễn cảnh phải theo dõi, quan lý và thay hàng triệu viên pin trong các thiết bị đầu cuối của một hệ thống quản lý giao thông trong kiến trúc thành phố thông minh.

c) Tầm hoạt động

Phạm vi hoạt động của công nghệ kết nối Internet mang lại lợi ích về chi phí và hiệu năng. Điều này có thể được thể hiện bằng sự cần thiết của một thiết bị biên kết nối thông qua một điểm truy cập hoặc một giao thức gateway nằm giữa các nút cuối và Internet hoặc một cơ sở hạ tầng công nghệ thông tin trong các công nghệ không dây. Càng cần có ít giao thức gateway và các điểm truy nhập, thiết kế càng trở nên hiệu quả và giảm chi phí của toàn bộ hệ thống.

d) Khả năng mở rộng

Một hệ thống mạng phải có khả năng mở rộng, vì nhu cầu về mạng sẽ không tránh khỏi việc thay đổi theo thời gian. Do đó, các điểm truy nhập sẽ có thể xử lý các trường hợp tăng cường bổ sung mà không cần nâng cấp thêm vào cơ sở hạ tầng. Tương tự, phổ tần số sẽ được cân nhắc cẩn thận đặc biệt là khi hoạt động trong các dải tần số không có cấp phép như nhiễu và ảnh hưởng từ các mạng không dây lân cận hoạt động trên cùng một băng tần và kênh sẽ gây ra xung đột.

1.3 Công nghệ mạng định nghĩa bằng phần mềm

1.3.1 Giới thiệu chung

Mặc dù được sử dụng rộng rãi, song mạng IP truyền thống rất phức tạp và khó quản lý. Để thực hiện các chính sách mạng cấp cao như mong muốn, các nhà khai thác mạng cần phải cấu hình từng thiết bị mạng riêng biệt các thao tác sử dụng ở cấp thấp hơn và thường sử dụng các tập lệnh của nhà cung cấp định nghĩa trước

và mỗi nhà cung cấp lại có các tập lệnh khác nhau. Ngoài việc cấu hình phức tạp, môi trường mạng phải có khả năng chịu lỗi và thích ứng với sự thay đổi của tải [7-9].

Tự động cấu hình và cơ chế tự phản hồi gần như không tồn tại trong các mạng IP hiện tại. Thực thi các chính sách cần thiết trong một môi trường mang tính động như vậy là một việc khó khăn đầy thách thức. Để làm cho nó thậm chí còn phức tạp hơn, các mạng hiện nay cũng được tích hợp theo chiều dọc.

Mặt phẳng điều khiển (nơi đưa ra các quyết định làm xử lý lưu lượng mạng) và mặt phẳng dữ liệu (nơi chuyển tiếp lưu lượng theo các quyết định của mặt phẳng điều khiển) được gắn lại với nhau bên trong các thiết bị mạng, làm giảm tính linh hoạt, gây trở ngại cho sự đổi mới và phát triển của cơ sở hạ tầng mạng.

Sự chuyển đổi từ IPv4 sang IPv6, đã bắt đầu hơn một thập niên trước và phần lớn vẫn chưa hoàn thành, là một minh chứng điển hình cho việc thay đổi, cải tiến hạ tầng mạng, trong khi trên thực tế IPv6 chỉ đơn thuần tượng trưng cho một bản cập nhật giao thức. Do sự trì trệ của các mạng IP hiện tại, một giao thức định tuyến mới có thể mất 5 đến 10 năm nữa để được thiết kế hoàn chỉnh, đánh giá và triển khai. Cũng tương tự như vậy, một phương thức tiếp cận làm clean-slate (bỏ tất cả để làm lại) để thay đổi kiến trúc Internet (ví dụ, thay thế IP), được coi là một nhiệm vụ khó khăn - không khả thi trong thực tế. Cuối cùng, tình trạng này đã nhanh chóng làm tăng vốn chi phí hoạt động của một mạng chạy IP.

Mạng Internet đã và đang tạo ra một xã hội số, một kỷ nguyên mà tất cả mọi vật có thể được kết nối và truy cập bất kì nơi đâu. Cùng với đó là các yêu cầu chất lượng trải nghiệm của người dùng ngày càng cao. Mạng IP truyền thống đứng trước thách thức sống còn, cùng lúc này mạng định nghĩa bằng phần mềm nổi lên với một hệ tư tưởng hoàn toàn mới, một giải pháp đầy hứa hẹn sẽ giải quyết được các vấn đề trong mạng IP truyền thống.

Hệ tư tưởng này đề xuất một kiến trúc mới phá vỡ kiến trúc tích hợp dọc của mạng IP, tách rời phần điều khiển logic từ các bộ định tuyến và bộ chuyển mạch,

tập trung các phần logic này lại để kiểm soát tập trung hệ thống mạng đồng thời đề xuất xây dựng bộ điều khiển tập trung có thể lập trình được. Đứng trước thời cuộc này, SDN đang có cơ hội thực hiện một cuộc cách mạng trong ngành truyền thông.

Kỷ nguyên của SDN được tạo ra đầu tiên để thể hiện các ý tưởng nghiên cứu xoay quanh giao thức OpenFlow của đại học Stanford. Như thiết kế ban đầu, SDN thể hiện một kiến trúc mạng mới mà trạng thái chuyển tiếp trong mặt phẳng dữ liệu được quản lý bởi một bộ điều khiển ở xa, được tách biệt khỏi kiến trúc truyền thống của các phần tử mạng. Ngành công nghiệp mạng có rất nhiều cơ hội để biến chuyển ngay từ thời điểm đầu tiên của thời đại SDN, tất cả mọi thứ kể cả phần mềm đều được SDN hoá. Cộng đồng phát triển SDN đã đưa ra định nghĩa cho SDN như một kiến trúc mạng với bốn nguyên lý sau:

- Mặt phẳng dữ liệu và mặt phẳng điều khiển được tách rời nhau. Các chức năng điều khiển được đưa ra khỏi thiết bị mạng và các thiết bị này trở thành các phần tử chuyển mạch (gói tin) đơn giản.
- Các quyết định chuyển tiếp được dựa trên cơ chế luồng dữ liệu, hay còn gọi là flow-base, thay vì dựa trên địa chỉ đích. Một luồng dữ liệu được định nghĩa là một dòng so khớp, khi các giá trị của một trường trong gói tin phù hợp với một tiêu chí nào đấy thì một hành động sẽ được thiết lập. Trong SDN, một luồng dữ liệu là một chuỗi của các gói tin giữa một nguồn và một đích. Tất cả các gói tin của một luồng sẽ được xử lý như nhau tại thiết bị mạng. Việc hình thành nên khái niệm luồng dữ liệu cho phép đồng nhất cách xử lý tại các thiết bị mạng, dù là bộ định tuyến, bộ chuyển mạch hay là tường lửa, …
- Chức năng điều khiển được đưa đến một phần tử bên ngoài hạ tầng mạng truyền tải và được gọi là bộ điều khiển SDN hoặc hệ điều hành mạng (NOS). NOS là một nền tảng phần mềm chạy trên các máy chủ và cung cấp các tài nguyên cần thiết, điều khiển các thiết bị mạng tập trung.

- Hệ thống mạng có thể lập trình điều khiển thông qua các ứng dụng chạy phía trên nền tảng NOS, và NOS tương tác với các thiết bị hạ tầng mạng (Data plane) nằm ở phía dưới. Đây chính là tư tưởng chủ đạo trong SDN.

Đã có rất nhiều cuộc tranh luận để bàn về vấn đề mạng IP và mạng SDN loại nào tốt hơn. Mặc dù cả hai đều có những ưu nhược điểm riêng nhưng với những thuộc tính quan trọng như thân thiện với người sử dụng, chi phí triển khai và độ phức tạp của kiến trúc mạng giảm thì người ta cho rằng mạng SDN phù hợp hơn so với mạng IP. Và SDN ngày càng chiếm được thiện cảm, quan tâm của các nhà cung cấp dịch vụ mạng, trung tâm dữ liệu lớn trên thế giới.

Mạng định nghĩa bằng phần mềm có một số ưu điểm sau:

- Dựa vào SDN các nhà quản trị có thể kiểm soát mạng một cách đơn giản và hiệu quả mà không cần có truy cập trực tiếp đến hạ tầng phần cứng. Thêm vào đó SDN cung cấp một cơ chế điều khiển duy nhất đối với cơ sở hạ tầng mạng và giảm bớt sự phức tạp của các quá trình xử lý thông qua sự tự động hóa. Điều đó rất có lợi đối với các nhà mạng để có thể quản lý các thay đổi của mạng theo thời gian thực. Thời gian thực ở đây có nghĩa là các nhà mạng có thể đáp ứng gần như là tức thời các yêu cầu của khách hàng, xử lý nhanh các sự cố trong mạng và hoạt động tự động.
- Các nhà quản trị còn có thể điều khiển việc thay đổi cần thiết đúng thời điểm ở bất cứ nơi đâu. Việc truy cập từ xa và thay đổi mạng có thể được thực hiện bởi hệ thống truy cập dựa trên vai trò của người quản trị (Role based access system là một hệ thống cho phép người dùng cá nhân dựa trên vai trò xác định trong doanh nghiệp để có thể thực hiện các thay đổi của mạng). Hệ thống truy cập này được cung cấp các giải pháp bảo mật để có thể loại bỏ sự tấn công của các tin tặc vào mạng. Đối với mạng IP, việc truy cập và thay đổi từ xa lại không thể thực hiện được. Nhà quản trị phải có quyền truy cập và chỉnh sửa cấu hình bằng tay để thực hiện bất kỳ sự thay đổi chính sách nào trên mạng. Việc thay đổi một chính sách mạng dẫn đến

việc tác động trực tiếp đến phần cứng đó làm cho hệ thống mạng trở nên cứng nhắc.

- SDN cho phép sử dụng không hạn chế và có thể thay đổi các chính sách mạng để phát hiện sự xâm nhập, tường lửa và tạo sự cân bằng với sự thay đổi của phần mềm. Điều đó làm cho sự quản lý mạng trở nên linh hoạt hơn.
- Mạng SDN có khả năng phân tách phần điều khiển và phần chuyển tiếp dữ liệu, với khả năng đó cho phép người quản trị có thể tương tác và thay đổi các luồng dữ liệu, đảm bảo các gói dữ liệu không phải xếp hàng đợi và làm giảm hiệu suất mạng. Một lợi thế quan trọng hơn nữa của mạng SDN là chi phí dành cho nó rất thấp. Nó rẻ hơn so với mạng IP bởi vì nó không yêu cầu nhiều người làm việc và các công ty đều có thể cắt hết chi phí của các kỹ sư hệ thống và chỉ cần một vài quản trị viên hệ thống là đủ.

Bên cạnh những ưu điểm đó thì SDN cũng còn tồn tại một số nhược điểm so với mạng IP:

- Vấn đề đầu tiên là an toàn, nếu tin tặc có thể tấn công vào hệ thống thì chúng có thể truy cập các thiết lập và thay đổi chúng bất kỳ ở nơi đâu, tại thời điểm nào. Và chúng có thể truy cập bất kỳ tập tin được mã hóa nào miễn là nó ở trong mạng. Đối với mạng IP thì điều này không thể xảy ra bởi để có thể truy cập vào mạng ta phải có quyền truy cập vào phần cứng của nó. Hầu hết các công ty chỉ cho phép một số cá nhân được quyền đó bởi vậy hệ thống sẽ an toàn và ít có khả năng bị truy cập bởi các tin tặc.
- Thứ hai đó là quá trình triển khai SDN không thể hoàn thiện trong chốc lát mà nó phải theo từng bước một. Chúng ta không thể một lúc thay thế toàn bộ các thiết bị hiện có thành bộ chuyển mạch OpenFlow được bởi vì điều đó rất tốn kém.
- Thứ ba, SDN là một kiến trúc mạng kiểu mới, các giao thức tương tác giữa các controller với nhau còn chưa được phát triển toàn diện nên việc phát triển mạng SDN trên phạm vi toàn cầu vẫn còn nhiều hạn chế.

1.3.2 Kiến trúc mạng định nghĩa bằng phần mềm

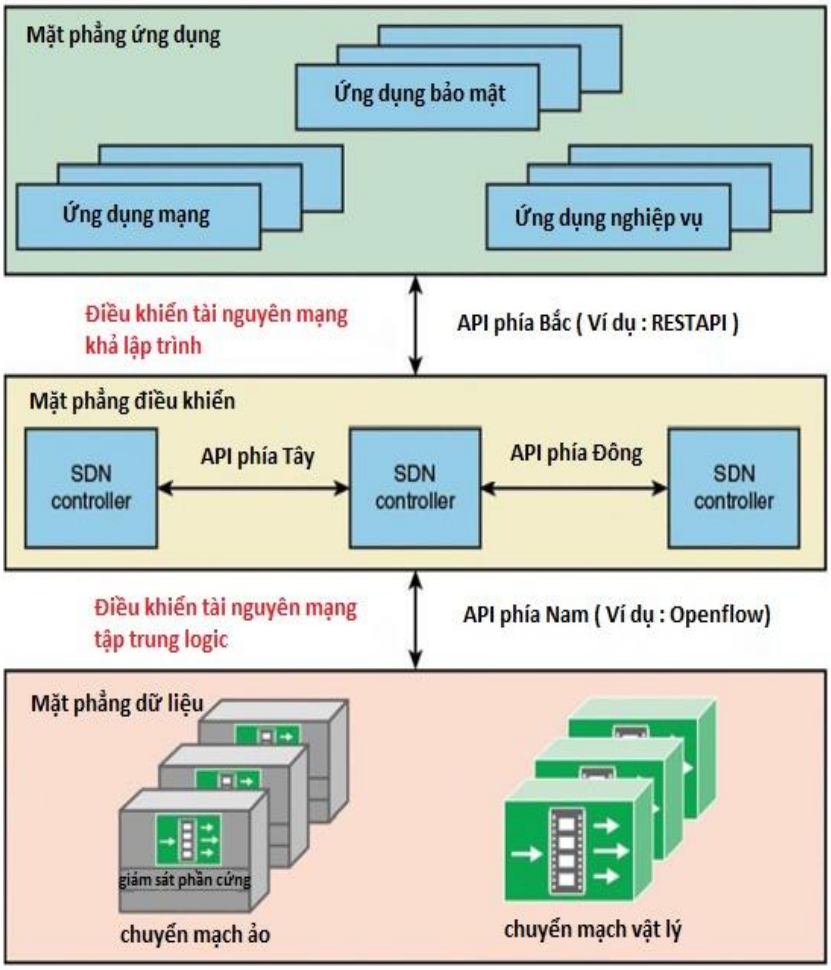
a) Kiến trúc chức năng SDN

SDN là một kiến trúc mới được thiết kế để mạng nhanh hơn, sử dụng chi phí hiệu quả hơn. Mô hình kiến trúc SDN như mô tả trong Hình 1.6. Mặt phẳng dữ liệu bao gồm các bộ chuyển mạch vật lý và bộ chuyển mạch ảo. Các bộ chuyển mạch này chịu trách nhiệm chuyển tiếp các gói tin. Việc thực thi nội bộ các bộ đệm, các tham số ưu tiên, và các cấu trúc dữ liệu liên quan đến việc chuyển tiếp có thể phụ thuộc vào nhà cung cấp. Tuy nhiên, mỗi bộ chuyển mạch phải thực hiện một mô hình, hoặc trừu tượng hóa, gói tin chuyển tiếp đó là điểm thống nhất và mở cho các bộ điều khiển SDN. Mô hình này được định nghĩa trong các điều khoản của một giao diện lập trình ứng dụng mở (API) giữa mặt phẳng điều khiển và mặt phẳng dữ liệu (southbound API). Ví dụ nổi bật nhất của một API mở là OpenFlow. Đặc tả OpenFlow định nghĩa cả giao thức giữa mặt phẳng điều khiển và mặt phẳng dữ liệu và một API mà mặt phẳng điều khiển có thể gọi đến giao thức OpenFlow [9, 10].

Bộ điều khiển SDN có thể được thực hiện trực tiếp trên một máy chủ hoặc trên một máy chủ ảo. OpenFlow hoặc một số API mở khác được sử dụng để điều khiển các thiết bị chuyển mạch trong mặt phẳng dữ liệu. Ngoài ra, các bộ điều khiển sử dụng thông tin về khả năng và nhu cầu lấy được từ thiết bị mạng. Bộ điều khiển SDN có các API phía Bắc, cho phép các nhà phát triển và nhà quản lý mạng triển khai một loạt các sản phẩm off-the-shelf và các ứng dụng mạng được xây dựng tùy chỉnh, rất nhiều trong số đó không khả thi trước khi xuất hiện SDN. Tuy nhiên, không có tiêu chuẩn hóa cho API phía bắc cũng như sự đồng thuận để mở một API phía bắc. Một số nhà cung cấp cung cấp một API dựa trên REST để cung cấp một giao diện lập trình cho bộ điều khiển SDN của họ.

Tại mặt phẳng ứng dụng là một loạt các ứng dụng tương tác với bộ điều khiển SDN. Các ứng dụng SDN là các chương trình có thể sử dụng một khung nhìn trừu tượng về mạng cho mục tiêu ra quyết định. Các ứng dụng này truyền tải yêu cầu về mạng và hành vi mạng mong muốn với bộ điều khiển SDN thông qua API

hướng Bắc. Ví dụ về ứng dụng như là mạng lưới tiết kiệm năng lượng, giám sát an ninh, kiểm soát truy cập, và quản lý mạng.

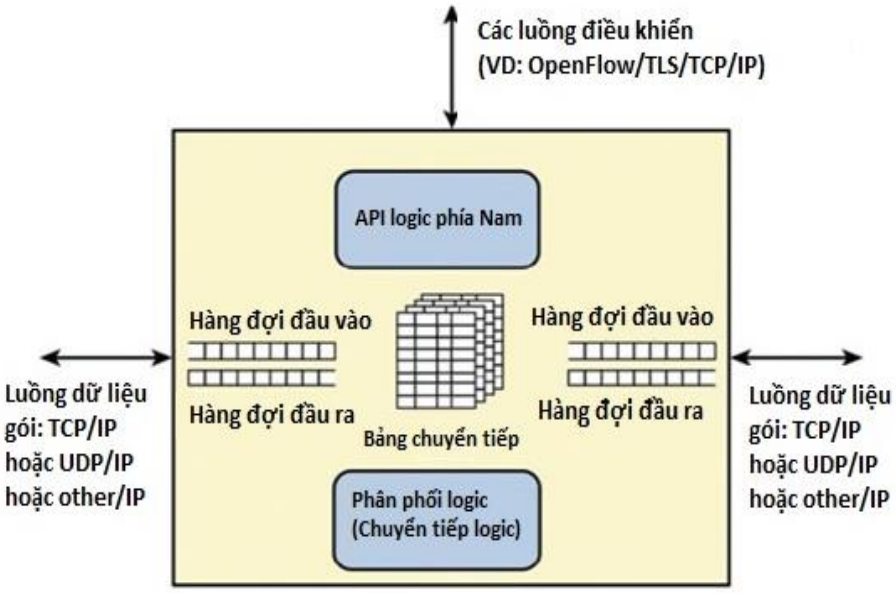


Hình 1.6 Kiến trúc SDN

b) Mặt phẳng dữ liệu

Mặt phẳng dữ liệu SDN, còn được gọi là lớp tài nguyên trong ITU-T Y.3300 và cũng được gọi là lớp cơ sở hạ tầng, là nơi các thiết bị chuyển tiếp mạng thực hiện vận chuyển và xử lý dữ liệu theo các quyết định của mặt phẳng điều khiển SDN. Các đặc tính quan trọng của các thiết bị mạng trong một mạng lưới SDN là những thiết bị thực hiện chức năng chuyển tiếp đơn giản, không có phần mềm nhúng để tạo các quyết định tự trị. Hình 1.4 cũng mô tả các chức năng được thực hiện bởi các thiết bị mạng tại mặt phẳng dữ liệu (như các thiết bị chuyển mạch). Các chức năng bao gồm:

- Chức năng hỗ trợ điều khiển: tương tác với lớp điều khiển SDN để hỗ trợ khả năng lập trình qua các giao diện điều khiển tài nguyên. Bộ chuyển mạch trao đổi với bộ điều khiển và bộ điều khiển quản lý bộ chuyển mạch qua giao thức chuyển mạch OpenFlow.
- Chức năng chuyển tiếp dữ liệu: nhận luồng dữ liệu đến từ các thiết bị mạng khác và các hệ thống đầu cuối và chuyển tiếp chúng theo đường chuyển tiếp dữ liệu đã được tính toán và thiết lập theo các quy tắc được định nghĩa bởi các ứng dụng SDN.



Hình 1.7 Thiết bị mạng mặt phẳng dữ liệu

Những quy tắc chuyển tiếp được sử dụng bởi thiết bị mạng được thể hiện trong các bảng chuyển tiếp chỉ ra cho các loại gói tin xác định địa chỉ chặng tiếp theo trong tuyến đường. Ngoài việc chuyển tiếp đơn giản một gói tin, thiết bị mạng có thể thay đổi tiêu đề gói tin trước khi chuyển tiếp hoặc hủy bỏ gói. Các gói tin đến có thể được đặt trong một hàng đợi đầu vào, đang chờ xử lý bởi thiết bị mạng và các gói chuyển tiếp thường được đặt trong một hàng đợi đầu ra, đang chờ truyền. Thiết bị mạng trong Hình 1.7 được hiển thị với ba cổng I/O: một cổng điều khiển cung cấp truyền thông với một bộ điều khiển SDN, và hai cổng cho đầu vào và đầu

ra của các gói dữ liệu. Đây là một ví dụ đơn giản. Thiết bị mạng có thể có nhiều cổng để giao tiếp với nhiều bộ điều khiển SDN và có thể có nhiều hơn hai cổng I/O cho luồng gói tin vào và ra khỏi thiết bị.

Các giao thức mặt phẳng dữ liệu: Hình 1.7 cho thấy các giao thức được hỗ trợ bởi thiết bị mạng. Luồng gói dữ liệu bao gồm các luồng các gói tin IP. Bảng chuyển tiếp cần định nghĩa các mục dựa trên các trường trong các tiêu đề giao thức cấp trên, chẳng hạn như TCP, UDP, hoặc một số các giao thức lớp giao vận hoặc ứng dụng khác. Thiết bị mạng kiểm tra tiêu đề IP và có thể cả các tiêu đề khác trong mỗi gói và đưa ra quyết định chuyển tiếp. Luồng lưu lượng quan trọng khác được chuyển qua giao diện lập trình ứng dụng hướng nam (API), bao gồm các đơn vị dữ liệu giao thức OpenFlow (PDU) hoặc một số lưu lượng giao thức API hướng nam tương tự.

c) Mặt phẳng điều khiển

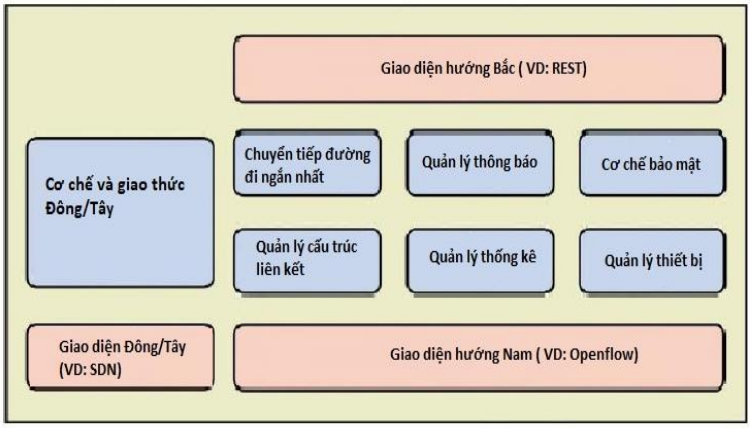
Lớp điều khiển SDN ánh xạ các yêu cầu dịch vụ lớp ứng dụng vào các lệnh cụ thể và chỉ thị cho thiết bị chuyển mạch mặt phẳng dữ liệu và cung cấp các ứng dụng với các thông tin về cấu hình và hoạt động của mặt phẳng dữ liệu. Lớp điều khiển được thực hiện như một máy chủ hoặc một tập các máy chủ hợp tác được gọi là các bộ điều khiển SDN. Phần này cung cấp tổng quan về chức năng mặt phẳng điều khiển. Sau đó, chúng ta xem các giao thức và tiêu chuẩn cụ thể được thực hiện trong mặt phẳng điều khiển. Hình 1.8 chỉ ra các chức năng được thực hiện bởi các bộ điều khiển SDN.

Các chức năng này bao gồm:

- Chuyển tiếp đường đi ngắn nhất: Sử dụng thông tin định tuyến thu thập được từ các bộ chuyển mạch để thiết lập các tuyến đường thích hợp.
- Quản lý thông báo: Nhận, xử lý và chuyển tiếp đến các sự kiện ứng dụng, chẳng hạn như thông báo cảnh báo, cảnh báo an ninh, và thay đổi trạng thái.

- Cơ chế bảo mật: Cung cấp cách ly và thực thi bảo mật giữa các ứng dụng và các dịch vụ.
- Quản lý cấu trúc liên kết: Xây dựng và duy trì thông tin cấu hình kết nối chuyển mạch
- Quản lý thống kê: Thu thập dữ liệu về lưu lượng chuyển qua các thiết bị chuyển mạch.
- Quản lý thiết bị: cấu hình thông số và thuộc tính chuyển mạch và quản lý các bảng lưu lượng.

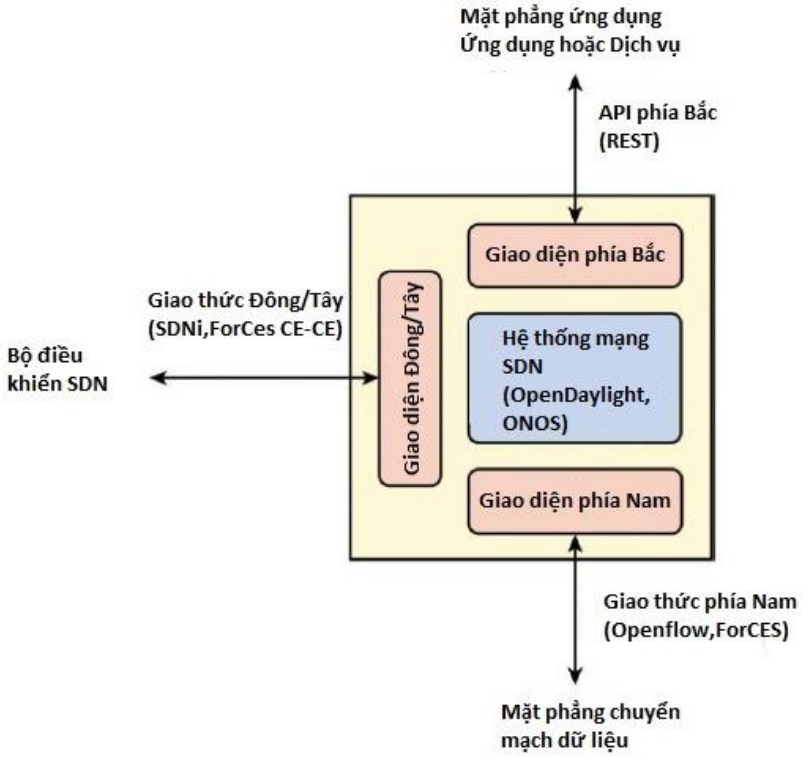
Các chức năng được cung cấp bởi bộ điều khiển SDN có thể được xem như là một hệ thống điều hành mạng (NOS). Như với một hệ điều hành thông thường, NOS cung cấp các dịch vụ thiết yếu, các giao diện lập trình ứng dụng phổ biến và trừu tượng hóa các phần tử lớp dưới cho các nhà phát triển mạng. Các chức năng của một NOS quản lý các mạng mà không quan tâm đến chi tiết đến các đặc tính thiết bị mạng. Giao diện phía bắc cung cấp một phương tiện thống nhất cho phép các nhà phát triển ứng dụng và người quản lý mạng truy cập dịch vụ SDN và thực hiện các nhiệm vụ quản lý mạng. Hơn nữa, giao diện phía bắc được xác định rõ ràng cho phép các nhà phát triển tạo ra phần mềm độc lập không chỉ với mặt phẳng dữ liệu mà còn ở mức độ lớn có thể sử dụng được với nhiều máy chủ điều khiển SDN.



Hình 1.8 Các giao diện và mặt phẳng chức năng điều khiển

Một số sáng kiến khác nhau, cả thương mại và mã nguồn mở, đã đưa ra các thực thi bộ điều khiển SDN. Một số bộ điều khiển SDN gồm OpenDaylight, ONOS, POX, Beacon, Floodlight, Ryu và Onix.

- *Giao diện hướng nam*



Hình 1.9 Các giao diện bộ điều khiển SDN

Giao diện hướng nam cung cấp kết nối logic giữa bộ điều khiển SDN và thiết bị chuyển mạch dữ liệu (Hình 1.9). Một số sản phẩm điều khiển và cấu hình chỉ hỗ trợ một giao thức hướng nam. Cách tiếp cận linh hoạt hơn là sử dụng một lớp trừu tượng phía nam cung cấp một giao diện chung cho các chức năng mặt phẳng điều khiển trong khi hỗ trợ nhiều API hướng nam. API hướng nam được sử dụng phổ biến là OpenFlow. Các giao diện hướng nam khác bao gồm giao thức quản lý cơ sở dữ liệu Open vSwitch (OVSDB), chuyển tiếp và điều khiển phân tách phần tử (ForCES) và POF.

- *Giao diện hướng bắc*

Giao diện phía bắc cho phép các ứng dụng truy cập các dịch vụ và các chức năng của mặt phẳng điều khiển mà không cần phải biết chi tiết của các thiết bị

chuyển mạch mạng bên dưới. Các giao diện phía bắc thường được xem như một API phần mềm chứ không phải là một giao thức. Không giống như các giao diện hướng nam và hướng đông /hướng Tây, nơi mà một số giao diện không đồng nhất đã được định nghĩa, không có tiêu chuẩn được chấp nhận rộng rãi cho giao diện phía bắc. Kết quả là một số API duy nhất đã được phát triển cho các bộ điều khiển khác nhau, làm phức tạp thêm nỗ lực phát triển ứng dụng SDN. Để giải quyết vấn đề này ONF đã thành lập nhóm làm việc giao diện hướng bắc (NBI-WG) vào năm 2013 với mục tiêu định nghĩa và chuẩn hóa một số APIs hướng Bắc rộng rãi.

- *Định tuyến*

Giống như bất kỳ mạng nào hoặc mạng Internet, mạng SDN cũng yêu cầu chức năng định tuyến. Nói chung, chức năng định tuyến bao gồm một giao thức để thu thập thông tin về cấu hình và điều kiện lưu lượng của mạng, và một thuật toán để thiết kế các tuyến đường qua mạng. Có hai loại giao thức định tuyến: các giao thức định tuyến nội (IRPs) hoạt động bên trong một hệ thống tự trị (AS) và các giao thức định tuyến bên ngoài (ERPs) hoạt động giữa hệ thống tự trị.

Một IRP liên quan đến việc khám phá topo của các bộ định tuyến trong một AS và sau đó xác định tuyến đường tốt nhất đến từng điểm đến dựa trên các số liệu khác nhau. Hai giao thức IRP được sử dụng rộng rãi là giao thức OSPF và EIGRP. Giao thức ERP không cần thu thập thông tin chi tiết về lưu lượng. Thay vào đó, mối quan tâm chính của một ERP là để xác định khả năng kết nối với các mạng khác và các hệ thống đầu cuối bên ngoài AS. Do đó, ERP thường được thực hiện chỉ trong các nút biên mà kết nối AS với AS khác. Giao thức Border Gateway Protocol (BGP) thường được sử dụng cho ERP.

Theo truyền thống, chức năng định tuyến được phân phối giữa các bộ định tuyến trong mạng. Mỗi router có trách nhiệm xây dựng một cấu hình của mạng. Đối với định tuyến nội, mỗi router cũng phải thu thập thông tin về kết nối và trễ và sau đó tính toán các tuyến đường thích hợp đến mỗi địa chỉ IP đích. Tuy nhiên, trong một mạng SDN, chức năng định tuyến được tập trung hóa trong các bộ điều khiển SDN. Bộ điều khiển có thể phát triển một cái nhìn nhất quán về trạng thái mạng cho

việc tính toán đường đi ngắn nhất, và có thể thực hiện các chính sách định tuyến nhận thức ứng dụng. Các bộ chuyển mạch trong mặt phẳng dữ liệu được giải phóng khỏi gánh nặng xử lý và lưu trữ liên quan đến định tuyến, dẫn đến hiệu suất được cải thiện.

Ứng dụng định tuyến tập trung thực hiện hai chức năng riêng biệt: phát hiện liên kết và quản lý cấu hình.

- Đối với phát hiện liên kết, chức năng định tuyến cần phải nhận thức được các liên kết giữa các thiết bị chuyển mạch trong mặt phẳng dữ liệu. Ngoài ra, phát hiện liên kết phải được thực hiện giữa một bộ định tuyến và một hệ thống máy chủ lưu trữ và giữa một bộ định tuyến trong miền của bộ điều khiển này và một bộ định tuyến trong một miền lân cận. Khám phá được kích hoạt bởi lưu lượng truy cập không xác định vào miền mạng của bộ điều khiển từ máy chủ hoặc từ một bộ định tuyến lân cận.
- Quản lý cấu hình duy trì thông tin cấu trúc mạng cho mạng và tính toán các tuyến trong mạng. Tính toán đường bao gồm việc xác định đường đi ngắn nhất giữa hai nút dữ liệu mặt phẳng hoặc giữa một nút mặt phẳng dữ liệu và một máy chủ.

d) Mặt phẳng ứng dụng

Mặt phẳng ứng dụng chứa các ứng dụng và các dịch vụ mà định nghĩa, giám sát và điều khiển hành vi và tài nguyên mạng. Các ứng dụng này tương tác với mặt phẳng điều khiển SDN thông qua các giao diện điều khiển ứng dụng, cho lớp điều khiển SDN để tự động tùy chỉnh hành vi và các thuộc tính tài nguyên mạng. Lập trình ứng dụng SDN sử dụng cách nhìn trừu tượng về các tài nguyên mạng được cung cấp bởi lớp điều khiển SDN bằng các phương tiện thông tin và các mô hình dữ liệu thông qua giao diện điều khiển ứng dụng. Phần này cung cấp một cách tổng quan về chức năng của mặt phẳng ứng dụng, được mô tả trong Hình 1.10. Các thành phần trong hình này được phân tích thông qua phương pháp tiếp cận từ dưới lên.

Hình 1.10 Các giao diện và các chức năng mặt phẳng SDN

- *Giao diện hướng bắc*

Hình 1.10 chỉ ra rằng giao diện hướng bắc có thể là một giao diện cục bộ hoặc từ xa. Đối với một giao diện cục bộ, các ứng dụng SDN đang chạy trên cùng một máy chủ với phần mềm mặt phẳng điều khiển (hệ điều hành mạng của bộ điều khiển). Ngoài ra, các ứng dụng có thể chạy trên các hệ thống từ xa và giao diện hướng bắc là một giao thức hoặc giao diện lập trình ứng dụng (API) kết nối các ứng dụng với hệ điều hành mạng (NOS) của bộ điều khiển chạy trên máy chủ trung tâm. Cả hai kiến trúc trên đều có thể được thực hiện. Một ví dụ về giao diện phía bắc là API REST cho hệ điều hành mạng Ryu SDN.

- *Lớp trừu tượng hóa dịch vụ mạng*

RFC7426 định nghĩa một lớp trừu tượng hóa dịch vụ mạng giữa mặt phẳng điều khiển và mặt phẳng ứng dụng và mô tả nó như là một lớp cung cấp trìu tượng hóa dịch vụ có thể được sử dụng bởi các ứng dụng và dịch vụ. Một số chức năng được cung cấp bởi lớp này trong kiến trúc SDN:

Lớp này có thể cung cấp một cái nhìn trừu tượng về các tài nguyên mạng mà có thể che giấu các chi tiết của các thiết bị mặt phẳng dữ liệu ở dưới.

- Lớp này có thể cung cấp một cái nhìn khái quát về chức năng của mặt phẳng điều khiển, do đó các ứng dụng có thể được hoạt động trên một loạt các hệ điều hành mạng của bộ điều khiển.
- Chức năng này tương tự như chức năng của một thiết bị giám sát máy ảo mà tách riêng các ứng dụng từ hệ điều hành bên dưới và phần cứng lớp dưới.
- Lớp này có thể cung cấp khả năng ảo hóa mạng cho phép các cách nhìn khác nhau về cơ sở hạ tầng mặt phẳng dữ liệu bên dưới.

Có thể cho rằng lớp trừu tượng hóa dịch vụ mạng có thể được coi là một phần của giao diện hướng bắc, với chức năng kết hợp trong mặt phẳng điều khiển hoặc mặt phẳng ứng dụng.

➢ *Các ứng dụng mạng*

Có rất nhiều các ứng dụng mạng có thể được triển khai cho một SDN. Hình 1.8 chỉ ra có sáu loại ứng dụng SDN.

➢ *Giao diện người sử dụng*

Giao diện người sử dụng cho phép người dùng cấu hình các tham số trong các ứng dụng SDN và để tương tác với các ứng dụng khác có hỗ trợ tương tác người dùng. Có hai giao diện người dung, gồm giao diện cục bộ và giao diện điều khiển từ xa. Một người dùng được đặt cùng với máy chủ ứng dụng SDN (có hoặc không bao gồm mặt phẳng điều khiển) có thể sử dụng bàn phím / màn hình của máy chủ. Cụ thể hơn, người dùng sẽ đăng nhập vào máy chủ ứng dụng qua mạng hoặc phương tiện truyền thông.

1.4 IoT định nghĩa bằng phần mềm

Với sự phát triển của Internet, điện thoại thông minh và đặc biệt là các thiết bị cảm biến, công nghệ vạn vật kết nối (IoT) đang trở thành xu hướng mới của thế giới. Mặc dù IoT vẫn đang trong giai đoạn mới bắt đầu được áp dụng triển khai, phạm vi ứng dụng đã rất lớn và danh mục ứng dụng IoT đang ngày càng kéo dài. Công nghệ IoT hứa hẹn khả năng giải quyết và đáp ứng được nhiều nhu cầu đa

dạng, bao gồm cải thiện, nâng cao năng suất khai thác, sử dụng nguồn tài nguyên và quản lý cơ sở hạ tầng, đồng thời công nghệ IoT cũng có thể tác động trực tiếp lên tiến trình nâng cao chất lượng sống và sức khỏe của con người cũng như đang góp phần tạo ra các chuyển biến tích cực trong đời sống, xã hội hiện đại.

Tuy nhiên, IoT phát triển đồng nghĩa với nhu cầu về lưu lượng và thiết bị mạng tăng, báo cáo ước tính có khoảng 24 tỷ thiết bị IoT trên thế giới vào năm 2020. Cùng với sự tăng trưởng nhanh chóng của các thiết bị kết nối mạng và nhu cầu dịch vụ mới, hạ tầng mạng truyền thông cần có khả năng đáp ứng sự bùng nổ về lưu lượng, hỗ trợ kết nối với băng thông hỗn tạp và các yêu cầu chất lượng dịch vụ đa dạng cũng như có khả năng hỗ trợ triển khai dịch vụ mới nhanh chóng. Bên cạnh đó, cùng với vấn đề gia tăng về lưu lượng và số lượng. mạng một cách bùng nổ thì vấn đề quản lý và kiểm soát là một nhiệm vụ phức tạp cho các hệ thống mạng phân phối lớn.

Nhằm hỗ trợ cho số lượng lớn ứng dụng IoT với các yêu cầu phong phú và đa dạng về tính năng, chủng loại, chất lượng, …, nhu cầu phát triển hạ tầng truyền thông IoT với các kiến trúc và thiết bị truyền thông mới, hiệu quả và giá thành phải chăng đang ngày càng trở nên cấp thiết hơn. Các công nghệ truyền thông IoT đòi hỏi phải đáp ứng các yêu cầu ngày càng khắt khe hơn như việc tiêu thụ năng lượng ít hơn, tiêu tốn ít băng thông hơn cũng như việc "trong suốt" với các môi trường đa dạng. Hơn nữa, các thiết bị truyền thông IoT cũng cần phải có khả năng cung cấp các dịch vụ IoT hỗn tạp, nhiều mức độ về nhu cầu chất lượng dịch vụ với chi phí thấp và tính linh hoạt cao.

Như đã phân tích trong các phần trên, công nghệ mạng định nghĩa bằng phần mềm đang nổi lên là một trong những giải pháp công nghệ mạng linh hoạt và hiệu quả nhất có khả năng ảo hóa nguồn tài nguyên mạng, cấp phát theo yêu cầu và quản lí, hỗ trợ hiệu quả các yêu cầu và đòi hỏi phức tạp của các hệ thống IoT. Trong SDN, chức năng của các thiết bị mạng được thay thế bằng các thiết bị chuyển tiếp luồng dữ liệu và phần thông minh của thiết bị được thực hiện bằng phần mềm, khả lập trình và triển khai trung tâm điều khiển (bộ điều khiển SDN). Các sản phẩm

SDN đã bước đầu được mô phỏng thành công và xuất hiện một số sản phẩm thương mại trên thị trường một số nước phát triển. Tuy nhiên, hầu hết các sản phẩm đều mới chỉ tập trung vào hạ tầng mạng lõi, các trung tâm dữ liệu với dung lượng lớn và giá thành còn rất cao.

Xu hướng ứng dụng giải pháp SDN cho IoT (SD-IoT) mới được đề xuất và bắt đầu được nghiên cứu, phát triển trong vài năm gần đây. Trên thế giới, một số các công trình nghiên cứu về SD-IoT và một vài mô phỏng xây dựng hệ thống SDN mẫu cỡ nhỏ đã được thực hiện. Tuy nhiên, các hệ thống này mới chỉ chủ yếu tập trung mô phỏng về phần cứng nhằm xây dựng thiết bị mạng theo một số tính năng của thiết bị mạng IP hiện nay như firewall, bộ định tuyến, ... mà chưa xem xét đến khả năng đáp ứng các yêu cầu và đòi hỏi của các ứng dụng IoT. Tại Việt Nam công nghệ IoT cũng đang được quan tâm nghiên cứu và ứng dụng ngày càng rộng rãi trong nhiều lĩnh vực. Cũng như thị trường quốc tế, thị trường Việt Nam cũng có nhu cầu cho các giải pháp thiết bị và hạ tầng truyền thông IoT. Việc làm chủ công nghệ sẽ cung cấp các đặc tính phù hợp yêu cầu thị trường và người sử dụng ở Việt Nam. Tuy nhiên việc nghiên cứu các công nghệ và thiết bị truyền thông IoT vẫn còn khá mới mẻ. Theo khảo sát của nhóm nghiên cứu thì trong nước hiện nay mới chỉ có các nhóm tại VNPT và Viettel đang có những nghiên cứu triển khai bước đầu về công nghệ SDN với mục tiêu ứng dụng trong các trung tâm dữ liệu (clouding).

Mặc dù việc phát triển các thiết bị truyền thông SDN kích thước nhỏ đã có một số kết quả từ các nhóm nghiên cứu ngoài nước, nhưng những sản phẩm này vẫn chưa được thương mại hóa và chưa xem xét đến các yêu cầu của ứng dụng IoT cũng như còn có những khó khăn nhất định như chưa phù hợp về giá thành, khó sử dụng, ..., đặc biệt là đối với người dùng Việt nam. Bên cạnh đó, vấn đề quản lý và điều khiển lưu lượng đảm bảo QoS theo nhu cầu trong SDIoT, một trong các thách thức lớn trong việc ứng dụng SDN trong IoT, vẫn chưa được chú trọng nghiên cứu, phát triển. Chính vì vậy, việc nghiên cứu, chế tạo mô phỏng thiết bị mạng định nghĩa bằng phần mềm cho các ứng dụng IoT có ý nghĩa quan trọng trong nghiên cứu và

làm chủ công nghệ SD-IoT hướng đến xây dựng và phát triển hạ tầng truyền thông-thông tin IoT linh hoạt và hiệu quả.

1.5 Kết luận chương

Trọng tâm của mạng tương lai là mạng định nghĩa bằng phần mềm. Nội dung chương này giới thiệu tổng quan về IoT, các công nghệ và ứng dụng. Khả năng ứng dụng SDN trong lĩnh vực vạn vật kết nối (IoT), một công nghệ đang trở thành xu hướng mới của thế giới, cũng được khảo sát cùng với các yêu cầu của hạ tầng và thiết bị truyền thông IoT. Việc nghiên cứu phát triển các thiết bị truyền thông dựa trên nền tảng SDN có khả năng đáp ứng linh hoạt và hiệu quả các yêu cầu đa dạng của IoT nhằm xây dựng hạ tầng thông tin cho các ứng dụng và dịch vụ IoT ở Việt nam hứa hẹn sẽ mang lại nhiều kết quả và tiềm năng phát triển.

CHƯƠNG 2: CÁC GIẢI PHÁP ĐẢM BẢO CHẤT LƯỢNG DỊCH VỤ TRONG IoT ĐỊNH NGHĨA BẰNG PHẦN MỀM

2.1 Giới thiệu chung

Xem xét về trạng thái hiện tại của Internet, phương pháp để cải thiện chất lượng dịch vụ (QoS) cho từng dịch vụ và khách hàng đặc biệt còn nhiều hạn chế và không linh hoạt. Nhà cung cấp dịch vụ cần có giải pháp tốt hơn, có khả năng mở rộng và cho phép tinh chỉnh lưu lượng mạng. Mạng lưới ngày nay bao gồm tập hợp các giao thức cài đặt rời rạc, xác định cách các máy chủ trong các mạng khác nhau có thể được kết nối tin cậy với nhau. Tuy nhiên, các giao thức có xu hướng được định nghĩa trong sự cô lập và được thiết kế để giải quyết vấn đề cụ thể. Điều này dẫn đến độ phức tạp của mạng và khản năng cung cấp QoS. Do thực tế này, các mạng ngày nay đều là tĩnh. Thông thường tất cả các quyết định chuyển tiếp gói tin được thực hiện tại các thiết bị riêng biệt. Điều này là trở ngại khi xét đến tính động của QoS: quản trị viên phải cấu hình riêng cho từng thiết bị của nhà cung cấp, điều chỉnh thông số (bandwith) để đáp ứng được các quy tắc và chính sách đã được xác định trước. Cách này không thể tự động thích nghi với sự thay đổi liên tục của ứng dụng và nhu cầu của người sử dụng. Do vậy, SDN là giải pháp tốt nhất cho vấn đề này và trên cơ sở đó, IoT định nghĩa bằng phần mềm cũng được hỗ trợ bởi các giải pháp QoS khả dụng cho SDN.

Kiểm soát chất lượng dịch vụ (QoS) là một cơ chế được sử dụng trong mạng để đảm bảo hiệu năng cao. Bằng cách sử dụng QoS, các quản trị viên mạng có thể quản lý tài nguyên hiệu quả hơn và cung cấp dịch vụ mức cao mà không phải cung cấp qua mức mạng. QoS có tầm quan trọng riêng trong các ứng dụng cần được đảm bảo đặc biệt. Một ứng dụng như cuộc trò chuyện thoại hoặc xem video đòi hỏi trễ nhỏ và jitter tốt như điện thoại và truyền hình truyền thống, đó là yêu cầu của người dùng. Mặt khác, truyền thông dữ liệu ít nhạy cảm với trễ và jitter, nhưng nhạy cảm

hơn với việc mất gói tin. Việc không đạt được tiêu chuẩn này có thể làm giảm chất lượng trải nghiệm (QoE).

Có hai loại chính của kĩ thuật QoS là QoS trước khi sử dụng SDN và QoS trong công nghệ SDN. Khi xem xét kĩ thuật QoS truyền thống, không sử dụng SDN, hai loại chính được chuẩn hóa. Các dịch vụ tích hợp (IntServ) là một kiến trúc điều khiển lưu lượng theo từng dòng. Nghĩa là mọi thành phần mạng (router, switch) đều phải dự trữ riêng từng tài nguyên cho mỗi luồng. Điều này hiện tại là rất khó để thực hiện trên Internet. Thứ nhất, router có tài nguyên tính toán hạn chế, ngăn cản việc phân loại các luồng ứng dụng có trong thiết bị. Thứ hai, cách tiếp cận này không thể mở rộng, vì mọi router trên một tuyến đường của dòng chảy cần phải hỗ trợ Integrated Services và lưu trữ tất cả các trạng thái và thông tin có thể có cho các luồng khác nhau. Điều này thường khó đạt được do nhà cung cấp thiết bị và giới hạn tài nguyên bộ nhớ. Do đó, phương pháp này chỉ áp dụng cho các mạng có quy mô nhỏ.

Phương pháp tiếp cận thứ hai là phân loại dịch vụ (DifServ) là một kiến trúc kiểm soát lưu lượng thô sơ, dựa vào 8 bit trường DS (thay cho trường TOS đã lỗi thời) trong tiêu đề gói tin IP. Trường này hỗ trợ tới 64 lớp lưu lượng khác nhau. Các router Diffserv quyết định dựa trên cơ sở mỗi hop để chuyển tiếp các gói dữ liệu dựa trên lớp của chúng. Mặc dù kỹ thuật này có thể áp dụng cho mạng có quy mô lớn hơn (vì chỉ có một hằng số của 64 lớp cần được phân biệt), mạng là tĩnh và thiếu khả năng tinh chỉnh QoS của các luồng riêng biệt.

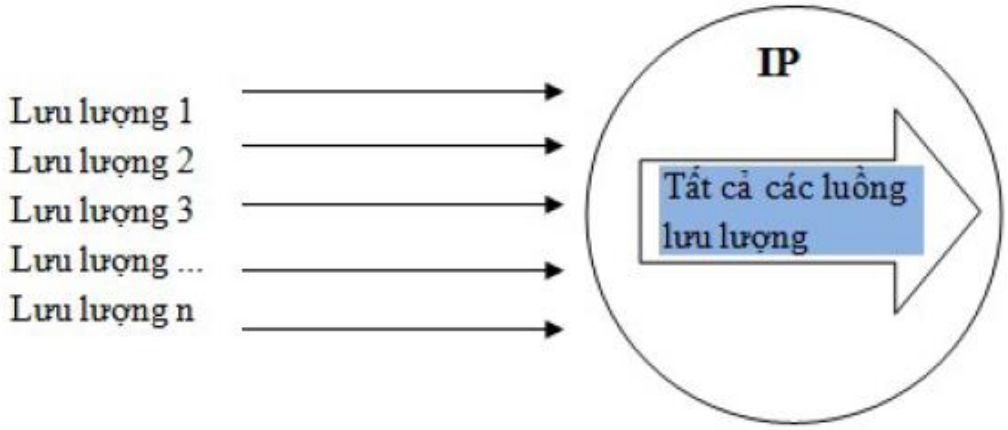
Đặc tả các chính sách và phân loại lưu lượng trong Diffserv được thực hiện tại ranh giới của các miền Diffserv (AS). Vì lý do này, không đảm bảo end-to-end qua các miền cho người sử dụng, vì các lớp DS khác nhau được đối xử khác nhau. Để giải quyết vấn đề này, RFC 2638 giới thiệu Broker Bandwidth cho mỗi miền. Để đảm bảo QoS đầu cuối, các phần trung gian cần được giao tiếp với nhau qua các miền khác để các chính sách được định nghĩa một cách chính xác. Điều này tương tự như các phương pháp tiếp cận hỗ trợ SDN, vì nó tách phần logic thành một agent tập trung. Trong thực tế một số khung đề xuất mở rộng ý tưởng của Diffserv bằng

cách sử dụng SDN. Tuy nhiên, thiếu các bộ điều khiển tập trung trong kiến trúc Internet hiện tại, Diffserv không thể được triển khai trên toàn cầu, vì không có chuẩn hóa các giao thức cấu hình lại router giữa các nhà cung cấp khác nhau. Mỗi Broker dựa trên việc tái cấu hình thiết bị cụ thể của nhà cung cấp, điều này làm hạn chế trên mạng.

2.2 QoS trong kiến trúc mạng truyền thống

2.2.1 Dịch vụ Best Effort

Dịch vụ Best Effort (nỗ lực tối đa) là cơ chế hỗ trợ cung cấp chất lượng dịch vụ đang được sử dụng trong mạng IP hiện hành. Hầu hết các ứng dụng dữ liệu đều được vận hành theo cách này. Chúng đợi dữ liệu đến và xử lý chúng càng sớm càng tốt ngay khi nhận được (Hình 2.1).



Hình 0.1 Mô hình dịch vụ Best-Effort

Lớp dịch vụ Best effort sẽ hỗ trợ thêm cho lớp lưu lượng thời gian thực. Các ứng dụng có thể lựa chọn sử dụng một trong các lớp dịch vụ đó, và khi chúng thấy không thể chấp nhận được độ trễ đó thì có thể sử dụng một trong các lớp dịch vụ khác.

Lớp dịch vụ best effort thì không có TSpec hoặc RSpec, không có đảm bảo từ mạng và không thực hiện bất kỳ điều khiển cấp phép nào. Mô hình Best-Effort sử dụng hàng đợi First In – First Out (FIFO). Các gói tin đến trước thì được quyền ra

trước. Do đó, mô hình không có khả năng dành trước băng thông cho các gói tin có quyền ưu tiên. Các mạng kiểu này không cung cấp bất kì một tính năng đặc biệt nào để khôi phục lại các gói bị hỏng hoặc bị mất.

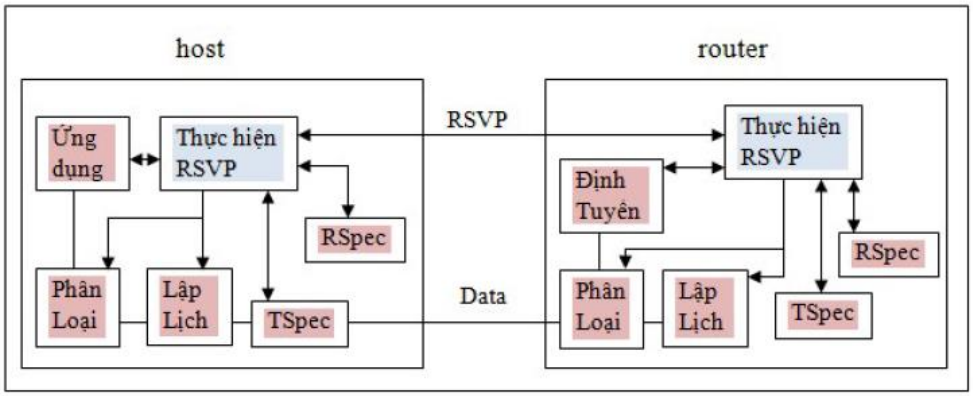
Trong chồng giao thức TCP/IP thì TCP cung cấp các dịch vụ đảm bảo trong khi IP cung cấp kiểu truyền Best effort (không có đảm bảo), nó sẽ cố gắng truyền các gói đến đích nhưng không có chức năng khôi phục lại các gói bị mất hoặc truyền sai. Bộ giao thức Internet trước đây chỉ bao gồm giao thức TCP/IP. Trong quá trình phát triển thì các nhà thiết kế đã nhận ra tầm quan trọng của thời gian truyền hơn là độ tin cậy truyền. Nói cách khác là quan tâm tới tốc độ hơn là việc khôi phục các gói. Trong kiểu truyền voice, video thời gian thực thì một số gói bị mất có thể bỏ qua được bởi việc khôi phục lại có thể tạo ra phần tiêu đề quá lớn làm giảm hiệu năng của mạng. Để cung cấp loại lưu lượng thì TCP đã được tổ chức lại trong TCP, IP, UDP. Các dịch vụ chuyển gói và đánh địa chỉ cơ bản trong lớp mạng được thực hiện bởi IP, TCP, UDP nằm ở lớp truyền tải, phía trên của IP. Tất cả đều sử dụng dịch vụ của IP, UDP là một phiên bản của TCP chấp nhận các dịch vụ Best effort của IP. Các ứng dụng có thể sử dụng UDP khi không cần các dịch vụ của TCP. Đối với các dịch vụ Best effort việc loại bỏ các gói có thể chấp nhận được do việc khôi phục được xử lí bởi các dịch vụ khác.

2.2.2 Dịch vụ tích hợp (IntServ)

Mô hình IntServ được IETF giới thiệu vào giữa thập niên 90 và được định nghĩa trong RFC 1633. Mạng đòi hỏi phải dành tuyệt đối tài nguyên (băng thông, độ trễ…) cho một số dịch vụ cụ thể. Nghĩa là, mô hình IntServ sẽ dành riêng tài nguyên mạng cho từng luồng thông tin xuyên suốt từ nguồn đến đích.

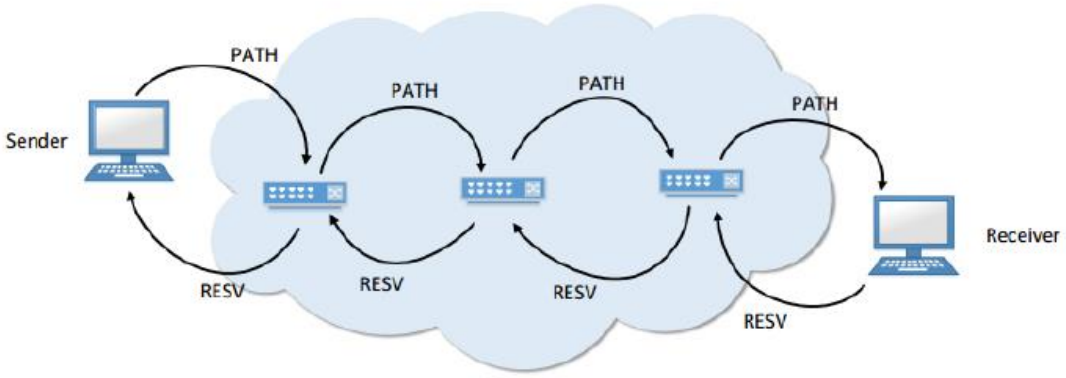
IntServ là nỗ lực đầu tiên để thiết lập kiểm soát QoS trong mạng IP. IntServ giả lập khái niệm phân bổ nguồn lực của chuyển mạch. Các phần tử mạng buộc phải phân bổ tài nguyên cho lưu lượng truy cập đặc biệt, tương tự như một cuộc gọi chuyển mạch. Có ba mức độ dịch vụ được xác định trong IntServ:

- Dịch vụ bảo đảm: dịch vụ được đảm bảo cung cấp giới hạn toán học cho phép giới hạn trên khi kết thúc đến đầu cuối, cho phép đảm bảo băng thông, trễ và mất gói tin. Nó được thực hiện thông qua một sự kết hợp của phân loại gói tin, lập lịch trình, và kiểm soát nhập học.
- Kiểm soát tải: dịch vụ Kiểm soát tải cung cấp lưu lượng với QoS xấp xỉ QoS mà nó sẽ nhận được từ dịch vụ nỗ lực tốt nhất trong một mạng không tải. Điều này đạt được thông qua điều khiển nhập liệu.
- Nỗ lực tốt nhất: lưu lượng truy cập tốt nhất không cung cấp bất kỳ đảm bảo QoS nào. Dịch vụ này tương tự như hoạt động hiện tại của mạng IP. Intserv sử dụng Giao thức Lưu trữ Tài nguyên (RSVP) để dự phòng tài nguyên. RSVP là một giao thức báo hiệu dựa trên multicast, nó là một tiêu chuẩn riêng biệt từ IntServ. Đây là một giao thức trạng thái mềm.



Hình 0.2 Mô hình nguyên lý hoạt động của InterServ

Trạng thái lưu trữ cần được làm mới theo định kỳ, nếu không, trạng thái sẽ bị mất. RSVP có hai loại bản tin để khởi tạo kết nối: *Path message* và *Resv message*. Bản tin Path là bản tin được gửi bởi bên gửi. Nó chứa thông tin về địa chỉ IP hop trước đó, đặc tả tuyến đường (bao gồm địa chỉ người gửi), băng thông và yêu cầu QoS đầu cuối. Bản tin Resv là bản tin trả lời *Path message* từ người nhận đến người gửi. Dọc theo tuyến đường, nó thiết lập lưu trữ tài nguyên cho dòng chảy dữ liệu.



Hình 0.3 Bản tin báo hiệu khởi tạo kết nối

Để hủy bỏ kết nối, RSVP sử dụng bản tin *Path Tear* và *Resv Tear*. Bản tin *Path Tear* cũng được gửi bởi phía người gửi đến bên nhận theo tuyến đường của bản tin Path, trong khi *Resv Tear* đi theo hướng ngược lại. Hai bản tin thông báo hủy kết nối xóa trạng thái đường dẫn và trạng thái dự trữ tài nguyên.

Đặc điểm của luồng lưu lượng:

• Kiểm soát đặc tả lưu lượng (TSpec): TSpec dùng để xác định đặc tính của luồng lưu lượng. Thông số quan trọng của TSpec là kích thước lớn nhất của gói tin. TSpec kiểm tra luồng lưu lượng, nếu không phù hợp thì loại bỏ luồng.

• Điều khiển đặc tả yêu cầu (RSpec): RSpec dùng để xác định các yêu cầu về QoS cho một dịch vụ mạng cụ thể. Thông số quan trọng của RSpec là tốc độ dịch vụ (băng thông mà lưu lượng cần khi đi trong mạng). RSpec kiểm tra xem tài nguyên mạng có đáp ứng được yêu cầu của ứng dụng hay không. Nếu không thể đáp ứng, mạng sẽ từ chối.

Ưu điểm chính của IntServ là nó cung cấp sự phân biệt các lớp dịch vụ. Người dùng có thể xác định loại lưu lượng truy cập của họ và sử dụng một dịch vụ phù hợp với ứng dụng của họ.

Những bất lợi chính của IntServ là vấn đề khả năng mở rộng của nó. RSVP yêu cầu phải có kết thúc báo hiệu và phải duy trì trạng thái mềm trên mỗi luồng tại

mỗi nút dọc theo con đường. Việc làm mới định kì đòi hỏi một lượng thông báo tín hiệu đáng kể và trở nên lớn hơn khi số luồng và nút trong mạng đang gia tăng. Vấn đề khác là IntServ phải lưu trữ tất cả các thông tin về dòng chảy trong tất cả các switch / routers có trên đường dẫn, dẫn đến chi phí cao trong một mạng với nhiều luồng. Do vấn đề này, nó đã không bao giờ được sử dụng trên Internet và chỉ phát triển mạnh trong các mạng local của doanh nghiệp.

a) Điều khiển đầu vào và chính sách

Điều khiển đầu vào chứa thuật toán quyết định rằng bộ định tuyến sử dụng để xác định xem có đủ tài nguyên định tuyến để chấp nhận QoS được yêu cầu cho một luồng mới. Nếu không có đủ tài nguyên định tuyến tự do, việc chấp nhận một luồng mới sẽ ảnh hưởng đến đảm bảo sớm hơn và luồng mới phải bị từ chối. Nếu luồng mới được chấp nhận, trường hợp dành trước trong bộ định tuyến được gán bộ phân loại gói

và bộ lập lịch gói cho việc dành trước được yêu cầu QoS cho luồng này.

Điều khiển đầu vào được gọi tại mỗi bộ định tuyến dọc theo đường dẫn dành trước, để tạo một bộ quyết định nội bộ là chấp nhận/từ chối tại thời điểm máy chủ yêu cầu dịch vụ thời gian thực. Thuật toán điều khiển đầu vào phải phù hợp với mô hình dịch vụ.

Điều khiển đầu vào đôi khi bị nhầm lẫn với điều khiển chính sách, đó là một gói theo gói chức năng, được xử lý bởi bộ lập lịch gói. Nó đảm bảo rằng một máy chủ không vi phạm thỏa thuận về tính chất luồng lưu lượng. Tuy nhiên, để đảm bảo rằng các điều kiện QoS được sử dụng, việc điều khiển đầu vào sẽ liên quan đến việc thực thi các chính sách bắt buộc về việc dành trước tài nguyên.

Một số chính sách sẽ được sử dụng để kiểm tra xác thực người dùng cho việc yêu cầu dành trước. Yêu cầu dành trước trái phép có thể bị từ chối. Do đó, điều khiển đầu vào có thể đóng một vai trò quan trọng trong chi phí tính toán cho các tài nguyên trên Internet trong tương lai.

b) Phân loại gói

Bộ phân loại gói xác định các gói của luồng IP trong máy chủ và bộ định tuyến, chúng sẽ nhận được một mức độ dịch vụ nhất định. Để thực hiện việc điều khiển lưu lượng hiệu quả, mỗi gói đến sẽ được ánh xạ bởi bộ phân loại thành một lớp cụ thể. Tất cả các gói được phân loại cùng một lớp giống nhau sẽ nhận được xử lý tương tự từ bộ lập lịch gói. Sự lựa chọn của một lớp học dựa trên địa chỉ IP nguồn và đích và số cổng trong trường tiêu đề gói hiện tại hoặc bổ sung một số phân loại, phải được thêm vào mỗi gói. Một lớp có thể tương ứng với một loại luồng. Ví dụ: tất cả các luồng video từ một hội nghị trực tuyến với một số người tham gia có thể thuộc về một lớp dịch vụ. Nhưng cũng có thể chỉ có một luồng thuộc về một lớp dịch vụ cụ thể.

c) Lập lịch gói

Bộ lập lịch gói quản lý việc chuyển tiếp các luồng gói khác nhau tại máy chủ và bộ định tuyến, dựa trên lớp dịch vụ của họ, sử dụng quản lý hàng đợi và thuật toán lập lịch khác nhau. Bộ lập lịch gói phải đảm bảo rằng việc phân phối gói tương ứng với tham số QoS cho mỗi luồng. Một bộ lập lịch cũng có thể giám sát hoặc định hình lưu lượng để phù hợp với một mức độ dịch vụ nhất định. Bộ lập lịch gói phải được thực hiện tại điểm mà gói đang nằm trong hàng đợi. Đây thường là cấp trình điều khiển đầu ra của một hệ điều hành và tương ứng với giao thức tầng liên kết.

d) Các lớp dịch vụ

Sơ đồ dịch vụ tích hợp (IntServ) sử dụng các lớp dịch vụ khác nhau được xác định bởi nhóm tích hợp dịch vụ IETF. Mô hình IS hiện tại bao gồm: *Dịch vụ đảm bảo (Guaranteed Service)* và *dịch vụ điều khiển tải (Controlled Load Service).*

Định nghĩa dịch vụ tích hợp (IntServ) cho các *dịch vụ điều khiển tải* nhằm cung cấp đầu cuối lưu lượng gần đúng với dịch vụ best-effort (nỗ lực tốt nhất) truyền thống với các tham số môi trường của một điều kiện mạng không tải hoặc lưu lượng nhỏ. Nói cách khác, dịch vụ tốt hơn những gì mà best-effort (nỗ lực tốt

nhất) có thể cung cấp tới một mạng thường bị tắc nghẽn. Các ứng dụng sử dụng dịch vụ này có thể cho rằng phần lớn các gói được truyền sẽ được phân phối thành công bởi mạng, tức là số lượng gói bị rơi phải tương đương với tỷ lệ lỗi của phương tiện truyền dẫn. Độ trễ được trên mạng sẽ không vượt quá độ trễ tối thiểu mà bất kỳ gói truyền thành công nào gặp phải. Các dịch vụ kiểm soát tải không sử dụng các giá trị mục tiêu cụ thể cho các tham số điều khiển đó như độ trễ hoặc mất gói. Dịch vụ điều khiển tải sẽ được sử dụng với các ứng dụng thời gian thực ví dụ. dàn âm thanh và video một chiều, không cần yêu cầu độ trễ tuyệt đối, nhưng có thể đệm dữ liệu nhận được.

Lớp *dịch vụ đảm bảo* cung cấp một khung để phân phối lưu lượng cho các ứng dụng yêu cầu việc đảm bảo về cả băng thông sẵn có và độ trễ biên từ mạng. Thuật ngữ "guaranteed bandwidth" - băng thông đảm bảo, ngụ ý rằng không có tổn thất hàng đợi do sự xảy ra bộ đệm nếu luồng nằm trong giới hạn của các tham số lưu lượng được chỉ định. "Guaranteed delay" - độ trễ đảm bảo, liên quan đến giới hạn biên của độ trễ từ đầu đến cuối mà gói tin sẽ tự học trên đường dẫn từ phía gửi đến phía nhận. *Dịch vụ đảm bảo* không cố gắng giảm thiểu sự quá tải, nó chỉ điều khiển độ trễ hàng đợi tối đa. *Dịch vụ đảm bảo* đại diện cho một điều khiển độ trễ đầu cuối cực lớn cho các mạng. Để cung cấp mức độ cao này của việc đảm bảo, dịch vụ được bảo đảm thường chỉ hữu ích nếu mọi thành phần mạng dọc theo đường dẫn hỗ trợ nó. Hơn nữa, tính di động của dịch vụ này yêu cầu thiết lập giao thức được sử dụng để yêu cầu dịch vụ và cập nhật chúng lên các bộ định tuyến trung gian.

Vấn đề chính trong mô hình dịch vụ tích hợp được liên kết với trạng thái luồng và phương pháp xử lý luồng. Nếu xử lý trên mỗi luồng tăng đáng kể, nó có thể trở thành một mối quan tâm về khả năng mở rộng cho các mạng lớn; trong thực tế các yêu cầu tài nguyên (xử lý tính toán và tiêu thụ bộ nhớ) để chạy việc để dành tài nguyên trên mỗi luồng trên bộ định tuyến tăng tỷ lệ trực tiếp với số lượng để dành riêng biệt cần phải có hợp lý. Do đó, cách tiếp cận này đại diện cho một giải pháp tốt chỉ ở các biên của mạng.

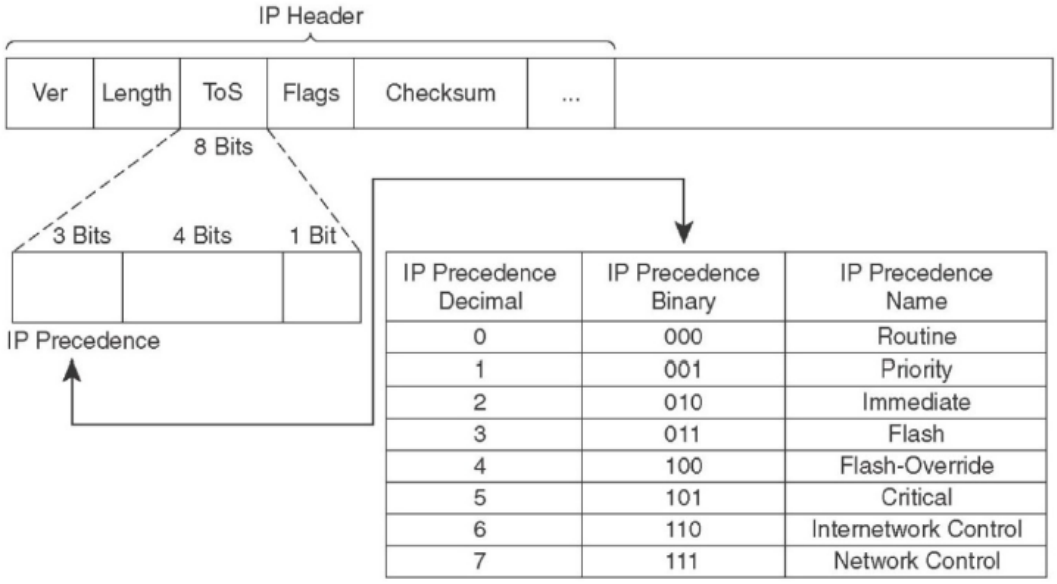
2.2.3 Dịch vụ phân biệt (Diffserv)

Việc thực hiện kiến trúc như trên gặp nhiều khó khăn do chỉ áp dụng được cho những mạng có số các luồng dữ liệu là nhỏ. Mục đích của việc đưa ra dịch vụ Diffserv để nhằm đạt được tính linh động. Diffserv trái ngược với Intserv là dựa trên từng luồng dữ liệu, nó phân loại các gói thành một số lượng không lớn các tập (gọi là các lớp) và do đó đạt được hiệu quả cho các mạng lớn. Các chức năng đơn giản được thực hiện tại router lõi, trong khi các chức năng phức tạp được triển khai tại các router biên. Tính linh động rất là cần thiết vì dịch vụ mới có thể xuất hiện và một số dịch vụ trở nên lỗi thời. Do đó Diffserv không cần thiết phải xác định dịch vụ như là Inserv, thay vào đó, nó cung cấp các thành phần chức năng mà trên đó dịch vụ có thể được xây dựng. Một gói đi vào mạng mà không đề cập gì đến dịch vụ và mạng sẽ xác định luồng và cung cấp dịch vụ thích hợp. Việc thông tin giữa người dùng và dịch vụ sẽ nằm trong bản thỏa thuận dịch vụ (SLA) và giàn xếp giữa một luồng xác định trước với bản thỏa thuận về lưu lượng. Việc xác định SLA sẽ được cung cấp bao nhiêu tài nguyên sẽ được cấu hình tay.

Kiến trúc Diffserv bao gồm hai tập các thành phần chức năng:

- Tại biên của mạng, việc phân loại và điều khiển lưu lượng được thực hiện và các gói được phân vào các lớp.
- Tại lõi, một cơ chế phân loại đơn giản được thực hiện. Cơ chế hàng đợi dựa trên lớp được áp dụng.

Tại cạnh biên đi vào của mạng, các gói đi vào được đánh dấu thông qua một quá trình phân loại phức tạp (thông qua một số luật được định nghĩa trước trong TCA). Việc đánh dấu bao gồm thiết lập một vài bit còn được gọi là bit của "trường phân biệt dịch vụ" (trường DS) của tiêu đề gói (của trường *Type-Of-Service* trong IPv4). Đánh dấu của một gói cũng được gọi là PHB (cách ứng xử tại mỗi nút). Trường DS được dùng thay cho các các định nghĩa trước như ToS (*Loại dịch vụ* trong IPv4) và trường *Lớp lưu lượng* của IPv6. Trường DS là 6 bít cao nhất của những trường trên và bao gồm thành phần đánh dấu hay (mã dịch vụ DS DSCP).



IP Precedence Decimal	IP Precedence Binary	IP Precedence Name
0	000	Routine
1	001	Priority
2	010	Immediate
3	011	Flash
4	100	Flash-Override
5	101	Critical
6	110	Internetwork Control
7	111	Network Control

Hình 0.4 Tiêu đề gói IP

DiffServ cũng hoạt động trên luồng tổng hợp, hơn là xử lý các luồng riêng biệt. Không giống như IntServ, trong đó các luồng được xử lý từ đầu đến cuối, DiffServ áp dụng chính sách của nó khi đi qua một bước nhảy, được gọi là PHB. Trường DSCP sẽ mã hóa PHB và được đưa vào gói. Sau khi được phân loại và đánh dấu, thành phần quy định lưu lượng được áp dụng, cũng tại biên của mạng. Chức năng đo cũng được thực hiện để đo tính chất thời gian của luồng được phân loại (hoặc thường là một tập) và thông qua so sánh với profile lưu lượng được thỏa thuận, xác định xem gói đó có thuộc profile hay không. Nếu như:

- Gói đó nằm ngoài profile, gói đó sẽ được đặt trong hàng đợi cho đến khi nó nằm trong profile và sau đó được chuyển tiếp, hoặc chúng có thể bị hủy hay được đánh dấu lại…
- Gói nằm trong profile được chuyển tiếp mà không thay đổi nhưng trong một vài trường hợp, chúng có thể được đánh dấu lại (nếu chúng đi vào một vùng Diffserv sử dụng ánh xạ PHB và DSCP khác).

Thiết bị đo lấy thông tin từ các thiết bị phân loại và hoạt động dựa gần đúng trên các thiết bị đánh dấu, hiệu chỉnh và hủy. Việc đánh dấu gói trong hay ngoài profile, quyết định đánh dấu, loại bỏ, đánh dấu lại… không được quy định trong

Diffserv, Diffserv chỉ cung cấp sườn và kiến trúc để linh động cung cấp tập các dịch vụ cho người dùng đầu cuối.

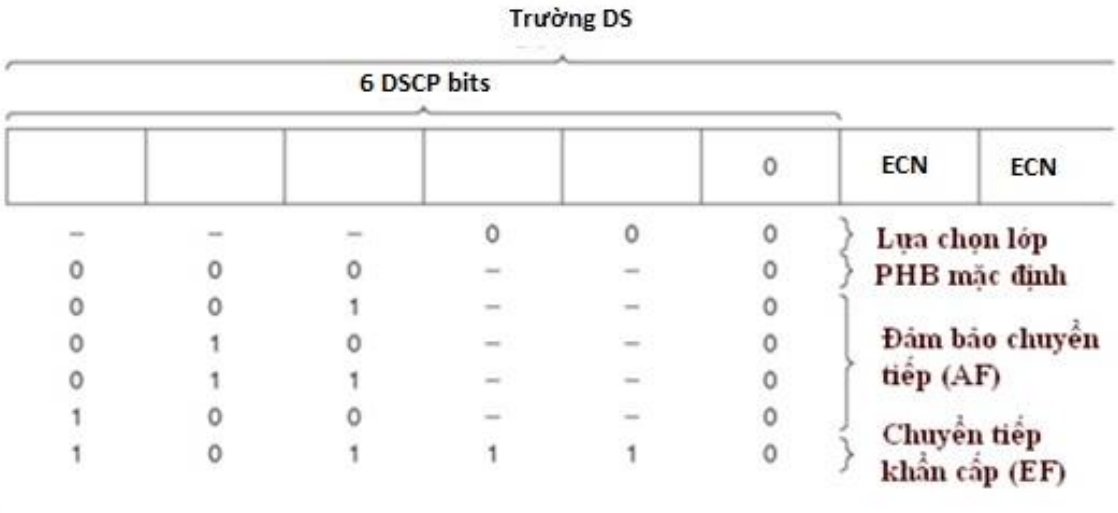
Chức năng chính của thành phần lõi là áp dụng cơ chế hàng đợi để cung cấp PHB thỏa thuận cho một lớp. Router thực hiện cơ chế phân loại đơn giản thông qua trường DSCP, nó sau đó được ánh xạ qua PHB. Ghi nhớ rằng PHB được áp dụng dựa trên đánh dấu và không quan tâm đến cặp nguồn/đích do đó có thể gọi Diffserv là không trạng thái. PHB không liên quan tới bất kỳ cơ chế nào về bộ đệm, hay chính sách áp dụng cho một liên kết mà nó định nghĩa ứng xử hoặc dịch vụ trên mỗi nút. Diffserv không định nghĩa dịch vụ, chỉ có PHBs. Dịch vụ cung cấp đầu cuối cho một luồng (hoặc một tập) là kết quả của việc áp dụng điều khiển lưu lượng tại router biên và PHBs qua các lớp trong lõi.

Các gói được phân loại dựa trên các bit điểm mã dịch vụ khác biệt (DCRP), được đặt trong tiêu đề IP. Các gói với các bit DSCP giống nhau nhận được cách xử lí như nhau trong bộ định tuyến / bộ chuyển mạch, cho dù chúng thuộc về luồng nào.

Có một số PHB được định nghĩa cho DiffServ, phổ biến nhất là:

- PHB mặc định: các gói với bit DSCP 000000 và các gói không đáp ứng các yêu cầu của các lớp khác được chuyển tiếp với một đặc tính nỗ lực tốt nhất.
- Chuyển tiếp khẩn cấp (EF): EF PHB có đặc điểm chậm trễ, tổn thất thấp và Jitter thấp. Các gói tin EF được ưu tiên xếp hàng trên các lớp lưu lượng truy cập khác. Flow sử dụng EF vẫn sẽ cạnh tranh với nhau.
- Đảm bảo chuyển tiếp (AF): AF PHB tương tự như kiểm soát tải của IntServ. Nó cung cấp sự đảm bảo chuyển tiếp miễn là lưu lượng truy cập không vượt quá một tỷ lệ cụ thể.
- Lựa chọn lớp (CS): PHB này được định nghĩa để bảo vệ tính tương thích ngược với lược đồ ưu tiên IP diễn ra trước DiffServ.

a) Trường DS



Hình 0.5 Trường DS và DSCP PHB

Mạng dịch vụ phân biệt tuy có ưu điểm hơn mạng dịch vụ tích hợp là không cần sử dụng giao thức báo hiệu và cơ chế sử dụng đơn giản, linh động, nhưng nó không cung cấp dịch vụ tốt nhất như trong mạng dịch vụ tích hợp (hay mạng lưu trạng thái).

b) Xử lý Per-hop

Khối xây dựng chính khác của DiffServ là xử lý per-hop (PHB). Một xử lý per-hop được định nghĩa là "mô tả của xử lý chuyển tiếp có thể quan sát được bên ngoài của nút DS được áp dụng đến một tập xử lý DS cụ thể". Điều quan trọng cần lưu ý là PHB không phải là một dịch vụ, nhưng dịch vụ hợp lý có thể được xây dựng bằng cách triển khai các PHB tương tự trong mỗi nút dọc theo đường truyền lưu lượng. Một tính năng quan trọng khác của PHB là xử lý chuyển tiếp được áp dụng cho tập lưu lượng truy cập, không phải luồng riêng micro. Thuật ngữ 'micro flow" đề cập đến một phân biệt luồng các gói tin liên quan từ một người dùng. Là một thuật ngữ kỹ thuật PHB là sự kết hợp của chuyển tiếp, phân loại, lập lịch và thả các xử lý tại mỗi hop. Hơn nữa, PHB là một thuật ngữ nhà cung cấp dịch vụ, quản trị viên và nhà cung cấp có thể sử dụng để thảo luận thực tế. PHB có thể được chỉ định theo mức độ ưu tiên tài nguyên của họ (ví dụ: băng thông) so với các PHB khác.

Như đã nói trước đây, trường DS sử dụng sáu bit để xác định codepoint của dịch vụ khác biệt (DSCP). Mỗi nút vào mạng để chọn PHB sẽ sử dụng các codepoint này. Mỗi thiết bị mạng DS phải có thông tin về cách các gói với các trường DS khác nhau đã được xử lý. Trong thông số kỹ thuật DS, thông tin này được gọi là xử lý per-hop (PHB). Hơn nữa, PHB có thể được mô tả như một tập hợp các tham số bên trong bộ định tuyến có thể được sử dụng để điều khiển cách các gói được lên lịch trên một giao diện đầu ra. Đây có thể là một số riêng biệt hàng đợi với mức độ ưu tiên có thể được đặt, tham số cho độ dài hàng đợi hoặc các thuật toán thả và thả ưu tiên trọng số đối với các gói. Hiện tại, có ba bộ PHB được chỉ định bởi nhóm DiffServ IETF là: *Bộ chọn lớp* (CS) PHB , *Bộ chuyển tiếp nhanh* (EF) PHB và *Bộ chuyển tiếp đảm bảo* (AF).

Bộ chọn lớp PHB cố gắng duy trì ý nghĩa cho trường ưu tiên IP của TOS (3 bit đầu tiên trong octet IP kiểu cũ). Các giá trị DSCP ‘xxx000’ được dành riêng cho *bộ chọn lớp PHB*. Tám codepoint CS này phải mang lại ít nhất hai chuyển tiếp độc lập các lớp lưu lượng. Một codepoint CS có giá trị số lớn sẽ nhận được thứ tự tương đối cao hơn hơn một gói được đánh dấu với giá trị số thấp hơn. CS PHB cố gắng tránh xung đột giữa các định nghĩa cũ và mới của trường DS. Hơn nữa, nó có thể được sử dụng để xây dựng một kiến trúc đơn giản nguyên mẫu DiffServ do các bộ định tuyến hiện có hỗ trợ việc sử dụng quyền ưu tiên trường IP.

Bộ chọn lớp PHB có thể được thực hiện theo nhiều quy tắc hàng đợi, ví dụ: WFQ, CBQ hoặc hàng đợi ưu tiên. Định nghĩa của CS PHB không chỉ định bất kỳ lưu lượng truy cập nào - chức năng điều kiện. Do đó, không hợp lý khi xây dựng các dịch vụ chỉ sử dụng nhóm CS PHB. Thay vào đó, nhà cung cấp dịch vụ có thể tránh việc nâng cấp tất cả các nút và vẫn cung cấp một số cấp độ riêng biệt bằng cách sử dụng các bộ định tuyến hiện có với CS là nút bên trong và chỉ nâng cấp các nút biên. Bản chất của loại dịch vụ này là tương đối và do đó phù hợp với các ứng dụng như duyệt Web.

Mục tiêu của *Bộ chuyển tiếp nhanh PHB* là hỗ trợ tập hợp xử lý ở mức thấp như mất gói, độ trễ , độ tải , đảm bảo dịch vụ đầu cuối băng thông thông qua mạng

DiffServ. Dịch vụ này cố gắng giống với dịch vụ thuê đường truyền ảo trên mạng IP. Từ việc mất gói, độ trễ và tải đều do hàng đợi mà lưu lượng truy cập gặp phải trong khi lướt mạng, cách để cung cấp dịch vụ tải và độ trễ thấp là để đảm bảo rằng tập hợp lưu lượng truy cập không sẽ không còn hàng đợi nào hoặc chỉ có những hàng rất nhỏ. Các hàng đợi được hình thành khi tốc độ lưu lượng truy cập vượt quá tốc độ lưu lượng ban đầu. Để đảm bảo rằng không có hàng đợi sẽ xảy ra đối với một số tập hợp lưu lượng truy cập, tốc độ truy cập tối đa của tập hợp hợp phải nhỏ hơn tổng các cấu hình tốc độ ban đầu. Do đó, EF PHB được định nghĩa là xử lý chuyển tiếp lưu lượng tập hợp hợp trong đó tốc độ dữ liệu gói được cấu hình phải bằng hoặc vượt trên tốc độ dữ liệu ban đầu. Lưu lượng EF sẽ nhận được tốc độ được định cấu hình độc lập với bất kỳ loại nào khác lưu lượng cố gắng đi qua nút. Điều này ngụ ý rằng, tốc độ bit điều khiển nghiêm ngặt trong các nút biên (tức là lưu lượng truy cập vượt quá tốc độ theo thỏa thuận phải được loại bỏ) và như chuyển tiếp nhanh trong các nút bên trong càng tốt.

EF có thể được thực hiện bởi một số cơ chế lập lịch hàng đợi khác nhau. Hàng đợi ưu tiên đơn giản sẽ hoạt động phù hợp miễn là mức hàng ưu tiên hàng đợi cao, tức là EF, không bỏ đi hàng đợi ưu tiên thấp hơn. Điều này có thể được thực hiện bằng cách sử dụng một số loại tốc độ giám sát (ví dụ: token bocket) trong mỗi hàng đợi trước đó xác định số lượng hàng đợi có thể bỏ lại các lưu lượng các. EF PHB có thể cung cấp dịch vụ thuê đường truyền qua Internet công cộng, do đó người dùng cảm thấy rằng họ có một ống tốc độ bit cố định giữa các mạng nội bộ của họ. Vì EF cung cấp thấp dịch vụ độ trễ này phù hợp với một số ứng dụng thời gian thực, như điện thoại IP và họp trực tuyến.

Bộ chuyển tiếp đảm bảo PHB PHB là một tập hợp các PHB đảm bảo dịch vụ chuyển tiếp phù hợp với thông tin dịch vụ nhất định hoặc thỏa thuận điều kiện lưu lượng (TCA). Nhóm chuyển tiếp đảm bảo (AF) cung cấp các mức đảm bảo chuyển tiếp khác nhau. AF hiện tại đặc tả kỹ thuật cung cấp phân phối các gói IP trong bốn lớp, mỗi lớp có ba cấp ưu tiên, xem bảng 2.1. Các gói trong một lớp phải được chuyển tiếp độc lập với các gói trong các lớp AF khác và nút DiffServ phải phân bổ

tài nguyên cho từng lớp AF được triển khai, nhưng tất cả các lớp không yêu cầu phải được thực hiện. Mức độ đảm bảo chuyển tiếp của một gói IP phụ thuộc vào lưu lượng băng thông và kích thước bộ đệm đã được định cấu hình cho lớp AF, tải hiện tại của lớp AF và mức độ ưu tiên thả của gói. Một điều quan trọng khác của tính năng của nhóm AF PHB là các gói IP của cùng một microflow không nên được yêu cầu lại lại nếu chúng thuộc cùng một lớp AF. Giống như các PHB khác, AF không phải là mô hình dịch vụ đầu cuối, nhưng là một khối xây dựng để xây dựng dịch vụ. Vì QoS phụ thuộc vào lưu lượng khác đi vào nút, dịch vụ có khả năng chỉ mang tính tương đối. Do đó, nhóm AF PHB không thể đảm bảo bất kỳ đặc điểm định lượng nào. Mặt khác, bằng cách cung cấp quá mức một lớp AF thậm chí các dịch vụ với mất gói và độ trễ thấp có thể được thực hiện.

Bảng 0.1 Các lớp AF

Thả ưu tiên	Lớp AF 1	Lớp AF 2	Lớp AF 3	Lớp AF 4
Thấp	001010	010010	011010	100010
Trung bình	001100	010100	011100	100100
Cao	001110	010110	011110	100110

Định nghĩa của nhóm AF PHB không chỉ định bất kỳ hàng đợi bắt buộc nào cho việc thực hiện. Tuy nhiên, cơ chế hàng đợi CBQ phân bổ băng thông cho các hàng đợi khác nhau và do đó chính xác là những gì AF yêu cầu. Sự ưu tiên thả trong lớp cần quản lý hàng đợi một cách tích cực. Thuật toán Random Early Drop (RED) với nhiều ngưỡng các cấp có thể xử lý được điều đó,. Việc triển khai nhóm AF PHB sẽ như vậy bao gồm một bộ lập lịch CBQ phân chia băng thông cho các lớp PHB và thuật toán RED trong đó thả các gói bên trong hàng đợi theo các ưu tiên thả. Ngoài việc quản lý hàng đợi, đánh dấu lưu lượng và đo lưu lượng là cần thiết để triển khai nhóm AF PHB. Các các nút biên đo luồng gói đến và lưu lượng vượt quá thỏa thuận trong SLA được đánh dấu cho mức độ quan trọng thấp hơn. Không giống như EF

PHB, lưu lượng vượt quá hoặc bùng nổ không phải là ngay lập tức giảm được nhưng thay vào đó được phân phối với xác suất chuyển tiếp nhỏ hơn.

c) Các miền DiffServ

Việc thiết lập các đảm bảo QoS không được thực hiện cho các kết nối đầu cuối cụ thể mà được xác định rõ tại lĩnh vực các miền dịch vụ riêng biệt. Nhóm IETF định nghĩa một miền dịch vụ phân biệt như một phần của Internet qua đó nó là một tập hợp nhất quán các chính sách dịch vụ phân biệt được quản lý theo cách thức ngang hàng. Nó có thể đại diện các miền quản trị riêng biệt hoặc hệ thống tự động và các công nghệ mạng khác nhau.

Một miền DS bao gồm các thành phần biên được sử dụng để kết nối các miền DS khác nhau với nhau và các thành phần bên trong chỉ được sử dụng bên trong các miền. Việc điều hòa lưu lượng được thực hiện bên trong một nút biên bởi bộ *điều hòa lưu lượng*. Nó phân loại, đánh dấu, và có thể tạo điều kiện các gói vào hoặc ra khỏi miền DS. Khái niệm gần đây nhất với mô hình của bộ định tuyến DiffServ bao gồm khóa sau các yếu tố dưới đây.

➢ *Các phần tử phân loại lưu lượng*

Phân loại được thực hiện bởi một phần tử phân loại. Một bộ phân loại chọn các gói tin dựa trên tiêu đề gói của chúng và chuyển tiếp các gói phù hợp theo quy tắc phân loại để xử lý thêm. Mô hình DS chỉ định đáng kể hai loại gói phân loại:

- Trình phân loại đa trường (MF), có thể phân loại trên trường DS cũng như trên bất kỳ trường tiêu đề IP nào khác, ví dụ, địa chỉ IP và số port, giống như trình phân loại RSVP. Một loại phân loại MF phổ biến gồm 6 bộ phân loại dựa trên sáu trường từ tiêu để của IP, TCP hoặc UDP (địa chỉ đích, địa chỉ nguồn, giao thức IP, port nguồn, port đích và DSCP).
- Trình phân loại tập hợp xử lý (BA), chỉ sử dụng mã hóa DiffServ (DSCP) trong một gói tiêu đề IP để xác định luồng đầu ra logic mà gói sẽ được truy cập đến.

Chỉ gần đây các loại phân loại khác được xác định: ví dụ như trình phân loại dạng tự do, trong đó sử dụng một tập hợp các bộ lọc tùy ý có thể xác định người dùng có kết hợp thành các nhóm bộ lọc để tạo thành bộ lọc mạnh mẽ hơn.

➢ *Đồng hồ đo*

Đồng hộ đo lưu lượng thực hiện việc chuyển tiếp các gói được chọn bởi trình phân loại tương ứng với thông tin lưu lượng mô tả QoS cho SLA giữa khách hàng và nhà cung cấp dịch vụ. Một đồng hồ truyền thông tin trạng thái đến các chức năng điều hòa để kích hoạt một hành động cụ thể cho từng gói, tuân thủ hoặc không tuân thủ với các yêu cầu QoS được yêu cầu.

Trong ví dụ DiffServ gần đây nhất, ba mức độ phù hợp được đề cập về mặt màu sắc, với màu xanh lá cây đại diện cho sự thích hợp, màu vàng đại diện cho sự thích hợp một phần và màu đỏ đại diện cho sự không thích hợp. Những mức độ phù hợp khác nhau có thể được sử dụng để kích hoạt hàng đợi khác nhau, đánh dấu hoặc thả các xử lý.

Một số ví dụ về máy đo có thể là như sau: Máy đo tốc độ trung bình, máy đo trung bình di chuyển có trọng số (EWMA), máy đo mã thông báo hai tham số…

➢ *Các phần tử thực hiện*

Các bộ phân loại và các máy đo đã được mô tả trước đó một cách tổng quát là để xác định các hành thực hiện tương ứng để áp dụng đối với gói tin. Tập các hành động thực hiện có thể áp dụng bao gồm: Đánh dấu, Loại bỏ (*drop*) tuyệt đối, Ghép kênh, Đếm, Không thực hiện.

- Bộ đánh dấu DSCP: Các điểm đánh dấu DSCP đặt trường DSCP của các gói tin IP đến thành một mẫu bit cụ thể. PHB được đặt trong 6 bit đầu tiên của trường DSCP để các gói được đánh dấu được chuyển tiếp bên trong miền DS theo SLA giữa nhà cung cấp dịch vụ và khách hàng.
- Bộ loại bỏ tuyệt đối: Các bộ loại bỏ tuyệt đối đơn giản chỉ cần loại bỏ các gói. Điều đáng nói là các bộ này không phải là các phần tử duy nhất

loại bỏ các gói, một phần tử khác là một giải thuật toán thuật toán thả trong các bộ hàng đợi chẳng hạn.

- Bộ ghép kênh: Đôi khi, cần phải ghép các luồng lưu lượng truy cập thành một phần tử với một đầu vào duy nhất. Bộ ghép kênh là một thiết bị logic đơn giản để hợp nhất các luồng lưu lượng.
- Bộ đếm: Bộ đếm chỉ đơn giản là giám sát các gói dữ liệu được xử lý bởi bộ định tuyến. Kết quả số liệu thống kê có thể được sử dụng sau này cho mục đích thanh toán của khách hàng, xác minh dịch vụ hoặc các kỹ thuật mạng.
- Bộ không thực hiện: Một phần tử này không thực hiện bất kỳ hành động nào trên gói. Một phần tử như vậy rất hữu ích để xác định trong trường hợp cấu hình hoặc quản lý không có tính linh hoạt để bỏ qua một phần tử hành động trong phân đoạn dữ liệu.

➢ *Các phần tử hàng đợi*

Các quy tắc hàng đợi được triển khai trong các phần tử hàng đợi điều chỉnh truyền các gói thuộc các luồng lưu lượng khác nhau và xác định thứ tự của chúng, có thể lưu trữ hoặc loại bỏ chúng. Thông thường các gói được lưu trữ vì có sẵn tài nguyên không đủ để chuyển tiếp chúng ngay lập tức hoặc bởi vì các yếu tố này được sử dụng để thay đổi các thuộc tính tạm thời của một luồng lưu lượng.

Trong sơ đồ này này, các gói bị loại bỏ do giới hạn bộ đệm, vì bộ đệm vượt quá ngưỡng, để đáp ứng với phản ứng của tín hiệu giao thức điều khiển hoặc bởi vì đồng hồ đo vượt quá mức ban đầu. Cơ chế khác nhau của quy tắc hàng đợi có mặt trong tài liệu, và sẽ được thảo luận sau trong tài liệu này.

2.3 QoS trong IoT định nghĩa bằng phần mềm

Trái ngược với các phương pháp đảm bảo QoS truyền thống, mạng IoT định nghĩa bằng phần mềm sử dụng các phương pháp tiếp cận dựa trên công nghệ SDN giải quyết tất cả các vấn đề được mô tả ở trên. Thông qua mức độ trừu tượng cao hơn được cung cấp bởi bộ điều khiển riêng biệt, người ta có thể chỉ định các chính

sách mà không cần phải cấu hình lại các thiết lập mức thấp ở mỗi thiết bị chuyển tiếp. Thiết lập chính sách và các lớp lưu lượng khác nhau không hạn chế, cho phép điều chỉnh tỉ mỉ dựa trên nhu cầu của người dùng (đối chiếu với tối đa 64 lớp được xác định trước trong trường DS). Do đó, các quy tắc có thể được định nghĩa cho mỗi luồng (nếu cần) và bộ điều khiển có nhiệm vụ áp dụng chúng cho các phần tử mạng khác nhau. Về vấn đề này, có hai khía cạnh chính cần xem xét:

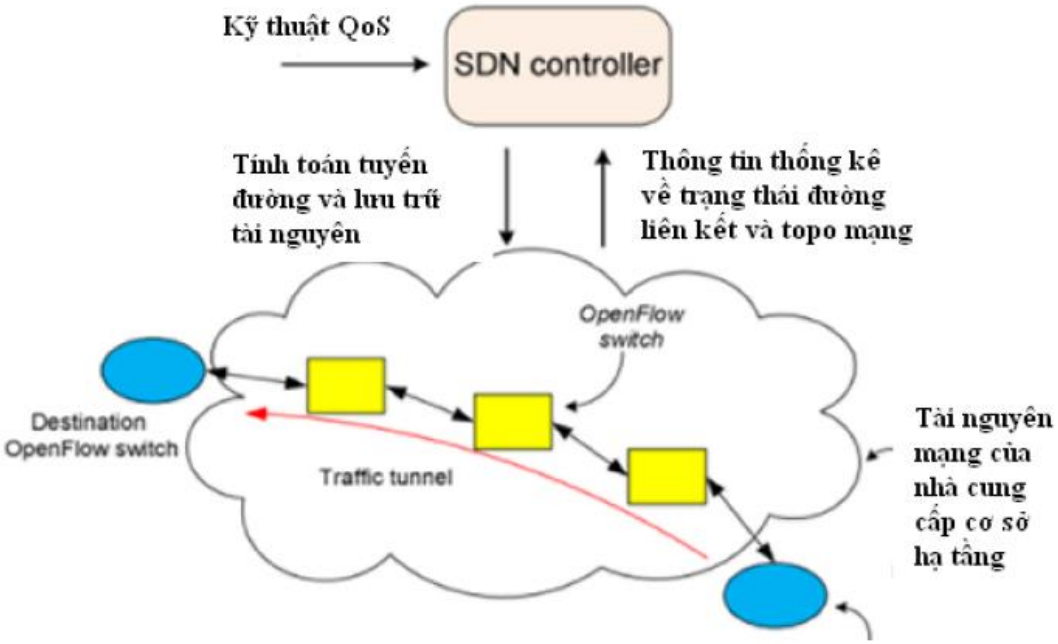
- Cung cấp QoS cho người dùng cụ thể hoặc cho luồng khách hàng doanh nghiệp.
- Cung cấp QoS cho dòng ứng dụng cụ thể.

Một số giải pháp khác nhau cho những vấn đề tồn tại. Phương pháp phổ biến nhất sử dụng framework là một hình thức cắt lát ảo của băng thông sẵn có. Cách này có thể tiết kiệm tài nguyên, trong đó mỗi luồng được gán một phần của dung lượng truyền tải tổng thể. Ngược lại việc định tuyến động theo luồng đã được đề xuất như một giải pháp thay thế khả thi mà không trực tiếp phân bổ bất kỳ tài nguyên nào cho luồng. Hơn nữa các phương pháp tiếp cận tập trung cụ thể vào gói tin hàng đợi và framework cho chính sách cũng được thảo luận. Nhiều framework hỗ trợ SDN cũng yêu cầu một khuôn lấy mẫu để liên tục giám sát mạng. Nếu tất cả các node được giám sát, thông tin thu thập có thể được ngoại suy cho toàn bộ mạng, do đó tạo ra một cái nhìn tổng quan về trạng thái mạng. Cách tiếp cận này tạo ra chi phí cao và cần được áp dụng theo cách có kiểm soát.

2.3.1 Framework lưu trữ tài nguyên

Đây là giải pháp phổ biến nhất cung cấp QoS, do đó số lượng lớn các framework đều thuộc thể loại này. Thông thường, các framework cho loại này bao gồm hai mô-đun chính: một bộ phân loại lưu lượng luồng và SDN dựa trên rate shapper. Các phân lớp đọc các trường của gói tin và cố gắng gán một mức độ ưu tiên nhất định cho các luồng khác nhau dựa trên các chính sách được định nghĩa trong bộ điều khiển.

Mặt khác, các rate shaper cài đặt các quy tắc dự phòng tài nguyên trong một Openflow kích hoạt chuyển đổi dựa trên sự phân lớp này. Hiện tại, việc triển khai OpenFlow cho phạm vi mặt phẳng dữ liệu (giống như OpenvSwitch) không hỗ trợ rõ ràng cho mỗi tốc độ luồng.



Hình 0.6 Mô hình mạng đường trục của SDN/ Openflow

Một ví dụ cho framework như vậy là FlowQoS. Nguyên mẫu này sử dụng cho các mạng quy mô nhỏ và sử dụng SDN để cấu hình lại các bộ định tuyến gia đình theo một loạt các chính sách rate shaping được định nghĩa bởi người sử dụng. Việc phân loại các dòng chảy khác nhau được thực hiện bởi hai mô-đun riêng biệt. Mô đun đầu tiên thực hiện phân loại lưu lượng web (HTTP và HTTPS), được xác định bởi số cổng tương ứng (80 và 443). Sau đó phân loại thêm lưu lượng truy cập web này dựa trên phản hồi DNS, so khớp bản ghi A và CNAME với một danh sách các biểu thức thông thường được xác định trước. Mô-đun thứ hai xử lý phân loại các luồng khác (không phải là HTTP/HTTPS). Chúng được phân loại dựa trên flow-tuple, bốn byte đầu tiên được gửi, nhận và các kích thước tải gửi và nhận. Sau khi các luồng đã được phân loại thành công, thì việc định dạng tốc độ được thực hiện. Để khắc phục tình trạng thiếu định dạng Openflow cho mỗi luồng định dạng tỉ lệ,

framework giới thiệu hai chuyển mạch ảo bên trong bộ định tuyến trong nhà. Các kết nối chuyển mạch bên trong khác nhau giữa hai thiết bị này được cấu hình thông qua tiện ích tc của Linux và được gán các tốc độ khác nhau bởi bộ điều khiển (được xác định trước bởi người dùng). Đây là hình thức đơn giản của lát cắt mạng (slicing). Đây là một cách tiếp cận còn hạn chế vì nó chỉ cho thấy QoS có thể được áp dụng cho bộ định tuyến gia đình. FlowQoS là một phần nhỏ của định nghĩa Framework chỉ ra cách SDN có thể được sử dụng cho phép QoS, tại cùng một thời điểm đơn giản hóa cấu hình QoS cho người dùng đơn lẻ. Nó có thể thay thế sự cần thiết của việc sử dụng bằng tay các tiện ích tc của Linux hay các thẻ iptable, điều này có vẻ rất phức tạp.

Đối với giải pháp quy mô lớn hơn, một framework tổng quát hơn là một phần của dự án EuQoS. Trọng tâm của giải pháp này cho phép QoS cho khách hàng doanh nghiệp và ưu tiên cho khách hàng này theo yêu cầu của họ. Việc phân loại trong cách tiếp cận này dựa trên giá trị của trường DS kết hợp với IP đích của một gói tin vì chỉ cần phân biệt các luồng của của các khách hàng khác nhau. Điều này rất giống với cách tiếp cận chuẩn được áp dụng trong DiffServ, nhưng khác biệt là phân biệt khách hàng dựa vào địa chỉ IP nguồn của họ.

Tất cả các bộ định tuyến được tái cấu hình bởi bộ điều khiển. Đối với mỗi mục định tuyến trong một router hai mục Openflow flow table được tạo ra, khớp với địa chỉ IP đích trong mục định tuyến. Một trong số chúng phù hợp với các gói tin có TOS bị vô hiệu hóa (mức ưu tiên thấp), một số khác phù hợp với gói tin có TOS được kích hoạt (ưu tiên cao). Bằng cách này thì tuyến đường có besteffort và high-priority của khách hàng doanh nghiệp được phân biệt. Vì vậy, số lượng các mục flow gấp hai lần số lượng các mục định tuyến. Điều này là chấp nhận được.

Đối với định dạng tốc độ, hai hàng đợi với tốc độ truyền được cấu hình ở mỗi router. Điều này làm cho mạng phù hợp với sự khác biệt của các luồng đến từ các máy chủ khác nhau. Tuy nhiên, nó không bao gồm luồng của các ứng dụng khác nhau. Ngoài việc lưu trữ tài nguyên, EuQoS tập trung vào hai khía cạnh khác:

- Inter-AS QoS – cho mỗi hệ tự trị, một bộ điều khiển SDN đang hoạt động, nó tạo ra một mục flow ở router biên thiết lập trường khác. Vì việc sửa đổi các trường trong gói tin chỉ diễn ra ở biên của AS và các quyết định chuyển tiếp được dựa trên giá trị TOS, việc chuyển tiếp DiffServ cũng có thể diễn ra bên trong AS. Đó là lí do tại sao chuẩn BGP hoặc OSPF có thể được sử dụng tại đây. Điều này cung cấp một con đường động để ưu tiên các luồng nhất định đi qua nhiều AS.
- Failure recovery – mỗi lần kết nối không thành công, framework tự động đảm bảo tuyến đường có ưu tiên cao không bị ảnh hưởng và được khôi phục TOS để cho phép luồng có ưu tiên cao đến từ một AS đến tốc độ truyền tải được xác định trước. Điều này cũng có nghĩa là nếu không đủ băng thông mạng (vì kết nối bị lỗi), một số gói tin best-effort sẽ không được chuyển tiếp ngay lập tức.

2.3.2 Framework định tuyến theo luồng

Khái niệm này phân biệt các luồng đa phương tiện với các luồng dữ liệu thông thường thông qua một trình phân loại, tương tự như kĩ thuật dự phòng tài nguyên. Tuy nhiên, thay vì dự trữ tài nguyên (băng thông) tại mỗi thiết bị chuyển tiếp, bộ điều khiển tự động đặt các luồng ưu tiên cao trên các tuyến đảm bảo QoS. Đồng thời luồng dữ liệu thường xuyên vẫn còn trên các tuyến đường thông thường của chúng. Trong trường hợp này, định tuyến QoS được coi là một vấn đề hạn chế đường dẫn ngắn nhất (CSP). Để đảm bảo đường đi là tối ưu, vấn đề tắc nghẽn mạng cũng được tính đến. OpenQoS xem xét một liên kết bị tắc nghẽn nếu việc sử dụng nó vượt quá 70% băng thông liên kết. Ưu điểm chính của phương pháp dự trữ tài nguyên này là giảm thiểu độ trễ và mất gói cho luồng không hỗ trợ QoS.

Một framework đại diện cho loại này là OpenQoS. Nó đảm bảo dịch vụ (đường truyền ứng dụng) trên một đường dẫn tối ưu và tập trung chủ yếu vào các luồng đa phương tiện, chẳng hạn như VoIP hoặc video. Framework này dựa trên Floodlight, bộ điều khiển Java phổ biến.

Để tạo ra các tuyến đường hiệu quả cho các luồng dịch vụ ưu tiên cao, bộ điều khiển cần nắm tổng quan về trạng thái mạng hiện tại (mất gói, trễ, cho mỗi liên kết). Trạng thái mạng được thu thập từ các yếu tố chuyển tiếp đơn giản bằng cách tích cực gửi bản tin *Feature request* cho mỗi tính năng quan tâm (lấy mẫu tích cực). Hiệu suất của framework phụ thuộc vào tính chính xác của dữ liệu thu thập được, do đó framework truy vấn các phần tử trong mỗi giây. Yêu cầu này có ảnh hưởng tiêu cực đến khả năng mở rộng, thu thập dữ liệu chính xác cho các mạng lớn là không hiệu quả. Một ưu điểm khác của OpenQoS là nỗ lực xác định các luồng chỉ sử dụng các trường từ OSI mức thấp hơn.

2.3.3 Framework với tập trung quản lí hàng đợi và lập lịch gói

Phiên bản Openflow 1.3 xác định một hàng đợi vào trước ra trước (FIFO) tiêu chuẩn của tất cả các gói tin đến. Điều này có thể ngăn các luồng nhất định (ví dụ như luồng đa phương tiện) đáp ứng yêu cầu QoS nếu phân đoạn mạng bị tắc nghẽn. Để khắc phục hạn chế này, một khung gọi là QoSFlow được đề xuất, cho phép sắp xếp lại các gói tin trong một hàng đợi nhất định. Framework thực hiện một vài cơ chế lập lịch gói khác nhau, đáng lưu ý nhất là Stochastic Fairness Queuing (SFQ). Thuật toán này nhằm mục đích đưa ra mỗi luồng ưu tiên cao một cơ hội công bằng được chuyển tiếp, trái ngược với xếp hàng chuẩn FIFO. Mỗi luồng được gán vào một nhóm, sau đó lần lượt theo vòng, các luồng sẽ được đưa ra lần lượt từ các nhóm tương ứng để đảm bảo công bằng.

Tuy nhiên, có một số yếu tố hạn chế, framework này có giới hạn trong 8 hàng đợi cho mỗi cổng chuyển mạch. Hạn chế này tồn tại do cơ chế slicing được sử dụng trong Openflow 1.0. Mặc dù không phải do bản thân framework, nhưng thực tế này vẫn cần được tính đến. Một bất lợi khác là nó đòi hỏi rằng các switch chạy một bản phân phối Linux thay vì một phần mềm cụ thể của nhà cung cấp. Do đó giải pháp này không trực tiếp áp dụng cho Internet trong tình trạng hiện tại.

2.3.4 Framework cho thực thi chính sách

Vấn đề chính với các SLA được xác định trước, liên quan đến QoS, đó là với kiến trúc Internet hiện tại, không có một con đường tiêu chuẩn và linh hoạt để áp

dụng vào mạng. Hầu hết các công nghệ mới nhất được áp dụng để đạt được các SLA cũng là sở hữu độc quyền. Do đó thấy được sự cần thiết của các nhiệm vụ quản lí linh hoạt và khản năng mở rộng. Thông qua các API phía bắc của bộ điều khiển SDN, nó có thể định nghĩa các SLA và sau đó thực thi chúng trên mặt phẳng chuyển tiếp nằm phía dưới API phía Nam.

Một trong những giải pháp đề xuất trong SDN là PolicyCop, dựa trên trên bộ điều khiển Floodlight. Nó khác với các phương pháp tiếp cận DiffServ hiện tại vì nó không bị giới hạn bởi các lớp traffic tĩnh. Framework này có thể mở rộng dễ dàng vì nó có kiến trúc lớp. Mỗi lớp giao tiếp với các lớp khác thông qua API RESTful JSON, do đó khản năng dễ dàng thêm các lớp mới hoặc trao đổi những lớp cũ. Giải pháp này rất linh hoạt vì nó không đòi hỏi bất kì tài nguyên nào đặc biệt, ngoại trừ các thiết bị chuyển mạch OpenFlow.

Chính sách quản lí được thực hiện bởi hai thành phần: trình xác nhận chính sách (PV) và nhà thi hành chính sách (PE). PV theo dõi mạng để đảm bảo các chính sách đang được áp dụng đúng cách. Để làm được điều này, bộ điều khiển cần thu thập thông tin thường xuyên từ các luồng, bằng cách đưa các gói tin dò vào mạng. Trong khi đó, thống kê thụ động cũng được tập hợp từ các gói tin chuyển hướng đến bộ điều khiển. Thu thập các thông tin chỉ số mô tả trạng thái mạng, như sử dụng băng thông, dung lượng dư, số gói tin bị giảm, độ trễ, tỉ lệ lỗi và jitter. PE đặt ra các quy tắc mới trong các phần tử chuyển tiếp để tính toán các thay đổi có thể xảy ra hoặc sai lệch so với các chính sách đã xác định.

2.4 Kết luận

Trong chương này ta có thể thấy việc đảm bảo chất lượng dịch vụ ngày càng không ngừng nâng cao và cải tiến đặc biệt. Từ những giải pháp hạn chế ban đầu như InServ hay DiffServ là động lực thúc đẩy nghiên cứu. Sự ra đời của các giải pháp trên mạng định nghĩa bằng phần mềm (SDN) là điều tất yếu. Nhờ khả năng điều khiển linh hoạt, có giao diện điều khiển đơn giản và lập trình được giúp cho các nhà cung cấp dịch vụ dễ dàng trong quản lý cũng như xây dựng các mô hình dịch vụ tới khách hàng.

CHƯƠNG 3: TRIỂN KHAI MÔ PHỎNG VÀ ĐÁNH GIÁ HIỆU NĂNG GIẢI PHÁP ĐẢM BẢO QoS

3.1 Giới thiệu chung

Ngày nay các ứng dụng thường được phát triển trong môi trường mạng hỗn hợp có sử dụng nhiều chủng loại công nghệ rất khác nhau. Trong môi trường mạng như vậy, một đặc điểm nổi bật là chất lượng dịch vụ của các phần mạng khác nhau không hoàn toàn giống nhau. Mặt khác, các ứng dụng đa phương tiện phân tán chạy trên môi trường mạng như vậy lại có nhu cầu khắt khe về chất lượng dịch vụ. Một hệ thống đầu cuối có thể chạy đồng thời nhiều ứng dụng và các ứng dụng này cũng có yêu cầu rất khác nhau về mức chất lượng dịch vụ. Để đảm bảo chất lượng dịch vụ cho nhiều ứng dụng cùng chạy đồng thời trên một hệ thống đầu cuối, cùng chia sẻ và cạnh tranh tài nguyên hữu hạn của hệ thống và mạng, vấn đề đặt ra là cần có kiến trúc và cơ chế điều khiển thích nghi QoS theo ứng dụng để có thể vừa đáp ứng được yêu cầu chung về chất lượng dịch vụ của tất cả các ứng dụng, vừa có khả năng điều khiển thích nghi QoS cho từng ứng dụng cụ thể. Kỹ thuật QoS trong mạng điều khiển bằng phần mềm là giải pháp tiện lợi giúp hỗ trợ giảm những tác động bất lợi trong việc truyền các luồng dữ liệu như mất gói tin, trễ, tắc nghẽn… nhưng vẫn đảm bảo được chất lượng đầy đủ, tăng tính tin cậy của các luồng dữ liệu.

Nhằm đáp ứng các yêu cầu truyền dữ liệu đảm bảo chất lượng cho các luồng dữ liệu ưu tiên khác nhau qua mạng, trong chương 3 của luận văn này chủ yếu tập trung nghiên cứu, xây dựng và triển khai kịch bản đảm bảo QoS cho kết nối trong mạng SDN được thực hiện trên nền tảng mạng SDN với bộ điều khiển Ryu và các chuyển mạch OVS. Trong hầu hết các trường hợp, bộ điều khiển là một máy chủ lưu trữ và xử lý thông tin điều khiển chung. Một thiết bị chuyển mạch SDN đơn, ví dụ OVS, có thể được "định nghĩa" để hoạt động như bất kỳ phần tử mạng nào mà người quản trị mong muốn nó trở thành, cho dù nó là một switch, một router hoặc thậm chí là một middlebox hay firewall, NAT hoặc bộ cân bằng tải. Với một lợi ích

vô cùng lớn của mạng SDN là thuận tiện trong việc cấu hình các thiết bị, định tuyến QoS cho luồng dữ liệu ưu tiên. Nhà sản xuất không cần phải cấu hình trên từng thiết bị đơn lẻ, do vậy giảm bớt được gánh nặng cho nhà vận hành mạng. Trong nội dung luận văn này, học viện thực hiện triển khai mô phỏng và đánh giá hiệu năng của giải pháp đảm bảo QoS dựa trên kỹ thuật DiffServ trên hạ tầng mạng truyền thông SDN-IoT.

3.2 Kịch bản mô phỏng QoS trong SDN-IoT

3.2.1 Giới thiệu các công cụ mô phỏng

a) Mininet

Mininet là một trình giả lập để triển khai mạng lớn trên các nguồn tài nguyên hạn chế của một máy tính đơn giản hay máy ảo, là phần mềm mã nguồn mở miễn phí mà giả lập thiết bị và bộ điều khiển cho phép nghiên cứu trong SDN và OpenFlow. Mininet là một công cụ quan trọng đối với cộng đồng mã nguồn mở SDN bởi nó thường được sử dụng như công cụ để mô phỏng, kiểm tra, xác minh các ứng dụng mới của SDN.

Mininet cho phép tạo topo kích thước quy mô rất lớn, tạo ra một mạng lưới các host, switch, bộ điều khiển và các liên kết ảo, và thực hiện các mô phỏng trên chúng rất dễ dàng. Một số chức năng cơ bản của Mininet:

- ✓ Cung cấp một mô hình mạng mô phỏng đơn giản và rẻ (do không tốn kém chi phí mua các OpenFlow Switch) để phát triển các ứng dụng mạng. Do các OpenFlow Switch trong Mininet có tất cả các tính chất mà OpenFlow Switch thật có được nên việc sử dụng mạng ảo hóa bằng Mininet có ý nghĩa về mặt thực tế.
- ✓ Cho phép các nhà phát triển ứng dụng làm việc đồng thời, một cách độc lập trên cùng một mô hình mạng.
- ✓ Cho phép kiểm thử các mô hình phức tạp mà không cần phải nối dây cho mạng vật lý.
- ✓ Cho phép debug và chạy các bài kiểm tra của các mạng lớn, sử dụng CLI.

- ✓ Hỗ trợ thiết lập các mô hình tùy biến bất kỳ, gồm tập cơ bản các thông số của mô hình.
- ✓ Có thể đem các ứng dụng trên Mininet đi triển khai trên mạng thật với code hoàn toàn không cần thay đổi.
- ✓ Cung cấp Python API dễ dàng sử dụng và có khả năng mở rộng.

Ưu điểm của Mininet so với các phương pháp ảo hóa khác như OpenFlowVMS hay Noxrepo.org VM environment gồm có: khởi động nhanh hơn (chỉ tốn vài giây để khởi động một mạng có nhiều OpenFlow switch), tính mở rộng lớn hơn (có thể chứa hàng ngàn nút mạng), cung cấp nhiều băng thông hơn (tổng 2 Gbps với phần cứng bình thường), cài đặt dễ dàng (có sẵn bản VMware để tải về).

Nhược điểm hiện tại của Mininet là chỉ hỗ trợ chạy trên 1 máy tính Linux nên hạn chế về mặt hiệu năng, tuy nhiên trong tương lai gần, nhược điểm này sẽ sớm được khắc phục.

Các thành phần cơ bản trong *mininet*:

- Liên kết: các liên kết trong mạng *mininet* là một cặp Ethernet ảo hoạt động như một đường dẫn kết nối hai giao diện ảo. Các gói tin được gửi từ giao diện này tới giao diện kia, các giao diện này hoàn toàn giống như các cổng Ethernet cho hệ thống và các phần mềm ứng dụng.

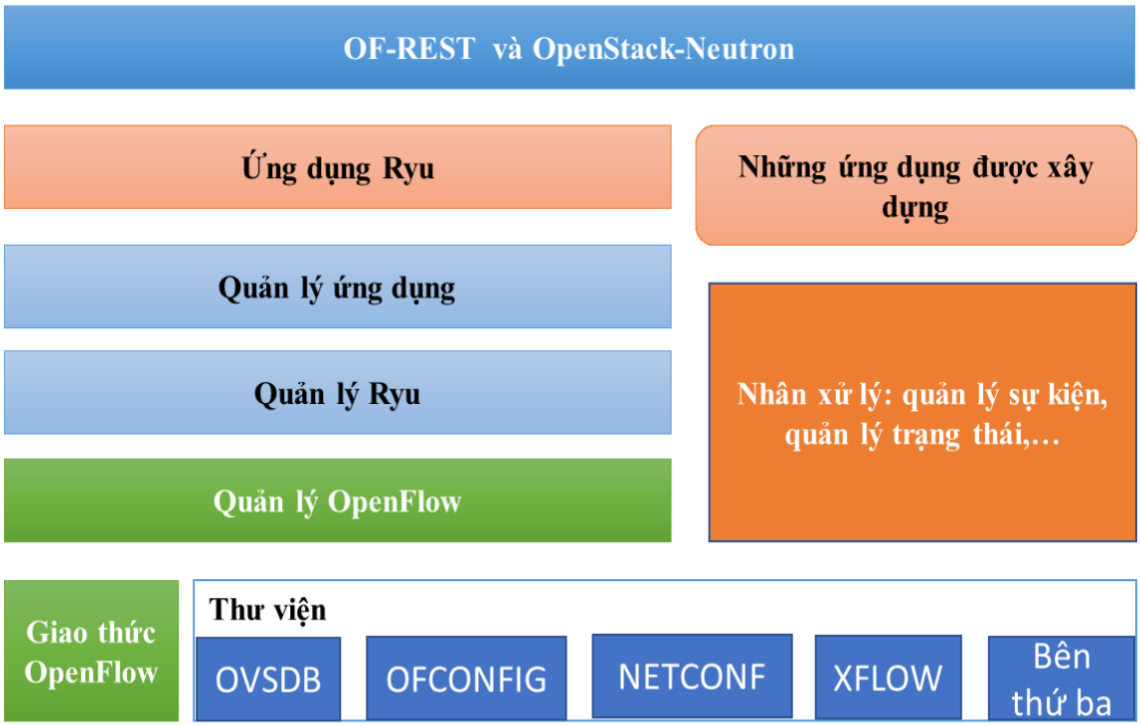
- *Host*: Các network namespaces chứa trạng thái của mạng. Chúng cung cấp các tiến trình (hoặc các nhóm tiến trình) với quyền điều khiển các giao diện, các cổng và bảng định tuyến. Mỗi host có các giao diện Ethernet riêng (khởi tạo và cài đặt bởi lệnh *ip link add/set*) và một đường kết nối tới tiến trình *mininet*, tiến trình này gửi các lệnh và hiển thị thông tin của mạng.

- *Switch*: Các bộ chuyển mạch hỗ trợ giao thức OpenFlow cung cấp cùng một ngữ nghĩa cho việc gửi các gói tin giống như chuyển mạch phần cứng.

- Bộ điều khiển: Bộ điều khiển có thể ở bất cứ nút nào trong mạng thật hoặc trong môi trường mô phỏng.

b) Bộ điều khiển Ryu (Ryu Controller)

Hiện nay có nhiều nhà phát triển và các cộng đồng nghiên cứu SDN đã đưa ra bộ điều khiển SDN của riêng họ. Hầu hết các bộ điều khiển này đều là mã nguồn mở với tính đa dạng và khả năng mở rộng rất cao theo nhiều phiên bản và hướng đi khác nhau cho các phần mềm điều khiển. Với sự đa dạng như vậy, ta có thể lựa chọn một phần mềm phù hợp với các tình huống và tiêu chí khác nhau. Bộ điều khiển Ryu do các nhóm nghiên cứu của Nhật cùng nhau phát triển đã được lựa chọn cho thiết kế mô phỏng. Mặc dù không được sử dụng nhiều như OpenDaylight nhưng Ryu được coi là khá đơn giản và dễ học cho người mới bắt đầu. Mã nguồn Ryu được lưu trữ trên GitHub và quản lí duy trì bởi cộng đồng Ryu mở. Kiến trúc cơ bản của bộ điều khiển Ryu được mô tả trên hình 3.1.



Hình 3.1 Kiến trúc bộ điều khiển Ryu

- *Thư viện Ryu*

Bộ thư viện của Ryu rất đồ sộ, hỗ trợ cho nhiều giao thức cho phía giao diện hướng nam đến các hoạt động xử lý gói tin đa dạng. Đối với các giao thức hỗ trợ giao diện phía nam, Ryu hỗ trợ OF-config, giao thức quản lý cơ sở dữ liệu Open vSwitch (OVSDB), NETCONF, Xflow (Netflow và Sflow) và các giao thức ở các

bên thứ ba. Hai giao thức Netflow và Sflow sử dụng để đo lưu lượng mạng bằng cách lấy mẫu và tập hợp gói tin. Các thư viện của bên thứ ba như Open vSwitch, thư viện cấu hình Oslo và thư viện Python cho NETCONF. Thư viện gói Ryu giúp phân tích và xây dựng các gói giao thức khác nhau, chẳng hạn như VLAN, MPLS, GRE,...

- *Giao thức và bộ điều khiển OpenFlow*

Ryu hỗ trợ giao thức OpenFlow phiên bản từ 1.0 đến phiên bản 1.5. Bộ điều khiển này bao gồm một thư viện mã hóa và giải mã giao thức OpenFlow. Ngoài ra, một trong những thành phần chủ chốt trong kiến trúc Ryu chính là bộ điều khiển giao thức OpenFlow, khối này đảm nhận trách nhiệm quản lý việc chuyển mạch OpenFlow và cấu hình từng luồng tin, quản lý sự kiện,…

- *Khối quản lý và nhân xử lý*

Khối quản lý Ryu thực hiện việc lắng nghe các kết nối trên địa chỉ IP (như 0.0.0.0) và số hiệu cổng (như cổng mặc định 6633) từ những bộ chuyển mạch ở lớp mặt bằng dữ liệu như phần cứng chuyển mạch hoặc bộ chuyển mạch Open vSwitch. Trình quản lý ứng dụng là thành phần cơ bản cho tất cả các ứng dụng của bộ điều khiển Ryu. Tất cả các ứng dụng đều kế thừa từ lớp quản lý ứng dụng. Thành phần xử lý lõi của kiến trúc bao gồm quản lý sự kiện, gửi thông điệp, quản lý trạng thái trong bộ nhớ,…

- *Giao diện phía Bắc*

Tại lớp API, Ryu bao gồm một *plug-in* Neutron Openstack hỗ trợ cả cấu hình đường hầm bảo mật GRE-Tunnel và cấu hình VLAN. Ryu cũng hỗ trợ giao diện REST cho các hoạt động OpenFlow. Ngoài ra, sử dụng WSGI (một *framework* để kết nối các ứng dụng web và các máy chủ web bằng Python), có thể dễ dàng áp dụng các API REST mới hơn vào một ứng dụng.

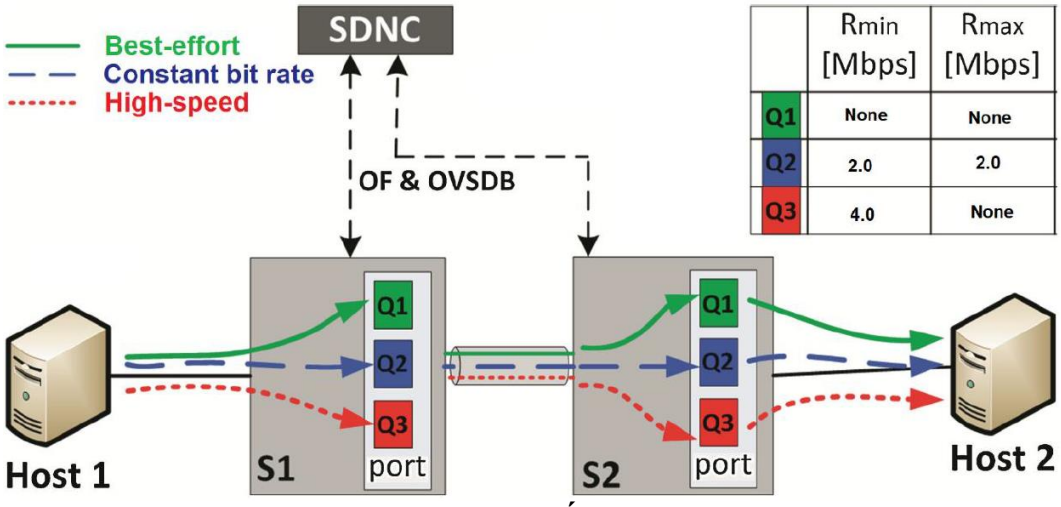
- *Khối ứng dụng*

Ryu hỗ trợ rất nhiều ứng dụng như chuyển mạch, định tuyến, thiết lập tường lửa, đường hầm GRE, cấu hình topo mạng, VLAN,… Những ứng dụng này sử dụng đơn luồng với nhiều hàm xử lý khác nhau. Các ứng dụng gửi các sự kiện không đồng bộ cho nhau.

Mỗi ứng dụng trong Ryu nằm trong hàng đợi sự kiện, sử dụng phương thức vào trước ra trước để xử lý từng ứng dụng trong hàng đợi. Ngoài ra mỗi ứng dụng bao gồm một luồng để xử lý các sự kiện từ hàng đợi. Luồng xử lý chính sẽ gọi trình xử lý sự kiện để chạy sự kiện thích hợp. Như vậy các ứng dụng trong Ryu sẽ chạy tuần tự và thứ tự chạy sẽ do trình quản lý sự kiện đảm nhận.

3.2.2 Kịch bản mô phỏng

Hệ thống mạng ngày càng phát triển, các thiết bị đầu cuối với nhu cầu kết nối vào hệ thống mạng ngày càng tăng, việc quản lí một số thiết bị không đòi hỏi trao đổi thông tin băng thông lớn nhưng lại yêu cầu tính thời gian thực cao như điều khiển lò vi sóng, các thiết bị có độ nhạy thời gian cao như báo cháy, báo khói, … Bên cạnh đó, một số dịch vụ yêu cầu băng thông càng lớn càng tốt để đảm bảo chất lượng, ví dụ các hệ thống camera giám sát chống trộm có khả năng phân tích và nhận dạng hình ảnh từ xa. Các hệ thống này yêu cầu loại lưu lượng có thỏa thuận băng thông tối thiểu. Nhiều loại dịch vụ khác có thể yêu cầu băng thông cố định hoặc không yêu cầu có thỏa thuận chất lượng dịch vụ. Với bản chất *best-effort* (nỗ lực tối đa), mạng IP thông thường sẽ khó có thể đảm bảo chất lượng dịch vụ đa dạng theo cùng kết nối trong mạng.



Hình 3.2 Mô hình hệ thống mô phỏng QoS

Để mô phỏng xây dựng hạ tầng thông tin đảm bảo QoS cho các luồng dữ liệu ưu tiên, trong nội dung luận văn này, các dịch vụ mô phỏng được giả định chia thành 03 loại lưu lượng điển hình như sau (Hình 3.2):

- *Lưu lượng tốc độ cố định*: dành cho các dịch vụ yêu cầu thời gian thực, băng thông không thay đổi theo thời gian .
- *Lưu lượng tốc độ giới hạn dưới*: dành cho các dịch vụyêu cầu băng thông lớnvới độ tin cậy cao, do vậy băng thông luôn phải lớn hơn hoặc bằng giá trị thiết lập trước.
- *Lưu lượng thông thường*: dành cho các dịch vụkhông có yêu cầu đặc biệt vềbăng thông. Băng thông cung cấp cho các dịch vụ này sẽ được xử lý theo cơ chế *best-effort* của mạng IP thông thường.

Như mô tả trên hình 3.2, trong hệ thống mô phỏng này, tổng tốc độ kết nối tối đa của liên kết cho cả ba dịch vụ là 10 Mbps. Kỹ thuật cắt lát băng thông được sử dụng để chia sẻ tài nguyên trong hệ thống. Các thiết bị chuyển mạch SDN và môi trường mạng SDN được mô phỏng trong mininet.

Bảng 3.1 thể hiện thông tin thiết lập hàng đợi cho các loại hình dịch vụ mô phỏng trong hệ thống. Trong đó, lần lượt 3 loại hình lưu lượng được thiết lập cho 3 cổng với định danh tương ứng là 9000 (ID =0), 9001 (ID =1) và 9002 (ID =2) của các hệ thống chuyển mạch SDN. Cổng 9000 được cấu hình lưu lượng thông thường, cổng 9001 được thiết lập QoS theo tốc độ cố định 2 Mbps (±5%) và cổng 9002 thiết lập QoS theo lưu lượng tốc độ giới hạn dưới là 4 Mbps. Tổng tốc độ kết nối tối đa của liên kết cho ba cổng ra là 10 Mbps.

Bảng 3.1 Lưu lượng cho các dịch vụ khác nhau

ID hàng đợi ưu tiên	Tốc độ tối đa	Tốc độ tối thiểu	Lớp ưu tiên
0	10 Mbps	Không có	Nỗ lực tối đa
1	2 Mbps	2 Mbps	AF3
2	(10 Mbps)	4.0 Mbps	AF4

3.3. Đánh giá hiệu năng giải pháp đảm bảo QoS

3.3.1 Cấu hình hệ thống

Kiến trúc hệ thống demo bao gồm một topo mạng SDN đơn giản được xây dựng trong mininet với một bộ điều khiển. Bộ điều khiển được sử dụng là bộ điều khiển tách rời Ryu. Để bắt đầu kịch bản, sử dụng mininet xây dựng môi trường ta tạo ra topo mạng đơn giản để mô phỏng. Topo mạng với 2 thiết bị chuyển mạch OpenvSwitch và 2 host như hình 3.3 với câu lệnh sử dụng sau:

$ sudo mn –topo linear,2 --mac --switch ovsk --controller remote –x

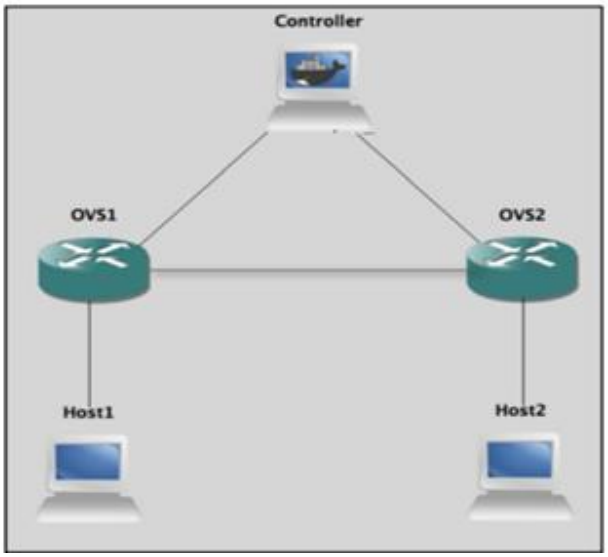
Bảng 3.2 và hình 3.3 thể hiện mô tả các tham số sử dụng trong mô phỏng cũng như kết quả đầu ra của câu lệnh trên. Hệ thống demo được tạo ra sẽ là một topo mạng SDN có 2 nút chuyển mạch SDN được kết nối trực tiếp với nhau và được điều khiển thông qua một bộ điều khiển Ryu. Hình 3.4 thể hiện rõ hơn cấu hình mạng mô phỏng.

```
mininet> nodes
available nodes are:
c0 h1 h2 s1 s2
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2
c0
mininet>
```

Hình 3.3 Môi trường mô phỏng trong mininet

Bảng 3.2 Tham số kịch bản mô phỏng

Tham số	Giá trị	Giải thích
Topo	Linear, 2	Topo với 2 switch được tạo ra
mac	None	Thiết lập địa chỉ mac tự động cho host
Switch	ovsk	Sử dụng OpenvSwitch
controller	remote	Sử dụng một bộ điều khiển tách rời
x	None	Mở cửa sổ xterm



Hình 3.4 Cấu hình mạng mô phỏng

Hình 3.5 thể hiện các thông số phiên bản của Switch và các thông số trên cổng chuyển mạch của các Switch được kiểm tra bằng câu lệnh ***ovs-vsctl show*** thực hiện trong giao diện điều khiển mininet.

```
root@mininet-vm:~# ovs-vsctl show
918037ec-b307-45d7-a75a-f1ac4337d135
    Manager "ptcp:6632"
    Bridge "s1"
        Controller "ptcp:6654"
        Controller "tcp:192.168.60.100:6633"
        fail_mode: secure
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1-eth2"
            Interface "s1-eth2"
    Bridge "s2"
        Controller "ptcp:6655"
        Controller "tcp:192.168.60.100:6633"
        fail_mode: secure
        Port "s2"
            Interface "s2"
                type: internal
        Port "s2-eth2"
            Interface "s2-eth2"
        Port "s2-eth1"
            Interface "s2-eth1"
    ovs_version: "2.0.2"
root@mininet-vm:~#
```

Hình 3.5 Show cấu hình phiên bản và cổng chuyển mạch của OvS

Kết quả thể hiện trên hình 3.5 cho thấy hệ thống chuyển mạch đang trao đổi với bộ điều khiển thông qua giao thức Openflow phiên bản 1.3 và Openflow lắng nghe trên cổ ng ptcp: 6632 để truy cập OVSDB. Phiên bản OpenvSwitch sử dụng là phiên bản 2.0.2. Hình 3.6 thể hiện bộ điều khiển Ryu đã kết nối thành công và điều khiển thiết bị chuyển mạch.

```
mininet@mininet-vm:~$ ryu-manager ryu.app.rest_qos ryu.app.qos_simple_switch_13
ryu.app.rest_conf_switch
loading app ryu.app.rest_qos
loading app ryu.app.qos_simple_switch_13
loading app ryu.app.rest_conf_switch
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
instantiating app None of ConfSwitchSet
creating context conf_switch
creating context wsgi
instantiating app ryu.app.rest_conf_switch of ConfSwitchAPI
instantiating app ryu.app.qos_simple_switch_13 of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app ryu.app.rest_qos of RestQoSAPI
(4975) wsgi starting up on http://0.0.0.0:8080
unsupported version 0x1. If possible, set the switch to use one of the versions
[4]
[QoS][INFO] dpid=0000000000000002: Join qos switch.
```

Hình 3.6 Kết nối thành công giữa bộ điều khiển Ryu và thiết bị chuyển mạch

3.3.2 Module điều khiển định tuyến đảm bảo QoS

Trong hệ thống mô phỏng, để thực hiện điều khiển định tuyến cho bộ chuyển mạch ở lớp cơ sở hạ tầng, phần mềm ứng dụng cho bộ điều khiển dựa trên ngôn ngữ Python đã được xây dựng. Việc sử dụng máy tính cỡ nhỏ với giá thành rẻ kết hợp với các phần mềm điều khiển mã nguồn mở cho phép giảm thiểu giá thành hệ thống là tăng tính linh hoạt trong việc ứng dụng hệ thống. Thiết bị điều khiển SDN kích thước nhỏ đã điều khiển chuyển mạch qua giao thức OpenFlow.

Để xử lý tác vụ thêm vào bảng định tuyến, giá trị địa chỉ sẽ được kiểm tra sự trùng lặp địa chỉ. Lỗi xảy ra khi (1) địa chỉ đích (*dst_nw_addr*) là d*efault route* (0.0.0.0/0) và (2) địa chỉ đích trùng với địa chỉ nguồn của bản tin, thông báo lỗi sẽ được hiển thị trên màn hình điều khiển. Nếu không có lỗi xảy ra, địa chỉ mới sẽ được gán thêm vào bảng với giá trị địa chỉ đích (*dst_ip*), định danh luồng tin (*route_id*), mặt nạ địa chỉ (*netmask*) kèm địa chỉ bộ chuyển mạch hàng xóm chuyển tiếp gói tin (*gateway_ip*). Mỗi giá trị *route_id* được tăng lên để định danh cho một

luồng dữ liệu khác. Đoạn code ở Hình 3.7 minh họa cho quá trình cập nhật vào bảng định tuyến.

```
class RoutingTable(dict):
    def add(self, dst_nw_addr, gateway_ip):
        err_msg = 'Invalid [%s] value.'

        if dst_nw_addr == DEFAULT_ROUTE:
            dst ip  = 0      #dst ip: địa chỉ đích của gói tin đến
            netmask =  0     #netmask: subnet mask của gói tin đến
        else:
            dst ip, netmask, dummy = nw addr aton(
               dst nw addr, err msg=err msg % REST DESTINATION)
        #gateway: địa chỉ next-hop
        gateway_ip = ip_addr_aton(gateway_ip, err_msg=err_msg %
REST_GATEWAY)
        # Kiểm tra trùng lặp địa chỉ
        overlap_route = None
        if dst_nw_addr == DEFAULT_ROUTE:
            if DEFAULT_ROUTE in self:
                overlap route = self[DEFAULT ROUTE].route id
        elif dst_nw_addr in self:
            overlap_route = self[dst_nw_addr].route_id
        if overlap_route is not None:
            msg = 'Destination overlaps [route_id=%d]' % overlap_route
            raise CommandFailure(msg=msg)
        # Lưu địa chỉ vào bảng
        routing_data = Route(self.route_id, dst_ip, netmask, gateway_ip)
        ip_str = ip_addr_ntoa(dst_ip)
        key = '%s/%d' % (ip_str, netmask)
        self[key] = routing_data

        self.route_id += 1
        self.route_id &= UINT32_MAX
        if self.route_id == COOKIE_DEFAULT_ID:
            self.route_id = 1

        return routing_data
```

Hình 3.7 Đoạn chương trình cập nhật vào bảng định tuyến

Thêm nữa, hình 3.8 mô tả cách xử lý luồng dữ liệu gửi đến bộ chuyển mạch. Các địa chỉ hàng xóm của một bộ chuyển mạch được lấy ra từ bảng định tuyến thông qua biến *routing_tbl.* Sau đó, luồng tin đi đến bộchuyển mạch sẽ được xử lý ở hàm *packet_in_handler().* Các gói tin được phân loại theo bản tin ARP, bản tin ICMP, TCPhay UDP. Mỗi loại bản tin sẽ được phân tích địa chỉ đích và gửi đến đúng thiết bị đầu cuối kết nối với bộ chuyển mạch. Nếu các bản tin thuộc loại khác thì sẽ được xử lý để chuyển đến một nút chuyển mạch khác.

```
    # Lấy address_id(dpid) của các bộ chuyển mạch liền kề
    def _chk_addr_relation_route(self, address_id):
        # Kiểm tra dữ liệu định tuyến liên quan
        relate_list = []
        gateways = self.routing_tbl.get_gateways()
        for gateway in gateways:
            address = self.address_data.get_data(ip=gateway)
            if address is not None:
                if (address_id == REST_ALL
                        and address.address_id not in relate_list):
                    relate_list.append(address.address_id)
                elif address.address_id == address_id:
                    relate_list = [address_id]
                    break
        return relate_list

    # Hàm xử lý luồng tin đến
    def packet_in_handler(self, msg, header_list):
        # Phân tích loại gói tin
        if ARP in header_list:
            self._packetin_arp(msg, header_list)
            return

        if IPV4 in header_list:
            rt_ports = self.address_data.get_default_gw()
            if header_list[IPV4].dst in rt_ports:
                # Chuyển gói tin đến host kết nối với bộ chuyển mạch
                if ICMP in header_list:
                    if header_list[ICMP].type == icmp.ICMP_ECHO_REQUEST:
                        self._packetin_icmp_req(msg, header_list)
                        return
                elif TCP in header_list or UDP in header_list:
                    self._packetin_tcp_udp(msg, header_list)
                    return
            else:
                # Chuyển gói tin đến bộ chuyển mạch khác
                self._packetin_to_node(msg, header_list)
                return
```

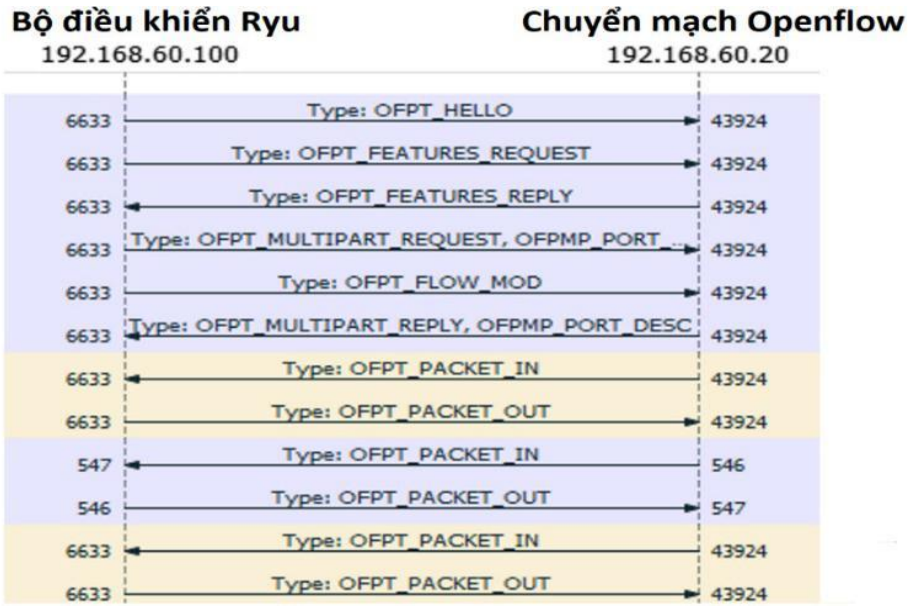
Hình 3.8 Đoạn chương trình xử lý luồng dữ liệu đến

3.3.3 Đánh giá và nhận xét kết quả

Trong kịch bản đã mô phỏng hoạt động của hệ thống và sử dụng công cụ WireShark (phần mềm bắt các gói tin) khi thiết lập kết nối và truyền thông qua hệ thống chuyển mạch được thiết kế. Hình 3.9 thể hiện kết quả đạt được trong đó cho thấy hệ thống hoạt động đúng theo thiết kế. Các bản tin Openflow được sử dụng để trao đổi thông tin kết nối giữa bộ điều khiển (192.168.60.100) và hệ thống chuyển mạch trên mininet (192.168.60.20).

192.168.60.100	192.168.60.20	OpenFlow	76 Type: OFPT_HELLO
192.168.60.100	192.168.60.20	OpenFlow	76 Type: OFPT_FEATURES_REQUEST
192.168.60.20	192.168.60.100	OpenFlow	100 Type: OFPT_FEATURES_REPLY
192.168.60.100	192.168.60.20	OpenFlow	84 Type: OFPT_MULTIPART_REQUEST, OFPMP_PORT_DESC
192.168.60.100	192.168.60.20	OpenFlow	148 Type: OFPT_FLOW_MOD
192.168.60.20	192.168.60.100	OpenFlow	276 Type: OFPT_MULTIPART_REPLY, OFPMP_PORT_DESC
192.168.60.20	192.168.60.100	OpenFlow	170 Type: OFPT_PACKET_IN
192.168.60.100	192.168.60.20	OpenFlow	168 Type: OFPT_PACKET_OUT
192.168.60.20	192.168.60.100	OpenFlow	262 Type: OFPT_PACKET_IN
192.168.60.100	192.168.60.20	OpenFlow	259 Type: OFPT_PACKET_OUT
192.168.60.20	192.168.60.100	OpenFlow	170 Type: OFPT_PACKET_IN
192.168.60.100	192.168.60.20	OpenFlow	168 Type: OFPT_PACKET_OUT

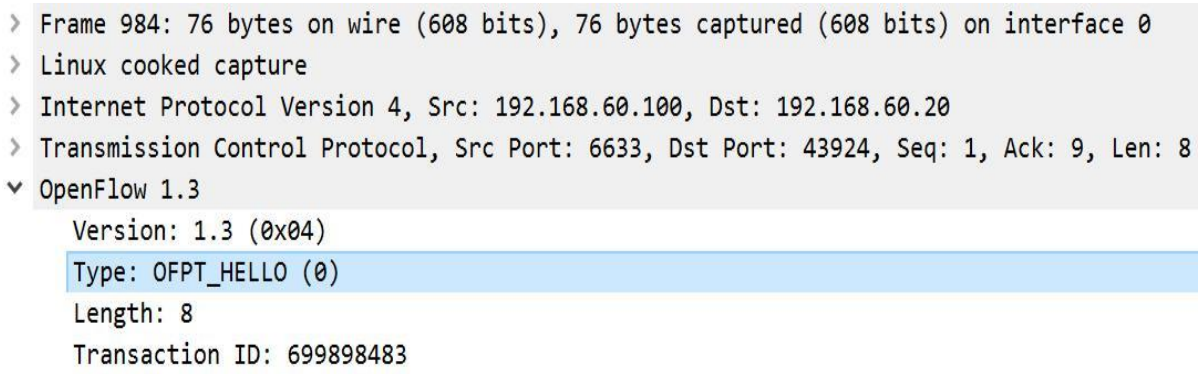
Hình 3.9 Bắt gói tin trao đổi bằng Wireshark



Hình 3.10 Trao đổi bản tin giữa bộ điều khiển Ryu và hệ thống chuyển mạch OpenvSwitch

Hoạt động của hệ thống có thể được tóm tắt như sau: khi mới kết nối, bộ điều khiển và hệ thống chuyển mạch sẽ trao đổi bản tin Hello với nhau (xem Hình 3.10) để thiết lập kết nối đàm phán phiên bản. Nếu đàm phán thất bại chúng sẽ gửi bản tin Error. Sau khi thiết lập kết nối xong, bộ điều khiển sẽ gửi bản tin Feature_Request để xác định đặc tính và chức năng của OVS. OVS sẽ trả lời bằng bản tin Feature_Respond. Tiếp theo bộ điều khiển gửi bản tin Multipath Request để truy vấn thông tin theo từng luồng dữ liệu (port, interface, …). OVS trả lời bằng Multipath_Respond chứa các thông tin trên. Tiếp đến bộ điều khiển gửi bản tin Flow_Mod (như trình bày trên Hình 3.11) khi muốn thay đổi trạng thái của OVS.

Khi các bản tin cơ bản được trao đổi xong thì 2 thiết bị sẽ trao đổi bản tin Echo để trao đổi thông tin về độ trễ, băng thông và tính linh hoạt. Bản tin Packet_In và Packet_Out dùng để trao đổi dữ liệu.



Hình 3.11 Bản tin Openflow OFPT_FLOW_MOD

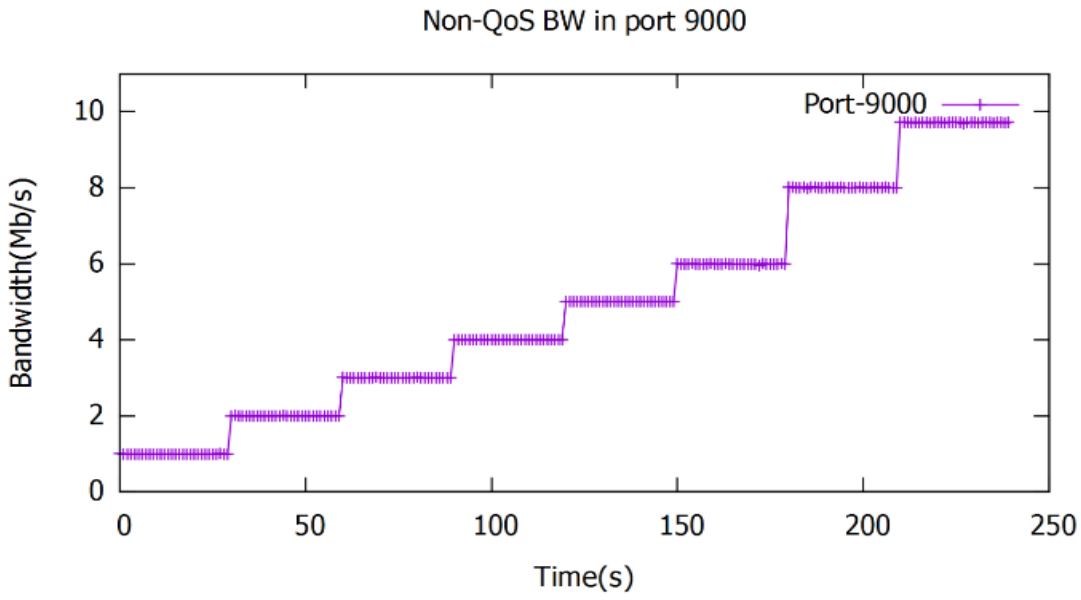
Trong kịch bản này đã thực hiện khảo sát tốc độ tối đa của cổng Ethernet trên hệ thống chuyển mạch được thiết kế. Để thực hiện điều này, chúng ta có thể sử dụng cả 2 cách là: 1 là đo kiểm bằng lệnh hiển thị cấu hình (*ovs-ofctrl show*) và 2 là tạo lưu lượng tăng dần và truyền qua hệ thống chuyển mạch.

Để xác thực khả năng hoạt động của hệ thống, lưu lượng ngẫu nhiên theo các tốc độ tăng dần được tạo ra và truyền qua các luồng dữ liệu được thiết lập QoS khác nhau của hệ thống chuyển mạch. Trong kịch bản này, lưu lượng tại cổng 9000 được yêu cầu truyền với tốc độ tối đa của liên kết là 4 Mbps. Lưu lượng trên cổng 9001 truyền với tốc độ cố định là 2 Mbps trong khi tốc độ truyền trên cổng 9002 được tăng dần sau mỗi khoảng thời gian 30 giây. Lưu lượng trên hai cổng 9001 và 9002 được khởi phát bắt đầu chậm 30 giây so với lưu lượng trên cổng 9000. Kết quả cho thấy, lưu lượng trên cổng 9000 bị giảm dần nhường lại bang thông cho các cổng có được thiết lập QoS. Tổng lưu lượng trên cả 3 cổng luôn đảm bảo là khoảng 10 Mbps (bằng tốc độ tối đa cho phép). Hai cổng 9001 và 9002 có tốc độ lưu lượng theo đúng yêu cầu QoS được thiết lập.

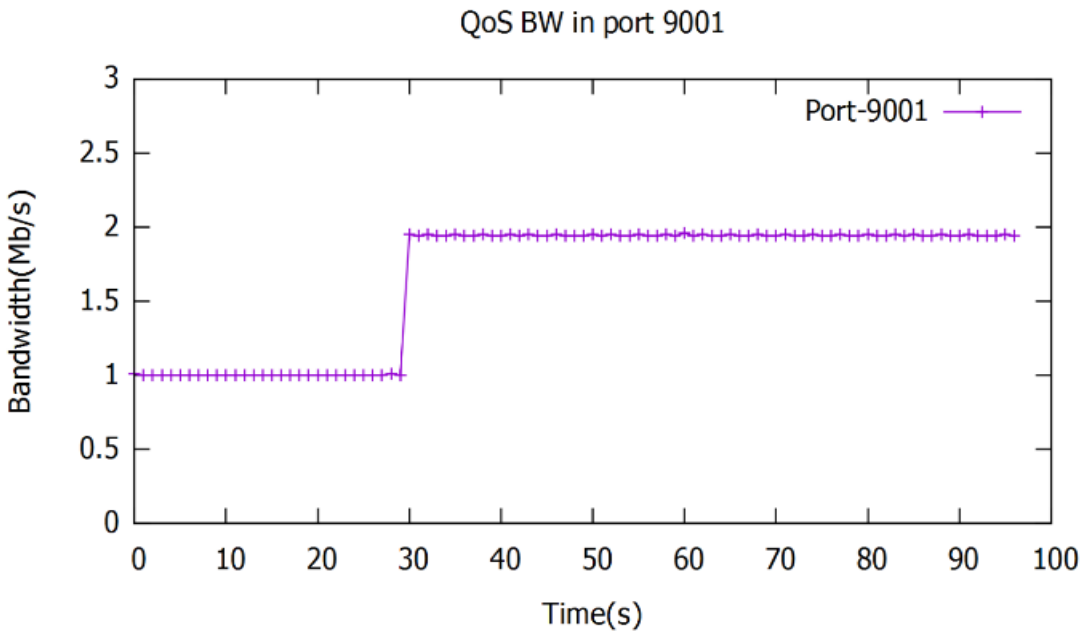
Để xác định khả năng hoạt động của hệ thống, ta tạo luồng lưu lượng ngẫu nhiên tăng dần và truyền qua các cổng được thiết lập QoS khác nhau của hệ thống chuyển mạch. Sử dụng phần mềm iperf, đây là một công cụ đơn giản dễ sử dụng để kiểm tra băng thông qua hệ thống mạng, bắn lưu lượng qua các cổng với một host đóng vai trò làm máy chủ và một host đóng vai trò là máy khách.

Hình 3.12, 3.13 và 3.14 thể hiện kết quả về băng thông đầu ra tại các cổng

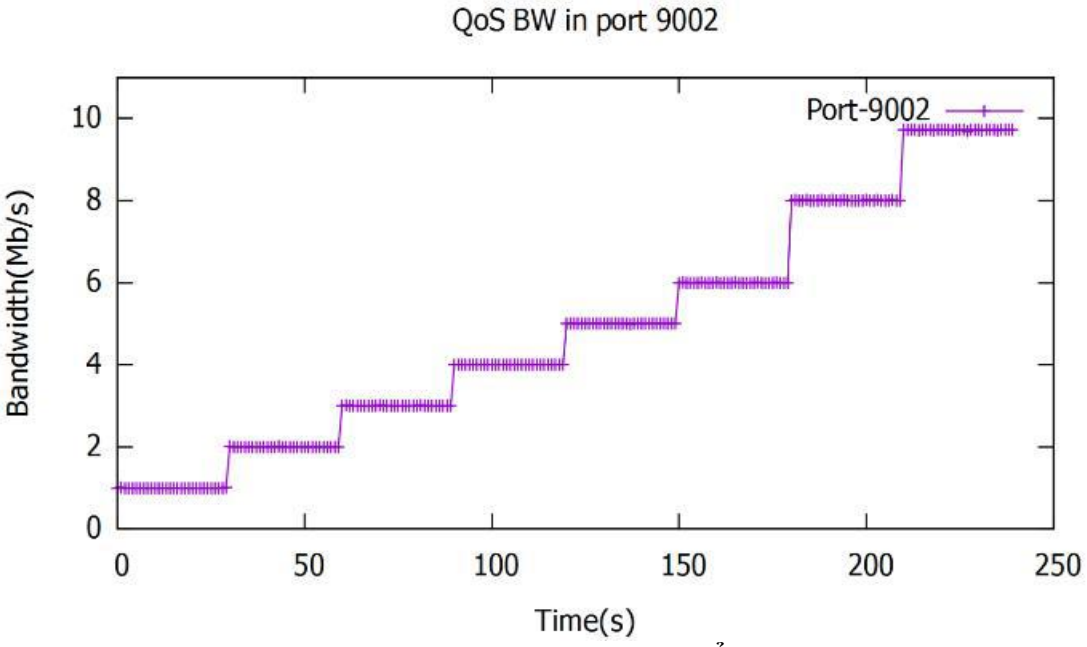
của hệ thống chuyển mạch Openflow. Kết quả cho thấy đối với cổng 9000 và 9002, do lưu lượng không giám sát hoặc chỉ giới hạn tốc độ tối thiểu, tốc độ lưu lượng đầu ra của cổng vẫn như tốc độ truyền vào hệ thống. Tuy nhiên, đối với cổng 9001, do tốc độ bị giới hạn cố định là 2 Mbps, nên khi truyền dữ liệu có tốc độ cao thì đều bị giới hạn không lớn hơn 2 Mbps. Khi tốc độ quá cao, ví dụ từ 4 Mbps tại giây thứ 100 thì cổng bị ngắt kết nối do tỉ lệ mất gói quá lớn.



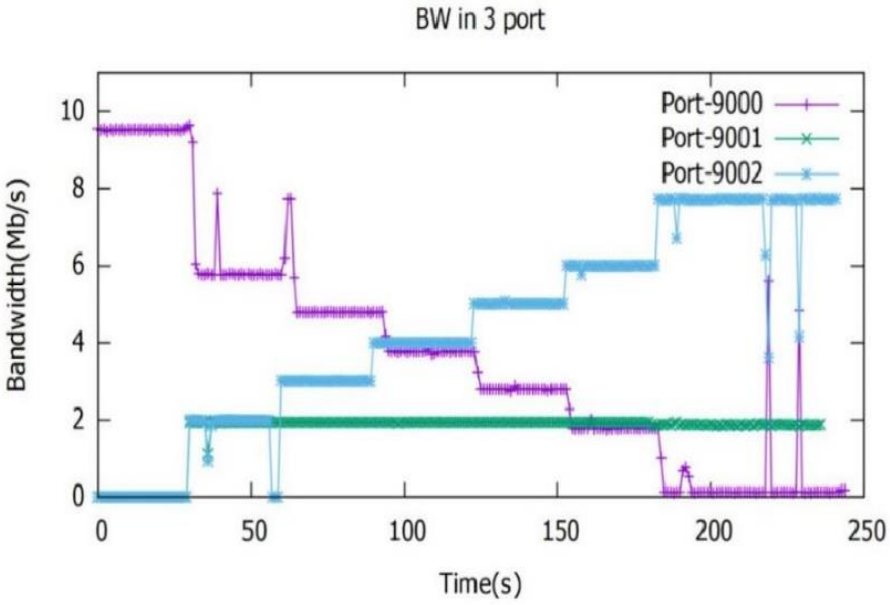
Hình 3.12 Băng thông qua cổng 9000



Hình 3.13 Băng thông qua cổng 9001



Hình 3.14 Băng thông qua cổng 9002



Hình 3.15 Kịch bản đảm bảo QoS cho các cổng theo cấu hình thiết lập trước

Ngoài ra, hình 3.15 thể hiện kịch bản đảm bảo QoS cho lưu lượng tại các cổng của hệ thống chuyển mạch theo cấu hình đã thiết lập. Trong kịch bản này, lưu lượng tại cổng 9000 được yêu cầu truyền với tốc độ tối đa của liên kết là 10 Mbps. Lưu lượng trên cổng 9001 truyền với tốc độ cố định là 2 Mbps trong khi tốc độ truyền trên cổng 9002 được tăng dần sau mỗi khoảng thời gian nhất định. Lưu

lượng trên hai cổng 9001 và 9002 được khởi phát bắt đầu chậm 30 giây so với lưu lượng trên cổng 9000. Kết quả cho thấy, lưu lượng trên cổng 9000 bị giảm dần để ưu tiên cho các cổng có QoS được thiết lập. Tổng lưu lượng trên cả 3 cổng luôn đảm bảo là khoảng 10 Mbps (bằng tốc độ tổng cho phép). Hai cổng 9001 và 9002 có tốc độ lưu lượng theo đúng yêu cầu QoS được thiết lập.

3.4 Kết luận chương

Nội dung chương 3 trình bày triển khai mô phỏng kỹ thuật đảm bảo QoS trong hệ thống mạng SDN-IoT sử dụng công cụ mô phỏng mininet và bộ điều khiển SDN Ryu. Các mô phỏng được triển khai cho ba loại hình chất lượng dịch vụ cơ bản là best-effort, tốc độ cố định và tốc độ cao. Các kết quả đạt được thể hiện khả năng đảm bảo chất lượng dịch vụ linh hoạt của hệ thống dựa trên nền tảng công nghệ mạng định nghĩa bằng phần mềm.

KẾT LUẬN

Trong thời đại bùng nổ công nghệ thông tin, IoT đã và đang phát triển ngày càng mạnh về số lượng thiết bị kết nối, số lượng kết nối và loại hình kết nối,… Qua đó đòi hỏi yêu cầu càng cao của chất lượng các kết nối như về băng thông, độ trễ và các tham số khác. Do đó các giải pháp và công nghệ mạng đang hướng đến việc giải quyết các thách thức trên nhằm hỗ trợ số lượng kết nối lớn cũng như thiết bị và sự đa dạng của chủng loại thiết bị kết nối.Việc xây dựng các giải pháp để đảm bảo chất lượng mà không ảnh hưởng tới các luồng dữ liệu khác trên mạng ngày càng thực sự cần thiết.

Trong luận văn này, học viên đã thực hiện nghiên cứu khái quát về công nghệ IoT và công nghệ mạng định nghĩa bằng phần mềm cũng như khả năng ứng dụng của SDN trong IoT (IoT định nghĩa bằng phần mềm). Nội dung luận văn đã tập trung khảo sát các giải pháp đảm bảo chất lượng dịch vụ (QoS) trong mạng IP và trong mạng IoT định nghĩa bằng phần mềm. Trên cơ sở đó, luận văn cũng trình bày triển khai mô phỏng ứng dụng giải pháp đảm bảo QoS trong mạng IoT định nghĩa bằng phần mềm và đánh giá hiệu năng của hệ thống. Việc đánh giá hiệu năng của giải pháp đảm bảo chất lượng dịch vụ được thực hiện theo ba loại hình dịch vụ cơ bản tiêu biểu bao gồm dịch vụ lưu lượng Best-effort (dịch vụ truyền thống của mạng IP), dịch vụ tốc độ không đổi và dịch vụ lưu lượng tốc độ cao. Các kết quả đạt được cho thấy sự thành công của giải pháp DiffServ trong hạ tầng mạng SDN-IoT.

Trên cơ sở các kết quả đã đạt được trong nội dung luận văn này, một trong các hướng phát triển nghiên cứu tiếp theo của đề tài luận văn là nghiên cứu triển khai ứng dụng công nghệ IoT định nghĩa bằng phần mềm và kỹ thuật đảm bảo QoS trong thực tế mạng lưới của VNPT Bắc Ninh trong tương lai gần nhằm cung cấp các giải pháp cho hạ tầng truyền thông thành phố thông minh.

TÀI LIỆU THAM KHẢO

[1] Vermesan Ovidiu, and Peter Friess (2014) "*Internet of things-from research and innovation to market deployment*," *Vol. 29,* Aalborg: River Publishers.

[2] Alasdair Gilchrist(2016), "*Industry 4.0: The Industrial Internet of Things*".

[3] Wollschlaeger Martin, Thilo Sauter, and Juergen Jasperneite (2017), "*The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0*," IEEE Industrial Electronics Magazine, *vol. 11*, no. 1, pp. 17-27.

[4] Bizanis, Nikos, and Fernando A. Kuipers (2016), "*SDN and virtualization solutions for the Internet of Things: A survey*," IEEE Access, *vol. 4*, pp. 5591-5606.

[5] Samaresh Bera, Sudip Misra, and Athanasios V. Vasilakos (2017), "*Software-DefinedNetworking for Internet of Things: A Survey*," IEEE Internet of Things Journal, *vol. 4*, no. 6, pp. 1994-2008.

[6] Omnes, Nathalie, Marc Bouillon, Gael Fromentoux, and Olivier Le Grand (2015), "*A programmable and virtualized network & IT infrastructure for the internet of things: How can NFV & SDN help for facing the upcoming challenges*," 18th International Conference on Intelligence in Next Generation Networks (ICIN), pp. 64-69.

[7] Diego Kreutz, Fernando M.V. Ramos, Paulo Esteves Verıssimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig (2015), "*Software-defined networking: A comprehensive survey*," Proceedings of the IEEE, *vol. 103*, no. 1, pp. 14-76.

[8] Andreas, Arsanasty Ba, Martin Reisslein, and Wolfgang Kellerer (2016), "*Survey on network virtualization hypervisors for software defined networking*," IEEE Communications Surveys & Tutorials, *vol. 18*, no. 1, pp. 655-685.

[9] Hu, Fei, Qi Hao, and Ke Bao (2014), "*A survey on software-defined networkand openflow: From concept to implementation*, " IEEE Communications Surveys & Tutorials, *vol. 16*, no. 4, pp. 2181-2206.

[10] https://thenewstack.io/sdn-series-part-iv-ryu-a-rich-featured-open-source-sdn-controller-supported-by-ntt-labs