ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ – VIỄN THÔNG

TRƯƠNG ĐỨC DUY

FLOWSENSE: HỆ THỐNG GIÁM SÁT MẠNG THỤ ĐỘNG DỰA TRÊN THÔNG TIN ĐIỀU KHIỂN TRONG SDN

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: KỸ THUẬT ĐIỆN TỬ – VIỄN THÔNG

Thành phố Hồ Chí Minh - Tháng 8 năm 2025

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ – VIỄN THÔNG

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: KỸ THUẬT ĐIỆN TỬ – VIỄN THÔNG

FLOWSENSE: HỆ THỐNG GIÁM SÁT MẠNG THỤ ĐỘNG DỰA TRÊN THÔNG TIN ĐIỀU KHIỂN TRONG SDN

Họ và tên sinh viên: Trương Đức Duy

Mã số SV: 21207148

NGƯỜI HƯỚNG DẪN KHOA HỌC

TS. Nguyễn Minh Trí

Thành phố Hồ Chí Minh - Tháng 8 năm 2025

XÁC NHẬN HOÀN TẤT CHỈNH SỬA BÁO CÁO TỐT NGHIỆP

FLOWSENSE: HỆ THỐNG GIÁM SÁT MẠNG THỤ ĐỘNG DỰA TRÊN THÔNG TIN ĐIỀU KHIỂN TRONG SDN

Tên tiếng Anh: FLOWSENSE: PASSIVE NETWORK MONITORING SYSTEM BASED ON SDN CONTROL MESSAGE

Sinh viên: Trương Đức Duy

Mã số sinh viên: 21207148

Hội đồng đánh giá họp ngày.... tháng.... năm....

1. Chủ tịch: TS. Nguyễn Minh Trí xác nhận:
2. Thư ký: ThS. Nguyễn Thái Công Nghĩa xác nhận:
3. Ủy viên: ThS. Nguyễn Thị Hồng Hà xác nhận:

Người hướng dẫn: TS. Nguyễn Minh Trí

Xác nhận của người hướng dẫn:

Lời cam kết

Tôi xin cam đoan rằng toàn bộ nội dung, số liệu và kết quả nghiên cứu được trình bày trong đồ án tốt nghiệp này là do chính tôi thực hiện, trung thực và không sao chép từ bất kỳ nguồn nào mà không trích dẫn rõ ràng. Tôi cam kết không sử dụng trái phép các kết quả từ công trình nghiên cứu khác đã được công bố. Mọi tài liệu tham khảo được sử dụng trong quá trình thực hiện đồ án đều đã được trích dẫn và ghi nguồn đầy đủ theo đúng quy định.

Tôi xin chịu trách nhiệm hoàn toàn nếu vi phạm các cam kết nêu trên.

Ký tên

Trương Đức Duy

Lời cảm ơn

Lời đầu tiên, em xin gửi lời cảm ơn chân thành đến quý thầy cô thuộc Bộ môn Viễn Thông – Mạng, Khoa Điện tử – Viễn thông, Trường Đại học Khoa học Tự nhiên – Đại học Quốc gia TP. Hồ Chí Minh. Quá trình học tập tại trường đã giúp em tích lũy được nền tảng kiến thức vững chắc và phát triển tư duy nghiên cứu, là hành trang quý báu để em thực hiện đề tài đồ án tốt nghiệp này. Em xin bày tỏ lòng biết ơn sâu sắc đến thầy hướng dẫn TS. Nguyễn Minh Trí, người đã tận tình chỉ dẫn và định hướng cho em từ những bước đầu tiên cho đến khi hoàn thiện đề tài. Sự kiên nhẫn, nhiệt tình và những góp ý cụ thể của thầy đã giúp em hiểu rõ vấn đề và từng bước cải thiện chất lượng công việc của mình. Những góp ý và trao đổi từ thầy cô và bạn bè trong suốt quá trình học tập đã mở rộng góc nhìn của em đối với lĩnh vực chuyên môn, cũng như tiếp thêm động lực để em hoàn thành tốt hơn. Cuối cùng, em muốn dành lời cảm ơn sâu sắc nhất đến gia đình – những người luôn ở phía sau âm thầm ủng hộ, động viên và tạo điều kiện tốt nhất để em có thể an tâm học tập và theo đuổi mục tiêu của mình.

Em chân thành cảm ơn các thầy cô trong hội đồng và giảng viên phản biện đã dành thời gian góp ý và đưa ra những nhận xét sâu sắc. Những góp ý quý báu ấy giúp em hoàn thiện hơn kiến thức chuyên môn và định hướng rõ ràng hơn trong quá trình nghiên cứu.

Dù đã nỗ lực hoàn thiện đề tài với tinh thần trách nhiệm cao, em vẫn không tránh khỏi những thiếu sót do hạn chế về thời gian và kinh nghiệm. Em rất mong nhận được những góp ý từ quý thầy cô để có thể cải thiện và phát triển tốt hơn trong tương lai.

Xin chân thành cảm ơn!

Trương Đức Duy

Tóm tắt

Trong bối cảnh mạng máy tính hiện đại ngày càng phát triển theo hướng linh hoạt và dễ điều khiển, SDN (Software-Defined Networking) đã trở thành một xu hướng thiết kế hạ tầng mạng mới, tách biệt rõ ràng giữa mặt điều khiển và mặt dữ liệu. Một trong những vấn đề quan trọng cần giải quyết trong SDN là khả năng giám sát mức sử dụng băng thông một cách hiệu quả, chính xác và ít gây ảnh hưởng đến lưu lượng điều khiển. Việc nắm bắt kịp thời tình trạng tải mạng không chỉ giúp điều phối tài nguyên tối ưu mà còn đóng vai trò cốt lõi trong việc phát hiện tắc nghẽn, bất thường hoặc mất cân bằng trong hệ thống. Trong các nghiên cứu trước đây, nhiều giải pháp giám sát chủ động đã được đề xuất. Các phương pháp này thường dựa vào việc gửi truy vấn định kỳ từ controller đến các switch để thu thập thông tin thống kê về lưu lượng mạng. Tuy nhiên, đặc điểm chung của các giải pháp này là tạo ra lượng lớn lưu lượng điều khiển bổ sung, gây tăng độ trễ, tiêu tốn tài nguyên xử lý và gặp khó khăn khi mở rộng hệ thống với số lượng thiết bị và luồng dữ liệu lớn. Xuất phát từ những hạn chế đó, đề tài này triển khai và đánh giá phương pháp FlowSense – một kỹ thuật giám sát thụ động được thiết kế nhằm tận dụng các thông điệp điều khiển vốn có trong mạng như PacketIn và FlowRemoved. FlowSense cho phép ước lượng mức sử dụng băng thông trên từng liên kết mạng mà không cần gửi thêm yêu cầu truy vấn nào từ controller, từ đó loại bỏ hoàn toàn chi phí giám sát trên mặt điều khiển mà vẫn duy trì độ chính xác cần thiết trong việc giám sát lưu lượng. Trong phạm vi đề tài, hệ thống được xây dựng bằng cách sử dụng Mininet để mô phỏng môi trường mạng SDN, kết hợp với Ryu Controller để hiện thực thuật toán FlowSense bằng ngôn ngữ Python. Mô hình thử nghiệm bao gồm các host phát sinh tải

UDP với cấu hình linh hoạt để tạo ra các kịch bản thay đổi mức sử dụng mạng đa dạng. Song song đó, một phương pháp chủ động dựa trên truy vấn định kỳ cũng được triển khai như một hệ thống đối chứng nhằm so sánh hiệu quả. Các kết quả thực nghiệm cho thấy FlowSense có thể tái hiện khá chính xác các thay đổi băng thông trong mạng, với sai số nhỏ so với phương pháp chủ động, đồng thời hoàn toàn không phát sinh thêm lưu lượng điều khiển. Ngoài ra, đề tài cũng khảo sát khả năng phản hồi theo thời gian (granularity) của FlowSense và nhận thấy hệ thống có thể cập nhật phần lớn thông tin băng thông chỉ trong vòng vài giây sau khi flow kết thúc, cho thấy tiềm năng ứng dụng trong các hệ thống yêu cầu giám sát gần thời gian thực.

Tháng 8 năm 2025

Trương Đức Duy

Abstract

As modern computer networks continue to evolve towards greater flexibility and centralized control, Software-Defined Networking (SDN) has emerged as a prominent architecture that clearly separates the control plane from the data plane. One of the critical challenges in SDN is the ability to monitor bandwidth usage accurately and efficiently without introducing significant overhead on the control channel. Timely visibility into network load conditions is essential not only for optimal resource allocation but also for detecting congestion, anomalies, or imbalances in the system. In previous studies, various active monitoring solutions have been proposed. These methods typically rely on periodic polling from the controller to the switches to retrieve traffic statistics. However, such approaches tend to generate considerable control traffic, which increases latency, consumes processing resources, and poses scalability issues in networks with a large number of switches and flows. To address these limitations, this project implements and evaluates FlowSense — a passive monitoring technique designed to leverage existing control messages such as PacketIn and FlowRemoved. By utilizing these inherent messages, FlowSense estimates bandwidth usage on network links without requiring additional polling, thereby eliminating monitoring overhead while still maintaining acceptable accuracy. In this work, the network architecture is emulated using Mininet, and the FlowSense algorithm is implemented on the Ryu SDN controller using Python. The testbed consists of multiple hosts generating configurable UDP traffic, simulating various usage scenarios. In parallel, an active monitoring approach based on periodic FlowStats queries is also deployed to serve as a baseline for comparison. Experimental results show that FlowSense

effectively captures bandwidth fluctuations with a small margin of error compared to the active method, while completely avoiding extra control-plane traffic. Furthermore, the system demonstrates promising temporal granularity, being able to update most of the bandwidth usage information within a few seconds after each flow ends. These findings highlight FlowSense's potential for deployment in near real-time traffic monitoring applications.

August, 2025

Truong Duc Duy

Mục lục

Danh sách hình vẽ

Danh sách chữ viết tắt

API Application Programming Interface

IP Internet Protocol

MAC Media Access Control

QoS Quality of Service

SDN Software-Defined Networking

UDP User Datagram Protocol

UT Utilization Table

Chương 1

TỔNG QUAN

1.1 Giới thiệu đề tài

Trong bối cảnh các hệ thống mạng ngày càng phát triển theo hướng mềm dẻo, linh hoạt và có thể lập trình được, kiến trúc mạng điều khiển bằng phần mềm SDN (Software-Defined Networking) đang dần trở thành một xu thế chủ đạo. Với khả năng tách biệt rõ ràng giữa mặt điều khiển và mặt dữ liệu, SDN cho phép controller tập trung toàn bộ logic điều khiển mạng và tương tác trực tiếp với các thiết bị thông qua các giao thức như OpenFlow.

Trong các hệ thống dựa trên flow, việc đảm bảo chất lượng dịch vụ (QoS - Quality of Service) theo thời gian thực cho các yêu cầu khắt khe – chẳng hạn như ràng buộc về thời hạn trễ [1], khôi phục lỗi nhanh [2] hoặc truyền tải dữ liệu lớn với độ tin cậy cao [3] – đòi hỏi mạng phải luôn theo dõi và điều chỉnh lưu lượng một cách chính xác và kịp thời. Điều này đặt ra yêu cầu cấp thiết về việc giám sát hiệu năng sử dụng mạng (network utilization monitoring) một cách chính xác, nhạy với biến động nhưng vẫn phải tiêu tốn ít tài nguyên điều khiển.

Tuy nhiên, các phương pháp giám sát truyền thống như SNMP polling hay các công cụ thu thập và phân tích như NetFlow, SPAN, tcpdump... thường đòi hỏi triển khai phần cứng hoặc phần mềm bổ sung và gây ra độ trễ cũng như tiêu tốn đáng kể tài nguyên. Các

giải pháp sử dụng controller để polling trực tiếp các switch tuy loại bỏ được nhu cầu cài đặt thêm thiết bị, nhưng vẫn tạo ra overhead lớn và ảnh hưởng đến độ trễ, nhất là trong các hệ thống có số lượng flow lớn và thay đổi liên tục [4].

1.2 Mục tiêu đề tài

Mục tiêu của đề tài là hiện thực và đánh giá phương pháp giám sát mức sử dụng băng thông trong môi trường mạng lập trình SDN với độ chính xác cao, độ trễ thấp và không phát sinh thêm chi phí đo lường. Giải pháp được áp dụng là FlowSense – một kỹ thuật giám sát thụ động, khai thác các đặc điểm vốn có trong hệ thống SDN như việc tách biệt mặt điều khiển và mặt dữ liệu, cho phép phân tích các thông điệp điều khiển sẵn có (PacketIn và FlowRemoved) thay vì phải gửi thêm truy vấn từ controller đến switch. Nhờ đó, FlowSense có thể suy ra mức sử dụng băng thông mà không cần cài đặt thêm thiết bị hay tạo thêm lưu lượng điều khiển.

Trong phạm vi đề tài, thuật toán FlowSense được triển khai trên nền tảng Ryu Controller bằng ngôn ngữ Python và được thử nghiệm trên môi trường mô phỏng Mininet. Mô hình được xây dựng với các kịch bản tạo tải UDP (User Datagram Protocol) nhằm đánh giá khả năng phản ánh của thuật toán trước các thay đổi về mức sử dụng mạng. Song song đó, một hệ thống giám sát chủ động sử dụng truy vấn FlowStats định kỳ cũng được triển khai nhằm so sánh hiệu quả giữa hai phương pháp.

Việc đánh giá hiệu quả của FlowSense được thực hiện theo hai hướng chính. Thứ nhất là độ chính xác của phương pháp trong việc phản ánh đúng mức sử dụng băng thông thông qua so sánh kết quả với phương pháp polling định kỳ. Thứ hai là khả năng phản hồi nhanh với thay đổi lưu lượng, hay còn gọi là granularity – tức mức độ chi tiết và kịp thời trong việc cập nhật dữ liệu giám sát. Để đảm bảo kết quả phản ánh sát thực tế, các luồng lưu lượng trong thí nghiệm được thiết kế đa dạng về thời lượng, tốc độ và đặc điểm khởi phát.

1.3 Cấu trúc đồ án

Đồ án có bố cục bao gồm 5 chương:

- **Chương 1 - Tổng quan**: Giới thiệu tổng quan về đề tài, mục tiêu của đề tài và cấu trúc của đồ án.
- **Chương 2 - Cơ sở lý thuyết và phương pháp** : Trình bày các kiến thức nền tảng về SDN, OpenFlow, các phương pháp giám sát mạng và phương pháp giám sát mạng thụ động dựa trên thông tin điều khiển - FlowSense.
- **Chương 3 - Mô phỏng**: Triển khai mô phỏng, xây dựng mô hình đánh giá phương pháp FlowSense.
- **Chương 4 - Kết quả mô phỏng**: Đánh giá kết quả của mô hình và thảo luận về kết quả thu được.
- **Chương 5 - Kết luận và hướng phát triển**: Đưa ra kết luận và tính tiềm năng cũng như hướng phát triển tương lai của đề tài này.

Chương 2

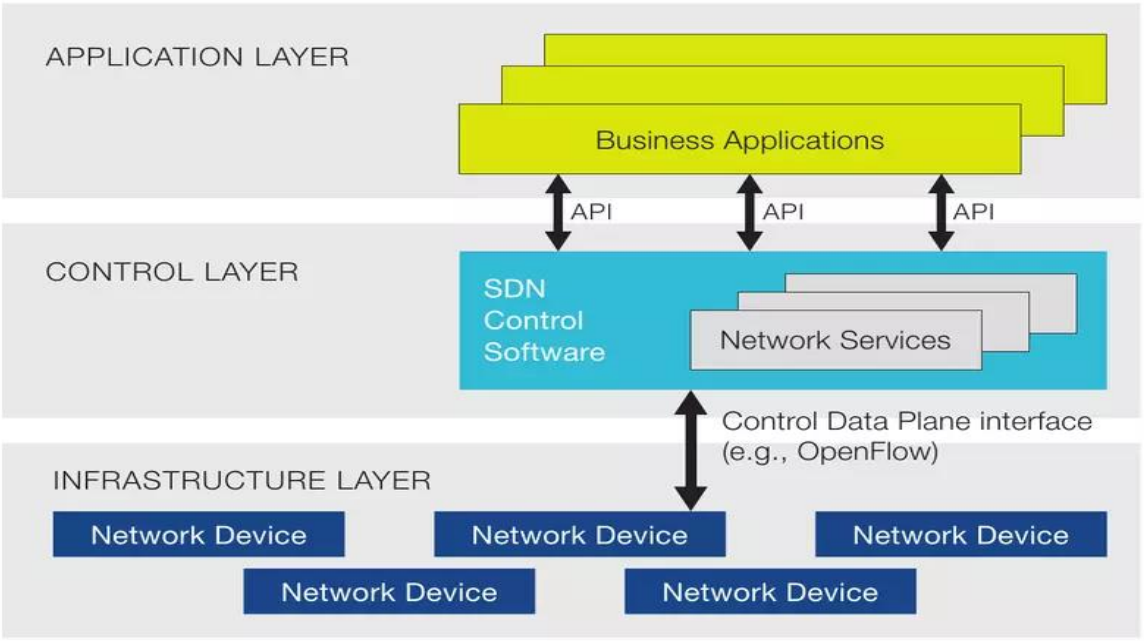
CƠ SỞ LÍ THUYẾT VÀ PHƯƠNG PHÁP

2.1 Kiến trúc SDN

SDN [5] là một kiến trúc mạng hiện đại, trong đó hai mặt chức năng chính của hệ thống mạng là điều khiển (control plane) và dữ liệu (data plane) được tách biệt rõ ràng. Thay vì để mỗi thiết bị mạng tự đưa ra quyết định xử lý gói tin, SDN sử dụng một thực thể điều khiển tập trung gọi là controller để điều phối toàn bộ hoạt động của mạng. Các thiết bị chuyển tiếp như switch chỉ thực hiện chức năng chuyển tiếp theo lệnh được lập trình từ controller.

Việc tập trung hóa điều khiển giúp mạng trở nên linh hoạt hơn, dễ mở rộng và thuận tiện trong việc tự động hóa các tác vụ cấu hình, giám sát và xử lý sự cố. Controller có thể được lập trình bằng các ngôn ngữ phổ biến như Python, đồng thời cho phép triển khai nhiều thuật toán giám sát và tối ưu hóa tùy biến theo nhu cầu.

Các bộ điều khiển tương tác với các thiết bị mạng vật lý thông qua một giao thức chuẩn OpenFlow. Hình 2.1 cho thấy kiến trúc của SDN gồm 3 lớp riêng biệt: lớp ứng dụng, lớp điều khiển, và lớp cơ sở hạ tầng (lớp chuyển tiếp).



Hình 2.1: Kiến trúc SDN.

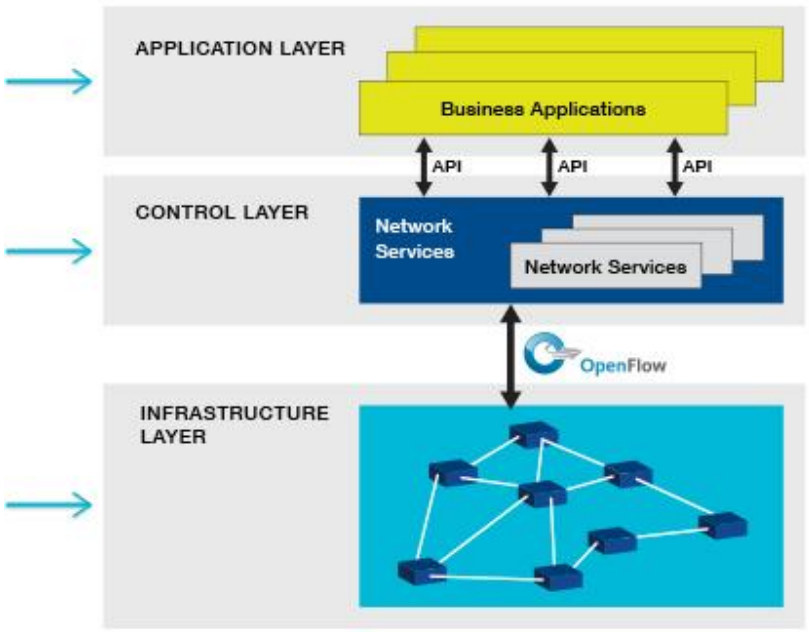
Lớp ứng dụng: Là nơi triển khai các ứng dụng kinh doanh, dịch vụ mạng hoặc chính sách vận hành. Lớp này giao tiếp với lớp điều khiển thông qua các API (giao diện lập trình ứng dụng), cho phép triển khai các tính năng như lập trình lại mạng, điều chỉnh thông số định tuyến, tối ưu băng thông, giảm độ trễ... mà không cần can thiệp trực tiếp vào lớp thiết bị.

Lớp điều khiển: Đây là thành phần trung tâm trong mô hình SDN, nơi tập hợp các bộ điều khiển có nhiệm vụ điều phối và cấu hình hoạt động của mạng dựa trên yêu cầu từ lớp ứng dụng. Các bộ điều khiển này thường là phần mềm có khả năng lập trình linh hoạt, cho phép quản lý tập trung và tự động hóa quá trình vận hành mạng.

Lớp cơ sở hạ tầng: Bao gồm các thiết bị mạng vật lý hoặc ảo (chẳng hạn như switch, router...) có nhiệm vụ chuyển tiếp gói tin dựa trên các chỉ dẫn được gửi từ lớp điều khiển. Các thiết bị này không tự quyết định hành vi định tuyến, thay vào đó hoạt động dưới sự điều phối tập trung của bộ điều khiển. Cấu trúc này giúp mạng trở nên linh hoạt và dễ dàng thay đổi theo nhu cầu sử dụng.

2.2 Giao thức OpenFlow

OpenFlow [6, 7] là một trong những giao thức đầu tiên và quan trọng nhất được sử dụng trong các hệ thống mạng lập trình được (SDN). Giao thức này quy định cách thức mà controller trung tâm giao tiếp với các thiết bị chuyển tiếp như switch trong mạng. Mỗi thiết bị OpenFlow duy trì một bảng dòng (flow table), trong đó các flow entry được định nghĩa để xử lý các gói tin đi qua.



Hình 2.2: Giao thức OpenFlow trong SDN

Mỗi entry trong bảng dòng bao gồm ba thành phần chính. Thứ nhất là trường điều kiện (match fields), cho phép xác định những gói tin nào sẽ khớp với rule này, ví dụ dựa trên địa chỉ MAC (Media Access Control) , IP (Internet Protocol) , cổng hoặc giao thức. Thứ hai là các bộ đếm (counters), dùng để ghi nhận thống kê như số lượng byte, số gói tin đã xử lý, và thời gian tồn tại của flow. Cuối cùng là hành động (instructions/actions), xác định cách mà gói tin khớp sẽ được xử lý, chẳng hạn như chuyển tiếp tới một cổng cụ thể hoặc gửi về controller. Tiếp theo là các bước giao tiếp giữa controller với các thiết bị mạng:

- **Flow Arrival**: Khi một gói tin không khớp với bất kỳ rule nào trong bảng dòng, switch sẽ tạo một thông điệp PacketIn và gửi về controller để xin quyết định xử lý. Nếu controller thấy cần thiết, nó sẽ cài đặt một flow entry mới vào switch để xử lý các gói tin tương tự trong tương lai.:
- **Flow Completion**: Khi một flow không còn hoạt động, hoặc hết thời gian sống do timeout, switch sẽ gửi thông điệp FlowRemoved về controller. Thông điệp này chứa thông tin thống kê đầy đủ về flow đã bị xóa, bao gồm số byte, số gói tin và thời gian hoạt động. Đây là nguồn dữ liệu quan trọng để các hệ thống giám sát mạng khai thác, đặc biệt trong các phương pháp giám sát thụ động.

2.3 Giám sát mạng bằng OpenFlow

Giao thức OpenFlow cung cấp cơ chế để truy vấn các switch nhằm thu thập số liệu về số byte hoặc số gói tin đã truyền, từ đó hỗ trợ các hệ thống giám sát lưu lượng mạng. Các nghiên cứu trước đây như OpenTM [8] và Jose et al [9] đã tận dụng khả năng này bằng cách định kỳ gửi truy vấn (polling) tới các switch để đo mức sử dụng mạng. Tuy nhiên, phương pháp polling thường phát sinh độ trễ đo đạc và tạo ra lưu lượng điều khiển không nhỏ, đặc biệt khi triển khai trên quy mô lớn hoặc trong các mạng có nhiều luồng đồng thời.

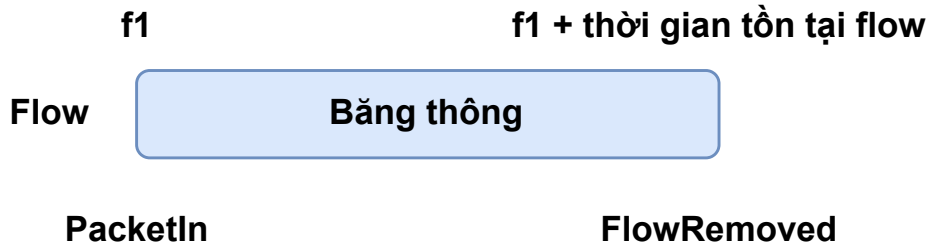
Chẳng hạn, OpenTM [8] thực hiện đo ma trận lưu lượng toàn mạng bằng cách polling lần lượt từng switch trên đường đi của mỗi flow và tổng hợp kết quả, điều này làm tăng đáng kể tải xử lý tại các switch biên và gây khó khăn trong việc mở rộng hệ thống. Trong khi đó, phương pháp của Jose et al. [9] đề xuất cải tiến bằng cách áp dụng chiến lược polling thích ứng – ưu tiên các flow có khả năng cao là "heavy hitters". Dù giúp cải thiện độ chính xác, giải pháp này vẫn yêu cầu điều phối các phiên đo một cách cẩn trọng để tránh làm tăng chi phí giám sát và gây ảnh hưởng đến độ chính xác khi có quá nhiều truy vấn đồng thời.

FlowSense [10] ra đời như một giải pháp khắc phục những hạn chế đó bằng cách

loại bỏ hoàn toàn nhu cầu polling chủ động. Thay vì gửi truy vấn, FlowSense tận dụng luồng thông điệp điều khiển vốn đã được switch gửi về controller như PacketIn và FlowRemoved. Nhờ đó, hệ thống có thể ước lượng chính xác mức sử dụng băng thông mà không tạo thêm bất kỳ lưu lượng đo đạc nào, tức zero measurement cost. Đây là một bước tiến lớn trong việc giám sát mạng thụ động, đặc biệt phù hợp với các môi trường SDN hiện đại, nơi tối ưu hóa tài nguyên điều khiển và giảm độ trễ là rất quan trọng.

2.4 Phương pháp FlowSense

FlowSense được thiết kế để giám sát mức sử dụng băng thông trong mạng SDN mà không tạo thêm lưu lượng điều khiển. Khác với các phương pháp giám sát chủ động vốn cần thực hiện polling định kỳ, FlowSense tận dụng trực tiếp các thông điệp PacketIn và FlowRemoved do switch sinh ra trong quá trình xử lý lưu lượng.



Hình 2.3: Phương pháp Flowsense.

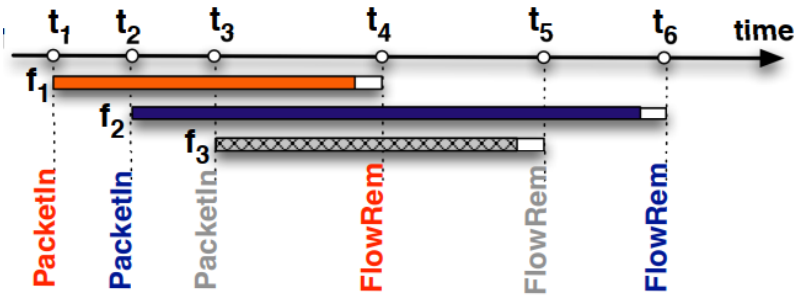
Khi một flow mới khởi phát và không khớp với rule nào trong bảng dòng, switch gửi thông điệp PacketIn đến controller. Sự kiện này đánh dấu thời điểm bắt đầu của flow. Sau một khoảng thời gian hoạt động, khi rule bị xóa khỏi switch do hết hạn, sự kiện FlowRemoved sẽ được gửi về. Dựa trên nội dung của thông điệp FlowRemoved, thuật toán có thể trích xuất ba thông tin quan trọng: thời lượng tồn tại của flow trong bảng dòng, tổng số byte mà flow đã truyền và cổng vào nơi lưu lượng đi vào switch. Các thuộc tính này kết hợp với thời điểm bắt đầu từ PacketIn cho phép tính toán được mức đóng góp của mỗi flow vào lưu lượng trên liên kết tương ứng.

Mỗi khi nhận được sự kiện FlowRemoved, hệ thống tạo ra một checkpoint cho liên

kết nơi flow kết thúc. Checkpoint này đánh dấu thời điểm sử dụng băng thông được cập nhật. Tại mỗi checkpoint, mức sử dụng băng thông được ước lượng bằng cách tổng hợp lưu lượng của các flow đã kết thúc trong khoảng thời gian vừa qua.

2.5 Tính băng thông trong FlowSense

Hình 2.4 minh họa cụ thể cách thức hoạt động của FlowSense trong việc ghi nhận mức sử dụng băng thông: ví dụ đơn giản với ba flow lần lượt là f1, f2 và f3, bắt đầu tại các thời điểm t1, t2 và t3, và kết thúc tại t4, t6 và t5. FlowSense xác định thời điểm bắt đầu và kết thúc của mỗi flow dựa trên các thông điệp PacketIn và FlowRemoved tương ứng.



Hình 2.4: Cách tính băng thông của phương pháp Flowsense.

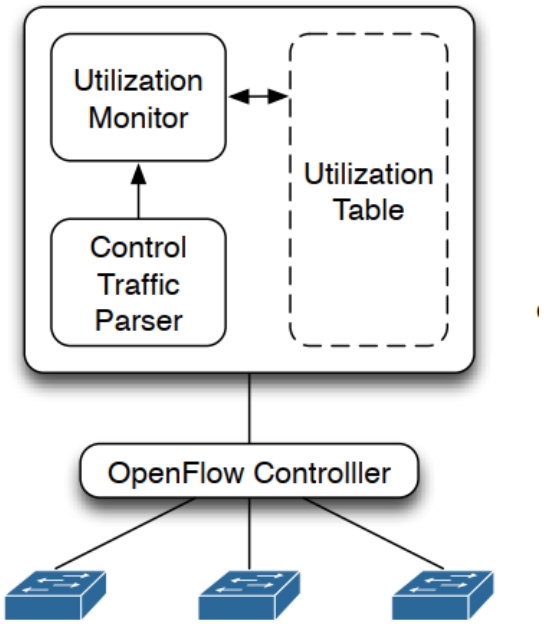
Giả sử các flow này có mức sử dụng băng thông lần lượt là 10, 20 và 40 Mbps. Khi FlowRemoved của f1 đến tại thời điểm t4, hệ thống ghi nhận mức sử dụng của f1 và tạo một checkpoint tại t4 với giá trị 10 Mbps. Tiếp theo, khi f3 kết thúc tại t5, FlowSense cộng thêm 40 Mbps vào checkpoint tại t4, do f3 vẫn còn hoạt động tại thời điểm đó, đồng thời tạo một checkpoint mới tại t5 với giá trị 40 Mbps. Cuối cùng, khi f2 kết thúc tại t6, mức sử dụng 20 Mbps của f2 được cộng vào cả hai checkpoint trước đó (t4 và t5) vì f2 cũng đang hoạt động tại các thời điểm đó, và một checkpoint mới được tạo tại t6

với giá trị 20 Mbps.

Kết quả cuối cùng là: checkpoint tại t4 ghi nhận tổng mức sử dụng là 70 Mbps (f1 + f3 + f2), tại t5 là 60 Mbps (f3 + f2), và tại t6 là 20 Mbps (f2). Ví dụ này cho thấy FlowSense có thể tổng hợp băng thông từ nhiều flow bằng cách cộng dồn đóng góp của các flow vào các checkpoint mà chúng còn hoạt động, từ đó tái tạo lại bức tranh sử dụng liên kết theo thời gian một cách hiệu quả.

2.6 Kiến trúc FlowSense

Để hiện thực hóa nguyên lý hoạt động đã trình bày ở mục trước. Hình 2.5 sau đây minh họa kiến trúc tổng quát của hệ thống, trong đó thể hiện rõ quá trình xử lý thông điệp điều khiển và cập nhật trạng thái sử dụng băng thông.



Hình 2.5: Hệ thống FlowSense.

Hệ thống FlowSense bao gồm hai thành phần chức năng chính: bộ phân tích lưu lượng điều khiển (control traffic parser) và bộ giám sát mức sử dụng (utilization monitor).

- **Control traffic parser**: có nhiệm vụ thu thập và phân tích các thông điệp từ control

plane, đặc biệt là PacketIn và FlowRemoved. Khi một flow mới khởi phát và chưa có rule khớp trên switch, sự kiện PacketIn được gửi về controller. Ngược lại, khi flow kết thúc do timeout hoặc bị xóa, switch phát sinh sự kiện FlowRemoved kèm theo các thống kê về flow đã xử lý. Parser sẽ trích xuất các thông tin quan trọng từ hai loại thông điệp này như thời gian, byte count, cổng vào, và thông tin định danh flow.

- **Utilization Monitor**: chịu trách nhiệm cập nhật và duy trì một bảng gọi là Bảng sử dụng băng thông (Utilization Table (UT)). Tại đây, hệ thống lưu trữ các checkpoint – các mốc thời gian tại đó mức sử dụng liên kết được ghi nhận, cùng với thông tin về số lượng flow đang hoạt động tại thời điểm đó và mức sử dụng ban đầu do flow vừa kết thúc đóng góp.

Trong bài báo gốc, hai thành phần này được tích hợp trực tiếp vào controller dưới dạng một plugin mở rộng của FlowSense. Tuy nhiên, trong phạm vi đề tài này sẽ không phát triển plugin tích hợp, thay vào đó hiện thực từng thành phần dưới dạng các script Python độc lập. Cách tiếp cận này vẫn đảm bảo thực hiện đầy đủ các chức năng của từng khối, đồng thời giúp quá trình phát triển, kiểm thử và điều chỉnh thuật toán trở nên linh hoạt hơn.

2.7 Thuật toán giám sát băng thông

Dựa trên nguyên lý hoạt động và kiến trúc hệ thống đã trình bày, FlowSense hiện thực quá trình giám sát mức sử dụng liên kết thông qua một thuật toán theo dõi các flow đang hoạt động và tạo checkpoint mỗi khi một flow kết thúc. Mục tiêu của thuật toán là ước lượng chính xác mức băng thông đã sử dụng tại các mốc thời gian cụ thể mà không cần sử dụng cơ chế polling chủ động.

Hình 2.6 thể hiện các bước xử lý theo trình tự logic mà FlowSense thực hiện cho mỗi sự kiện PacketIn và FlowRemoved.

Algorithm 1 Pseudocode of FlowSense's utilization monitor.

```
1: procedure UTILIZATIONMONITOR(Utilization Table UT, Packet p)
2:     Active_List ← set of p.in_port's active flows
3:     if p is a PacketIn packet then
4:         if p's flow ∉ Active_List then
5:             Flow active_flow
6:             active_flow.flow ← p.flow
7:             active_flow.time ← p.time
8:             Add active_flow to Active_List
9:         end if
10:    else if p is a FlowRemoved packet then
11:        flow ← matching flow from A
12:        Remove flow from Active_List
13:        Checkpoint chkpt
14:        chkpt.time ← p.time
15:        if p was from soft timeout then
16:            chkpt.time ← chkpt.time − p.soft_timeout
17:        end if
18:        chkpt.active ← |Active_List|
19:        chkpt.util ← p.byte_count/p.flow_length
20:        for active c in UT do
21:            if c.time is between flow.time and chkpt.time then
22:                c.active ← c.active − 1
23:                c.util ← c.util + chkpt.util
24:            end if
25:            if c.active = 0 then
26:                Declare c final and inactive
27:            end if
28:        end for
29:        Insert chkpt into UT
30:    end if
31: end procedure
```

Hình 2.6: Thuật toán giám sát băng thông.

Thuật toán hoạt động bằng cách lần lượt xử lý các sự kiện điều khiển PacketIn và FlowRemoved. Khi nhận được một thông điệp PacketIn, hệ thống khởi tạo một flow mới, ghi nhận thời điểm bắt đầu của flow và thêm vào danh sách các flow đang hoạt động (Active List) tương ứng với cổng đầu vào.

Ngược lại, khi một flow kết thúc và phát sinh thông điệp FlowRemoved, hệ thống tiến hành xoá flow đó khỏi Active List để đảm bảo không bị cộng dồn trùng lặp. Sau đó, một checkpoint mới được tạo ra với thời điểm đánh dấu là thời gian hiện tại. Nếu flow bị xoá do idle timeout, checkpoint sẽ được điều chỉnh bằng cách lùi lại đúng bằng giá trị timeout để phản ánh chính xác thời điểm ngừng truyền thực tế. Mức sử dụng băng thông của flow vừa kết thúc được tính toán theo công thức:

$$\text{utilization} = \frac{\text{byte\_count} \times 8}{\text{duration}} \quad \text{(Mbps)} \tag{2.1}$$

Tại thời điểm checkpoint, hệ thống cũng ghi lại số lượng flow đang hoạt động hiện tại trên liên kết, gọi là chkpt.active. Tiếp theo, hệ thống duyệt qua toàn bộ các checkpoint đã tồn tại trong bảng sử dụng băng thông (Utilization Table (UT)) trên cùng một cổng đầu vào. Nếu một checkpoint trước đó có thời gian nằm trong khoảng hoạt động của flow vừa kết thúc (tức nằm giữa thời điểm bắt đầu và kết thúc của flow), checkpoint đó sẽ được cập nhật bằng cách cộng thêm mức sử dụng chkpt.util và giảm số lượng flow đang hoạt động c.active xuống 1. Khi số flow còn hoạt động tại một checkpoint giảm về 0, checkpoint đó sẽ được đánh dấu là hoàn tất (finalized) và không còn thay đổi về sau.

Cuối cùng, checkpoint mới chkpt sẽ được thêm vào bảng UT để phục vụ cho các lần xử lý tiếp theo. Quá trình này được thực hiện liên tục, cho phép hệ thống từng bước tái dựng lại biểu đồ sử dụng băng thông của liên kết theo thời gian, mà không cần tạo thêm bất kỳ gói tin đo lường nào.

Chương 3

MÔ PHỎNG

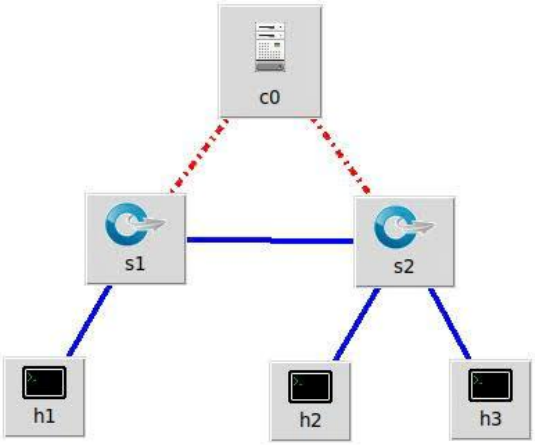
3.1 Giới thiệu

Chương này trình bày chi tiết quá trình triển khai hệ thống giám sát mạng dựa trên thuật toán FlowSense. Quá trình triển khai được thực hiện trong môi trường mô phỏng Mininet [11] – một công cụ mạnh mẽ cho phép xây dựng mạng ảo trên máy tính cá nhân với các thành phần như host, switch và controller được ảo hóa ở mức hệ điều hành. Nhờ đó, Mininet hỗ trợ mô phỏng hành vi mạng thực một cách hiệu quả và được sử dụng rộng rãi trong nghiên cứu, giảng dạy và kiểm thử các giải pháp mạng SDN

Trong mô hình này, bộ điều khiển mạng Ryu được lựa chọn làm nền tảng triển khai thuật toán FlowSense. Ryu [12] là một framework mã nguồn mở được phát triển bằng ngôn ngữ Python, hỗ trợ đầy đủ các phiên bản của giao thức OpenFlow và cung cấp các API (Application Programming Interface) linh hoạt, dễ sử dụng. Đây là một công cụ lý tưởng để phát triển các ứng dụng điều khiển mạng như giám sát, điều phối và tối ưu hóa lưu lượng trong môi trường SDN.

Dữ liệu được thu thập từ các sự kiện mạng trong suốt quá trình mô phỏng và được xử lý, phân tích bằng ngôn ngữ lập trình Python. Các kết quả phân tích sẽ được sử dụng để đánh giá hai tiêu chí chính của hệ thống: độ chính xác trong việc phản ánh mức sử dụng băng thông và khả năng phản hồi kịp thời (granularity) đối với các thay đổi của mạng.

3.2 Mô hình mạng đánh giá độ chính xác của FlowSense với Polling



Hình 3.1: Mô hình mạng.

Mô hình mạng khảo sát độ chính xác của FlowSense so với Polling trong đồ án này được thiết kế dựa trên khảo sát gốc trong bài báo FlowSense [10]. Để đảm bảo điều kiện tương đương trong việc so sánh hai phương pháp giám sát, mô hình đã được tái tạo lại dưới dạng đơn giản hơn, phù hợp với phạm vi thực hiện trong môi trường Mininet.

Mô hình mạng cơ bản gồm hai switch kết nối nối tiếp (s1 → s2). Host h1 được nối với s1, trong khi các host h2 và h3 được nối với s2. Trong quá trình đánh giá:

- Host h1 đóng vai trò là nguồn phát lưu lượng chính, mô phỏng các ứng dụng tạo tải trên mạng.
- Host h2 là đích đo băng thông, nơi lưu lượng được tập trung ghi nhận để đánh giá hiệu suất giám sát của thuật toán FlowSense.
- Host h3 nhận một luồng nền cố định (background traffic), nhằm mô phỏng môi trường có nhiễu, phản ánh điều kiện mạng thực tế với sự đồng thời của nhiều flow.

3.3 Thực hiện đánh giá độ chính xác

Thuật toán FlowSense được hiện thực bằng Python trên nền tảng Ryu Controller, gồm hai giai đoạn chính: ghi nhận sự kiện mạng thông qua các thông điệp điều khiển (PacketIn và FlowRemoved) và xử lý theo logic của thuật toán giám sát băng thông (Hình 2.6) để tái tạo lại quá trình truyền tải và ước lượng mức sử dụng băng thông.

3.3.1 Giai đoạn 1: Ghi nhận sự kiện mạng

Khi một flow mới bắt đầu, switch không có rule phù hợp sẽ gửi một gói PacketIn về controller. Tại đây:

- Controller ghi nhận thời điểm bắt đầu của flow dựa vào timestamp kèm theo, và ánh xạ vào một bảng flow starts.
- Controller đồng thời cài đặt rule mới với idle_timeout = 3 để sau thời gian này sẽ nhận được FlowRemoved nếu flow không còn hoạt động.
- Dữ liệu được ghi vào file log dưới dạng sự kiện PacketIn.

```
match = parser.OFPMatch(in_port=in_port, eth_src=eth.src,
 ↪  eth_dst=eth.dst)
actions = [parser.OFPActionOutput(ofproto.OFPP_FLOOD)]
inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
 ↪  actions)]
mod = parser.OFPFlowMod(
    datapath=datapath,
    priority=1,
    match=match,
    instructions=inst,
    idle_timeout=3,
    flags=ofproto.OFPFF_SEND_FLOW_REM,
    cookie=in_port
)
```

```
# Tạo flow_id → timestamp mapping cho PacketIn
flow_starts = {
    row['flow_id']: row['timestamp']
    for _, row in pktin_df.iterrows()
    if pd.notna(row['flow_id'])
}
```

Khi một flow bị xóa khỏi bảng định tuyến do hết timeout, switch sẽ gửi thông điệp FlowRemoved về controller. Trong đoạn mã dưới, hệ thống trích xuất địa chỉ MAC nguồn/đích từ match field, sau đó kết hợp với in port để tạo flow id dùng làm khóa lưu trữ. Dữ liệu thu được sẽ được ghi vào Exel để phục vụ phân tích.

```
for field in msg.match.fields:
    if field.header == 3:
        eth_src = field.value
    elif field.header == 4:
        eth_dst = field.value

timestamp = time.time()
if self.start_time is None:
    return
elapsed = timestamp - self.start_time

flow_id = generate_flow_id(in_port, eth_src, eth_dst)

self._log_event([
    f"{elapsed:.6f}", "FlowRemoved", dpid, in_port, eth_src, eth_dst,
    msg.byte_count, msg.duration_sec, msg.idle_timeout, flow_id
])

self.logger.info(f"[FlowRemoved] dpid={dpid}, in_port={in_port},
 ↪  flow_id={flow_id}, bytes={msg.byte_count},
 ↪  duration={msg.duration_sec}s")
```

3.3.2 Thực hiện thuật toán giám sát băng thông

Thuật toán giám sát băng thông (Hình 2.6) là thành phần cốt lõi trong giải pháp FlowSense [10] – một phương pháp giám sát thụ động được thiết kế nhằm loại bỏ hoàn toàn nhu cầu thăm dò (polling) chủ động trong mạng SDN. Ý tưởng trung tâm của thuật toán là khi một flow mới xuất hiện, switch sẽ gửi một gói PacketIn về controller do chưa có rule phù hợp, từ đó đánh dấu thời điểm bắt đầu của flow. Sau đó, controller sẽ cài đặt rule kèm theo tham số idle_timeout. Khi flow kết thúc (do timeout hoặc không còn hoạt động), switch sẽ gửi thông điệp FlowRemoved chứa toàn bộ thống kê về lưu lượng. Bằng cách ghép nối hai sự kiện này theo cùng một flow_id, hệ thống có thể tính toán chính xác lượng byte đã truyền và thời gian tồn tại của mỗi flow. Thuật toán này chỉ cần lắng nghe luồng sự kiện tự nhiên của mạng. Nhờ đó, hệ thống không những giảm được chi phí điều khiển mà còn hạn chế việc làm chậm hệ thống do quá tải.

Dựa trên nguyên lý đó, các bước hiện thực thuật toán trong hệ thống được triển khai bao gồm: lọc và chuẩn hóa dữ liệu, xác định thời gian thực tế của flow, tính toán thông lượng, tạo checkpoint và cập nhật danh sách các flow đang hoạt động. Dưới đây là phần trình bày chi tiết từng bước thực hiện thuật toán:

Bước 1: Lọc và chuẩn hóa dữ liệu

Trong mô hình triển khai, luồng lưu lượng chính được truyền từ host h1 đến các host h2 hoặc h3 thông qua hai switch nối tiếp là s1 và s2. Tuy nhiên, để đo chính xác mức sử dụng băng thông trên liên kết giữa s1 $\rightarrow$ s2 – nơi bị ảnh hưởng bởi lưu lượng thực nghiệm – chỉ các sự kiện xảy ra trên switch s2 (DPID = 2) mới được xem là hợp lệ.

Do đó, trong quá trình xử lý log, tất cả các bản ghi từ sự kiện FlowRemoved hoặc FlowStatsReply đều được lọc theo điều kiện dpid == 2. Việc này giúp loại bỏ các flow nội bộ hoặc nhiễu đo được từ switch s1, vốn không phản ánh chính xác mức độ tiêu thụ băng thông tại điểm giám sát mong muốn.

```
# Tách riêng PacketIn và FlowRemoved
pktin_df = df[df['event'] == 'PacketIn']
flowrm_df = df[(df['event'] == 'FlowRemoved') & (df['dpid'] == 2)]
flowrm_df = flowrm_df[(flowrm_df['byte_count'] > 0) &
 ↪  (flowrm_df['duration'] > 0)]

# Tạo flow_id → timestamp mapping cho PacketIn
flow_starts = {
    row['flow_id']: row['timestamp']
    for _, row in pktin_df.iterrows()
    if pd.notna(row['flow_id'])
}
```

Đoạn mã trên thực hiện phân tích hai loại sự kiện: Packetin và FlowRemoved, chỉ giữ lại các dòng FlowRemoved từ switch s2 (DPID=2) vì đó là nơi đo băng thông trên liên kết chính, lọc các flow không hợp lệ (có byte count hoặc duration = 0). Tiếp theo tạo một flow starts là ánh xạ từ flow id sang timestamp của sự kiện PacketIn - đây là thời điểm khởi đầu của flow, sẽ dùng về sau để tính thời lượng flow.

Bước 2: Xác định thời gian bắt đầu và kết thúc thực tế của flow

```
# Nếu timeout là do idle_timeout thì lùi thời gian kết thúc
if abs(duration - soft_timeout) < 0.5:
    end -= soft_timeout

start = flow_starts.get(flow_id, end - duration)
```

Với thời gian bắt đầu của flow, nếu tìm được flow id từ PacketIn thì dùng timestamp tương ứng, nếu không thì tính ngược từ end – duration. Với thời gian kết thúc của flow, nếu duration bằng với soft_timeout, tức flow không truyền dữ liệu, thì cần lùi thời điểm end lại để chỉ lấy thời gian thực sự có truyền.

Bước 3: Tính toán mức sử dụng băng thông

Sử dụng công thức thông lượng trung bình = (số bit) / (thời gian tồn tại), đơn vị tính theo Mbps.

```
util = (row['byte_count'] * 8) / ((end - start) * 1e6)   # Mbps
```

Bước 4: Tạo checkpoint

```
# Tính số flow đang active tại thời điểm này
active_now = sum(1 for f in active_list if f['start'] <= chkpt_time
 ↪  <= f['end'])

# Tạo checkpoint mới
checkpoints.append({'time': chkpt_time, 'util': chkpt_util,
 ↪  'active': active_now})
```

Mỗi lần flow kết thúc, tạo checkpoint mới lưu giá trị băng thông riêng của flow này. Dòng active now đếm tổng số flow vẫn còn hoạt động tại thời điểm tạo checkpoint chkpt time. Điều này giúp xác định có bao nhiêu flow cùng đồng thời tồn tại trong khoảng thời gian đó – là cơ sở để cập nhật các checkpoint về sau. Việc theo dõi số lượng flow đang active cho phép kiểm soát khi nào một checkpoint trở nên "final" – tức khi active == 0.

Bước 5: Cập nhật các checkpoint cũ

```
for c in checkpoints:
    if flow['start'] <= c['time'] <= flow['end']:
        c['util'] += flow['util']
        c['active'] -= 1
```

Nếu flow tồn tại trong khoảng thời gian của checkpoint cũ, cộng thêm giá trị vào checkpoint tương ứng và giảm số lượng active flow. khi active của một checkpoint giảm về 0, checkpoint đó được xem là kết thúc và không còn thay đổi trong các bước cập nhật tiếp theo.

Bước 6: Cập nhật Active list

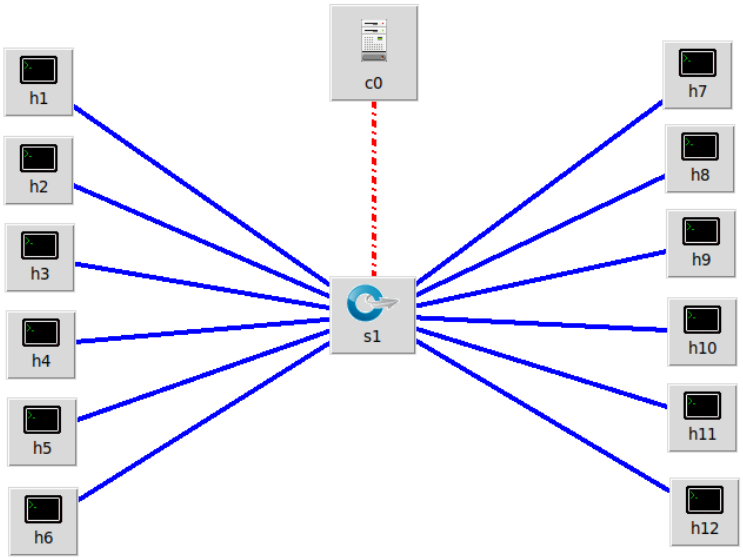
```
# Cập nhật Active List
active_list = [f for f in active_list if f['flow_id']!=
 ↪  flow['flow_id']]
active_list.append(flow)
```

Loại flow vừa kết thúc khỏi danh sách đang hoạt động, và cập nhật flow mới để tiếp tục theo dõi các flow còn lại.

3.4 Đánh giá Granularity của FlowSense

Trong phần này, hệ thống được mở rộng để khảo sát khả năng phản hồi chi tiết (granularity) của thuật toán FlowSense – tức khả năng phát hiện và phản ánh các thay đổi nhỏ về mức sử dụng băng thông trong thời gian ngắn. Hai tiêu chí được sử dụng để đánh giá bao gồm: độ trễ của checkpoint và tỷ lệ dữ liệu ghi nhận theo thời gian.

3.4.1 Mô hình mạng khảo sát Granularity



Hình 3.2: Mô hình mạng khảo sát Granularity.

Do giới hạn về điều kiện triển khai trong khuôn khổ đồ án, mô hình khảo sát Granularity không thể áp dụng trực tiếp cấu trúc mạng campus phức tạp như trong bài báo gốc FlowSense[10]. Thay vào đó, đồ án sử dụng một mô hình mạng đơn giản hơn, với kiến trúc sao gồm một switch trung tâm (s1) kết nối với 12 host (h1 → h12). Mô hình này vẫn đảm bảo các yếu tố quan trọng để khảo sát khả năng cập nhật lưu lượng của FlowSense trong điều kiện có mức độ thay đổi lưu lượng cao và không đồng đều theo thời gian:

- Các host h1 đến h6 đóng vai trò nguồn phát lưu lượng.
- Các host h7 đến h12 là các đích nhận dữ liệu.
- Các cặp truyền được lựa chọn ngẫu nhiên và điều chỉnh theo kịch bản để tạo ra thay đổi lưu lượng có chủ đích.

3.4.2 Độ trễ của checkpoint

Mục tiêu của tiêu chí này là kiểm tra tại thời điểm một checkpoint được tạo ra – tức khi một flow kết thúc – còn bao nhiêu flow vẫn đang hoạt động và mất bao lâu để tất cả các flow này thực sự kết thúc. Thông số này phản ánh độ trễ cần chờ để thu thập đầy đủ thông tin về mức sử dụng tại thời điểm đó. Nếu thời gian này lớn, checkpoint đang phản ánh thiếu và chưa thể hiện được đúng mức sử dụng mạng. Đoạn mã xử lí chính:

```
# Tính thời gian chờ từ mỗi checkpoint đến khi flow cuối kết thúc
checkpoints = [f['end'] for f in flows]
delays = []
for t in checkpoints:
    active_flows = [f for f in flows if f['start'] <= t <= f['end']]
    if not active_flows:
        continue
    latest_end = max([f['end'] for f in active_flows])
    delays.append(latest_end - t)
```

Với mỗi checkpoint t, hệ thống tìm tất cả flow còn đang active tại thời điểm đó, lấy thời điểm kết thúc cuối cùng latest end, và tính độ trễ latest end - t. Toàn bộ phân phối

các độ trễ này sau đó được vẽ thành đồ thị CDF (phân phối tích lũy) để đánh giá khả năng thu thập dữ liệu đúng thời điểm.

3.4.3 Tỷ lệ dữ liệu được ghi nhận theo thời gian kể từ checkpoint

Tiêu chí này được sử dụng để đánh giá tốc độ phản ứng và khả năng bao phủ dữ liệu của FlowSense sau khi một checkpoint được tạo ra – một yếu tố quan trọng trong các ứng dụng yêu cầu phát hiện sớm sự thay đổi lưu lượng. Cụ thể, hệ thống sẽ được kiểm tra xem sau 1, 5 và 10 giây kể từ thời điểm checkpoint, đã thu thập được bao nhiêu phần trăm tổng lượng byte từ các flow đang hoạt động tại thời điểm đó.

```
# Tính % utilization report sau 1s, 5s, 10s
percentages_1s, percentages_5s, percentages_10s = [], [], []
for t in checkpoints:
    active = [f for f in flows if f['start'] <= t <= f['end']]
    if not active:
        continue
    total_bytes = sum(f['bytes'] for f in active)

    def pct(after):
        reported = sum(f['bytes'] for f in active if f['end'] <= t +
        ↪  after)
        return reported / total_bytes if total_bytes > 0 else 0
```

Tại mỗi checkpoint t, hệ thống tính tổng lượng byte thực tế (ground truth) từ các flow đang active. Sau đó, lần lượt đo lượng dữ liệu mà hệ thống đã ghi nhận được sau after = 1, 5, 10 giây. Kết quả được tổng hợp thành ba đường CDF biểu diễn tỷ lệ phần trăm mức sử dụng được cập nhật sau mỗi khoảng thời gian.

3.5 Kết luận

Trong chương này đã trình bày chi tiết quá trình triển khai hệ thống giám sát mạng sử dụng thuật toán FlowSense trong môi trường Mininet và Ryu controller. Quá trình hiện

thực bao gồm việc xử lý sự kiện PacketIn và FlowRemoved, tái dựng luồng truyền, tính toán mức sử dụng băng thông theo thuật toán giám sát băng thông và quản lý checkpoint để phản ánh chính xác tải mạng theo thời gian.

Bên cạnh đó, chương cũng giới thiệu hai kịch bản khảo sát granularity, nhằm đánh giá độ chi tiết và khả năng phản hồi của thuật toán trong các tình huống thực tế, với các flow kết thúc tại thời điểm khác nhau.

Toàn bộ dữ liệu từ các thí nghiệm sẽ được trình bày, phân tích và so sánh trong Chương 4, nhằm làm rõ hiệu quả của FlowSense so với phương pháp polling, cũng như khả năng phản ánh chính xác mức sử dụng mạng trong các điều kiện khác nhau.

Chương 4

KẾT QUẢ MÔ PHỎNG

4.1 Giới thiệu

Chương này trình bày các kết quả thu được từ quá trình triển khai mô hình FlowSense trong môi trường mô phỏng SDN. Hai khảo sát chính được thực hiện bao gồm: đánh giá độ chính xác của FlowSense so với phương pháp polling truyền thống và khảo sát độ chi tiết (granularity) trong việc cập nhật dữ liệu sử dụng băng thông theo thời gian. Nội dung chương bao gồm mô tả kịch bản truyền, phương pháp đo và kết quả trực quan dưới dạng biểu đồ.

4.2 Đánh giá độ chính xác của FlowSense so với Polling

4.2.1 Kịch bản thực hiện

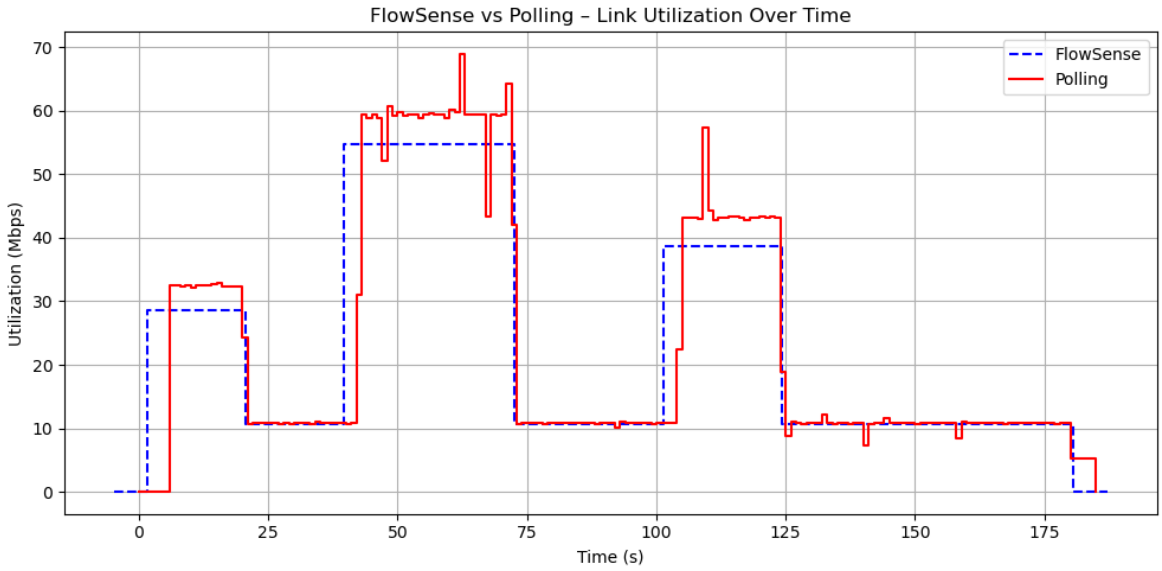
Kịch bản thực hiện trong đồ án này được xây dựng mô phỏng lại theo đúng khảo sát trong bài báo FlowSense[10], nhằm đảm bảo điều kiện so sánh độ chính xác giữa phương pháp FlowSense và Polling trong cùng một môi trường mạng đơn giản nhưng dễ kiểm soát. Mô hình mạng được triển khai như đã mô tả ở mục 3.2, với 3 host và 2 switch theo cấu trúc tuyến tính, trong đó các luồng UDP được thiết lập theo chu kỳ.

Cụ thể, một luồng nền UDP có băng thông 10 Mbps được truyền liên tục từ host 1 đến host 3 trong suốt thời gian mô phỏng. Bên cạnh đó, một loạt các pha truyền UDP từ

host 1 đến host 2 được thiết lập tuần tự, mỗi pha kéo dài 20 giây với các mức tải tăng dần hoặc giảm dần lần lượt là 20 Mbps, 45 Mbps và 30 Mbps. Mục tiêu của các pha truyền này là tạo ra các thay đổi rõ rệt trong lưu lượng mạng để kiểm tra khả năng phản ứng của các phương pháp giám sát.

Hệ thống giám sát lưu lượng được triển khai theo hai phương pháp: FlowSense – giám sát thụ động bằng cách tận dụng các sự kiện PacketIn và FlowRemoved từ switch, và Polling – giám sát chủ động bằng cách gửi truy vấn FlowStats định kỳ mỗi giây để thu thập thông tin từ switch. Kết quả ghi nhận được từ hai phương pháp sẽ được chuẩn hóa và phân tích để so sánh độ chính xác trong ước lượng băng thông theo thời gian.

4.2.2 Kết quả



Hình 4.1: Độ chính xác giám sát băng thông.

Dựa trên Hình 4.1, có thể nhận thấy rằng kết quả đo mức sử dụng băng thông từ hai phương pháp giám sát – FlowSense và Polling (phân tích theo chênh lệch byte đếm được) – có sự tương quan rất sát trong suốt quá trình thay đổi lưu lượng mạng. Đường biểu diễn của FlowSense có khả năng phản ánh đúng các pha thay đổi lưu lượng, với độ chính xác

cao tại cả các giai đoạn tăng và giảm. Đặc biệt, FlowSense có thể phản ứng tức thì với sự xuất hiện của luồng mới bằng cách tận dụng thông tin PacketIn và thời điểm flow bị gỡ bỏ (FlowRemoved), qua đó xác định chính xác thời gian hoạt động và lưu lượng thực tế.

Phương pháp Polling tuy có thể cung cấp chi tiết hơn do đo liên tục theo chu kỳ, nhưng có thể tạo ra lưu lượng mạng bổ sung, nhìn vào kết quả thấy được Polling xuất hiện nhiễu nhẹ do sự dao động trong byte_count giữa các lần thăm dò. Ngoài ra, ở các đoạn duy trì mức lưu lượng ổn định, biểu đồ FlowSense vẫn thể hiện đúng vì nó được tính trung bình trong suốt thời gian giữa các flow nhưng FlowSense không thể đo được giá trị băng thông tức thời.

Có thể khẳng định rằng, việc kết hợp thông tin từ các sự kiện FlowRemoved và PacketIn, cùng với thuật toán khôi phục hoạt động flow theo đoạn (step-wise), giúp FlowSense đạt được độ chính xác cao trong giám sát. Kết quả này đặc biệt ấn tượng nếu xét đến việc FlowSense không cần bất kỳ gói giám sát chủ động nào, giúp giảm thiểu tối đa lưu lượng điều khiển trong mạng SDN. Với các kết quả thu được, hoàn toàn có thể xem FlowSense là một phương pháp thay thế phù hợp cho Polling trong các ứng dụng yêu cầu độ chính xác tương đương nhưng tối ưu tài nguyên mạng tốt hơn.

4.3 Đánh giá Granularity của FlowSense

4.3.1 Kịch bản thực hiện

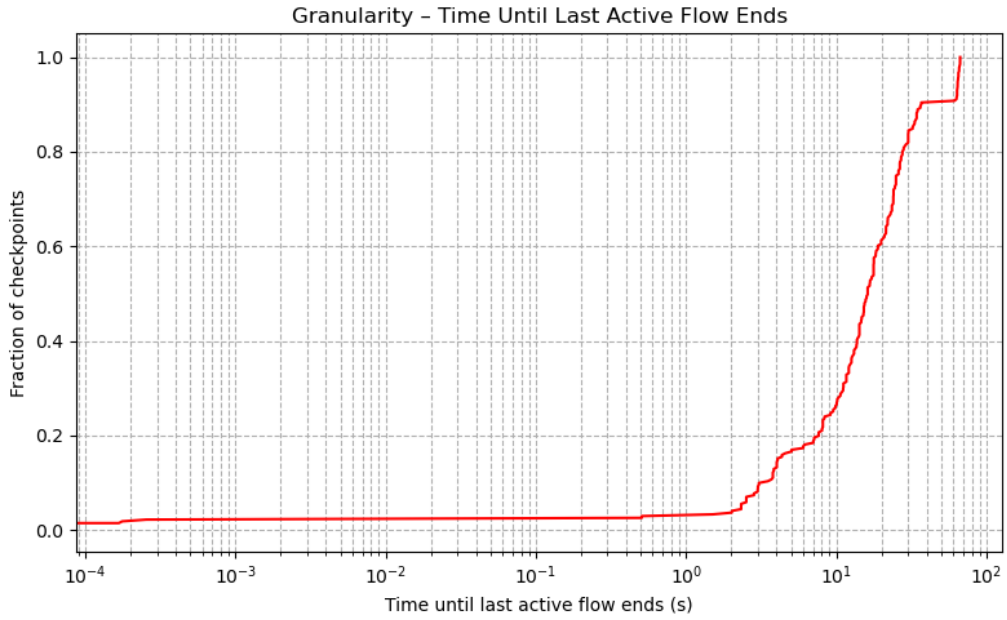
Do điều kiện thực tế trong khuôn khổ đồ án, mô hình khảo sát không thể triển khai trên một mạng campus quy mô lớn như trong bài báo FlowSense [10]. Thay vào đó, một mô hình mạng nhỏ hơn đã được thiết kế nhưng vẫn đảm bảo mô phỏng được các đặc điểm quan trọng để khảo sát khả năng cập nhật lưu lượng của FlowSense trong môi trường có tính biến động cao.

Cụ thể, mô hình gồm một switch trung tâm kết nối đến 12 host (6 host nguồn và 6 host đích) được mô tả ở mục 3.4.1, với các luồng UDP được thiết lập từ nguồn đến

đích theo các chu kỳ ngẫu nhiên. Trong mỗi vòng kiểm thử, các host nguồn sẽ tạo ra các luồng truyền UDP đến các host đích với tốc độ ngẫu nhiên từ 5–10 Mbps. Mỗi luồng có thời lượng khác nhau, gồm các luồng ngắn (5–10 giây) và luồng dài (40–60 giây), được trộn lẫn ngẫu nhiên nhằm tạo ra đặc tính bất định trong thời gian tồn tại của các flow.

Việc sử dụng các luồng có tính chất thay đổi không đồng đều cho phép đánh giá mức độ chi tiết (granularity) của FlowSense trong việc theo dõi băng thông. Mặc dù quy mô mô hình nhỏ hơn so với nghiên cứu gốc nhưng tính chất và hành vi luồng được mô phỏng vẫn đủ để khảo sát khả năng cập nhật và phản ánh lưu lượng theo thời gian thực của hệ thống giám sát.

4.3.2 Kết quả



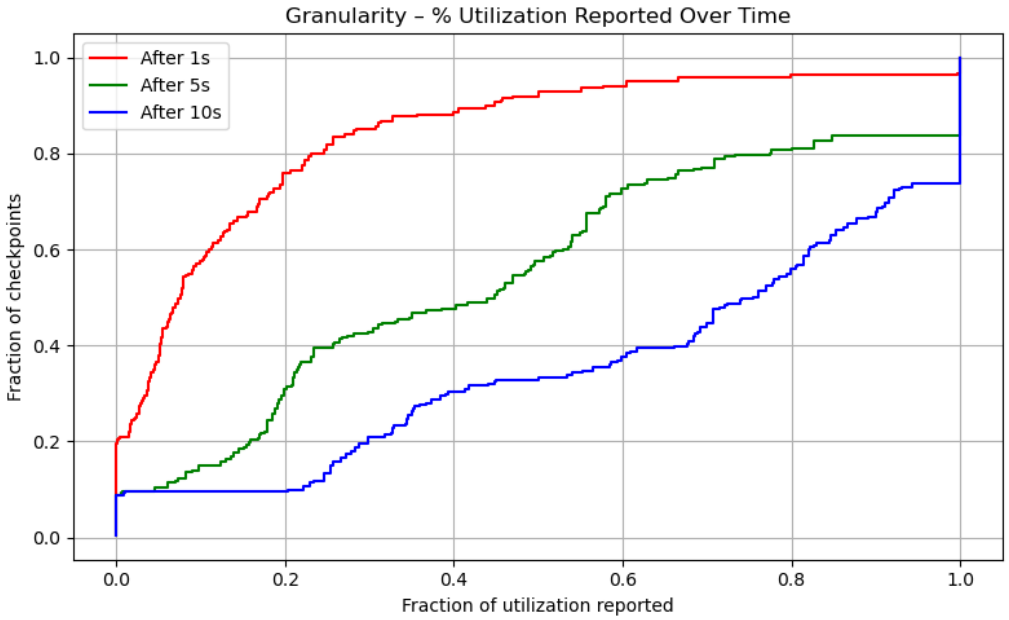
Hình 4.2: Thời gian ghi nhận đầy đủ byte.

Hình 4.2 là biểu đồ thể hiện khoảng thời gian kể từ thời điểm một flow kết thúc (checkpoint) đến khi toàn bộ các flow còn đang hoạt động tại thời điểm đó cũng kết thúc, nghĩa là toàn bộ dữ liệu cần thiết đã được ghi nhận. Trục hoành là thời gian chờ, còn trục tung là phần trăm các checkpoint. Nhìn vào biểu đồ có thể thấy:

- Dưới 10 giây, chỉ 25–30% checkpoint có thể thu thập đủ dữ liệu, cho thấy nhiều flow vẫn còn đang hoạt động sau checkpoint và hệ thống cần thời gian dài hơn để phản ánh đúng mức sử dụng thực tế.
- Khoảng 90% checkpoint cần ít nhất 60 giây để toàn bộ flow kết thúc, tức có một tỷ lệ lớn flow kéo dài gây ra độ trễ trong việc thu thập đầy đủ dữ liệu.

Điều này phản ánh rằng phần lớn các checkpoint phải chờ khá lâu mới ghi nhận đầy đủ dữ liệu sử dụng, nguyên nhân là do tồn tại nhiều flow dài đan xen với các flow ngắn trong toàn bộ quá trình truyền.

Tuy nhiên, biểu đồ cũng cho thấy độ dốc lớn bắt đầu từ mốc 30 giây trở đi, chứng tỏ sự khác biệt rõ rệt giữa nhóm checkpoint có flow ngắn và nhóm có flow dài. Kết quả này phù hợp với đặc điểm của kịch bản thử nghiệm – trong đó các flow ngắn và dài được tạo ra xen kẽ theo chu kỳ. Như vậy, với các ứng dụng yêu cầu phản hồi nhanh, nếu chấp nhận một phần độ trễ, hệ thống vẫn có thể thu thập thông tin sử dụng tương đối đầy đủ mà không cần đợi tất cả flow kết thúc.



Hình 4.3: Tỷ lệ byte được ghi nhận sau 1s, 5s, 10s.

Hình 4.3 là biểu đồ thể hiện tỷ lệ phần trăm lưu lượng (tính theo byte) được ghi nhận tại mỗi checkpoint sau 1, 5 và 10 giây kể từ thời điểm flow kết thúc. Trục hoành là phần trăm dữ liệu đã được ghi nhận, trục tung là phần trăm checkpoint tương ứng. Nhìn vào biểu đồ có thể thấy:

- Sau 1 giây chỉ có khoảng 5% checkpoint có thể báo cáo 80% việc sử dụng băng thông.
- Nếu tăng thời gian chờ lên 5 giây và 10 giây sẽ có nhiều % checkpoint có thể đạt được mức sử dụng băng thông lớn hơn 80%.

Tổng thể, kết quả cho thấy hệ thống FlowSense cần một khoảng thời gian nhất định (trên 5 giây) để ghi nhận phần lớn dữ liệu từ các flow ngắn. Tuy nhiên, trong vòng 10 giây, hệ thống vẫn đảm bảo khả năng thu thập gần đầy đủ thông tin, phù hợp với các ứng dụng yêu cầu giám sát gần thời gian thực.

Chương 5

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Trong quá trình thực hiện đồ án, hệ thống giám sát lưu lượng mạng SDN dựa trên thuật toán FlowSense đã được xây dựng và triển khai thành công. Bằng cách tận dụng các sự kiện điều khiển sẵn có trong mạng OpenFlow như PacketIn và FlowRemoved, hệ thống có thể giám sát mức sử dụng băng thông theo thời gian một cách thụ động, mà không cần tạo thêm lưu lượng điều khiển như phương pháp Polling truyền thống.

Một trong những điểm nổi bật đạt được là việc triển khai mô hình mô phỏng có khả năng tái hiện hành vi mạng thực tế, qua đó kiểm chứng độ chính xác và khả năng phản hồi của FlowSense. Trong đó, hệ thống đã được so sánh trực tiếp với phương pháp Polling định kỳ thông qua cùng một tập lưu lượng truyền tải UDP. Kết quả biểu đồ cho thấy FlowSense có thể phản ánh gần đúng mức sử dụng băng thông theo thời gian với độ trễ thấp hơn và không gây thêm lưu lượng điều khiển trên mạng. Đặc biệt, trong các pha truyền tải có tải trọng lớn, mức độ khớp giữa đường biểu diễn của FlowSense và Polling đạt mức rất cao, thể hiện khả năng theo dõi chính xác theo thời gian thực.

Tuy vậy, cũng cần thừa nhận rằng kết quả so sánh này chỉ được thực hiện trong môi trường giả lập quy mô nhỏ với kịch bản truyền tải đơn giản. Độ chính xác của FlowSense phụ thuộc vào thời điểm các flow kết thúc và sự kiện FlowRemoved được gửi

về controller, do đó trong một số trường hợp độ trễ có thể tăng đáng kể nếu flow kéo dài hoặc không kết thúc đúng hạn. Ngoài ra, Polling cung cấp số liệu chi tiết hơn ở từng thời điểm cụ thể, trong khi FlowSense chỉ ghi nhận thông tin tại các checkpoint ngắt quãng. Điều này tạo ra sự sai lệch nhẹ trong các giai đoạn đầu và cuối mỗi pha truyền tải.

Bên cạnh đó, hai tiêu chí dùng để đánh giá granulariy đã được kế thừa từ bài báo gốc và áp dụng thành công để kiểm tra khả năng phản ánh dữ liệu chi tiết của hệ thống. Kết quả khảo sát cho thấy hệ thống có thể ghi nhận đầy đủ dữ liệu sau 60–100 giây trong phần lớn các checkpoint và thậm chí thu thập được phần lớn lưu lượng chỉ sau 10 giây, minh chứng cho khả năng giám sát tương đối kịp thời trong môi trường có nhiều flow ngắn. Tuy nhiên, vẫn tồn tại những checkpoint cần chờ đến 30 giây trở lên để hoàn tất ghi nhận toàn bộ dữ liệu, chủ yếu do sự hiện diện của các flow dài kéo dài thời gian hoàn tất.

Tổng thể, đồ án đã triển khai thành công một hệ thống giám sát mạng SDN thụ động dựa trên FlowSense, xác thực tính hiệu quả của phương pháp này trong môi trường mô phỏng và làm rõ các giới hạn cần lưu ý nếu triển khai thực tế ở quy mô lớn hơn.

5.2 Hướng phát triển

Trong tương lai, hệ thống có thể được mở rộng và phát triển theo nhiều hướng nhằm nâng cao hiệu quả và tính ứng dụng thực tế. Trước hết, cần triển khai FlowSense trong các mô hình mạng phức tạp hơn như mạng trung tâm dữ liệu spine-leaf [13], để đánh giá khả năng mở rộng, độ tin cậy và độ trễ trong điều kiện thật. Các thách thức như luồng tồn tại kéo dài, timeout lớn hoặc wildcard rules sẽ cần được phân tích sâu hơn để hiểu rõ mức độ ảnh hưởng đến hiệu quả giám sát.

Ngoài ra, việc tích hợp thêm các thuật toán lọc và phân loại lưu lượng tại controller sẽ giúp hệ thống tập trung vào các flow quan trọng, giúp giảm tải xử lý và nâng cao hiệu quả giám sát. Bên cạnh đó, kết hợp phương pháp thụ động như FlowSense với cơ chế chủ động (hybrid monitoring) [14] sẽ cải thiện khả năng phản hồi tức thì đối với các sự kiện mạng đột biến, đồng thời duy trì được độ chính xác ở mức cao trong điều kiện mạng

động.

Với các hướng phát triển này, hệ thống giám sát dựa trên FlowSense hứa hẹn sẽ trở thành một công cụ hiệu quả, tiết kiệm tài nguyên và có tính ứng dụng cao trong các hệ thống SDN hiện đại.

Tài liệu tham khảo

[1] A. Kashinath, M. Hasan, S. Mohan, R. B. Bobba, and R. Mittal, "Improving dependability via deadline guarantees in commodity real-time networks," in *Proceedings of the 2020 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2020, pp. 1–6.

[2] G. S. Dong, J. Shen, and L.-Q. Sun, "Fast failure recovery in software-defined networks," in *Proceedings of the 2017 5th International Conference on Frontiers of Manufacturing Science and Measuring Technology (FMSMT 2017)*, 2017.

[3] L. Cui, F. R. Yu, and Q. Yan, "When big data meets software-defined networking: Sdn for big data and big data for sdn," *IEEE Network*, vol. 30, no. 1, pp. 58–65, Jan. 2016.

[4] Z. Cai, A. L. Cox, and T. S. E. Ng, "Maestro: A system for scalable openflow control," Rice University, Tech. Rep. TR10-11, 2010.

[5] V. H. Kha, "Software defined networking - github repository," 2022, https://github.com/Van-Hoang-Kha/Software-Defined-Networking.

[6] HocChuDong Team, "Tìm hiểu sdn - openflow," 2017, https://github.com/hocchudong/thuctap012017/blob/master/TamNT/TimHieuSDN/docs/1.Gioi_thieu_SDN-OpenFlow.md.

[7] TechTarget, "Openflow," https://www.techtarget.com/whatis/definition/OpenFlow.

[8] M. G. A. Tootoonchian and Y. Ganjali., "Opentm: Traffic matrix estimator for openflow networks," in *In PAM*, 2010.

[9] L. Jose, M. Yu, and J. Rexford, "Online measurement of large traffic aggregates on commodity switches," in *In USENIX Hot-ICE*, 2011.

[10] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "Flowsense: Monitoring network utilization with zero measurement cost," in *Passive and Active Measurement Conference (PAM)*, ser. Lecture Notes in Computer Science, vol. 7799. Springer, 2013, pp. 31–41.

[11] Mininet Team, "Mininet: An instant virtual network on your laptop." [Online]. Available: http://mininet.org/

[12] Ryu SDN Framework, "Ryu sdn framework: Component-based software defined networking framework," https://ryu-sdn.org/.

[13] Leviton, "Spine-leaf architecture." [Online]. Available: https://leviton.com/markets/data-centers/network-architectures/spine-leaf-architecture

[14] Z. Su, T. Wang, Y. Xia, and M. Hamdi, "Cemon: A cost-effective flow monitoring system in software defined networks," *Computer Networks*, vol. 92, no. 1, pp. 101–115, 2015.