TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TRẦN THANH HUY - 51900800**

**NGUYỄN TRƯỜNG KHẢ - 51900752**

**PHÁT HIỆN TẤN CÔNG DDOS**

**DỰ ÁN CÔNG NGHỆ THÔNG TIN 2**

**MẠNG MÁY TÍNH**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TRẦN THANH HUY - 51900800**

**NGUYỄN TRƯỜNG KHẢ- 51900752**

**PHÁT HIỆN TẤN CÔNG DDOS**

**DỰ ÁN CÔNG NGHỆ THÔNG TIN 2**

**MẠNG MÁY TÍNH**

Người hướng dẫn

**TS. TRƯƠNG ĐÌNH TÚ**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

**LỜI CẢM ƠN**

Lời đầu tiên em xin chân thành cảm ơn quý thầy, cô khoa Công Nghệ Thông Tin đã tận tình giảng dạy, chỉ bảo em trong những năm học vừa qua, những kiến thức mà em nhận được trên giảng đường đại học là nền tảng cho quá trình nghiên cứu đề tài này. Em xin chân thành cảm ơn Khoa Công nghệ thông tin đã tạo điều kiện để em có thể thực hiện đề tài nghiên cứu.

Em xin chân thành cảm ơn thầy Trương Đình Tú – giảng viên hướng dẫn em môn học Dự án Công nghệ thông tin, đã luôn theo sát, nhiệt tình giúp đỡ em trong suốt quá trình thực hiện đề tài **Phát Hiện tấn công DDoS**.

Mặc dù bản thân em đã nỗ lực hết mức, nhưng bài báo cáo này không thể tránh khỏi còn nhiều thiếu sót. Vì thế, em mong rằng sẽ nhận được những lời nhận xét, góp ý từ quý thầy, cô để em có thể tiếp tục phát triển và hoàn thiện hơn nữa.

Một lần nữa, em xin chân thành cảm ơn!

*TP. Hồ Chí Minh, ngày ... tháng … năm 20..*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của TS. Trương Đình Tú. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình**. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày … tháng … năm 20..*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

**PHÁT HIỆN TẤN CÔNG DDoS**

**TÓM TẮT**

Các cuộc tấn công từ chối dịch vụ phân tán (DDoS) đã trở thành vũ khí ưa thích của tin tặc, tống tiền mạng và khủng bố mạng. Những cuộc tấn công này có thể nhanh chóng vô hiệu hóa nạn nhân, gây thiệt hại lớn về doanh thu. Mặc dù có nhiều giải pháp truyền thống để giảm thiểu, các cuộc tấn công DDoS vẫn tiếp tục gia tăng về tần suất, quy mô và mức độ nghiêm trọng. Điều này đòi hỏi một mô hình mạng mới để đối phó với các mối đe dọa an ninh đầy thách thức ngày nay. Mạng điều khiển bằng phần mềm (SDN) là một mô hình mạng mới nổi, đã thu hút được sự quan tâm đáng kể từ nhiều nhà nghiên cứu để đáp ứng yêu cầu của các trung tâm dữ liệu hiện nay.

Phương pháp học máy đang ngày càng được ứng dụng rộng rãi trong việc phát hiện tấn công từ chối dịch vụ phân tán (DDoS). So với các phương pháp truyền thống, như dựa trên các quy tắc tĩnh hoặc ngưỡng phát hiện cố định, phương pháp học máy có khả năng phân tích và nhận diện các mẫu hành vi phức tạp và bất thường trong lưu lượng mạng. Các mô hình học máy có thể học từ dữ liệu lịch sử, cải thiện độ chính xác và khả năng phát hiện các cuộc tấn công mới hoặc chưa từng gặp trước đây. Điều này làm cho phương pháp học máy trở nên linh hoạt và hiệu quả hơn trong môi trường mạng liên tục biến đổi, giúp bảo vệ hệ thống tốt hơn trước các mối đe dọa DDoS ngày càng tinh vi.

**DDoS ATTACK DETECTION**

**ABSTRACT**

Distributed denial of service (DDoS) attacks have become the preferred weapon of hackers, cyber extortion, and cyber terrorism. These attacks can quickly disable victims, causing major revenue losses. Despite many traditional solutions for mitigation, DDoS attacks continue to increase in frequency, scale, and severity. This requires a new networking paradigm to deal with today's challenging security threats. Software-driven networking (SDN) is an emerging networking paradigm that has attracted significant attention from many researchers to meet the requirements of today's data centers.

Machine learning methods are increasingly being widely applied in detecting distributed denial of service (DDoS) attacks. Compared to traditional methods, such as based on static rules or fixed detection thresholds, machine learning methods are capable of analyzing and identifying complex and unusual behavioral patterns in network traffic. Machine learning models can learn from historical data, improving accuracy and the ability to detect new or never-before-seen attacks. This makes machine learning more flexible and effective in constantly changing network environments, helping to better protect systems against increasingly sophisticated DDoS threats.

**MỤC LỤC**

[DANH MỤC HÌNH VẼ ix](#_Toc173012545)

[DANH MỤC BẢNG BIỂU xi](#_Toc173012546)

[DANH MỤC CÁC CHỮ VIẾT TẮT xii](#_Toc173012547)

[CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI 1](#_Toc173012548)

[1.1 Giới thiệu đề tài 1](#_Toc173012549)

[1.2 Mục tiêu thực hiện đề tài 2](#_Toc173012550)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 3](#_Toc173012551)

[2.1 Tấn cộng từ chối dịch vụ phân tán (DDoS) 3](#_Toc173012552)

[2.1.1 Định nghĩa 3](#_Toc173012553)

[2.1.2 So sánh DoS và DDoS 4](#_Toc173012554)

[2.1.3 Các loại tấn công DDoS 5](#_Toc173012555)

[2.1.4 Tác động của tấn công DDos 6](#_Toc173012556)

[2.1.5 Các phương pháp phát hiện DDoS 6](#_Toc173012557)

[2.2 Công nghệ SDN 8](#_Toc173012558)

[2.2.1 Lịch sử và sự ra đời của SDN 8](#_Toc173012559)

[2.2.2 Định nghĩa về SDN 9](#_Toc173012560)

[2.2.3 Kiến trúc SDN 10](#_Toc173012561)

[2.2.4 Một số mô hình SDN phổ biến 12](#_Toc173012562)

[2.2.5 So sánh SDN và mạng truyền thống 13](#_Toc173012563)

[2.2.6 Ưu điểm và nhược điểm của SDN 13](#_Toc173012564)

[2.3 Giao thức OpenFlow 16](#_Toc173012565)

[2.3.1 Giới thiệu OpenFlow 16](#_Toc173012566)

[2.3.2 Kiến trúc OpenFlow 16](#_Toc173012567)

[2.3.3 Hoạt động của OpenFlow 17](#_Toc173012568)

[2.3.4 Lợi ích của OpenFlow 17](#_Toc173012569)

[2.3.5 Ứng dụng của OpenFlow 18](#_Toc173012570)

[2.4 Mô hình K-Nearest Neighbor 18](#_Toc173012571)

[2.4.1 Lịch sử hình thành 18](#_Toc173012572)

[2.4.2 Ý tưởng cơ bản của thuật toán KNN 18](#_Toc173012573)

[2.4.3 Ví dụ về thuật toán KNN 19](#_Toc173012574)

[2.4.4 Các nguyên tắc chọn k cơ bản 21](#_Toc173012575)

[2.4.5 Ưu điểm và Nhược điểm của KNN 21](#_Toc173012576)

[2.4.6 Ứng dụng của KNN 21](#_Toc173012577)

[2.5 Ryu controller 22](#_Toc173012578)

[2.5.1 Giới thiệu về Ryu Controller 22](#_Toc173012579)

[2.5.2 Kiến trúc của Ryu Controller 22](#_Toc173012580)

[2.5.3 Hoạt động của Ryu Controller 23](#_Toc173012581)

[2.5.4 Ưu điểm của Ryu Controller 23](#_Toc173012582)

[2.5.5 Ứng dụng của Ryu Controller trong phát hiện và ngăn chặn tấn công DDoS 24](#_Toc173012583)

[2.6 Mininet 24](#_Toc173012584)

[2.6.1 Giới thiệu về Mininet 24](#_Toc173012585)

[2.6.2 Kiến trúc của Mininet 25](#_Toc173012586)

[2.6.3 Cách hoạt động của Mininet 25](#_Toc173012587)

[2.6.4 Lợi ích của Mininet 26](#_Toc173012588)

[2.7 Wireshark 26](#_Toc173012589)

[2.7.1 Giới thiệu về Wireshark 26](#_Toc173012590)

[2.7.2 Lịch sử phát triển 26](#_Toc173012591)

[2.7.3 Chức năng chính của Wireshark 26](#_Toc173012592)

[2.7.4 Kiến trúc và hoạt động của Wireshark 27](#_Toc173012593)

[2.7.5 Sử dụng Wireshark trong thực tế 27](#_Toc173012594)

[2.7.6 Lợi ích của Wireshark 28](#_Toc173012595)

[2.8 Hping 28](#_Toc173012596)

[2.8.1 Giới thiệu về Hping 28](#_Toc173012597)

[2.8.2 Các tính năng chính của Hping 28](#_Toc173012598)

[2.8.3 Ứng dụng của Hping 28](#_Toc173012599)

[2.8.4 Ưu điểm và hạn chế của Hping 29](#_Toc173012600)

[CHƯƠNG 3. THỰC NGHIỆM 29](#_Toc173012601)

[3.1 Thiết kế hệ thống 29](#_Toc173012602)

[3.2 Thu thập dữ liệu 30](#_Toc173012603)

[3.2.1 Mô tả các bước thu thập dữ liệu 30](#_Toc173012604)

[3.2.2 Mô tả dữ liệu 31](#_Toc173012605)

[3.3 Đánh giá mô hình KNN 33](#_Toc173012606)

[3.4 Mô tả cách hoạt động của quá trình thực nghiệm 35](#_Toc173012607)

[3.5 Kịch bản thực nghiệm 36](#_Toc173012608)

[3.5.1 Chuẩn bị 37](#_Toc173012609)

[3.5.2 Thực nghiệm phát hiện DDoS 38](#_Toc173012610)

[3.5.3 Migration 43](#_Toc173012611)

[CHƯƠNG 4. KẾT LUẬN 48](#_Toc173012612)

[4.1 Kết luận 48](#_Toc173012613)

[4.2 Hướng phát triển 49](#_Toc173012614)

[TÀI LIỆU THAM KHẢO 50](#_Toc173012615)

# DANH MỤC HÌNH VẼ

[Hình 1: Kiến trúc cơ bản của SDN 11](#_Toc173012616)

[Hình 2: Open SDN 12](#_Toc173012617)

[Hình 3: SDN via Hypervisor-based Overlay Network 13](#_Toc173012618)

[Hình 4: Các thành phần chính của OpenFlow 16](#_Toc173012619)

[Hình 5 19](#_Toc173012620)

[Hình 6 20](#_Toc173012621)

[Hình 7 20](#_Toc173012622)

[Hình 8: Thiết kế hệ thống SDN 30](#_Toc173012623)

[Hình 9: Sơ đồ quá trình thu thập dataset 30](#_Toc173012624)

[Hình 10: Ma trận nhầm lẫn 33](#_Toc173012625)

[Hình 11: Biểu đồ ROC 34](#_Toc173012626)

[Hình 12: 34](#_Toc173012627)

[Hình 13: Tập dữ liệu của bài báo 34](#_Toc173012628)

[Hình 14: Kết quả đánh giá trong bài báo 35](#_Toc173012629)

[Hình 15: Chỉ số AUC 35](#_Toc173012630)

[Hình 16: Sơ đồ tổng quan cách hoạt động của quá trình thực nghiệm 35](#_Toc173012631)

[Hình 17: Đổi địa chỉ IP thành địa chỉ của máy Ryu 38](#_Toc173012632)

[Hình 18: Khởi chạy KNN\_controller 39](#_Toc173012633)

[Hình 19: Khởi tạo mô hình mạng 39](#_Toc173012634)

[Hình 20: Ping đến node h1 40](#_Toc173012635)

[Hình 21: Ping đến node h5 40](#_Toc173012636)

[Hình 22: Kết quả dự đoán khi traffic bình thường 40](#_Toc173012637)

[Hình 23: Phát hiện host h1 bị tấn công 41](#_Toc173012638)

[Hình 24: Biểu đồ traffic trên host h1 41](#_Toc173012639)

[Hình 25: Phát hiện host h2 bị tấn công 42](#_Toc173012640)

[Hình 26: Biểu đồ traffic tại host h2 42](#_Toc173012641)

[Hình 27: Phát hiện host h4 bị tấn công 43](#_Toc173012642)

[Hình 28: Biểu đồ traffic tại host h4 43](#_Toc173012643)

[Hình 29: Khởi chạy migration controller 44](#_Toc173012644)

[Hình 30: Khởi tạo mô hình mạng 44](#_Toc173012645)

[Hình 31: Test Ping bình thường 45](#_Toc173012646)

[Hình 32: Thực hiện tấn công ICMP Flood tới host h1 45](#_Toc173012647)

[Hình 33: Biểu đồ host h1 bị tấn công ICMP Flood, sau đó hệ thống phát hiện và block port của h1 trên switch 1 46](#_Toc173012648)

[Hình 34: Thực hiện tấn công UDP Flood tới host h2 46](#_Toc173012649)

[Hình 35: 47](#_Toc173012650)

[Hình 36: Thực hiện tấn công TCP-SYN Flood tới h4 47](#_Toc173012651)

[Hình 37: 48](#_Toc173012652)

# DANH MỤC BẢNG BIỂU

[Bảng 4.1: Thống kê kiểu thực thể trong tập VLSP 2016 11](#_Toc171792390)

# DANH MỤC CÁC CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| DDoS | Distributed denial of service |
| DoS | Denial of service |
| SDN | Software defined Networking |
| KNN | K-Nearest Neighbor |
|  |  |

# TỔNG QUAN ĐỀ TÀI

## Giới thiệu đề tài

Các cuộc tấn công từ chối dịch vụ phân tán (DDoS) đã trở thành mối đe dọa thực sự đối với mạng, hạ tầng kỹ thuật số, và hạ tầng an ninh mạng. Những cuộc tấn công này có khả năng gây ra sự gián đoạn lớn trong bất kỳ cơ sở hạ tầng công nghệ thông tin và truyền thông (ICT) nào. Có nhiều lý do khác nhau để tiến hành các cuộc tấn công DDoS, bao gồm lợi ích tài chính, lợi ích chính trị, và gây gián đoạn. Các cuộc tấn công DDoS có thể làm tê liệt mạng lưới và dịch vụ bằng cách làm ngập các máy chủ, liên kết mạng, và các thiết bị mạng (bộ định tuyến, bộ chuyển mạch, v.v.) với lưu lượng truy cập không hợp lệ. Chúng có thể gây ra sự suy giảm dịch vụ hoặc từ chối dịch vụ hoàn toàn, dẫn đến tổn thất lớn.

Có nhiều giải pháp độc quyền và mã nguồn mở để phát hiện và giảm thiểu các cuộc tấn công DDoS. Tuy nhiên, những cuộc tấn công này tiếp tục gia tăng về tần suất, độ phức tạp, và mức độ nghiêm trọng. Việc phát hiện và giảm thiểu nhanh chóng các cuộc tấn công DDoS đã trở nên vô cùng thách thức khi kẻ tấn công liên tục sử dụng các kỹ thuật mới để thực hiện tấn công DDoS. Số lượng ngày càng tăng của các cuộc tấn công DDoS, cùng với sự đa dạng ngày càng lớn của chúng, gây ra tác động thảm khốc, đã làm cho việc phát hiện, giảm thiểu và phòng chống tấn công DDoS trở thành ưu tiên hàng đầu.

Ví dụ, Arbor Networks Inc., một trong những nhà cung cấp giải pháp bảo vệ khỏi mối đe dọa DDoS hàng đầu, đã báo cáo một cuộc tấn công 334 Gbps nhắm vào một nhà điều hành mạng ở châu Á gần đây. Ngoài ra, công ty này cũng báo cáo nhiều cuộc tấn công lớn hơn 100 Gbps trên toàn cầu vào năm 2015. Nhiều sự cố như vậy cho thấy rõ rằng chúng ta cần có những phương pháp tiếp cận mới để giải quyết vấn đề tấn công DDoS. Những phương pháp mới này phải được thiết kế để đáp ứng các yêu cầu về hiệu suất và khả năng mở rộng của các trung tâm dữ liệu hiện đại và cung cấp mức độ bảo vệ tối đa chống lại các cuộc tấn công mới nổi, phức tạp và khó nắm bắt.

Với những tiến bộ gần đây trong công nghệ mạng điều khiển bằng phần mềm (SDN) và sự chấp nhận nhanh chóng, rộng rãi của cộng đồng mạng, nhiều nhà nghiên cứu đã tích cực tham gia phát triển các giải pháp bảo mật mạng dựa trên SDN. Các giải pháp dựa trên SDN đã thu hút được nhiều sự chú ý hơn kể từ khi chúng được áp dụng trong các mạng diện rộng quy mô lớn. Công nghệ này cho phép các nhà phát triển lập trình, kiểm soát, và quản lý tài nguyên mạng một cách trực tiếp và tập trung thông qua bộ điều khiển SDN.

Bài báo cáo này sẽ tập trung vào việc nghiên cứu và phân tích phương pháp phát hiện tấn công DDoS bằng mô hình học máy trong môi trường SDN. Chúng ta sẽ xem xét các thuật toán học máy, đánh giá hiệu quả và tính khả thi của chúng trong việc bảo vệ hệ thống mạng SDN trước các mối đe dọa DDoS. Qua đó, chúng ta sẽ đề xuất những hướng đi và giải pháp mới nhằm nâng cao khả năng phòng chống tấn công DDoS trong môi trường mạng hiện đại.

## Mục tiêu thực hiện đề tài

Mục tiêu của đồ án là xây dựng được hệ thống SDN, mô phỏng các cuộc tấn công DDOS và thử nghiệm phương pháp phát hiện và giảm thiểu. Kết quả thực nghiệm cho thấy phương pháp phát hiện và giảm thiểu tấn công DDoS dựa trên Machine Learning kết hợp được đề xuất có tỷ lệ phát hiện tốt đối với tấn công DDoS phổ biến.

# CƠ SỞ LÝ THUYẾT

## Tấn cộng từ chối dịch vụ phân tán (DDoS)

### Định nghĩa

Trước khi định nghĩa về DDoS chúng ta sẽ nói về DoS (Denial of Service). Tấn công DoS là một cuộc tấn công mạng nhằm mục đích làm cho một dịch vụ, hệ thống hoặc mạng trở nên không thể truy cập được bởi người dùng hợp pháp. Điều này được thực hiện bằng cách làm quá tải hệ thống mục tiêu với lưu lượng truy cập hoặc yêu cầu không hợp lệ, khiến hệ thống bị chậm hoặc ngừng hoạt động.

* Phương thức tấn công: Tấn công DoS thường được thực hiện từ một nguồn duy nhất, như một máy tính hoặc một địa chỉ IP.
* Cách thức: Tin tặc gửi một lượng lớn yêu cầu hoặc gói dữ liệu tới mục tiêu, khiến hệ thống không thể xử lý kịp và dẫn đến tình trạng quá tải hoặc lỗi.

Tấn công DDoS là một loại tấn công DoS nhưng với quy mô lớn hơn, được thực hiện từ nhiều nguồn khác nhau. Mục tiêu của tấn công DDoS cũng là làm cho hệ thống, dịch vụ hoặc mạng trở nên không thể truy cập được, nhưng với cách tấn công phân tán từ nhiều máy tính hoặc thiết bị.

* Phương thức tấn công: Tấn công DDoS được thực hiện từ nhiều nguồn khác nhau, thường là từ các mạng botnet bao gồm hàng ngàn hoặc hàng triệu thiết bị bị chiếm quyền điều khiển.
* Cách thức: Tin tặc kiểm soát mạng botnet để gửi lượng lớn lưu lượng truy cập tới mục tiêu, làm cho hệ thống bị quá tải và không thể phục vụ người dùng hợp pháp.

Ví dụ thực tế về tấn công DDoS trong thập kỷ qua:

* Năm 2013: Một cuộc tấn công DDoS nhắm vào dịch vụ DNS của Dyn đã khiến nhiều website và dịch vụ trực tuyến lớn bị sập, bao gồm Amazon, Twitter, Reddit và PayPal.
* Năm 2016: Một cuộc tấn công DDoS nhắm vào Mirai botnet đã tấn công nhà cung cấp dịch vụ Internet (ISP) Dyn DNS, khiến nhiều website và dịch vụ trực tuyến ở Mỹ và châu Âu bị gián đoạn.
* Năm 2017: Một cuộc tấn công DDoS nhắm vào dịch vụ chơi game trực tuyến PlayerUnknown's Battlegrounds (PUBG) đã khiến hàng triệu người chơi bị mất kết nối.
* Năm 2018: Một cuộc tấn công DDoS nhắm vào Google đã sử dụng hơn 100 triệu thiết bị IoT bị nhiễm mã độc để tạo ra lưu lượng truy cập khổng lồ.
* Năm 2020: Một cuộc tấn công DDoS nhắm vào Cloudflare đã sử dụng hơn 15 triệu botnet để tạo ra lưu lượng truy cập khổng lồ, đây là một trong những cuộc tấn công DDoS lớn nhất từng được ghi nhận.

### So sánh DoS và DDoS

Bảng 1 so sánh sự khác nhau cơ bản giữa DoS và DDoS trên một số phương diện khác nhau.

|  |  |  |
| --- | --- | --- |
| **Phương diện** | **DoS** | **DDoS** |
| Khái niệm | Tấn công DoS làm ngập một hệ thống hoặc mạng bằng cách sử dụng một nguồn tấn công duy nhất. | Tấn công DDoS làm ngập một hệ thống hoặc mạng bằng cách sử dụng nhiều nguồn tấn công phân tán. |
| Nguồn tấn công | Một máy tính hoặc thiết bị đơn lẻ. | Nhiều máy tính hoặc thiết bị, thường là một mạng botnet. |
| Quy mô tấn công | Quy mô nhỏ hơn, dễ bị phát hiện và ngăn chặn hơn. | Quy mô lớn hơn, khó phát hiện và ngăn chặn hơn. |
| Khả năng gây thiệt hại | Thường gây thiệt hại hạn chế do sử dụng ít tài nguyên. | Có thể gây thiệt hại nghiêm trọng do sử dụng nhiều tài nguyên và lưu lượng lớn. |
| Phương pháp tấn công | Gửi một lượng lớn yêu cầu hoặc lưu lượng tới mục tiêu từ một nguồn duy nhất. | Gửi một lượng lớn yêu cầu hoặc lưu lượng tới mục tiêu từ nhiều nguồn phân tán. |
| Khả năng truy vết | Dễ dàng truy vết nguồn tấn công do chỉ có một nguồn duy nhất. | Khó truy vết nguồn tấn công do sử dụng nhiều nguồn phân tán. |
| Ví dụ phổ biến | SYN Flood, Ping of Death. | DNS Amplification, HTTP Flood, UDP Flood. |

Bảng 1: Sự khác nhau giữa DoS và DDoS

### Các loại tấn công DDoS

Có nhiều loại tấn công DDoS khác nhau, mỗi loại sử dụng một phương pháp tấn công khác nhau để làm gián đoạn dịch vụ mục tiêu. Dưới đây là một số loại tấn công DDoS phổ biến:

* Tấn công SYN Flood: Tấn công SYN Flood khai thác quy trình bắt tay ba bước trong giao thức TCP. Kẻ tấn công gửi một loạt các yêu cầu SYN (synchronize) đến máy chủ mục tiêu, nhưng không hoàn thành bước bắt tay cuối cùng. Điều này khiến máy chủ duy trì một số lượng lớn các kết nối mở, làm cạn kiệt tài nguyên và từ chối các kết nối mới.
* Tấn công UDP Flood: Tấn công UDP Flood gửi một lượng lớn các gói tin UDP (User Datagram Protocol) không hợp lệ đến các cổng ngẫu nhiên trên máy chủ mục tiêu. Máy chủ phải xử lý mỗi gói tin này và trả lời với một gói tin ICMP "Destination Unreachable" nếu không có ứng dụng nào đang lắng nghe tại các cổng đó, điều này gây quá tải cho máy chủ.
* Tấn công HTTP Flood: Tấn công HTTP Flood mô phỏng các yêu cầu HTTP hợp pháp để làm quá tải máy chủ web mục tiêu. Các yêu cầu này thường đòi hỏi tài nguyên cao như các truy vấn cơ sở dữ liệu phức tạp hoặc tải các tệp lớn, gây ra tình trạng quá tải và làm chậm hoặc ngừng dịch vụ.
* Tấn công DNS Amplification: Tấn công DNS Amplification sử dụng các máy chủ DNS để làm tăng lượng lưu lượng gửi đến mục tiêu. Kẻ tấn công gửi các yêu cầu DNS nhỏ nhưng hợp lệ từ một địa chỉ IP giả mạo (địa chỉ IP của mục tiêu) đến các máy chủ DNS mở. Các máy chủ DNS sau đó trả lời với các phản hồi lớn hơn nhiều, làm ngập mục tiêu với lưu lượng tăng lên.

### Tác động của tấn công DDos

Tấn công DDoS có thể gây ra những hậu quả nghiêm trọng cho các doanh nghiệp và tổ chức, bao gồm:

* Mất dịch vụ: Hệ thống hoặc dịch vụ bị tấn công trở nên không thể truy cập được, ảnh hưởng đến khách hàng và người dùng cuối.
* Mất doanh thu: Các doanh nghiệp có thể mất doanh thu do gián đoạn dịch vụ, đặc biệt là các doanh nghiệp dựa vào dịch vụ trực tuyến.
* Tổn hại uy tín: Các cuộc tấn công DDoS có thể làm tổn hại uy tín của doanh nghiệp hoặc tổ chức nếu khách hàng và đối tác mất niềm tin vào khả năng bảo mật của họ.
* Chi phí khắc phục: Khôi phục từ một cuộc tấn công DDoS có thể tốn kém về thời gian và tiền bạc, bao gồm việc nâng cấp hệ thống và thực hiện các biện pháp bảo mật bổ sung.

### Các phương pháp phát hiện DDoS

Trước sự bùng nổ của các kỹ thuật tấn công DDoS tinh vi trong những năm gần đây, các phương pháp phát hiện tấn công DDoS truyền thống đóng vai trò quan trọng trong việc bảo vệ hệ thống khỏi các mối đe dọa này. Dưới đây là một số phương pháp phổ biến:

* Phân tích thống kê lưu lượng truy cập:
* Theo dõi các chỉ số thống kê của lưu lượng truy cập mạng như số lượng gói tin, byte, tỷ lệ gói tin lỗi, v.v. để phát hiện các thay đổi bất thường có thể báo hiệu tấn công DDoS.
* So sánh lưu lượng truy cập hiện tại với lưu lượng truy cập bình thường trong quá khứ để xác định các đột biến bất thường.
* Phân tích địa chỉ IP:
* Phân tích nguồn gốc địa chỉ IP của lưu lượng truy cập để xác định các địa chỉ IP đáng ngờ hoặc có hoạt động bất thường.
* Sử dụng danh sách đen địa chỉ IP đã biết là nguồn tấn công DDoS để chặn lưu lượng truy cập từ các địa chỉ này.
* Phân tích giao thức:
* Phân tích các đặc điểm của giao thức được sử dụng trong lưu lượng truy cập để phát hiện các mẫu tấn công DDoS phổ biến, chẳng hạn như tấn công SYN flood, tấn công UDP flood, v.v.
* Sử dụng các công cụ phân tích giao thức chuyên dụng để xác định các gói tin độc hại và chặn chúng trước khi chúng đến được máy chủ mục tiêu.
* Giám sát thời gian phản hồi:
* Theo dõi thời gian phản hồi của máy chủ đối với các yêu cầu truy cập để phát hiện các dấu hiệu quá tải do tấn công DDoS.
* Sử dụng các công cụ giám sát hiệu suất để xác định các vấn đề về hiệu suất có thể do tấn công DDoS gây ra.
* Hệ thống phát hiện xâm nhập (IDS):
* Sử dụng hệ thống IDS để theo dõi lưu lượng truy cập mạng và phát hiện các hoạt động đáng ngờ có thể báo hiệu tấn công DDoS.
* Cấu hình IDS để tạo cảnh báo cho quản trị viên hệ thống khi phát hiện các dấu hiệu tấn công DDoS.
* Hệ thống ngăn chặn xâm nhập (IPS):
* Sử dụng hệ thống IPS để tự động chặn lưu lượng truy cập độc hại được IDS phát hiện.
* Cấu hình IPS để chặn các loại tấn công DDoS phổ biến, chẳng hạn như tấn công SYN flood, tấn công UDP flood, v.v.
* Giám sát hoạt động người dùng:
* Theo dõi hành vi của người dùng trên website hoặc ứng dụng để phát hiện các hoạt động đáng ngờ có thể báo hiệu tấn công DDoS.
* Sử dụng các hệ thống phân tích hành vi người dùng để xác định các mẫu truy cập bất thường có thể do botnet hoặc các tác nhân độc hại tạo ra.

## Công nghệ SDN

### Lịch sử và sự ra đời của SDN

Cuối những năm 1990 - đầu 2000:

* Các nghiên cứu ban đầu về SDN bắt đầu từ những năm 1990, khi các nhà nghiên cứu nhận thấy sự hạn chế của mô hình mạng truyền thống trong việc quản lý và điều khiển mạng.
* Dự án "4D" (Data, Discovery, Dissemination, and Decision) tại Đại học Stanford và Berkeley là một trong những nỗ lực nghiên cứu ban đầu để tách biệt chức năng điều khiển khỏi phần cứng mạng.

Năm 2005:

* Dự án Ethane, do Giáo sư Nick McKeown và một nhóm các nhà nghiên cứu tại Đại học Stanford khởi xướng, đã đặt nền móng cho SDN. Ethane tập trung vào việc kiểm soát và quản lý mạng dựa trên các chính sách được định nghĩa phần mềm.

Năm 2008:

* Dự án OpenFlow được khởi xướng, mở đường cho sự phát triển của SDN. OpenFlow là giao thức cho phép các bộ điều khiển mạng tương tác với các switch và router để thiết lập các quy tắc chuyển tiếp dữ liệu. OpenFlow chính là yếu tố then chốt giúp hiện thực hóa khái niệm SDN.

Năm 2011:

* Sự ra đời của Open Networking Foundation (ONF), một tổ chức phi lợi nhuận, nhằm thúc đẩy sự phát triển và tiêu chuẩn hóa của SDN và OpenFlow. ONF được thành lập bởi các công ty công nghệ lớn như Google, Facebook, Microsoft, Verizon, và Yahoo.

### Định nghĩa về SDN

SDN (Software-Defined Networking) là một phương pháp tiếp cận quản lý mạng dựa trên phần mềm. Nó tách biệt các chức năng điều khiển mạng khỏi các thiết bị phần cứng truyền thống như router và switch, thay vào đó sử dụng phần mềm để quản lý và điều khiển mạng.

Về cơ bản, SDN chia tách độc lập hai cơ chế hiện đang tồn tại trong cùng một thiết bị mạng: Cơ chế điều khiển (Control Plane), cơ chế chuyển tiếp dữ liệu (Data Plane) để có thể tối ưu hoạt động của hai cơ chế này.

Control Plane (Mặt phẳng điều khiển)

* Chức năng: Control Plane chịu trách nhiệm ra quyết định về cách các gói dữ liệu sẽ được chuyển tiếp qua mạng. Nó bao gồm các chức năng như định tuyến, định tuyến lại (rerouting), xây dựng bảng định tuyến (routing tables), và các giao thức điều khiển như OSPF, BGP, và RIP.
* Hoạt động: Các thiết bị mạng như router và switch sử dụng Control Plane để trao đổi thông tin về trạng thái mạng, từ đó đưa ra quyết định về đường đi của các gói dữ liệu. Ví dụ, router sử dụng giao thức định tuyến để học biết về các mạng khác và xác định đường đi tốt nhất cho các gói dữ liệu.
* Vị trí: Control Plane thường được tích hợp trong các thiết bị mạng (router, switch). Mỗi thiết bị mạng có Control Plane riêng của nó để quản lý và ra quyết định.

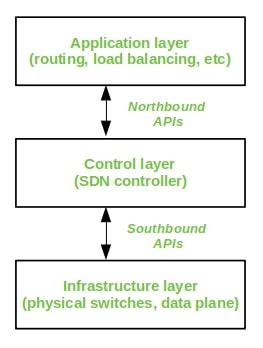
Data Plane (Mặt phẳng dữ liệu)

* Chức năng: Data Plane chịu trách nhiệm chuyển tiếp các gói dữ liệu theo các quyết định được đưa ra bởi Control Plane. Nó thực hiện các hoạt động như truyền tải, định tuyến và lọc gói tin.
* Hoạt động: Khi một gói dữ liệu đến một thiết bị mạng, Data Plane sẽ dựa vào bảng định tuyến (được xây dựng bởi Control Plane) để xác định đường đi của gói tin và chuyển tiếp nó đến đích. Data Plane thực hiện các hoạt động này ở tốc độ cao để đảm bảo hiệu suất mạng.
* Vị trí: Data Plane cũng được tích hợp trong các thiết bị mạng, nơi nó hoạt động song song với Control Plane nhưng với vai trò khác biệt.

Trong các mạng truyền thống, Control Plane và Data Plane thường được tích hợp trong cùng một thiết bị mạng. Tuy nhiên, với sự ra đời của SDN (Software-Defined Networking), hai mặt phẳng này được tách biệt rõ ràng:

* Control Plane: Được chuyển lên các bộ điều khiển trung tâm (controllers), quản lý và ra quyết định cho toàn bộ mạng.
* Data Plane: Vẫn được duy trì trong các thiết bị mạng nhưng hoạt động dưới sự chỉ đạo của Control Plane từ các bộ điều khiển.

### Kiến trúc SDN



Hình 1: Kiến trúc cơ bản của SDN

Kiến trúc SDN bao gồm ba lớp chính:

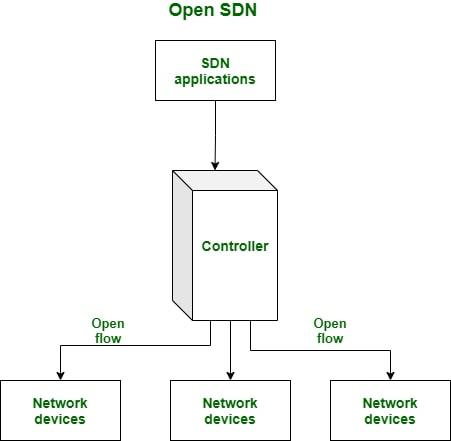
* Lớp hạ tầng mạng (Infrastructure Layer):
  + Bao gồm các thiết bị mạng như switch và router.
  + Chịu trách nhiệm truyền tải dữ liệu.
* Lớp điều khiển (Control Layer):
  + Gồm các bộ điều khiển (controller), đây là nơi diễn ra việc ra quyết định và quản lý toàn bộ mạng.
  + Controller giao tiếp với các thiết bị mạng qua giao diện phía Nam (Southbound Interface), thường sử dụng giao thức OpenFlow.
* Lớp ứng dụng (Application Layer):
  + Chứa các ứng dụng mạng chạy trên đỉnh của controller.
  + Các ứng dụng này cung cấp các chức năng và dịch vụ mạng cụ thể, như quản lý lưu lượng, bảo mật, và ảo hóa mạng.

Các lớp giao tiếp thông qua một tập hợp các giao diện được gọi là northbound APIs và southbound APIs.

### Một số mô hình SDN phổ biến

Open SDN (hình 2): được triển khai bằng cách sử dụng switch OpenFlow. Đây

là một cách triển khai đơn giản của SDN. Trong Open SDN, bộ điều khiển giao tiếp với các switch bằng cách sử dụng southbound APIs thông qua giao thức OpenFlow.



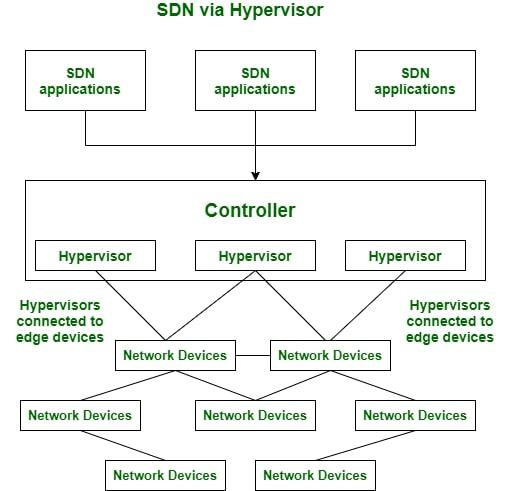
Hình 2: Open SDN

SDN via APIs: các chức năng trong các thiết bị từ xa như switch được gọi

bằng các phương pháp truyền thống như SNMP hoặc CLI hoặc thông qua các phương pháp mới như Rest API. Ở đây, các thiết bị được cung cấp với điểm điều khiển cho phép bộ điều khiển thao tác các thiết bị từ xa bằng cách sử dụng các API.

SDN via Hypervisor-based Overlay Network (hình 3): cấu hình của các thiết

bị vật lý không thay đổi. Thay vào đó, các mạng lớp phủ dựa trên Hypervisor được tạo ra trên mạng vật lý. Chỉ có các thiết bị ở biên của mạng vật lý được kết nối với các mạng ảo hóa, do đó che giấu thông tin của các thiết bị khác trong mạng vật lý.



Hình 3: SDN via Hypervisor-based Overlay Network

Hybrid SDN là sự kết hợp của Mạng truyền thống với Mạng được định nghĩa

bằng phần mềm trong một mạng duy nhất để hỗ trợ các loại chức năng khác nhau trên một mạng.

### So sánh SDN và mạng truyền thống

|  |  |
| --- | --- |
| **Mạng Định Nghĩa Bằng Phần Mềm** | **Mạng Truyền Thống** |
| Mạng Định Nghĩa Bằng Phần Mềm là một cách tiếp cận mạng ảo. | Mạng truyền thống là cách tiếp cận mạng cổ điển. |
| Kiểm soát tập trung. | Kiểm soát phân tán. |
| Có thể lập trình được. | Không thể lập trình được |
| Có giao diện mở. | Có giao diện đóng. |
| Data plane và Control Plane được tách biệt bởi phần mềm. | Data Plane và Control Plane được gắn kết trên cùng một mặt phẳng. |

Bảng 2: So sánh giữa SDN và mạng truyền thống

### Ưu điểm và nhược điểm của SDN

Ưu điểm của SDN:

* Tập trung quản lý và kiểm soát mạng: SDN tách biệt phần điều khiển mạng (control plane) khỏi phần chuyển tiếp dữ liệu (data plane), cho phép quản lý và kiểm soát toàn bộ mạng từ một điểm trung tâm duy nhất.
* Khả năng lập trình cao: Với việc tách biệt phần điều khiển, SDN cung cấp khả năng lập trình mạng, giúp quản lý và cấu hình mạng một cách linh hoạt và hiệu quả hơn.
* Ảo hóa mạng: SDN cho phép ảo hóa tài nguyên mạng vật lý, tạo ra các mạng ảo riêng biệt cho các ứng dụng và dịch vụ khác nhau, tăng cường hiệu quả sử dụng tài nguyên.
* Đổi mới và triển khai nhanh: Nhờ khả năng lập trình, SDN giúp đơn giản hóa việc triển khai và đổi mới các tính năng mạng mới, giảm thời gian và chi phí.
* Hỗ trợ điện toán đám mây: SDN là một công nghệ quan trọng trong việc hỗ trợ và cải thiện các dịch vụ điện toán đám mây, cho phép quản lý tài nguyên mạng một cách hiệu quả hơn.
* An ninh mạng tăng cường: Với khả năng giám sát và kiểm soát tập trung, SDN có thể được sử dụng để triển khai các giải pháp an ninh mạng hiệu quả hơn.

Nhược điểm của SDN:

* Phụ thuộc vào bộ điều khiển trung tâm: SDN phụ thuộc vào bộ điều khiển trung tâm để quản lý và kiểm soát mạng. Nếu bộ điều khiển bị lỗi hoặc tấn công, toàn bộ mạng sẽ bị ảnh hưởng.
* Vấn đề tương thích và di chuyển: Việc chuyển đổi từ mạng truyền thống sang SDN có thể gặp phải vấn đề tương thích và khó khăn trong di chuyển.
* Yêu cầu về nhân lực có kỹ năng chuyên môn: Triển khai và vận hành SDN đòi hỏi nhân lực có kỹ năng chuyên môn cao về lập trình mạng và quản lý hệ thống phức tạp.
* Rủi ro về bảo mật: Tập trung điều khiển mạng tại một điểm duy nhất cũng làm tăng nguy cơ bị tấn công và xâm nhập bảo mật.
* Chi phí triển khai ban đầu cao: Chi phí triển khai SDN ban đầu có thể cao hơn so với mạng truyền thống, đặc biệt là đối với các tổ chức quy mô lớn.

Mặc dù có một số nhược điểm, nhưng SDN vẫn được coi là một công nghệ mạng tiên tiến và có tiềm năng lớn trong tương lai. Mạng định nghĩa bằng phần mềm (Software-Defined Networking - SDN) đã mở ra nhiều ứng dụng tiềm năng trong các lĩnh vực khác nhau. Một số ứng dụng quan trọng của SDN bao gồm:

* Quản lý và kiểm soát mạng tập trung: SDN cho phép quản lý và kiểm soát toàn bộ mạng từ một điểm trung tâm duy nhất, điều này giúp đơn giản hóa việc triển khai và cấu hình mạng, cũng như nâng cao hiệu quả quản lý.
* Ảo hóa mạng: Nhờ khả năng lập trình của SDN, các tài nguyên mạng vật lý có thể được ảo hóa và cấp phát một cách linh hoạt cho các ứng dụng và dịch vụ khác nhau, tăng cường hiệu quả sử dụng tài nguyên.
* Điện toán đám mây: SDN là một công nghệ quan trọng trong việc hỗ trợ và cải thiện các dịch vụ điện toán đám mây, cho phép quản lý tài nguyên mạng một cách hiệu quả và linh hoạt hơn.
* An ninh mạng: Với khả năng giám sát và kiểm soát tập trung, SDN có thể được sử dụng để triển khai các giải pháp an ninh mạng hiệu quả hơn, như phát hiện và ngăn chặn các cuộc tấn công mạng, triển khai tường lửa và chính sách kiểm soát truy cập.
* Internet of Things (IoT): Với khả năng quản lý và cấu hình mạng linh hoạt, SDN có thể đóng một vai trò quan trọng trong việc kết nối và quản lý các thiết bị IoT một cách hiệu quả.
* Mạng doanh nghiệp: SDN cung cấp khả năng quản lý và kiểm soát tập trung, giúp đơn giản hóa việc triển khai và vận hành mạng doanh nghiệp, đồng thời tăng cường an ninh và hiệu quả sử dụng tài nguyên.
* Mạng trung tâm dữ liệu: Trong các trung tâm dữ liệu lớn, SDN giúp tối ưu hóa việc sử dụng tài nguyên mạng, nâng cao hiệu năng và khả năng mở rộng của mạng.

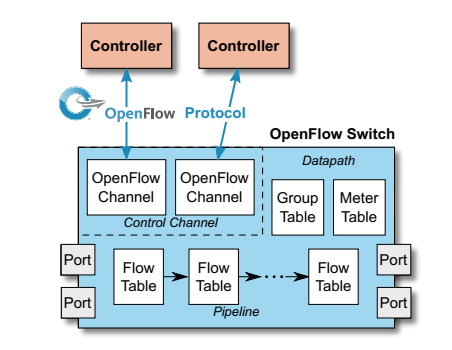
## Giao thức OpenFlow

Giao thức OpenFlow là một thành phần quan trọng trong kiến trúc Mạng điều khiển bằng phần mềm (SDN - Software-Defined Networking). OpenFlow cho phép các bộ điều khiển SDN trực tiếp tương tác với tầng dữ liệu của các thiết bị mạng như switch và router, để quản lý luồng dữ liệu trên mạng.

### Giới thiệu OpenFlow

OpenFlow là giao thức mở chuẩn hóa được thiết kế bởi nhóm Open Networking Foundation (ONF). Nó cho phép sự tách rời của tầng điều khiển (control plane) và tầng dữ liệu (data plane), một khái niệm cốt lõi của SDN. Tầng điều khiển quyết định đường đi của dữ liệu, trong khi tầng dữ liệu chịu trách nhiệm chuyển tiếp dữ liệu dựa trên những quyết định đó.

### Kiến trúc OpenFlow



Hình 4: Các thành phần chính của OpenFlow

Kiến trúc OpenFlow bao gồm các thành phần chính sau:

* Controller (Bộ điều khiển): Là một hệ thống phần mềm trung tâm quản lý và điều khiển các thiết bị mạng bằng cách sử dụng giao thức OpenFlow.
* OpenFlow Switch: Là thiết bị phần cứng hoặc phần mềm hỗ trợ giao thức OpenFlow, có chức năng chuyển tiếp dữ liệu dựa trên các quy tắc (flow rules) được thiết lập bởi Controller.
* Flow Table (Bảng luồng): Mỗi switch OpenFlow có một hoặc nhiều bảng luồng, chứa các quy tắc để xác định cách xử lý các gói tin.
* Secure Channel (Kênh bảo mật): Là kênh giao tiếp an toàn giữa switch và Controller, thường sử dụng TLS (Transport Layer Security) để bảo mật.

### Hoạt động của OpenFlow

Quy trình hoạt động của OpenFlow có thể được mô tả như sau:

* Thiết lập Kết nối: Switch OpenFlow khởi tạo một kết nối bảo mật với Controller thông qua kênh bảo mật.
* Trao đổi Thông tin: Switch gửi thông tin về các cổng (ports) và khả năng của mình tới Controller. Controller có thể sử dụng thông tin này để xây dựng các bảng luồng và cấu hình switch.
* Quản lý Luồng: Khi một gói tin mới đến switch mà không có quy tắc nào trong bảng luồng để xử lý nó, gói tin sẽ được gửi đến Controller. Controller sẽ quyết định cách xử lý gói tin đó và cài đặt một quy tắc mới vào bảng luồng của switch.
* Chuyển tiếp Gói Tin: Dựa trên các quy tắc trong bảng luồng, switch sẽ chuyển tiếp các gói tin theo các hành động được xác định trước (ví dụ: chuyển tiếp, bỏ qua, thay đổi tiêu đề, v.v.).

### Lợi ích của OpenFlow

Một số lợi ích chính của OpenFlow:

* Linh hoạt và Tự động hóa: Dễ dàng triển khai các chính sách mạng và tự động hóa quản lý mạng.
* Hiệu suất Cao: Giảm tải cho các switch bằng cách tập trung các quyết định điều khiển tại Controller.
* Khả năng Mở Rộng: Dễ dàng thêm vào hoặc thay đổi các quy tắc chuyển tiếp mà không cần thay đổi phần cứng.
* Tối ưu Hóa Tài Nguyên: Sử dụng tài nguyên mạng hiệu quả hơn bằng cách điều khiển lưu lượng mạng theo thời gian thực.

### Ứng dụng của OpenFlow

Các ứng dụng của OpenFlow trong thực tiễn bao gồm:

* Data Center Networking: Tối ưu hóa lưu lượng dữ liệu trong các trung tâm dữ liệu.
* Wide Area Network (WAN): Quản lý và tối ưu hóa lưu lượng mạng diện rộng.
* Network Virtualization: Tạo ra các mạng ảo hóa để thử nghiệm hoặc cách ly các ứng dụng khác nhau.
* Network Security: Áp dụng các chính sách bảo mật động và tự động dựa trên lưu lượng mạng thực tế.

## Mô hình K-Nearest Neighbor

### Lịch sử hình thành

Thuật toán K-Nearest Neighbors (KNN) là một trong những thuật toán đơn giản nhất trong học máy, đặc biệt trong lĩnh vực phân loại và hồi quy. KNN được phát triển và xuất bản lần đầu tiên bởi Evelyn Fix và Joseph Hodges vào năm 1951, trong bài báo "Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties." Thuật toán này ban đầu được sử dụng chủ yếu trong lĩnh vực thống kê để phân tích và phân loại dữ liệu.

### Ý tưởng cơ bản của thuật toán KNN

KNN dựa trên một nguyên tắc đơn giản: các đối tượng gần nhau có xu hướng có các thuộc tính giống nhau. Trong bối cảnh phân loại, nguyên tắc này có thể được diễn giải như sau:

* + Dữ liệu Huấn luyện và Dữ liệu Kiểm tra: Dữ liệu huấn luyện là một tập hợp các điểm dữ liệu có nhãn đã biết. Dữ liệu kiểm tra là những điểm dữ liệu mới mà ta cần phân loại hoặc dự đoán giá trị.
  + Khoảng cách: Để xác định các điểm lân cận, ta cần một cách để đo khoảng cách giữa các điểm dữ liệu. Khoảng cách Euclid là lựa chọn phổ biến nhất, nhưng có thể sử dụng các khoảng cách khác như khoảng cách Manhattan, khoảng cách Minkowski, hay khoảng cách Hamming.
  + K (Nearest Neighbors): Số K xác định số lượng điểm dữ liệu lân cận gần nhất mà ta sẽ xem xét để phân loại một điểm dữ liệu mới. Ví dụ, nếu K=3, thuật toán sẽ xem xét ba điểm dữ liệu lân cận gần nhất.
  + Phân loại: Sau khi xác định K điểm lân cận gần nhất, thuật toán sẽ tiến hành phân loại dựa trên nhãn của các điểm lân cận này. Nhãn của điểm mới sẽ được xác định bằng cách sử dụng phương pháp bỏ phiếu đa số từ nhãn của K điểm lân cận.

### Ví dụ về thuật toán KNN

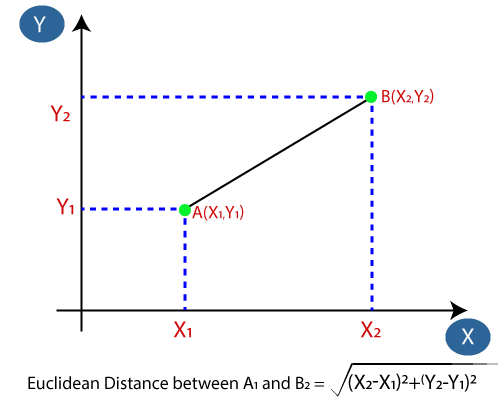


Hình 5

Giả sử chúng ta có tập dữ liệu (Hình 5) gồm hai loại: A và B. Khi một điểm dữ liệu mới được thêm vào cần được phân loại là A hay B.

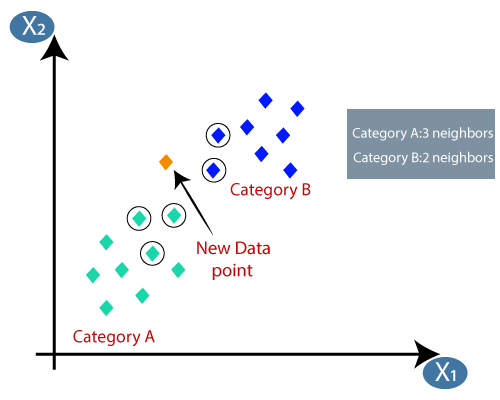
Đầu tiên, chọn số lượng hàng xóm ‘K’, ví dụ K=5

Tiếp theo, chúng ta sẽ tính khoảng cách Euclide giữa các điểm dữ liệu. Khoảng cách Euclide là khoảng cách giữa hai điểm mà chúng ta đã nghiên cứu trong hình học. Nó có thể được tính như sau:



Hình 6

Bằng cách tính khoảng cách Euclide, chúng ta có được những người hàng xóm gần nhất, như ba người hàng xóm gần nhất thuộc loại A và hai người hàng xóm gần nhất thuộc loại B.



Hình 7

Như chúng ta có thể thấy 3 người hàng xóm gần nhất thuộc loại A, do đó điểm dữ liệu mới này phải thuộc loại A.

### Các nguyên tắc chọn k cơ bản

Giá trị k nhỏ (ví dụ: k = 1 hoặc k = 3): Mô hình sẽ rất nhạy cảm với dữ liệu huấn luyện và dễ dẫn đến hiện tượng quá khớp. Điều này có nghĩa là mô hình có thể ghi nhớ các điểm nhiễu trong dữ liệu.

Giá trị k lớn (ví dụ: k > 20): Mô hình sẽ trở nên quá tổng quát và có thể dẫn đến hiện tượng thiếu khớp. Điều này có nghĩa là mô hình không thể nắm bắt được các mẫu trong dữ liệu.

Chọn giá trị lẻ cho k: Trong các bài toán phân loại nhị phân, chọn giá trị lẻ cho k giúp tránh tình trạng hòa (tie) khi bỏ phiếu nhãn.

### Ưu điểm và Nhược điểm của KNN

Ưu điểm:

* Dễ hiểu và dễ triển khai.
* Hiệu quả trong các bài toán với dữ liệu có cấu trúc rõ ràng và không quá phức tạp.
* Không cần quá trình huấn luyện, chỉ cần tính toán khi phân loại.

Nhược điểm:

* Tốc độ chậm khi số lượng điểm dữ liệu lớn, do phải tính toán khoảng cách cho tất cả các điểm trong tập huấn luyện.
* Hiệu suất kém với dữ liệu có nhiều nhiễu hoặc nhiều chiều (high-dimensional data).
* Yêu cầu bộ nhớ lớn để lưu trữ toàn bộ dữ liệu huấn luyện.

### Ứng dụng của KNN

Một số ứng dụng thực tế của KNN:

* Nhận dạng mẫu: Nhận dạng chữ viết tay, nhận diện khuôn mặt.
* Phát hiện bất thường: Phát hiện gian lận, bất thường trong dữ liệu cảm biến.
* Y học: Chẩn đoán bệnh dựa trên các triệu chứng và chỉ số sinh học.

## Ryu controller

### Giới thiệu về Ryu Controller

Ryu là một framework mã nguồn mở được thiết kế để phát triển các ứng dụng điều khiển mạng SDN (Software-Defined Networking). Được viết bằng ngôn ngữ lập trình Python, Ryu cung cấp một bộ công cụ và thư viện phong phú để phát triển các ứng dụng điều khiển mạng một cách nhanh chóng và dễ dàng.

Ryu hỗ trợ nhiều giao thức điều khiển mạng khác nhau, bao gồm OpenFlow (từ phiên bản 1.0 đến 1.5), Netconf, OF-config và nhiều giao thức khác. Điều này giúp Ryu trở thành một công cụ linh hoạt và mạnh mẽ trong việc triển khai và quản lý mạng SDN.

### Kiến trúc của Ryu Controller

Ryu Controller được thiết kế với kiến trúc module hóa, cho phép các nhà phát triển dễ dàng thêm, xóa hoặc tùy chỉnh các chức năng của nó. Dưới đây là các thành phần chính của Ryu:

* Core Components (Thành phần lõi): Bao gồm các thư viện và công cụ cơ bản để quản lý và điều khiển mạng. Thành phần lõi của Ryu chịu trách nhiệm xử lý các yêu cầu từ các thiết bị mạng và điều khiển chúng theo các quy tắc và chính sách được định nghĩa.
* Application Components (Thành phần ứng dụng): Đây là nơi mà các ứng dụng mạng cụ thể được triển khai. Các ứng dụng này có thể bao gồm các chức năng như định tuyến, quản lý băng thông, phát hiện tấn công DDoS, v.v. Các ứng dụng này tương tác với các thành phần lõi của Ryu để thực hiện các tác vụ điều khiển mạng.
* API (Giao diện lập trình ứng dụng): Ryu cung cấp các API phong phú cho phép các nhà phát triển tương tác với controller và các thiết bị mạng. API của Ryu được thiết kế theo hướng RESTful, giúp dễ dàng tích hợp với các ứng dụng và dịch vụ khác.

### Hoạt động của Ryu Controller

Ryu Controller hoạt động bằng cách nhận và xử lý các sự kiện mạng từ các thiết bị mạng, sau đó thực hiện các hành động điều khiển dựa trên các sự kiện này. Quá trình hoạt động của Ryu có thể được mô tả qua các bước sau:

* Kết nối và đăng ký: Các thiết bị mạng (như OpenFlow switch) kết nối với Ryu Controller thông qua giao thức OpenFlow. Khi kết nối được thiết lập, các thiết bị này đăng ký với Ryu Controller.
* Nhận sự kiện: Ryu Controller lắng nghe và nhận các sự kiện mạng từ các thiết bị đã đăng ký. Các sự kiện này có thể bao gồm các gói tin mới, thay đổi trạng thái của các cổng, và các sự kiện khác.
* Xử lý sự kiện: Ryu Controller sử dụng các ứng dụng và các module để xử lý các sự kiện nhận được. Các ứng dụng này có thể áp dụng các quy tắc và chính sách điều khiển để quyết định hành động cần thực hiện.
* Thực hiện hành động: Dựa trên kết quả xử lý sự kiện, Ryu Controller gửi các lệnh điều khiển đến các thiết bị mạng. Các lệnh này có thể bao gồm cài đặt các quy tắc mới trong flow table, thay đổi cấu hình của các cổng, hoặc các hành động khác.

### Ưu điểm của Ryu Controller

Mã nguồn mở: Ryu là một framework mã nguồn mở, cho phép các nhà phát triển tự do sửa đổi và tùy chỉnh theo nhu cầu cụ thể.

Hỗ trợ nhiều giao thức: Ryu hỗ trợ nhiều phiên bản của OpenFlow cũng như các giao thức khác, giúp nó linh hoạt trong việc triển khai mạng SDN.

Dễ dàng phát triển và mở rộng: Kiến trúc module hóa của Ryu cho phép các nhà phát triển dễ dàng thêm, xóa hoặc tùy chỉnh các chức năng.

Tài liệu phong phú: Ryu đi kèm với tài liệu chi tiết và nhiều ví dụ mã nguồn, giúp các nhà phát triển dễ dàng học và triển khai.

### Ứng dụng của Ryu Controller trong phát hiện và ngăn chặn tấn công DDoS

Ryu Controller có thể được sử dụng để phát hiện và ngăn chặn các cuộc tấn công DDoS thông qua việc triển khai các ứng dụng mạng cụ thể. Ví dụ:

* Giám sát lưu lượng mạng: Sử dụng Ryu để giám sát lưu lượng mạng và phát hiện các mẫu lưu lượng bất thường có thể là dấu hiệu của một cuộc tấn công DDoS.
* Áp dụng các quy tắc lọc: Triển khai các quy tắc lọc để chặn các gói tin đáng ngờ hoặc đến từ các địa chỉ IP bị nghi ngờ.
* Điều chỉnh băng thông: Điều chỉnh băng thông và giới hạn tốc độ cho các nguồn lưu lượng đáng ngờ để giảm thiểu tác động của tấn công DDoS.

## Mininet

### Giới thiệu về Mininet

Mininet là một trình giả lập mạng mã nguồn mở, được sử dụng rộng rãi để tạo ra các mạng SDN (Software-Defined Networking) trong môi trường ảo hóa. Mininet cho phép người dùng tạo ra các topologies mạng phức tạp với các nút mạng ảo (hosts), switches và controllers chỉ bằng một máy tính duy nhất. Điều này giúp cho việc phát triển, kiểm thử và triển khai các ứng dụng mạng SDN trở nên dễ dàng và tiết kiệm chi phí.

Mininet đặc biệt hữu ích trong các tình huống cần kiểm thử và phát triển mạng mà không cần tới phần cứng đắt tiền. Nó cung cấp một môi trường thực tế để kiểm tra các giao thức mạng, các thuật toán điều khiển và các chính sách bảo mật trước khi triển khai vào môi trường thực tế.

### Kiến trúc của Mininet

Mininet sử dụng các công nghệ ảo hóa nhẹ như Linux Containers (LXC) và network namespaces để tạo ra các nút mạng ảo hóa, mà mỗi nút hoạt động như một thiết bị mạng thực. Các thành phần chính của Mininet bao gồm:

* Hosts (Các nút máy chủ): Các host ảo trong Mininet hoạt động như các máy tính thật, có thể chạy các ứng dụng mạng, gửi và nhận gói tin.
* Switches (Các thiết bị chuyển mạch): Mininet hỗ trợ cả Open vSwitch và các switch ảo khác, cho phép triển khai các giao thức SDN như OpenFlow.
* Controllers (Các bộ điều khiển): Mininet có thể kết nối với các bộ điều khiển SDN bên ngoài (như Ryu, OpenDaylight) hoặc sử dụng các bộ điều khiển tích hợp.
* Links (Các kết nối): Mininet tạo ra các kết nối mạng ảo giữa các host và switch, mô phỏng các kết nối mạng thực tế.

### Cách hoạt động của Mininet

Mininet hoạt động bằng cách sử dụng các công nghệ ảo hóa của Linux để tạo ra một mạng ảo trên một máy tính duy nhất. Quá trình hoạt động của Mininet có thể được mô tả qua các bước sau:

* Khởi tạo Topology: Người dùng định nghĩa topology mạng bằng cách sử dụng Python API của Mininet. Topology này bao gồm các host, switch và các liên kết giữa chúng.
* Tạo Nút Mạng Ảo: Mininet sử dụng các container Linux và network namespaces để tạo ra các host và switch ảo.
* Thiết lập Kết Nối: Mininet tạo ra các kết nối mạng ảo giữa các nút, mô phỏng các kết nối vật lý trong mạng thực.
* Kết Nối Bộ Điều Khiển: Nếu cần, Mininet kết nối các switch ảo với bộ điều khiển SDN bên ngoài để điều khiển mạng.
* Chạy Thử Nghiệm: Người dùng có thể chạy các thử nghiệm mạng, gửi và nhận gói tin, kiểm tra hiệu suất và hành vi của mạng ảo.

### Lợi ích của Mininet

Tiết kiệm chi phí: Mininet cho phép tạo ra các mạng ảo phức tạp mà không cần đầu tư vào phần cứng mạng đắt tiền.

Kiểm thử linh hoạt: Mininet cung cấp môi trường kiểm thử linh hoạt và có thể cấu hình, giúp phát triển và kiểm tra các ứng dụng mạng một cách nhanh chóng.

Dễ sử dụng: Mininet có giao diện đơn giản và API Python thân thiện, giúp dễ dàng định nghĩa và quản lý các topologies mạng.

Khả năng mô phỏng thực tế: Mininet mô phỏng chính xác hành vi của mạng thực, giúp phát hiện và khắc phục các vấn đề mạng trước khi triển khai.

## Wireshark

### Giới thiệu về Wireshark

Wireshark là một phần mềm phân tích giao thức mạng mã nguồn mở, mạnh mẽ và phổ biến nhất trên thế giới. Nó cho phép người dùng xem chi tiết từng gói tin lưu thông qua mạng. Với Wireshark, các nhà quản trị mạng, chuyên gia bảo mật và nhà phát triển có thể phân tích lưu lượng mạng để phát hiện lỗi, phân tích hiệu suất, và xác định các hoạt động bất thường.

### Lịch sử phát triển

Wireshark, ban đầu được gọi là Ethereal, được phát triển bởi Gerald Combs vào năm 1998. Đến năm 2006, do vấn đề về thương hiệu, phần mềm được đổi tên thành Wireshark. Wireshark đã nhận được sự đóng góp từ nhiều nhà phát triển trên khắp thế giới, giúp nó trở thành công cụ mạnh mẽ và linh hoạt như ngày nay.

### Chức năng chính của Wireshark

Phân tích gói tin: Wireshark cho phép bắt và hiển thị các gói tin lưu thông qua mạng. Nó hỗ trợ hầu hết các giao thức mạng hiện đại và có thể giải mã chúng để hiển thị thông tin chi tiết.

Lọc gói tin: Người dùng có thể sử dụng các bộ lọc mạnh mẽ của Wireshark để chỉ hiển thị các gói tin quan tâm, giúp tập trung vào các vấn đề cụ thể.

Theo dõi phiên làm việc: Wireshark có thể theo dõi các phiên làm việc của giao thức, giúp người dùng hiểu rõ hơn về quá trình truyền thông giữa các thiết bị.

Phân tích thống kê: Wireshark cung cấp các công cụ thống kê để phân tích lưu lượng mạng, giúp nhận diện các xu hướng và vấn đề.

### Kiến trúc và hoạt động của Wireshark

Wireshark bao gồm ba thành phần chính:

* Packet Capture Library (Thư viện bắt gói tin): Thường là libpcap trên Unix/Linux và WinPcap trên Windows, cho phép Wireshark bắt gói tin từ các giao diện mạng.
* Packet Analyzer (Bộ phân tích gói tin): Bộ phân tích này giải mã và hiển thị chi tiết nội dung của các gói tin dựa trên các giao thức mà chúng sử dụng.
* User Interface (Giao diện người dùng): Giao diện đồ họa của Wireshark cho phép người dùng tương tác, lọc và phân tích các gói tin một cách dễ dàng.

### Sử dụng Wireshark trong thực tế

Chẩn đoán mạng: Các nhà quản trị mạng sử dụng Wireshark để chẩn đoán và khắc phục sự cố mạng, xác định các điểm tắc nghẽn và các lỗi cấu hình.

Bảo mật mạng: Các chuyên gia bảo mật sử dụng Wireshark để phân tích lưu lượng mạng nhằm phát hiện các hoạt động bất thường hoặc các cuộc tấn công mạng.

Phát triển phần mềm: Các nhà phát triển sử dụng Wireshark để kiểm tra và gỡ lỗi các ứng dụng mạng, đảm bảo rằng chúng hoạt động đúng và hiệu quả.

### Lợi ích của Wireshark

Mã nguồn mở: Wireshark là phần mềm mã nguồn mở và miễn phí, cho phép bất kỳ ai cũng có thể sử dụng và đóng góp vào sự phát triển của nó.

Hỗ trợ rộng rãi: Wireshark hỗ trợ hầu hết các giao thức mạng và có thể chạy trên nhiều hệ điều hành khác nhau.

Mạnh mẽ và linh hoạt: Wireshark cung cấp các công cụ mạnh mẽ để phân tích và lọc gói tin, giúp người dùng nhanh chóng tìm ra các vấn đề mạng.

## Hping

### Giới thiệu về Hping

Hping là một công cụ dòng lệnh mã nguồn mở dùng để tạo và phân tích các gói tin mạng TCP/IP. Được thiết kế bởi Salvatore Sanfilippo, Hping là một công cụ mạnh mẽ cho việc kiểm tra bảo mật mạng, chẩn đoán lỗi mạng, và nghiên cứu các giao thức mạng. Nó hoạt động tương tự như lệnh ping nhưng có thể làm việc với nhiều giao thức hơn và cung cấp nhiều tính năng linh hoạt hơn.

### Các tính năng chính của Hping

Gửi gói tin tùy chỉnh: Hping cho phép người dùng tạo và gửi các gói tin TCP, UDP, ICMP, và RAW IP với các trường tiêu đề tùy chỉnh.

Kiểm tra kết nối TCP: Hping có thể được sử dụng để kiểm tra các cổng TCP mở, tương tự như công cụ Nmap.

Tấn công mạng: Hping có thể thực hiện các cuộc tấn công mạng như DoS, DDoS, và tấn công SYN flood để kiểm tra tính bảo mật của hệ thống.

Traceroute: Hping hỗ trợ thực hiện traceroute để xác định đường đi của các gói tin trong mạng.

Chẩn đoán lỗi mạng: Hping giúp kiểm tra và phân tích các vấn đề mạng bằng cách gửi các gói tin tùy chỉnh và xem phản hồi từ các thiết bị mạng.

### Ứng dụng của Hping

Kiểm tra bảo mật: Hping được sử dụng để thực hiện các cuộc kiểm tra bảo mật mạng như quét cổng, tấn công DoS, DDoS, và SYN flood.

Chẩn đoán mạng: Hping giúp xác định và phân tích các vấn đề mạng bằng cách tạo và gửi các gói tin tùy chỉnh.

Nghiên cứu và phát triển: Hping là công cụ hữu ích cho việc nghiên cứu các giao thức mạng và phát triển các ứng dụng mạng.

### Ưu điểm và hạn chế của Hping

Ưu điểm:

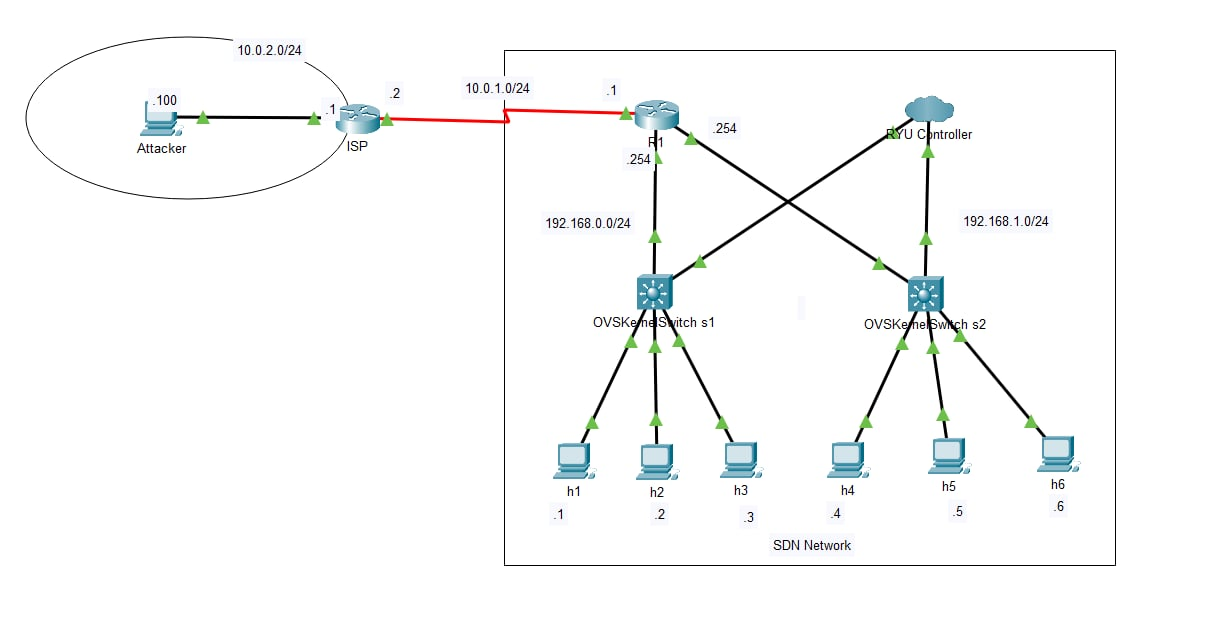
* Linh hoạt: Hping cho phép tạo và gửi các gói tin với các trường tiêu đề tùy chỉnh, giúp linh hoạt trong việc kiểm tra và phân tích mạng.
* Mã nguồn mở: Là phần mềm mã nguồn mở, Hping cho phép người dùng sửa đổi và tùy chỉnh theo nhu cầu cụ thể.
* Nhiều tính năng: Hping cung cấp nhiều tính năng mạnh mẽ cho việc kiểm tra bảo mật, chẩn đoán mạng và nghiên cứu các giao thức mạng.

Hạn chế:

* Phức tạp: Sử dụng Hping có thể đòi hỏi kiến thức chuyên sâu về mạng và các giao thức TCP/IP, điều này có thể gây khó khăn cho người mới bắt đầu.
* Tiềm ẩn nguy cơ bảo mật: Hping có thể được sử dụng cho các mục đích xấu như tấn công DoS, DDoS, do đó cần sử dụng cẩn thận và có trách nhiệm.

# THỰC NGHIỆM

## Thiết kế hệ thống



Hình 8: Thiết kế hệ thống SDN

## Thu thập dữ liệu

### Mô tả các bước thu thập dữ liệu



Hình 9: Sơ đồ quá trình thu thập dataset

Hình 9 mô tả quá trình thu thập dữ liệu để xây dựng dataset, dữ liệu được thu thập trong hai loại lưu lượng là bình thường và lưu lượng trong trường hợp tấn công DDoS. Dữ liệu thu thập từ hai loại lưu lượng được tiến hành trích xuất đặc trưng , nhãn 0 sẽ được gán cho lưu lượng bình thường, nhãn 1 sẽ được gán cho trường hợp có DDoS. Dataset này sau đó được sử dụng để huấn luyện mô hình máy học sử dụng thuật toán KNN đã nhắc đến ở chương 2. Mô hình được huấn luyện có thể được sử dụng để phân loại và phát hiện tấn công DDoS dựa trên dữ liệu lưu lượng mạng. Quy trình này thể hiện một hệ thống phát hiện tấn công DDoS bằng cách sử dụng kỹ thuật học máy có giám sát, trong đó dữ liệu bình thường và dữ liệu tấn công được gán nhãn khác nhau để huấn luyện mô hình phân loại.

### Mô tả dữ liệu

Dữ liệu sau khi thu thập được lưu trữ tại:

<https://drive.google.com/file/d/18FvTV4Oe_2bYOmXdSi16d5pXhAhGaypx/view?usp=drive_link>

Mô tả:

* Tên file: DDos\_and\_Normal\_Traffic\_Dataset.csv.
* Kích thước: 112.4MB.
* Cấu trúc: 544993 dòng, 28 cột.
* Các đặc trưng được mô tả trong bảng 3.

|  |  |
| --- | --- |
| **Tên trường** | **Mô tả** |
| timestamp | Thời gian thu thập thống kê, ở định dạng timestamp (số giây từ thời điểm bắt đầu của epoch). |
| datapath\_id | ID của datapath (switch OpenFlow) mà thống kê được thu thập từ đó. |
| flow\_id | ID được tính toán dựa trên thông tin của luồng, bao gồm địa chỉ IP nguồn, cổng nguồn, địa chỉ IP đích, cổng đích và giao thức IP. |
| ip\_src | Địa chỉ IP nguồn của luồng. |
| tp\_src | Cổng nguồn của luồng (TCP hoặc UDP). |
| ip\_dst | Địa chỉ IP đích của luồng. |
| tp\_dst | Cổng đích của luồng (TCP hoặc UDP). |
| ip\_proto | Giao thức IP của luồng (ví dụ: 1 cho ICMP, 6 cho TCP, 17 cho UDP). |
| icmp\_code | Mã ICMP (nếu giao thức IP là ICMP). |
| icmp\_type | Loại ICMP (nếu giao thức IP là ICMP). |
| flow\_duration\_sec | Thời gian tồn tại của luồng (tính bằng giây). |
| flow\_duration\_nsec | Thời gian tồn tại của luồng (tính bằng nano giây). |
| idle\_timeout | Giá trị idle timeout của luồng (thời gian tối đa luồng không hoạt động trước khi bị xóa). |
| hard\_timeout | Giá trị hard timeout của luồng (thời gian tối đa luồng tồn tại trước khi bị xóa). |
| flags | Các cờ liên quan đến luồng. |
| packet\_count | Số lượng gói tin trong luồng. |
| byte\_count | Lượng dữ liệu (tính bằng byte) trong luồng. |
| packet\_count\_per\_second | Tốc độ gói tin mỗi giây của luồng. |
| packet\_count\_per\_nsecond | Tốc độ gói tin mỗi nano giây của luồng. |
| byte\_count\_per\_second | Tốc độ dữ liệu mỗi giây của luồng (tính bằng byte/giây). |
| byte\_count\_per\_nsecond | Tốc độ dữ liệu mỗi nano giây của luồng (tính bằng byte/nano giây). |
| label | Được sử dụng để gán nhãn cho các luồng khi sử dụng dữ liệu cho mục đích huấn luyện mô hình ML. |

Bảng 3: Mô tả các đặc trưng của dataset

## Đánh giá mô hình KNN

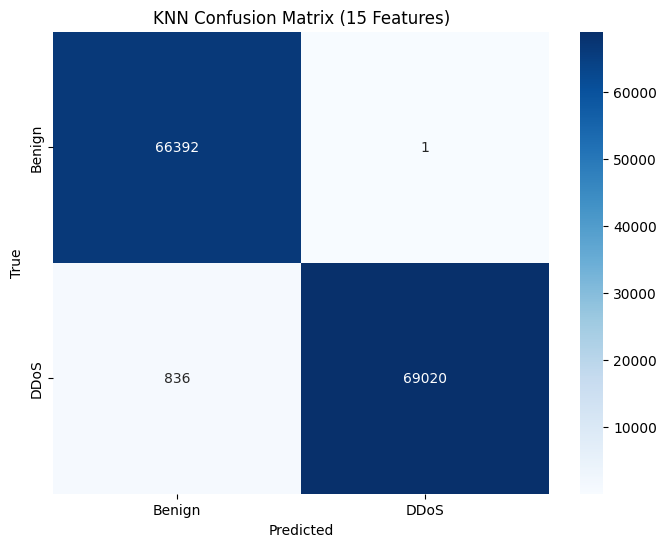
|  |  |  |  |
| --- | --- | --- | --- |
|  | Label 0 (Normal) | Label 1 (DDoS) | Tổng |
| Tập train | 199378 | 209366 | 408744 |
| Tập test | 66460 | 69789 | 136249 |

Bảng 4: Chia tập dữ liệu thành train và test

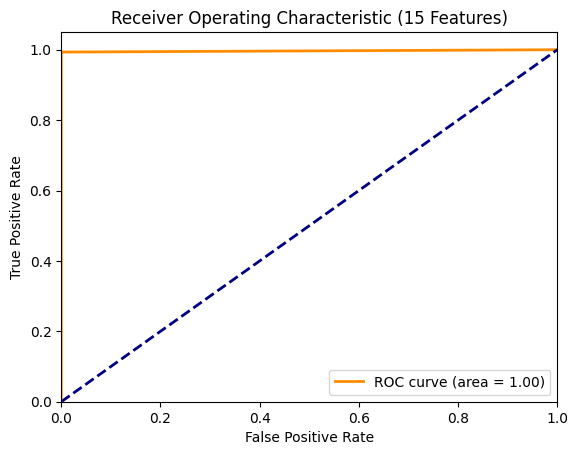
Dataset được chia thành tập train (huấn luyện) và tập test (kiểm tra) với tỉ lệ 75% và 25%. Điều này mang lại một số lợi ích, bao gồm việc cân bằng giữa huấn luyện và kiểm tra, giảm nguy cơ overfitting, đánh giá tổng quát hoá mô hình.

|  |  |
| --- | --- |
| Thông tin | Giá trị |
| Tên mô hình | KNN |
| Thời gian train | 59,58 giây |
| Accuracy Score | 99.39% |
| Precision Score | 99.39% |
| Recall Score | 99.39% |
| F1 score | 99.39% |
| AUC | 0.996657 |

Bảng 5: Bảng đánh giá mô hình



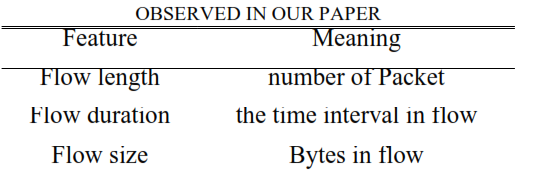
Hình 10: Ma trận nhầm lẫn



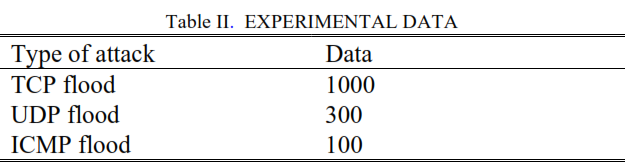
Hình 11: Biểu đồ ROC

So sánh với bài báo [1]

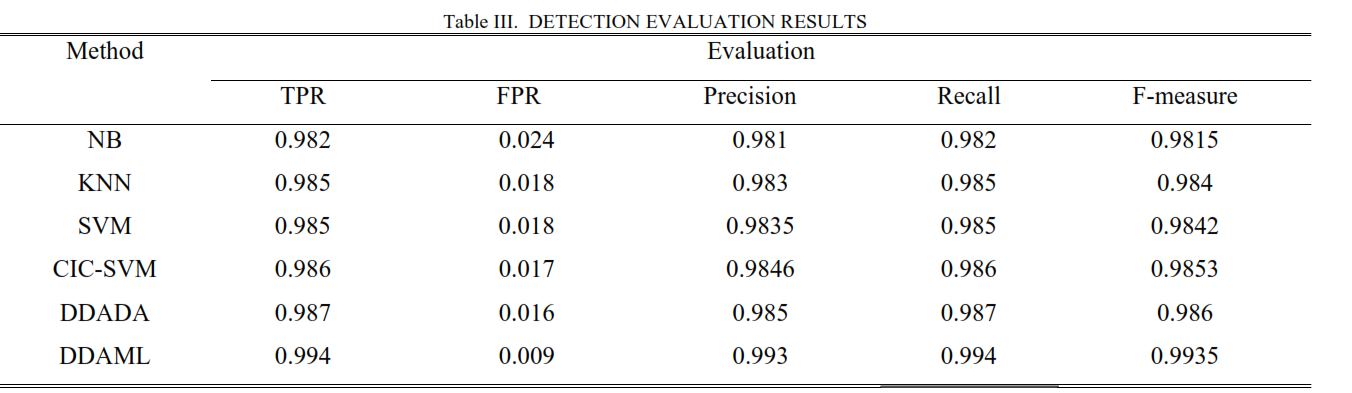
Bài báo sử dụng bốn feature như ảnh 34



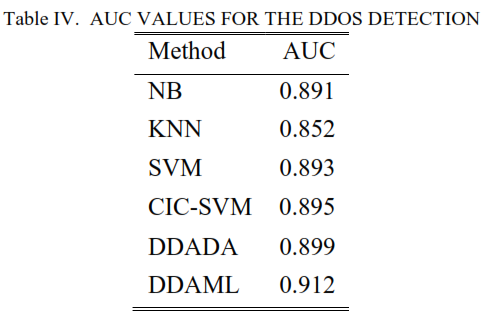
Hình 12:



Hình 13: Tập dữ liệu của bài báo

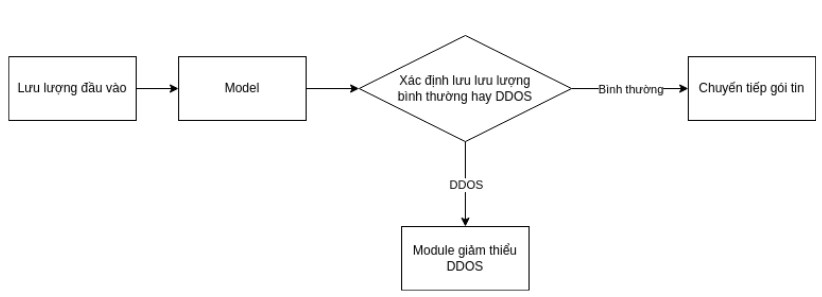


Hình 14: Kết quả đánh giá trong bài báo



Hình 15: Chỉ số AUC

## Mô tả cách hoạt động của quá trình thực nghiệm



Hình 16: Sơ đồ tổng quan cách hoạt động của quá trình thực nghiệm

Sau khi xây dựng Dataset và lựa chọn Model, quá trình xử lý lưu lượng đầu

vào của hệ thống được mô tả như hình 12. Quá trình xử lý diễn ra như sau:

* Lưu lượng đầu vào: Lưu lượng truy cập hệ thống từ các nguồn khác nhau được nhận vào khối "Lưu lượng đầu vào". Thông tin về luồng dữ liệu (flow traffic) từ flow table của từng switch được thu thập liên tục. Các thông tin này sẽ được trích xuất đặc trưng để đưa vào mô hình phân loại dự đoán xác định loại lưu lượng là lưu lượng bình thường hay là lưu lượng DDoS.
* Mô hình phân tích: Lưu lượng đầu vào sau khi trích xuất đặc trưng được đưa vào bộ phân loại dựa trên thuật toán K-Nearest Neighbors (KNN). KNN sử dụng khoảng cách giữa các điểm dữ liệu để xác định loại của một điểm dữ liệu mới dựa trên các điểm dữ liệu gần nhất trong không gian đặc trưng. Việc sử dụng KNN giúp xác định xem lưu lượng đầu vào là bình thường hay là lưu lượng tấn công DDoS dựa trên các đặc trưng đã được huấn luyện từ trước..
* Xác định lưu lượng bình thường hay DDoS: Bộ phân loại sẽ dự đoán lưu lượng đầu vào và xác định xem đó là lưu lượng bình thường hay lưu lượng tấn công DDoS.
* Lưu lượng bình thường: Nếu lưu lượng được xác định là bình thường, nó sẽ được chuyển tiếp đến "Chuyển tiếp gói tin" để xử lý tiếp theo trong hệ thống.
* Lưu lượng DDoS: Nếu lưu lượng được xác định là tấn công DDoS, nó sẽ được chuyển đến "Module giảm thiểu DDoS". Module này có chức năng block\_port là thêm một luồng (flow) mới vào bảng luồng của switch để chặn tất cả lưu lượng qua một cổng nhất định trong một khoảng thời gian được thiết lập sẵn. Hàm này được sử dụng để ngăn chặn tấn công từ chối dịch vụ (DoS) bằng cách chặn cổng đang nhận lưu lượng tấn công.

## Kịch bản thực nghiệm

### Chuẩn bị

Mở Ubuntu (nơi chứa [ryu-controller](https://ryu.readthedocs.io/en/latest/getting_started.html))

Mở Mininet VM [mininet](https://github.com/mininet/mininet/releases/)

Connect ssh 2 máy ảo trên trên mobaXterm

Trên máy Ubuntu ryu

$ cd ryu/ryu/controller

$ git clone https://github.com/tranthanhhuy19032001/DACNTT2\_DDos\_ML.git

username: tranthanhhuy19032001

password: ghp\_dJ13ANgAhoPo1DMLIQECFZGixaRluE1VT6Vc

$ cd DACNTT2\_DDos\_ML/FinalVersion/controller

$ gdown https://drive.google.com/u/0/uc?id=18FvTV4Oe\_2bYOmXdSi16d5pXhAhGaypx

Trên Mininet VM

$ cd mininet/mininet

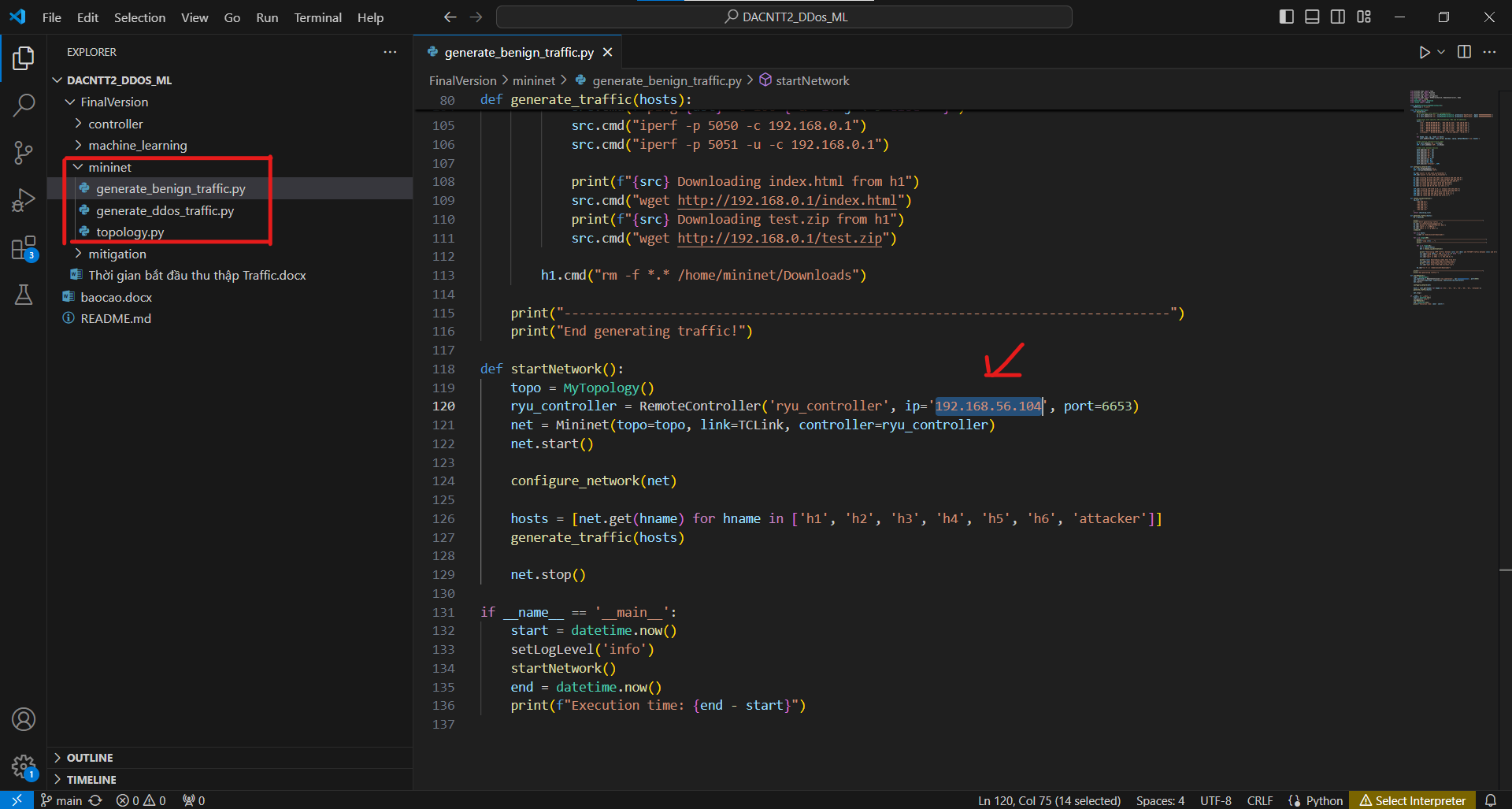
$ git clone https://github.com/tranthanhhuy19032001/DACNTT2\_DDos\_ML.git

username: tranthanhhuy19032001

password: ghp\_dJ13ANgAhoPo1DMLIQECFZGixaRluE1VT6Vc

$ cd DACNTT2\_DDos\_ML/FinalVersion/mininet

Đổi địa chỉ ip trong 3 file generate\_benign\_traffic.py, generate\_benign\_traffic.py, topology.py thành ip máy ryu của bạn.



Hình 17: Đổi địa chỉ IP thành địa chỉ của máy Ryu

Lệnh cấu hình để mở xterm trên mobaxterm

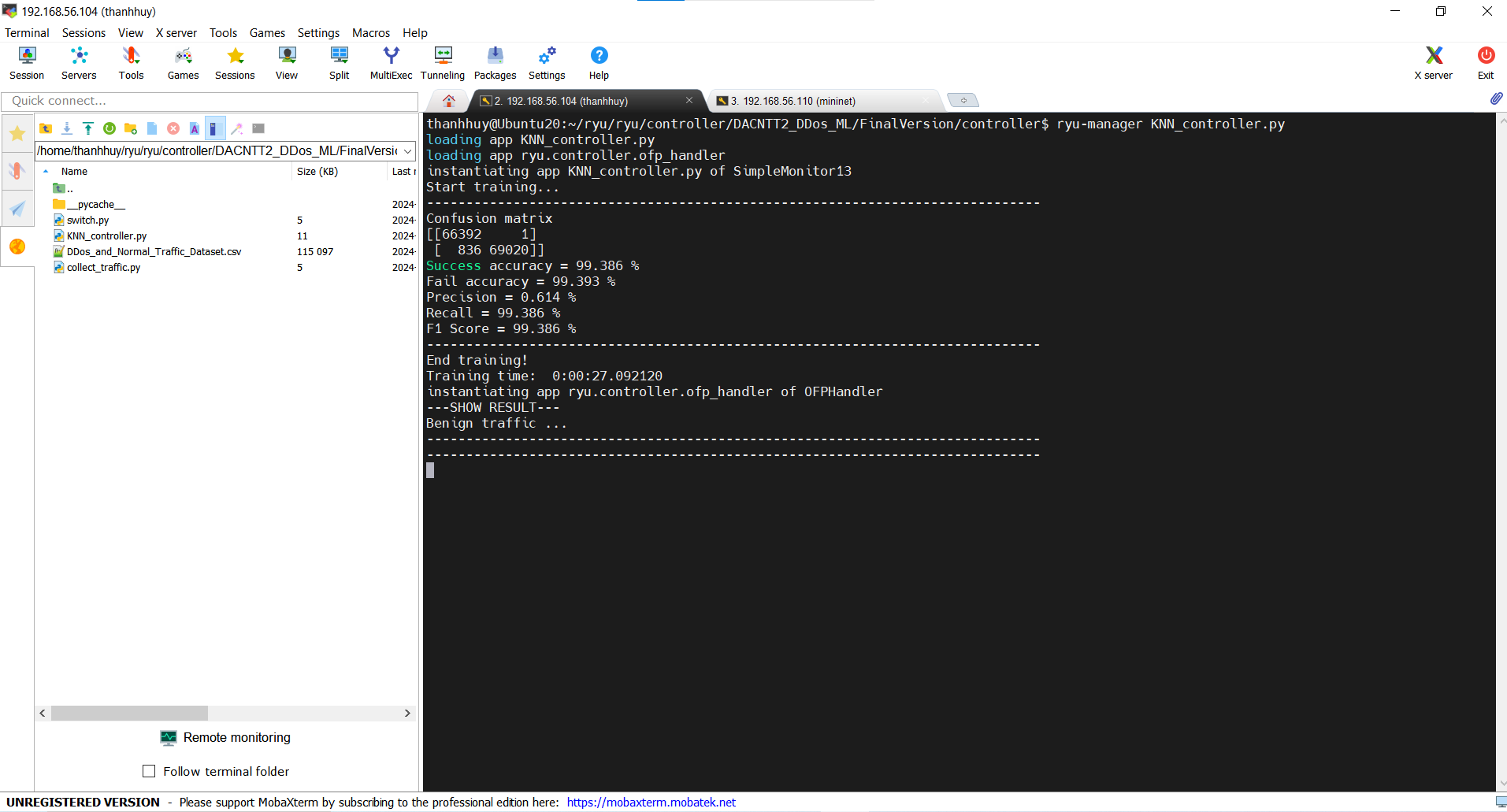
$ export DISPLAY=localhost:10.0

$ sudo xauth add $(xauth -f ~mininet/.Xauthority list|tail -1)

### Thực nghiệm phát hiện DDoS

#### Trên máy Ryu Controller

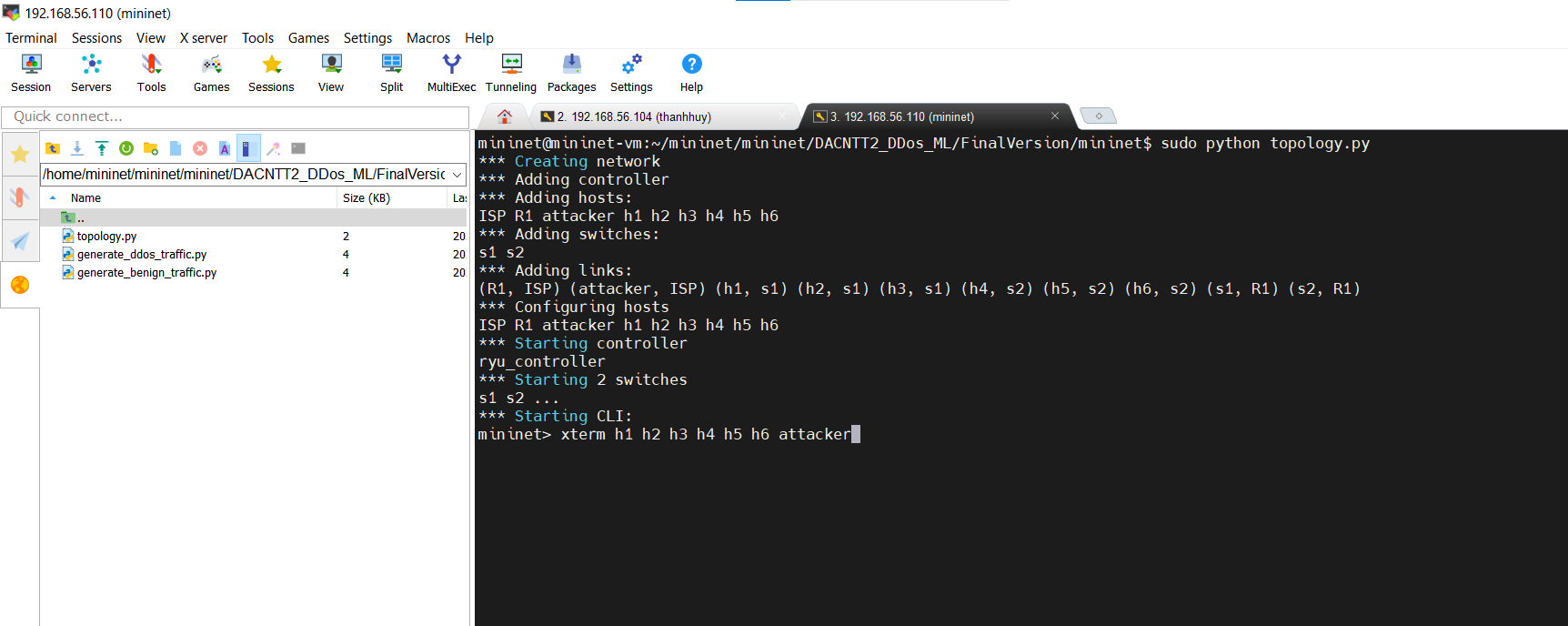
$ ryu-manager KNN\_controller.py



Hình 18: Khởi chạy KNN\_controller

#### Trên Mininet VM

$ sudo python topology.py



Hình 19: Khởi tạo mô hình mạng

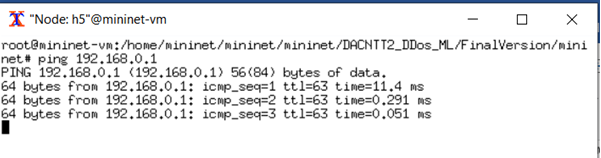
Mở các node mạng bằng xterm

$ xterm h1 h2 h3 h4 h5 attacker

Thực hiện các lệnh ping bình thường để kiểm tra traffic bình thường.

Tại node h5 ping đến h1

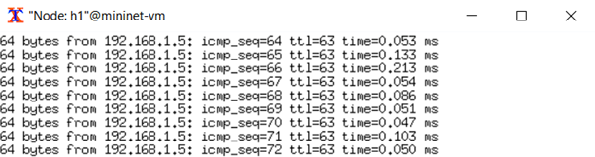
$ ping 192.168.0.1



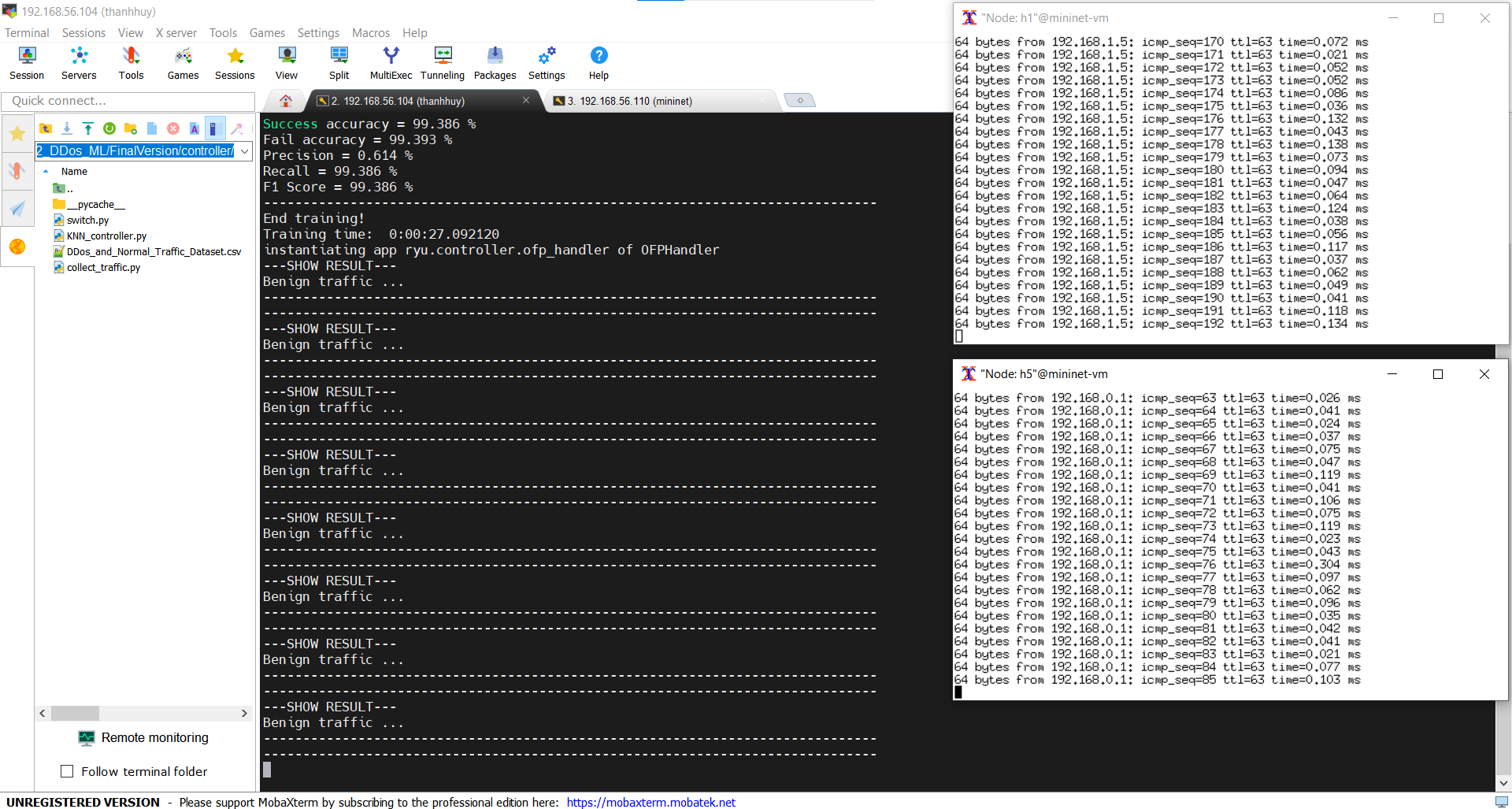
Hình 20: Ping đến node h1

Tại node h1 ping đến h5

$ ping 192.168.1.5



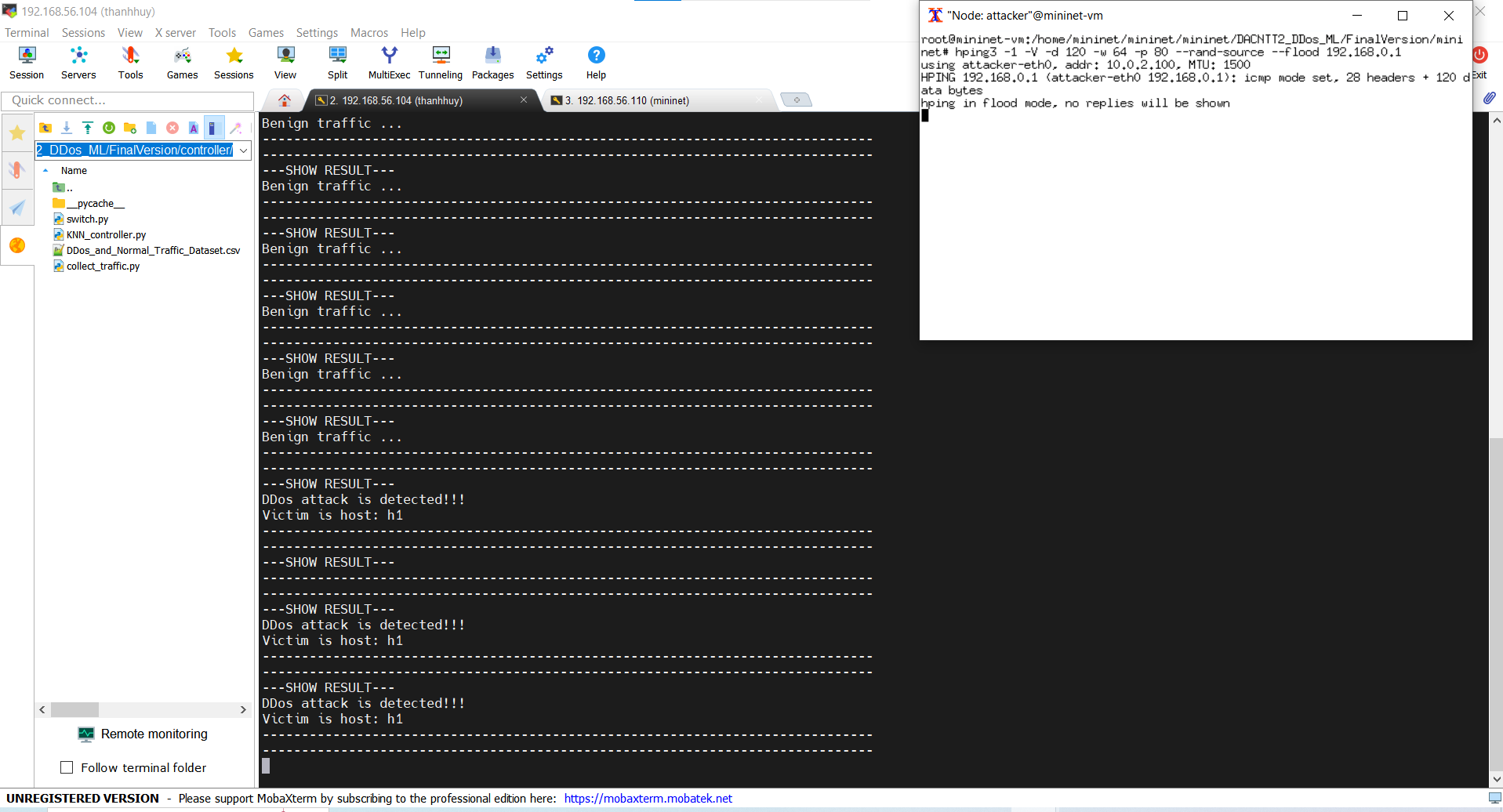
Hình 21: Ping đến node h5



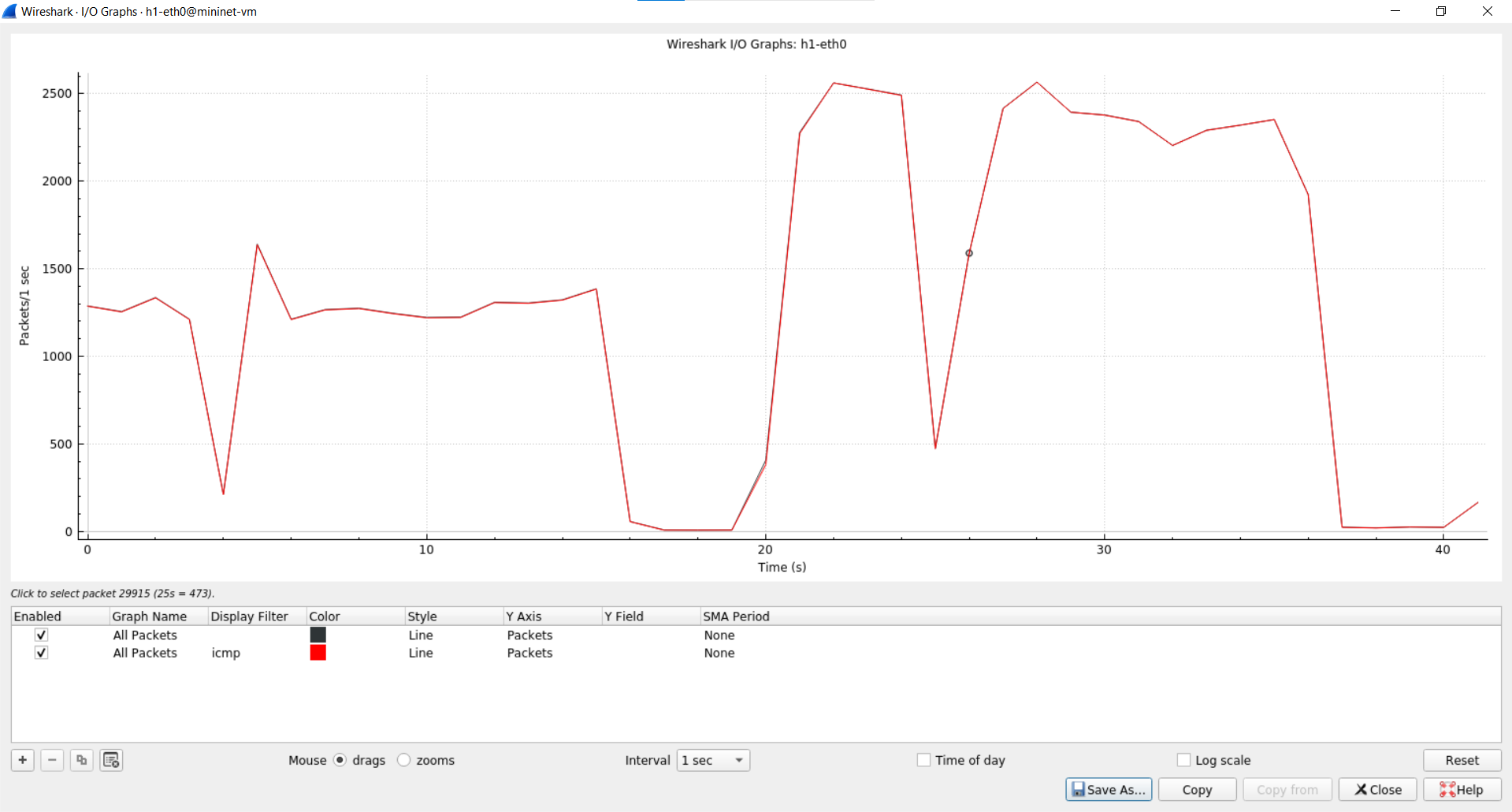
Hình 22: Kết quả dự đoán khi traffic bình thường

Thực hiện tấn cống ICMP Flood tới h1:

$ hping3 -1 -V -d 120 -w 64 -p 80 --rand-source –flood 192.168.0.1



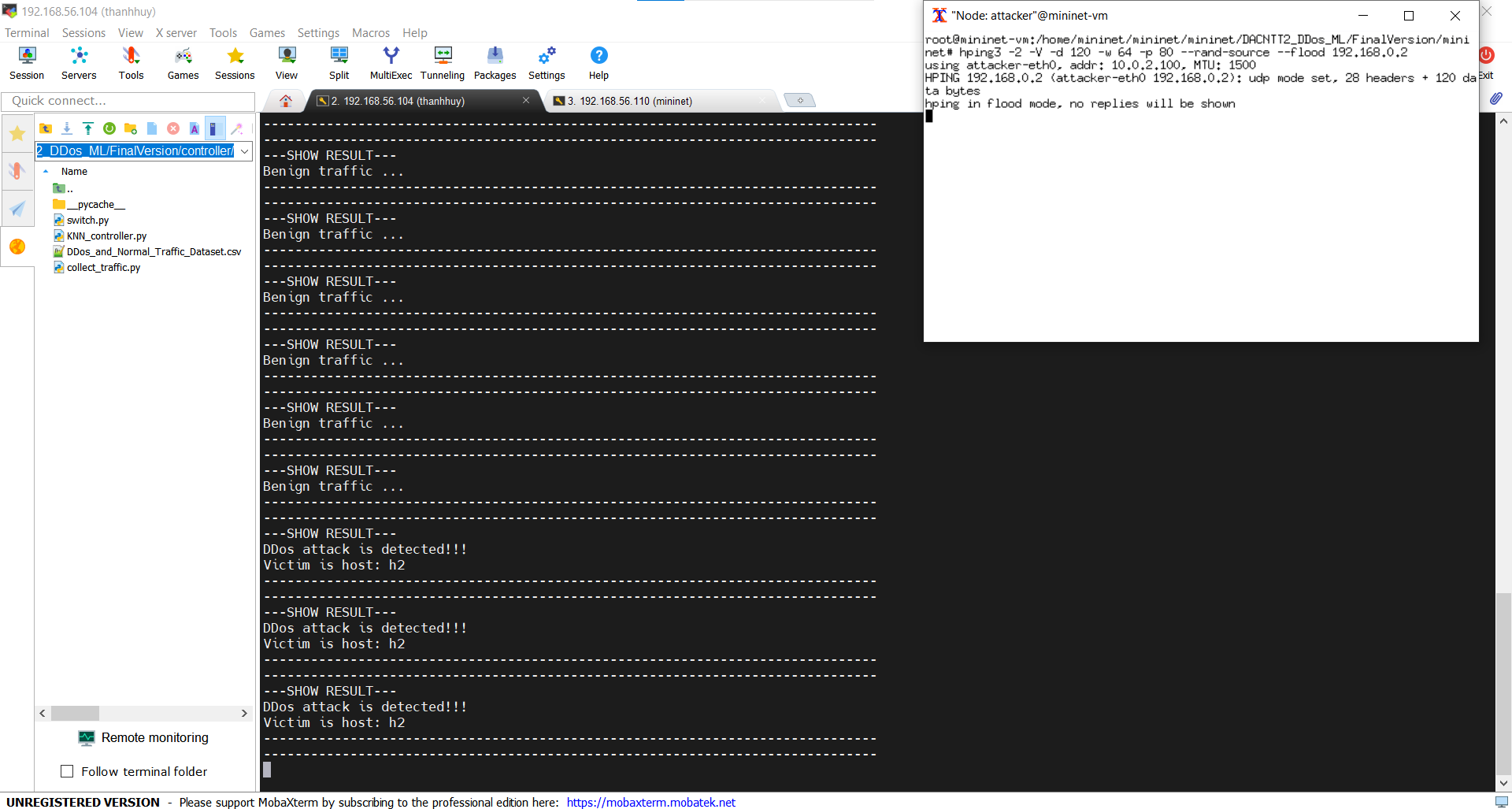
Hình 23: Phát hiện host h1 bị tấn công



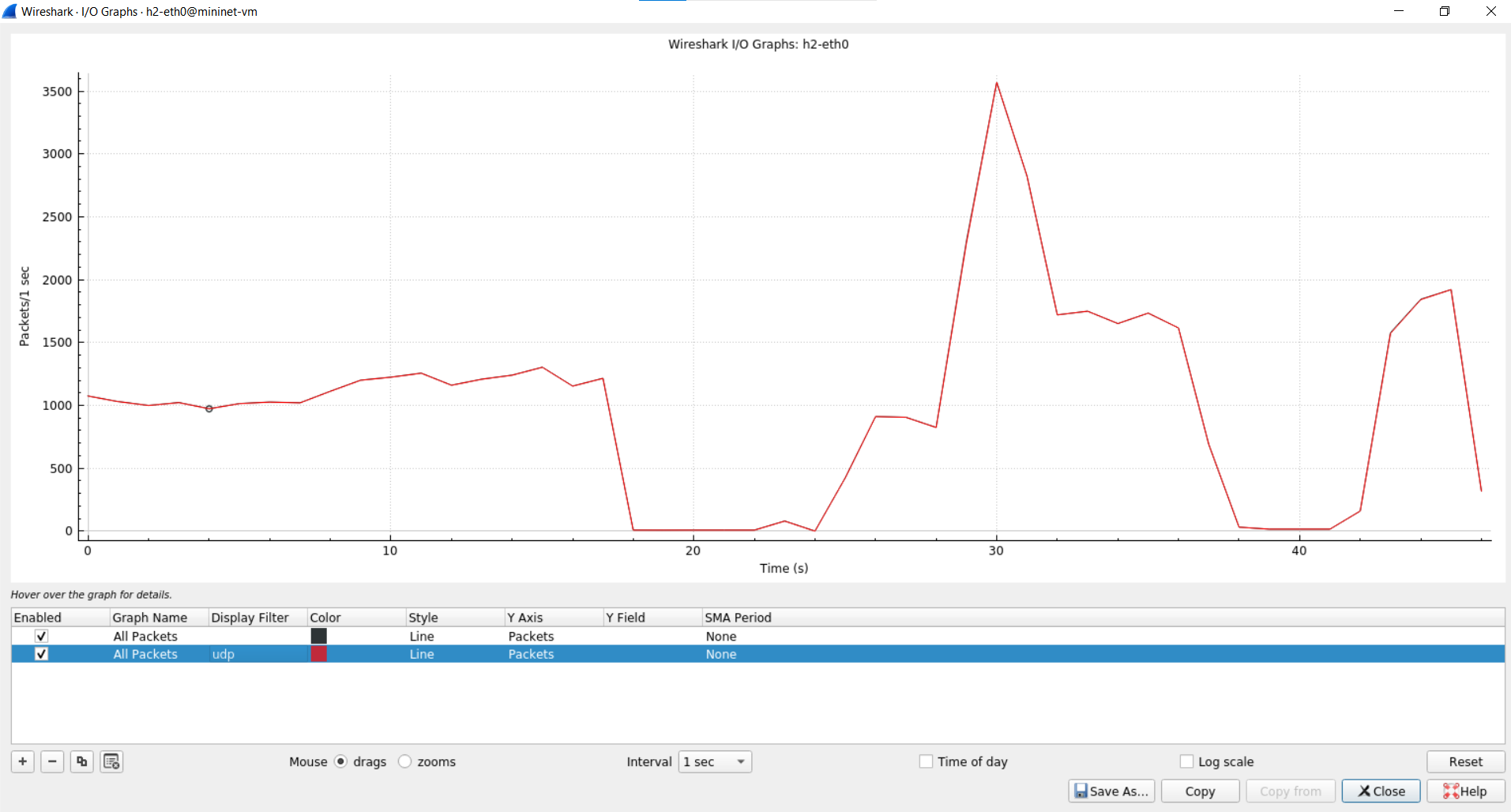
Hình 24: Biểu đồ traffic trên host h1

Thực hiện tấn công UDP Flood tới h2:

$ hping3 -2 -V -d 120 -w 64 -p 80 --rand-source –flood 192.168.0.2



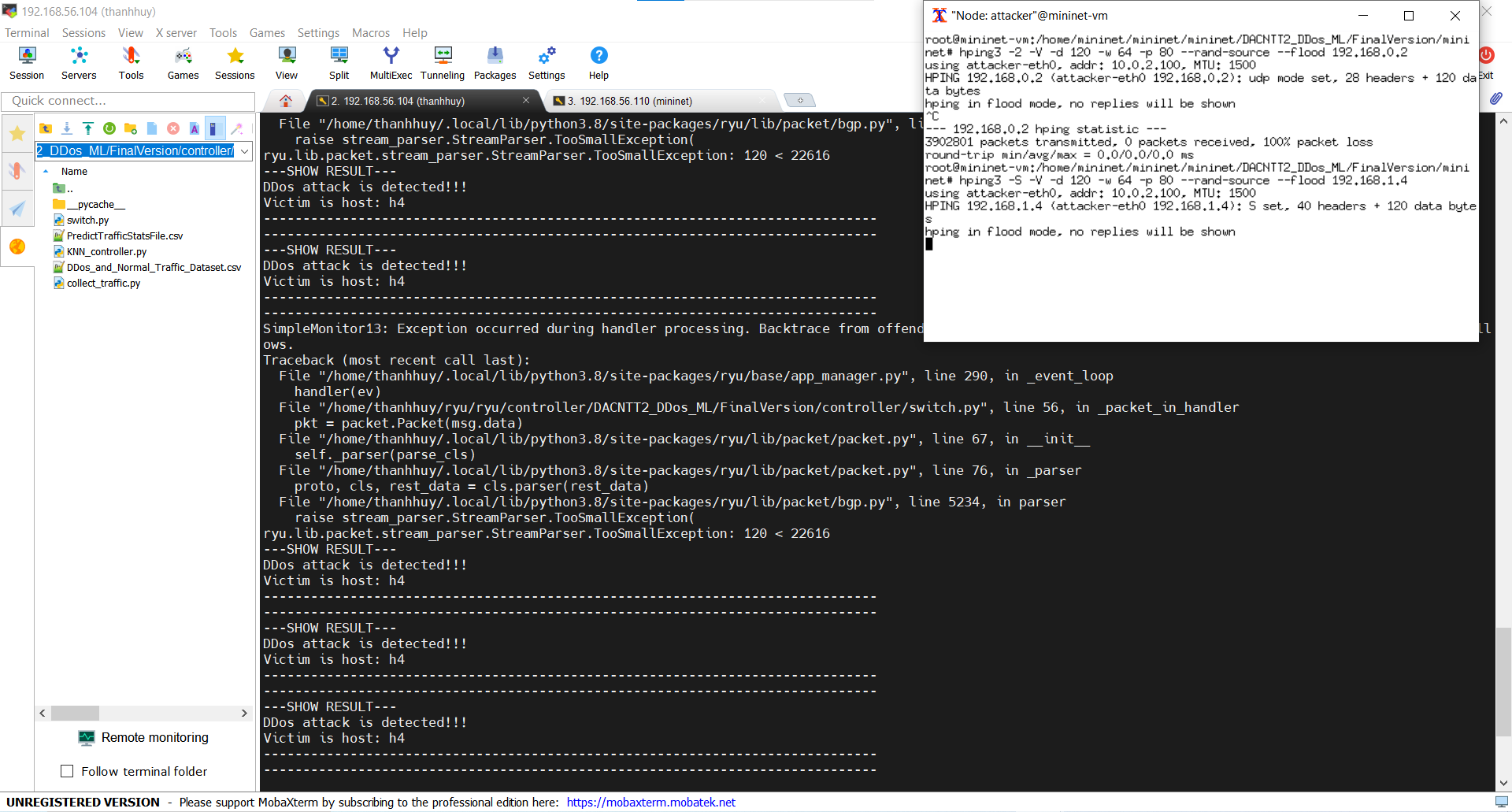
Hình 25: Phát hiện host h2 bị tấn công



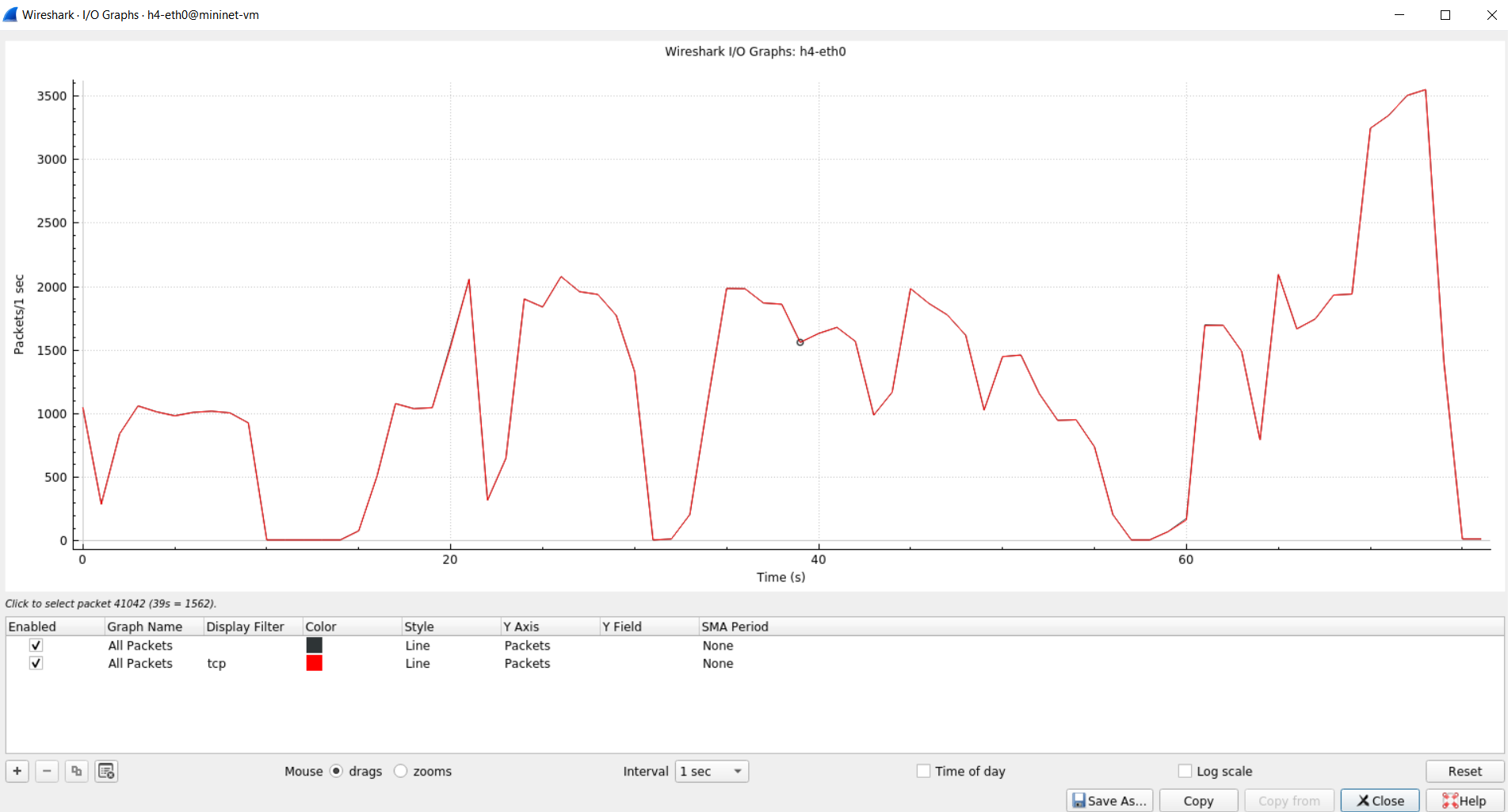
Hình 26: Biểu đồ traffic tại host h2

Thực hiện tấn công TCP-SYN Flood tới h4

$ hping3 -S -V -d 120 -w 64 -p 80 --rand-source –flood 192.168.1.4



Hình 27: Phát hiện host h4 bị tấn công



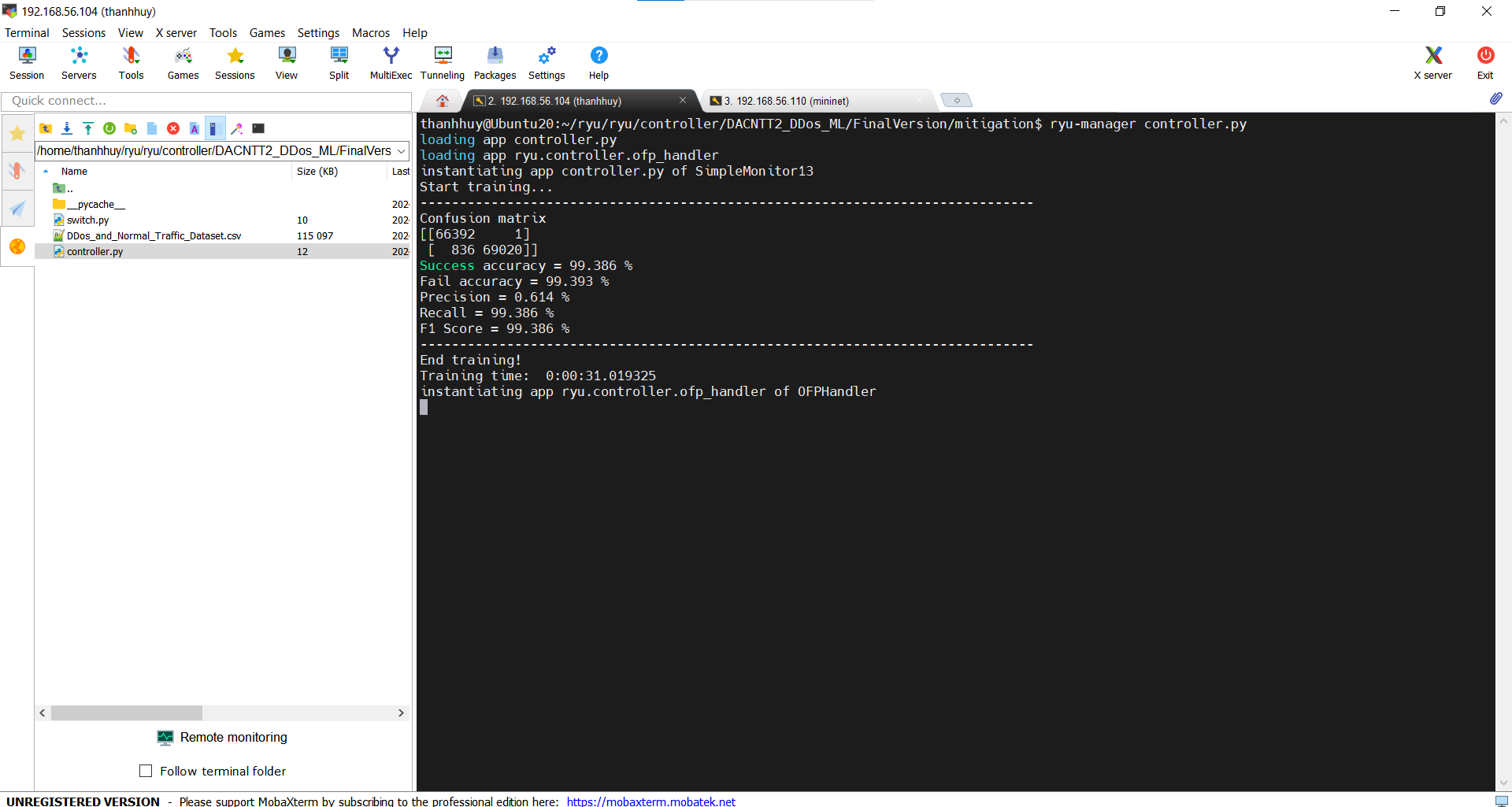
Hình 28: Biểu đồ traffic tại host h4

### Migration

Trên RYU

$ cd DACNTT2\_DDos\_ML/FinalVersion/migration

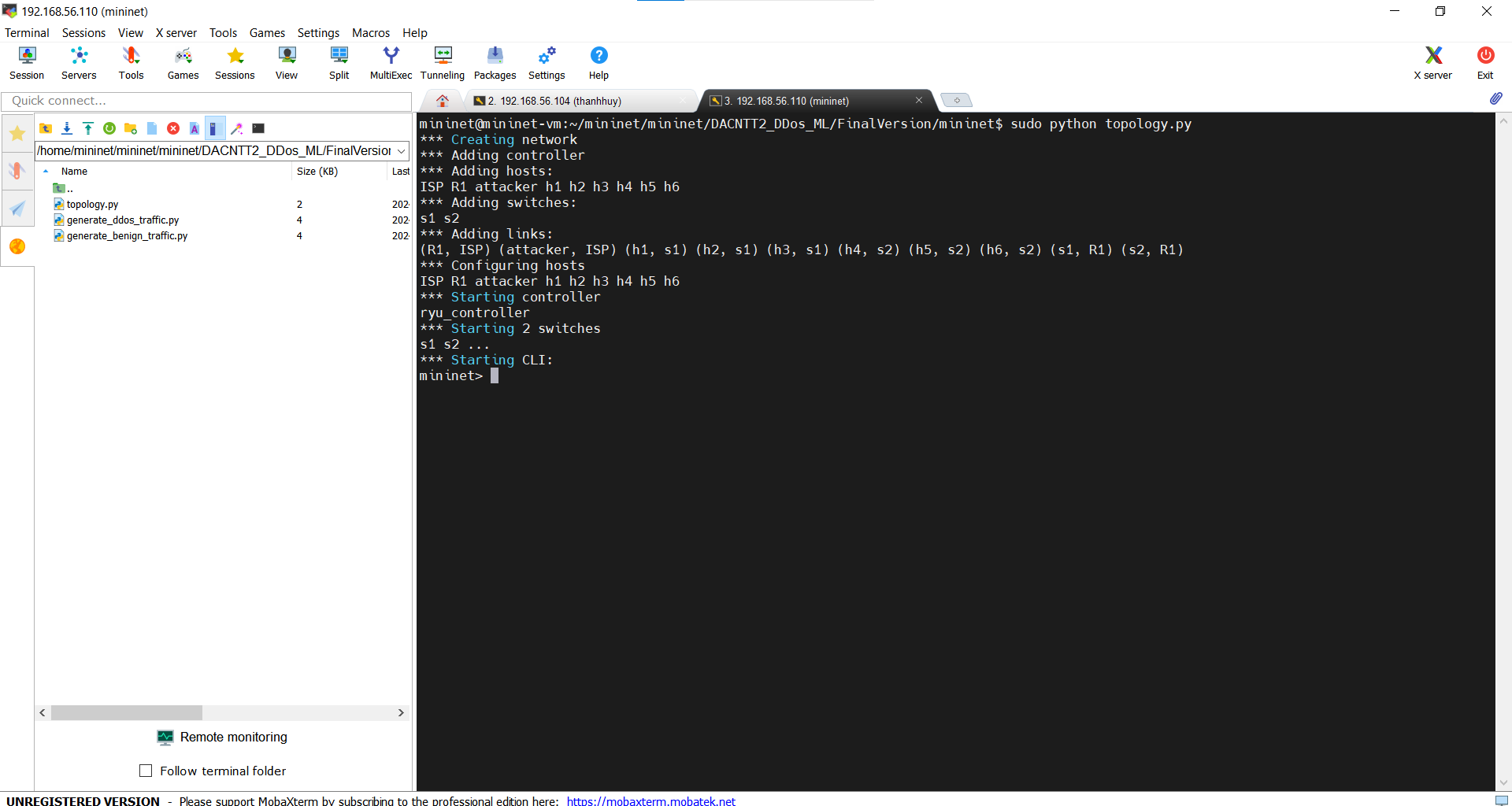
$ ryu-manager controller.py



Hình 29: Khởi chạy migration controller

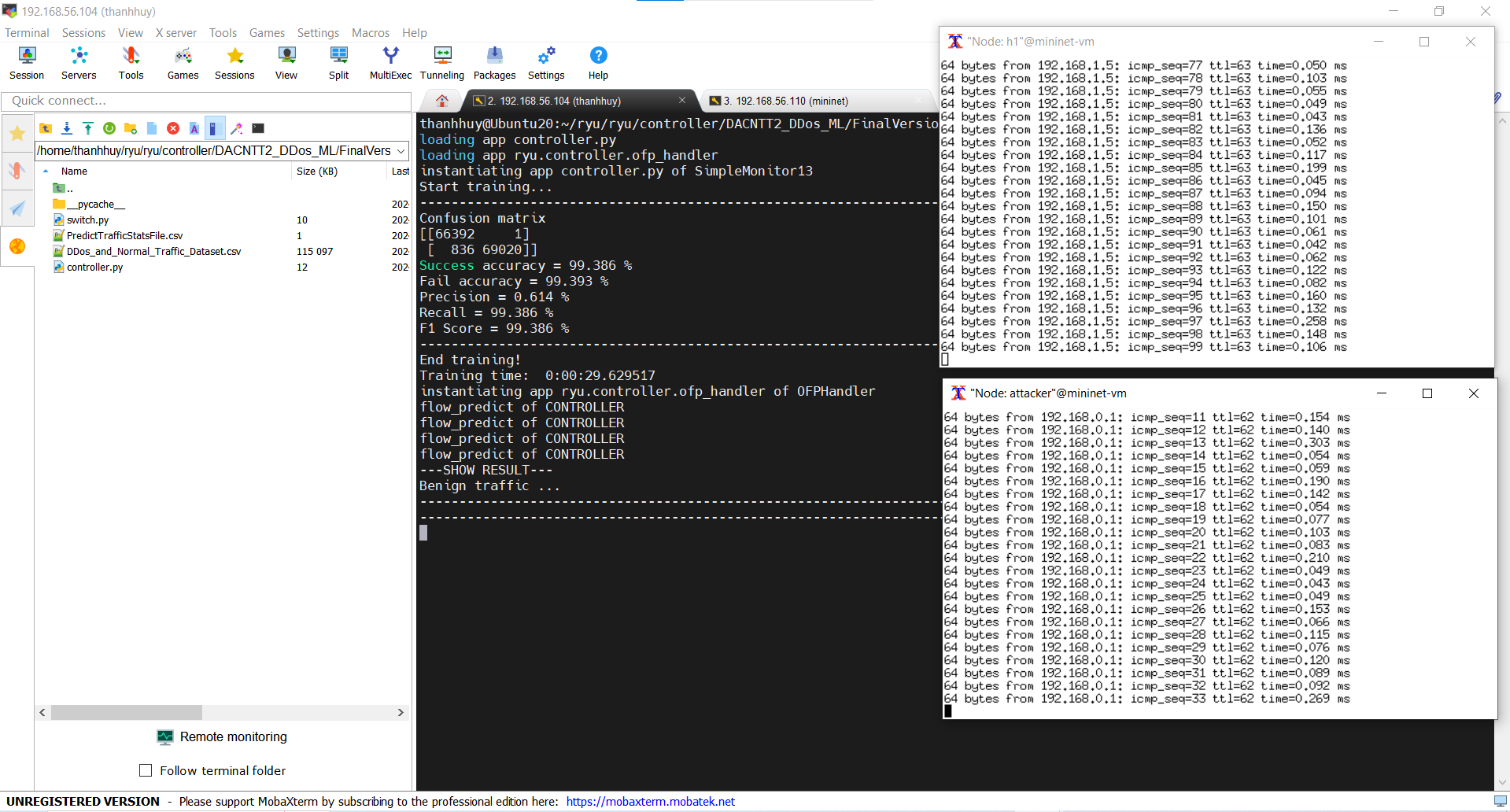
Trên Mininet VM

$ sudo python topology

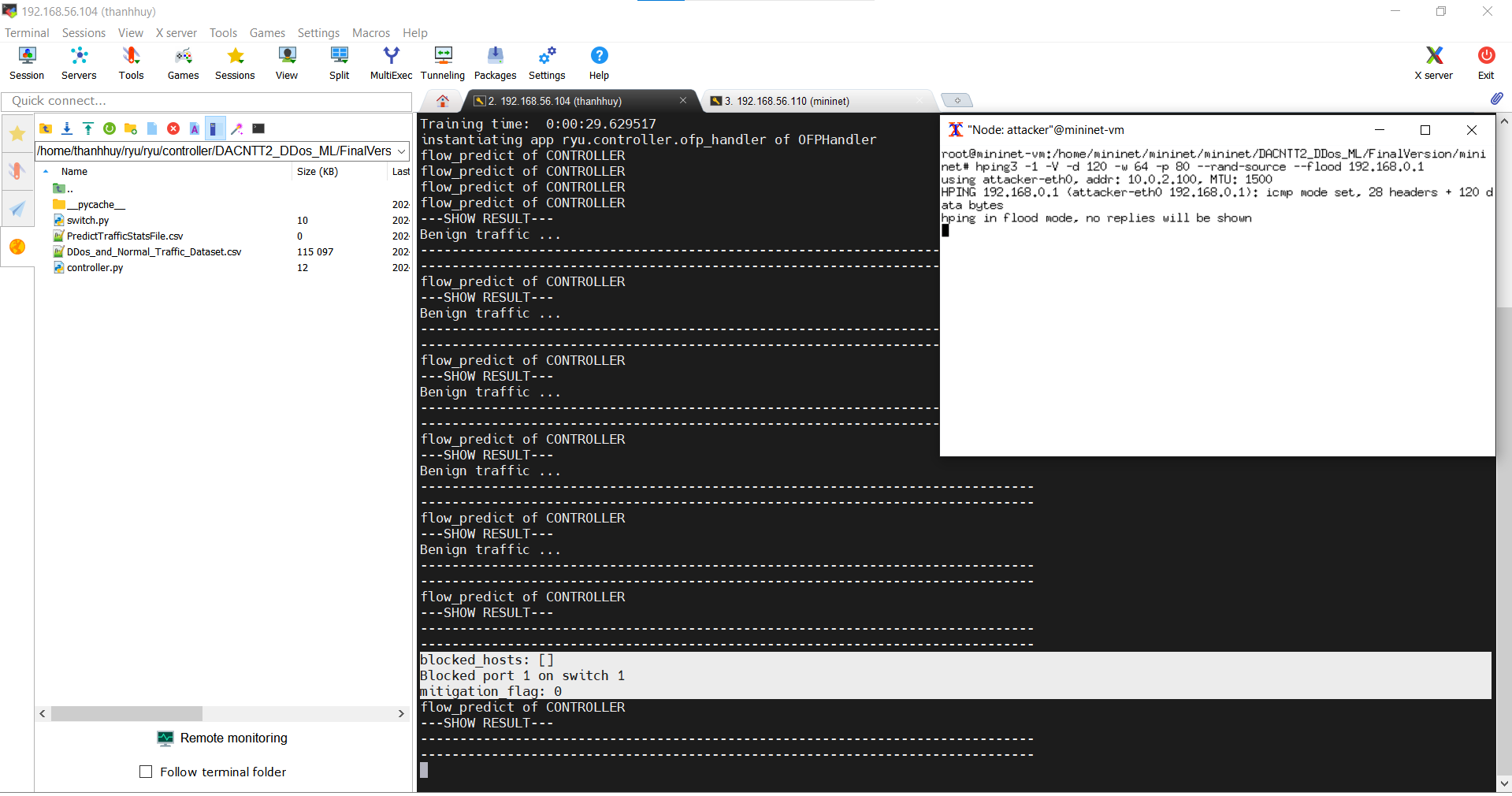


Hình 30: Khởi tạo mô hình mạng

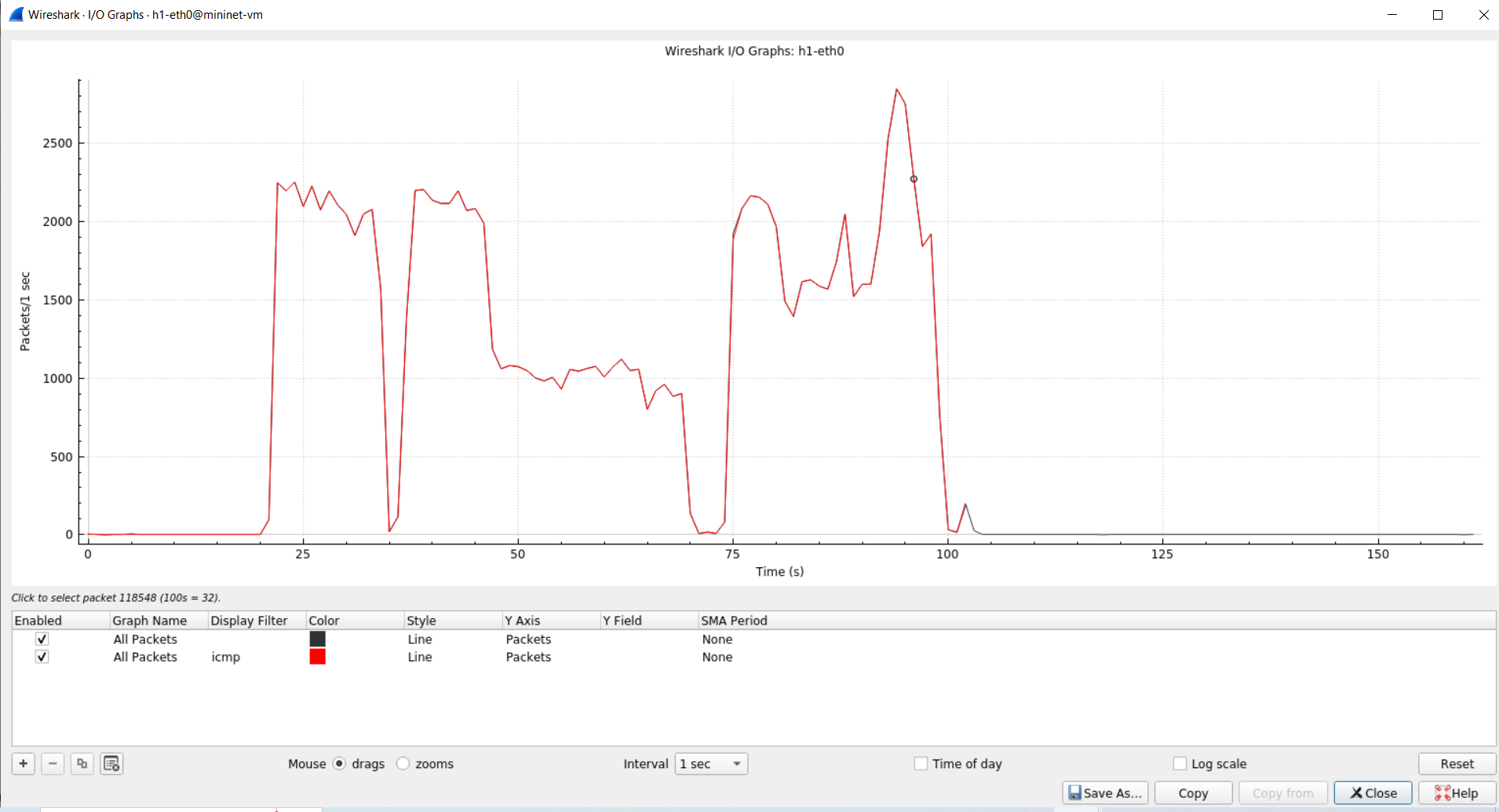
Chạy lại các lệnh tấn công ở phần 3.5.2.2 và kiểm tra sau khi phát hiện bị tấn công DDos thì port của host đó bị block.



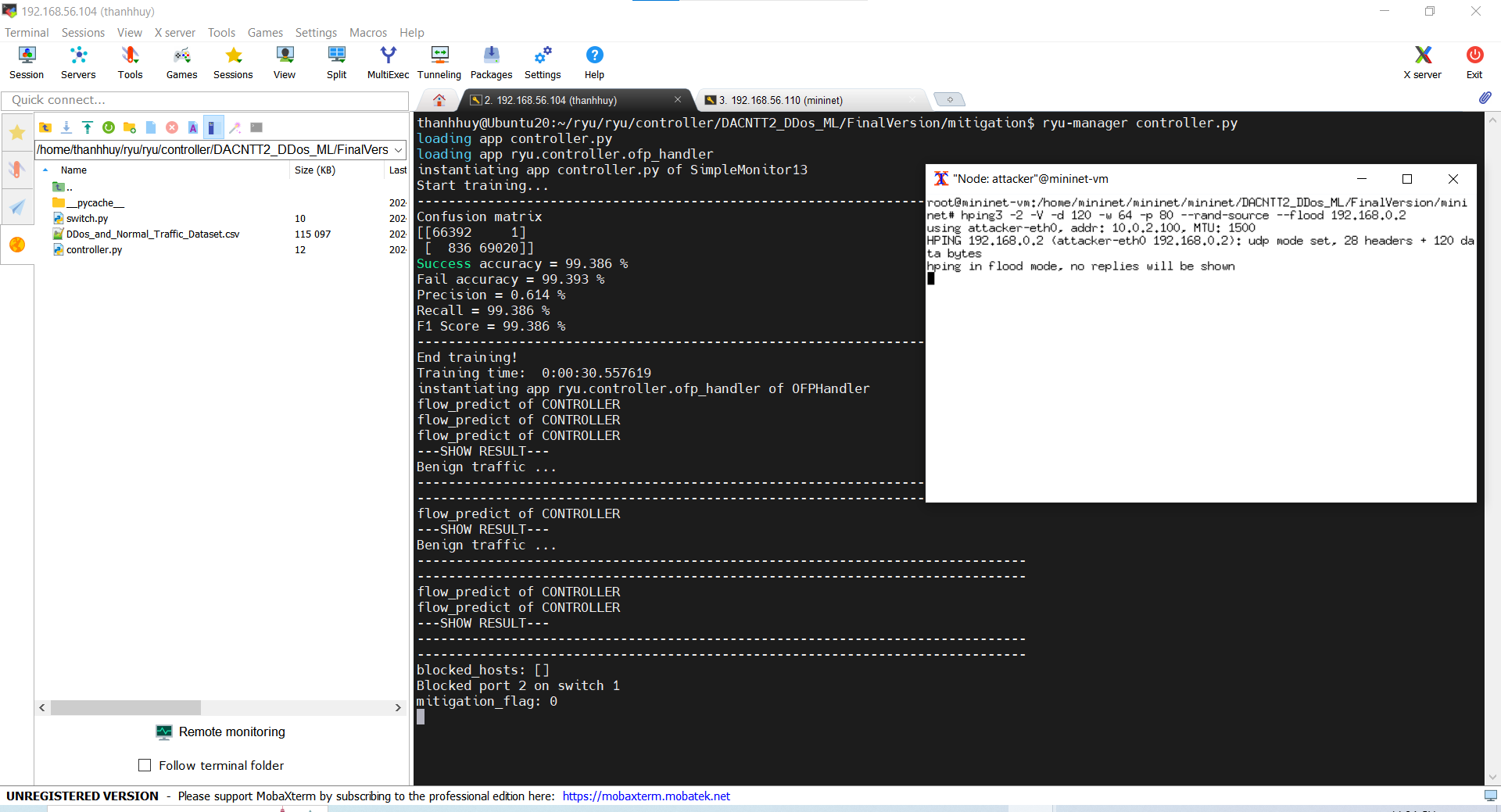
Hình 31: Test Ping bình thường



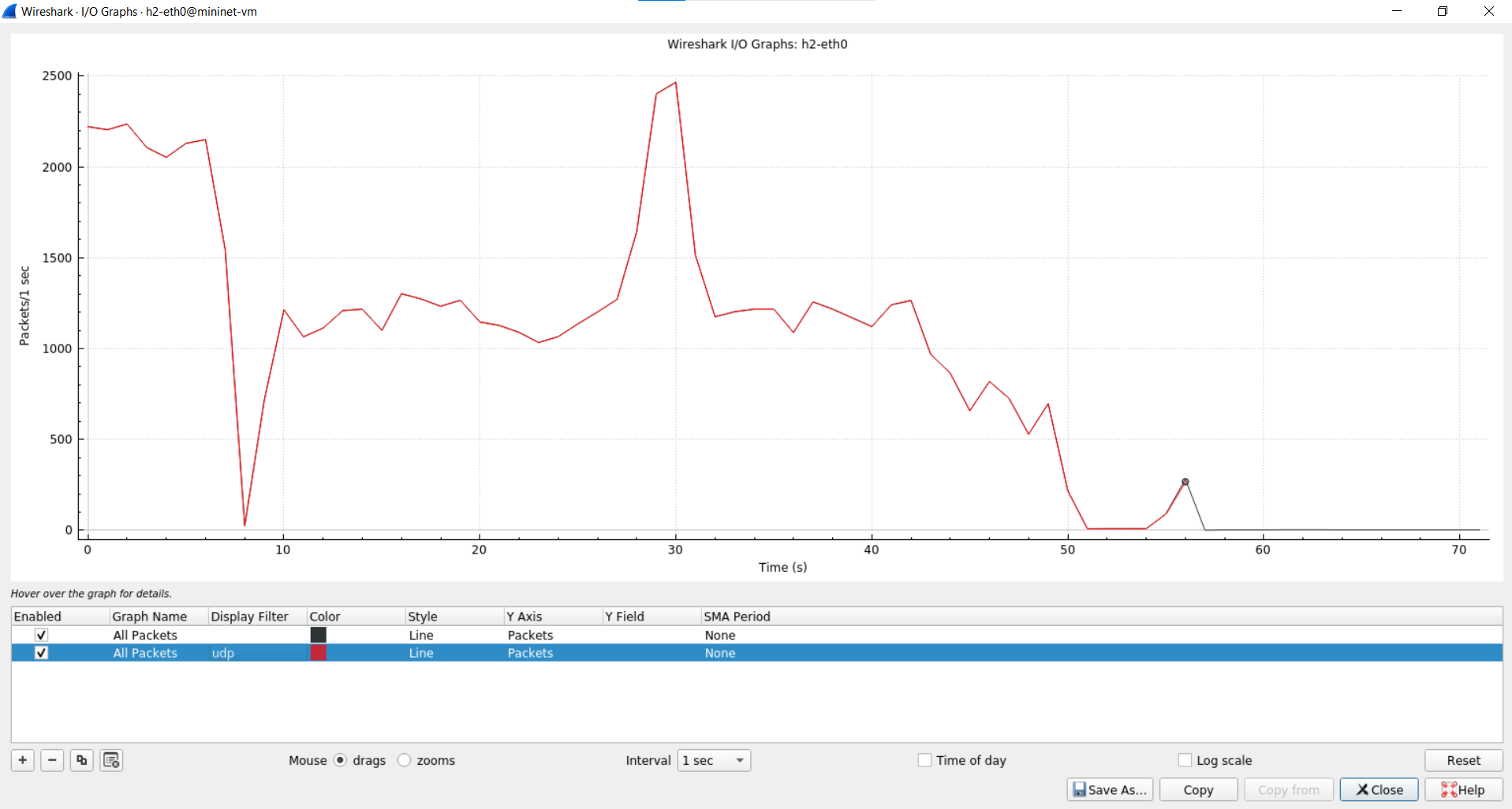
Hình 32: Thực hiện tấn công ICMP Flood tới host h1



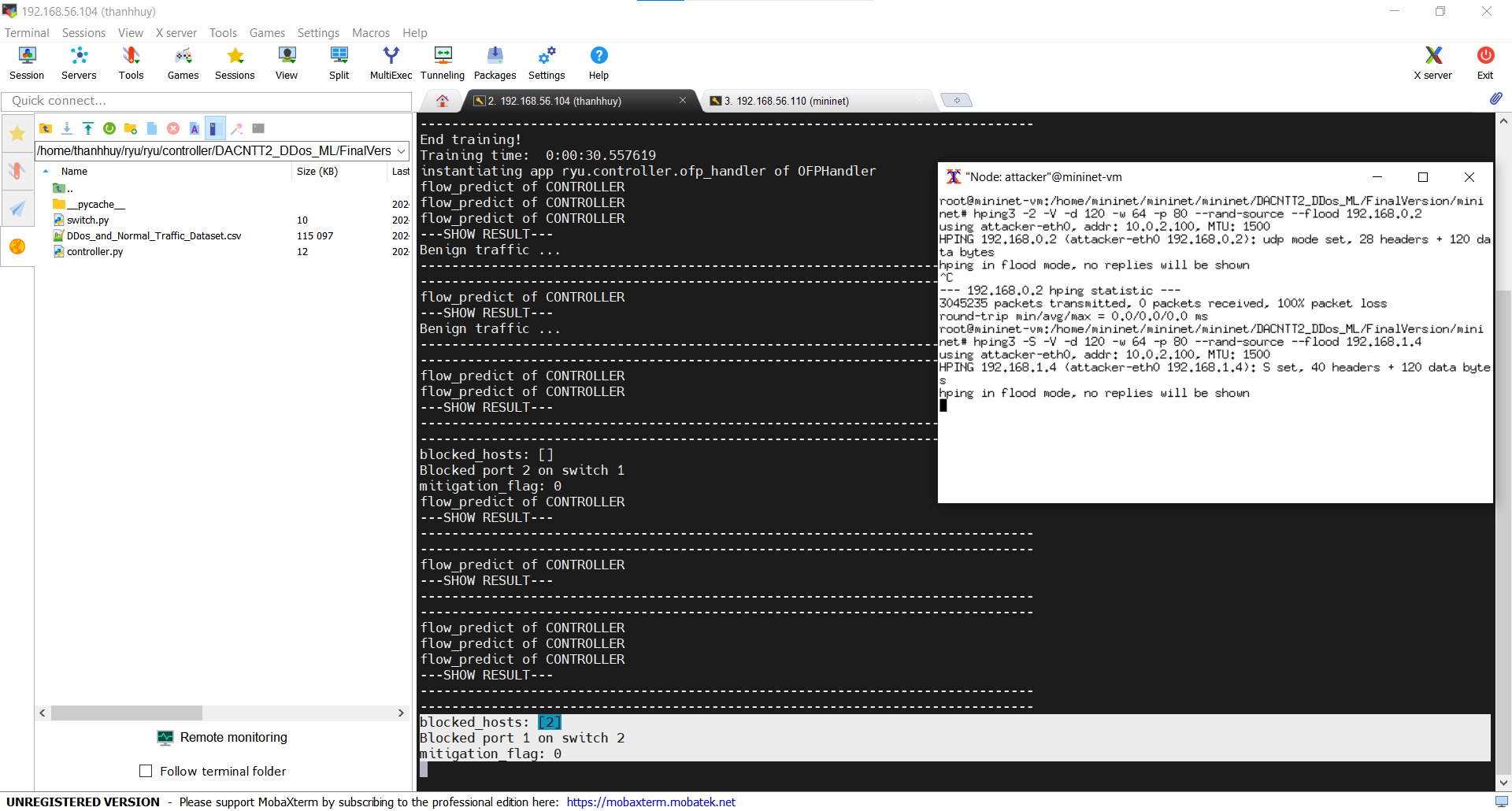
Hình 33: Biểu đồ host h1 bị tấn công ICMP Flood, sau đó hệ thống phát hiện và block port của h1 trên switch 1



Hình 34: Thực hiện tấn công UDP Flood tới host h2



Hình 35:



Hình 36: Thực hiện tấn công TCP-SYN Flood tới h4



Hình 37:

# KẾT LUẬN

## Kết luận

Đánh giá kết quả thu được:

* Triển khai thành công mạng SDN trên nền tảng mininet, sử dụng OpenFlow v1.3, với controller là RYU.
* Giả lập thành công trương hợp tấn công DDoS, với các loại tấn công phổ biến là SYN, IMCP, UDP.
* Với mô hình KNN, cho kết quả phân loại lưu lượng bình thường và tấn công có độ chính xác cao, việc mô hình hoạt động chính xác đã giúp cho việc phát hiện phát hiện DDoS đạt kết quả cao, module giảm thiểu dựa trên block port đã hoạt động tốt nhờ kết quả phát hiện chính xác.

Tóm lại, báo cáo này đã triển khai thành công một hệ thống mạng SDN mô phỏng trên mininet, với controller là RYU, sử dụng giao thức OpenFlow v1.3 với sự hỗ trợ của OpenvSwitch. Thuật toán KNN mang lại hiệu quả cao trong việc phát hiện tấn công DDoS từ đó có biện pháp giảm thiểu nhằm ngăn chặn cuộc tấn công, giảm thiểu thiệt hại cho hệ thống.

## Hướng phát triển

# TÀI LIỆU THAM KHẢO

Tiếng Việt

Tiếng Anh

[1] Shi Dong, Mudar Sarem. “DDoS\_Attack\_Detection\_Method\_Based\_on\_Improved\_KNN”. 2019