



BÁO CÁO CUỐI KỲ

# Ứng dụng SDN xây dựng Firewall và Application quản lý Network

*Nhóm 9*

Ngày 16 tháng 5 năm 2024



---

**Sinh viên thực hiện**

106200284 - Hồ Đức Vũ - 20KTMT2

106200241 - Nguyễn Minh Phương - 20KTMT1

106200240 - Huỳnh Vũ Đình Phước - 20KTMT1

---

**Giáo viên hướng dẫn**TS. Tăng Anh Tuấn

---

**Môn học**Software Define Network

---

**Đề tài**Ứng dụng SDN xây dựng Firewall và Application quản lý Network

---

**Xuất bản**

Đà Nẵng, Ngày 16 tháng 5 năm 2024

**Số trang**16

---

## Mục lục

<b>Nội dung báo cáo</b>	<b>2</b>
<b>1 Giới thiệu</b>	<b>2</b>
<b>2 Nội dung nghiên cứu</b>	<b>2</b>
2.1 Cấu trúc Network . . . . .	2
2.2 Firewall . . . . .	4
2.3 Application . . . . .	8
<b>3 Thủ nghiệm</b>	<b>9</b>
3.1 Ping theo database . . . . .	9
3.1.1 ICMP . . . . .	9
3.1.2 TCP . . . . .	10
3.1.3 UDP . . . . .	11
3.2 Thêm rule bằng ứng dụng web . . . . .	12
3.3 Ping liên tục . . . . .	13
3.4 Giám sát với Wireshark . . . . .	14
<b>4 Kết luận và đánh giá</b>	<b>15</b>
<b>Tài liệu tham khảo</b>	<b>16</b>
<b>Phụ lục</b>	<b>16</b>

# Nội dung báo cáo

## 1 Giới thiệu

Trong thời đại số hóa hiện nay, bảo mật mạng đã trở thành một yếu tố quan trọng không thể thiếu. Firewall là một trong những công cụ bảo mật cơ bản và quan trọng nhất, được sử dụng rộng rãi để kiểm soát và bảo vệ các hệ thống mạng khỏi các mối đe dọa và truy cập trái phép. Tuy nhiên, các phương pháp firewall truyền thống gặp phải nhiều hạn chế khi đối mặt với sự phát triển nhanh chóng của công nghệ và sự phức tạp ngày càng tăng của các mạng máy tính hiện đại.

Mạng được xác định bằng phần mềm (Software-Defined Networking - SDN) đã nổi lên như một giải pháp tiên tiến, mang lại khả năng quản lý mạng linh hoạt và hiệu quả hơn. SDN tách biệt chức năng điều khiển mạng khỏi phần cứng trong các bộ điều hướng như switch hay router và cho phép quản lý mạng thông qua phần mềm, từ đó giảm thiểu chi phí và độ phức tạp. Trong đó OpenFlow là một trong những giao thức chính của SDN, cho phép các bộ điều khiển (controller) tương tác trực tiếp với các switch và router trong mạng. Ryu là một controller mã nguồn mở mạnh mẽ cho SDN, hỗ trợ đầy đủ OpenFlow và cung cấp một nền tảng phát triển linh hoạt cho các ứng dụng mạng được sử dụng để tương tác và điều khiển hệ thống mạng lưới một cách hiệu quả.

Đối với Firewall truyền thống có nhiều nhược điểm như khả năng mở rộng kém và gặp khó khăn khi đối mặt với lưu lượng mạng lớn, việc cấu hình và quản lý các quy tắc firewall phức tạp và tốn thời gian, các firewall truyền thống không thể dễ dàng thích ứng và thay đổi nhanh chóng trong môi trường mạng.

Mục tiêu của chúng tôi trong báo cáo này là xây dựng một hệ thống firewall với việc ứng dụng SDN nhằm khắc phục những nhược điểm của firewall truyền thống, hệ thống linh hoạt, quản lý đơn giản và hiệu quả.

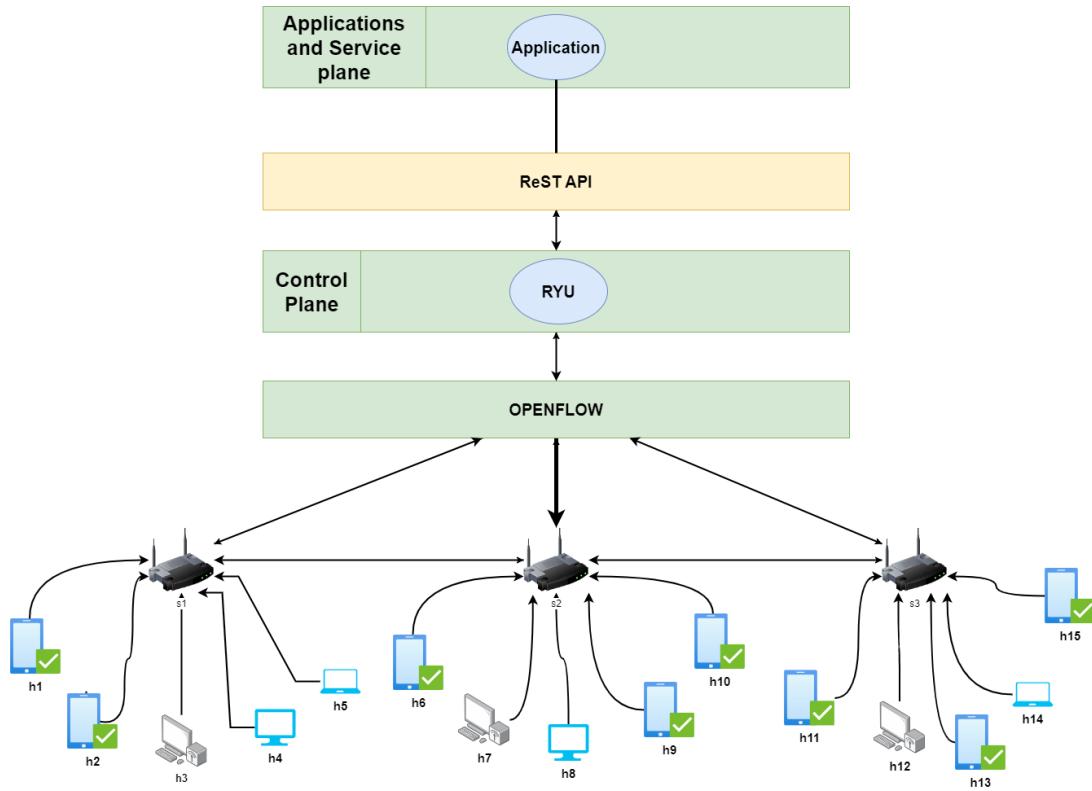
## 2 Nội dung nghiên cứu

Để ứng dụng SDN trong việc xây dựng Firewall và Application điều khiển mạng lưới, chúng tôi sử dụng các công cụ bao gồm Ryu để xây dựng controller cho hệ thống, OpenFlow được dùng để controller tương tác với nền tảng network và ReST API được sử dụng để Application tương tác và điều khiển Firewall. Chúng tôi sử dụng mininet, một công cụ được dùng để mô phỏng mạng lưới với mục đích triển khai và kiểm thử Firewall trong báo cáo này. Tại hình 2.1 mô tả cấu trúc hệ thống được triển khai trong dự án của chúng tôi, trong các phần tiếp theo, chúng ta sẽ phân tích cụ thể hơn về cấu trúc này.

### 2.1 Cấu trúc Network

Cấu trúc mạng lưới được triển khai trong dự án này là mạng lưới có cấu trúc mạng hình cây, gồm có 3 switch là s1, s2 và s3 với mỗi switch kết nối 5 host được gán nhãn từ h1 đến h15. Các switch được điều khiển bởi 1 controller thông qua OpenFlow, controller sử dụng Ryu để triển

khai, đóng vai trò là Firewall trong hệ thống mạng. Chúng tôi tạo ra một ứng dụng được dùng để giám sát và điều khiển Firewall (controller) thông qua ReST API.



Hình 2.1: Cấu trúc Network.

Các host (thiết bị đầu cuối) được gán địa chỉ bắt đầu từ 10.0.0.1/24 đến 10.0.0.15/24 tương ứng từ h1 đến h15. Cấu trúc kết nối của network được biểu diễn tại hình 2.2.

```

mininet> net
h1 h1-eth0:s1-eth2
h2 h2-eth0:s1-eth3
h3 h3-eth0:s1-eth4
h4 h4-eth0:s1-eth5
h5 h5-eth0:s1-eth6
h6 h6-eth0:s2-eth3
h7 h7-eth0:s2-eth4
h8 h8-eth0:s2-eth5
h9 h9-eth0:s2-eth6
h10 h10-eth0:s2-eth7
h11 h11-eth0:s3-eth2
h12 h12-eth0:s3-eth3
h13 h13-eth0:s3-eth4
h14 h14-eth0:s3-eth5
h15 h15-eth0:s3-eth6
s1 lo: s1-eth1:s1-eth1 s1-eth2:h1-eth0 s1-eth3:h2-eth0 s1-eth4:h3-eth0 s1-eth5:h4-eth0 s1-eth6:h5-eth0
s2 lo: s2-eth1:s1-eth1 s2-eth2:s3-eth1 s2-eth3:h6-eth0 s2-eth4:h7-eth0 s2-eth5:h8-eth0 s2-eth6:h9-eth0 s2-eth7:h10-eth0
s3 lo: s3-eth1:s2-eth2 s3-eth2:h11-eth0 s3-eth3:h12-eth0 s3-eth4:h13-eth0 s3-eth5:h14-eth0 s3-eth6:h15-eth0
c0r code here

```

Hình 2.2: Cấu trúc kết nối của mạng lưới.

Controller được kết nối đến địa chỉ 127.0.0.1 với port là 6653, tại đây chúng tôi triển khai

controller có các chức năng như một Firewall với vai trò ngăn chặn các tương tác có rủi ro tiềm ẩn giữa các host trong mạng lưới.

## 2.2 Firewall

Firewall được xây dựng với vai trò quản lý và ngăn chặn các rủi ro trong mạng lưới, với hai chức năng chính: chức năng thứ nhất là ngăn chặn các packet được truyền từ host này đến host khác khi mà flow này match với rules trong Firewall, chức năng thứ hai là ngăn chặn packets khi một host gửi quá nhiều packets trong một thời gian ngắn, với mục đích ngăn chặn các hành vi bất thường trong network.

Các rules của Firewall được biểu diễn trong database với cấu trúc được mô tả như hình 2.3. Khi flow match với dữ liệu trong database theo quy tắc của Firewall thì flow này sẽ bị drop. src\_ip và dst\_ip là địa chỉ ip của packets, protocol gồm có 3 loại là ICMP, TCP và UDP. src\_port và dst\_port là number port của nguồn và đích trong protocol TCP và UDP. actions được dùng để định nghĩa flow có được cho phép forwarding hay không, với giá trị DROP thì Firewall sẽ chặn packets lại, nếu ALLOW thì dù cho flow có match với rules thì flow vẫn được cho phép forwarding trong network.



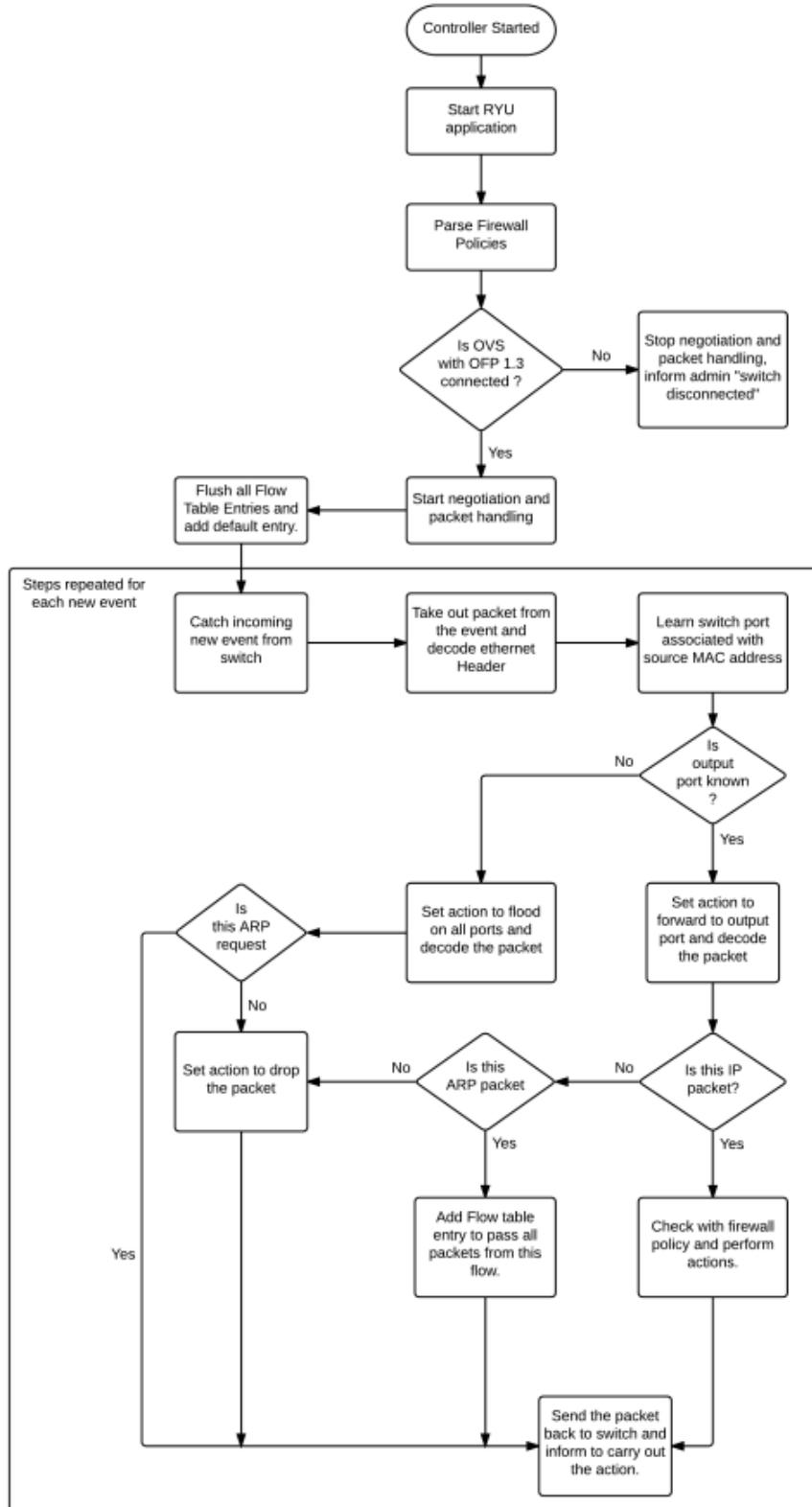
```
CREATE TABLE firewall_rules (
    id INTEGER PRIMARY KEY,
    src_ip TEXT,
    dst_ip TEXT,
    protocol TEXT,
    src_port INTEGER,
    dst_port INTEGER,
    state TEXT,
    action TEXT
);
```

codesnap.dev

Hình 2.3: Quy tắc của Firewall được biểu diễn trong database.

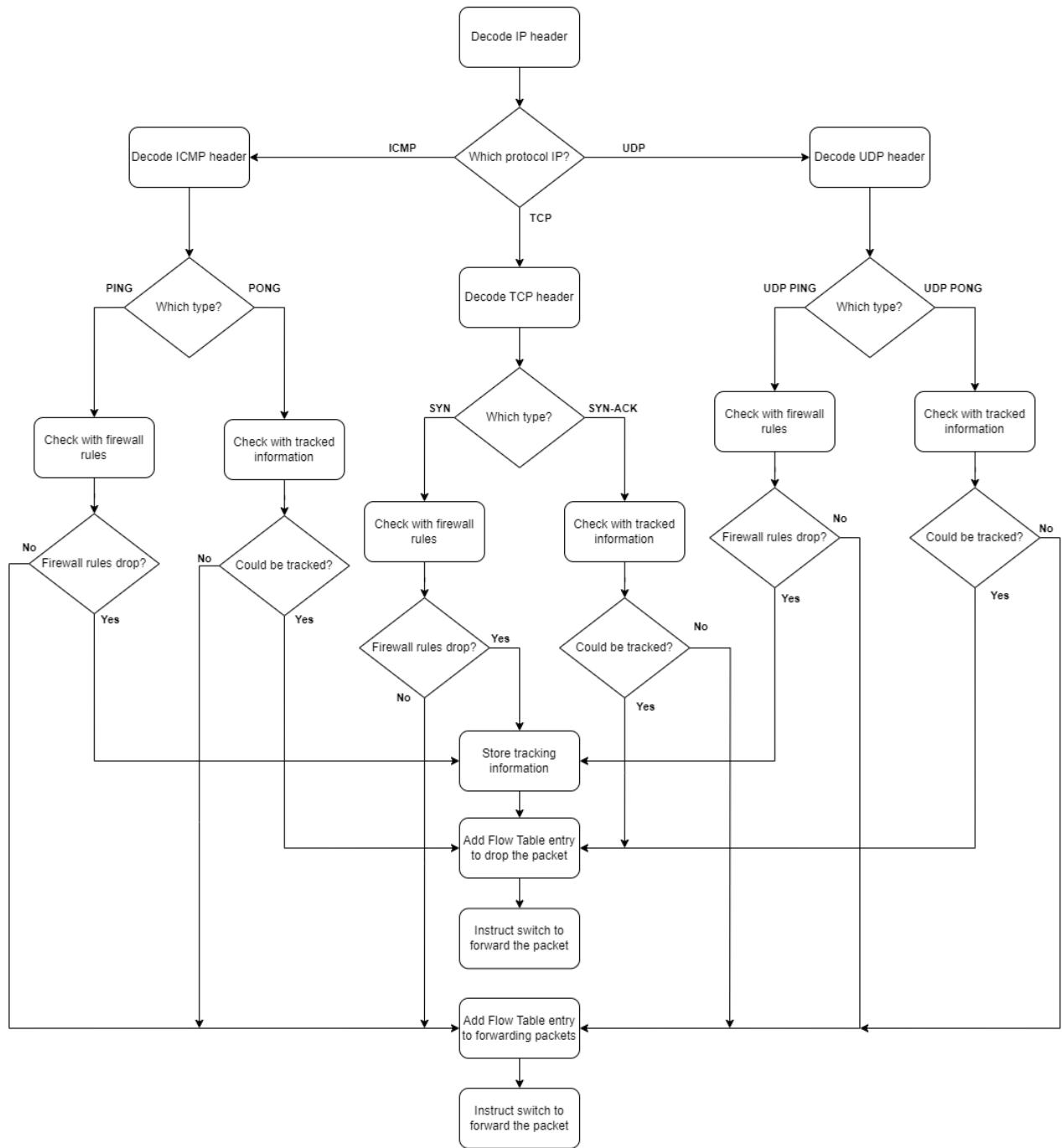
Công nghệ sử dụng để triển khai Firewall như đã được nhắc đến ở trước đó là Ryu với ngôn ngữ lập trình python3, ngoài ra chúng tôi sử dụng sqlite để ứng dụng database vào Firewall với mục đích triển khai Firewall theo thời gian thực, sqlite là một công cụ nhẹ, dễ dàng triển khai, phù hợp với tài nguyên trên các thiết bị mạng. Database được triển khai để lưu trữ các flows bị chặn bởi Firewall cũng như giúp giảm dung lượng nhớ cho các thiết bị định hướng trong network.

Hình 2.4 mô tả nguyên lý hoạt động của controller. Khi Controller được khởi động, nó sẽ trích xuất dữ liệu trong database, sau đó khi có packets đi đến, controller sẽ trích xuất thông tin từ packets và kiểm tra tính hợp lệ của packets với các quy tắc của Firewall, sau đó controller sẽ gửi về switch hành động thích hợp cho packet vừa được xử lý.



Hình 2.4: Luồng hoạt động của controller.

Tại hình 2.5 mô tả chi tiết cấu trúc và nguyên lý hoạt động của Firewall. Đầu tiên ta sẽ kiểm



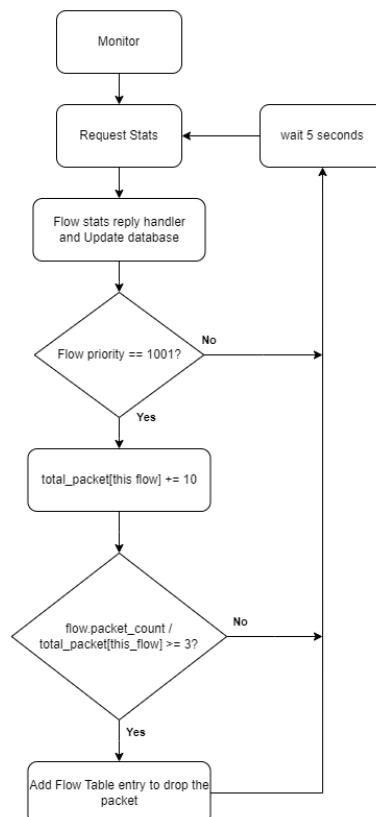
Hình 2.5: Nguyên lý hoạt động của Firewall.

tra protocol của packet thuộc loại nào, có 3 trường hợp: ICMP, TCP và UDP. Sau khi xác định được protocol của packet, trong trường hợp protocol là ICMP thì tiếp tục kiểm tra kiểu gói, bao gồm PING và PONG, trong trường hợp PING (nếu ping thông thường) thì Firewall sẽ check rules, ta sẽ so sánh thông tin từ packet có match với các flows tồn tại trong database (như mô tả tại hình 2.6, nếu match và actions là DROP thì packet sẽ bị chặn lại và lưu vào một mảng (tracking information) dùng để theo dõi gói này và chặn nó mỗi khi controller nhận gói này trong tương lai, nếu là kiểu PONG thì ta sẽ kiểm tra thông tin tracking information, nếu

src_ip	dst_ip	protocol	src_port	dst_port	state	action
10.0.0.1	10.0.0.2	ICMP	-	-	PING	DROP
10.0.0.1	10.0.0.2	TCP	1000	8080	NEW	DROP
10.0.0.1	10.0.0.2	UDP	1000	8080	-	DROP

Hình 2.6: Mô tả các flows trong rules Firewall.

flow match với tracking information thì ta sẽ chặn packet từ flow này, nếu flow không match với database cũng như với tracking information thì packet được forwarding. Tương tự như vậy với hai protocol còn lại là TCP và UDP, đối với TCP thì ta sẽ xử lý chặn hai kiểu là SYN và SYN-ACK tương ứng với các flow tồn tại trong database.



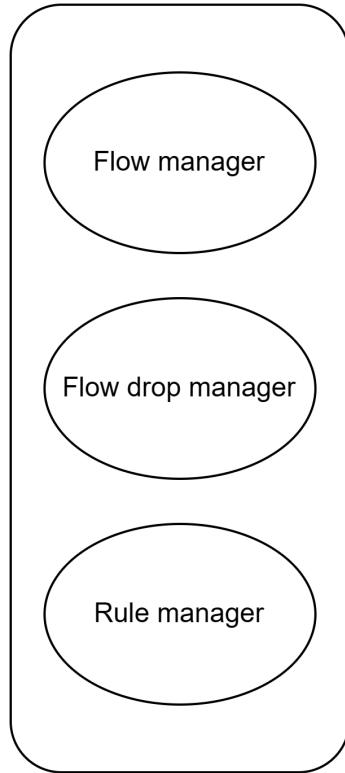
Hình 2.7: Giám sát tính bất thường trong network.

Ngoài việc chặn các flow match với database, Firewall còn có chức năng khác là giám sát sự bất thường trong network. Hình 2.7 mô tả hoạt động của việc giám sát các gói forwarding trong network. Firewall sẽ giám sát các packets thông qua Stats trong các switch, chúng tôi sử dụng hub, một luồng hoạt động độc lập để cập nhật dữ liệu Stats từ switch. Đầu tiên Firewall gọi

Stats trong các switch, sau khi có được Stats, ta sẽ lọc các flows nào có priority = 1001, trong hệ thống này, các flows bị Firewall chặn có priority là 1002 trong khi các flows được forwarding có priority là 1001, khi giám sát packets trong network thì ta chỉ quan tâm đến các packets có thể forwarding. Tiếp theo chúng tôi lấy giá trị trung bình số lượng packet trong 5 giây, nếu trong 5 giây số lượng packets lớp hơn 60 packets, thì ta sẽ chặn packet này và lưu vào Flow Table, và tiếp tục kiểm tra lưu lượng các packets khác xuyên suốt thời gian triển khai hệ thống.

### 2.3 Application

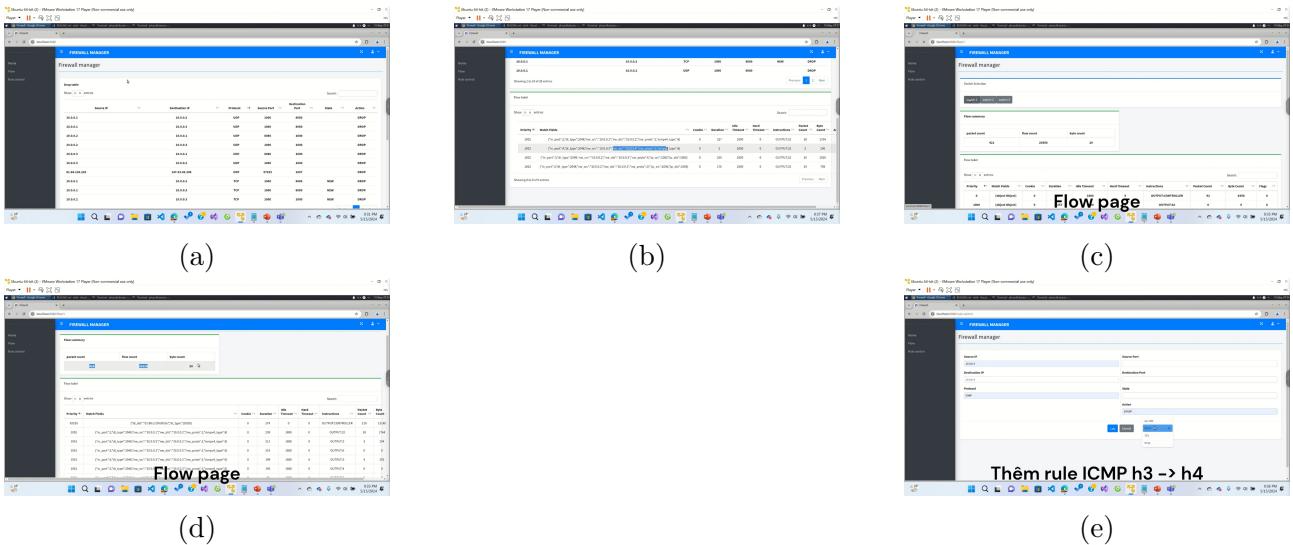
Xây dựng ứng dụng cho phép kiểm soát thông tin về giám sát lưu lượng mạng thời gian thực, giúp phát hiện và phản ứng nhanh chóng với các mối đe dọa bảo mật; phân tích và báo cáo tích hợp giúp hiểu rõ hơn về các xu hướng lưu lượng và hành vi mạng, từ đó đưa ra các quyết định bảo mật tốt hơn. Giao diện trực quan và các công cụ quản lý mạnh mẽ giúp người dùng dễ dàng thiết lập và duy trì các quy tắc firewall.



Hình 2.8: Các tính năng của ứng dụng

Hình 2.8 mô tả tổng quan các tính năng của ứng dụng gồm có quản lý luồn, luồn chặn, các quy luật.

Giao diện trực quan và các công cụ quản lý mạnh mẽ giúp người dùng dễ dàng thiết lập và duy trì các quy tắc firewall. Hình 2.9 là giao diện của ứng dụng, 2.9a giao diện để quản lý các quy luật, 2.9b để theo dõi các gói bị chặn, 2.9c và 2.9d là trang để quản lý toàn bộ luồn, 2.9e là giao diện để thêm các quy luật.



Hình 2.9: Giao diện người dùng

### 3 Thủ nghiệm

Trong phần này thực hiện thử nghiệm trên network như đã giới thiệu trong 2.1 và ứng dụng firewall mà nhóm đã xây dựng. Trong đó 3.1 thực hiện kiểm tra các rule đã được lưu sẵn trong database, 3.2 sẽ tiến hành thêm các rule từ ứng dụng web là kiểm tra các rule đã thêm, 3.3 sẽ thực hiện ping liên tục và firewall sẽ tự động chặn.

#### 3.1 Ping theo database

Ping theo database là thực hiện kiểm tra 3 trường hợp ICMP, TCP, UDP xem firewall có thực hiện chặn các luồng vi phạm đã được lưu trong database.

##### 3.1.1 ICMP

Để thực hiện ta dùng lệnh "h1 ping h2" để ping ICMP từ host1(h1) đến host2(h2).

Priority ↑↓	Match Fields	Cookie ↑↓	Duration ↑↓	Idle Timeout ↑↓	Hard Timeout ↑↓	Instructions ↑↓	Packet Count ↑↓	Byte Count ↑↓	Actions ↑↓
1002	{"in_port":2,"dl_type":2048,"nw_src":"10.0.0.1","nw_dst":"10.0.0.2","nw_proto":1,"icmpv4_type":8}	0	35	1800	0	OUTPUT:32	18	1764	DROP

Hình 3.1: Kết quả h1 đến h2 bị chặn

Khi đó firewall sẽ chặn và hiển thị kết quả lên bảng Flow drop như 3.1.

Tương tự ta sẽ ping từ h2 đến h3 và h5 như 3.2 đây là các luồng chưa được lưu trong database và có thể thấy các gói có thể được gửi đi thành công bên cạnh đó các gói thành công cũng được hiển thị ở trang Flow.

```

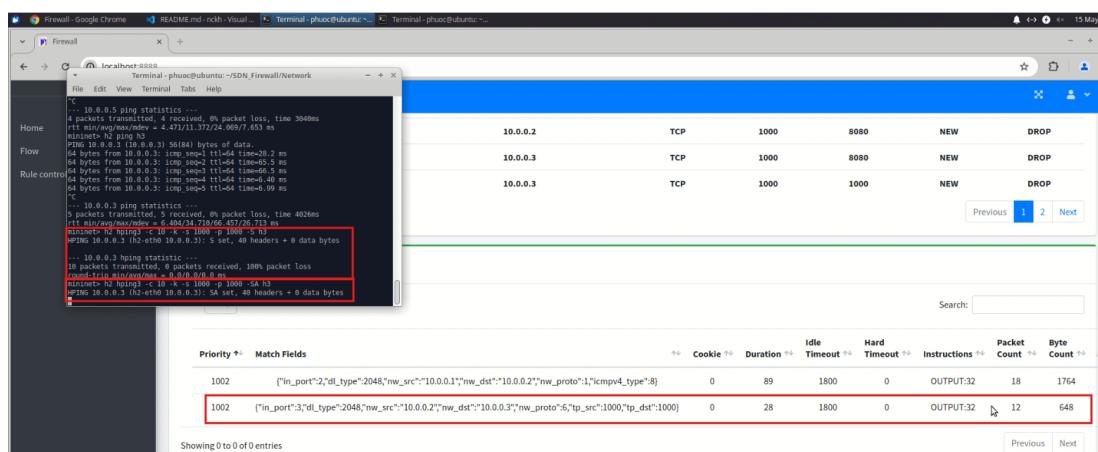
mininet> h2 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=24.1 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=6.39 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=4.47 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=10.6 ms
^C
--- 10.0.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3040ms
rtt min/avg/max/mdev = 4.471/11.372/24.069/7.653 ms
mininet> h2 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=28.2 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=65.5 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=66.5 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=6.40 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=6.99 ms
^C
--- 10.0.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4026ms
rtt min/avg/max/mdev = 6.404/34.710/66.457/26.713 ms

```

Hình 3.2: Kết quả h2 đến h3 và h5

### 3.1.2 TCP

TCP được thực hiện với h2 bắt đầu kết nối với h3. Khi nhận được gói SYN và SYN-ACK, ứng dụng Firewall sẽ kiểm tra yêu cầu khởi tạo kết nối mới này với firewall rule. Khi đó cả 2 gói đều bị chặn như trong 3.3. Các thử nghiệm khác cũng được thực hiện để xem tính chính xác của ứng dụng với các kết hợp cổng khác nhau và với các host khác nhau (cả hai đều được chấp nhận). Kết quả của các thử nghiệm như vậy được thể hiện trong Hình 3.4.



Hình 3.3: Kết quả TCP ping bị chặn

```

round-trip min/avg/max = 0.0/0.0/0.0 ms
mininet> h2 hping3 -c 10 -k -s 1000 -p 1000 -S h4
HPING 10.0.0.4 (h2-eth0 10.0.0.4): S set, 40 headers + 0 data bytes
len=40 ip=10.0.0.4 ttl=64 DF id=0 sport=1000 flags=RA seq=0 win=0 rtt=172.4 ms
DUP! len=40 ip=10.0.0.4 ttl=64 DF id=0 sport=1000 flags=RA seq=0 win=0 rtt=1050.4 ms
DUP! len=40 ip=10.0.0.4 ttl=64 DF id=0 sport=1000 flags=RA seq=0 win=0 rtt=2070.3 ms
DUP! len=40 ip=10.0.0.4 ttl=64 DF id=0 sport=1000 flags=RA seq=0 win=0 rtt=3070.4 ms
DUP! len=40 ip=10.0.0.4 ttl=64 DF id=0 sport=1000 flags=RA seq=0 win=0 rtt=4070.5 ms
`C
--- 10.0.0.4 hping statistic ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 172.4/2086.9/4070.6 ms
mininet> h2 hping3 -c 10 -k -s 1000 -p 1000 -S h7
HPING 10.0.0.7 (h2-eth0 10.0.0.7): S set, 40 headers + 0 data bytes
len=40 ip=10.0.0.7 ttl=64 DF id=0 sport=1000 flags=RA seq=0 win=0 rtt=71.8 ms
DUP! len=40 ip=10.0.0.7 ttl=64 DF id=0 sport=1000 flags=RA seq=0 win=0 rtt=1004.6 ms
DUP! len=40 ip=10.0.0.7 ttl=64 DF id=0 sport=1000 flags=RA seq=0 win=0 rtt=2059.1 ms

```

Hình 3.4: Kết quả TCP ping được chấp nhận

### 3.1.3 UDP

UDP được thực hiện tương tự như TCP. Khi thực hiện ping UDP với luồng có trong database như từ h2 đến h3 thì sẽ bị chặn, còn với các luồng không có trong database sẽ được chấp nhận, kết quả được thể hiện trong hình 3.5 và 3.6

Priority	Match Fields	Cookie	Duration	Idle Timeout	Hard Timeout	Instructions	Packet Count	Byte Count
1002	["in_port":2,"dl_type":2048,"nw_src":"10.0.0.1","nw_dst":"10.0.0.2","nw_proto":1,"lcmv4_type":8]	0	164	1800	0	OUTPUT:32	18	1764
1002	["in_port":3,"dl_type":2048,"nw_src":"10.0.0.2","nw_dst":"10.0.0.3","nw_proto":6,"tp_src":1000,"tp_dst":1000]	0	103	1800	0	OUTPUT:32	19	1026
1002	["in_port":3,"dl_type":2048,"nw_src":"10.0.0.2","nw_dst":"10.0.0.3","nw_proto":17,"tp_src":1000,"tp_dst":1000]	0	23	1800	0	OUTPUT:32	9	378

Hình 3.5: Kết quả UDP ping bị chặn

```

Terminal - phuoc@ubuntu: ~/SDN_Firewall/Network
File Edit View Terminal Tabs Help
--- 10.0.0.3 hping statistic ---
10 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
mininet> h2 hping3 --udp -c 10 -k -s 1000 -p 1000 -S h4
HPING 10.0.0.4 (h2-eth0 10.0.0.4): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from ip=10.0.0.4 name=UNKNOWN
status=0 port=1000 seq=0
ICMP Port Unreachable from ip=10.0.0.4 name=UNKNOWN
status=1 port=1000 seq=0
ICMP Port Unreachable from ip=10.0.0.4 name=UNKNOWN
status=1 port=1000 seq=0
ICMP Port Unreachable from ip=10.0.0.4 name=UNKNOWN
status=1 port=1000 seq=0
ICMP Port Unreachable from ip=10.0.0.4 name=UNKNOWN
status=1 port=1000 seq=0
ICMP Port Unreachable from ip=10.0.0.4 name=UNKNOWN
status=1 port=1000 seq=0
ICMP Port Unreachable from ip=10.0.0.4 name=UNKNOWN
status=1 port=1000 seq=0
ICMP Port Unreachable from ip=10.0.0.4 name=UNKNOWN
status=1 port=1000 seq=0
ICMP Port Unreachable from ip=10.0.0.4 name=UNKNOWN
status=1 port=1000 seq=0
ICMP Port Unreachable from ip=10.0.0.4 name=UNKNOWN
status=1 port=1000 seq=0

```

Hình 3.6: Kết quả UDP ping được chấp nhận

### 3.2 Thêm rule bằng ứng dụng web

Để dễ dàng phát hiện, ngăn chặn kịp thời và cải thiện khả năng phản ứng trước các sự cố bảo mật, quản trị viên có thể thực hiện ngăn chặn các luồng bất thường ngay trực tiếp trên ứng dụng web. Để ngăn chặn các luồng bất thường ta nhập thông tin vào bảng như Hình 3.7 sau đó lưu, dữ liệu sẽ được cập nhật thời gian thực vào database dựa vào đó firewall sẽ chặn luồng vừa thêm vào.

Source IP	Source Port
10.0.0.3	
Destination IP	Destination Port
10.0.0.4	
Protocol	State
ICMP	
Action	
DROP	
<input type="button" value="Lưu"/> <input type="button" value="Cancel"/>	

Hình 3.7: Bảng nhập thông tin

Ta thử nghiệm tính năng với cả ICMP, TCP, UDP. ICMP ta thêm luồng từ h3 đến h4 sau đó thử ping lại và kết quả luồng này bị chặn như trong Hình 3.8. Với TCP ta thêm luồng h5 đến

h9, UDP ta thêm luồng từ h4 đến h8 kết quả các luồng này bị chặn như trong Hình 3.8. Để khách quan hơn ta ping lại luồng vừa thêm vào tuy nhiên khác cổng thì firewall vẫn chấp nhận như Hình 3.9.

Flow table								
Priority ↑	Match Fields	Cookie ↑	Duration ↑	Idle Timeout ↑	Hard Timeout ↑	Instructions ↑	Packet Count ↑	Byte Count ↑
1002	{"in_port":2,"dl_type":2048,"nw_src":"10.0.0.1","nw_dst":"10.0.0.2","nw_proto":1,"icmpv4_type":8}	0	595	1800	0	OUTPUT:32	18	1764
1002	{"in_port":4,"dl_type":2048,"nw_src":"10.0.0.3","nw_dst":"10.0.0.4","nw_proto":1,"icmpv4_type":8}	0	280	1800	0	OUTPUT:32	11	1078
1002	{"in_port":3,"dl_type":2048,"nw_src":"10.0.0.2","nw_dst":"10.0.0.3","nw_proto":6,"tp_src":1000,"tp_dst":1000}	0	533	1800	0	OUTPUT:32	19	1026
1002	{"in_port":3,"dl_type":2048,"nw_src":"10.0.0.2","nw_dst":"10.0.0.3","nw_proto":17,"tp_src":1000,"tp_dst":1000}	0	454	1800	0	OUTPUT:32	19	798
1002	{"in_port":6,"dl_type":2048,"nw_src":"10.0.0.5","nw_dst":"10.0.0.9","nw_proto":6,"tp_src":1000,"tp_dst":1000}	0	187	1800	0	OUTPUT:32	9	486
1002	{"in_port":5,"dl_type":2048,"nw_src":"10.0.0.4","nw_dst":"10.0.0.8","nw_proto":17,"tp_src":1000,"tp_dst":8080}	0	87	1800	0	OUTPUT:32	9	378

Hình 3.8: Thông tin các luồng vừa thêm vào bị chặn

```

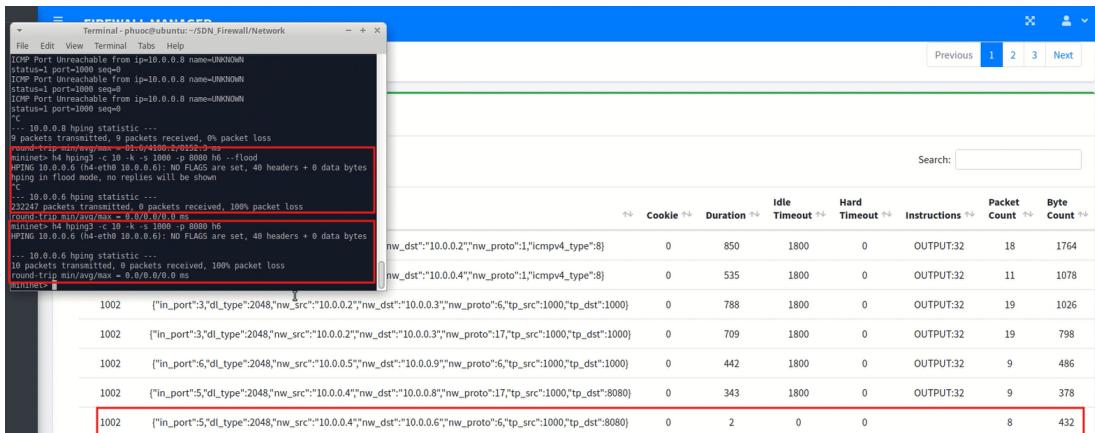
mininet> h5 hping3 -c 10 -k -s 1000 -p 8080 -S h9
HPING 10.0.0.9 (h5-eth0 10.0.0.9): S set, 40 headers + 0 data bytes
len=40 ip=10.0.0.9 ttl=64 DF id=0 sport=8080 flags=RA seq=0 win=0 rtt=96.8 ms
DUP! len=40 ip=10.0.0.9 ttl=64 DF id=0 sport=8080 flags=RA seq=0 win=0 rtt=101.2 ms
DUP! len=40 ip=10.0.0.9 ttl=64 DF id=0 sport=8080 flags=RA seq=0 win=0 rtt=200.8 ms
DUP! len=40 ip=10.0.0.9 ttl=64 DF id=0 sport=8080 flags=RA seq=0 win=0 rtt=302.8 ms
^C
--- 10.0.0.9 hping statistic ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 26.8/1535.2/3023.0 ms
mininet> h4 hping3 --udp -c 10 -k -s 1000 -p 1000 -S h8
HPING 10.0.0.8 (h4-eth0 10.0.0.8): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from ip=10.0.0.8 name=UNKNOWN
status=0 port=1000 seq=0
ICMP Port Unreachable from ip=10.0.0.8 name=UNKNOWN
status=1 port=1000 seq=0
ICMP Port Unreachable from ip=10.0.0.8 name=UNKNOWN
status=1 port=1000 seq=0
ICMP Port Unreachable from ip=10.0.0.8 name=UNKNOWN
status=1 port=1000 seq=0

```

Hình 3.9: Kết quả khi ping khác cổng

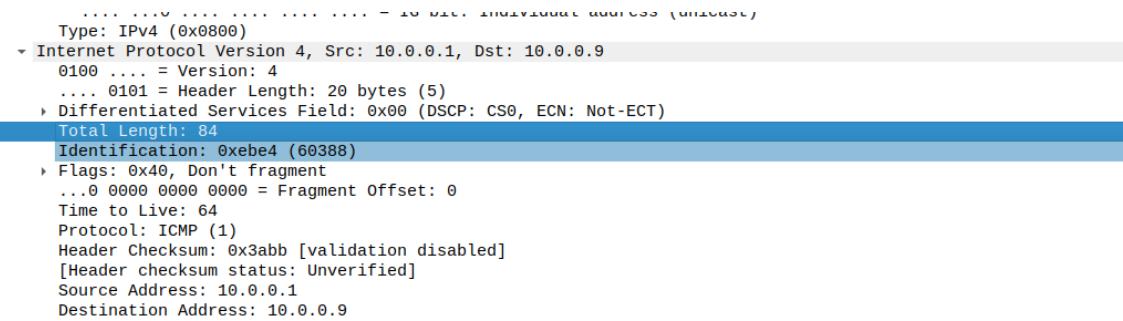
### 3.3 Ping liên tục

Việc thực hiện giám sát bằng cách thông thường là chưa đủ, tự động hóa quy trình phát hiện ngăn chặn là vô cùng cần thiết. Để thực nghiệm ta sẽ ping liên tục từ h4 đến h6 trong tầm khoảng 5s sau đó ping lại có thể thấy firewall sẽ tự động cập nhật và chặn luồng này như Hình 3.10.

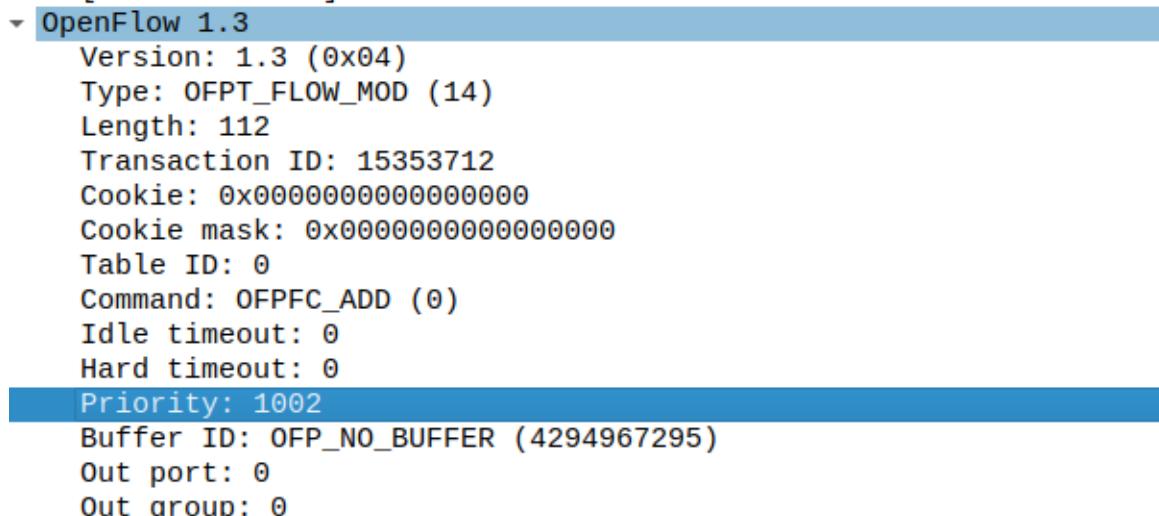


Hình 3.10: Thực hiện ping liên tục từ h4 đến h6

### 3.4 Giám sát với Wireshark



Hình 3.11: Packet request từ h1 đến h10.



Hình 3.12: Packet request từ h1 đến h10.

Hình 3.11 mô tả packet có protocol ICMP được truyền từ h1 đến h4, thời gian giữa h1

request đến h4 và h4 reply đến h1 là 0.011 giây. Đối với các packets bị chặn, như biếu diễn tại hình 3.12 (với priority là 1002) thì thời gian để Firewall chặn packet này là 0.00011713 giây.

## 4 Kết luận và đánh giá

Trong báo cáo này, nhóm đã triển khai và thử nghiệm một ứng dụng firewall cho SDN và ứng dụng web được giao tiếp thông qua REST API, nhằm mục đích giám sát và quản lý lưu lượng mạng một cách hiệu quả. Các chức năng chính của firewall này bao gồm: ngăn chặn gói tin theo rule định sẵn; giám sát hành vi bất thường, chẳng hạn như một host gửi quá nhiều gói tin trong một khoảng thời gian ngắn, điều này giúp ngăn chặn các cuộc tấn công từ chối dịch vụ (DoS) và các hành vi gây rối khác; Ứng dụng cung cấp giao diện người dùng trực quan, giúp người quản trị dễ dàng thiết lập và quản lý các quy tắc bảo mật, theo dõi các gói tin bị chặn, và phân tích các xu hướng lưu lượng mạng.

Qua các thử nghiệm thực tế trên mạng lưới mô phỏng với các host từ h1 đến h15, ứng dụng firewall đã chứng minh được khả năng hoạt động hiệu quả trong việc ngăn chặn các luồng vi phạm đã được lưu trong cơ sở dữ liệu. Các thử nghiệm ping với giao thức ICMP, TCP, và UDP đều cho kết quả phù hợp với các quy tắc bảo mật đã định nghĩa.

Kết quả này khẳng định rằng hệ thống Firewall SDN và ứng dụng web mà nhóm xây dựng đã đáp ứng được yêu cầu cơ bản và đem lại hiệu quả trong việc ngăn các mối đe dọa.

## Tài liệu tham khảo

- [1] Shah, Jay, et al. "Implementation and performance analysis of firewall on open vSwitch." Conducted at the Department of Network Architectures and Network Services, Faculty of Informatics Technical University Munich 29 (2015).

## Phụ lục

- [1] Link Github code Firewall
- [2] Link Github code Application
- [3] Link demo