

# BÁO CÁO THỰC HÀNH

**Môn học:** Công nghệ mạng khả lập trình

**Buổi báo cáo:** Lab 04

**Tên chủ đề:** Lập trình hiện thực và test thử nghiệm một Network Monitor trên mạng SDN/OpenFlow

*GVHD: Phan Xuân Thiện*

*Ngày thực hiện: 18/04/2025*

## **THÔNG TIN CHUNG:**

Lớp: NT541.P21

STT	Họ và tên	MSSV	Email
1	Đình Huỳnh Gia Bảo	22520101	22520101@gm.uit.edu.vn

## **1. ĐÁNH GIÁ KHÁC:**

Nội dung	Kết quả
Tổng thời gian thực hiện bài thực hành trung bình	3 ngày (18/04/2025 – 21/04/2025)
Link file .pcapng (nếu có)	<a href="#">Link file bắt gói tin</a>
Ý kiến (nếu có) + Khó khăn + Đề xuất ...	
Điểm tự đánh giá	10/10

## BÁO CÁO CHI TIẾT

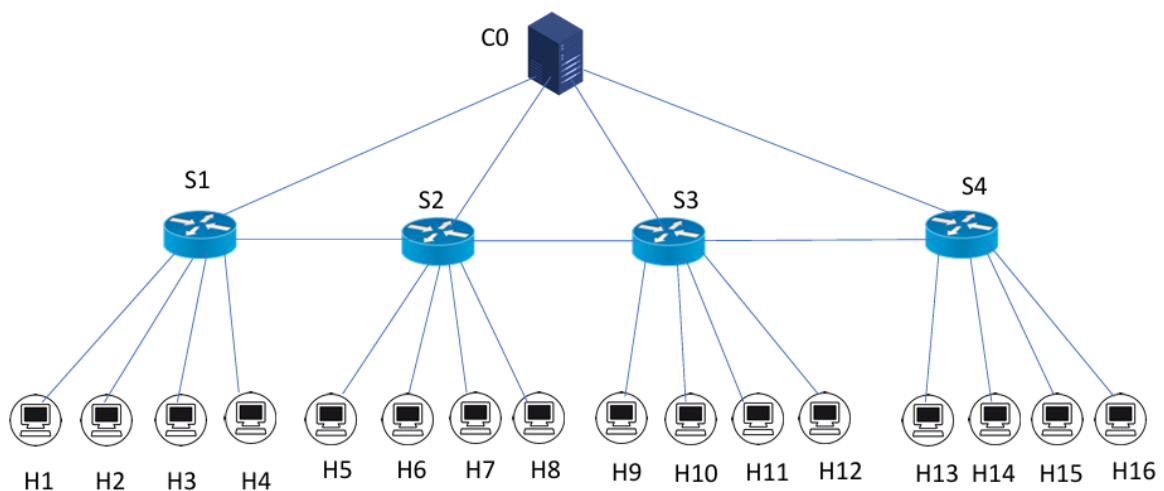
### Mục đích bài thực hành:

- Lập trình và thử nghiệm Network Monitor trên mạng SDN/OpenFlow.
- Lịch truy cập thông tin thống kê từ switch và hiển thị trên Controller.
- Test hiệu năng khi tăng lưu lượng trên mạng.

### Mô hình triển khai:

Phân tích topology bao gồm:

- 1 controller C0
- 4 switch (S1 – S4), mỗi switch kết nối với 4 host:
  - Switch S1 gồm host H1, H2, H3, H4.
  - Switch S2 gồm host H5, H6, H7, H8
  - Switch S3 gồm host H9, H10, H11, H12.
  - Switch S4 gồm host H13, H14, H15, H16.



Hình 1. Mô hình giả lập

### Chuẩn bị:

- VMware Workstation Pro 17: Phần mềm cài đặt và quản lý máy ảo.
- Ryu Controller: Framework của mạng SDN dựa trên kiến trúc thành phần.
- Mininet: Công cụ tạo ra mạng ảo bằng cách lập trình.

- Open vSwitch: Switch ảo mã nguồn mở, được sử dụng phổ biến trong ảo hóa mạng và mạng SDN.
- Wireshark: Công cụ mã nguồn mở được sử dụng để phân tích gói tin
- Ubuntu 20.04: Hệ điều hành Ubuntu phiên bản 20.04

### Triển khai yêu cầu:

#### Câu 1: Tạo mạng SDN/OpenFlow với Topology tùy ý.

##### ➤ Khởi động Mininet.

- Bước 1: Trên hệ điều hành Ubuntu 20.04, thực hiện câu lệnh sau trong Terminal để tiến hành cài đặt Mininet:

```
$ sudo apt install mininet
```

```
giabao@ubuntu:~$ sudo apt install mininet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cgrouper-tools iperf libgroup1 libevent-2.1-7 libpython2-stdlib libpython2.7-minimal
  libpython2.7-stdlib libunbound8 openvswitch-common openvswitch-switch python-pkg-resources
  python2 python2-minimal python2.7 python2.7-minimal python3-openvswitch python3-sortedcontainers
  socat
Suggested packages:
  ethtool openvswitch-doc python-setuptools python2-doc python-tk python2.7-doc binfmt-support
  python-sortedcontainers-doc
The following NEW packages will be installed:
  cgrouper-tools iperf libgroup1 libevent-2.1-7 libpython2-stdlib libpython2.7-minimal
  libpython2.7-stdlib libunbound8 mininet openvswitch-common openvswitch-switch
  python-pkg-resources python2 python2-minimal python2.7 python2.7-minimal python3-openvswitch
  python3-sortedcontainers socat
0 upgraded, 19 newly installed, 0 to remove and 0 not upgraded.
Need to get 7,881 kB of archives.
After this operation, 34.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libpython2.7-minimal amd64 2.7
.18-1-20.04.7 [336 kB]
```

Hình 2. Cài đặt mininet

- Bước 2: Kiểm tra mininet xem đã cài đặt thành công hay chưa
  - Kiểm tra phiên bản của mininet

```
$ mn --version
```

```
giabao@ubuntu:~/lab4$ mn --version
2.2.2
giabao@ubuntu:~/lab4$
```

Hình 3. Kiểm tra phiên bản mininet

- Kiểm tra mininet xem có hỗ trợ switch và bộ điều khiển OpenFlow hay không. Đối với lượt kiểm tra này, chúng ta sẽ sử dụng VSwitch mở ở chế độ Bridge/Standalone.

```
$ sudo mn --switch ovsbr --test pingall
```

```
glabao@ubuntu:~$ sudo mn --switch ovsbr --test pingall
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller

*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
*** Stopping 0 controllers

*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 0.368 seconds
glabao@ubuntu:~$
```

Hình 4. Kiểm tra mininet

➤ **Viết chương trình tạo mạng SDN/OpenFlow với topology như hình 1.**

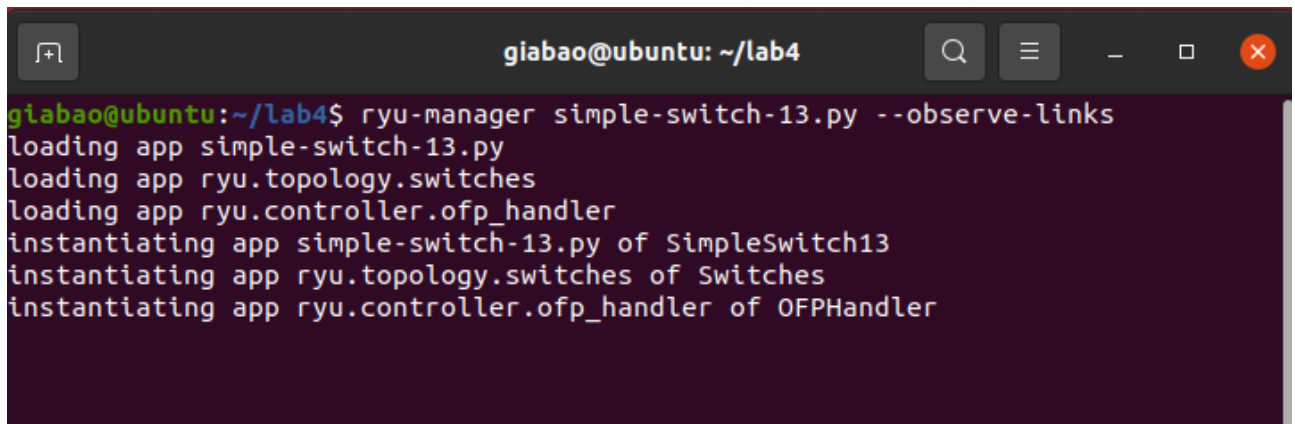
Quan sát hình 1, chúng ta có thể thấy được cách triển khai tổng quát của sơ đồ mạng như sau:

- 1 controller C0
- 4 switch (S1 – S4), mỗi switch kết nối với 4 host:
  - Switch S1 gồm host H1, H2, H3, H4.
  - Switch S2 gồm host H5, H6, H7, H8
  - Switch S3 gồm host H9, H10, H11, H12.
  - Switch S4 gồm host H13, H14, H15, H16.
- Bước 1: Khởi động Ryu Controller với đoạn mã được tùy chỉnh theo nhu cầu:
  - Trước tiên, tạo hoặc chỉnh sửa tập tin điều khiển bằng lệnh:

```
$ nano simple-switch-13.py
```

- Nội dung của tập tin được truy cập qua đường dẫn dưới đây: [simple-switch-13.py](#)
- Sau khi hoàn tất, khởi chạy Ryu Controller với tập tin simple-switch-13.py và bật tính năng quan sát liên kết bằng lệnh:

```
$ ryu-manager simple-switch-13.py --observe-links
```



```
giabao@ubuntu: ~/lab4
giabao@ubuntu:~/lab4$ ryu-manager simple-switch-13.py --observe-links
loading app simple-switch-13.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app simple-switch-13.py of SimpleSwitch13
instantiating app ryu.topology.switches of Switches
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Hình 5. Khởi động Ryu controller

- Bước 2: Tạo topology tùy chỉnh trên Mininet
  - Tiếp theo, tạo hoặc chỉnh sửa tập tin định nghĩa topology bằng lệnh bên dưới. Nội dung của tập tin được truy cập qua đường dẫn: [custom-topology.py](#)

```
$ nano custom-topology.py
```

- Cấp quyền thực thi cho tập tin:

```
$ chmod +x custom-topology.py
```

- Khi đã hoàn thiện tập tin, khởi chạy topology tùy chỉnh này trên Mininet bằng lệnh (dùng python2):

```
$ sudo python custom-topology.py
```



```
giabao@ubuntu:~/lab4$ sudo python custom-topology.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (h1, s1) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (h2, s1) (10.00Mbit 1ms de
lay) (10.00Mbit 1ms delay) (h3, s1) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (h4, s1) (10.00Mbit 1ms delay) (10.00Mbit 1
ms delay) (h5, s2) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (h6, s2) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (h7, s2
) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (h8, s2) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (h9, s3) (10.00Mbit 1ms
delay) (10.00Mbit 1ms delay) (h10, s3) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (h11, s3) (10.00Mbit 1ms delay) (10.00Mb
it 1ms delay) (h12, s3) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (h13, s4) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (
h14, s4) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (h15, s4) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (h16, s4) (20.00
Mbit 2ms delay) (20.00Mbit 2ms delay) (s1, s2) (20.00Mbit 2ms delay) (20.00Mbit 2ms delay) (s2, s3) (20.00Mbit 2ms delay) (
20.00Mbit 2ms delay) (s3, s4)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ... (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (20.00Mbit 2ms delay
) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (20.00Mbit 2ms delay) (20.00Mbit
2ms delay) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (20.00Mbit 2ms delay) (2
0.00Mbit 2ms delay) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (10.00Mbit 1ms delay) (20.00Mbit 2ms
delay)
*** Waiting for network to settle
*** Pre-populating ARP caches
*** Running CLI
*** Starting CLI:
mininet>
```

Hình 6. Khởi chạy topology của Mininet

- Test mạng SDN/OpenFlow được tạo ra, gồm: Test kết nối, test hiệu suất của liên kết giữa hai host bất kỳ trong mạng.
- Sau khi triển khai thành công mô hình mạng SDN theo yêu cầu với Mininet và Ryu Controller, bước tiếp theo là kiểm tra hoạt động của mạng để đảm bảo các thành phần được cấu hình đúng cách. Việc kiểm tra bao gồm hai phần chính:
  - *Kiểm tra kết nối:* Sử dụng lệnh **pingall** để kiểm tra kết nối giữa các host để xác minh rằng kết nối trong mạng hoạt động bình thường.

```
mininet > pingall
```

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12 h13 h14 h15 h16
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12 h13 h14 h15 h16
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12 h13 h14 h15 h16
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12 h13 h14 h15 h16
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12 h13 h14 h15 h16
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h13 h14 h15 h16
h13 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h14 h15 h16
h14 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h15 h16
h15 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h16
h16 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
*** Results: 0% dropped (240/240 received)
```

Hình 7. Lệnh pingall

- Ngoài ra, chúng ta có thể dùng lệnh **ping** để kiểm tra kết nối giữa 2 hosts bất kỳ trong mạng.

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=5.14 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=5.77 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=6.88 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 5.139/5.932/6.884/0.721 ms
```

Hình 8. Kiểm tra 2 hosts bất kỳ

- *Kiểm tra hiệu suất:* Đánh giá băng thông giữa 2 hosts (h1 và h2) bằng công cụ **iperf**, giúp xác định hiệu suất truyền tải dữ liệu trong mạng. Trong đó:
  - h1 đóng vai trò là server đang “lắng nghe” kết nối từ các host khác.
  - h2 gửi dữ liệu đến h1.

```
mininet > h1 iperf -s &
```



```
mininet > h2 iperf -c 5 h1
```

```
mininet> h1 iperf -s &
mininet> h2 ping -c 5 h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=7.06 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=5.86 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=6.31 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=6.09 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=5.78 ms

--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 5.781/6.220/7.059/0.458 ms
mininet> h2 iperf -c h1
-----
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 136 KByte (default)
-----
[  3] local 10.0.0.2 port 42076 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer    Bandwidth
[  3]  0.0-10.9 sec  15.1 MBytes  11.6 Mbits/sec
```

Hình 9. Kiểm tra hiệu suất giữa h1 và h2

## Câu 2: Viết chương trình để hiện thực một network monitor trên mạng SDN/OpenFlow

➤ Sử dụng chương trình Learning switch của Controller (Ryu controller). Viết phần chương trình để thực hiện Request các thông tin mạng (Statistics) từ các switch. Sau đó tổng hợp và in ra các thông tin statistics nhận được trên controller. Tần suất gửi Request: 1 giây, 10 giây, ...

- Gợi ý: dùng thông điệp Flow Statistics Request/Reply hoặc một số loại thông điệp tương tự (Port Statistics Request/Reply, ...)

- Tiến hành viết một chương trình Python sử dụng Ryu controller để làm Network Monitor (giám sát mạng). Chương trình này sẽ kết hợp Learning Switch với việc thu thập thông tin thống kê từ các switch. Nội dung của chương trình ở file sau: [ryu-network-monitor.py](http://ryu-network-monitor.py)

```
$ nano network-monitor.py
```

- Thực hiện lệnh ping giữa các Host trong mạng, quan sát và báo cáo thông tin nhận được (và được in ra console) trên controller.

- Bước 1: Trên terminal Mininet, chạy lệnh ping giữa các host, ví dụ:

```
mininet> h1 ping h2
```

- Bước 2: Quan sát kết quả được in ra console trên controller:

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=9.48 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=6.29 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=6.07 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=5.22 ms
```

Hình 10. Kết quả ở mininet CLI

```
Packet in 1 00:00:00:00:00:01 00:00:00:00:00:02 1
Packet in 1 00:00:00:00:00:02 00:00:00:00:00:01 2

=== Requesting statistics from switches ===

=== Flow Stats Reply from Switch 0000000000000003 ===
In-Port      Src-MAC      Dst-MAC      Out-Port Packets  Bytes
-----
=== Port Stats Reply from Switch 0000000000000003 ===
Port      Rx-Packets  Rx-Bytes  Rx-Errors  Tx-Packets  Tx-Bytes  Tx-Errors
-----
1         109        8674      0          227         16011     0
2         109        8674      0          227         16011     0
3         116        8968      0          234         16305     0
4         132        10592     0          250         17929     0
5         477        38303     0          477         38303     0
6         335        27803     0          395         30323     0

=== Flow Stats Reply from Switch 0000000000000001 ===
In-Port      Src-MAC      Dst-MAC      Out-Port Packets  Bytes
-----
1           00:00:00:00:00:01 00:00:00:00:00:02 2          2          196
2           00:00:00:00:00:02 00:00:00:00:00:01 1          2          196

=== Port Stats Reply from Switch 0000000000000001 ===
Port      Rx-Packets  Rx-Bytes  Rx-Errors  Tx-Packets  Tx-Bytes  Tx-Errors
-----
1         114        9080      0          231         16347     0
2         112        8996      0          229         16263     0
3         109        8702      0          226         15969     0
4         130        10536     0          247         17803     0
5         406        31737     0          346         29217     0

=== Flow Stats Reply from Switch 0000000000000002 ===
In-Port      Src-MAC      Dst-MAC      Out-Port Packets  Bytes
-----
=== Port Stats Reply from Switch 0000000000000002 ===
Port      Rx-Packets  Rx-Bytes  Rx-Errors  Tx-Packets  Tx-Bytes  Tx-Errors
-----
1         109        8702      0          226         15969     0
2         114        8996      0          227         16263     0
```

Hình 11. Kết quả khi ping ở controller console

- Bước 2: Phân tích kết quả trong controller console:

- Sự kiện Packet-In:

```
Packet in 1 00:00:00:00:00:01 00:00:00:00:00:02 1
Packet in 1 00:00:00:00:00:02 00:00:00:00:00:01 2
```

- Đây là gói tin từ H1 gửi đến H2 nhưng chưa có rule xử lý, nên switch gửi thông tin về controller (gói tin Packet\_in).
  - 00:00:00:00:00:01: Địa chỉ MAC của H1.
  - 00:00:00:00:00:02: Địa chỉ MAC của H2.
  - Số 1 và 2 ở cuối: là cổng mà gói tin đến (tức port kết nối với H1 và H2).



• Flow Stats từ Switch 1:

==== Flow Stats Reply from Switch 0000000000000001 ====					
In-Port	Src-MAC	Dst-MAC	Out-Port	Packets	Bytes
1	00:00:00:00:00:01	00:00:00:00:00:02	2	2	196
2	00:00:00:00:00:02	00:00:00:00:00:01	1	2	196

- Đã có 2 flow được cài đặt (một chiều từ H1 → H2 và ngược lại).
- Mỗi flow truyền 2 gói tin, tổng dung lượng 196 bytes.
- Đây là minh chứng cho việc gói tin ICMP (sử dụng câu lệnh **ping**) đã truyền đi thành công.

• Port Stats từ Switch 1:

==== Port Stats Reply from Switch 0000000000000001 ====						
In-Port	Rx-Packets	Rx-Bytes	Rx-Errors	Tx-Packets	Tx-Bytes	Tx-Errors
1	114	9080	0	213	16347	0
2	112	8996	0	229	16263	0
3	109	8702	0	226	15969	0
4	130	10536	0	247	17803	0
5	406	31737	0	346	29217	0

- Đã có 2 flow được cài đặt (một chiều từ H1 → H2 và ngược lại).
- Thống kê này thể hiện số lượng gói nhận và gửi qua từng cổng, bao gồm cả các gói broadcast như ARP, LLDP, ...
- Không có lỗi truyền hoặc nhận.

• Các Switch khác (Switch 2, 3, 4):

- Các switch còn lại cũng trả về port stats và flow stats.
- Tuy không tham gia trực tiếp vào quá trình ping, nhưng vẫn có thống kê về lưu lượng (có thể đến từ các gói ARP, LLDP, hoặc các gói broadcast khác).
- Không có lỗi trên bất kỳ cổng nào.

➤ **Mở wireshark và tiến hành bắt các gói tin trao đổi giữa switch và controller (statistics request và reply). Chụp hình màn hình làm minh chứng và đưa vào báo cáo.**

- Bước 1: Mở terminal mới và thực hiện câu lệnh sau và chọn interface “lo”

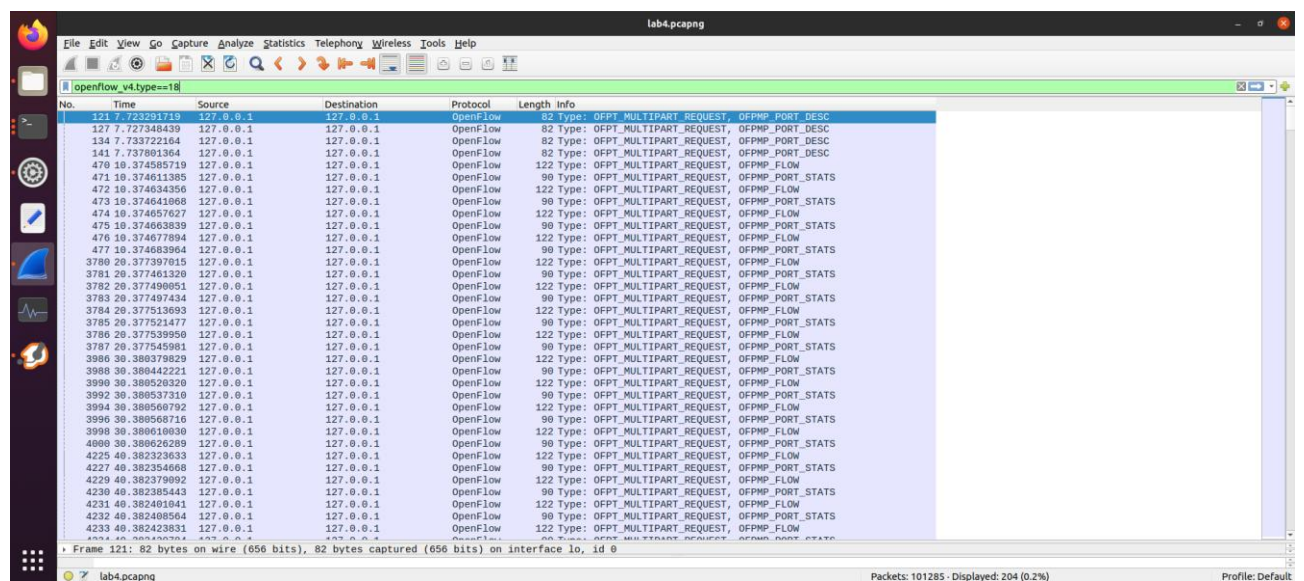
```
$ sudo wireshark
```

- Bước 2: Bắt gói tin Statistics Request và Statistics Reply, sử dụng bộ lọc sau:

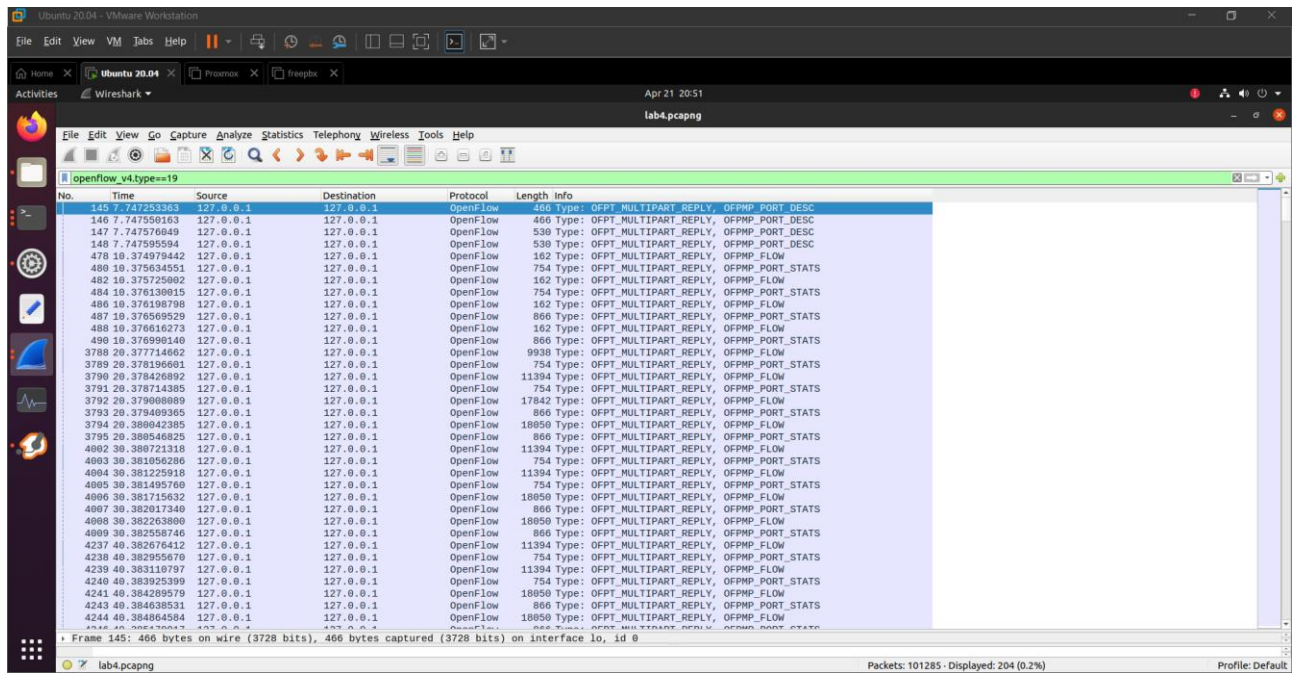
```
$ openflow_v4.type==18      # Statistic Request
```

```
$ openflow_v4.type==19      # Statistic Reply
```

- Bước 3: Quan sát kết quả sau khi áp dụng bộ lọc:



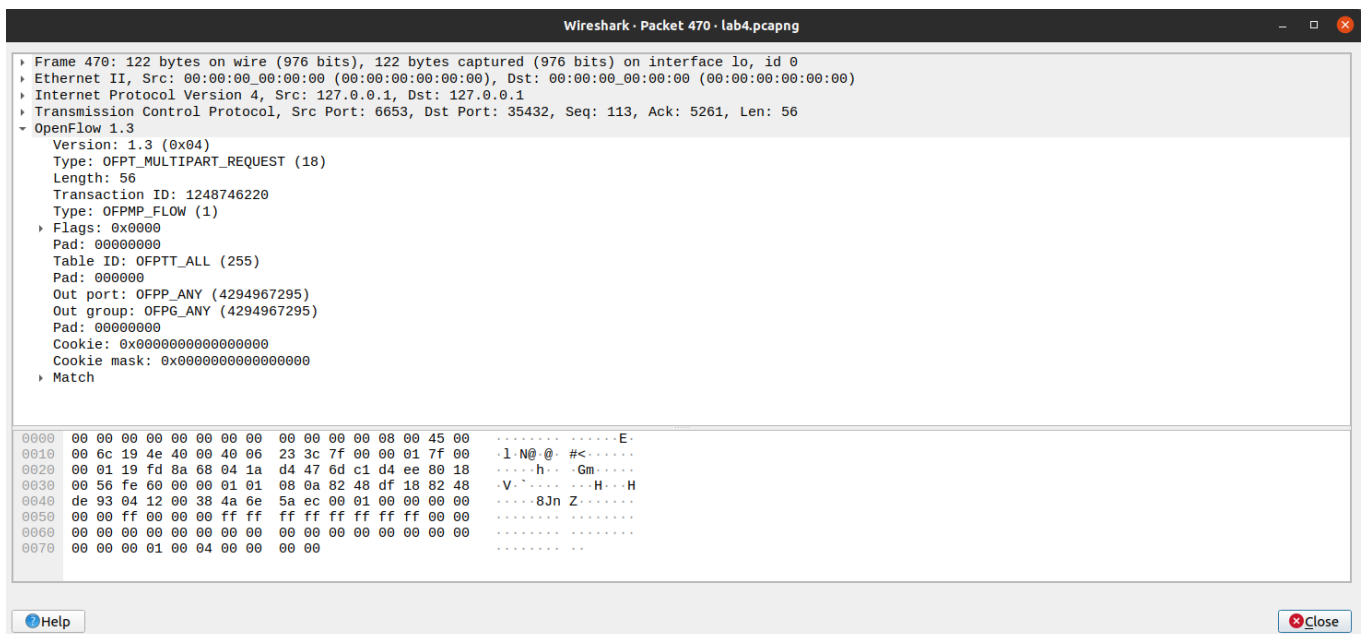
Hình 12. Kết quả khi bắt gói tin Statistics Request



Hình 13. Kết quả khi bắt gói tin Statistics Reply

- Bước 4: Tiến hành phân tích gói tin:

- Gói tin OFPT\_MULTIPART\_REQUEST
  - *OFPMP\_FLOW*: Yêu cầu switch gửi về tất cả thông tin liên quan đến các flow hiện đang tồn tại (trên tất cả các bảng, mọi cổng, mọi nhóm). Nó là cách mà controller lấy dữ liệu thống kê từ switch để hiển thị ở các đoạn log: flow stats, port stats.



Hình 14. Gói tin OFPMP\_FLOW

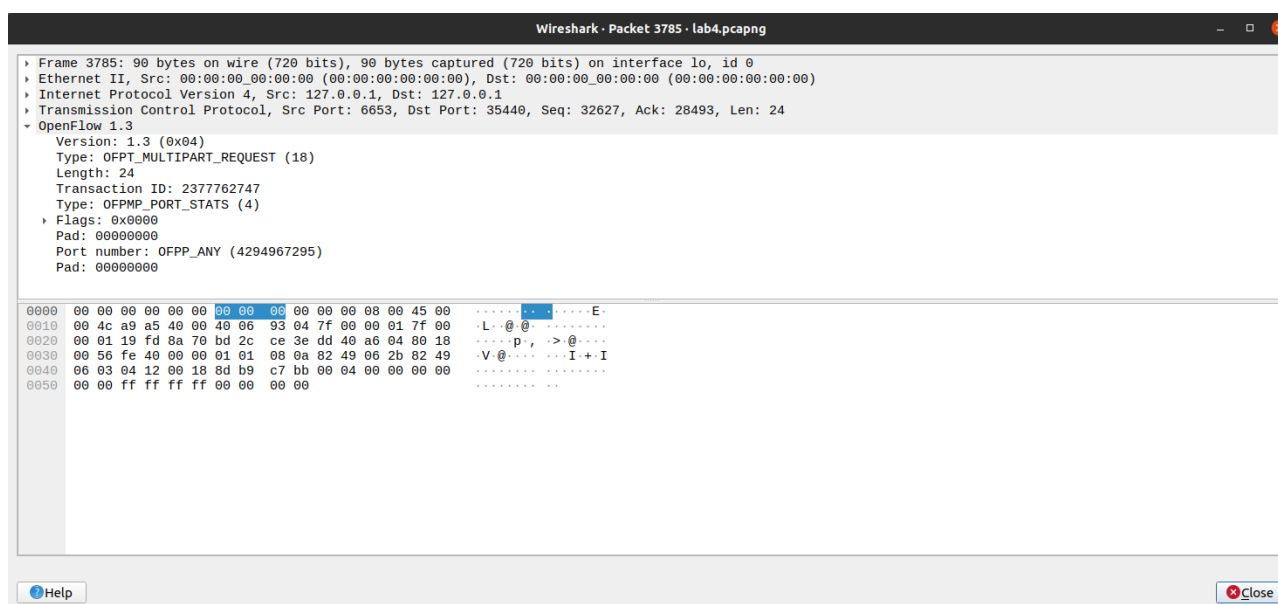
- Lớp giao thức OpenFlow 1.3:

Type	OFPT_MULTIPART_REQUEST (giá trị = 18)
Length	56 bytes
Transaction ID	1248746220 → Dùng để ghép nối với gói phản hồi OFPT_MULTIPART_REPLY
Multipart Type	OFPP_FLOW (1) → Nghĩa là yêu cầu thông tin về các luồng (flows)

- Các trường quan trọng:

Trường	Giá trị	Ý nghĩa
Flags	0x0000	Không có flag đặc biệt
Table ID	OFPP_ALL (255)	Yêu cầu thống kê tất cả các bảng
Out Port	OFPP_ANY (0xffffffff)	Không giới hạn cổng đích
Out Group	OFPG_ANY (0xffffffff)	Không giới hạn nhóm
Cookie	0x0000000000000000	Giá trị cookie để filter (0 là không lọc)
Cookie Mask	0x0000000000000000	Không áp dụng filter cookie
Match	Empty	Yêu cầu thống kê tất cả các flow (không giới hạn điều kiện)

- *OFPT\_MULTIPART\_REPLY*: Yêu cầu thu thập thống kê về cổng (port) trên switch.



Hình 15. Gói tin *OFPT\_MULTIPART\_REPLY*

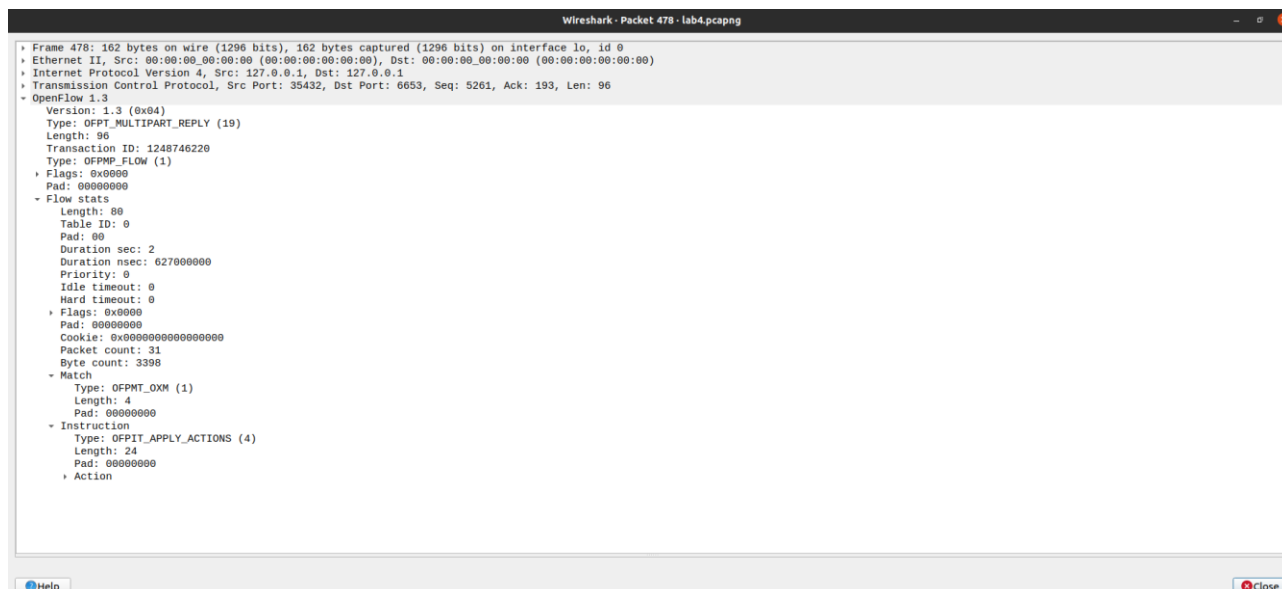
- Lớp giao thức OpenFlow 1.3:

Type	OFPT_MULTIPART_REQUEST (giá trị = 18)
Length	24 bytes
Transaction ID	2377762747 → Dùng để ghép nối với gói yêu cầu OFPT_MULTIPART_REQUEST
Multipart Type	OFPMPT_PORT_STAT (4) → Nghĩa là yêu cầu thông tin về các cổng (ports)

- Các trường quan trọng:

Trường	Giá trị	Ý nghĩa
Flags	0x0000	Không có flag đặc biệt
Port Number	OFPTPT_ALL (4294967295)	Yêu cầu thông kê tất cả các cổng
Padding	00000000	Đệm để đảm bảo căn chỉnh byte

- Gói tin OFPT\_MULTIPART\_REPLY
  - *OFPMPT\_FLOW*: Gói Multipart Reply được gửi từ switch về cho controller, trả lời truy vấn thống kê các luồng (flow stats).



Hình 16. Gói tin *OFPMPT\_FLOW\_STATS*

- Lớp giao thức OpenFlow 1.3:

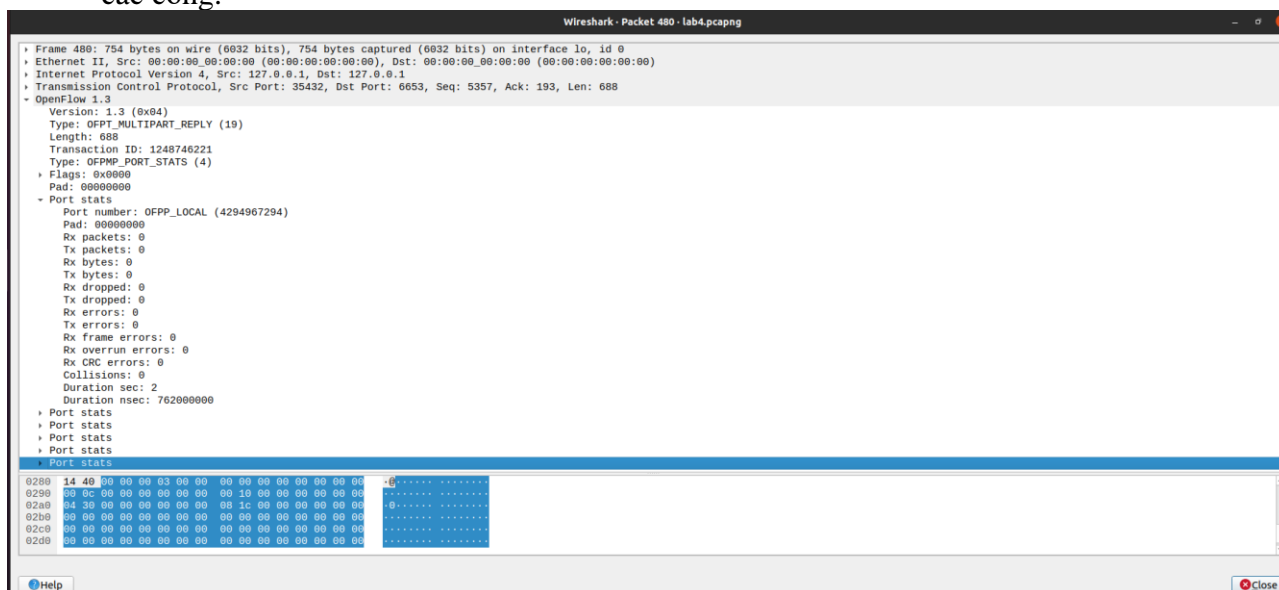
Type	OFPT_MULTIPART_REPLY (giá trị = 19)
Length	96 bytes
Transaction ID	1248746220 → Dùng để trả lời với gói yêu cầu OFPMP_FLOW
Multipart Type	OFPMP_FLOW (1) → Nghĩa là trả lời thông tin về các luồng (flows)

- Các trường quan trọng:

Trường	Mục đích
Length	Tổng độ dài của bản ghi flow entry này (tính theo byte), bao gồm phần match, instruction, và action.
Table ID	ID của bảng luồng (flow table) mà flow này nằm trong. OpenFlow switch có thể có nhiều bảng (table 0, table 1, ...).
Duration Sec / Duration nsec	Thời gian (giây và nano giây) kể từ khi flow được thêm vào bảng. Giúp controller biết tuổi thọ của flow.
Priority	Độ ưu tiên của flow. Flow có priority cao hơn sẽ được so sánh và chọn trước trong quá trình match.
Idle Timeout	Số giây flow sẽ bị xóa nếu không có gói tin nào khớp flow này. Nếu = 0 nghĩa là không có giới hạn idle.
Hard Timeout	Số giây flow sẽ bị xóa sau khi được thêm, dù có gói khớp hay không. Nếu = 0 là không giới hạn thời gian.
Flags	Cờ điều khiển bổ sung. Ở đây là 0x0000 nên không có cờ đặc biệt.
Cookie	Một giá trị 64-bit định danh flow, do controller gán. Dùng để quản lý flow hoặc lọc kết quả thống kê.
Packet Count	Số lượng gói tin đã match với flow này. Giúp kiểm tra mức độ sử dụng của flow.
Byte Count	Tổng số byte từ các gói tin đã match với flow này. Phục vụ mục đích thống kê băng thông.
Match	Thể hiện sự khớp nối đối với gói tin OFPMP_FLOW tương ứng
Instruction	Yêu cầu của gói tin Request



- **OFPMMP\_PORT\_STAT**: Được gửi từ switch về controller nhằm báo cáo số liệu về hiệu suất các cổng.



Hình 17. Gói tin OFPMMP\_PORT\_STAT

- Lớp giao thức OpenFlow 1.3:

Version	1.3 – Phiên bản OpenFlow
Type	OFPMMP_MULTIPART_REPLY
Length	688 byte
Transaction ID	1248746221 – để khớp với request ban đầu từ controller
Multipart Reply Type	OFPMMP_PORT_STATS (4) – phản hồi chứa thông tin thống kê về các cổng

- Các trường quan trọng:

Trường	Mục đích
Rx packets	Số gói tin đã nhận được qua cổng này
Tx packets	Số gói tin đã gửi đi từ cổng này
Rx bytes	Số byte đã nhận qua cổng
Tx bytes	Số byte đã gửi đi qua cổng
Rx dropped	Gói tin nhận bị loại bỏ (ví dụ: quá tải bộ đệm)
Tx dropped	Gói tin gửi bị loại bỏ
Rx errors	Số lỗi nhận gói (CRC error, frame error...)

Tx errors	Số lỗi khi truyền gói
Rx frame errors	Lỗi khung trong khi nhận
Rx overrun errors	Quá tải bộ đệm khi nhận
Rx CRC errors	Lỗi kiểm tra CRC khi nhận
Collisions	Số lần xảy ra va chạm gói (trên mạng Ethernet half-duplex)
Duration sec/nsec	Thời gian công đã tồn tại tính từ khi được kích hoạt

**Câu 3: Tiến hành test chương trình với lưu lượng network traffic lớn. Sử dụng chương trình Traffic Generator có sẵn trong Linux/Window (ví dụ: iperf,...) hoặc chương trình tự cài đặt.**

➤ Tiến hành gửi network traffic với tốc độ cao (vd: 1000 gói tin/giây, 10000 gói tin/giây,...) giữa 2 host bất kỳ trong mạng (vd: từ H4 đến H12,...)

- 1000 gói tin/giây:

o Dùng H4 làm server:

```
$ h4 iperf -s &
```

o Dùng H12 làm client gửi gói tin đến H4:

```
$ h12 iperf -c h4 -u -b 12M -t 10
```

```
mininet> h4 iperf -s &
mininet> h12 iperf -c h4 -u -b 12M -t 10
-----
Client connecting to 10.0.0.4, UDP port 5001
Sending 1470 byte datagrams, IPG target: 934.60 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 3] local 10.0.0.12 port 60391 connected with 10.0.0.4 port 5001
read failed: Connection refused
[ 3] WARNING: did not receive ack of last datagram after 1 tries.
[ ID] Interval          Transfer        Bandwidth
[ 3] 0.0-10.0 sec 15.0 MBytes 12.6 Mbits/sec
[ 3] Sent 10700 datagrams
mininet>
```

Hình 18. Tốc độ 1000 gói tin/giây

- 10000 gói tin/giây

o Dùng H4 làm server:

```
$ h4 iperf -s &
```

o Dùng H12 làm client gửi gói tin đến H4:

```
$ h12 iperf -c h4 -u -b 120M -t 10
```

```
mininet> h12 iperf -c h4 -u -b 120M -t 10
-----
Client connecting to 10.0.0.4, UDP port 5001
Sending 1470 byte datagrams, IPG target: 93.46 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 3] local 10.0.0.12 port 53257 connected with 10.0.0.4 port 5001
read failed: Connection refused
[ 3] WARNING: did not receive ack of last datagram after 1 tries.
[ ID] Interval          Transfer       Bandwidth
[ 3]  0.0-10.0 sec    150 MBytes    126 Mbits/sec
[ 3] Sent 106997 datagrams
mininet>
```

Hình 19. Tốc độ 10000 gói tin/giây

➤ Quan sát và báo cáo thông tin tổng hợp được trên controller

- 1000 gói tin/giây

```
=== Requesting statistics from switches ===

=== Flow Stats Reply from Switch 0000000000000004 ===
In-Port      Src-MAC      Dst-MAC      Out-Port  Packets      Bytes
-----
=== Port Stats Reply from Switch 0000000000000004 ===
Port      Rx-Packets  Rx-Bytes  Rx-Errors  Tx-Packets  Tx-Bytes  Tx-Errors
-----
1         75         5482      0          190         12210     0
2         74         5392      0          190         12190     0
3         74         5392      0          190         12190     0
4         73         5302      0          190         12170     0
5        282        19662     0          222         17142     0

=== Flow Stats Reply from Switch 0000000000000001 ===
In-Port      Src-MAC      Dst-MAC      Out-Port  Packets      Bytes
-----
4             00:00:00:00:00:04  00:00:00:00:00:0c  5          11          5394
5             00:00:00:00:00:0c  00:00:00:00:00:04  4         7066        10682322

=== Port Stats Reply from Switch 0000000000000001 ===
Port      Rx-Packets  Rx-Bytes  Rx-Errors  Tx-Packets  Tx-Bytes  Tx-Errors
-----
1         76         5552      0          191         7066        10682322

=== Port Stats Reply from Switch 0000000000000001 ===
Port      Rx-Packets  Rx-Bytes  Rx-Errors  Tx-Packets  Tx-Bytes  Tx-Errors
```

Hình 20. Kết quả khi gửi 1000 gói tin/giây

- Lệnh iperf từ H12 đến H4 đã sinh ra một lượng lưu lượng UDP tương đối lớn.
  - Dữ liệu được chuyển tiếp qua switch s1 (0000000000000001) và s4 (0000000000000004).
  - Các port từ s4 (Port 1 – 4) có lưu lượng Tx đều nhau, cho thấy switch đang sử dụng cơ chế phân phối tải (ECMP) hoặc có chính sách load balancing giữa các link.
- 10000 gói tin/giây

```

=== Requesting statistics from switches ===

=== Flow Stats Reply from Switch 0000000000000003 ===
In-Port  Src-MAC  Dst-MAC  Out-Port  Packets  Bytes
-----
4         00:00:00:00:00:0c  00:00:00:00:00:04  5         3390    5125680
5         00:00:00:00:00:04  00:00:00:00:00:0c  4          4        2360

=== Port Stats Reply from Switch 0000000000000003 ===
Port  Rx-Packets  Rx-Bytes  Rx-Errors  Tx-Packets  Tx-Bytes  Tx-Errors
-----
1      77        5622      0          194        12756     0
2      77        5622      0          194        12756     0
3      77        5622      0          194        12756     0
4     12997    19539164  0          224        28812     0
5      341       39078     0         13230     19553610  0
6      225       17618     0          285        20138     0

=== Flow Stats Reply from Switch 0000000000000001 ===
In-Port  Src-MAC  Dst-MAC  Out-Port  Packets  Bytes
-----
4         00:00:00:00:00:04  00:00:00:00:00:0c  5          4        2360
5         00:00:00:00:00:0c  00:00:00:00:00:04  4         3380    5110560

=== Port Stats Reply from Switch 0000000000000001 ===
Port  Rx-Packets  Rx-Bytes  Rx-Errors  Tx-Packets  Tx-Bytes  Tx-Errors
-----
1      78        5692      0          194        12756     0
2      77        5622      0          194        12756     0
3      77        5622      0          194        12756     0
4      107       21678     0         13093     19513104  0
5     13202    19547702  0          255        33674     0

=== Flow Stats Reply from Switch 0000000000000002 ===
In-Port  Src-MAC  Dst-MAC  Out-Port  Packets  Bytes
-----
5         00:00:00:00:00:04  00:00:00:00:00:0c  6          4        2360

```

Hình 21. Kết quả khi gửi 10000 gói tin/giây

- Lưu lượng đi rõ ràng từ H12 → H4 thông qua các switch trung gian.
- Dữ liệu đi chính yếu là một chiều (UDP), lưu lượng phản hồi rất nhỏ (4 gói), điều này bình thường trong kiểm thử UDP.
- Các giá trị Tx-Packets và Tx-Bytes khớp với tốc độ truyền 120 Mbps trong 10 giây:
  - 120 Mbps x 10 giây = 1.2 Gbit = 150 MB
  - Tổng số Bytes gửi (từ s3 port 5): khoảng **19.5 MB** (chưa đạt kỳ vọng do UDP có thể bị drop hoặc không gửi đủ nếu thiếu tài nguyên/băng thông thực).