

## รายงานการทดลอง

การศึกษาและทดลองใช้ซอฟต์แวร์เพื่อจำลองโครงข่ายที่ถูกระบุโดยซอฟต์แวร์  
(Software-Defined Networking)

เสนอ

รศ.ดร.กุลธิดา ไรจน์วิบูลย์ชัย

คณะผู้ทดลอง

นายพิธาน หาญพานิชย์พันธ์ 5631068821

รายงานนี้เป็นส่วนหนึ่งของวิชา

เอกัตศึกษาทางวิศวกรรมคอมพิวเตอร์ 4

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ภาคการศึกษาปลาย ปีการศึกษา 2558

## สารบัญ

	หน้า
สารบัญ.....	2
สารบัญภาพ.....	3
บทคัดย่อ.....	4
บทที่ 1 บทนำ.....	5
1.1 ความเป็นมาและความสำคัญของปัญหา	
1.2 วัตถุประสงค์ของการทดลอง	
1.3 ประโยชน์ที่คาดว่าจะได้รับ	
1.4 ขอบเขตของการทดลอง	
บทที่ 2 งานวิจัยที่เกี่ยวข้อง.....	6
บทที่ 3 วิธีดำเนินการทดลอง.....	7
3.1 วิธีดำเนินการ	
บทที่ 4 ผลการทดลอง.....	8
บทที่ 5 สรุปและข้อเสนอแนะ.....	26
ภาคผนวก.....	27
ภาคผนวก ก รูปภาพที่ถูกจับจากหน้าจอระหว่างการทดลอง	
ภาคผนวก ข เอกสารอ้างอิง	

## สารบัญภาพ

	หน้า
รูปที่ 1 แสดงการเปรียบเทียบระหว่างโครงข่ายปัจจุบันกับโครงข่ายที่ถูกระบุโดยซอฟต์แวร์	9
รูปที่ 2 แสดงสถาปัตยกรรมของโครงข่ายที่ถูกระบุโดยซอฟต์แวร์	10
รูปที่ 3 แสดงหน้า download ของ software OpenVSwitch	11
รูปที่ 4 แสดงไฟล์ที่อยู่ในโฟลเดอร์ openvswitch-2.5.0	12
รูปที่ 5 แสดงผลลัพธ์ที่เกิดจากการป้อนคำสั่ง เพื่อเชื่อมต่อ OpenVSwitch database server และเพื่อทดสอบการเปิดใช้งาน OpenVSwitch	13
รูปที่ 6 แสดงหน้า download ของ mininet	14
รูปที่ 7 แสดงไฟล์ที่อยู่ในโฟลเดอร์ mininet	15
รูปที่ 8 แสดงผลลัพธ์จากการป้อนคำสั่ง mn	16
รูปที่ 9 แสดงผลลัพธ์ของคำสั่ง nodes	16
รูปที่ 10 แสดงผลลัพธ์ของคำสั่ง links	17
รูปที่ 11 แสดงผลลัพธ์ของคำสั่ง h1 ifconfig	17
รูปที่ 12 แสดง topology ของแบบจำลองโครงข่ายที่ถูกระบุโดยซอฟต์แวร์	18
รูปที่ 13 แสดงการสร้าง topology ของแบบจำลองโครงข่ายที่ถูกระบุโดยซอฟต์แวร์	19
รูปที่ 14 แสดงการ ping ระหว่าง h1 กับ h3	19
รูปที่ 15 แสดงการ ping ระหว่าง h3 กับ h1	20
รูปที่ 16 แสดงการ ping ระหว่าง h2 กับ h4	20
รูปที่ 17 แสดงการ ping ระหว่าง h4 กับ h2	21
รูปที่ 18 ทำการเติม flow table เข้าไปใน switch S1, S2 และ S4	22
รูปที่ 19 ทำการเติม flow table เข้าไปใน switch S1, S3 และ S4	22
รูปที่ 20 แสดงผลลัพธ์การ ping ระหว่าง h1 กับ h3 หลังการเติม flow table	23
รูปที่ 21 แสดงผลลัพธ์การ ping ระหว่าง h3 กับ h1 หลังการเติม flow table	24
รูปที่ 22 แสดงผลลัพธ์การ ping ระหว่าง h2 กับ h4 หลังการเติม flow table	25
รูปที่ 23 แสดงผลลัพธ์การ ping ระหว่าง h4 กับ h2 หลังการเติม flow table	25

**ชื่อการทดลอง** การศึกษาเพื่อจำลองโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ (Software-Defined Networking)

**หน่วยงาน** ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

**ปีการศึกษา** 2558

### บทคัดย่อ

โครงข่ายในปัจจุบัน ประกอบไปด้วยอุปกรณ์โครงข่ายเป็นจำนวนมาก ซึ่งอุปกรณ์โครงข่ายเหล่านี้ถูกผลิตจากผู้ผลิตหลากหลายราย ที่มีการตั้งค่าและกำหนดนโยบายให้อุปกรณ์โครงข่ายของตนที่แตกต่างกัน เมื่ออุปกรณ์โครงข่ายเหล่านี้ถูกติดตั้งเข้าไปในโครงข่ายแล้ว การพัฒนาหรือปรับปรุงเปลี่ยนแปลงอุปกรณ์โครงข่ายเหล่านี้ให้ทันสมัยยิ่งขึ้น เช่นการติดตั้งโปรโตคอลใหม่นั้น ถือได้ว่าเป็นความยากลำบากของผู้ดูแลโครงข่าย รวมถึงผู้ผลิตอุปกรณ์โครงข่าย จากปัญหาข้างต้น จึงมีแนวคิดของรูปแบบโครงข่ายใหม่เรียกว่าโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ (Software-Defined Networking (SDN)) ที่แยกส่วนควบคุมกับส่วนการส่งต่อข้อมูลออกจากกัน ซึ่งสามารถทำให้โครงข่ายมีความยืดหยุ่นและเอื้อต่อการพัฒนา ควบคุม หรือเปลี่ยนแปลงให้ดีขึ้นได้ การทดลองถูกจัดขึ้นเพื่อจำลองรูปแบบของ SDN ซึ่งผลการทดลองพบว่า สามารถสร้างแบบจำลองของ SDN ได้

## บทที่ 1

### บทนำ

#### 1.1 ความเป็นมาและความสำคัญของปัญหา

โครงข่ายที่ถูกใช้งานในปัจจุบันประกอบด้วยอุปกรณ์โครงข่ายจำนวนมาก ซึ่งแต่ละตัวก็ถูกผลิตจากผู้ผลิตที่แตกต่างกัน ดังนั้นการจะปรับปรุงหรือพัฒนาอุปกรณ์โครงข่ายเหล่านี้จึงทำได้ยาก อีกทั้งลักษณะของอุปกรณ์โครงข่ายที่มีส่วนควบคุมและส่วนส่งต่อข้อมูลอยู่ในอุปกรณ์เดียวกัน ก็ทำให้การพัฒนาหรือติดตั้งโปรโตคอลใหม่ลงไปให้อุปกรณ์ก็ทำได้ยากลำบากเช่นกัน จึงทำให้มีการคิดค้นรูปแบบใหม่ของโครงข่ายที่เรียกว่า โครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ (Software-Defined Networking)

#### 1.2 วัตถุประสงค์ของการทดลอง

เพื่อศึกษาและจำลองโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

#### 1.3 ประโยชน์ที่คาดว่าจะได้รับ

เข้าใจรูปแบบและแนวคิดของโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ และสามารถจำลองโครงสร้างอย่างง่ายได้

#### 1.4 ขอบเขตของการทดลอง

ทดลองโดยไม่ใช้ซอฟต์แวร์ควบคุมที่มีผู้ผลิตออกมาให้ใช้ แต่ให้ผู้ทดลองทำตัวเสมือนเป็นซอฟต์แวร์ควบคุม

## บทที่ 2

### งานวิจัยที่เกี่ยวข้อง

ในการศึกษาและจำลองโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์นั้น ผู้ทดลองได้ทำการศึกษาผลงานวิจัยต่างๆ เพื่อนำมาเป็นแนวทางการศึกษา ได้แก่

- 1) Astuto et al[1] ได้กล่าวถึงลักษณะของโครงข่ายในปัจจุบันที่กระตุ้นให้เกิดแนวคิดและรูปแบบใหม่ของโครงข่าย แนวคิด สถาปัตยกรรม ซอฟต์แวร์ควบคุม ภาษาในการโปรแกรม และเครื่องมือที่เกี่ยวข้อง ของโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ พวกเขายังได้นำผลสำรวจถึงความก้าวหน้าในการพัฒนาโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ที่เกิดขึ้นในวงการ ทั้งในแง่ของตัวซอฟต์แวร์เองรวมทั้งตัวชิ้นส่วนหลักตลอดจนสถาปัตยกรรมรูปแบบต่างๆที่เกิดขึ้นและอุปกรณ์ซอฟต์แวร์ตัวช่วยต่างๆที่จะนำมาอำนวยความสะดวกให้กับผู้พัฒนาโครงข่าย

## บทที่ 3

### วิธีดำเนินการทดลอง

#### 3.1 วิธีดำเนินการ

##### 3.1.1 ศึกษางานวิจัยที่เกี่ยวข้อง เพื่อให้เข้าใจถึงรูปแบบของโครงข่ายที่ถูกลิขิตโดยซอฟต์แวร์

เข้าใจถึงปัญหาของโครงข่ายในปัจจุบัน ทราบถึงแนวคิดและรูปแบบของโครงข่ายที่ถูกลิขิตโดยซอฟต์แวร์

##### 3.1.2 ศึกษาและทดลองใช้ซอฟต์แวร์เพื่อจำลองโครงข่ายที่ถูกลิขิตโดยซอฟต์แวร์

เข้าใจวิธีการติดตั้ง การใช้งาน รวมถึงการสร้างแบบจำลองของโครงข่ายที่ถูกลิขิตโดยซอฟต์แวร์ อย่างง่าย โดยใช้ซอฟต์แวร์เฉพาะ เพื่อการจำลองโครงข่ายที่ถูกลิขิตโดยซอฟต์แวร์

##### 3.1.3 ทดลองจำลองโครงข่ายที่ถูกลิขิตโดยซอฟต์แวร์

ใช้ซอฟต์แวร์ในการสร้าง topology ของโครงข่ายที่ถูกลิขิตโดยซอฟต์แวร์ และใช้โปรแกรม ping เพื่อทดสอบการสื่อสารระหว่าง virtual host ใน topology ของโครงข่ายที่ถูกลิขิตโดยซอฟต์แวร์

#### 3.2 อุปกรณ์ในการทดลอง

ในการทดลองใช้อุปกรณ์เครื่องมือต่อไปนี้

1. Operating System: Ubuntu 16.04 LTS
2. Simulator: OpenVSwitch, Mininet
3. Programming Language: Python

#### 3.3 การเก็บข้อมูล

ในการเก็บข้อมูล ใช้การจับภาพจากหน้าจอ ที่แสดงผลลัพธ์ใน terminal และ การจับภาพหน้าจอจากโปรแกรม Wireshark

## บทที่ 4

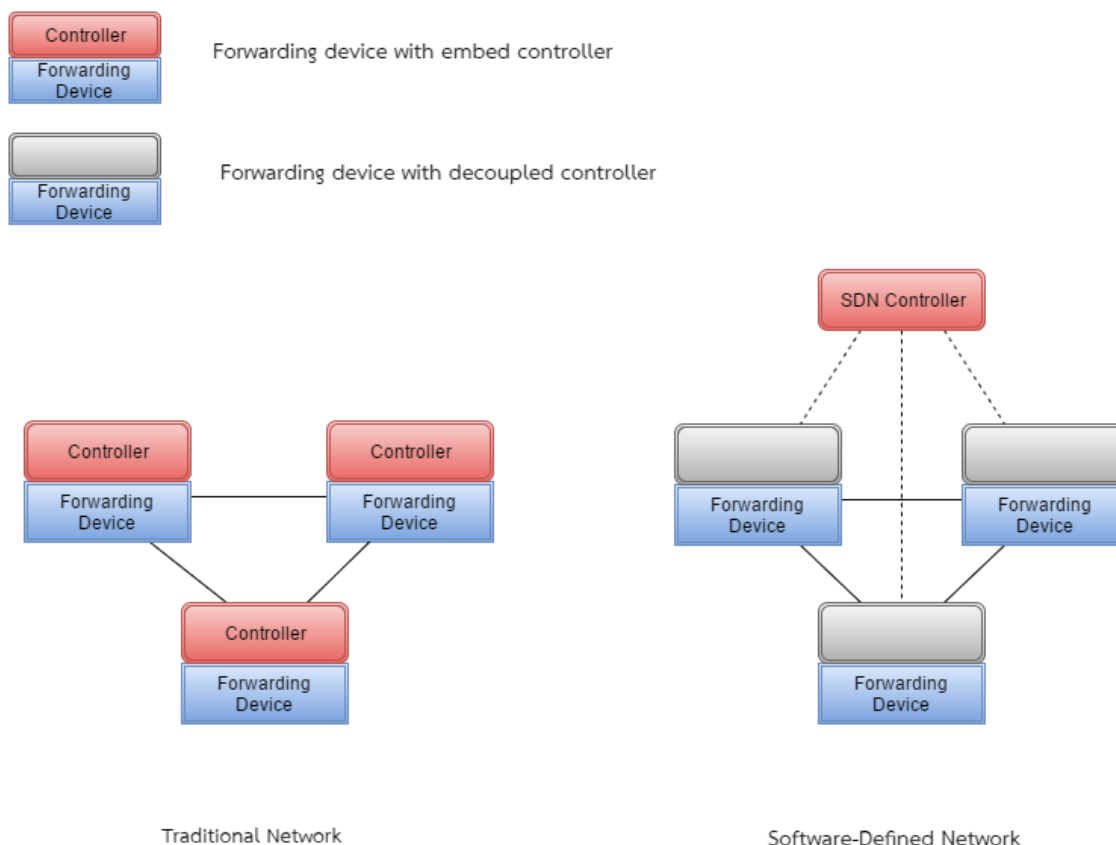
### ผลการทดลอง

#### 4.1 ศึกษางานวิจัยที่เกี่ยวข้อง เพื่อให้เข้าใจถึงโครงข่ายที่ถูกลิขิตโดยซอฟต์แวร์

โครงข่ายที่ถูกลิขิตโดยซอฟต์แวร์ (SDN) เป็นแนวคิดใหม่ของโครงข่ายที่สามารถโปรแกรมได้ โดย SDN เป็นแบบจำลองใหม่ซึ่งแยกส่วนการส่งต่อข้อมูล (data forwarding plane) ออกจากส่วนควบคุม (control decision plane) นี่ทำให้การบริหารจัดการโครงข่ายทำได้ง่ายขึ้น อีกทั้งยังทำให้โครงข่ายนั้นยืดหยุ่นและกระตุ้นให้เกิดนวัตกรรม แนวคิดก็คือ ให้ให้ผู้พัฒนาโครงข่ายนั้นสามารถพึ่งพาทรัพยากรของโครงข่ายได้ง่ายเหมือนกับที่ผู้พัฒนาซอฟต์แวร์นั้นพึ่งพาทรัพยากรของหน่วยความจำและหน่วยประมวลผลในคอมพิวเตอร์ ใน SDN นั้น ส่วนคำนวณและสั่งการ (network intelligence) จะอยู่ในส่วน software-based controller หรือส่วน control plane และอุปกรณ์โครงข่ายต่าง ๆ ก็จะกลายเป็นเพียงอุปกรณ์พื้นฐานที่ใช้ในการส่งต่อ packet ที่สามารถถูกโปรแกรมได้โดยใช้ open interface เช่น Openflow

การรับส่งข้อมูลเพื่อการสื่อสารในโครงข่ายนั้น จะทำกันผ่าน end devices หรือ host ที่เชื่อมต่อกัน ภายในโครงข่าย กลายเป็นโครงสร้างพื้นฐาน โดยจะประกอบไปด้วย host และอุปกรณ์โครงข่ายต่างๆ เช่น switch, router ที่เชื่อมต่อกันผ่าน communication link เพื่อรับส่งข้อมูล โดยปกติแล้ว switch และ router เมื่อทำการติดตั้ง เข้าไปในโครงข่ายแล้วนั้น จะเป็นการยากมากที่จะพัฒนาหรือปรับปรุงเปลี่ยนแปลงมันได้ มันจึงดูเหมือนเป็นระบบปิด ลักษณะนี้เรียกว่า network ossification หรืออีกนัยหนึ่งเรียกว่า “ความแข็งแกร่งของโครงข่าย” โดยสาเหตุหลักของปัญหานี้เกิดจากการที่ส่วนควบคุม และส่วนการส่งต่อข้อมูลนั้นอยู่ด้วยกันในแต่ละอุปกรณ์โครงข่าย ทำให้เวลาอุปกรณ์โครงข่ายเหล่านี้ต้องการจะส่งข้อมูลไปทางไหน ก็จะขึ้นอยู่กับที่ตัดสินใจในแต่ละอุปกรณ์ การติดตั้งแอปพลิเคชันของโครงข่าย หรือนโยบายใหม่ของโครงข่ายก็ต้องทำลงไปในการสร้างพื้นฐาน และยังคงใช้ความพยายามอย่างมาก จากปัญหาต่างๆที่ได้กล่าวไปในข้างต้น จึงได้มีแนวคิดของโครงข่ายที่ถูกลิขิตโดยซอฟต์แวร์ เพื่อมาอำนวยความสะดวกและทำให้การควบคุมแบบโปรแกรมได้นั้น ทำได้ง่ายขึ้น





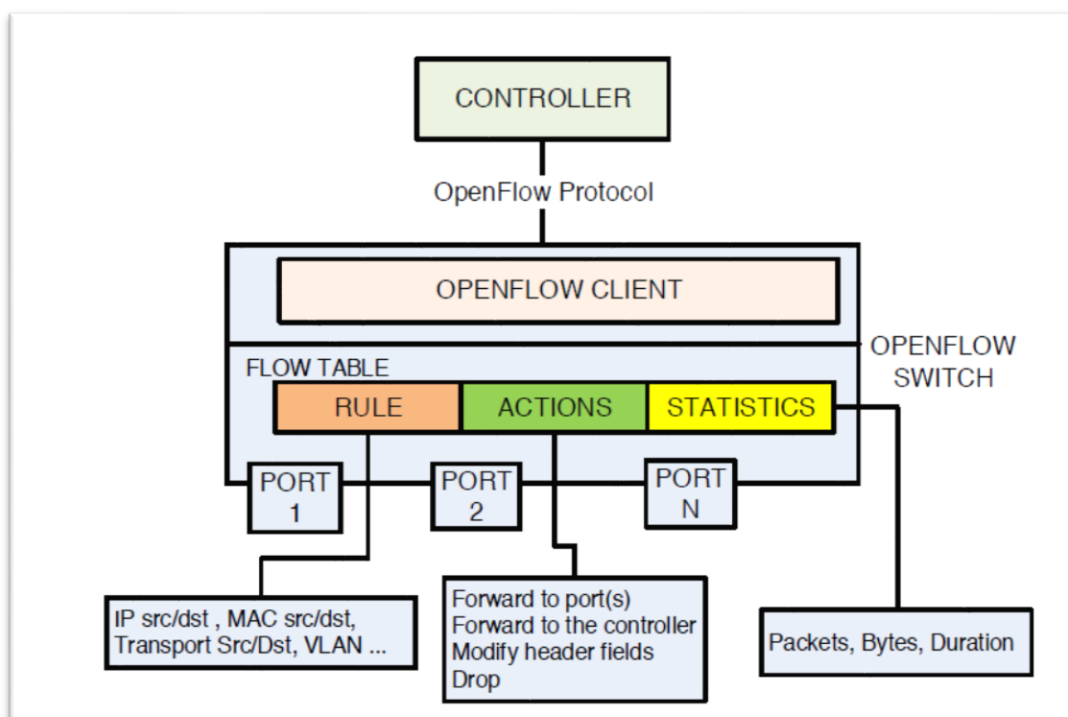
รูปที่ 1 แสดงการเปรียบเทียบระหว่างโครงข่ายปัจจุบันกับโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

### SDN architecture

ในที่นี้จะพูดถึง Openflow ซึ่งเป็นสถาปัตยกรรมที่ใช้หลักการของ SDN ที่แยกส่วน control plane กับ data forwarding plane ออกจากกัน โดยจะทำการแลกเปลี่ยนข้อมูลกันระหว่าง 2 ระดับนี้ ดังแสดงในรูปที่ 2 forwarding device หรือ Openflow switch จะประกอบไปด้วย flow table 1 ตารางหรือมากกว่า และมี abstraction layer ที่ทำหน้าที่ติดต่อกับ controller ในส่วนของ flow table นั้น ก็จะประกอบไปด้วย flow entry ที่ให้ packet นั้นสามารถรู้ได้ว่า จะต้องถูกส่งต่อไปทางไหน โดย flow entry จะประกอบไปด้วย 3 ส่วน ดังต่อไปนี้

- 1) Match field ใช้เพื่อจับคู่ข้อมูลกับ packet ที่เข้ามา โดยข้อมูลส่วนที่จะจับคู่นั้น อยู่ในส่วนของ packet header ของ packet
- 2) Counter ใช้เก็บสถิติของ flow ใดๆ เช่นจำนวนของ packet ที่เข้ามา เป็นต้น
- 3) Action ใช้เพื่อเป็นคำสั่ง ที่จะกระทำกับ packet ที่ถูกจับคู่ตรงกับ flow entry

เมื่อ packet เข้ามาใน Openflow switch ส่วน packet header จะถูกแบ่งออกมา และนำมาทำการจับคู่กับ match field ที่อยู่ใน flow entry ถ้าสามารถจับคู่ได้ switch ก็จะนำเซตของคำสั่งมาใช้กับ packet แต่ถ้าไม่สามารถจับคู่ได้ switch ก็จะนำคำสั่งที่อยู่ใน table-miss flow entry มาแทน โดยในทุกๆ flow table จะมี table-miss flow entry ไว้เพื่อจัดการกับ packet ที่ไม่สามารถจับคู่กับ match field ใดๆได้เลย



รูปที่ 2 แสดงสถาปัตยกรรมของโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

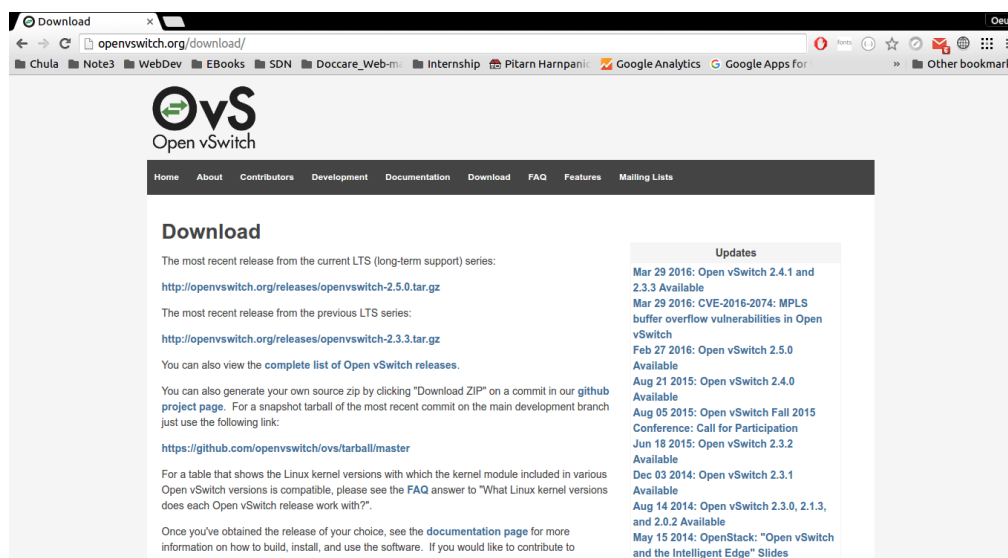
## 4.2 ศึกษาและทดลองใช้ซอฟต์แวร์เพื่อจำลองโครงข่ายที่ถูกระบุโดยซอฟต์แวร์

ในการทดลอง ผู้ทดลองได้ใช้ซอฟต์แวร์เพื่อจำลองโครงข่ายที่ถูกระบุโดยซอฟต์แวร์เป็นจำนวน 2 ซอฟต์แวร์ ได้แก่ OpenVSwitch และ Mininet

OpenVSwitch (OVS) เป็น Open-source virtual multilayer network switch ที่มีจุดประสงค์หลักเพื่อจำลองฮาร์ดแวร์ของโครงข่าย ในขณะเดียวกันก็เป็นซอฟต์แวร์ ที่รองรับโปรโตคอลและมาตรฐานทางโครงข่ายที่หลากหลาย OpenVSwitch สามารถถูกทำให้เป็น software-based network switch ได้ จึงสามารถนำมาใช้จำลองโครงข่ายที่ถูกระบุโดยซอฟต์แวร์

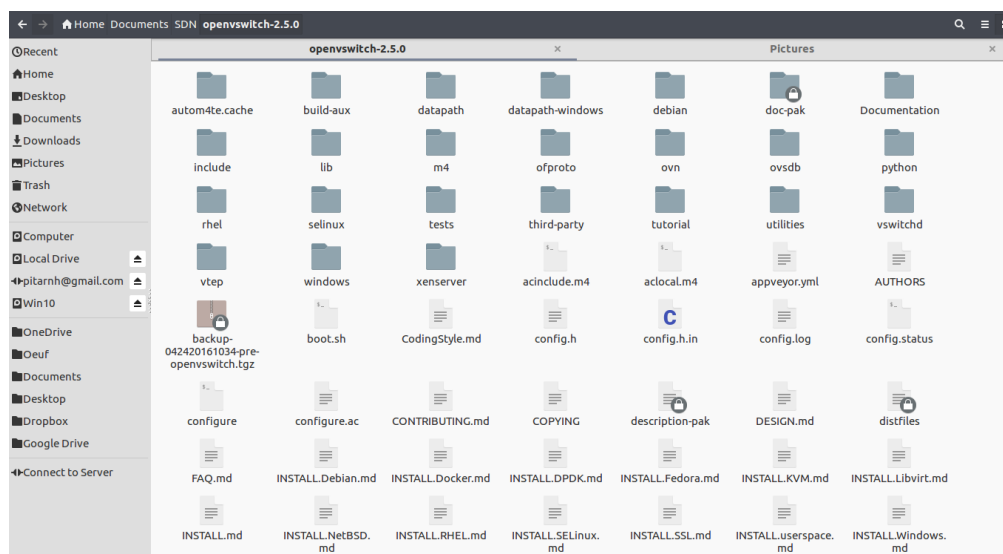
### ขั้นตอนการติดตั้ง

- 1) Download OpenVswitch จาก <http://openvswitch.org/download>



รูปที่ 3 แสดงหน้า Download ของซอฟต์แวร์ OpenVSwitch

- 2) เมื่อ download สำเร็จ ให้ทำการ unzip ไฟล์ และเข้าไปในโฟลเดอร์ openvswitch-2.5.0



รูปที่ 4 แสดงไฟล์ที่อยู่ในโฟลเดอร์ openvswitch-2.5.0

- 3) เพื่อทำการติดตั้ง ให้ป้อนคำสั่งต่อไปนี้ใน terminal ที่ถูกเปลี่ยน path มาที่โฟลเดอร์ openvswitch-2.5.0

```
$ ./boot.sh
```

```
$ ./configure
```

```
$ sudo make (terminal จะให้กรอกรหัสผู้ใช้งานของ Ubuntu)
```

```
$ sudo make install
```

- 4) เมื่อจะใช้งาน OpenVSwitch ให้ทำการเชื่อมต่อกับ OpenVSwitch database server โดยป้อนคำสั่งต่อไปนี้ใน terminal

```
$ ovsdb-server --remote=punix:/usr/local/var/run/openvswitch/db.sock \
--remote=db:Open_vSwitch,Open_vSwitch,manager_options \
--private-key=db:Open_vSwitch,SSL,private_key \
--certificate=db:Open_vSwitch,SSL,certificate \
--bootstrap-ca-cert=db:Open_vSwitch,SSL,ca_cert \
--pidfile --detach
$ ovs-vsctl --no-wait init
$ ovs-vswitchd --pidfile --detach
```

- 5) จากนั้นตรวจสอบว่า OpenVSwitch ได้ถูกเปิดใช้งานแล้วโดยการป้อน `ovs-vsctl show` ใน terminal

```

root@Latte: ~/Documents/SDN/SDN_text&sheet
oeufhp@Latte:~/Documents/SDN/SDN_text&sheet$ ./ovsdb-server.sh
modprobe: ERROR: could not insert 'openvswitch': Operation not permitted
ovsdb-server: /usr/local/var/run/openvswitch/ovsdb-server.pid.tmp: create failed
(Permission denied)
ovs-vsctl: unix:/usr/local/var/run/openvswitch/db.sock: database connection failed
(No such file or directory)
ovs-vswitchd: /usr/local/var/run/openvswitch/ovs-vswitchd.pid.tmp: create failed
(Permission denied)
oeufhp@Latte:~/Documents/SDN/SDN_text&sheet$ sudo bash
[sudo] password for oeufhp:
root@Latte:~/Documents/SDN/SDN_text&sheet# ./ovsdb-server.sh
2016-05-17T12:46:20Z|00001|ovs_numa|INFO|Discovered 4 CPU cores on NUMA node 0
2016-05-17T12:46:20Z|00002|ovs_numa|INFO|Discovered 1 NUMA nodes and 4 CPU cores
2016-05-17T12:46:20Z|00003|reconnect|INFO|unix:/usr/local/var/run/openvswitch/db
.sock: connecting...
2016-05-17T12:46:20Z|00004|reconnect|INFO|unix:/usr/local/var/run/openvswitch/db
.sock: connected
root@Latte:~/Documents/SDN/SDN_text&sheet# ovs-vsctl show
2251a049-139f-4a49-8fd6-c114968f3197
    ovs_version: "2.5.0"
root@Latte:~/Documents/SDN/SDN_text&sheet#

```

รูปที่ 5 แสดงผลลัพธ์ที่เกิดจากการป้อนคำสั่ง เพื่อเชื่อมต่อ OpenVSwitch database server และเพื่อทดสอบการเปิดใช้งาน OpenVSwitch

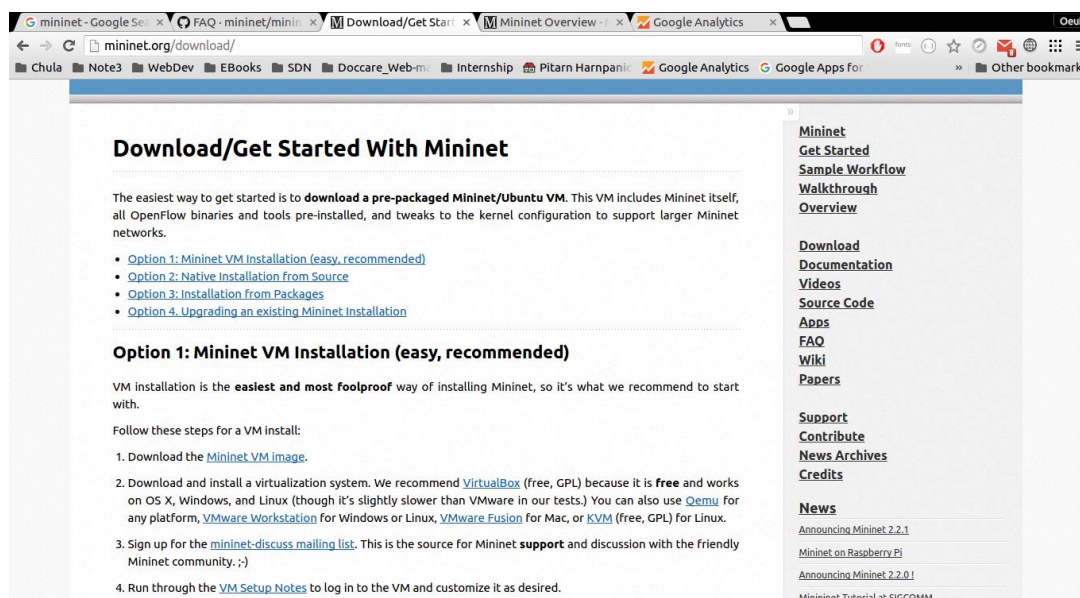
อนึ่ง ซอฟต์แวร์ OpenVSwitch ไม่สามารถทำงานเพียงลำพังได้ ต้องใช้ควบคู่ไปกับ Mininet

**Mininet** เป็นซอฟต์แวร์จำลองโครงข่ายซึ่งสามารถสร้าง virtual host, virtual switch, controller, links โดย host ของ Mininet นั้นทำงานโดยใช้ซอฟต์แวร์โครงข่ายมาตรฐานของ Linux อีกทั้ง switch ของ Mininet นั้นยังรองรับ OpenFlow protocols และสามารถนำมาจำลองโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ได้

Mininet ทำงานโดยใช้ process abstraction ในการ virtualize computing resource และใช้ process-based virtualization สำหรับการรัน virtual host ทุกๆตัวที่ถูกสร้างขึ้นมาใช้ topology ใน Linux OS kernel เพียงตัวเดียว

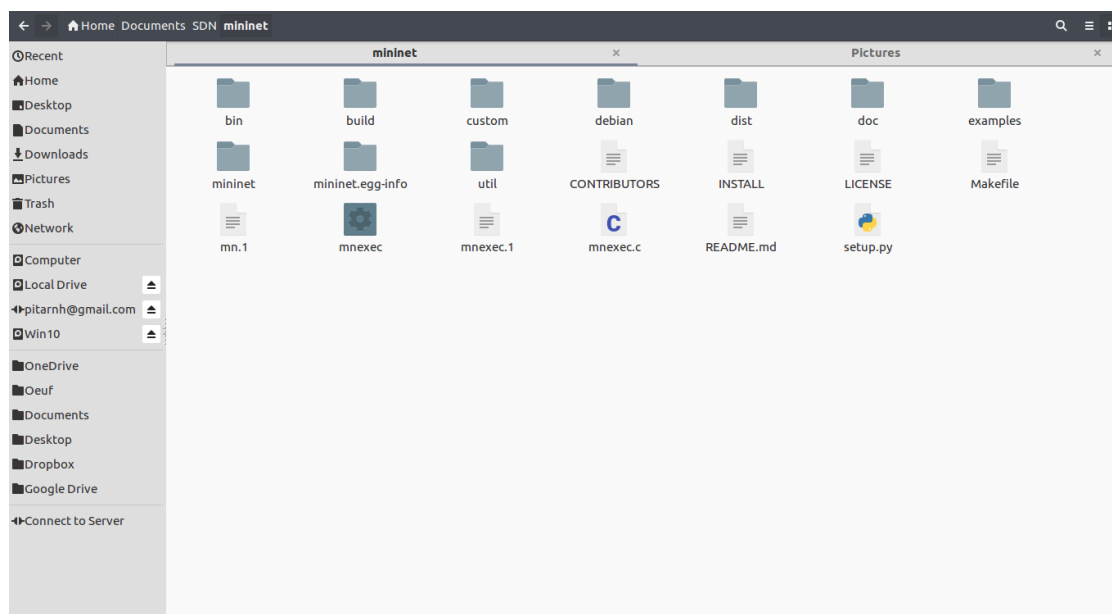
## ขั้นตอนการติดตั้ง

- 1) Download Mininet จาก <http://mininet.org/download/> โดยมีให้ download ทั้งแบบเป็น virtual machine และแบบเป็น packages โดยในที่นี้จะเลือก download แบบเป็น packages



รูปที่ 6 แสดงหน้า download ของ Mininet

- 2) เมื่อ download สำเร็จ ให้ unzip ไฟล์ และเข้าไปในโฟลเดอร์ mininet



รูปที่ 7 แสดงไฟล์ที่อยู่ในโฟลเดอร์ Mininet

- 3) ทำการติดตั้ง โดยการป้อนคำสั่งต่อไปนี้ใน terminal ที่เปลี่ยน path มาที่โฟลเดอร์ mininet และรอจนกว่าการติดตั้งจะสำเร็จ
- ```
$ sudo ./util/install.sh -a
```
- 4) ทำการทดสอบ Mininet โดยใช้คำสั่ง mn ใน terminal (ต้องสามารถเข้าถึง root user ของ Ubuntu ได้)

```

root@Latte: ~
root@Latte:~# mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>

```

รูปที่ 8 แสดงผลลัพธ์จากการป้อนคำสั่ง mn

จากรูปที่ 9 การสั่ง mn เป็นการสั่งให้ Mininet สร้าง topology อย่างง่ายที่ประกอบไปด้วย switch 1 ตัว, host 2 ตัว และ host ทั้ง 2 ตัว เชื่อมต่อกับ switch

4.1) ดูว่ามี nodes ไตบ้างใน topology ให้สั่งโดยใช้คำสั่ง

Mininet> nodes

```

root@Latte: ~ Edit View Search Terminal Help
root@Latte:~# mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet>

```

รูปที่ 9 แสดงผลลัพธ์ของคำสั่ง nodes



4.2) ดู links ที่เชื่อมต่อกันระหว่าง nodes โดยใช้คำสั่ง

mininet> links

```

root@Latte: ~
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet> link
invalid number of args: link end1 end2 [up down]
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s1-eth2 (OK OK)
mininet>

```

รูปที่ 10 แสดงผลลัพธ์ของคำสั่ง links

4.3) ดูทุกๆ interface ของ nodes ใด ใช้คำสั่งต่อไปนี้

mininet> “ชื่อ nodes” ifconfig

เช่น mininet> h1 ifconfig

```

root@Latte: ~
invalid number of args: link end1 end2 [up down]
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s1-eth2 (OK OK)
mininet> h1 ifconfig
h1-eth0  Link encap:Ethernet  HWaddr fa:10:2f:e7:5b:9b
         inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
         inet6 addr: fe80::f810:2fff:fee7:5b9b/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:33 errors:0 dropped:0 overruns:0 frame:0
         TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:4791 (4.7 KB)  TX bytes:648 (648.0 B)

lo       Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.0.0.0
         inet6 addr: ::1/128 Scope:Host
         UP LOOPBACK RUNNING  MTU:65536  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

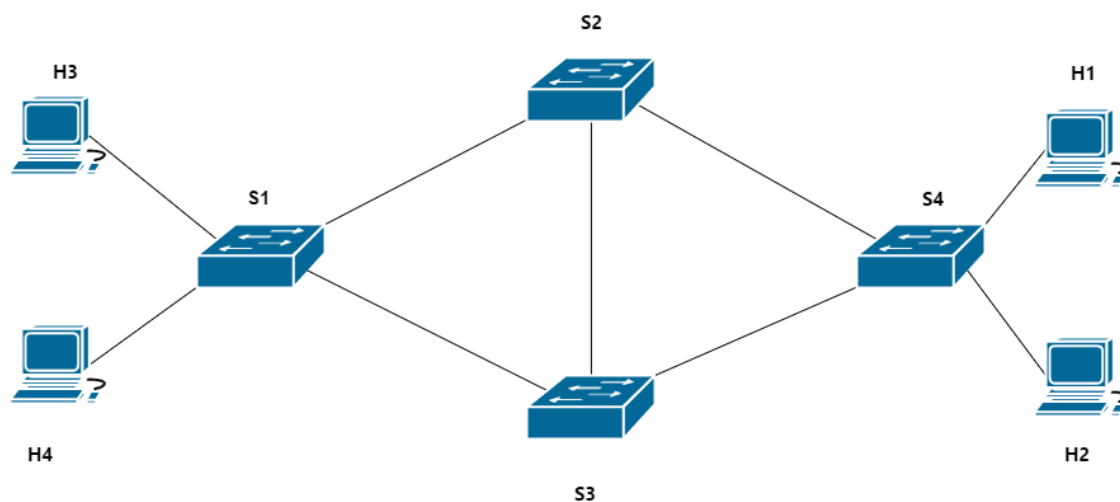
mininet>

```

รูปที่ 11 แสดงผลลัพธ์ของคำสั่ง h1 ifconnfig

### 4.3 ทดลองจำลองโครงข่ายที่ถูกระบุโดยซอฟต์แวร์

1) สร้าง topology ที่ประกอบด้วย host 4 ตัว และ switch 4 ตัว ดังรูป



รูปที่ 12 แสดง topology ของแบบจำลองโครงข่ายที่ถูกระบุโดยซอฟต์แวร์

โดยในการสร้าง topology ที่ซับซ้อน ต้องเขียนโปรแกรมด้วยภาษา Python จากนั้นสั่งด้วยคำสั่ง

```
$ sudo mn --custom=topo.py --mac --switch=ovsk --topo=myTopo
```

```
--controller=remote,ip=127.0.0.1:6634
```

```

root@Latte: ~/Documents/SDN
root@Latte:~/Documents/SDN# mn --custom=topo.py --mac --switch=ovsk --topo=myTop
o --controller=remote,127.0.0.1:6634
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6634
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s4) (h2, s4) (h3, s1) (h4, s1) (s1, s2) (s1, s3) (s2, s3) (s2, s4) (s3, s4)

*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet>

```

รูปที่ 13 แสดงการสร้าง topology ของแบบจำลองโครงข่ายที่ถูกระบุโดยซอฟต์แวร์

- ทดลองทำการสื่อสารระหว่าง host ใน topology โดยใช้โปรแกรม ping ระหว่าง h1 กับ h3 และ h2 กับ h4

```

root@Latte: ~/Documents/SDN
h3-eth0<->s1-eth1 (OK OK)
h4-eth0<->s1-eth2 (OK OK)
s1-eth3<->s2-eth1 (OK OK)
s1-eth4<->s3-eth1 (OK OK)
s2-eth2<->s3-eth2 (OK OK)
s2-eth3<->s4-eth1 (OK OK)
s3-eth3<->s4-eth2 (OK OK)
mininet> h1 ping -c10 h3
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
From 10.0.1.1 icmp_seq=1 Destination Host Unreachable
From 10.0.1.1 icmp_seq=2 Destination Host Unreachable
From 10.0.1.1 icmp_seq=3 Destination Host Unreachable
From 10.0.1.1 icmp_seq=4 Destination Host Unreachable
From 10.0.1.1 icmp_seq=5 Destination Host Unreachable
From 10.0.1.1 icmp_seq=6 Destination Host Unreachable
From 10.0.1.1 icmp_seq=7 Destination Host Unreachable
From 10.0.1.1 icmp_seq=8 Destination Host Unreachable
From 10.0.1.1 icmp_seq=9 Destination Host Unreachable
From 10.0.1.1 icmp_seq=10 Destination Host Unreachable

--- 10.0.1.2 ping statistics ---
10 packets transmitted, 0 received, +10 errors, 100% packet loss, time 9047ms
pipe 3
mininet>

```

รูปที่ 14 แสดงการ ping ระหว่าง h1 กับ h3

```

root@Latte:~/Documents/SDN Terminal Help
From 10.0.1.1 icmp_seq=8 Destination Host Unreachable
From 10.0.1.1 icmp_seq=9 Destination Host Unreachable
From 10.0.1.1 icmp_seq=10 Destination Host Unreachable

--- 10.0.1.2 ping statistics ---
10 packets transmitted, 0 received, +10 errors, 100% packet loss, time 9047ms
pipe 3
mininet> h3 ping -c10 h1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
From 10.0.1.2 icmp_seq=1 Destination Host Unreachable
From 10.0.1.2 icmp_seq=2 Destination Host Unreachable
From 10.0.1.2 icmp_seq=3 Destination Host Unreachable
From 10.0.1.2 icmp_seq=4 Destination Host Unreachable
From 10.0.1.2 icmp_seq=5 Destination Host Unreachable
From 10.0.1.2 icmp_seq=6 Destination Host Unreachable
From 10.0.1.2 icmp_seq=7 Destination Host Unreachable
From 10.0.1.2 icmp_seq=8 Destination Host Unreachable
From 10.0.1.2 icmp_seq=9 Destination Host Unreachable
From 10.0.1.2 icmp_seq=10 Destination Host Unreachable

--- 10.0.1.1 ping statistics ---
10 packets transmitted, 0 received, +10 errors, 100% packet loss, time 9047ms
pipe 3
mininet>

```

รูปที่ 15 แสดงการ ping ระหว่าง h3 กับ h1

```

root@Latte:~/Documents/SDN Terminal Help
From 10.0.1.2 icmp_seq=8 Destination Host Unreachable
From 10.0.1.2 icmp_seq=9 Destination Host Unreachable
From 10.0.1.2 icmp_seq=10 Destination Host Unreachable

--- 10.0.1.1 ping statistics ---
10 packets transmitted, 0 received, +10 errors, 100% packet loss, time 9047ms
pipe 3
mininet> h2 ping -c10 h4
PING 10.1.2.2 (10.1.2.2) 56(84) bytes of data.
From 10.1.2.1 icmp_seq=1 Destination Host Unreachable
From 10.1.2.1 icmp_seq=2 Destination Host Unreachable
From 10.1.2.1 icmp_seq=3 Destination Host Unreachable
From 10.1.2.1 icmp_seq=4 Destination Host Unreachable
From 10.1.2.1 icmp_seq=5 Destination Host Unreachable
From 10.1.2.1 icmp_seq=6 Destination Host Unreachable
From 10.1.2.1 icmp_seq=7 Destination Host Unreachable
From 10.1.2.1 icmp_seq=8 Destination Host Unreachable
From 10.1.2.1 icmp_seq=9 Destination Host Unreachable
From 10.1.2.1 icmp_seq=10 Destination Host Unreachable

--- 10.1.2.2 ping statistics ---
10 packets transmitted, 0 received, +10 errors, 100% packet loss, time 8999ms
pipe 4
mininet>

```

รูปที่ 16 แสดงการ ping ระหว่าง h2 กับ h4

```

root@latte:/Documents/SDMh Terminal Help
From 10.1.2.1 icmp_seq=8 Destination Host Unreachable
From 10.1.2.1 icmp_seq=9 Destination Host Unreachable
From 10.1.2.1 icmp_seq=10 Destination Host Unreachable

--- 10.1.2.2 ping statistics ---
10 packets transmitted, 0 received, +10 errors, 100% packet loss, time 8999ms
pipe 4
mininet> h4 ping -c10 h2
PING 10.1.2.1 (10.1.2.1) 56(84) bytes of data.
From 10.1.2.2 icmp_seq=1 Destination Host Unreachable
From 10.1.2.2 icmp_seq=2 Destination Host Unreachable
From 10.1.2.2 icmp_seq=3 Destination Host Unreachable
From 10.1.2.2 icmp_seq=4 Destination Host Unreachable
From 10.1.2.2 icmp_seq=5 Destination Host Unreachable
From 10.1.2.2 icmp_seq=6 Destination Host Unreachable
From 10.1.2.2 icmp_seq=7 Destination Host Unreachable
From 10.1.2.2 icmp_seq=8 Destination Host Unreachable
From 10.1.2.2 icmp_seq=9 Destination Host Unreachable
From 10.1.2.2 icmp_seq=10 Destination Host Unreachable

--- 10.1.2.1 ping statistics ---
10 packets transmitted, 0 received, +10 errors, 100% packet loss, time 9046ms
pipe 3
mininet>

```

รูปที่ 17 แสดงการ ping ระหว่าง h4 กับ h2

จากรูปที่ 15 – 18 จะพบว่า host ไม่สามารถติดต่อสื่อสารกันได้เลย เนื่องจากใน switch นั้น flow table ยังไม่มี flow entry ใดๆ ทำให้ packet ที่เข้ามาใน switch ไม่ได้ถูกมอบหมายชุดคำสั่งที่ถูกต้องให้ ดังนั้น จึงต้องใส่ข้อมูล flow entry เข้าไปใน flow table ของ switch แต่ละตัว

- 3) ทำการเติม flow table ของ switch ทุกๆตัว

```

root@Latte: ~/Documents/SDN
pipe 4
mininet> h4 ping -c10 h2
PING 10.1.2.1 (10.1.2.1) 56(84) bytes of data.
From 10.1.2.2 icmp_seq=1 Destination Host Unreachable
From 10.1.2.2 icmp_seq=2 Destination Host Unreachable
From 10.1.2.2 icmp_seq=3 Destination Host Unreachable
From 10.1.2.2 icmp_seq=4 Destination Host Unreachable
From 10.1.2.2 icmp_seq=5 Destination Host Unreachable
From 10.1.2.2 icmp_seq=6 Destination Host Unreachable
From 10.1.2.2 icmp_seq=7 Destination Host Unreachable
From 10.1.2.2 icmp_seq=8 Destination Host Unreachable
From 10.1.2.2 icmp_seq=9 Destination Host Unreachable
From 10.1.2.2 icmp_seq=10 Destination Host Unreachable

--- 10.1.2.1 ping statistics ---
10 packets transmitted, 0 received, +10 errors, 100% packet loss, time 9046ms
pipe 3
mininet> s4 ovs-ofctl add-flow "s4" in_port=3,actions:output=1
mininet> s2 ovs-ofctl add-flow "s2" in_port=3,actions:output=1
mininet> s1 ovs-ofctl add-flow "s1" in_port=3,actions:output=1
mininet> s1 ovs-ofctl add-flow "s1" in_port=1,actions:output=3
mininet> s2 ovs-ofctl add-flow "s2" in_port=1,actions:output=3
mininet> s4 ovs-ofctl add-flow "s4" in_port=1,actions:output=3
mininet>

```

รูปที่ 18 ทำการเติม flow table เข้าไปใน switch S1, S2 และ S4

```

root@Latte: ~/Documents/SDN
pipe 3
mininet> h4 ping -c10 h2
PING 10.1.2.1 (10.1.2.1) 56(84) bytes of data.
From 10.1.2.2 icmp_seq=1 Destination Host Unreachable
From 10.1.2.2 icmp_seq=2 Destination Host Unreachable
From 10.1.2.2 icmp_seq=3 Destination Host Unreachable
From 10.1.2.2 icmp_seq=4 Destination Host Unreachable
From 10.1.2.2 icmp_seq=5 Destination Host Unreachable
From 10.1.2.2 icmp_seq=6 Destination Host Unreachable
From 10.1.2.2 icmp_seq=7 Destination Host Unreachable
From 10.1.2.2 icmp_seq=8 Destination Host Unreachable
From 10.1.2.2 icmp_seq=9 Destination Host Unreachable
From 10.1.2.2 icmp_seq=10 Destination Host Unreachable

--- 10.1.2.1 ping statistics ---
10 packets transmitted, 0 received, +10 errors, 100% packet loss, time 9047ms
pipe 3
mininet> s4 ovs-ofctl add-flow "s4" in_port=4,actions:output=2
mininet> s3 ovs-ofctl add-flow "s3" in_port=3,actions:output=1
mininet> s1 ovs-ofctl add-flow "s1" in_port=4,actions:output=2
mininet> s1 ovs-ofctl add-flow "s1" in_port=2,actions:output=4
mininet> s3 ovs-ofctl add-flow "s3" in_port=1,actions:output=3
mininet> s4 ovs-ofctl add-flow "s4" in_port=2,actions:output=4
mininet>

```

รูปที่ 19 ทำการเติม flow table เข้าไปใน switch S1, S3 และ S4

4) ทำการ ping ระหว่าง h1 กับ h3 และ h2 กับ h4 อีกครั้ง พบว่า สามารถ ping ระหว่างกันได้

```

root@latte:/Documents/SDMh Terminal Help
mininet> s4 ovs-ofctl add-flow "s4" in_port=3,actions:output=1
mininet> s2 ovs-ofctl add-flow "s2" in_port=3,actions:output=1
mininet> s1 ovs-ofctl add-flow "s1" in_port=3,actions:output=1
mininet> s1 ovs-ofctl add-flow "s1" in_port=1,actions:output=3
mininet> s2 ovs-ofctl add-flow "s2" in_port=1,actions:output=3
mininet> s4 ovs-ofctl add-flow "s4" in_port=1,actions:output=3
mininet> xterm h1 h3
mininet> h1 ping -c10 h3
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=64 time=1.96 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=0.150 ms
64 bytes from 10.0.1.2: icmp_seq=3 ttl=64 time=0.131 ms
64 bytes from 10.0.1.2: icmp_seq=4 ttl=64 time=0.158 ms
64 bytes from 10.0.1.2: icmp_seq=5 ttl=64 time=0.161 ms
64 bytes from 10.0.1.2: icmp_seq=6 ttl=64 time=0.159 ms
64 bytes from 10.0.1.2: icmp_seq=7 ttl=64 time=0.148 ms
64 bytes from 10.0.1.2: icmp_seq=8 ttl=64 time=0.161 ms
64 bytes from 10.0.1.2: icmp_seq=9 ttl=64 time=0.160 ms
64 bytes from 10.0.1.2: icmp_seq=10 ttl=64 time=0.161 ms

--- 10.0.1.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8999ms
rtt min/avg/max/mdev = 0.131/0.334/1.960/0.542 ms
mininet>

```

รูปที่ 20 แสดงผลลัพธ์การ ping ระหว่าง h1 กับ h3 หลังการเติม flow table

```

root@Latte: ~/Documents/SDN
64 bytes from 10.0.1.2: icmp_seq=8 ttl=64 time=0.161 ms
64 bytes from 10.0.1.2: icmp_seq=9 ttl=64 time=0.160 ms
64 bytes from 10.0.1.2: icmp_seq=10 ttl=64 time=0.161 ms

--- 10.0.1.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8999ms
rtt min/avg/max/mdev = 0.131/0.334/1.960/0.542 ms
mininet> h3 ping -c10 h1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=64 time=0.992 ms
64 bytes from 10.0.1.1: icmp_seq=2 ttl=64 time=0.128 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=64 time=0.154 ms
64 bytes from 10.0.1.1: icmp_seq=4 ttl=64 time=0.180 ms
64 bytes from 10.0.1.1: icmp_seq=5 ttl=64 time=0.160 ms
64 bytes from 10.0.1.1: icmp_seq=6 ttl=64 time=0.144 ms
64 bytes from 10.0.1.1: icmp_seq=7 ttl=64 time=0.157 ms
64 bytes from 10.0.1.1: icmp_seq=8 ttl=64 time=0.157 ms
64 bytes from 10.0.1.1: icmp_seq=9 ttl=64 time=0.139 ms
64 bytes from 10.0.1.1: icmp_seq=10 ttl=64 time=0.158 ms

--- 10.0.1.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9000ms
rtt min/avg/max/mdev = 0.128/0.236/0.992/0.252 ms
mininet>

```

รูปที่ 21 แสดงผลลัพธ์ของการ ping ระหว่าง h3 กับ h1 หลังการเติม flow table

```

root@Latte: ~/Documents/SDN
pipe 3
mininet> s4 ovs-ofctl add-flow "s4" in_port=4,actions:output=2
mininet> s3 ovs-ofctl add-flow "s3" in_port=3,actions:output=1
mininet> s1 ovs-ofctl add-flow "s1" in_port=4,actions:output=2
mininet> s1 ovs-ofctl add-flow "s1" in_port=2,actions:output=4
mininet> s3 ovs-ofctl add-flow "s3" in_port=1,actions:output=3
mininet> s4 ovs-ofctl add-flow "s4" in_port=2,actions:output=4
mininet> h2 ping -c10 h4
PING 10.1.2.2 (10.1.2.2) 56(84) bytes of data.
64 bytes from 10.1.2.2: icmp_seq=1 ttl=64 time=2.09 ms
64 bytes from 10.1.2.2: icmp_seq=2 ttl=64 time=0.146 ms
64 bytes from 10.1.2.2: icmp_seq=3 ttl=64 time=0.149 ms
64 bytes from 10.1.2.2: icmp_seq=4 ttl=64 time=0.143 ms
64 bytes from 10.1.2.2: icmp_seq=5 ttl=64 time=0.160 ms
64 bytes from 10.1.2.2: icmp_seq=6 ttl=64 time=0.099 ms
64 bytes from 10.1.2.2: icmp_seq=7 ttl=64 time=0.162 ms
64 bytes from 10.1.2.2: icmp_seq=8 ttl=64 time=0.188 ms
64 bytes from 10.1.2.2: icmp_seq=9 ttl=64 time=0.134 ms
64 bytes from 10.1.2.2: icmp_seq=10 ttl=64 time=0.128 ms

--- 10.1.2.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9001ms
rtt min/avg/max/mdev = 0.099/0.340/2.096/0.585 ms
mininet>

```

รูปที่ 22 แสดงการ ping ระหว่าง h2 กับ h4 หลังเติม flow table



```

root@Latte: ~/Documents/SDN Terminal Help
64 bytes from 10.1.2.2: icmp_seq=8 ttl=64 time=0.188 ms
64 bytes from 10.1.2.2: icmp_seq=9 ttl=64 time=0.134 ms
64 bytes from 10.1.2.2: icmp_seq=10 ttl=64 time=0.128 ms

--- 10.1.2.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9001ms
rtt min/avg/max/mdev = 0.099/0.340/2.096/0.585 ms
mininet> h4 ping -c10 h2
PING 10.1.2.1 (10.1.2.1) 56(84) bytes of data.
64 bytes from 10.1.2.1: icmp_seq=1 ttl=64 time=1.00 ms
64 bytes from 10.1.2.1: icmp_seq=2 ttl=64 time=0.160 ms
64 bytes from 10.1.2.1: icmp_seq=3 ttl=64 time=0.160 ms
64 bytes from 10.1.2.1: icmp_seq=4 ttl=64 time=0.122 ms
64 bytes from 10.1.2.1: icmp_seq=5 ttl=64 time=0.143 ms
64 bytes from 10.1.2.1: icmp_seq=6 ttl=64 time=0.161 ms
64 bytes from 10.1.2.1: icmp_seq=7 ttl=64 time=0.151 ms
64 bytes from 10.1.2.1: icmp_seq=8 ttl=64 time=0.160 ms
64 bytes from 10.1.2.1: icmp_seq=9 ttl=64 time=0.154 ms
64 bytes from 10.1.2.1: icmp_seq=10 ttl=64 time=0.159 ms

--- 10.1.2.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9001ms
rtt min/avg/max/mdev = 0.122/0.237/1.004/0.256 ms
mininet>

```

รูปที่ 23 แสดงการ ping ระหว่าง h4 กับ h2 หลังเติม flow table

## บทที่ 5

### สรุปการทดลองและข้อเสนอแนะ

#### 5.1 สรุปผลการทดลอง

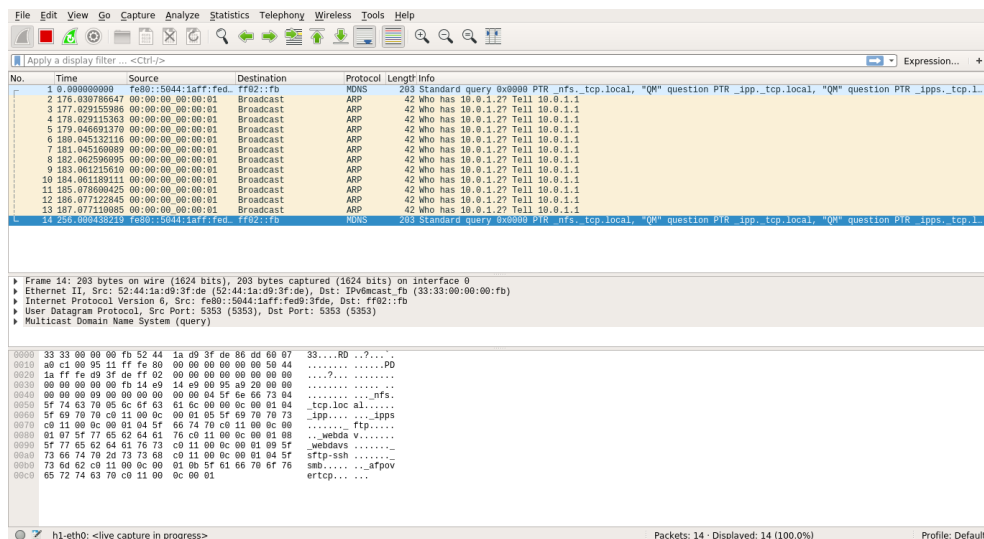
จากการทดลองสามารถสรุปได้ว่า โครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ช่วยอำนวยความสะดวกให้นักพัฒนาโครงข่าย อีกทั้งยังสามารถเอื้อให้โครงข่ายนั้นเกิดการพัฒนาหรือปรับปรุงเปลี่ยนแปลงได้ดีขึ้น

#### 5.2 ข้อเสนอแนะ

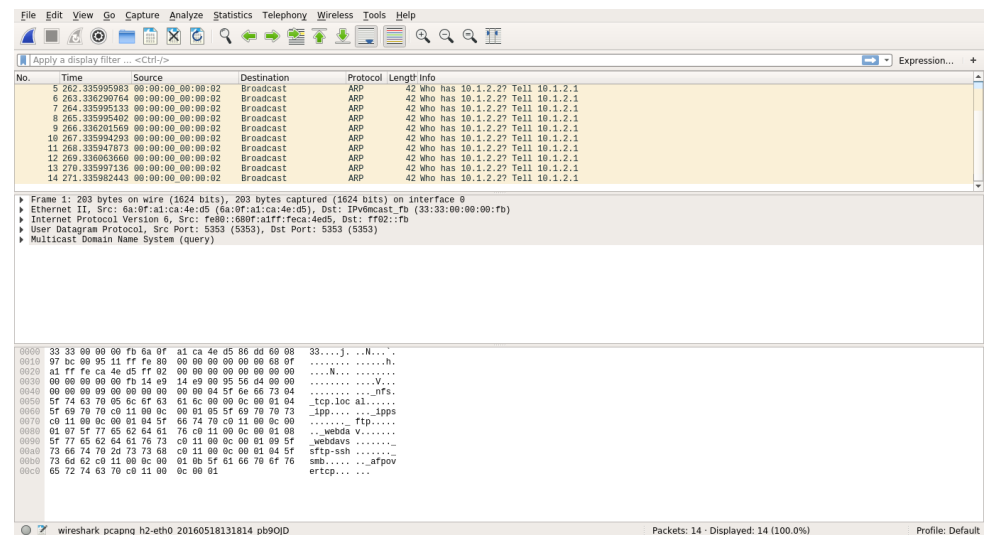
- 1) หากมีเวลา จะสร้างแบบจำลองที่มี topology ที่ซับซ้อนยิ่งขึ้น
- 2) หากมีเวลา จะศึกษาถึงข้อดีข้อเสียของโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ในหลายๆด้าน

## ภาคผนวก

### ภาคผนวก ก รูปภาพผลลัพธ์ที่ถูกบันทึกด้วยโปรแกรม Wireshark



### รูป แสดงผลลัพธ์ของ interface h1-eth0 ก่อนการเติม flow table



### รูป แสดงผลลัพธ์ของ interface h2-eth0 ก่อนการเติม flow table

| No. | Time          | Source                    | Destination | Protocol | Length | Info                                                                                                             |
|-----|---------------|---------------------------|-------------|----------|--------|------------------------------------------------------------------------------------------------------------------|
| 5   | 224.928521038 | 00:00:00:00:00:03         | Broadcast   | ARP      | 42     | Who has 10.0.1.1? Tell 10.0.1.2                                                                                  |
| 6   | 225.919144963 | 00:00:00:00:00:03         | Broadcast   | ARP      | 42     | Who has 10.0.1.1? Tell 10.0.1.2                                                                                  |
| 7   | 226.919148948 | 00:00:00:00:00:03         | Broadcast   | ARP      | 42     | Who has 10.0.1.1? Tell 10.0.1.2                                                                                  |
| 8   | 227.936597689 | 00:00:00:00:00:03         | Broadcast   | ARP      | 42     | Who has 10.0.1.1? Tell 10.0.1.2                                                                                  |
| 9   | 228.935216787 | 00:00:00:00:00:03         | Broadcast   | ARP      | 42     | Who has 10.0.1.1? Tell 10.0.1.2                                                                                  |
| 10  | 229.935213737 | 00:00:00:00:00:03         | Broadcast   | ARP      | 42     | Who has 10.0.1.1? Tell 10.0.1.2                                                                                  |
| 11  | 230.952534369 | 00:00:00:00:00:03         | Broadcast   | ARP      | 42     | Who has 10.0.1.1? Tell 10.0.1.2                                                                                  |
| 12  | 231.951152845 | 00:00:00:00:00:03         | Broadcast   | ARP      | 42     | Who has 10.0.1.1? Tell 10.0.1.2                                                                                  |
| 13  | 232.951255354 | 00:00:00:00:00:03         | Broadcast   | ARP      | 42     | Who has 10.0.1.1? Tell 10.0.1.2                                                                                  |
| 14  | 256.003119454 | fe80::18b3:10ff:fea5:e23c | ff02::fb    | MDNS     | 283    | Standard query 0x8690 PTR _nfs._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" question PTR _ipps._tcp.local |

▶ Frame 1: 283 bytes on wire (1824 bits), 283 bytes captured (1824 bits) on interface 0  
 ▶ Ethernet II, Src: la:93:10:a5:e2:3c (la:93:10:a5:e2:3c), Dst: IPv6multicast\_fb (33:33:00:00:00:fb)  
 ▶ Internet Protocol Version 6, Src: fe80::18b3:10ff:fea5:e23c, Dst: ff02::fb  
 ▶ User Datagram Protocol, Src Port: 5353 (5353), Dst Port: 5353 (5353)  
 ▶ Multicast Domain Name System (query)

```

0000 33 33 00 00 00 fb 1a b3 10 a5 e2 3c 86 dd 60 0d 33.....<...
0010 b1 3a 00 95 11 ff fe 80 00 00 00 00 00 18 b3 :.....
0020 10 ff fe a5 e2 3c ff 02 00 00 00 00 00 00 00 :.....<...
0030 00 00 00 00 00 fb 14 e9 14 e9 00 95 09 ba 00 00 :.....
0040 00 00 00 00 00 00 00 00 00 04 5f 6e 66 73 04 :....._nfs.
0050 5f 74 63 70 05 6c 6f 63 61 6c 00 00 00 01 04 :_tcp.local.....
0060 5f 69 70 70 c0 11 00 0c 00 01 95 5f 69 70 73 :_ipp....._ipps
0070 c0 11 00 0c 00 01 04 5f 66 74 70 c0 11 00 0c 00 :....._ftp.....
0080 01 07 5f 77 65 62 64 61 76 c0 11 00 0c 00 01 08 :.._webda.....
0090 5f 77 65 62 64 61 76 73 c0 11 00 0c 00 01 09 5f :.._webdav.....
00a0 73 66 74 70 2d 73 73 68 c0 11 00 0c 00 01 04 5f :_ftp-ssh.....
00b0 73 6d 62 c0 11 00 0c 01 0b 5f 61 66 70 6f 76 :_smb....._atpov
00c0 65 72 74 63 70 c0 11 00 0c 00 01 :_rtcp.....
  
```

h3-eth0: <live capture in progress>      Packets: 14 · Displayed: 14 (100.0%)      Profile: Default

รูป แสดงผลลัพธ์ของ interface h3-eth0 ก่อนการเติม flow table

| No. | Time          | Source                   | Destination | Protocol | Length | Info                                                                                                             |
|-----|---------------|--------------------------|-------------|----------|--------|------------------------------------------------------------------------------------------------------------------|
| 3   | 255.999898962 | fe80::8493:7fff:fe47::fb | ff02::fb    | MDNS     | 283    | Standard query 0x8690 PTR _nfs._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" question PTR _ipps._tcp.local |
| 4   | 263.97669571  | 00:00:00:00:00:04        | Broadcast   | ARP      | 42     | Who has 10.1.2.1? Tell 10.1.2.2                                                                                  |
| 5   | 264.976637403 | 00:00:00:00:00:04        | Broadcast   | ARP      | 42     | Who has 10.1.2.1? Tell 10.1.2.2                                                                                  |
| 6   | 305.994118635 | 00:00:00:00:00:04        | Broadcast   | ARP      | 42     | Who has 10.1.2.1? Tell 10.1.2.2                                                                                  |
| 7   | 306.992683593 | 00:00:00:00:00:04        | Broadcast   | ARP      | 42     | Who has 10.1.2.1? Tell 10.1.2.2                                                                                  |
| 8   | 307.992638175 | 00:00:00:00:00:04        | Broadcast   | ARP      | 42     | Who has 10.1.2.1? Tell 10.1.2.2                                                                                  |
| 9   | 309.010104295 | 00:00:00:00:00:04        | Broadcast   | ARP      | 42     | Who has 10.1.2.1? Tell 10.1.2.2                                                                                  |
| 10  | 310.008057107 | 00:00:00:00:00:04        | Broadcast   | ARP      | 42     | Who has 10.1.2.1? Tell 10.1.2.2                                                                                  |
| 11  | 311.008599091 | 00:00:00:00:00:04        | Broadcast   | ARP      | 42     | Who has 10.1.2.1? Tell 10.1.2.2                                                                                  |
| 12  | 312.026091676 | 00:00:00:00:00:04        | Broadcast   | ARP      | 42     | Who has 10.1.2.1? Tell 10.1.2.2                                                                                  |
| 13  | 313.024651445 | 00:00:00:00:00:04        | Broadcast   | ARP      | 42     | Who has 10.1.2.1? Tell 10.1.2.2                                                                                  |
| 14  | 314.024642947 | 00:00:00:00:00:04        | Broadcast   | ARP      | 42     | Who has 10.1.2.1? Tell 10.1.2.2                                                                                  |

▶ Frame 1: 283 bytes on wire (1824 bits), 283 bytes captured (1824 bits) on interface 0  
 ▶ Ethernet II, Src: 86:93:07:47:b1:1e (86:93:07:47:b1:1e), Dst: IPv6multicast\_fb (33:33:00:00:00:fb)  
 ▶ Internet Protocol Version 6, Src: fe80::8493:7fff:fe47:b11e, Dst: ff02::fb  
 ▶ User Datagram Protocol, Src Port: 5353 (5353), Dst Port: 5353 (5353)  
 ▶ Multicast Domain Name System (query)

```

0000 33 33 00 00 00 fb 86 93 07 47 b1 1e 86 dd 60 0c 33.....6.....
0010 0f ff 80 95 11 ff fe 80 00 00 00 00 00 04 93 :.....
0020 07 ff fe 47 b1 1e ff 02 00 00 00 00 00 00 00 :..6.....
0030 00 00 00 00 00 fb 14 e9 14 e9 00 95 3b 1e 00 00 :....._nfs.
0040 00 00 00 00 00 00 00 00 00 04 5f 6e 66 73 04 :....._tcp.local.....
0050 5f 74 63 70 05 6c 6f 63 61 6c 00 00 00 01 04 :_ipp....._ipps
0060 5f 69 70 70 c0 11 00 0c 00 01 95 5f 69 70 73 :_ftp.....
0070 c0 11 00 0c 00 01 04 5f 66 74 70 c0 11 00 0c 00 :.._webda V.....
0080 01 07 5f 77 65 62 64 61 76 c0 11 00 0c 00 01 08 :.._webdav.....
0090 5f 77 65 62 64 61 76 73 c0 11 00 0c 00 01 09 5f :_ftp-ssh.....
00a0 73 66 74 70 2d 73 73 68 c0 11 00 0c 00 01 04 5f :_smb....._atpov
00b0 73 6d 62 c0 11 00 0c 01 0b 5f 61 66 70 6f 76 :_rtcp.....
00c0 65 72 74 63 70 c0 11 00 0c 00 01 :
  
```

wireshark\_pcacng\_h4-eth0\_20160518131835\_#H9M3I      Packets: 14 · Displayed: 14 (100.0%)      Profile: Default

รูป แสดงผลลัพธ์ของ interface h4-eth0 ก่อนการเติม flow table

| No. | Time        | Source            | Destination       | Protocol | Length | Info                                                            |
|-----|-------------|-------------------|-------------------|----------|--------|-----------------------------------------------------------------|
| 2   | 0.001093277 | 00:00:00:00:00:00 | 00:00:00:00:00:00 | ARP      | 42     | 10.0.1.2 is at 00:00:00:00:00:00                                |
| 3   | 0.001107674 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=1/256, ttl=64 (reply in 4)     |
| 4   | 0.001847284 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply 10.0.1.2 seq=1/256, ttl=64 (request in 3)     |
| 5   | 0.001856466 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=2/512, ttl=64 (reply in 6)     |
| 6   | 0.001174925 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply 10.0.1.2 seq=2/512, ttl=64 (request in 5)     |
| 7   | 0.000975348 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=3/768, ttl=64 (reply in 8)     |
| 8   | 0.00156221  | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply 10.0.1.2 seq=3/768, ttl=64 (request in 7)     |
| 9   | 0.999855582 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=4/1024, ttl=64 (reply in 10)   |
| 10  | 0.999941759 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply 10.0.1.2 seq=4/1024, ttl=64 (request in 9)    |
| 11  | 0.999848049 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=5/1280, ttl=64 (reply in 12)   |
| 12  | 0.999943478 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply 10.0.1.2 seq=5/1280, ttl=64 (request in 11)   |
| 13  | 0.999857596 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=6/1536, ttl=64 (reply in 14)   |
| 14  | 0.99958223  | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply 10.0.1.2 seq=6/1536, ttl=64 (request in 13)   |
| 15  | 0.007816028 | 00:00:00:00:00:00 | 00:00:00:00:00:00 | ARP      | 42     | Who has 10.0.1.1? Tell 10.0.1.2                                 |
| 16  | 0.007834449 | 00:00:00:00:00:00 | 00:00:00:00:00:00 | ARP      | 42     | 10.0.1.1 is at 00:00:00:00:00:00                                |
| 17  | 0.999835876 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=7/1792, ttl=64 (reply in 18)   |
| 18  | 0.999923937 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply 10.0.1.2 seq=7/1792, ttl=64 (request in 17)   |
| 19  | 0.999854422 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=8/2048, ttl=64 (reply in 20)   |
| 20  | 0.999948858 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply 10.0.1.2 seq=8/2048, ttl=64 (request in 19)   |
| 21  | 0.999847472 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=9/2304, ttl=64 (reply in 22)   |
| 22  | 0.999941247 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply 10.0.1.2 seq=9/2304, ttl=64 (request in 21)   |
| 23  | 0.999855553 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=10/2560, ttl=64 (reply in 24)  |
| 24  | 0.999951026 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply 10.0.1.2 seq=10/2560, ttl=64 (request in 23)  |
| 25  | 0.999853335 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=1/256, ttl=64 (reply in 26)    |
| 26  | 0.999897812 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=1/256, ttl=64 (request in 25)  |
| 27  | 0.999826488 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=2/512, ttl=64 (reply in 28)    |
| 28  | 0.999857796 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=2/512, ttl=64 (request in 27)  |
| 29  | 0.999862383 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=3/768, ttl=64 (reply in 30)    |
| 30  | 0.999898771 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=3/768, ttl=64 (request in 29)  |
| 31  | 0.999844797 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=4/1024, ttl=64 (reply in 32)   |
| 32  | 0.999891729 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=4/1024, ttl=64 (request in 31) |
| 33  | 0.999896261 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=5/1280, ttl=64 (reply in 34)   |
| 34  | 0.999834759 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=5/1280, ttl=64 (request in 33) |
| 35  | 0.999853898 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=6/1536, ttl=64 (reply in 36)   |
| 36  | 0.999889795 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=6/1536, ttl=64 (request in 35) |

h1-eth0: <live capture in progress>

Packets: 44 · Displayed: 44 (100.0%)

Profile: Default

รูป แสดงผลลัพธ์ของ interface h1-eth0 หลังการเติม flow table

| No. | Time          | Source            | Destination       | Protocol | Length | Info                                                             |
|-----|---------------|-------------------|-------------------|----------|--------|------------------------------------------------------------------|
| 15  | 213.263161719 | 00:00:00:00:00:00 | 00:00:00:00:00:00 | ARP      | 42     | 10.1.2.2 is at 00:00:00:00:00:00                                 |
| 16  | 213.263175323 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=1/256, ttl=64 (reply in 17)     |
| 17  | 213.264082271 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) reply 10.0.1.2 seq=1/256, ttl=64 (request in 16)     |
| 18  | 214.263483964 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=2/512, ttl=64 (reply in 19)     |
| 19  | 214.263571599 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) reply 10.0.1.2 seq=2/512, ttl=64 (request in 18)     |
| 20  | 215.263707770 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=3/768, ttl=64 (reply in 21)     |
| 21  | 215.263877707 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=3/768, ttl=64 (request in 20)   |
| 22  | 216.263880540 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=4/1024, ttl=64 (reply in 23)    |
| 23  | 216.263880645 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=4/1024, ttl=64 (request in 22)  |
| 24  | 217.263798361 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=5/1280, ttl=64 (reply in 25)    |
| 25  | 217.263895862 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=5/1280, ttl=64 (request in 24)  |
| 26  | 218.263880450 | 00:00:00:00:00:00 | 00:00:00:00:00:00 | ARP      | 42     | Who has 10.1.2.1? Tell 10.1.2.2                                  |
| 27  | 218.263826580 | 00:00:00:00:00:00 | 00:00:00:00:00:00 | ARP      | 42     | 10.1.2.1 is at 00:00:00:00:00:00                                 |
| 28  | 218.263915146 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=6/1536, ttl=64 (reply in 29)    |
| 29  | 218.263955580 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=6/1536, ttl=64 (request in 28)  |
| 30  | 219.263846999 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=7/1792, ttl=64 (reply in 31)    |
| 31  | 219.263935931 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=7/1792, ttl=64 (request in 30)  |
| 32  | 220.263866859 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=8/2048, ttl=64 (reply in 33)    |
| 33  | 220.263981213 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=8/2048, ttl=64 (request in 32)  |
| 34  | 221.263890630 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=9/2304, ttl=64 (reply in 35)    |
| 35  | 221.263887235 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=9/2304, ttl=64 (request in 34)  |
| 36  | 222.263793230 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=10/2560, ttl=64 (reply in 37)   |
| 37  | 222.263873609 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=10/2560, ttl=64 (request in 36) |
| 38  | 247.478822710 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=1/256, ttl=64 (reply in 39)     |
| 39  | 247.478890753 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=1/256, ttl=64 (request in 38)   |
| 40  | 248.479031968 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) request 10.0.1.2 seq=2/512, ttl=64 (reply in 41)     |

h2-eth0: <live capture in progress>

Packets: 59 · Displayed: 59 (100.0%)

Profile: Default

รูป แสดงผลลัพธ์ของ interface h2-eth0 หลังการเติม flow table

| No. | Time         | Source            | Destination       | Protocol | Length | Info                                                              |
|-----|--------------|-------------------|-------------------|----------|--------|-------------------------------------------------------------------|
| 1   | 0.000000000  | 00:00:00:00:00:00 | 00:00:00:00:00:00 | ARP      | 42     | Who has 10.0.1.2? [all] 10.0.1.1                                  |
| 2   | 0.00042550   | 00:00:00:00:00:00 | 00:00:00:00:00:00 | ARP      | 42     | 10.0.1.2 is at 00:00:00:00:00:00                                  |
| 3   | 0.000742310  | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=1/256, ttl=64 (reply in 4)     |
| 4   | 0.000808394  | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=1/256, ttl=64 (request in 3)     |
| 5   | 1.000402431  | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=2/512, ttl=64 (reply in 6)     |
| 6   | 1.000442117  | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=2/512, ttl=64 (request in 5)     |
| 7   | 1.999389127  | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=3/768, ttl=64 (reply in 8)     |
| 8   | 1.999424403  | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=3/768, ttl=64 (request in 7)     |
| 9   | 2.999171210  | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=4/1024, ttl=64 (reply in 10)   |
| 10  | 2.999208966  | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=4/1024, ttl=64 (request in 9)    |
| 11  | 3.999167310  | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=5/1280, ttl=64 (reply in 12)   |
| 12  | 3.999208941  | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=5/1280, ttl=64 (request in 11)   |
| 13  | 4.999176621  | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=6/1536, ttl=64 (reply in 14)   |
| 14  | 4.999216031  | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=6/1536, ttl=64 (request in 13)   |
| 15  | 5.007821528  | 00:00:00:00:00:00 | 00:00:00:00:00:00 | ARP      | 42     | Who has 10.0.1.1? Tell 10.0.1.2                                   |
| 16  | 5.007124127  | 00:00:00:00:00:00 | 00:00:00:00:00:00 | ARP      | 42     | 10.0.1.1 is at 00:00:00:00:00:00                                  |
| 17  | 5.999153045  | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=7/1792, ttl=64 (reply in 18)   |
| 18  | 5.999189751  | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=7/1792, ttl=64 (request in 17)   |
| 19  | 6.999173337  | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=8/2048, ttl=64 (reply in 20)   |
| 20  | 6.999214481  | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=8/2048, ttl=64 (request in 19)   |
| 21  | 7.999160282  | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=9/2304, ttl=64 (reply in 22)   |
| 22  | 7.999200745  | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=9/2304, ttl=64 (request in 21)   |
| 23  | 8.999175525  | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=10/2560, ttl=64 (reply in 24)  |
| 24  | 8.999215296  | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=10/2560, ttl=64 (request in 23)  |
| 25  | 34.634674704 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=1/256, ttl=64 (reply in 26)    |
| 26  | 34.635596006 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=1/256, ttl=64 (request in 25)  |
| 27  | 35.635874758 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=2/512, ttl=64 (reply in 28)      |
| 28  | 35.635940635 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=2/512, ttl=64 (request in 27)  |
| 29  | 36.635103550 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=3/768, ttl=64 (reply in 30)    |
| 30  | 36.635130609 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=3/768, ttl=64 (request in 29)    |
| 31  | 37.635080787 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=4/1024, ttl=64 (reply in 32)   |
| 32  | 37.635160278 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=4/1024, ttl=64 (request in 31)   |
| 33  | 38.635130511 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=5/1280, ttl=64 (reply in 34)   |
| 34  | 38.635279680 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=5/1280, ttl=64 (request in 33) |
| 35  | 39.635007030 | 10.0.1.2          | 10.0.1.1          | ICMP     | 98     | Echo (ping) request id=0x1664, seq=6/1536, ttl=64 (reply in 36)   |
| 36  | 39.635122899 | 10.0.1.1          | 10.0.1.2          | ICMP     | 98     | Echo (ping) reply id=0x1664, seq=6/1536, ttl=64 (request in 35)   |

รูป แสดงผลลัพธ์ของ interface h3-eth0 หลังการเติม flow table

| No. | Time          | Source            | Destination       | Protocol | Length | Info                                                             |
|-----|---------------|-------------------|-------------------|----------|--------|------------------------------------------------------------------|
| 14  | 106.172465589 | 00:00:00:00:00:00 | 00:00:00:00:00:00 | ARP      | 42     | 10.1.2.2 is at 00:00:00:00:00:00                                 |
| 15  | 106.173256784 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request id=0x1868, seq=1/256, ttl=64 (reply in 16)   |
| 16  | 106.173329847 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) reply id=0x1868, seq=1/256, ttl=64 (request in 15)   |
| 17  | 107.173247871 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request id=0x1868, seq=2/512, ttl=64 (reply in 18)   |
| 18  | 107.173258061 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) reply id=0x1868, seq=2/512, ttl=64 (request in 17)   |
| 19  | 108.173555078 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request id=0x1868, seq=3/768, ttl=64 (reply in 20)   |
| 20  | 108.173592714 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) reply id=0x1868, seq=3/768, ttl=64 (request in 19)   |
| 21  | 109.173566557 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request id=0x1868, seq=4/1024, ttl=64 (reply in 22)  |
| 22  | 109.173603631 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) reply id=0x1868, seq=4/1024, ttl=64 (request in 21)  |
| 23  | 170.173581061 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request id=0x1868, seq=5/1280, ttl=64 (reply in 24)  |
| 24  | 170.173607856 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) reply id=0x1868, seq=5/1280, ttl=64 (request in 23)  |
| 25  | 171.173439124 | 00:00:00:00:00:00 | 00:00:00:00:00:00 | ARP      | 42     | Who has 10.1.2.2? Tell 10.1.2.1                                  |
| 26  | 171.173546440 | 00:00:00:00:00:00 | 00:00:00:00:00:00 | ARP      | 42     | 10.1.2.1 is at 00:00:00:00:00:00                                 |
| 27  | 171.173653266 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request id=0x1868, seq=6/1536, ttl=64 (reply in 28)  |
| 28  | 171.173683989 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) reply id=0x1868, seq=6/1536, ttl=64 (request in 27)  |
| 29  | 172.173611624 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request id=0x1868, seq=7/1792, ttl=64 (reply in 30)  |
| 30  | 172.173649114 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) reply id=0x1868, seq=7/1792, ttl=64 (request in 29)  |
| 31  | 173.173632150 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request id=0x1868, seq=8/2048, ttl=64 (reply in 32)  |
| 32  | 173.173694165 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) reply id=0x1868, seq=8/2048, ttl=64 (request in 31)  |
| 33  | 174.173568978 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request id=0x1868, seq=9/2304, ttl=64 (reply in 34)  |
| 34  | 174.173602548 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) reply id=0x1868, seq=9/2304, ttl=64 (request in 33)  |
| 35  | 175.173559253 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) request id=0x1868, seq=10/2560, ttl=64 (reply in 36) |
| 36  | 175.173590789 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) reply id=0x1868, seq=10/2560, ttl=64 (request in 35) |
| 37  | 200.388921751 | 10.1.2.2          | 10.1.2.1          | ICMP     | 98     | Echo (ping) request id=0x1868, seq=1/256, ttl=64 (reply in 38)   |
| 38  | 200.388955621 | 10.1.2.1          | 10.1.2.2          | ICMP     | 98     | Echo (ping) reply id=0x1868, seq=1/256, ttl=64 (request in 37)   |

รูป แสดงผลลัพธ์ของ interface h4-eth0 หลังการเติม flow table

**ภาคผนวก ข เอกสารอ้างอิง**

[1] Bruno Nunes Astuto, Marc Mendonca, Xuan Nam Nguyen, Katia Obraczka, Thierry Turletti. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. Communications Surveys and Tutorials, IEEE Communications Society, Institute of Electrical and Electronics Engineers, 2014, 16 (3),pp.1617 - 1634.