รายงานการทดลอง

การศึกษาและทดลองใช้ซอฟต์แวร์เพื่อจำลองโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ (Software-Defined Networking)

เสนอ

รศ.ดร.กุลธิดา โรจน์วิบูลย์ชัย

คณะผู้ทดลอง

นายพิธาน หาญพาณิชย์พันธ์ 5631068821

รายงานนี้เป็นส่วนหนึ่งของวิชา

เอกัตศึกษาทางวิศวกรรมคอมพิวเตอร์ 4

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ภาคการศึกษาปลาย ปีการศึกษา 2558

สารบัญ

หน้า

สารบัญภาพ

หน้า

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 ก็ทำให้การพัฒนาหรือติดตั้งโปรโตคอลใหม่ลงไปในอุปกรณ์ก็ทำได้ยากลำบากเช่นกัน 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 เพื่อให้เข้าใจถึงรูปแบบของโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

เข้าใจถึงปัญหาของโครงข่ายในปัจจุบัน ทราบถึงแนวคิดและรูปแบบของโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

3.1.2 ศึกษาและทดลองใช้ซอฟต์แวร์เพื่อจำลองโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

เข้าใจวิธีการติดตั้ง การใช้งาน รวมถึงการสร้างแบบจำลองของโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์อย่างง่าย โดยใช้ซอฟต์แวร์เฉพาะ เพื่อการจำลองโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

3.1.3 ทดลองจำลองโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

ใช้ซอฟต์แวร์ในการสร้าง topology ของโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ และใช้โปรแกรม ping เพื่อทดสอบการสื่อสารระหว่าง virtual host ใน topology ของโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

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

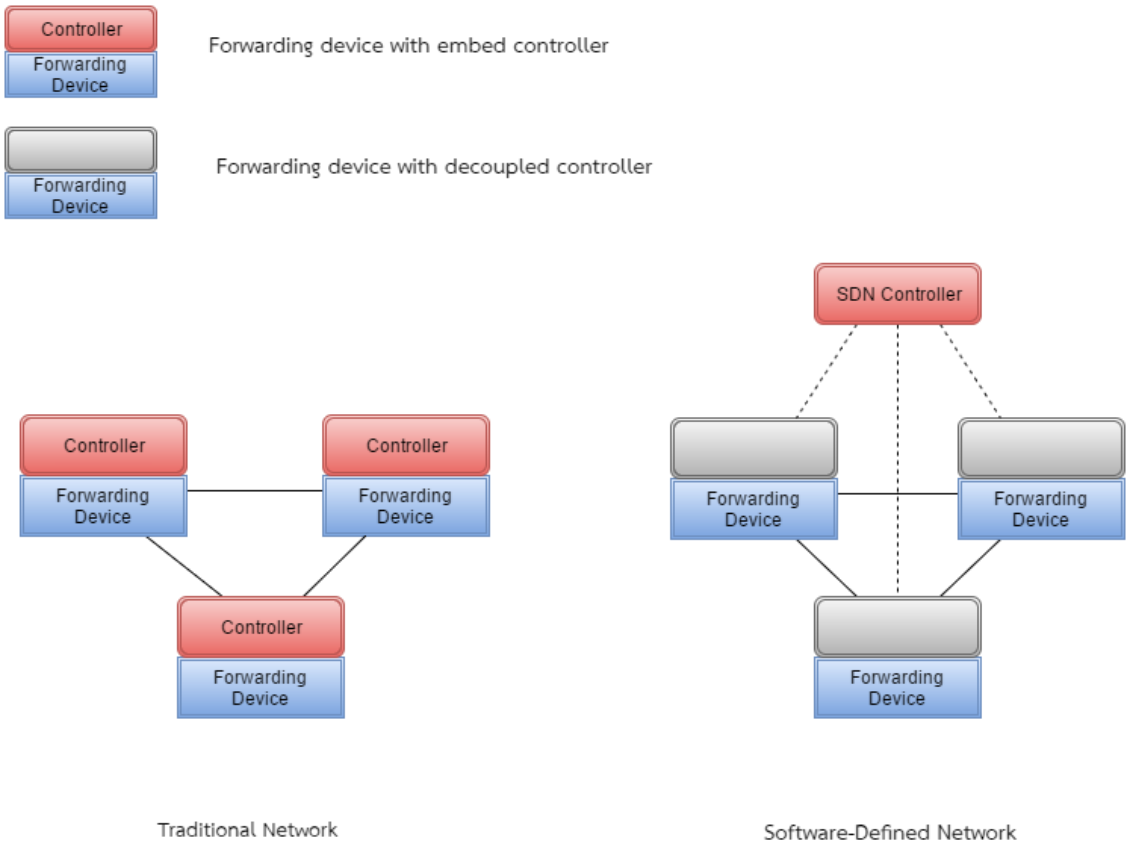
บทที่ 4

ผลการทดลอง

4.1 ศึกษางานวิจัยที่เกี่ยวข้อง เพื่อให้เข้าใจถึงโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

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

การรับส่งข้อมูลเพื่อการสื่อสารในโครงข่ายนั้น จะทำกันผ่าน end devices หรือ host ที่เชื่อมต่อกันภายในโครงข่าย กลายเป็นโครงสร้างพื้นฐาน โดยจะประกอบไปด้วย host และอุปกรณ์โครงข่ายต่างๆ เช่น switch, router ที่เชื่อมต่อกันผ่าน communication link เพื่อรับส่งข้อมูล โดยปกติแล้ว switch และ router เมื่อทำการติดตั้ง เข้าไปในโครงข่ายแล้วนั้น จะเป็นการยากมากที่จะพัฒนาหรือปรับปรุงเปลี่ยนแปลงมันได้ มันจึงดูเหมือนเป็นระบบปิด ลักษณะนี้เรียกว่า network ossification หรืออีกนัยหนึ่งเรียกว่า “ความแข็งเกร็งของโครงข่าย” โดยสาเหตุหลักของปัญหานี้เกิดจากการที่ส่วนควบคุม และส่วนการส่งต่อข้อมูลนั้นอยู่ด้วยกันในแต่ละอุปกรณ์โครงข่าย ทำให้เวลาอุปกรณ์โครงข่ายเหล่านี้ต้องการจะส่งข้อมูลไปทางไหน ก็จะขึ้นอยู่กับการตัดสินใจในแต่ละอุปกรณ์ การติดตั้งแอปพลิเคชันของโครงข่าย หรือนโยบายใหม่ของโครงข่ายก็จะต้องทำลงไปในโครงสร้างพื้นฐาน และยังต้องใช้ความพยายามอย่างมาก จากปัญหาต่างๆที่ได้กล่าวไปในข้างต้น จึงได้มีแนวคิดของโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ เพื่อมาอำนวยความสะดวกและทำให้การควบคุมแบบโปรแกรมได้นั้น ทำได้ง่ายขึ้น



รูปที่ 1 แสดงการเปรียบเทียบระหว่างโครงข่ายปัจจุบันกับโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

SDN architecture

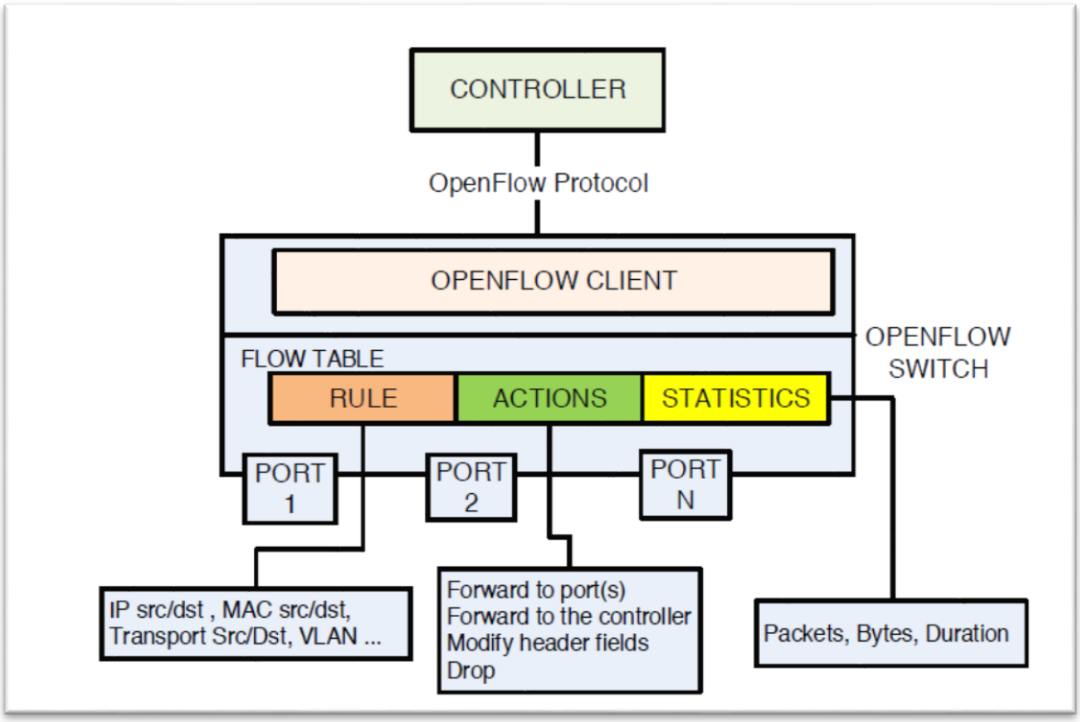
ในที่นี้จะพูดถึง Openflow ซึ่งเป็นสถาปัตยกรรมที่ใช้หลักการของ SDN ที่แยกส่วน control plane กับ data forwarding plane ออกจากกัน โดยจะทำการแลกเปลี่ยนข้อมูลกันระหว่าง 2 ระดับนี้ ดังแสดงในรูปที่ 2 forwarding device หรือ Openflow switch จะประกอบไปด้วย flow table 1 ตารางหรือมากกว่า และมี abstraction layer 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 packetที่เข้ามา เป็นต้น

3) Action ใช้เพื่อเป็นคาสั่ง ที่จะกระทำกับ packet ที่ถูกจับคู่ตรงกับ flow entry

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



รูปที่ 2 แสดงสถาปัตยกรรมของโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

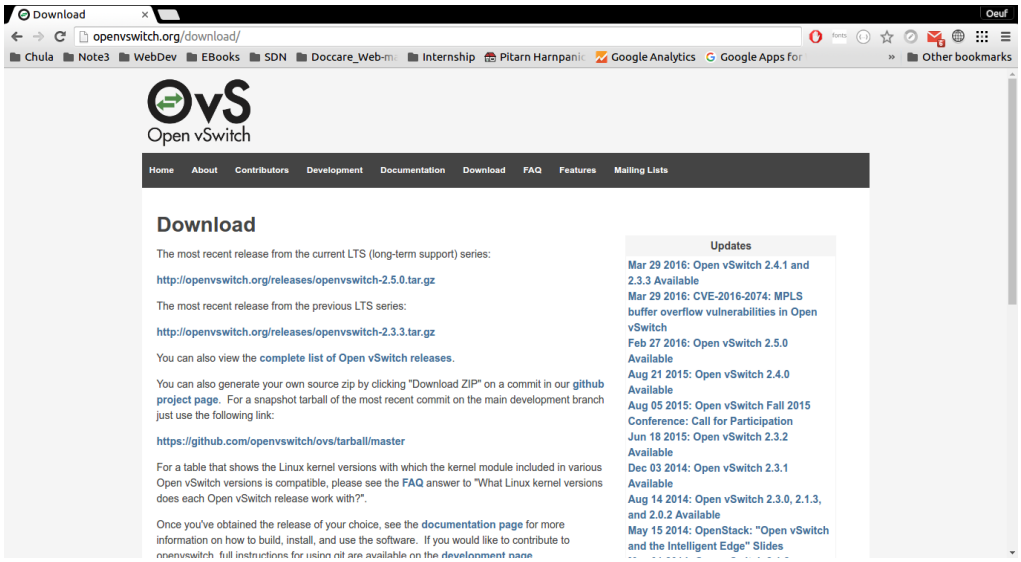
4.2 ศึกษาและทดลองใช้ซอฟต์แวร์เพื่อจำลองโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

ในการทดลอง ผู้ทดลองได้ใช้ซอฟต์แวร์เพื่อจำลองโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์เป็นจำนวน 2 ซอฟต์แวร์ ได้แก่ OpenVSwitch และ Mininet

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

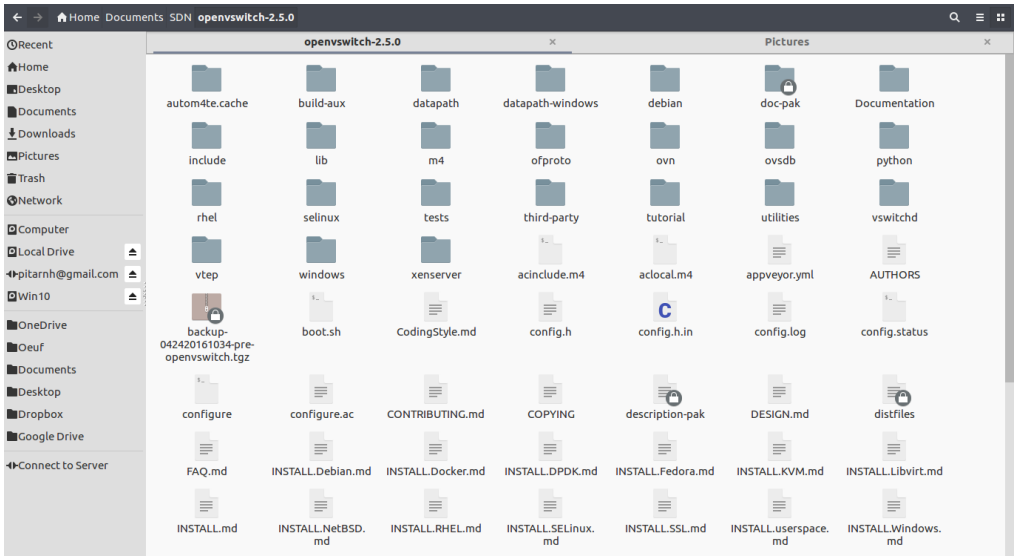
ขั้นตอนการติดตั้ง

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

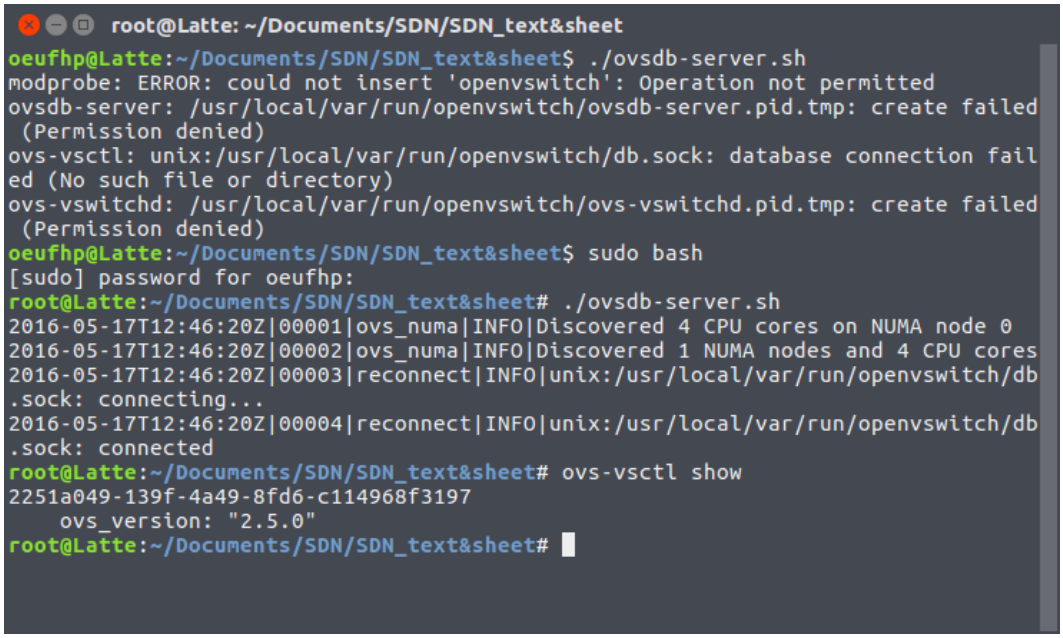
3) เพื่อทำการติดตั้ง ให้ป้อนคำสั่งต่อไปนี้ใน terminal ที่ถูกเปลี่ยน path มาที่โฟลเดอร์ openvswitch-2.5.0

```
$ ./boot.sh
$ ./configure
$ sudo make (terminal จะให้กรอกรหัสผู้ใช้ของ 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



รูปที่ 5 แสดงผลลัพธ์ที่เกิดจากการป้อนคำสั่ง เพื่อเชื่อมต่อ OpenVSwitch database server และเพื่อทดสอบการเปิดใช้งาน OpenVSwitch

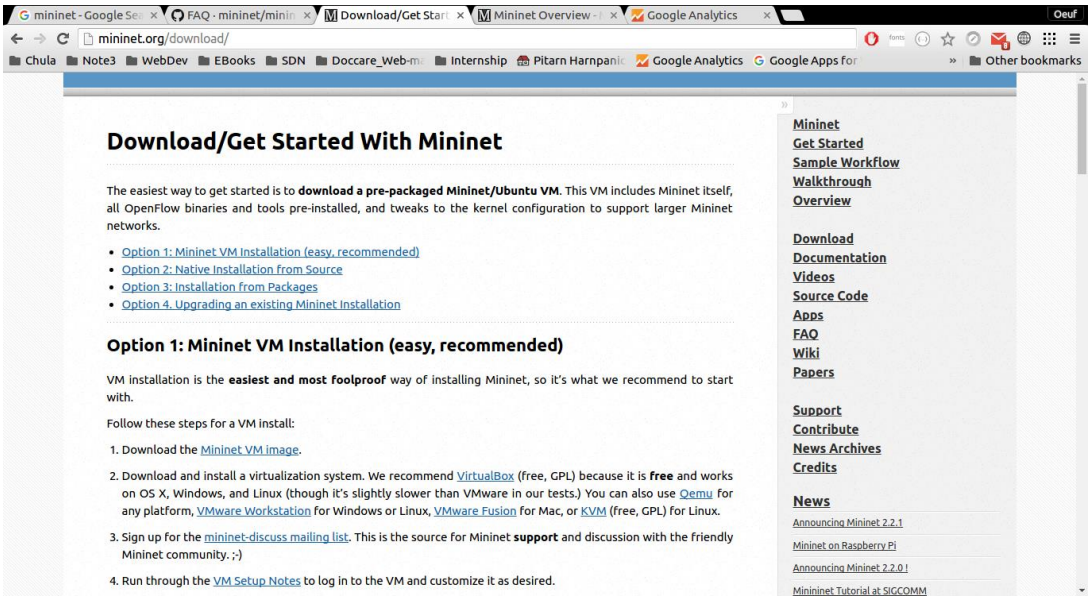
อนึ่ง ซอฟต์แวร์OpenVSwitch ไม่สามารถทำงานเพียงลำพังได้ ต้องใช้ควบคู่ไปกับ Mininet

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

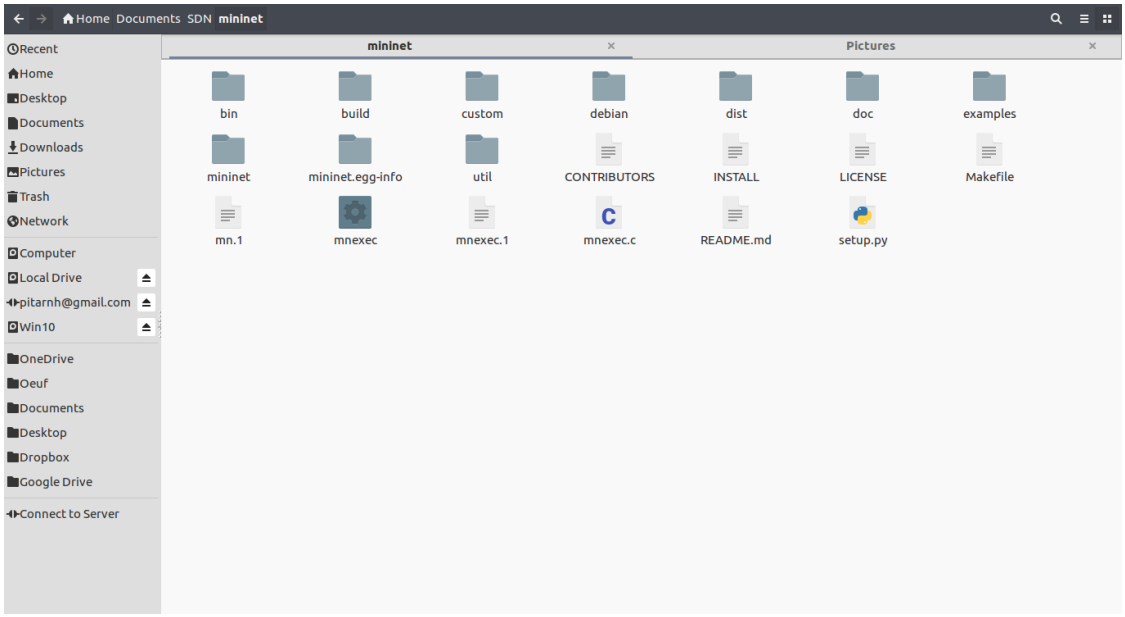
ขั้นตอนการติดตั้ง

1) Download Mininet จาก http://mininet.org/download/ โดยมีให้ download ทั้งแบบเป็น virtual machine และแบบเป็นpackages โดยในที่นี้จะเลือก download แบบเป็น packages



รูปที่ 6 แสดงหน้า download ของ Mininet

2) เมื่อ download สำเร็จ ให้ unzip ไฟล์ และเข้าไปในโฟลเดอร์ mininet

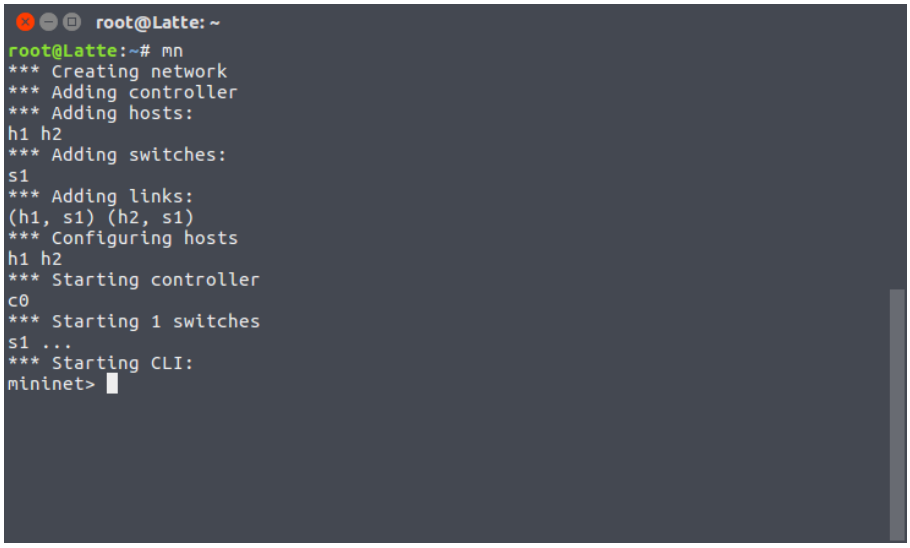


รูปที่ 7 แสดงไฟล์ที่อยู่ในโฟลเดอร์ Mininet

3) ทำการติดตั้ง โดยการป้อนคำสั่งต่อไปนี้ใน terminal ที่ปลี่ยน path มาที่โฟลเดอร์ mininet และรอจนกว่าการติดตั้งจะสำเร็จ

```
$ sudo ./util/install.sh -a
```

4) ทำการทดสอบ Mininet โดยใช้คำสั่ง mn ใน terminal (ต้องสามารถเข้าถึง root user ของ Ubuntu ได้)

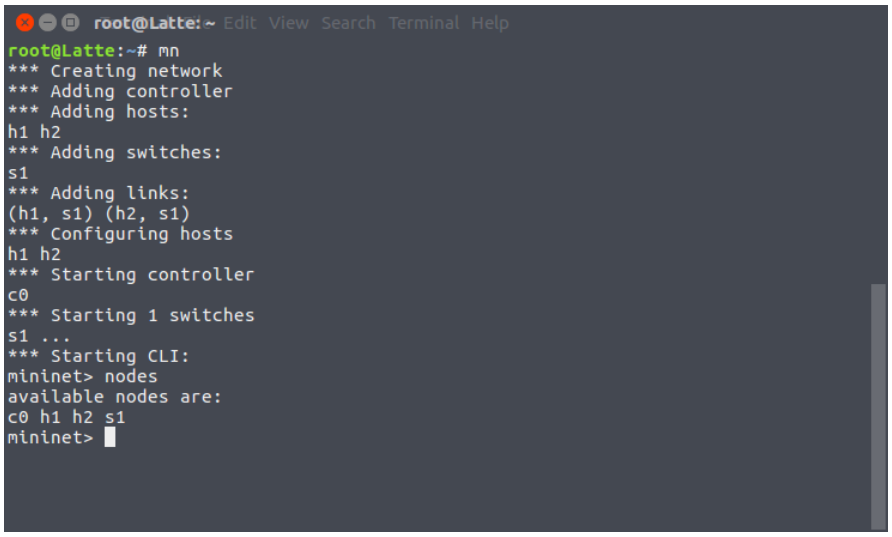


รูปที่ 8 แสดงผลลัพธ์จากการป้อนคำสั่ง mn

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 ใดบ้างใน topology ให้สั่งโดยใช้คำสั่ง

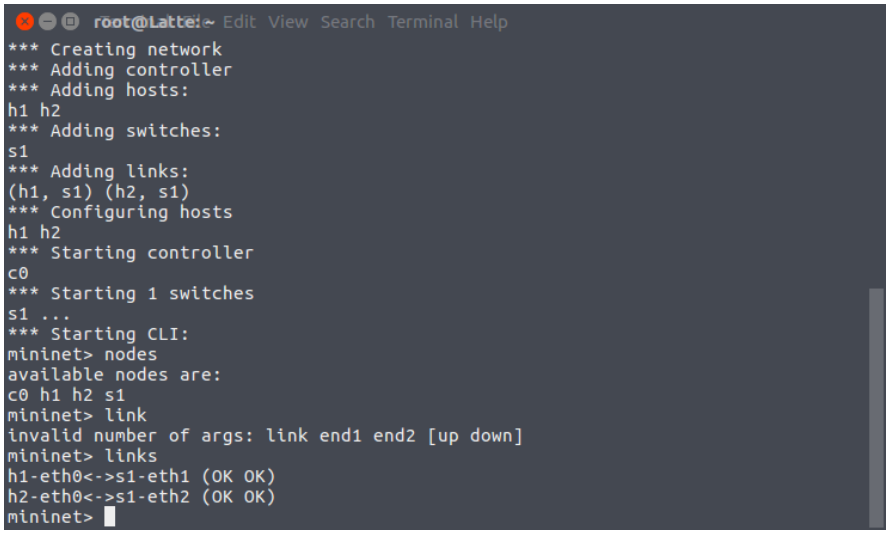
Mininet> nodes



รูปที่ 9 แสดงผลลัพธ์ของคำสั่ง nodes

4.2) ดู links ที่เชื่อมต่อกันระหว่าง nodes โดยใช้คำสั่ง

mininet> links

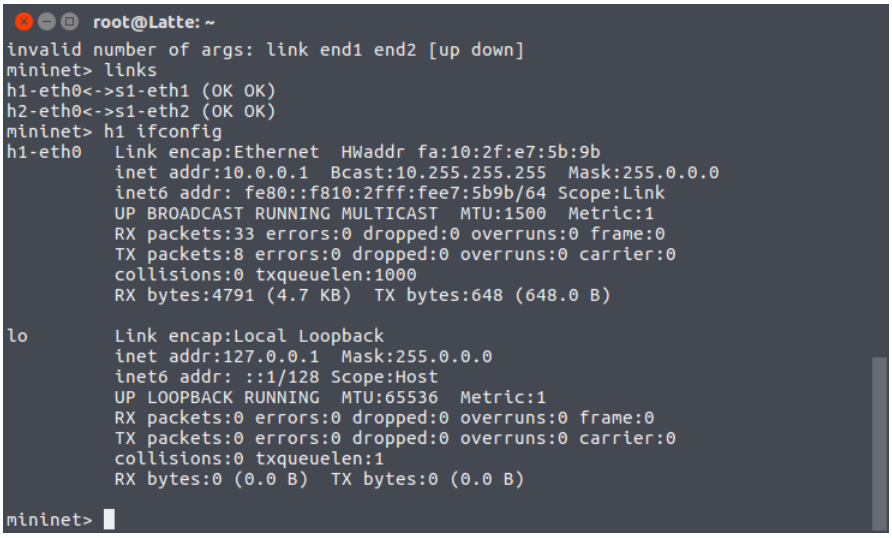


รูปที่ 10 แสดงผลลัพธ์ของคำสั่ง links

4.3) ดูทุกๆ interface ของnodes ใด ใช้คำสั่งต่อไปนี้

mininet> “ชื่อ nodes” ifconfig

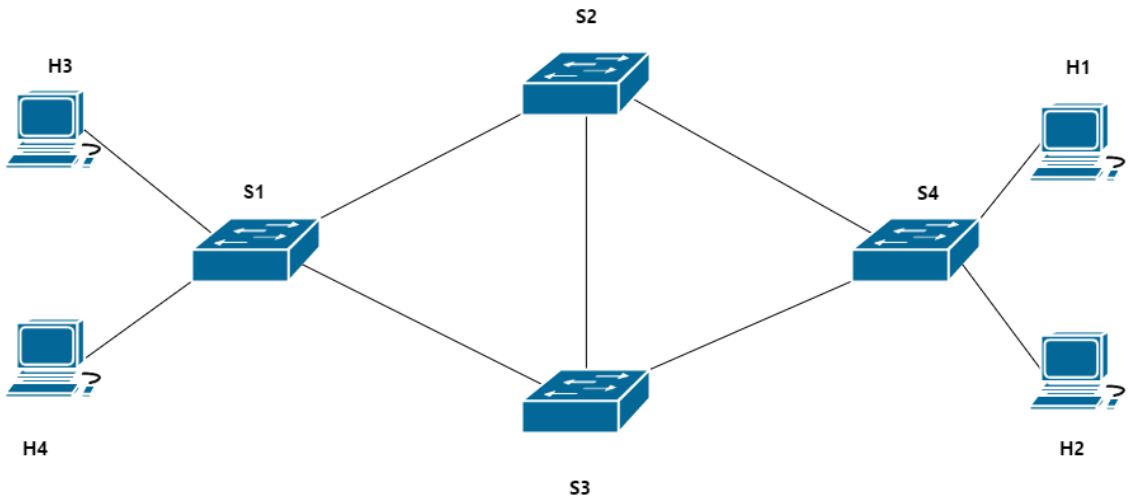
เช่น mininet> h1 ifconfig



รูปที่ 11 แสดงผลลัพธ์ของคำสั่ง h1 ifconnfig

4.3 ทดลองจำลองโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

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

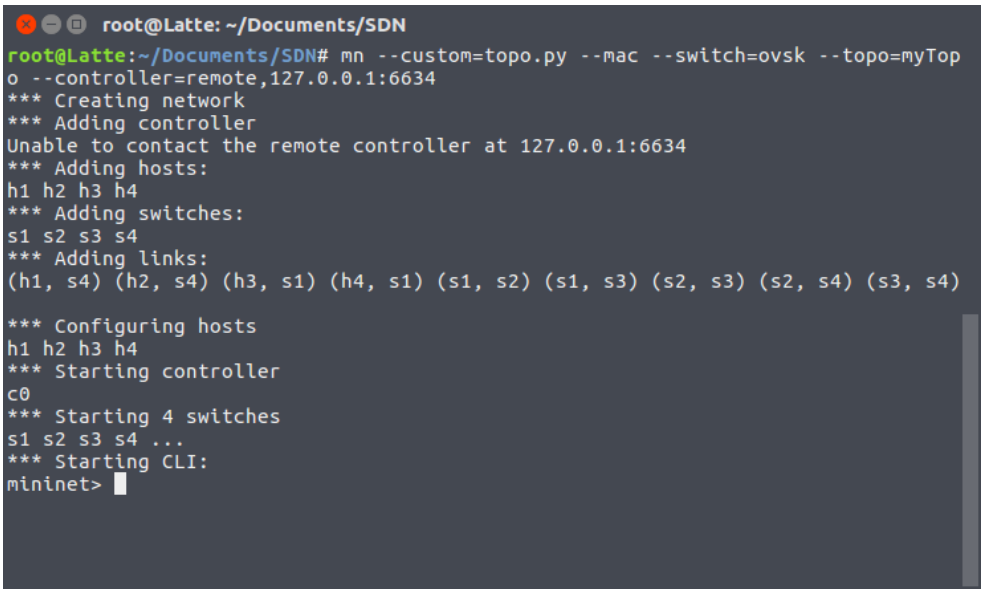


รูปที่ 12 แสดง topology ของแบบจำลองโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

โดยในการสร้าง topology ที่ซับซ้อน ต้องเขียนโปรแกรมด้วยภาษา Python จากนั้นสั่งด้วยคำสั่ง

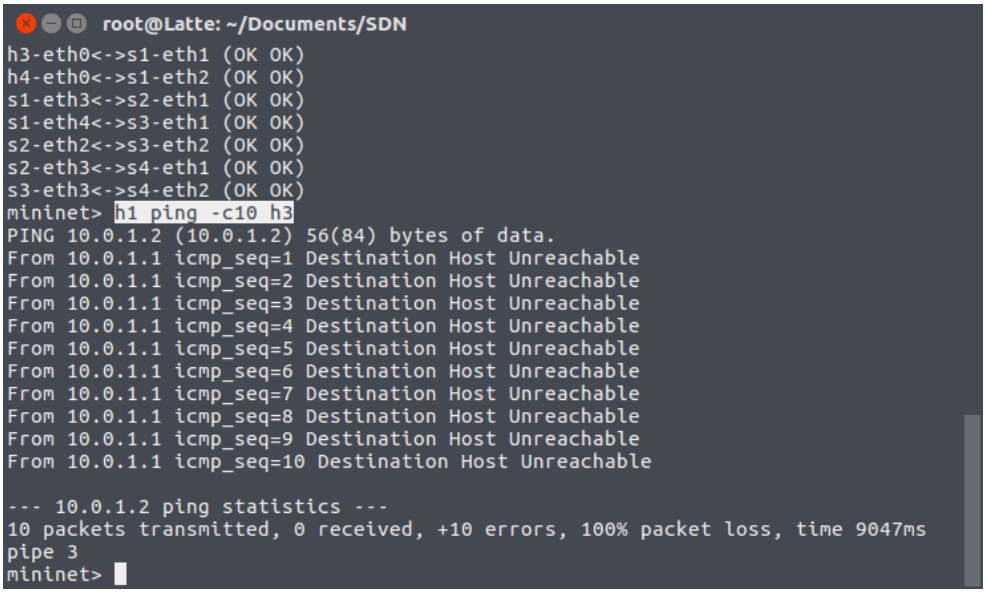
$ sudo mn --custom=topo.py --mac --switch=ovsk --topo=myTopo

--controller=remote,ip=127.0.0.1:6634

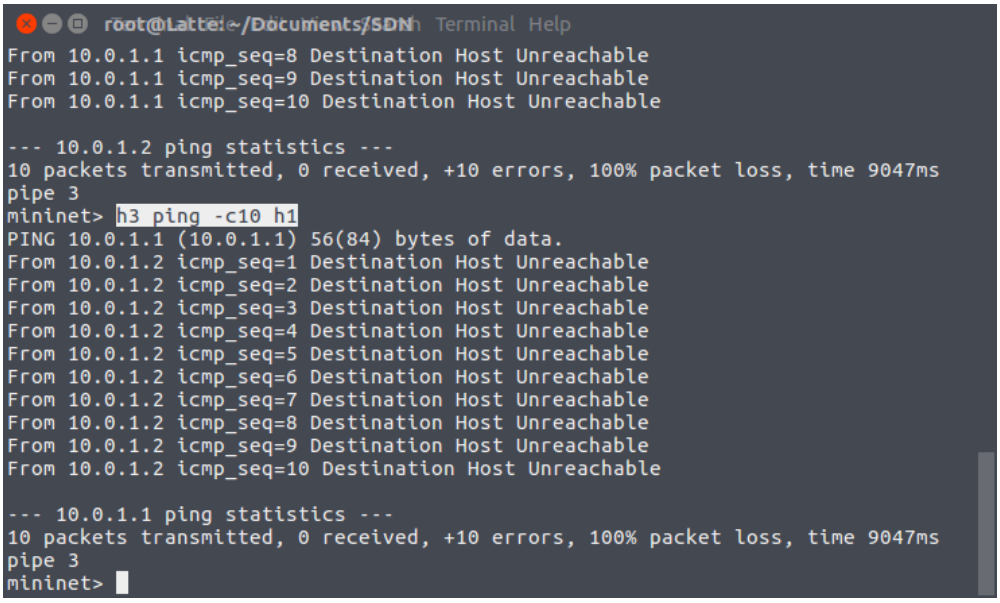


รูปที่ 13 แสดงการสร้าง topology ของแบบจำลองโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์

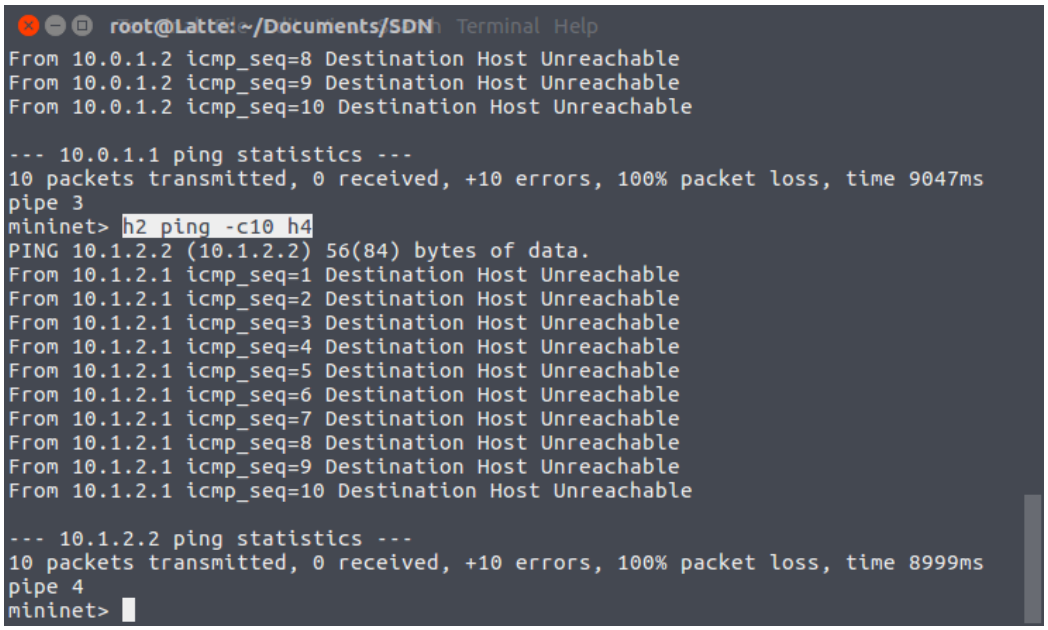
2) ทดลองทำการสื่อสารระหว่าง host ใน topology โดยการใช้โปรแกรม ping ระหว่าง h1 กับ h3 และ h2 กับ h4



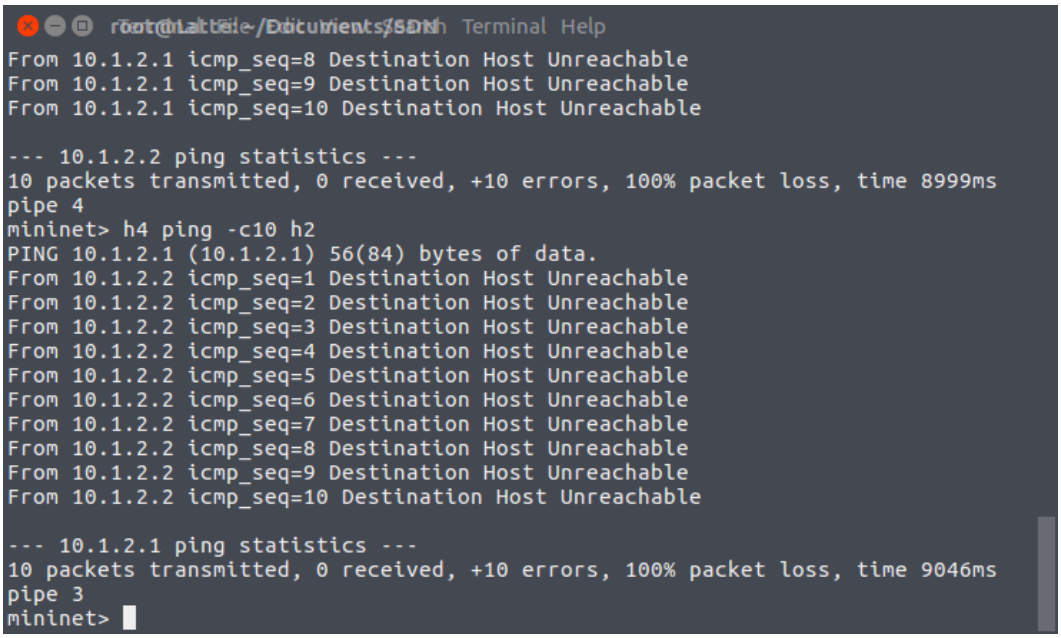
รูปที่ 14 แสดงการ ping ระหว่าง h1 กับ h3



รูปที่ 15 แสดงการ ping ระหว่าง h3 กับ h1



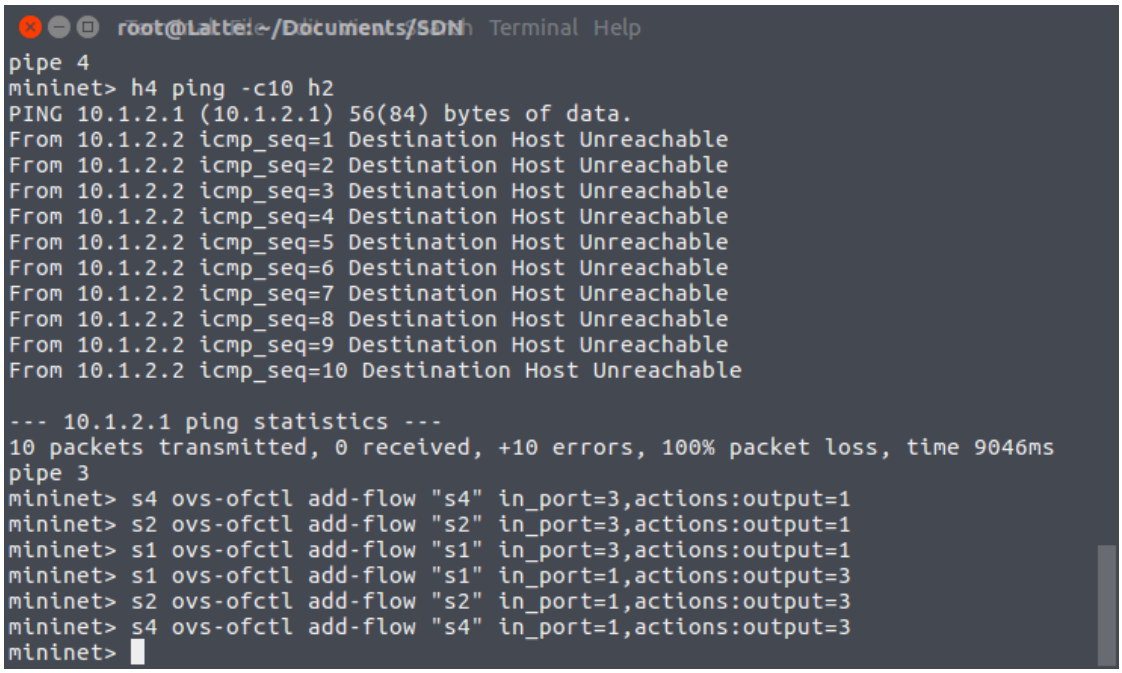
รูปที่ 16 แสดงการ ping ระหว่าง h2 กับ h4



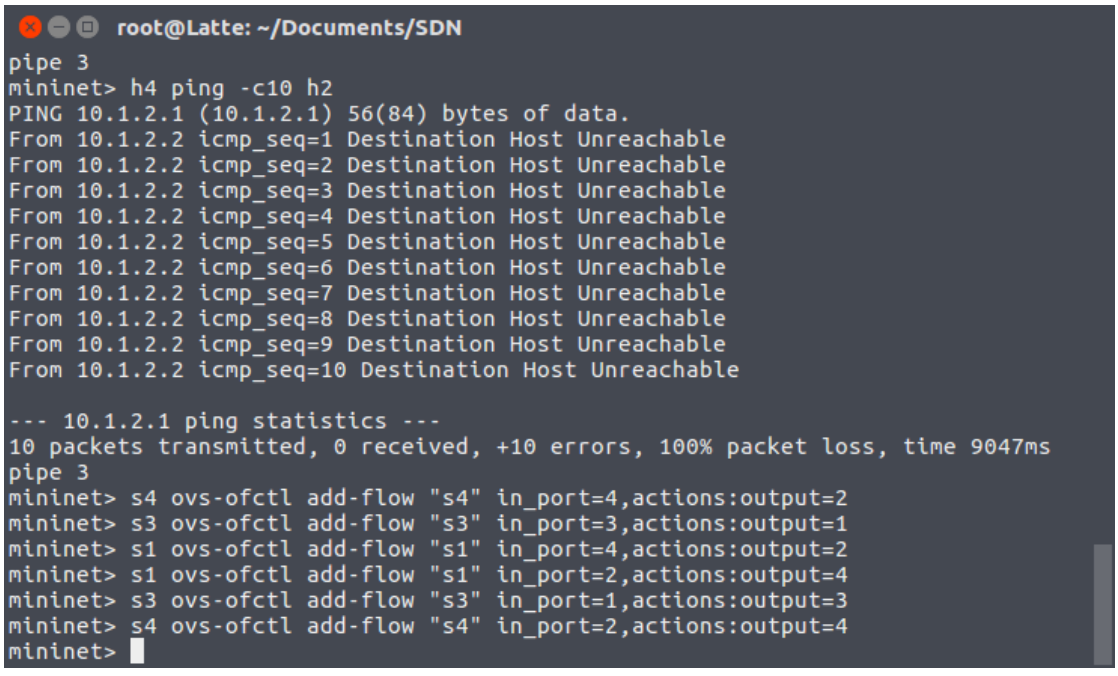
รูปที่ 17 แสดงการ ping ระหว่าง h4 กับ h2

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

3) ทำการเติม flow table ของ switch ทุกๆตัว

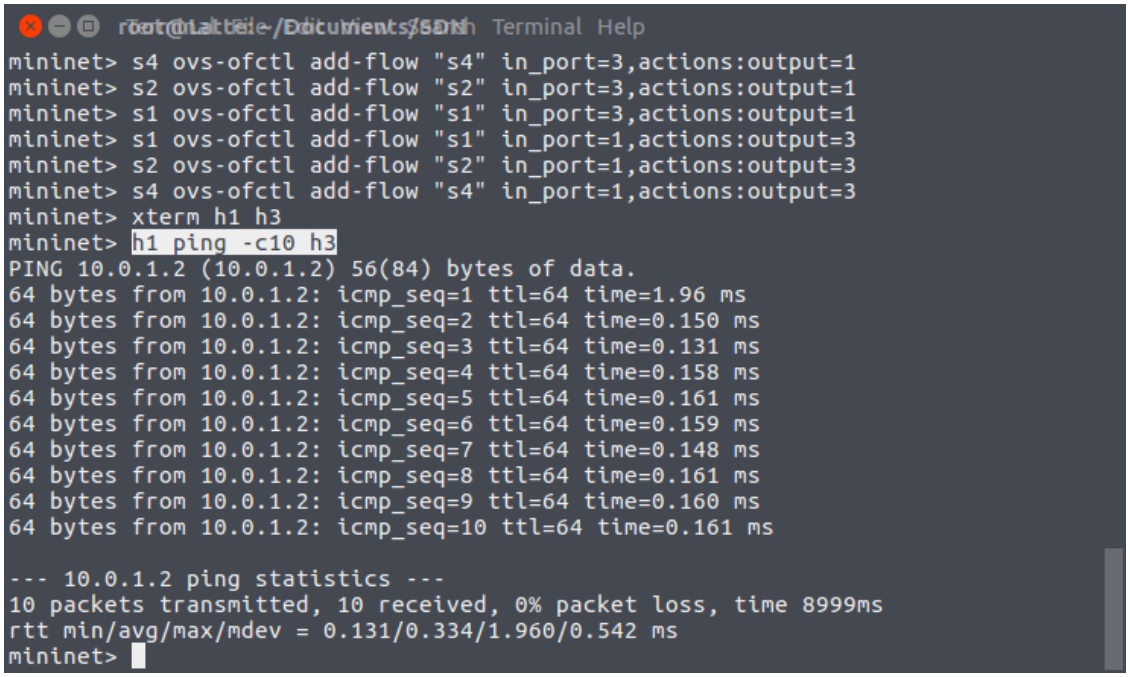


รูปที่ 18 ทำการเติม flow table เข้าไปใน switch S1, S2 และ S4

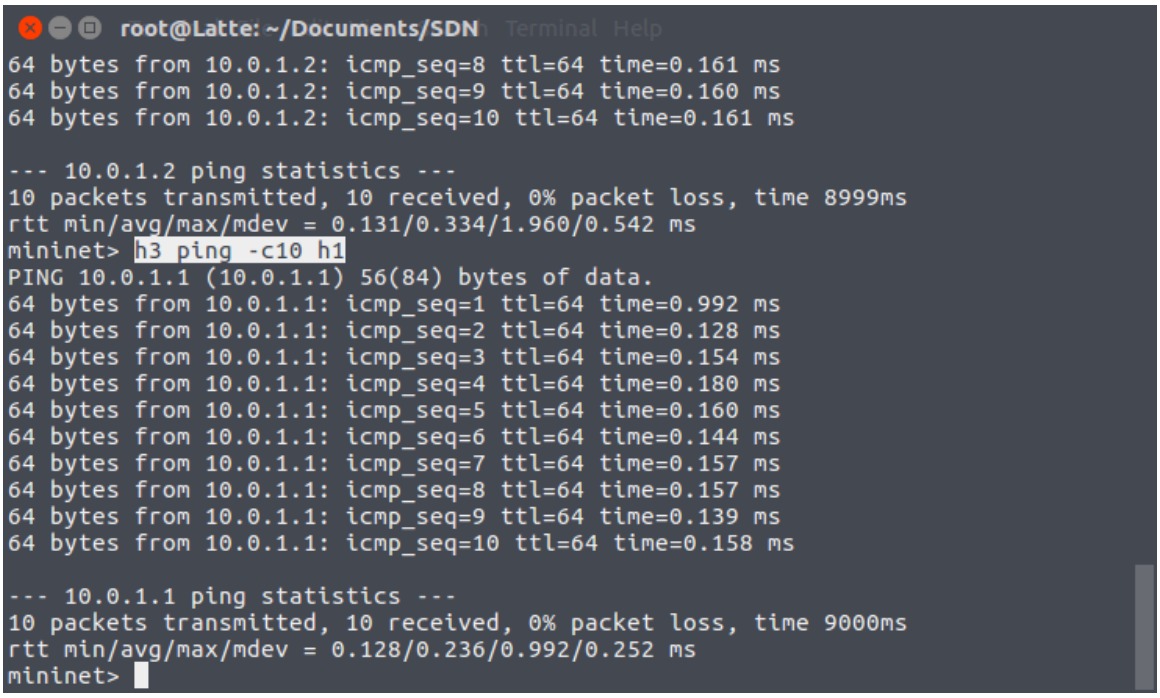


รูปที่ 19 ทำการเติม flow table เข้าไปใน switch S1, S3 และ S4

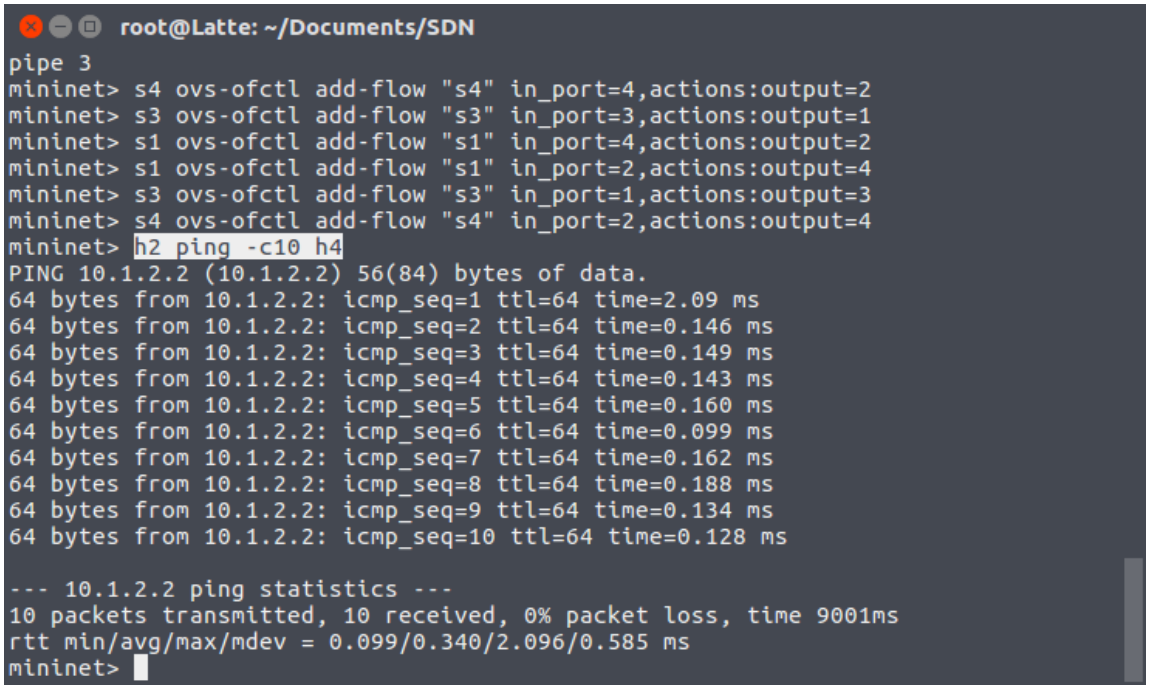
4) ทำการ ping ระหว่าง h1 กับ h3 และ h2 กับ h4 อีกครั้ง พบว่า สามารถ ping ระหว่างกันได้



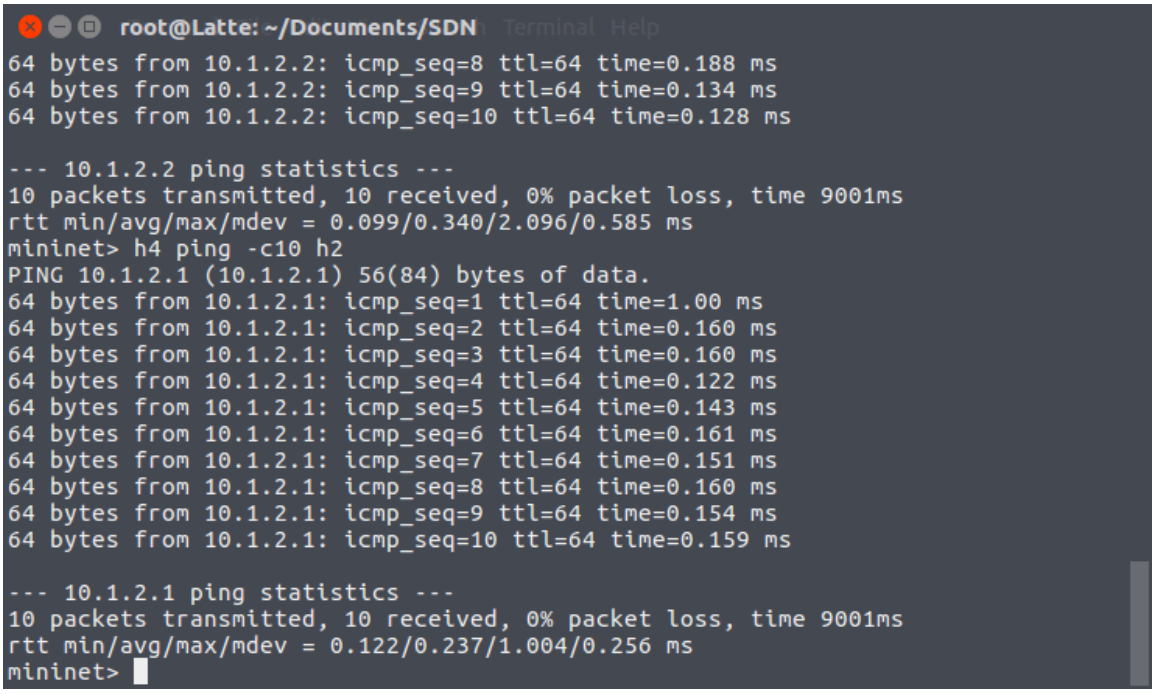
รูปที่ 20 แสดงผลลัพธ์การ ping ระหว่าง h1 กับ h3 หลังการเติม flow table



รูปที่ 21 แสดงผลลัพธ์ของการ ping ระหว่าง h3 กับ h1 หลังการเติม flow table



รูปที่ 22 แสดงการ ping ระหว่าง h2 กับ h4 หลังเติม flow table



รูปที่ 23 แสดงการ ping ระหว่าง h4 กับ h2 หลังเติม flow table

บทที่ 5

สรุปการทดลองและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

จากการทดลองสามารถสรุปได้ว่า โครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ช่วยอำนวยความสะดวกให้นักพัฒนาโครงข่าย อีกทั้งยังสามารถเอื้อให้โครงข่ายนั้นเกิดการพัฒนาหรือปรับปรุงเปลี่ยนแปลงได้ดีขึ้น

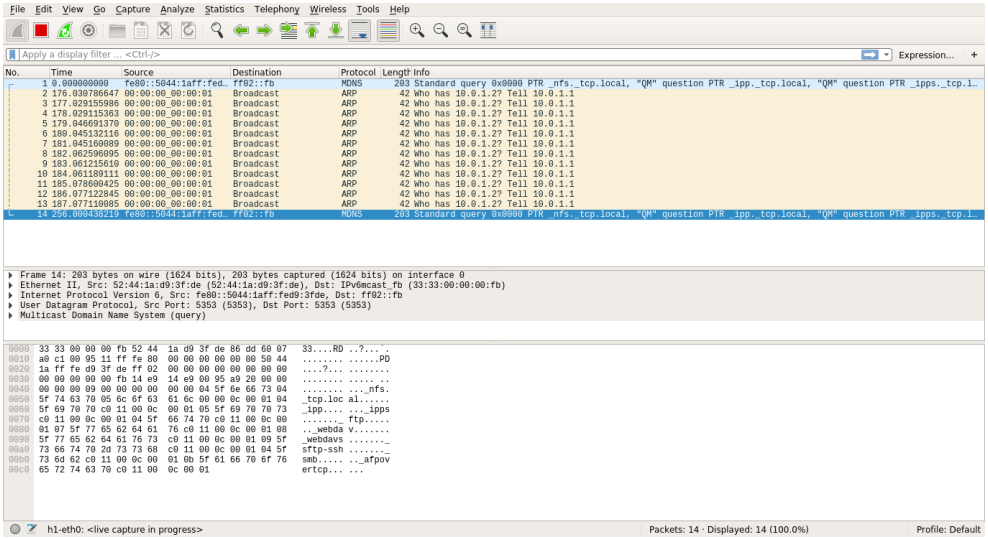
5.2 ข้อเสนอแนะ

1) หากมีเวลา จะสร้างแบบจำลองที่มี topologyที่ซับซ้อนยิ่งขึ้น

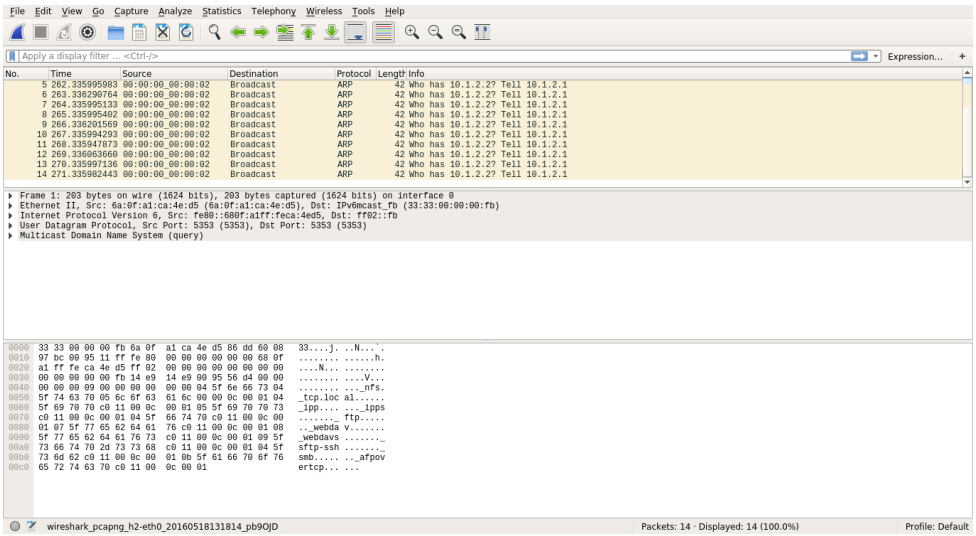
2) หากมีเวลา จะศึกษาถึงข้อดีข้อเสียของโครงข่ายที่ถูกนิยามโดยซอฟต์แวร์ในหลายๆด้าน

ภาคผนวก

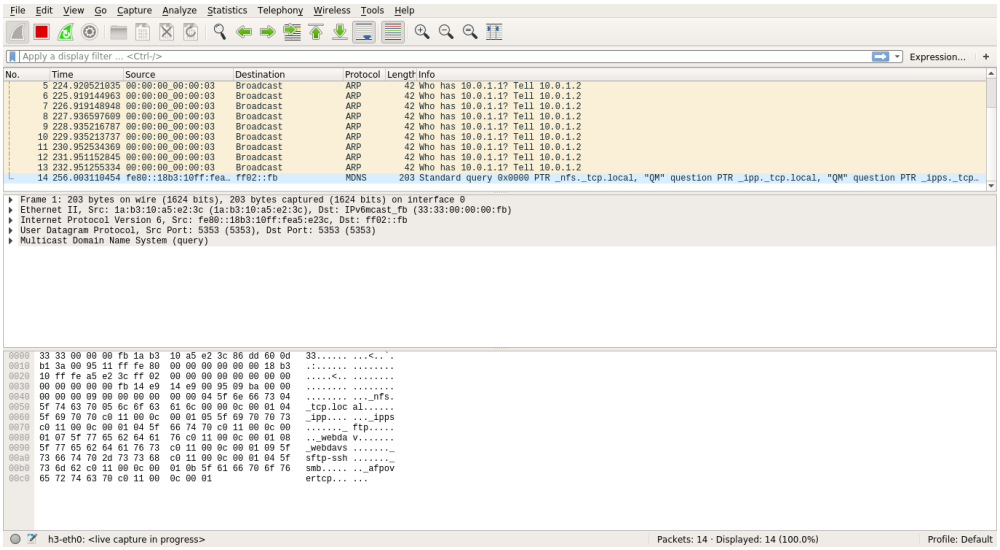
ภาคผนวก ก รูปภาพผลลัพธ์ที่ถูกบันทึกด้วยโปรแกรม Wireshark



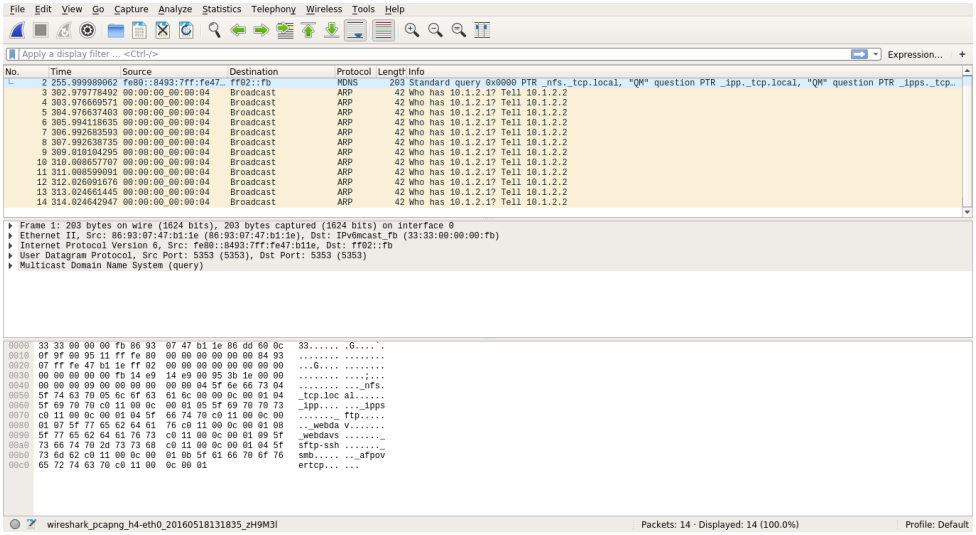
รูป แสดงผลลัพธ์ของ interafce h1-eth0 ก่อนการเติม flow table



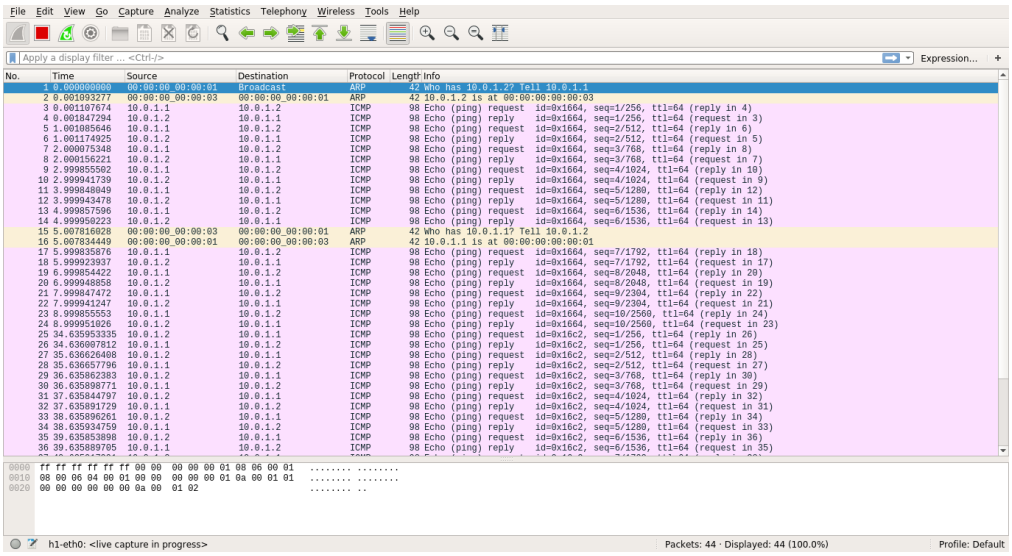
รูป แสดงผลลัพธ์ของ interface h2-eth0 ก่อนการเติม flow table



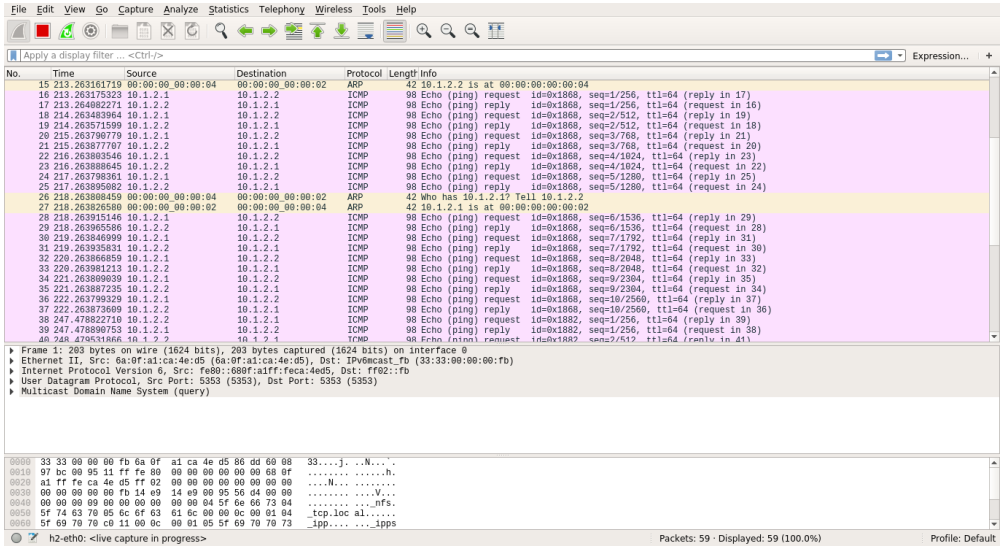
รูป แสดงผลลัพธ์ของ interface h3-eth0 ก่อนการเติม flow table



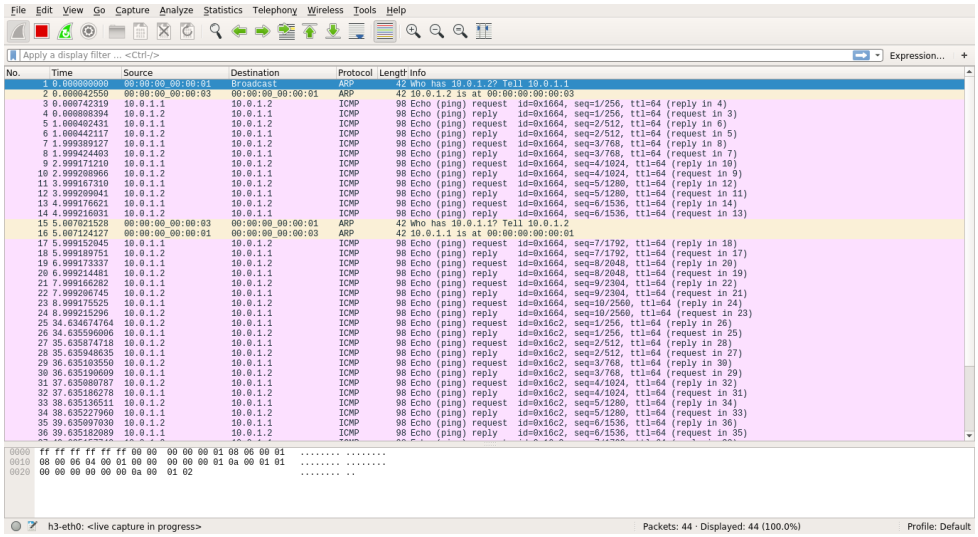
รูป แสดงผลลัพธ์ของ interface h4-eth0 ก่อนการเติม flow table



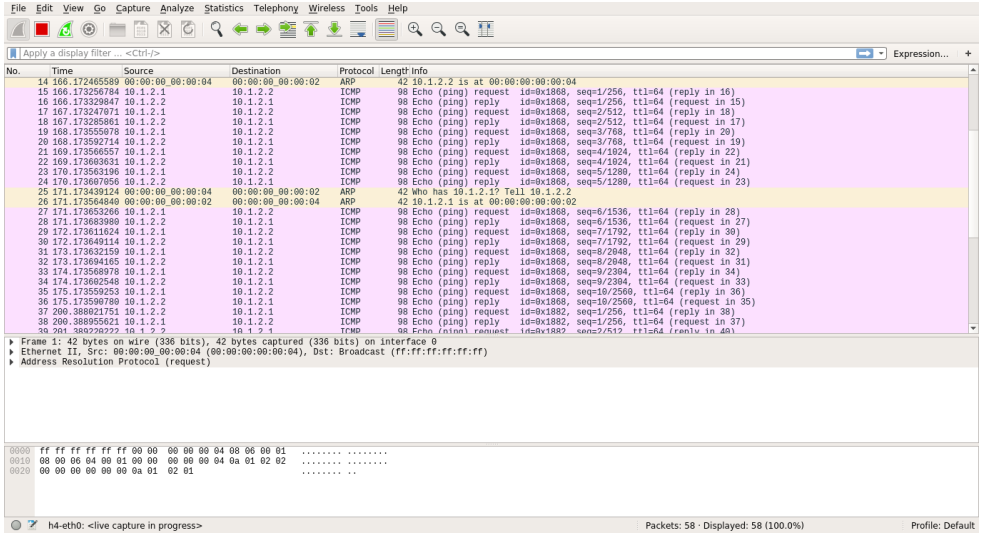
รูป แสดงผลลัพธ์ของ interface h1-eth0 หลังการเติม flow table



รูป แสดงผลลัพธ์ของ interface h2-eth0 หลังการเติม flow table



รูป แสดงผลลัพธ์ของ interface h3-eth0 หลังการเติม flow table



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.