บทความวิจัย/บทความวิชาการ

แนะนำและสาธิตการทดสอบการใช้งานเทคโนโลยีเปลี่ยนโลกอย่างง่าย: กรณีศึกษาโครงข่ายที่ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์ (เอสดีเอ็น)

INTRODUCTION AND DEMONSTRATION FOR SIMPLY TESTING DISRUPTIVE TECHNOLOGY : CASE STUDY OF SOFTWARE-DEFINED NETWORKING (SDN)

วิเชียร ปรีดาลัมพะบุตร
Wichean Preedalumpabut
สถาบันวิชาการทีโอที นนทบุรี 11000
TOT Academy, Nonthaburi 11000 Thailand

Corresponding E-mail : wpreeda@tot.co.th

Received Date August 2, 2019
Revised Date October 8, 2019
Accepted Date October 8, 2019

บทคัดย่อ

บทความนี้มีวัตถุประสงค์เพื่อนำเสนอการสาธิตการทดสอบการใช้งาน Software-Defined Networking (SDN) อย่างง่าย ๆ โดยมีวิธีการศึกษาคือการติดตั้งซอฟต์แวร์ต่าง ๆ ได้แก่ Mininet OpenDaylight และ OpenFlow Manager เพื่อสาธิตการใช้งาน SDN อย่างง่าย ๆ โดยการนำ Mininet มาใช้เพื่อจำลองการสร้างอุปกรณ์ของโครงข่ายที่ทำหน้าที่รองรับ Data plane สำหรับ OpenDaylight จะถูกใช้เป็น SDN controller ทำหน้าที่รองรับ Control plane ส่วน OpenFlow manager เป็นแอปพลิเคชันเพื่อทำงานร่วมกับ OpenDaylight ที่ใช้สั่งอุปกรณ์โครงข่ายให้ทำงานตามต้องการ สำหรับผลการศึกษาพบว่า สามารถทดสอบการทำงานพื้นฐานของโครงข่าย SDN ได้บรรลุตามวัตถุประสงค์โดยเป็นไปตามโครงสร้างของ SDN โดยมีสรุปและข้อเสนอแนะคือ ทำให้ผู้สนใจสามารถทดสอบการทำงานของโครงข่าย SDN อย่างง่าย ๆ ได้ด้วยตัวเองบนคอมพิวเตอร์ส่วนบุคคลโดยการ Download ซอฟต์แวร์ดังกล่าวข้างต้นและสามารถนำไปพัฒนาสร้างแอปพลิเคชันอื่น ๆ เช่น ด้านความปลอดภัยของโครงข่าย SDN วงจรเช่าเสมือนเลเยอร์ 2 (Virtual Private LAN Service หรือ VPLS) เป็นต้น

คำสำคัญ : เอสดีเอ็น มินิเนท โอเพนเดย์ไลท์ โอเพนโฟลว์เมเนเจอร์ วีพีแอลเอส

Abstract

The objective of this paper is illustrate how to test and operate software-defined networking (SDN). Case demonstration with the following software including Mininet, OpenDaylight, and OpenFlow Manager is organized. Mininet is used in simulating and creating the network devices (e.g. switches and hosts) which serve in data plane. OpenDaylight is deployed as SDN controller (the brain of the network devices used in control plane). OpenFlow Manager is also operated as an application to work according to OpenDaylight to define the functions to the network devices. The results of this study reveal that the test of basic functions in SDN network is achieved according to structure of SDN. The summary and suggestion from this study are that the learners can themselves test the functions of SDN network with their personal computer and download the software mentioned above. This enables them to develop and create other application software programs e.g. security in the SDN network, virtual private LAN service (VPLS).

Keywords : Software-Defined Networking (SDN), Mininet, OpenDaylight, OpenFlow Manager, VPLS

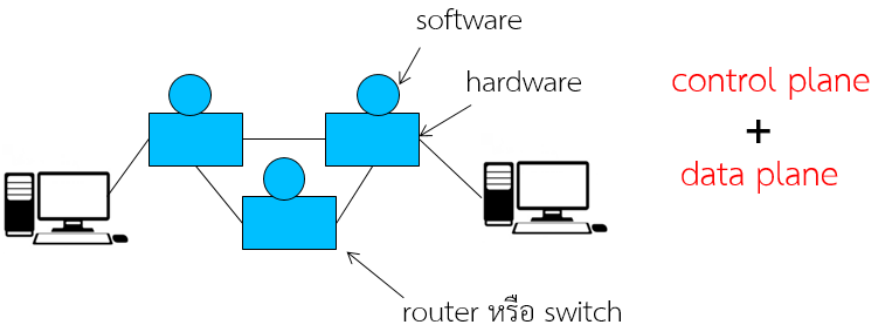
1. บทนำ

เป็นที่คาดการณ์ว่าโครงข่ายที่ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์ (Software-Defined Networking หรือ SDN) จะกลายเป็นหนึ่งในเทคโนโลยีเปลี่ยนโลกหรือ Disruptive technology เนื่องจากความสามารถในการใช้ซอฟต์แวร์บนโครงข่ายที่ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์ ทำให้โครงข่ายที่ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์ สามารถกำหนดหน้าที่การทำงานได้หลากหลายชนิดให้กับอุปกรณ์โครงข่ายที่เป็นมาตรฐานที่มีราคาไม่แพงในทำนองเดียวกับที่แอปพลิเคชันบนโทรศัพท์มือถือสามารถทำให้โทรศัพท์มือถือเป็นเครื่องบันทึกเสียง กล้องถ่ายรูป เครื่องเล่น CD บัตรถอนเงินสดจากตู้ ATM เป็นต้น จากข้อมูลของบริษัท AT&T (ผู้ให้บริการการสื่อสารโทรคมนาคมรายใหญ่ในประเทศสหรัฐอเมริกา) ระบุว่า ความต้องการการใช้งานที่เพิ่มขึ้นทั้งจากโทรศัพท์เคลื่อนที่ อุปกรณ์อินเทอร์เน็ตของสรรพสิ่ง (IoT) ทำให้ปริมาณการรับส่งข้อมูลบนโทรศัพท์เคลื่อนที่ในช่วงปี ค.ศ. 2007 - 2014 เพิ่มขึ้นถึง 100,000% (Donovan, 2014) ด้วยเหตุนี้ ผู้ให้บริการต้องวางแผนและออกแบบโครงข่ายเพื่อตอบสนองความต้องการของลูกค้าที่เพิ่มขึ้นอย่างมากได้อย่างรวดเร็ว AT&T มองเห็นว่าเทคโนโลยีหนึ่งที่มีความท้าทายและน่าสนใจที่จะนำมาใช้ในการออกแบบโครงข่ายคือโครงข่ายที่ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์ (AT&T, 2019) ในขณะเดียวกันในปี ค.ศ. 2012 Google มีแนวคิดใหม่ว่าควรนำโครงข่ายที่ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์ มาใช้ใน Data centers ของบริษัท Google เองทำให้โครงข่ายที่

ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์ ถูกใช้งานได้อย่างมีประสิทธิภาพเพิ่มมากขึ้น (Levy, 2012) และในปัจจุบันได้เริ่มมีการนำเอาแนวคิดของโครงข่ายที่ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์ มาให้บริการเสริมกับบริการวงจรเช่าเสมือนหรือ Virtual Private Network (VPN) services แก่ลูกค้ารายใหญ่ที่เรียกว่า Software-defined wide area network (SD-WAN) (AT&T, 2019) ทั้งนี้ในยุค 5G ได้มีการนำเทคโนโลยีของ SDN และ NFV (Network Function Virtualization) มาใช้ในการสร้างโครงข่ายหลักของ 5G ด้วย

แรงจูงใจได้เริ่มขึ้นจากทีมงานนักวิจัยจากมหาวิทยาลัย Stanford ประเทศสหรัฐอเมริกา (McKeown et al., 2008) ที่ต้องการสร้างนวัตกรรมให้กับโครงข่ายในมหาวิทยาลัยเพื่อให้นักวิจัยพัฒนาสามารถเข้าไปพัฒนาการทำงานของอุปกรณ์ในโครงข่าย เช่น Router หรือ Switch ให้มีประสิทธิภาพในการทำงานเพิ่มมากขึ้น เช่น การทำงานของการหาเส้นทาง (Routing) เป็นต้น แต่ไม่สามารถกระทำได้เนื่องจากระบบโครงสร้างภายในของอุปกรณ์ในโครงข่ายปัจจุบันยังมีการเชื่อมต่อที่ถูกกำหนดแบบเฉพาะสำหรับบริษัทตัวเองเท่านั้น ยังไม่มีการเชื่อมต่อแบบที่เป็นมาตรฐานหรือ Open interface อย่างไรก็ตาม ถ้ามีการพัฒนาเพื่อเอาแนวคิดของ SDN มาใช้จะทำให้การเชื่อมต่อของระบบโครงสร้างภายในของอุปกรณ์โครงข่ายที่มีการเชื่อมต่อที่เป็นมาตรฐานทำให้นักวิจัยสามารถสร้างนวัตกรรมใหม่ ๆ ให้เกิดขึ้นกับอุปกรณ์ในโครงข่ายได้

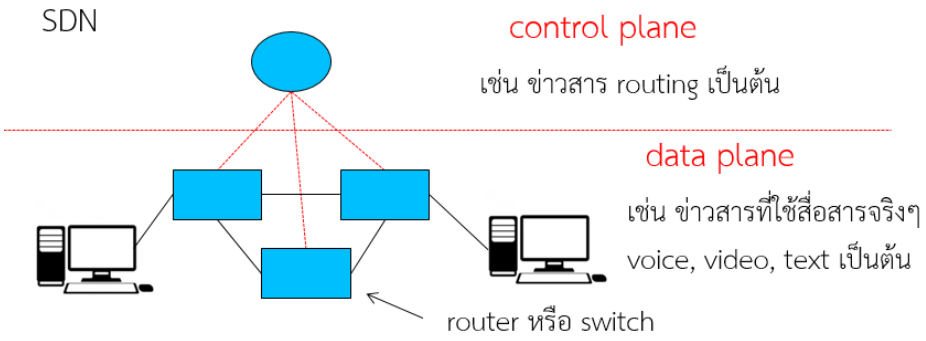
ก่อนที่จะกล่าวถึงแนวคิดของโครงข่ายที่ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์ จะขอกล่าวถึงโครงข่ายแบบดั้งเดิมก่อนที่ยังไม่ใช้แนวคิดของโครงข่ายที่ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์ ดังภาพที่ 1 ส่วนที่ทำหน้าที่ควบคุมการเชื่อมต่อ (Control plane) กับส่วนที่ทำหน้าที่รับส่งข้อมูล (Data plane) จะทำงานรวมอยู่บนอุปกรณ์ตัวเดียวกัน โดย Control plane จะทำหน้าที่ควบคุมการเชื่อมต่อของอุปกรณ์ในโครงข่ายเช่น Router หรือ Switch ที่ถูกดำเนินการโดย Software บน Router เช่น การหาเส้นทางการเชื่อมต่อให้กับ Router หรือ Switch ส่วน Data plane ทำหน้าที่รองรับข่าวสารทุกชนิดบนโครงข่าย เช่น ภาพ เสียง ตัวอักษร ที่ถูกดำเนินการโดย Hardware ของ Router หรือ Switch เป็นต้น



ภาพที่ 1 โครงสร้างของโครงข่ายแบบดั้งเดิมที่ยังไม่ใช้แนวคิดของ SDN

ที่มา : วิเชียร ปรีดาลัมพะบุตร (2561)

สำหรับแนวคิดของโครงข่ายที่ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์จะเป็นการแยกอุปกรณ์โครงข่ายทางกายภาพ เช่น Router หรือ Switch ออกเป็น 2 ส่วนคือ Control plane กับ Data plane ดังภาพที่ 2 คำสั่งควบคุมการทำงานของ Router หรือ switch จะอยู่บน Control plane ส่วนข่าวสารของผู้ใช้บริการ เช่น เสียง ตัวอักษร ภาพ จะอยู่บน Data plane ทั้งนี้เพื่อกำหนดหน้าที่การทำงานและการเชื่อมต่อที่เป็นมาตรฐาน ทำให้ SDN สามารถรองรับการเชื่อมต่อของอุปกรณ์ได้หลากหลายยี่ห้อ นอกจากนี้ยังทำให้นักวิจัยสามารถสร้างนวัตกรรมใหม่ ๆ ให้เกิดขึ้นกับอุปกรณ์ในโครงข่ายได้ด้วย เช่น นักวิจัยสามารถพัฒนาวิธีการหาเส้นทางที่ดีที่สุดใหม่ ๆ ได้ (Routing algorithm) บน Control plane แต่ถ้าเป็นโครงข่ายแบบดั้งเดิมที่ยังไม่ใช้แนวคิดโครงข่ายที่ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์ นักวิจัยจะไม่สามารถพัฒนาวิธีการหาเส้นทางที่ดีที่สุดใหม่ ๆ ได้ นอกจากนี้อุปกรณ์บน Data plane จะเป็นอุปกรณ์ที่เป็นมาตรฐาน ดังนั้นราคาจึงไม่แพง และผู้ใช้สามารถกำหนดให้ Data plane ทำงานเป็นอะไรก็ได้ตามต้องการ เช่น Router Switch Multiprotocol Label Switching (MPLS) หรือ Firewall เป็นต้น โดยสามารถไปกำหนดหน้าที่การทำงานดังกล่าวได้ที่ Control plane

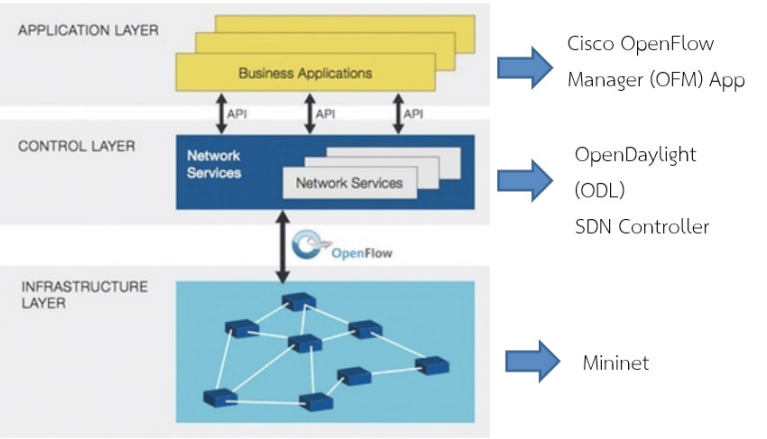


ภาพที่ 2 โครงสร้างของโครงข่ายแบบที่ใช้แนวคิดของ SDN

ที่มา : วิเชียร ปรีดาลัมพะบุตร (2561)

2. โครงสร้างของ SDN

กลุ่มของบริษัทผู้ผลิตและผู้ให้บริการได้แก่ AT&T Google China Unicom NTTGroup Deutsche Telekom China Telecom Ericsson NEC Huawei Samsung Cisco Microsoft และ Intel เป็นต้น ได้ร่วมกันก่อตั้งองค์กรที่ไม่แสวงหาผลกำไรเรียกว่า Open Networking Foundation (ONF) มีจุดมุ่งหมายเพื่อขับเคลื่อนการปรับเปลี่ยนโครงสร้างพื้นฐานของโครงข่ายและรูปแบบธุรกิจของผู้ให้บริการ (Open Networking Foundation, 2011) องค์กรที่ไม่แสวงหาผลกำไรได้กำหนดมาตรฐานต่าง ๆ ของโครงข่ายที่ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์ ยกตัวอย่างเช่น โครงสร้าง การเชื่อมต่อ และองค์ประกอบ เป็นต้น ดังภาพที่ 3



ภาพที่ 3 โครงสร้างของ SDN

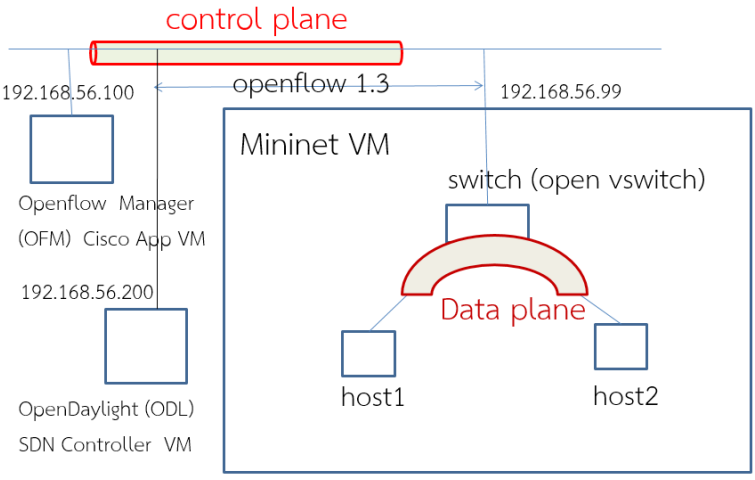
ที่มา : Open Networking Foundation (2011)

ภาพที่ 3 แสดงถึงโครงข่ายที่ใช้แนวคิดโครงข่ายที่ถูกกำหนดหน้าที่การทำงานโดยซอฟต์แวร์ ประกอบด้วย 3 ส่วนได้แก่ Application Control และ Infrastructure โดย Application เป็นส่วนที่กำหนดการทำงานของอุปกรณ์ในโครงข่าย เช่น หาเส้นทาง (Routing) การเชื่อมต่อให้กับลูกค้า หรือบริการเสริมอื่น ๆ เช่น ป้องกันความปลอดภัยจากการใช้อินเทอร์เน็ตให้กับลูกค้า ส่วนของ Control หรือบางทีเรียกว่า SDN controller จะจัดการดูสถานะของอุปกรณ์โครงข่าย ส่งผ่านคำสั่งการทำงานไปยังอุปกรณ์โครงข่ายด้วยมาตรฐานการเชื่อมต่อ OpenFlow และคอยรับคำสั่งการทำงานมาจากแอปพลิเคชันผ่านทางมาตรฐานการเชื่อมต่อ Application Programming Interface (API) ได้แก่ Representational State Transfer (REST) API และ JAVA API เป็นต้น สำหรับส่วนสุดท้ายคือ Infrastructure เป็นส่วนของอุปกรณ์ในโครงข่าย

สำหรับในส่วนของแอปพลิเคชัน จะสาธิตด้วยแอปพลิเคชันของบริษัท Cisco ที่ชื่อ OpenFlow Manager (OFM) (Medved, et al., 2016) ส่วนของ Control จะสาธิตด้วย SDN controller ที่เป็น Open source software คือ OpenDaylight (ODL) (OpenDaylight, 2013) และส่วนสุดท้าย คือ Infrastructure จะสาธิตด้วย Mininet (Lantz & Heller, 2010)

3. การติดตั้ง Software สำหรับการสาธิตการใช้งาน SDN

เริ่มจากการติดตั้งซอฟต์แวร์ทั้งหมดลงบนเซิร์ฟเวอร์ทั้ง 3 ตัวแยกกันสำหรับแต่ละส่วน โดยเซิร์ฟเวอร์เหล่านี้จะเป็น Virtual Machine (VM) ที่ถูกสร้างจำลองด้วยซอฟต์แวร์ ชื่อ VirtualBox ที่เป็น Open source ของบริษัท Oracle (VirtualBox, 2007) โดยมีการเชื่อมต่อของเซิร์ฟเวอร์ทั้ง 3 เซิร์ฟเวอร์ ดังภาพที่ 4



ภาพที่ 4 การเชื่อมต่อของ 3 เซิร์ฟเวอร์ ที่ทำหน้าที่ Application Control และ Infrastructure

ที่มา : วิเชียร ปรีดาลัมพะบุตร (2561)

3.1 การติดตั้งและกำหนดค่า สำหรับ SDN controller: OpenDaylight (ODL)

ในที่นี้จะเลือกใช้ OpenDaylight (ODL) ทำหน้าที่ SDN controller โดยในการติดตั้งเราจะต้องสร้าง Linux Ubuntu server version 14.04.6 LTS 64 bit หลังจากนั้นจึงติดตั้ง ODL (Linkletter, 2016) บน Linux Ubuntu server ดังกล่าวข้างต้น สำหรับการสร้าง Linux Ubuntu server (วิเชียร, 2559) นี้จะติดตั้งผ่าน VirtualBox ด้วยการสร้าง VM ใหม่ จากไฟล์ ubuntu-14.04.6-server-amd64.iso (Ubuntu, 2014) จากนั้นให้เลือก Network adapter 1 เป็น NAT เพื่อใช้ติดต่ออินเทอร์เน็ต และเลือก Network adapter 2 เป็น Host-only adapter กำหนด IP address สำหรับ ODL VM เป็น 192.168.56.200/24 เพื่อให้ ODL VM ติดต่อกับเครื่อง Host (Windows) Mininet VM และ OpenFlow Manager VM ได้บน Control plane ตามภาพที่ 4 หลังจากที่ได้สร้าง Linux Ubuntu server เสร็จเรียบร้อย จากนั้นจะเริ่มทำการติดตั้ง ODL บน Linux Ubuntu server โดยขั้นตอนแรกจะเป็นการติดตั้ง Java run-time environment ด้วยคำสั่ง

```
$ sudo apt-get update
$ sudo apt-get install default-jre-headless
```

ตั้งค่าตัวแปรของ JAVA_HOME environment โดยการแก้ไขไฟล์ bashrc

```
$ nano ~/.bashrc
```

โดยให้เพิ่มบรรทัดข้างล่างนี้ลงไปบนไฟล์ bashrc และทำการบันทึก

export JAVA_HOME=/usr/lib/jvm/default-java

จากนั้นให้ run ไฟล์

$ source ~/.bashrc

แล้วติดตั้ง OpenDaylight ด้วยการดาวน์โหลดไฟล์ .zip จาก Website ของ OpenDaylight (OpenDaylight, 2016) ด้วยคำสั่ง (โดยเลือก OpenDaylight version 0.4.0-Beryllium)

$ wget https://nexus.opendaylight.org/content/repositories/opendaylight.release/org/opendaylight/integration/distribution-karaf/0.4.0-Beryllium/distribution-karaf-0.4.0-Beryllium.zip

จากนั้น unzip ไฟล์ ด้วยคำสั่ง

$ unzip distribution-karaf-0.4.0-Beryllium.zip

ต่อจากนั้น run OpenDaylight ด้วยการ run คำสั่ง karaf

$ cd distribution-karaf-0.4.0-Beryllium

$ ./bin/karaf

จะได้หน้าจอตามภาพที่ 5 ดังนี้

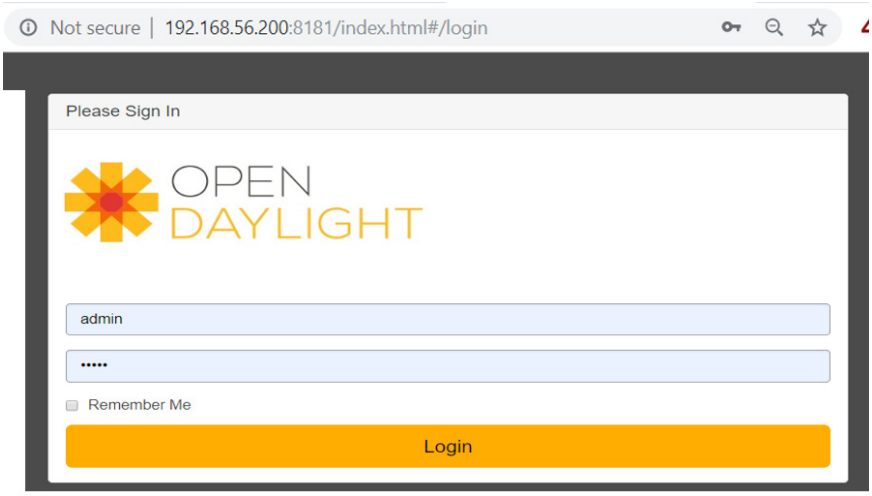
ภาพที่ 5 หน้าจอจากการสั่ง run คำสั่ง ./bin/karaf

ที่มา : OpenDaylight (2013)

จากนั้นจะติดตั้ง Feature ที่จำเป็นของ OpenDaylight

opendaylight-user@root> feature:install odl-restconf-all odl-l2switch-all odl-mdsal-apidocs odl-dlux-all odl-openflowplugin-all

จากนั้นทดสอบการทำงานของ OpenDaylight GUI ด้วยการใช้ Browser เช่น Chrome บนเครื่อง Windows ด้วยการเข้าไปที่ URL ที่ http://192.168.56.200:8181/index.html โดยมีค่า Default ของ Username และ Password เป็น Admin ทั้งคู่ จะได้หน้าจอดังแสดงตามภาพที่ 6



ภาพที่ 6 การเข้าถึง OpenDaylight ด้วยการตั้งค่า Default ของ Username และ Password เป็น Admin

ที่มา : OpenDaylight (2013)

3.2 การติดตั้งและกำหนดค่าสำหรับ Mininet

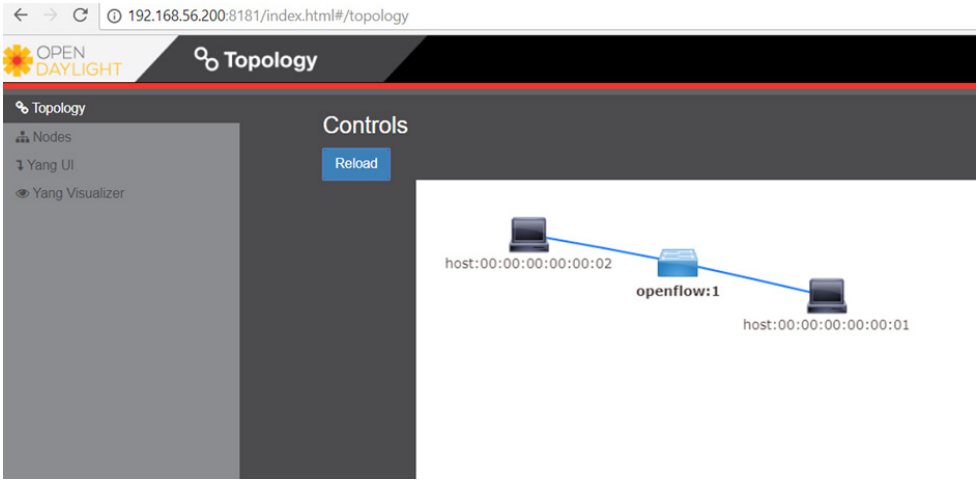
Mininet (Lantz & Heller, 2010) เป็น Open source สำหรับใช้จำลองอุปกรณ์ในโครงข่ายที่จะถูกใช้จำลอง Switch หรือเรียก Open Virtual Switch (OVS) อย่างไรก็ตาม ในตัว Mininet เองสามารถทำหน้าที่ได้ทั้ง Application, Control และ Infrastructure แต่ในที่นี้เราจะใช้จำลองเฉพาะในส่วนของ Infrastructure สำหรับการติดตั้ง Mininet (วิเชียร, 2559) (ในที่นี้จะใช้ Mininet 2.2.0 on Ubuntu 14.04 LTS - 32 bit (Lantz, 2017)) นั้น จะติดตั้งผ่าน VirtualBox ด้วยการนำเข้า (Import) ไฟล์ mininet-2.2.0-150106-ubuntu-14.04-server-i386.ovf ทำให้เราได้ Mininet ที่ทำงานบน Linux Ubuntu server version 14.04 32bit (ในที่นี้ไม่ต้องสร้าง Linux Ubuntu server ไว้ก่อนเหมือนกรณีการสร้าง ODL VM เนื่องจากไฟล์ mininet-2.2.0-150106-ubuntu-14.04-server-i386.ovf ที่เรานำเข้าบน VirtualBox มีการสร้าง Linux Ubuntu server ไว้แล้ว) จากนั้น ให้เลือก network เป็น host-only adapter และกำหนด IP address สำหรับ Mininet VM เป็น 192.168.56.99/24 จากนั้นจะสร้างอุปกรณ์บนโครงข่ายด้วยคำสั่ง mn[1] ตามด้วยค่าพารามิเตอร์ต่าง ๆ ดังนี้

[1] sudo mn help เพื่อดูการใช้คำสั่ง mn

$sudo mn[2]--controller=remote,ip=192.168.56.200 --switch=ovs,protocols=OpenFlow13 --topo=single,2 --mac

โดย Mininet VM ที่ run คำสั่ง mn นี้ จะมี IP address คือ 192.168.56.99 สำหรับค่าพารามิเตอร์ต่าง ๆ ของคำสั่ง mn จะเป็นดังนี้ --controller=remote,ip=192.168.56.200 สั่งให้ Mininet VM เชื่อมแบบ remote (ในความหมายที่เป็น VM คนละตัวกัน) กับ SDN controller (ODL VM) ที่มี IP address 192.168.56.200, -switch=ovs,protocols =OpenFlow13 คือ Switch ที่ถูกสร้างจะทำงานใน mode ของ OVS และ OVS switch นี้จะติดต่อกับ SDN controller ด้วย OpenFlow 1.3 และ --topo=single,2 จะสร้าง ovs switch 1 ตัว และมีจำนวน host 2 ตัวเชื่อมกับ OVS switch และ --mac จะมีการกำหนดค่า MAC address แบบอัตโนมัติ 00:00:00:00:00:01 และ 00:00:00:00:00:02 ให้กับ host h1, h2 ตามลำดับ ส่วนค่า default ของ IP address ของ host h1, h2 คือ 10.0.0.1/24 และ 10.0.0.2/24 ตามลำดับ

จากนั้นเราจะตรวจสอบโครงข่ายที่สร้างขึ้นที่ประกอบด้วย Switch 1 ตัว และ host 2 ตัว ด้วยการใช้ Browser เช่น Chrome เราเข้าไปที่ URL ที่ http://192.168.56.200:8181/index.html นั้นหมายความว่าเราเข้าไปตรวจสอบโครงข่ายที่เราสร้างขึ้นด้วย Mininet VM ผ่านทาง OpenDaylight SDN controller (192.168.56.200) เราจะได้โครงข่ายที่สร้างขึ้นเป็นไปตามที่เราต้องการดังแสดงในภาพที่ 7 คือประกอบด้วย Switch 1 ตัว และ Host 2 ตัว



ภาพที่ 7 การเชื่อมต่อของอุปกรณ์ในโครงข่ายที่ถูกสร้างขึ้นตามคำสั่ง mn ข้อ 3.2 ด้วย OpenDaylight

ที่มา : OpenDaylight Project (2013)

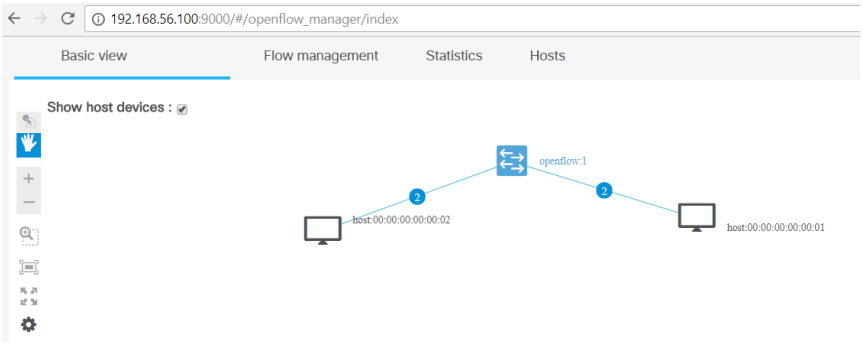
[2] ก่อนที่จะสั่ง ให้ Mininet VM ทำงานด้วยคำสั่ง sudo mn ข้างต้นจะต้องสั่ง run ODL VM ก่อน

3.3 การติดตั้งและกำหนดค่าสำหรับแอปพลิเคชัน : Cisco OpenFlow Manager (OFM)

ในที่นี้เราจะใช้ OpenFlow Manager (OFM) ของบริษัท Cisco เป็นแอปพลิเคชัน เพื่อทดสอบการทำงานร่วมกันกับ OpenDaylight และ Mininet โดยในการติดตั้งเราจะต้องสร้าง Linux Ubuntu server version 14.04.6 LTS 64 bit บน VirtualBox (ในทำนองเดียวกับการสร้าง ODL VM) โดยเลือก network adapter 1 เป็น NAT เพื่อใช้ติดต่อ Internet และเลือก Network adapter 2 เป็น Host-only adapter กำหนด IP address สำหรับ OFM VM เป็น 192.168.56.100/24 จากนั้นจึงติดตั้ง OFM Cisco Application (Medved, et al., 2016) บน Linux Ubuntu server โดยเริ่มใช้คำสั่ง ที่ Linux Ubuntu server ดังนี้

```
$ sudo apt-get update && sudo apt-get install
$ curl –sL https://deb.nodesource.com/setup_4.x | sudo -E bash -
$ sudo apt-get install -y nodejs
$ sudo aptitude install -y npm
$ sudo apt-get install -y git
$ sudo npm install -g grunt-cli
$ git clone https://github.com/CiscoDevNet/OpenDaylight-Openflow-App.git
$ sed -i ‘s/localhost/192.168.56.200/g’ ./OpenDaylight-Openflow-App/ofm/src/common/config/env.module.js
$ cd OpenDaylight-Openflow-App/ && sudo npm install
$ grunt
```

ณ จุดนี้ Ubuntu server ได้ทำงานแล้วและได้มีการสั่ง run OpenFlow Manager แล้ว จากนั้นเราสามารถตรวจสอบการทำงานของ OpenFlow Manager Application ได้ด้วยการใช้ Browser เช่น Chrome บนเครื่อง Windows เราเข้าไปที่ URL ที่ http://192.168.56.100:9000 จะได้ผลตามภาพที่ 8



ภาพที่ 8 การเชื่อมต่อของอุปกรณ์ในโครงข่ายที่ถูกสร้างขึ้น ตามคำสั่ง mn ข้อ 3.2 ด้วย OpenFlow Manager (OFM) Application ของบริษัท Cisco

ที่มา : CiscoDevNet (2014)

4. การทดสอบการใช้งานร่วมกันของ Mininet OpenDaylight และ OpenFlow Manager

ในการทดสอบนี้ในส่วนของ Infrastructure หรือ อุปกรณ์ในโครงข่ายเราใช้ Mininet VM สร้าง Host 2 ตัว เชื่อมต่อผ่าน Switch 1 ตัว สำหรับส่วนของ Control หรือ SDN controller เราใช้ OpenDaylight VM เพื่อจัดการและดูสถานะของโครงข่าย เพื่อสั่งการทำงานลงไปยัง switch สำหรับการเชื่อมต่อระหว่าง OpenDaylight กับ Switch บน Mininet จะใช้มาตรฐานการเชื่อมต่อ OpenFlow และส่วนสุดท้ายเป็นแอปพลิเคชัน ที่เราใช้ OpenFlow Manager เพื่อจัดการให้ Switch ทำงานตามที่เราต้องการ สำหรับการเชื่อมต่อระหว่าง OpenFlow Manager กับ OpenDaylight เราใช้มาตรฐานการเชื่อมต่อ REST API ที่ Mininet VM ก่อนที่จะมีการสั่งการทำงานเพิ่มเติมที่ OpenFlow Manager ให้ดำเนินการดังนี้

mininet> pingall

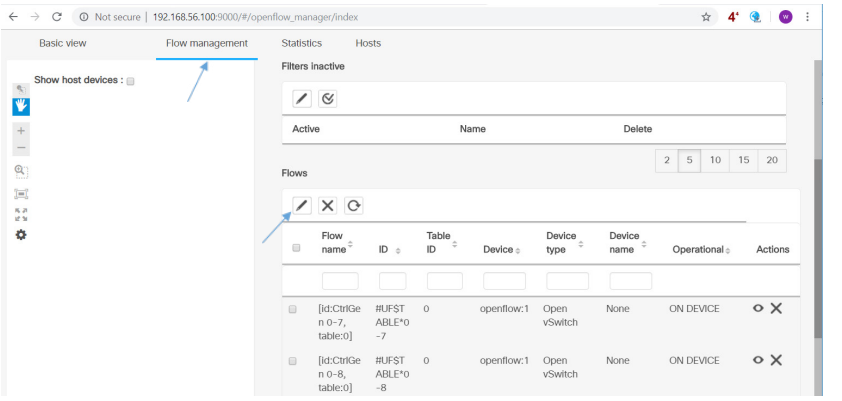
*** Ping: testing ping reachability

h1 -> h2 หมายถึง host1 (h1) สามารถติดต่อ host2 (h2)

h2 -> h1 หมายถึง host2 (h2) สามารถติดต่อ host1 (h1)

*** Results: 0% dropped (2/2 received)

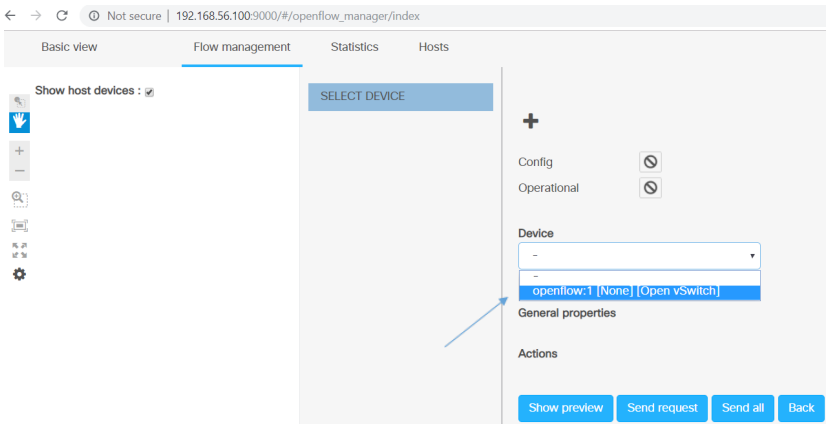
เราใช้คำสั่ง Pingall เพื่อทดสอบการเชื่อมต่อระหว่าง h1 กับ h2 ผ่าน Switch จะพบว่า การเชื่อมต่อระหว่าง h1 กับ h2 ผ่าน Switch สำเร็จเนื่องจากได้มีการติดตั้ง Feature ที่จำเป็นบน OpenDaylight เพื่อให้ Switch ทำงานพื้นฐานได้ในลำดับต่อไปจะมีการสั่งการ ที่ OpenFlow Manager เช่น สั่งให้ Packet ที่เข้ามายัง Port 1 ของ Switch ถูก Drop ทิ้ง โดยดำเนินการตามภาพที่ 9 ดังนี้ คลิกที่ Flow management จากนั้นคลิกที่รูปดินสอภายใต้หัวข้อ Flows จะได้ตามภาพที่ 10



ภาพที่ 9 การ set ค่าของ flow table เพื่อควบคุมการทำงานของ switch บน Cisco OpenFlow Manager (OFM) App

ที่มา : CiscoDevNet (2014)

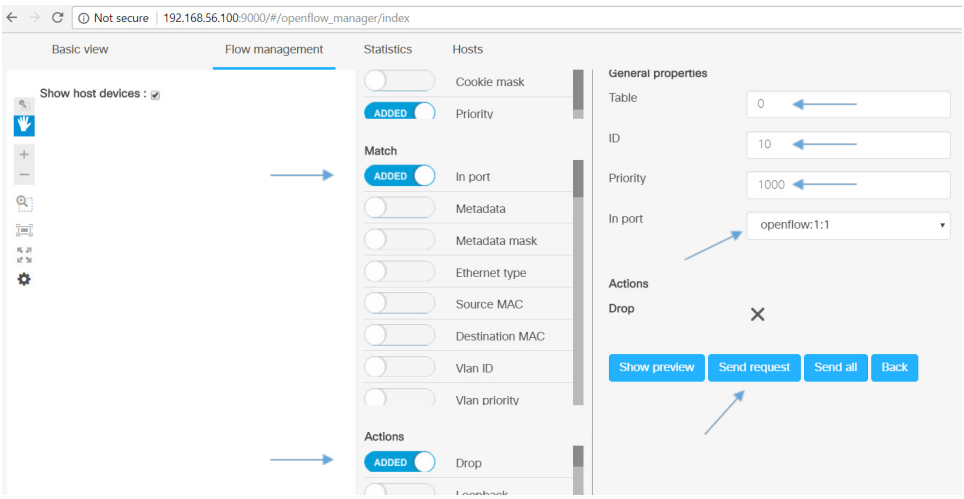
จากนั้นในภาพที่ 10 ที่ช่อง Device ให้เลือก Device : Openflow:1 เพื่อกำหนดการทำงานให้กับ Switch (ในที่นี้ก็คือ Openflow : 1) ก็จะได้ตามภาพที่ 11 เพื่อกำหนดค่าพารามิเตอร์ต่าง ๆ ของ Switch ต่อไป



ภาพที่ 10 เลือก device เป็น openflow:1 เพื่อควบคุมการทำงานของ switch ตามที่ต้องการ

ที่มา : CiscoDevNet (2014)

โดยตั้งค่าพารามิเตอร์ของ Openflow:1 หรือ Switch ตามภาพที่ 11 (Teeter, 2015) เช่น table : 0, ID : 10, Priority : 1000, match : In port, in port : openflow 1 : 1, actions:drop จากนั้น คลิกที่ Send request เพื่อส่งคำสั่งไปยัง OpenDaylight



ภาพที่ 11 การตั้งค่าเงื่อนไขเพื่อควบคุมการทำงานของ switch บน Cisco OpenFlow Manager (OFM) App

ที่มา : CiscoDevNet (2014)

ที่ Mininet VM หลังจากสั่งการทำงานเพิ่มเติม ที่ OpenFlow Manager โดยสั่งให้ Packet ที่เข้ามายัง Port 1 ของ Switch ถูก Drop ทิ้ง และใช้คำสั่ง Pingall ที่ Mininet จะได้

mininet> pingall

*** Ping: testing ping reachability

h1 -> X หมายถึง host1 (h1) ไม่สามารถติดต่อ host2 (h2) ได้

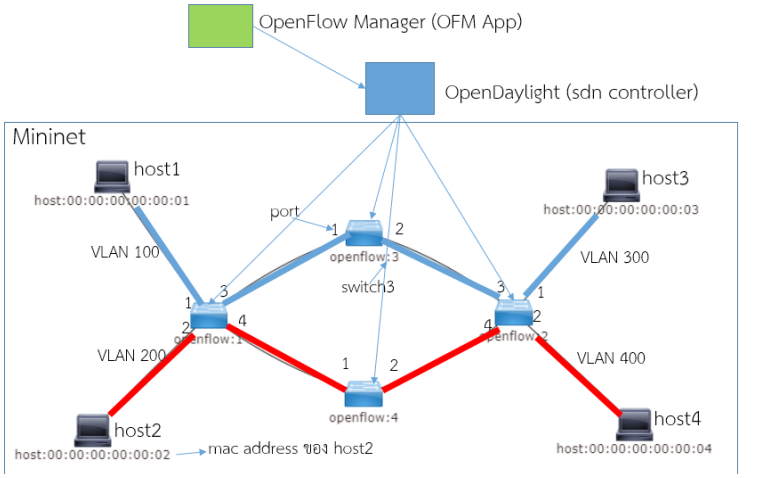
h2 -> X หมายถึง host2 (h2) ไม่สามารถติดต่อ host1 (h1) ได้

*** Results: 100% dropped (0/2 received)

ก็จะได้ผลเป็นไปตามที่ต้องการคือ การเชื่อมต่อระหว่าง h1 กับ h2 ผ่าน switch ไม่สำเร็จเนื่องจาก Packet ถูก Drop ทิ้งเมื่อถูกส่งเข้ามาที่ Port 1 ของ Switch

5. ทดสอบการสร้างวงจรเช่าเสมือนเลเยอร์ 2 (Virtual Private LAN Service หรือ VPLS) แบบ Manual

ได้มีการนำเอา SDN มาใช้ใน VPLS (Prete, 2017) โดย VPLS เป็นบริการหนึ่งที่ผู้ให้บริการสร้างวงจรเช่าเสมือนเลเยอร์ 2 แก่ลูกค้ารายใหญ่ที่มีสำนักงานอยู่ทั่วประเทศ ในที่นี้เราจะจำลองด้วยลูกค้าจำนวน 2 บริษัท โดยจำแนกลูกค้าแต่ละบริษัทด้วยสีคือ ลูกค้าที่อยู่ในสีเดียวกันจะสามารถติดต่อกันได้ ส่วนลูกค้าที่อยู่คนละสีไม่สามารถติดต่อกันได้ ในส่วนของอุปกรณ์โครงข่ายเราใช้ Mininet สร้าง Switches 4 ตัว และ Hosts 4 ตัว ที่จะถูกดูแลและควบคุมโดยส่วนของ Control หรือ SDN controller ซึ่งยังคงเป็น OpenDaylight และส่วนของแอปพลิเคชันจะใช้ OpenFlow Manager ซึ่งรูปการเชื่อมต่อที่กล่าวมาสามารถแสดงดังภาพที่ 12



ภาพที่ 12 การเชื่อมต่อของอุปกรณ์ ในโครงข่ายที่ถูกสร้างขึ้น ตามคำสั่ง mn เรียกดูด้วย OpenDaylight

ที่มา : OpenDaylight (2013)

ในที่นี้เราสมมุติมีลูกค้า 2 บริษัทโดยบริษัทแรกมีสมาชิกคือ host1 และ host3 และบริษัทที่สองมีสมาชิกคือ host2 และ host4 ในที่นี้เราจะสร้างวงจรเช่าเสมือนเลเยอร์ 2 (VPLS) แบบ Manual ผ่านแอปพลิเคชันชื่อ OpenFlow Manager (OFM) เพื่อกำหนดให้ switch1 switch2 switch3 และ switch4 ทำงานตามที่เราต้องการ (วิเชียร, 2562) คือทำให้สมาชิกของลูกค้าบริษัทเดียวกันสามารถติดต่อกันได้ เช่น host1 ติดต่อกับ host3 ได้และ host2 ติดต่อกับ host4 ได้ แต่สมาชิกของลูกค้าคนละบริษัทไม่สามารถติดต่อกันได้ เช่น host1 หรือ host3 ไม่สามารถติดต่อกับ host2 หรือ host4 ได้ในทำนองเดียวกัน host2 หรือ host4 ไม่สามารถติดต่อกับ host1 หรือ host3 ได้ โดยเราใช้คำสั่ง pingall (อยู่ในการทำงานภายใต้คำสั่ง mn) เพื่อทดสอบว่า Host แต่ละตัว สามารถติดต่อ Host ตัวอื่น ๆ ที่เหลือได้หรือไม่โดยมีผลการทดสอบดังนี้

mininet> pingall

*** Ping: testing ping reachability

h1 -> X h3 X หมายถึง host1 (h1) สามารถติดต่อ host3 (h3) แต่ไม่สามารถติดต่อ host2 (h2) และ host4 (h4) ได้

h2 -> X X h4 หมายถึง host2 (h2) สามารถติดต่อ host4 (h4) แต่ไม่สามารถติดต่อ host1 (h1) และ host3 (h3) ได้

h3 -> h1 X X หมายถึง host3 (h3) สามารถติดต่อ host1 (h1) แต่ไม่สามารถติดต่อ host2 (h2) และ host4 (h4) ได้

h4 -> X h2 X หมายถึง host4 (h4) สามารถติดต่อ host2 (h2) แต่ไม่สามารถติดต่อ host1 (h1) และ host3 (h3 ได้

*** Results: 66% dropped (4/12 received)

จะได้ผลเป็นไปตามที่ต้องการนั่นคือ สมาชิกของบริษัทเดียวกันสามารถติดต่อกันได้ และสมาชิกที่อยู่คนละบริษัทไม่สามารถติดต่อกันได้ นอกจากนี้เราสามารถตรวจสอบการทำงานว่าข่าวสารที่ออกมาจาก host1 สามารถถูกส่งต่อไปยัง host3 ได้และมีการแปะ VLAN ถูกต้องโดยการใช้ Wireshark วัดการทำงาน (วิเชียร, 2562) ดังภาพที่ 12 เริ่มจาก host1 ส่งข่าวสารที่มีการแปะ (tag) VLAN100 เข้าที่ port1 ของ switch1 และถูกพาออกที่ port3 ของ switch1 โดยยังคงแปะด้วย VLAN100 จากนั้นถูกส่งต่อมายัง port1 ของ switch3 และถูกพาออกที่ port2 ของ switch3 โดยยังคงแปะด้วย VLAN100 จากนั้นถูกส่งต่อมายัง port3 ของ switch2 โดยถูก switch3 แก้ไขค่า VLAN โดยเขียน VLAN 300 (ใหม่) ลงไปแทน VLAN100 (เดิม) และพาออกที่ port1 ของ switch2 เพื่อส่งต่อไปยัง host3

6. บทสรุป

บทความนี้ได้แนะนำและสาธิตแนวคิดการใช้งาน SDN อย่างง่าย ๆ 2 ตัวอย่าง โดยตัวอย่างแรกนำเสนอด้วยองค์ประกอบของ SDN จากทั้ง 3 ส่วน ได้แก่ Infrastructure ที่ใช้ Mininet จำลองสร้างอุปกรณ์บนโครงข่ายซึ่งในที่นี้เราสร้าง Switch 1 ตัว และ host 2 ตัว ส่วน Control เราใช้ OpenDaylight เป็น SDN controller และ ส่วนของแอปพลิเคชัน ที่ใช้ OpenFlow Manager เพื่อสั่งให้อุปกรณ์บนโครงข่ายทำงานตามที่ต้องการ (ในที่นี้คือการสั่ง Drop packet ทิ้งเมื่อ Packet ถูกส่งเข้ามาที่ port 1 ของ switch) โดยการติดต่อระหว่างส่วนของ Application Openflow Manager กับ SDN controller OpenDaylight นั้นเป็นไปตามมาตรฐาน REST API และการติดต่อระหว่าง Switch บน Mininet กับ SDN controller OpenDaylLight ใช้มาตรฐานการเชื่อมต่อ OpenFlow โดยผู้เขียนพบว่าด้วยเทคโนโลยีในปัจจุบันทำให้เราสามารถสร้างการทดสอบการใช้งาน SDN อย่างง่าย ๆ ได้ด้วยตัวเองเองบน โน้ตบุ๊กคอมพิวเตอร์ (ควรมีหน่วยความจำ RAM อย่างน้อย 8GB CPU INTEL Core i5 และ Hard disk 200 GB) ที่มีอยู่ทั่วไป และซอฟต์แวร์ทั้ง Mininet และ OpenDaylight ที่ต่างก็เป็นซอฟต์แวร์โอเพนซอร์ส (Open Source Software) ยกเว้น OpenFlow Manager ที่เป็นซอฟต์แวร์ของบริษัท Cisco แต่ก็เปิดโอกาสให้ทดสอบการใช้งานโดยไม่มีค่าใช้จ่าย สำหรับอีกตัวอย่างต้องการนำเสนอการประยุกต์เอา SDN มาร่วมให้บริการ VPLS แก่ลูกค้ารายใหญ่ที่มีสำนักงานอยู่ทั่วประเทศและจะพบว่าด้วยการนำ SDN เข้ามาควบคุมจะทำให้ผู้ให้บริการสามารถเลือกใช้อุปกรณ์ switch ได้อย่างแอปพลิเคชันที่สามารถลดข้อผิดพลาดจากผู้ปฏิบัติงาน ดังนั้น บทความนี้จึงเหมาะสำหรับผู้ที่เริ่มต้นสนใจเทคโนโลยี SDN ซึ่งได้มีการนำ SDN มาใช้เป็นส่วนหนึ่งของโครงข่ายพื้นฐานของโทรศัพท์เคลื่อนที่ 5G (5G Platform) โดยบริษัท Ericson (Ericsson, 2015) และ Huawei (Huawei, 2017) สำหรับในประเทศไทยการนำโครงข่าย 5G มาให้บริการก็ได้รับการสนับสนุนโดย กสทช.

7. ข้อเสนอแนะ

ในบทความนี้ผู้เขียนขอเสนอแนะแก่ผู้สนใจที่ศึกษาเทคโนโลยี SDN เพื่อนำไปพัฒนาความรู้เพิ่มเติมดังนี้

1. พัฒนาซอฟต์แวร์ประยุกต์มาทดแทน OpenFlow Manager ของบริษัท Cisco ซึ่งเป็นแอปพลิเคชันที่ใช้ควบคุมการทำงานของ Switches และยังใช้เพื่อศึกษาการทำงานของมาตรฐานการรับส่งข้อมูล OpenFlow ที่เกิดขึ้นระหว่าง SDN controller กับ Switches (Open switch)

2. พัฒนาแอปพลิเคชันเพื่อให้การสร้างบริการวงจรเช่าเสมือนเลเยอร์ 2 (VPLS) สามารถสั่งให้ Switches ทำงานแบบอัตโนมัติ (Prete, 2017) ทั้งนี้เนื่องจากในบทความนี้ผู้เขียนได้นำเสนอการสร้างบริการ VPLS แบบ Manual นั่นคือ เราจะต้องเข้าไปบันทึกการทำงานของ Switch แต่ละตัวด้วยตัวเองทุกขั้นตอนเพื่อให้ Switches ทำงานตามที่เราต้องการ

3. พัฒนาแอปพลิเคชันเพื่อทำหน้าที่ในด้านต่าง ๆ เช่น ดูแลความปลอดภัยให้กับอุปกรณ์ในโครงข่าย สร้างวงจรเช่าแบบส่วนตัว (Virtual Private Network (VPN)) เพื่อให้บริการแก่ลูกค้ารายใหญ่บนโครงข่ายที่มีใช้งานในปัจจุบัน เช่น Internet 4G และโครงข่าย Multiprotocol Label Switching (MPLS) เป็นต้น

บรรณานุกรม

วิเชียร ปรีดาลัมพะบุตร. (2559). Lab การติดตั้ง Virtual Ubuntu server บน VirtualBox, เอกสารประกอบการบรรยายหัวข้อการใช้งาน streaming, FTP, และ DNS Server บน โครงข่าย IPv4 และ IPv6, สถาบันวิชาการทีโอที, ระบบออนไลน์ สืบค้นจาก: https://testwich.s3-ap-southeast-1.amazonaws.com/sdn_doc/Install_vm_ubuntu_1404_02.pdf เมื่อวันที่ค้นข้อมูล 22 กรกฎาคม 2562.

วิเชียร ปรีดาลัมพะบุตร. (2559). Lab ศึกษาการทำงานของ Software-defined networking (SDN) และ OpenFlow ด้วย Mininet, เอกสารประกอบการบรรยายหลักสูตรแนะนำเชิงปฏิบัติการ SDN & NFV, สถาบันวิชาการทีโอที, ระบบออนไลน์ สืบค้นจาก: https://testwich.s3-ap-southeast-1.amazonaws.com/sdn_doc/Lab_SDN_mininet_July_62.pdf เมื่อวันที่ค้นข้อมูล 19 กรกฎาคม 2562.

วิเชียร ปรีดาลัมพะบุตร. (2561). แนะนำ SDN และ OpenFlow, เอกสารประกอบการบรรยายหลักสูตรแนะนำเชิงปฏิบัติการ SDN & NFV, สถาบันวิชาการทีโอที, ระบบออนไลน์ สืบค้นจาก: https://testwich.s3-ap-southeast-1.amazonaws.com/sdn_doc/SDN_Intro_Openflow_Sep61.pdf เมื่อวันที่ค้นข้อมูล 19 กรกฎาคม 2562.

วิเชียร ปรีดาลัมพะบุตร. (2562). Lab ทดสอบการสร้างวงจรเช่าเสมือนเลเยอร์ 2 (Virtual Private LAN Service หรือ VPLS) แบบ Manual, เอกสารประกอบการบรรยายหลักสูตรแนะนำเชิงปฏิบัติการ SDN & NFV, สถาบันวิชาการทีโอที, ระบบออนไลน์ สืบค้นจาก: https://testwich.s3-ap-southeast-1.amazonaws.com/sdn_doc/Lab_VPLS_ODL_Sep_62.pdf เมื่อวันที่ค้นข้อมูล 25 กันยายน 2562.

AT&T. (2019). Bridging 5G and SD-WAN with VMware SD-WAN by VeloCloud for a New Layer of Control at the Edge. Retrieved from https://about.att.com/story/2019/att_5g_sdwan.html.

AT&T. (2019). Vision Alignment Challenge Technology Survey, AT&T Domain 2.0 Vision White Paper November 2013. Retrieved from https://www.att.com/ Common/about_us/pdf/ AT&T%20 Domain%202.0%20Vision%20White%20Paper.pdf.

CiscoDevNet. (2014). OpenDaylight Openflow Manager (OFM) App. Retrieved from https://github.com/CiscoDevNet/OpenDaylight-Openflow-App.

Donovan, J. (2014). Building the network of the future. Retrieved June 7, 2019 from https://www.att.com/Investor/ATT_Annual/2014/att_introduces_new_concepts_for_telecom_network.html.

Ericsson. (2015). 5G transport network – demonstrating elastic mobile broadband using SDN. 2015. Retrieved from https://www.youtube.com/watch?v=1BA0qW14MAU.

Huawei. (2017). Meeting 5G Era Requirements with Huawei's 5G Ready Multi-wave. Retrieved from https://www.huawei.com/en/press-events/events/ubbf2017/5g-multi-wave.

Lantz, B. (2017). Mininet VM Images, Mininet 2.2.2 on Ubuntu 14.04 LTS - 32 bit. Retrieved June7, 2019 from https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-i386.zip.

Lantz, B. & Heller, B. (2010). Mininet, An Instant Virtual Network on your Laptop, Mininet homepage, Retrieved June 7, 2019 from http://mininet.org/.

Levy, S. (2012) “Going with the Flow: Google’s Secret Switch to the Next Wave of Networking”, Wired.

Linkletter, B. (2016). Using the OpenDaylight SDN Controller with the Mininet Network Emulator Retrieved June 7, 2019 from http://www.brianlinkletter.com/using-the-opendaylight-sdn-controller-with-the-mininet-network-emulator/.

McKeown, N., et al. (2008). OpenFlow: Enabling Innovation in Campus Networks. SIGCOMM Comput. Commun. Rev., 38(2): 69-74.

Medved, J., et al. (2016). OpenDaylight OpenFlow Manager (OFM) App. Retrieved June 7, 2019 from https://github.com/CiscoDevNet/OpenDaylight-Openflow-App.

Open Networking Foundation. (2011). Retrieved from https://www.opennetworking.org.

OpenDaylight. (2013). Retrieved from https://www.opendaylight.org/.

OpenDaylight. (2016). Downloads, Beryllium 0.4.0. Retrieved June 7, 2019 from https://nexus.opendaylight.org/content/repositories/opendaylight.release/org/opendaylight/integration/distribution-karaf/0.4.0-Beryllium/.

Prete, L. (2017). Virtual Private LAN Service – VPLS. Retrieved June 7, 2019, from https://wiki.onosproject.org/display/ONOS/Virtual+Private+LAN+Service+-+VPLS

Teeter, J. (2015). Exploring the Cisco OpenFlow Manager. Retrieved June 7, 2019 from https://www.youtube.com/watch?v=X7yIummgAqc.

Ubuntu. (2014). Server install image, Ubuntu 14.04.6 LTS (Trusty Tahr). Retrieved June 7, 2019 from http://releases.ubuntu.com/14.04/ubuntu-14.04.6-server-amd64.iso.

VirtualBox. (2007). Retrieved June 7, 2019 from https://www.virtualbox.org/.