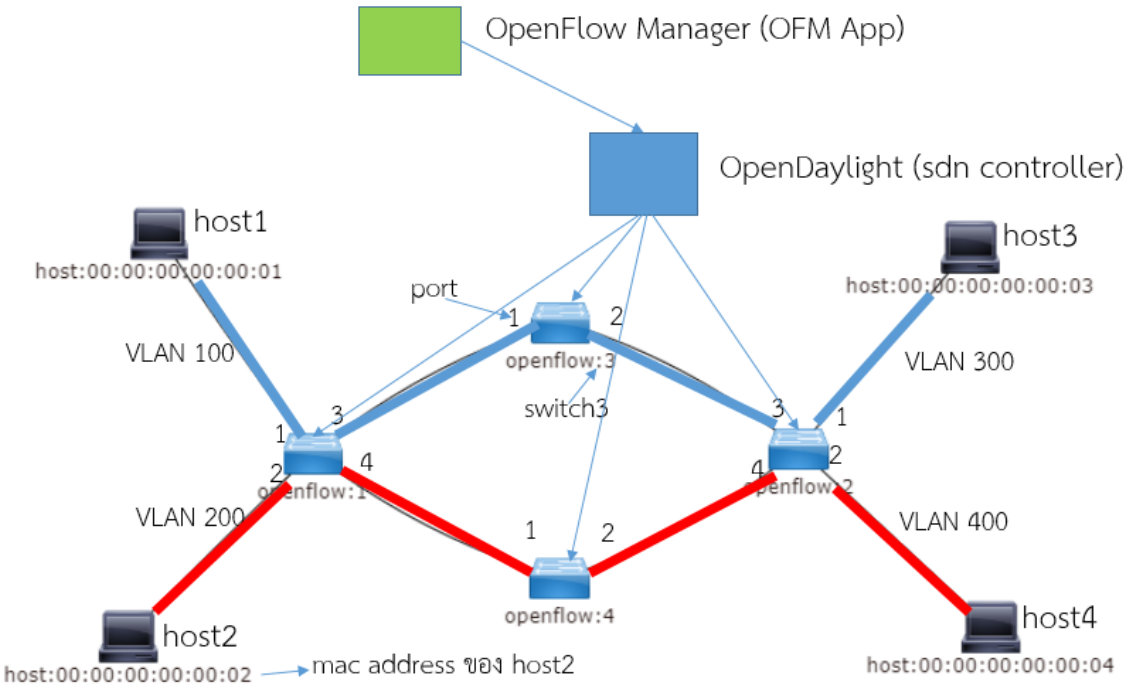
ทดสอบการสร้างวงจรเช่าเสมือนเลเยอร์ 2 (Virtual Private LAN Service หรือ VPLS) แบบ Manual

วิเชียร ปรีดาลัมพะบุตร สถาบันวิชาการทีโอที wpreeda@tot.co.th กันยายน 2562

VPLS เป็นบริการหนึ่งที่ผู้ให้บริการสร้างวงจรเช่าเสมือนเลเยอร์ 2 แก่ลูกค้ารายใหญ่ที่มีสำนักงานอยู่ทั่วประเทศ ในที่นี้เราจะจำลองด้วยลูกค้าจำนวน 2 ราย โดยจำแนกลูกค้าแต่ละรายด้วยสี นั่นคือลูกค้าที่อยู่ในสีเดียวกันจะสามารถติดต่อกันได้ ส่วนลูกค้าที่อยู่คนละสีก็ไม่สามารถติดต่อกันได้ ในส่วนของอุปกรณ์โครงข่ายเราใช้ Mininet VM สร้าง switches 4 ตัว และ hosts 4 ตัว ตามภาพที่ 12 สำหรับส่วนของ control หรือ sdn controller ก็ยังคงเป็น OpenDaylight VM และ ส่วนของ application จะใช้ OpenFlow Manager



ภาพที่ 1 การเชื่อมต่อของ อุปกรณ์ ในโครงข่ายที่ถูกสร้างขึ้น ตามคำสั่ง mn เรียกดูด้วย OpenDaylight
ที่มา : The OpenDaylight Project (2013)

สำหรับการสร้างโครงข่ายที่ใช้ทดสอบจะใช้ mininet VM ด้วยคำสั่ง

mininet@mininet-vm:~$ sudo mn --custom ~/mininet/custom/4sw_4host.py --topo=mytopo --mac --switch=ovs,protocols=OpenFlow13 --controller=remote,ip=192.168.56.200

โดยที่ไฟล์ 4sw_4host.py ใช้สร้างโครงข่ายสำหรับทดสอบ ดังแสดงได้ดังนี้

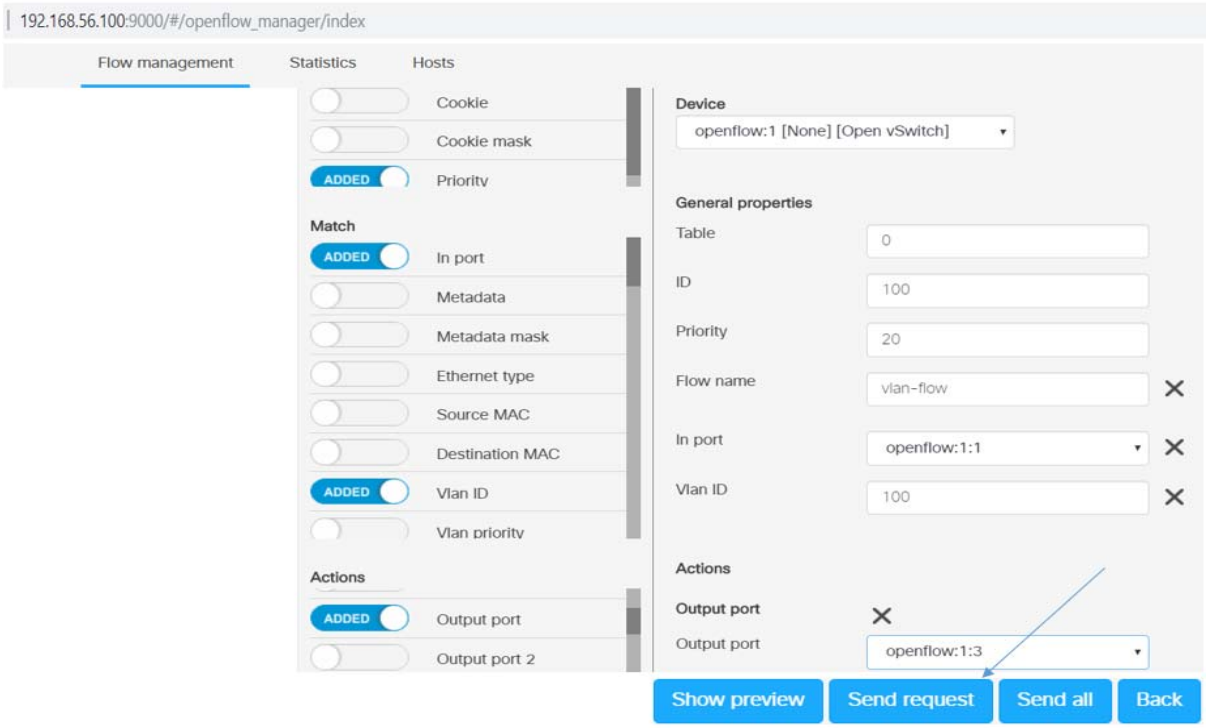
```
from mininet.topo import Topo
```

```
class MyTopo( Topo ):
  def __init__( self ):
    # Initialize topology
    Topo.__init__( self )
    # Add hosts and switches
    host1 = self.addHost( 'h1' )
    host2 = self.addHost( 'h2' )
    host3 = self.addHost( 'h3' )
    host4 = self.addHost( 'h4' )
    switch1 = self.addSwitch( 's1' )
    switch2 = self.addSwitch( 's2' )
    switch3 = self.addSwitch( 's3' )
    switch4 = self.addSwitch( 's4' )
    # Add links
    self.addLink( host1, switch1 )
    self.addLink( host2, switch1 )
    self.addLink( host3, switch2 )
    self.addLink( host4, switch2 )
    self.addLink( switch1, switch3 )
    self.addLink( switch1, switch4 )
    self.addLink( switch2, switch3 )
    self.addLink( switch2, switch4 )
topos = { 'mytopo': ( lambda: MyTopo() ) }
```

เพื่อจัดการสร้างการเชื่อมต่อให้ host1 ติดต่อกับ host3 ได้สำเร็จ เราจะส่งคำสั่งควบคุมจาก OpenFlow Manager Cisco (Application) ไปยัง OpenDaylight (sdn controller) ด้วยมาตรฐานการรับส่งข้อมูลชื่อ REST API จากนั้น OpenDaylight จะส่งคำสั่งควบคุมไปยัง switch ด้วยมาตรฐานการรับส่งข้อมูลชื่อ openflow เพื่อให้ switch1 (openflow:1), switch2 (openflow:2), และ switch3 (openflow:3) ทำงานดังนี้

งานที่ switch1 (openflow:1) จะต้องทำมีดังนี้

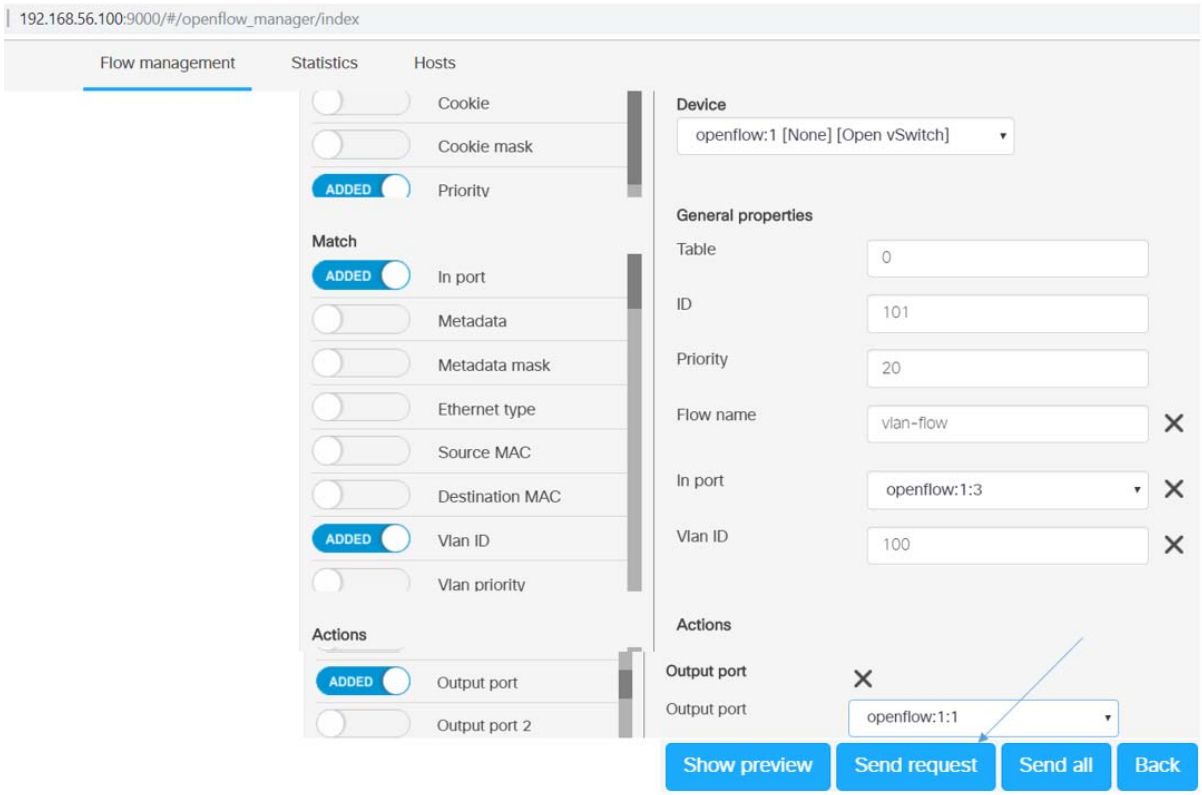
1. ข่าวสารที่ออกมาจาก host1 จะมีการแปะ(tag) VLAN 100 มาเข้าที่ port1 ของ switch1 (openflow:1:1) จะถูกพาออกที่ port3 ของ switch1 (openflow:1:3) และข่าวสารยังคงมีการแปะ VLAN100 ไปด้วย (ดูการตั้งค่าเงื่อนไขการ match และดำเนินการตามภาพที่ 13)



ภาพที่ 2 การตั้งค่าเงื่อนไขเพื่อควบคุมการทำงานของ switch1 บน Cisco OpenFlow Manager (OFM) App ที่มา : https://github.com/CiscoDevNet/OpenDaylight-Openflow-App (2014)

โดย ตั้งค่าพารามิเตอร์ของ openflow:1 หรือ switch1 ตามภาพที่ 13 เช่น table:0, ID:100, Priority:20,Flow name: vlan-flow match: In port, in port:openflow 1:1, vlan id:100, action:output, Output port:openflow:1:3 จากนั้น คลิกที่ send request เพื่อส่งคำสั่งไปยัง OpenDayLight โดยการส่งข้อมูลตรงนี้จะเป็นไปตามมาตรฐานการเชื่อมต่อ REST API

2. ข่าวสารที่ออกมาจากport1 ของswitch3 (openflow:3:1)ซึ่งจะมีการแปะ(tag) VLAN 100 มาด้วย เข้ามายัง port3 ของ switch1 (openflow:1:3) จะถูกพาออกที่ port1 ของ switch1 (openflow:1:1) โดยข่าวสารมีการแปะ VLAN100 ไปยัง host1



ภาพที่ 3 การตั้งค่าเงื่อนไขเพื่อควบคุมการทำงานของ switch1 บน Cisco OpenFlow Manager (OFM) App

ที่มา : https://github.com/CiscoDevNet/OpenDaylight-Openflow-App (2014)

โดย ตั้งค่าพารามิเตอร์ของ openflow:1 หรือ switch1 ตามภาพที่ 14 เช่น table:0, ID:101, Priority:20,Flow name: vlan-flow match: In port, in port:openflow 1:3, vlan id:100, action:output port, Output port:openflow:1:1 จากนั้น คลิกที่ send request เพื่อส่งคำสั่งไปยัง OpenDayLight โดยการส่งข้อมูลตรงนี้จะเป็นไปตามมาตรฐานการเชื่อมต่อ REST API

ตรวจสอบ ตารางการทำงาน ของ switch1 (openflow:1)

mininet> sh ovs-ofctl -O openflow13 dump-flows s1

OFPST_FLOW reply (OF1.3) (xid=0x2):

cookie=0x2b00000000000009, duration=3308.826s, table=0, n_packets=1326, n_bytes=112710, priority=100,dl_type=0x88cc actions=CONTROLLER:65535

cookie=0x2b00000000000006, duration=3308.826s, table=0, n_packets=18, n_bytes=1232, priority=0 actions=drop

cookie=0x0, duration=3308.826s, table=0, n_packets=18, n_bytes=1212, priority=20,in_port=3,dl_vlan=100 actions=output:1

cookie=0x0, duration=348.956s, table=0, n_packets=13, n_bytes=742, priority=25,in_port=3,dl_vlan=200 actions=output:2

cookie=0x0, duration=2820.139s, table=0, n_packets=87, n_bytes=4266,
priority=20,in_port=2,dl_vlan=200 actions=output:3
cookie=0x0, duration=3308.826s, table=0, n_packets=18, n_bytes=1212,
priority=20,in_port=1,dl_vlan=100 actions=output:3
cookie=0x2b0000000000000b, duration=3302.844s, table=0, n_packets=15, n_bytes=1162,
priority=2,in_port=1 actions=output:2,output:4,CONTROLLER:65535
cookie=0x2b0000000000000c, duration=3302.844s, table=0, n_packets=74, n_bytes=4552,
priority=2,in_port=4 actions=output:2,output:1
cookie=0x2b0000000000000a, duration=3302.844s, table=0, n_packets=14, n_bytes=1064,
priority=2,in_port=2 actions=output:1,output:4,CONTROLLER:65535

งานที่ switch3 (openflow:3) จะต้องทำมีดังนี้ (ทำในทำนองเดียวกับภาพที่ 12 และ 13)

1. ข่าวสารที่ออกมาจาก port3 ของ switch1 (openflow:1:3) มีการแปะ(tag) VLAN 100 จะมาเข้าที่ port1 ของ switch3 (openflow:3:1) จะถูกพาออกที่ port2 ของ switch3 (openflow3:2) และข่าวสารยังคงมีการแปะ VLAN100 ไปด้วย
2. ข่าวสารที่เข้ามายัง port2 ของ switch3 (openflow:3:2) มีการแปะ(tag) VLAN 100 มาด้วยจะถูกพาออกที่ port1 ของ switch3 (openflow:3:1) และข่าวสารยังคงมีการแปะ VLAN100 ไปด้วย

งานที่ switch2 (openflow:2) จะต้องทำมีดังนี้ (ทำในทำนองเดียวกับภาพที่ 12 และ 13)

1. ข่าวสารที่ออกมาจาก port2 ของ switch3 (openflow:3:2)มีการแปะ(tag) VLAN 100 จะมาเข้าที่ port3 ของ switch2 (openflow:2:3) จะถูกเขียนด้วย VLAN 300 (set vlan id) และจะถูกพาออกที่ port1 ของ switch2 (openflow2:1) โดยข่าวสารยังคงมีการแปะ VLAN 300 ไปยัง host3
2. ข่าวสารที่ออกมาจาก host3 จะมีการแปะ (tag) VLAN 300 มาเข้าที่ port1 ของ switch2 (openflow:2:1) จะถูกเขียนด้วย VLAN 100 (set vlan id) และพาออกที่ port4 ของ switch2 (openflow:2:4) โดยข่าวสารยังคงมีการแปะ VLAN 100 ไปด้วย

ที่ host1 และ host3 จะต้องสร้าง VLAN interface (นั่นคือข่าวสารที่ถูกส่งจาก host1 และhost3 จะถูกแปะ(tag) ด้วย VLAN 100 และ 300 ตามลำดับ) ด้วยการใช้คำสั่งดังนี้ (host2 และ host4 ทำในทำนองเดียวกัน)

mininet> h1 vconfig add h1-eth0 100

mininet> h3 vconfig add h3-eth0 300

อย่างไรก็ตามให้ทำการยกเลิกการกำหนด ip address ที่ interface eth0 ของ host1 และ host3 (host2 และ host4 ทำในทำนองเดียวกัน) ด้วยการใช้คำสั่งดังนี้

mininet> h1 route del -net 10.0.0.0 netmask 255.0.0.0

mininet> h3 route del -net 10.0.0.0 netmask 255.0.0.0

จากนั้นให้ทำการกำหนด ip address ที่ VLAN interface (นั่นคือข่าวสารที่ถูกส่งจาก host1 และhost3 จะถูกแปะ (tag) ด้วย VLAN 100 และ 300 ตามลำดับ) ด้วยการใช้คำสั่งดังนี้ (host2 และ host4 ทำในทำนองเดียวกัน)

mininet> h1 ifconfig h1-eth0.100 10.0.0.1 netmask 255.0.0.0 up

mininet> h3 ifconfig h3-eth0.300 10.0.0.3 netmask 255.0.0.0 up

ทำการทดสอบ ให้ host1 ติดต่อกับ host3 ด้วยคำสั่ง ping ดังนี้

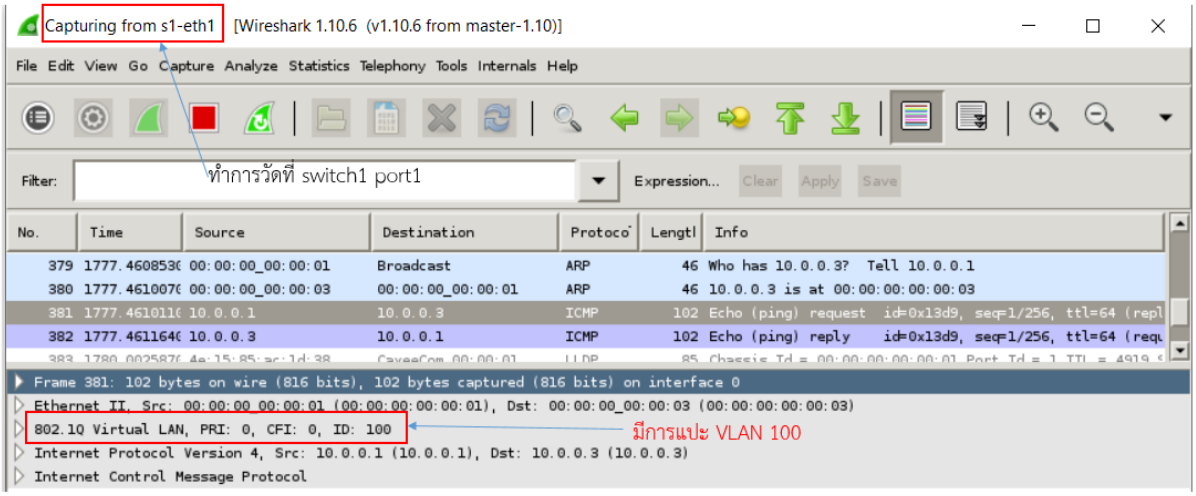
mininet> h1 ping h3 -c 1

PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.332 ms

1 packets transmitted, 1 received, 0% packet loss, time 0ms

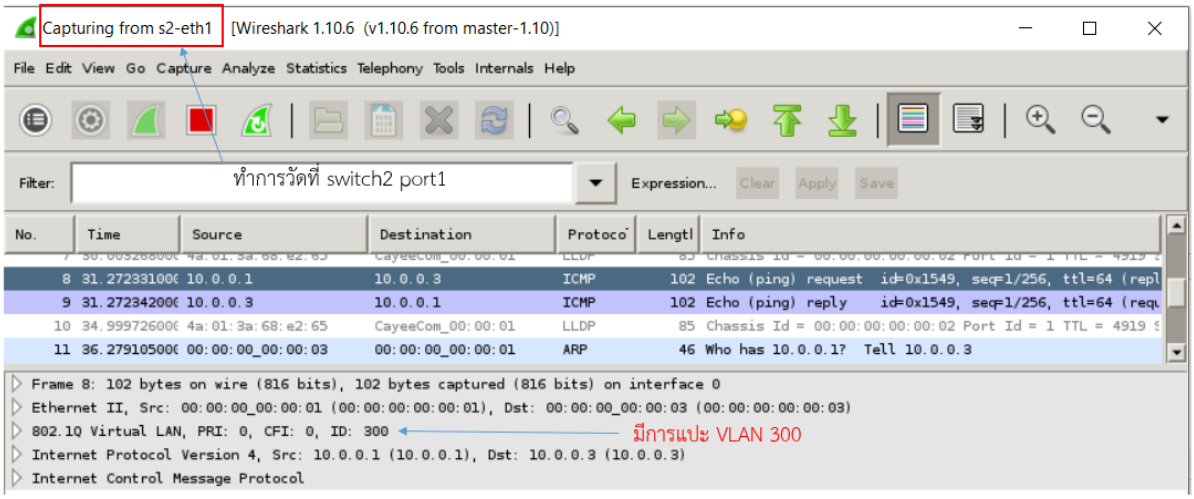
ก็พบว่า host1 สามารถติดต่อกับ host3 ได้สำเร็จ ก็เป็นไปตามวัตถุประสงค์ตามต้องการ จากนั้นทำการวัดข่าวสารที่ port1 switch1 (openflow:1:1) ด้วย wireshark (วิเชียร ตุลาคม 2559) จะได้ตามภาพที่15 เพื่อตรวจสอบว่า host1 ส่งข่าวสารออกมาและมีการแปะ VLAN 100 ออกมาหรือไม่ จากการตรวจวัดด้วย wireshark พบว่า host1 แปะ VLAN 100 ออกมาจริง



ภาพที่ 4 การตั้งค่าเงื่อนไขเพื่อควบคุมการทำงานของ switch1 บน Cisco OpenFlow Manager (OFM) App

ที่มา : https://github.com/CiscoDevNet/OpenDaylight-Openflow-App (2014)

ทำการวัดด้วย wireshark ที่ port1ของ switch2 พบว่าข่าวสารที่ออกจาก port ดังกล่าวไปยัง host3 มีการถเขียนใหม่จากเดิม VLAN 100 เป็น VLAN 300 โดย switch2 เป็นไปตามที่เราต้องการตามภาพที่ 16



ภาพที่ 5 การตั้งค่าเงื่อนไขเพื่อควบคุมการทำงานของ switch1 บน Cisco OpenFlow Manager (OFM) App
ที่มา : https://github.com/CiscoDevNet/OpenDaylight-Openflow-App (2014)

ในที่นี้เราสมมุติมีลูกค้า 2 ราย โดยบริษัทแรกมีสมาชิกคือ host1 และ host3 และ ลูกค้าของบริษัทที่สองมีสมาชิกคือ host2 และ host4 ดังนั้นผู้ให้บริการจะสร้างวงจรเช่าเสมือนเลเยอร์ 2 (VPLS) แบบ Manual ผ่าน application ชื่อ OpenFlow Manager (OFM) เพื่อกำหนดให้ switch1, switch2, switch3 และ switch4 ทำงานตามที่เราต้องการ คือ ทำให้สมาชิกของลูกค้ารายเดียวกันสามารถติดต่อกันได้ เช่น host1 ติดต่อกับ host3 ได้ และ host2 ติดต่อกับ host4 ได้ แต่สมาชิกของลูกค้าคนละบริษัทไม่สามารถติดต่อกันได้เช่น host1 ไม่สามารถติดต่อ host2 และ host4 หรือ host2 ไม่สามารถติดต่อ host1 และ host3 เป็นต้น โดยเราใช้คำสั่ง pingall (อยู่ในการทำงานภายใต้คำสั่ง mn) เพื่อทดสอบว่า host แต่ละตัว สามารถติดต่อ host ตัวอื่นๆที่เหลือได้หรือไม่ โดยมีผลการทดสอบดังนี้

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

h4 -> X h2 X

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

ก็จะได้ผลเป็นไปตามที่ต้องการนั่นคือ สมาชิกของบริษัทเดียวกันสามารถติดต่อกันได้ และสมาชิกที่อยู่คนละบริษัทไม่สามารถติดต่อกันได้