

Lab ศึกษาการทำงานของ Software-defined networking (SDN) และ OpenFlow ด้วย Mininet

วิเชียร ปรีดาลัมพะบุตร

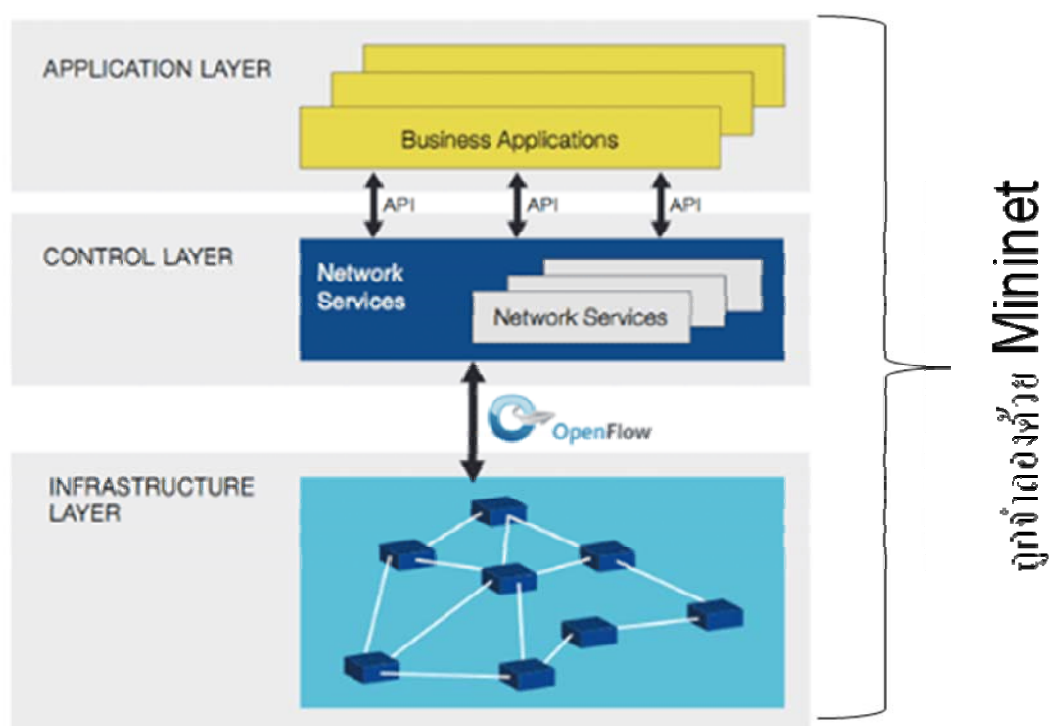
wpreeda@tot.co.th

สถาบันวิชาการทีไอที

21 ตุลาคม 2559

## บทนำ

ในการทดลองนี้ เราจะศึกษาการทำงานของ Software-defined networking (SDN), และ OpenFlow โดยเราจะใช้ simulation software ที่เรียกว่า Mininet ที่สามารถ สร้าง 3 ส่วน (layer) ได้แก่ infrastructure layer ที่ประกอบด้วย virtual host, virtual link และ virtual Switch, control layer และ application layer โดย Mininet ถูกสร้าง บน server ที่เป็น Virtual Machine หรือ VM ที่ถูกจำลองด้วย VirtualBox บน laptop



รูปแสดงการใช้ Mininet สร้าง SDN architecture

**หมายเหตุ** ในการทดลองนี้ เราใช้ Mininet สำหรับ สร้าง infrastructure layer ได้แก่ virtual switch และ virtual host ,โดยใน control layer จะใช้ ptcp และ pox controller

## วัตถุประสงค์

เพื่อทดสอบและศึกษาการทำงานของ SDN และ OpenFlow โดยใช้ mininet สร้าง 3 ส่วน (layer) ได้แก่ infrastructure layer ที่ประกอบด้วย virtual host, virtual link และ virtual Switch, control layer และ application layer บน laptop ที่ทำงานด้วย windows ทั้งนี้จะเป็นการสะดวกเพราะใช้ laptop เพียงเครื่องเดียว

## อุปกรณ์ที่ใช้ในการทดลอง

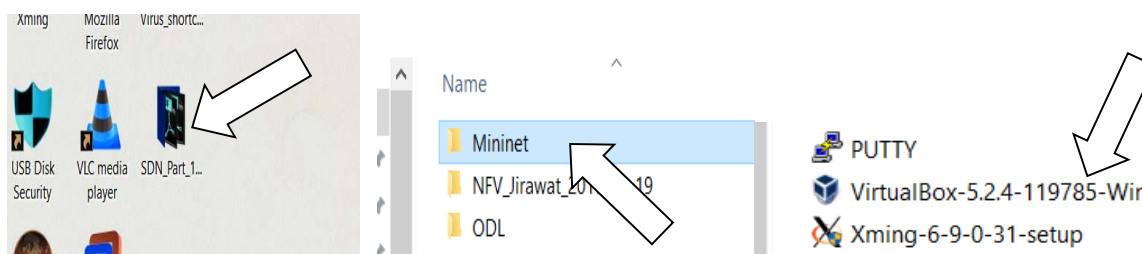
1. Laptop ที่ทำงานด้วย Windows 7
2. ใช้ VirtualBox เป็น virtual machine software
3. ใช้ Mininet VM Images ที่เป็น Mininet 2.2.0 บน Ubuntu version 14.04 -32 bit สำหรับ Windows users ที่ใช้ VirtualBox

## ขั้นตอนในการทดลอง

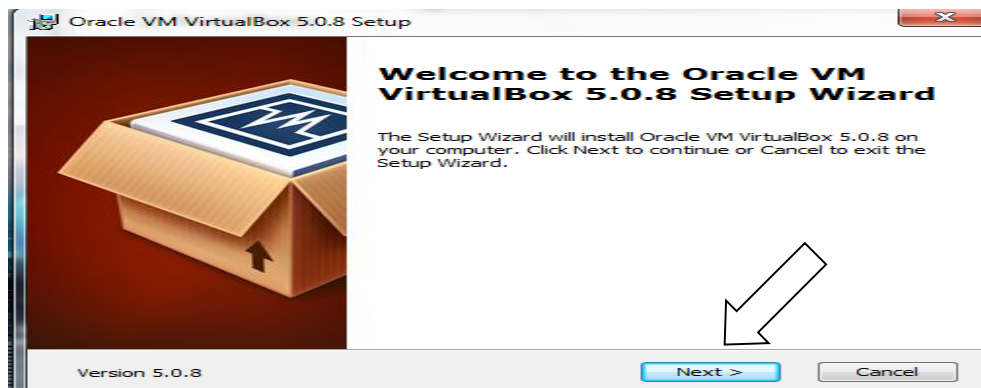
### 1. ที่ laptop ให้ติดตั้ง VirtualBox

ที่ laptop ให้ติดตั้ง VirtualBox , กรณีของ Windows ท่านสามารถ download ชุด package program ได้จาก website ของ VirtualBox ได้ที่ <https://www.virtualbox.org/> (ในที่นี้เรา download ไว้ให้ท่านแล้วที่ Desktop ชื่อ SDN Part 1 โดยทำตามขั้นตอนดังนี้:

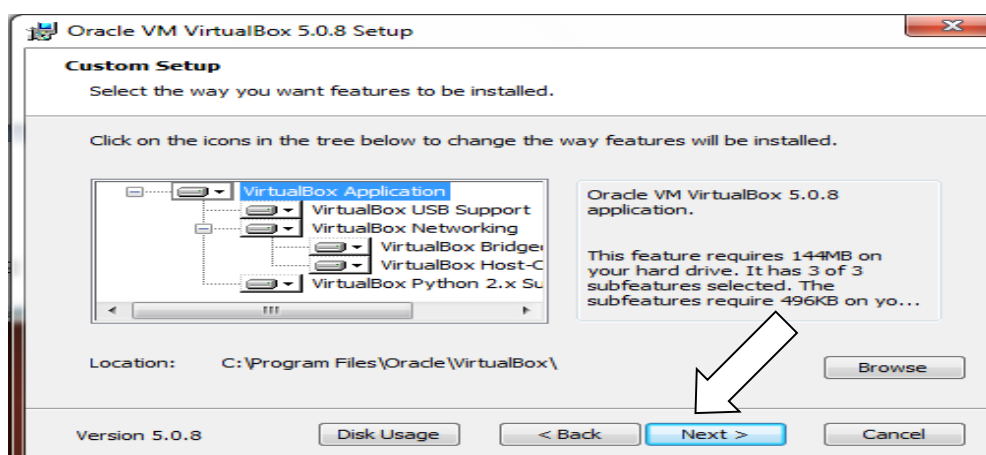
click 2 ครั้งที่ desktop ชื่อ SDN Part 1, click ที่ folder ชื่อ Mininet และ มา click 2 ครั้ง ที่ไฟล์ VirtualBox-5.2.4... เพื่อติดตั้ง



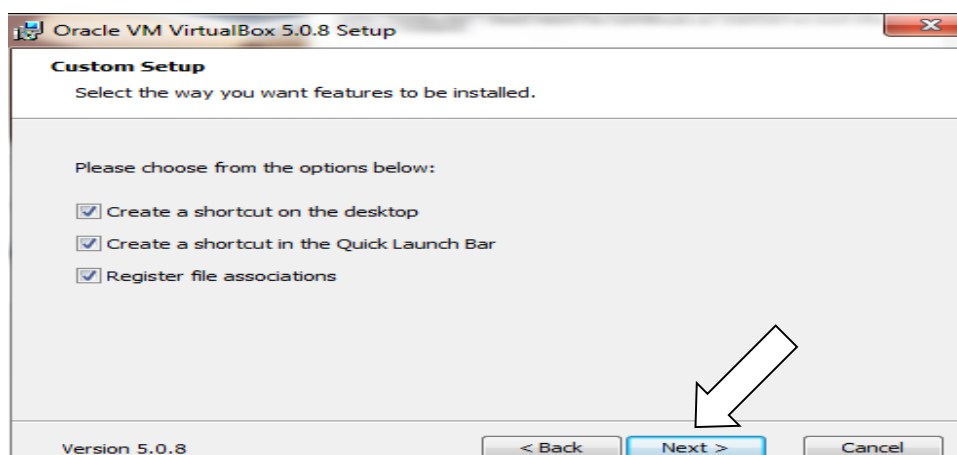
รอสักครู่นานกว่า จะได้ตามรูปข้างล่าง และให้ click ที่ next



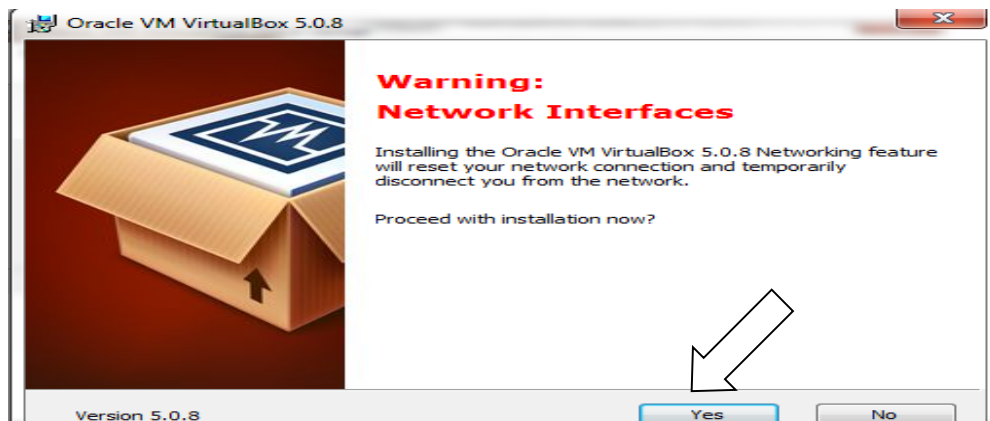
จะได้ตามรูปข้างล่าง และให้ click ที่ next



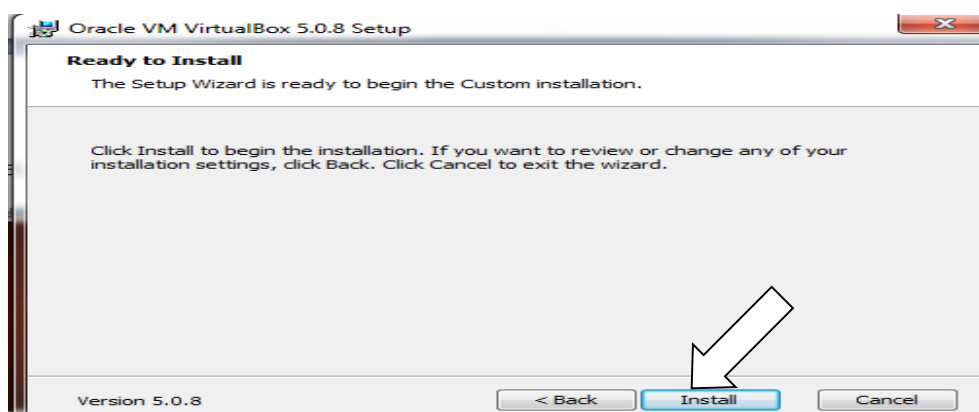
จะได้ตามรูปข้างล่าง จากนั้นให้ click ที่ next



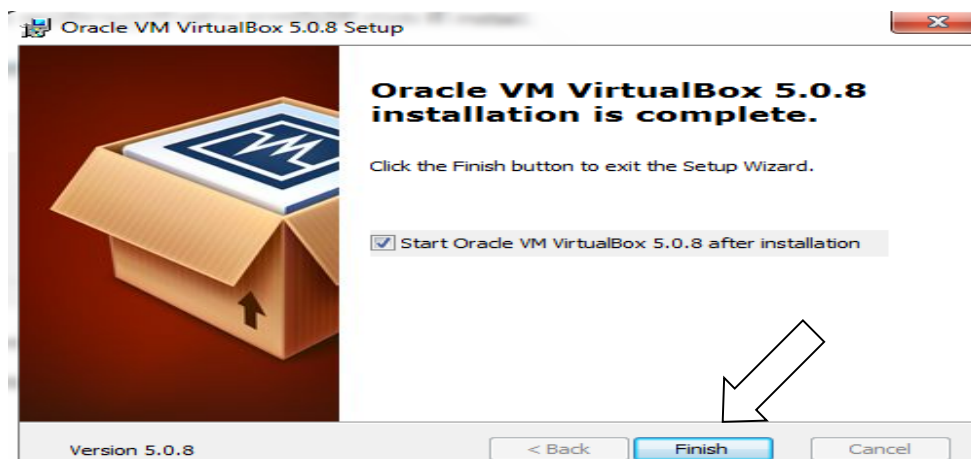
จะได้ตามรูปข้างล่าง จากนั้นให้ click ที่ yes



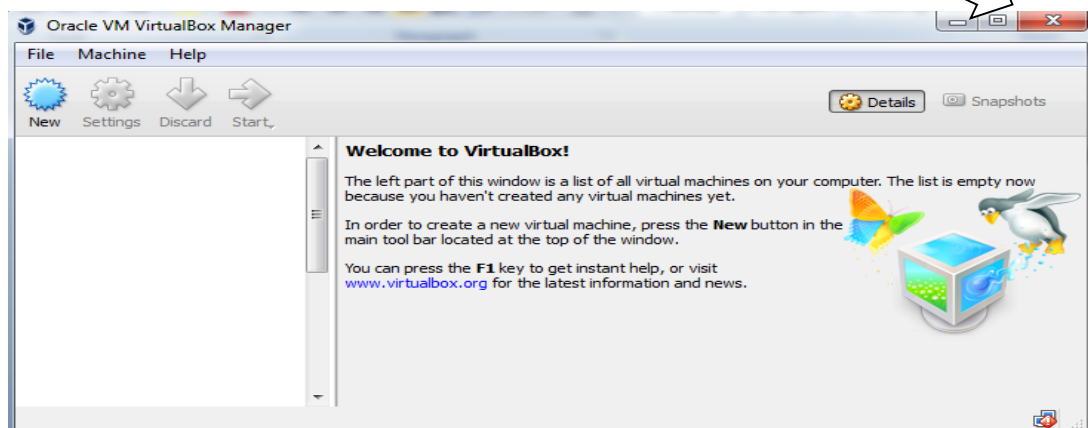
จะได้ตามรูปข้างล่าง จากนั้นให้ click ที่ install



จากนั้นให้รอสักครู่ และ ถ้ามีการถามจาก windows เพื่อขอติดตั้งอุปกรณ์ต่างๆ เช่น USB ก็ให้ click ที่ install เพื่ออนุญาต จากนั้นจะได้ตามหน้าจอข้างล่าง และ click ที่ finish



จากนั้นจะได้หน้าจอตามรูปข้างล่าง

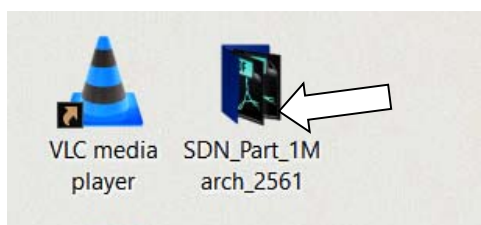


จากนั้นปิดการใช้งาน โดย click ที่ เครื่องหมาย x มุมบนขวา

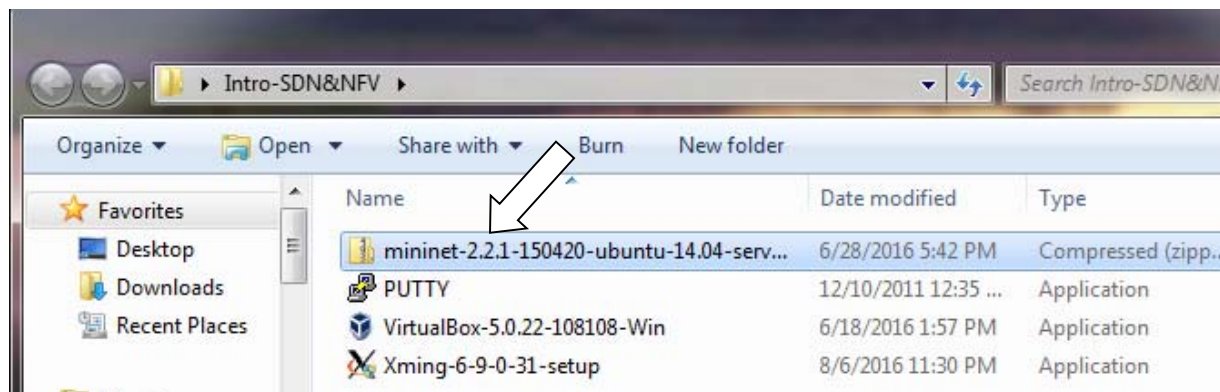
## 2. ติดตั้ง Mininet

การติดตั้งใช้งาน Mininet ทำได้หลายวิธี ในที่นี้จะขอแนะนำ วิธีที่ง่ายที่สุดเรียกว่า Mininet VM (virtual machine) โดยทำการ download Mininet VM image จาก website <http://mininet.org/download/>, ในที่นี้ได้ download ไฟล์ไว้ให้แล้วชื่อ mininet-2.2.0-150106-ubuntu-14.04-server-i386.zip  $\approx$  800 MB อยู่ใน directory ชื่อ Intro SDN & NFV บน Desktop)

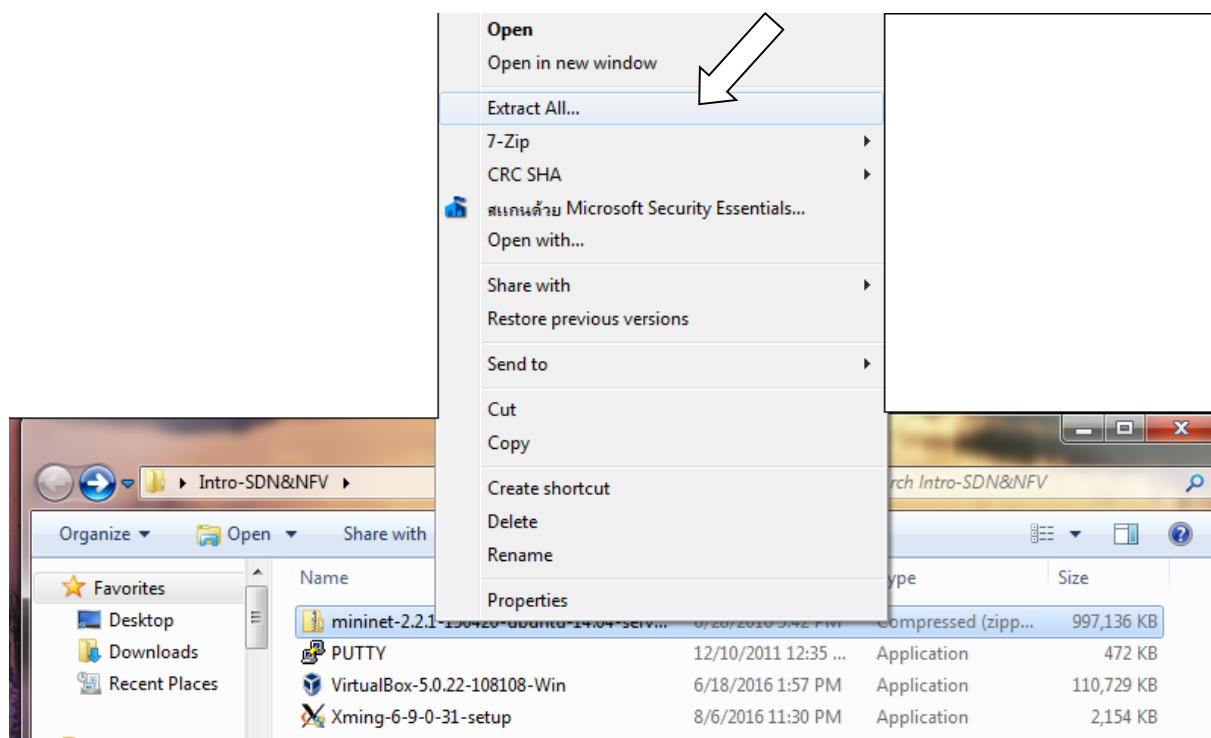
ให้ทำการ unzip ไฟล์ mininet-2.2.0-150106-ubuntu-14.04-server-i386.zip โดยการ click ไปที่ directory ชื่อ SDN Part 1



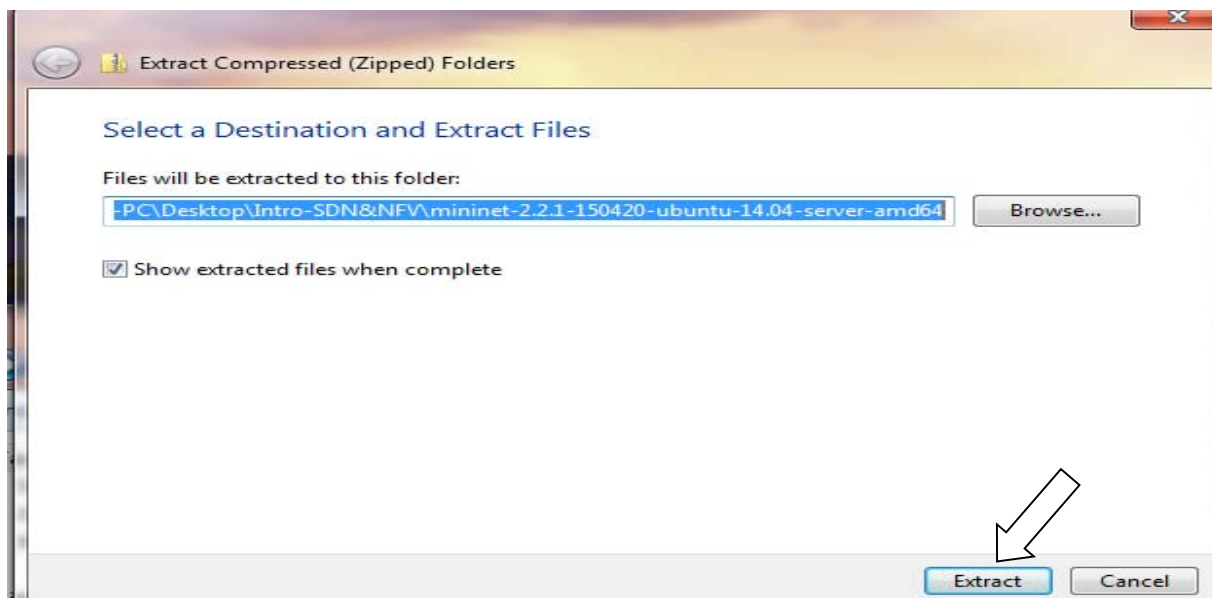
ใช้เมาส์คลิก ขวาไปที่ไฟล์ชื่อ mininet-2.2.0-150106-ubuntu-14.04-server-i386.zip



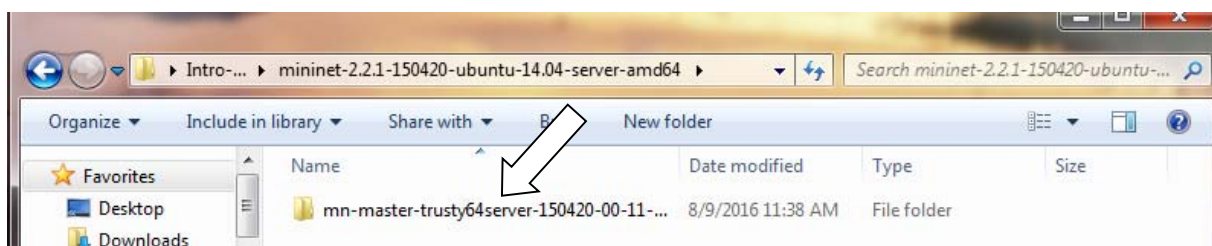
click ที่ Extract All..



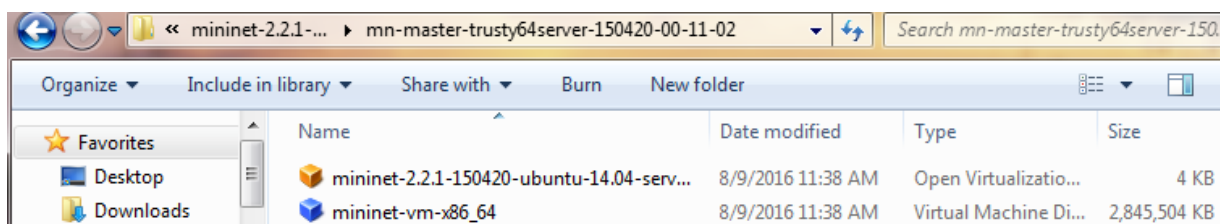
Click ที่ extract เพื่อขยายไฟล์ที่ถูกบีบอัดออกมาใช้งานปกติ



Click 2 ครั้งไปที่ directory mn-master-trusty64server-



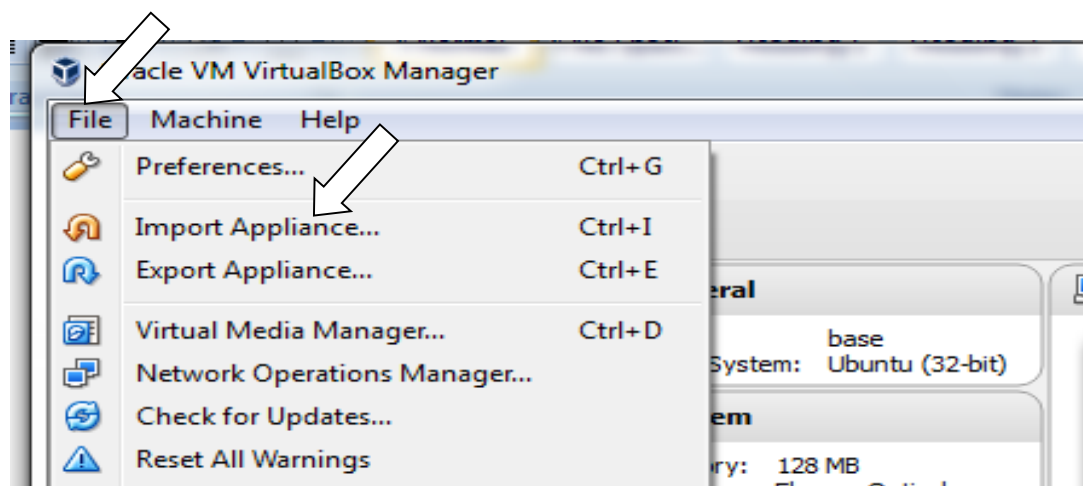
จะเห็น ไฟล์ mininet 2 ไฟล์ ที่จะถูกนำไปสร้างเป็น Virtual Machine (VM) ซึ่งก็คือ server ที่ใช้ ubuntu 14.04 เป็น operating system พร้อมกับ ติดตั้ง mininet โดยเราจะสร้าง VM ด้วย VirtualBox



เปิดการใช้งาน VirtualBox โดย click 2 ครั้ง ที่รูป Oracle VM VirtualBox

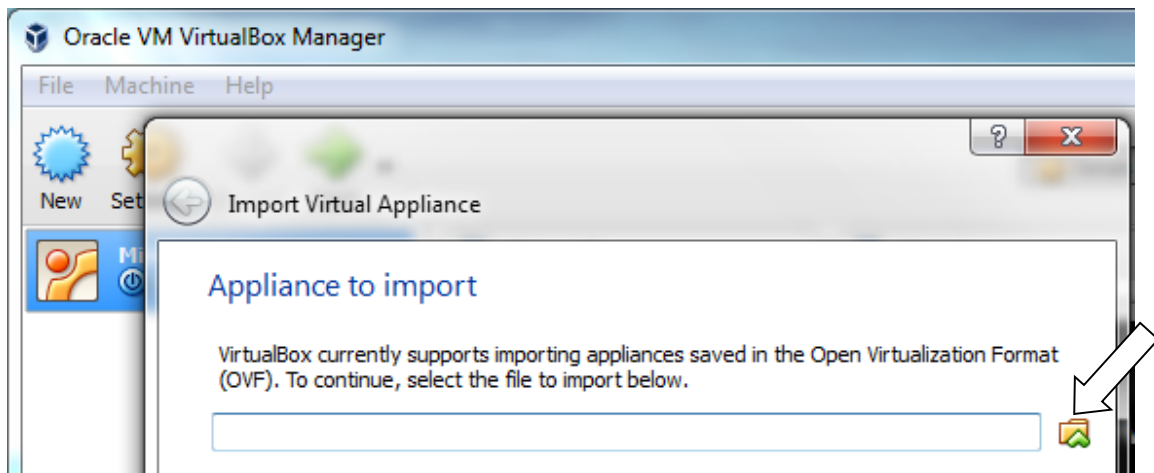


จากนั้น click ที่ menu File แล้ว click เลือก Import Appliance ตามรูป

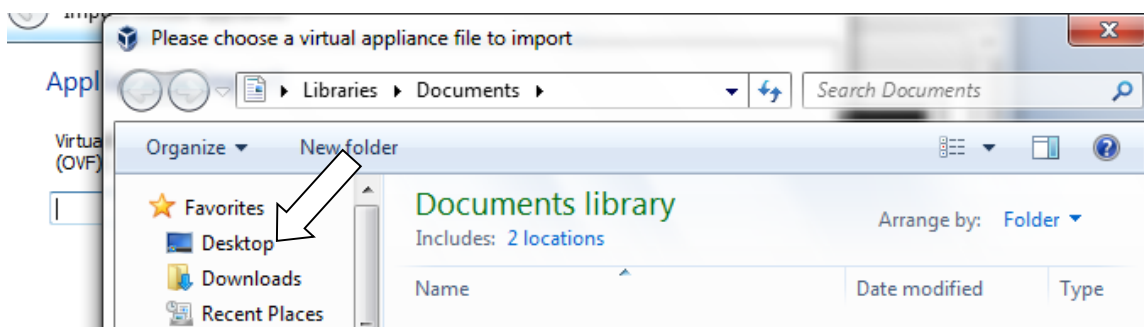




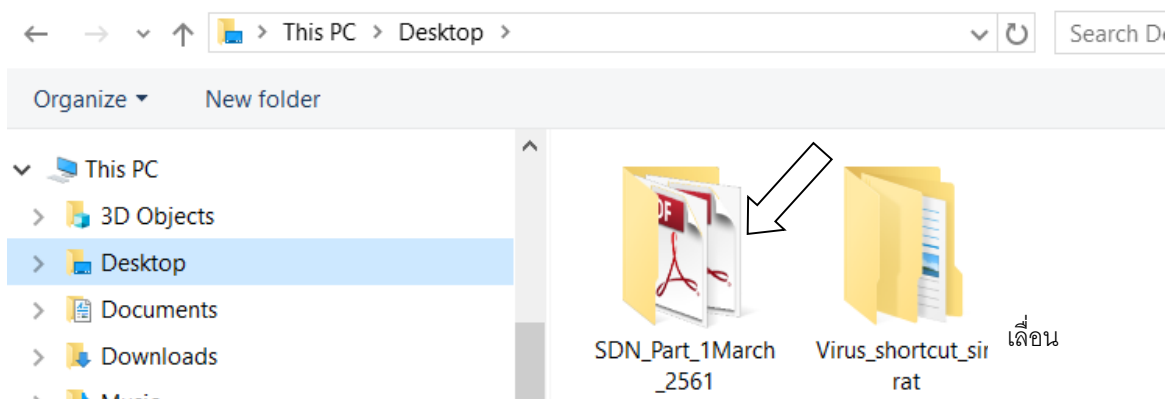
จากนั้น ใช้ เม้าส์ชี้ไปที่ สัญลักษณ์รูปแฟ้ม เพื่อเลือก virtual appliance file เพื่อ import เข้ามาใช้งาน



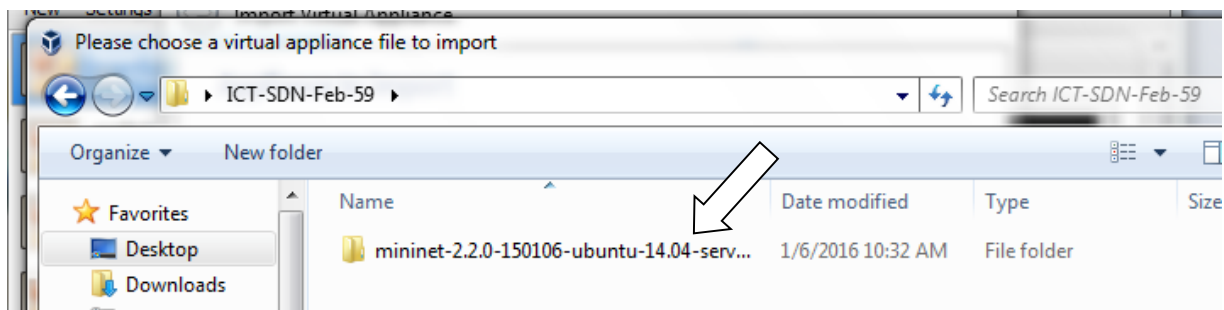
Click ไปที่ desktop



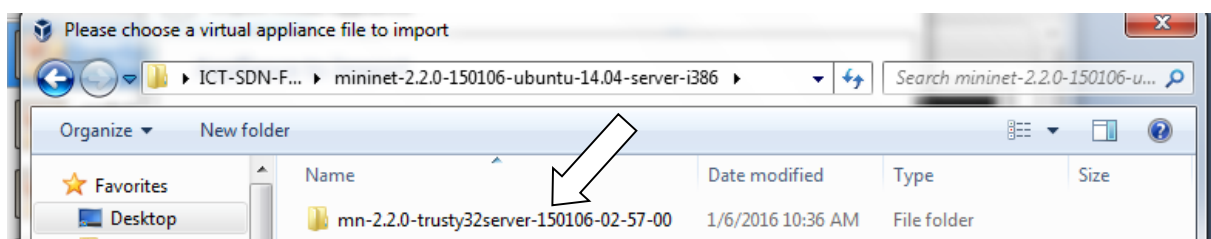
Click ไปที่ directory Intro-SDN&NFV



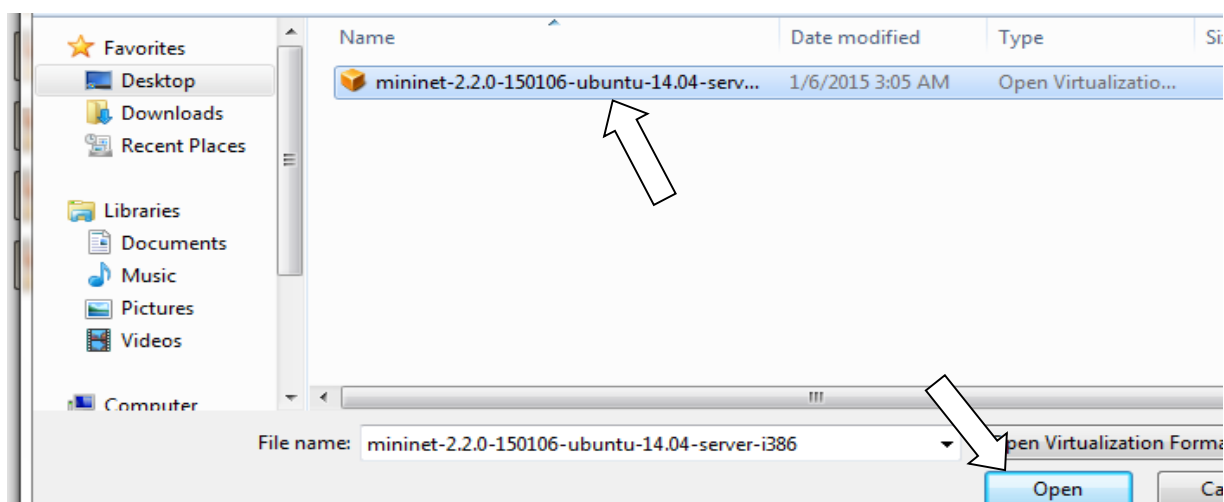
Click ไปที่ directory mininet-2.2.0-150106-ubuntu-14.04-serv...



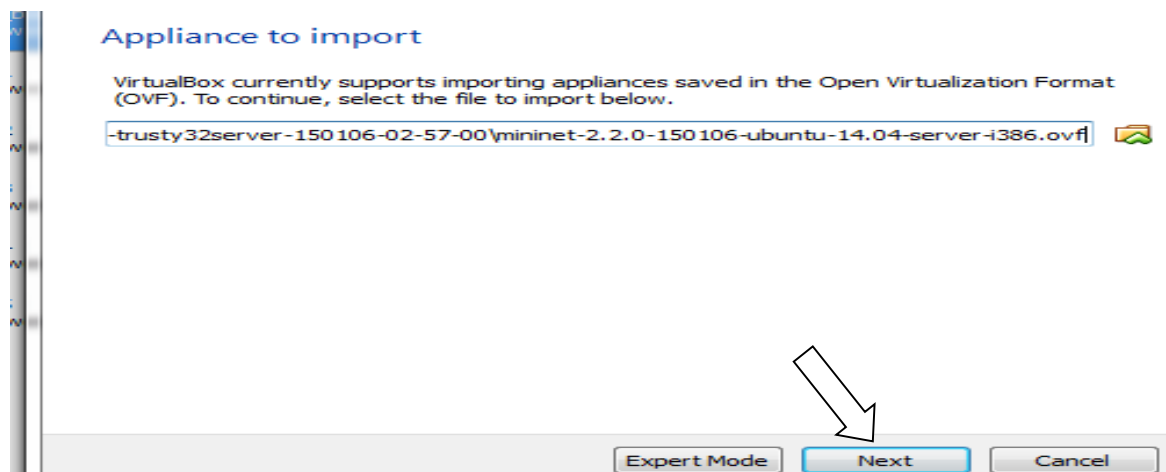
Click ไปที่ directoty ย่อยชื่อ mn-2.2.0-trusty32server-150106-02-57-00



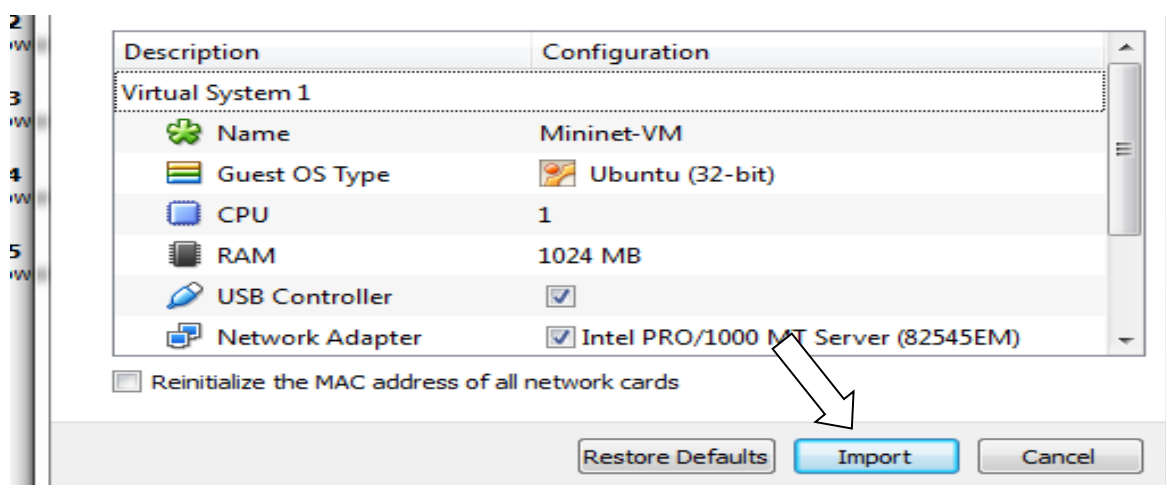
จากนั้น click ไปที่ชื่อไฟล์ mininet-2.2.0-150106-ubuntu-14.04-ser จากนั้น click ที่ open



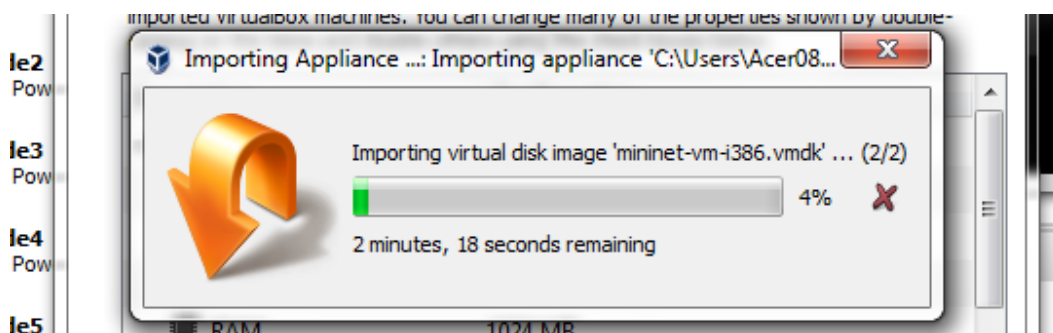
Click ที่ next



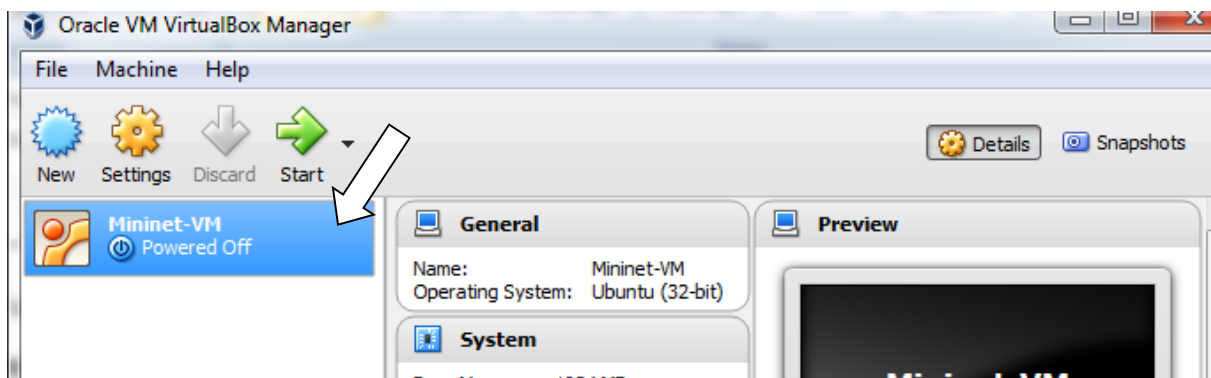
Click ที่ import



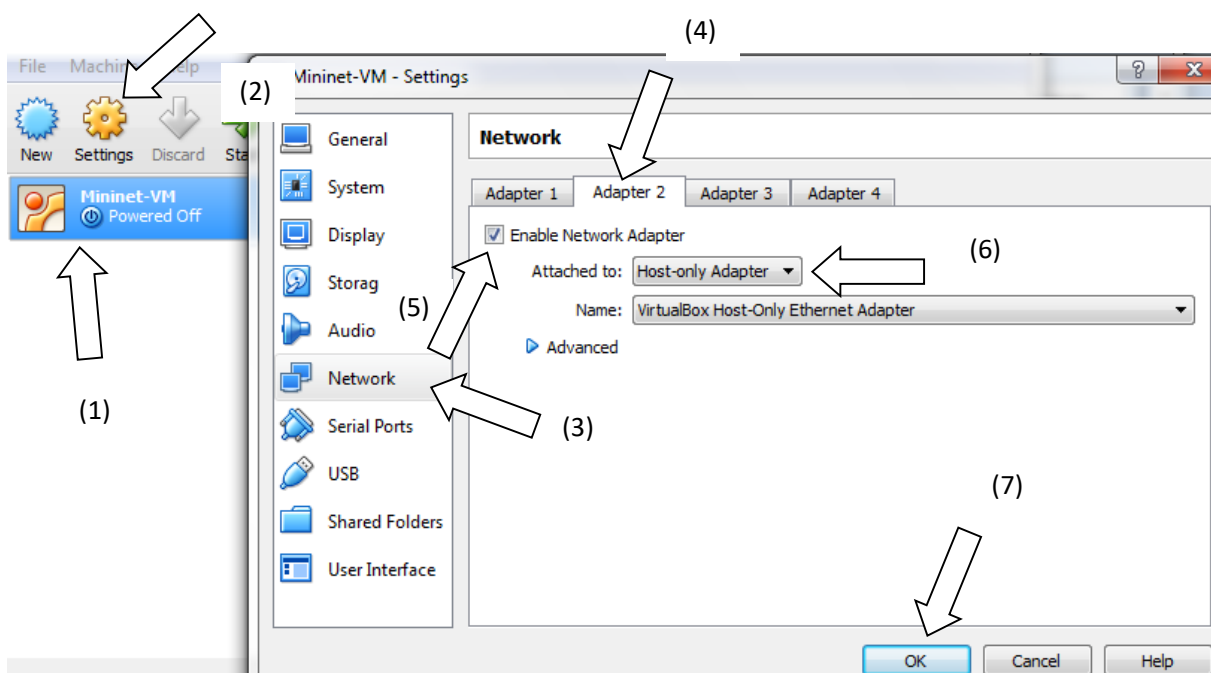
รอนกว่าการติดตั้งจะแล้วเสร็จ



เมื่อติดตั้งแล้วเสร็จจะได้ตามรูป ขณะนี้ virtual machine (VM) ได้ถูกสร้างแล้วชื่อ Mininet-VM



จากนั้นเพื่อให้สามารถเข้าใช้งาน virtual machine ได้ง่ายขึ้น ให้ติดตั้งเพิ่มเติมดังนี้, เนื่องจากอาจมีการติดตั้ง VM หลายชนิด ดังนั้นให้เลือกที่ (1) Mininet-VM, (2) setting, (3) network, (4) adapter 2, (5) check เลือก ค่า enable network adapter , (6) เลือกค่า Host-only Adapter และ (7) เลือกค่า OK

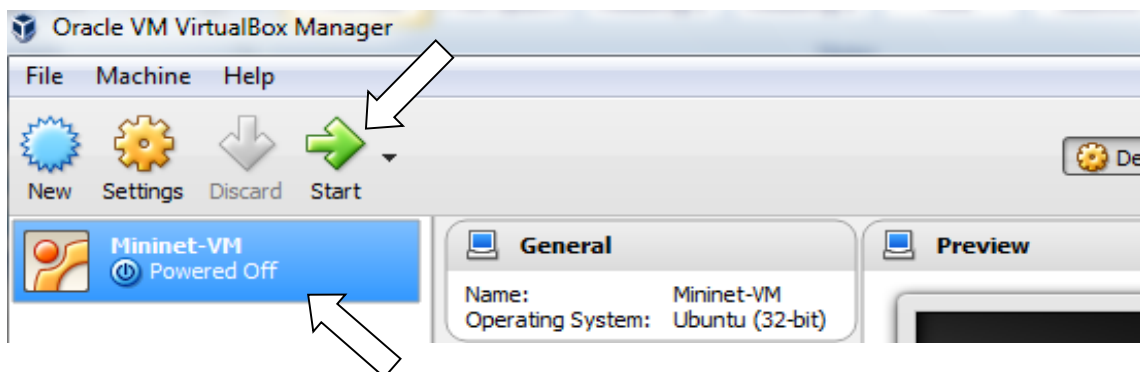


หมายเหตุ: **Adapter 1** โดย default ถูก set ให้ทำหน้าที่ NAT เพื่อให้ VM สามารถเข้าถึง internet

**Adapter 2** ถูกset เป็น host only interface เพื่อให้ program อื่นๆ เช่น putty ที่ run บน คอมพิวเตอร์จริง (host computer) ติดต่อกับ VM หรือ virtual machine ด้วย SSH (Secure Shell) เป็นรูปแบบการรับส่งข้อมูลระหว่างclient กับ serverโดยข้อมูลมีการเข้ารหัสลับ.

### 3. เตรียมความพร้อม การใช้งาน mininet

ในบางครั้งเราอาจติดตั้ง VM หลายชนิดไว้ใช้งาน ในที่นี้ มี VM ชนิดเดียว, ให้เลือก VM ชื่อ Mininet-VM , จากนั้น click ที่ start

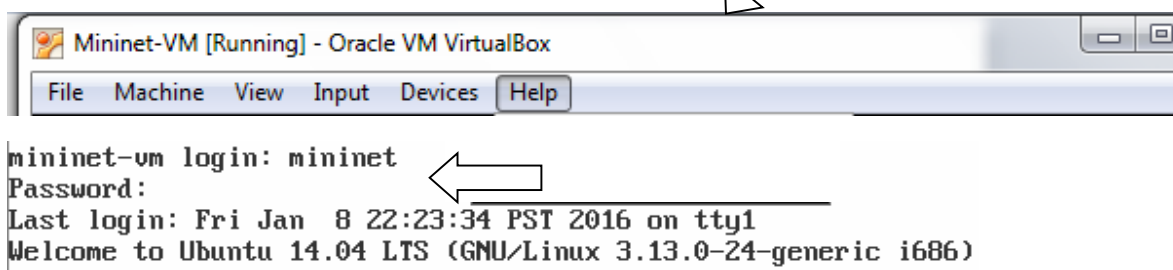


รอกจนกว่า virtual machine (VM) จะทำงานเสร็จ ,

หน้าจอ login จะมองเห็นไม่ชัด ให้ click เครื่องหมาย X ที่อยู่มุมขวาของ แถบเมนูที่มาปิดทับ login

พิมพ์ค่า login : mininet, Password: mininet

หน้าจอ ของ virtual machine



ใช้คำสั่ง ifconfig เพื่อดูค่า ip address ของเครื่อง VM มีค่า 192.168.56.101


mininet@mininet-vm:~\$ ifconfig

```
eth0      Link encap:Ethernet  HWaddr 08:00:27:48:e7:ca
          inet addr:192.168.56.101  Bcast:192.168.56.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:590 (590.0 B)  TX bytes:342 (342.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
```

และให้พิมพ์คำสั่ง sudo dhclient eth1 เพื่อกำหนด ip address ให้กับ เครื่อง VM สามารถติดต่อ internet

```
mininet@mininet-vm:~$ sudo dhclient eth1
```



(โดยสามารถตรวจสอบด้วยคำสั่ง mininet@mininet-vm:\$ ifconfig)

จะพบว่าเครื่องVM ที่ eth0 มีค่า ip address :192.168.56.101 ส่วนที่ eth1 มีค่า ip address :10.0.2.15

**หมายเหตุ** หน้าจอของ virtual machine นี้ที่ผ่าน console ของ VirtualBox ต่อจากนี้ไปเราจะย่อ และไม่ใช้งานอีกต่อไป โดย click ที่เครื่องหมาย – ที่มุมขวาบน, แต่ยังไม่ปิดการใช้งาน

### ข้อสังเกต

ก. การสั่ง execute commands หรือ ให้ คำสั่งทำงานนั้น จะอยู่ในรูป

```
mininet@mininet-vm:~$ sudo command
```

(ในที่นี้ command คือ คำสั่งต่าง ๆ เช่น dhclient, nano, ovs-ofctl เป็นต้น)

นั่นคือจะใช้ sudo นำหน้าคำสั่งต่างๆที่จะใช้ ทั้งนี้ sudo หมายถึง user ใช้งานได้ชื่อ root และ การได้รับสิทธิใช้งานสูงสุด

ข. เครื่องหมายบน command line มีข้อแตกต่างกันดังนี้:

**Mininet> command** (คำสั่ง เช่น pingall, sudo mn, iperf ตั้งจะได้กล่าวต่อไป)

เป็นคำสั่งหรือ commands ที่ถูกใช้หน้าหน้า ด้วย mininet> จะถูก run บน mininet console

**\$ command**

Command ที่ถูกใช้หน้าหน้าด้วย \$ จะถูก run ที่ VM ภายใต้สิทธิการใช้งาน user ธรรมดา

**# command**

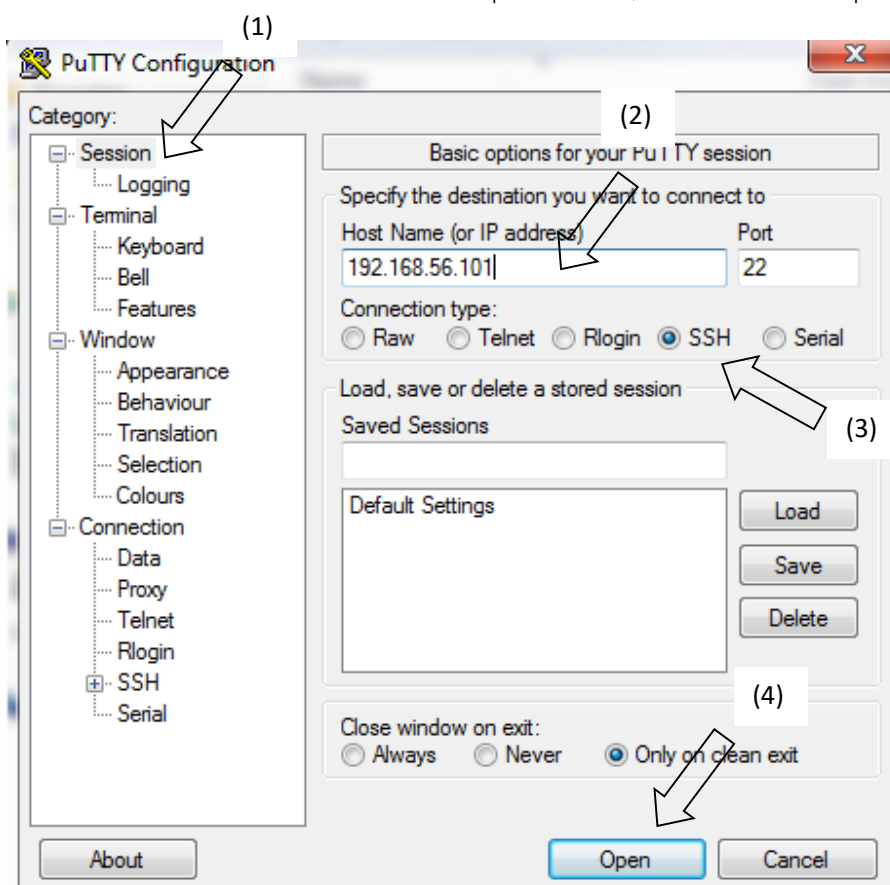
Command ที่ถูกใช้หน้าหน้าด้วย # จะถูก run ที่ VM (virtual machine) ภายใต้สิทธิการใช้งาน sudo

### 3.1 การใช้ mininet บน VM ด้วย SSH terminal (หรือ ด้วยputty ที่ใช้ SSH สำหรับ connection)

ตามที่ได้กล่าวไว้ก่อนหน้านี้ว่าการใช้ mininet บน console ของ VirtualBox โดยตรงค่อนข้างยุ่งยาก เช่น ไม่สามารถใช้งาน copy-and-paste ไป มา ระหว่าง host computer กับ VM, ดังนั้นเราจะใช้งาน mininet บน VM ต่อจากนี้ไปเราจะใช้ด้วย SSH terminal นั่นคือใช้ด้วย putty โดย click ที่ putty ตามรูป



จะได้ ตามรูปข้างล่าง จากนั้น (1) เลือก session, (2) พิมพ์ค่า IP address ของ VM คือ 192.168.56.101, (3) เลือก connection แบบ SSH (ค่า default ของ port คือ 22), และ (4) click ที่ Open



จะได้ว่าเราสามารถใช้งาน mininet บน VM ได้ตามรูปข้างล่าง

```

mininet@mininet-vm: ~
login as: mininet
mininet@192.168.56.101's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic i686)

 * Documentation:  https://help.ubuntu.com/
Last login: Wed Jan  6 15:44:29 2016
mininet@mininet-vm:~$

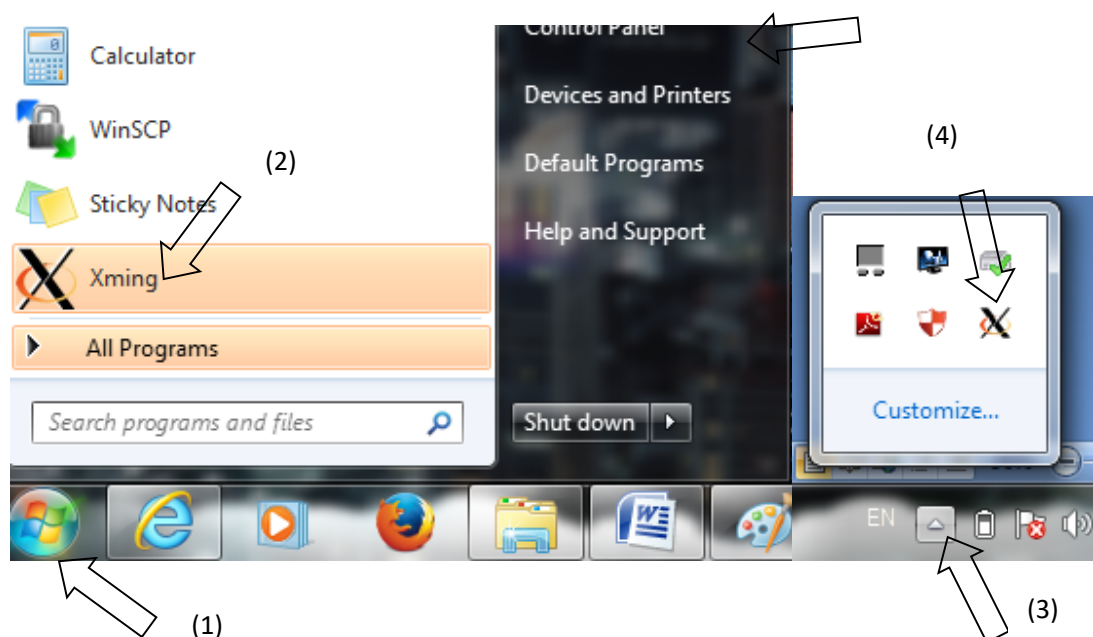
```

นั่นคือต่อจากนี้ไปการใช้งานใดๆบน mininet จะถูกใช้งานผ่านหน้าจอที่เราเรียกว่าผ่านทาง SSH terminal (หรือ ใช้ผ่าน PUTTY)

### 3.2 การเตรียมการตรวจจับข่าวสารว่าเป็นชนิดไหน ที่จุดต่างๆ (ตามลูกศรตามรูปข้างบน) ด้วย Wireshark

ก. ติดตั้ง Xming server , click 2 ครั้งที่ ไฟล์ชื่อ Xming-6-9-0-31-setup ที่อยู่ใน directory ชื่อ Intro SDN&NFV จากนั้นทำการติดตั้งโดยใช้ค่า default ทั้งหมด, จากนั้นเปิดการใช้งาน Xming server ตามรูป

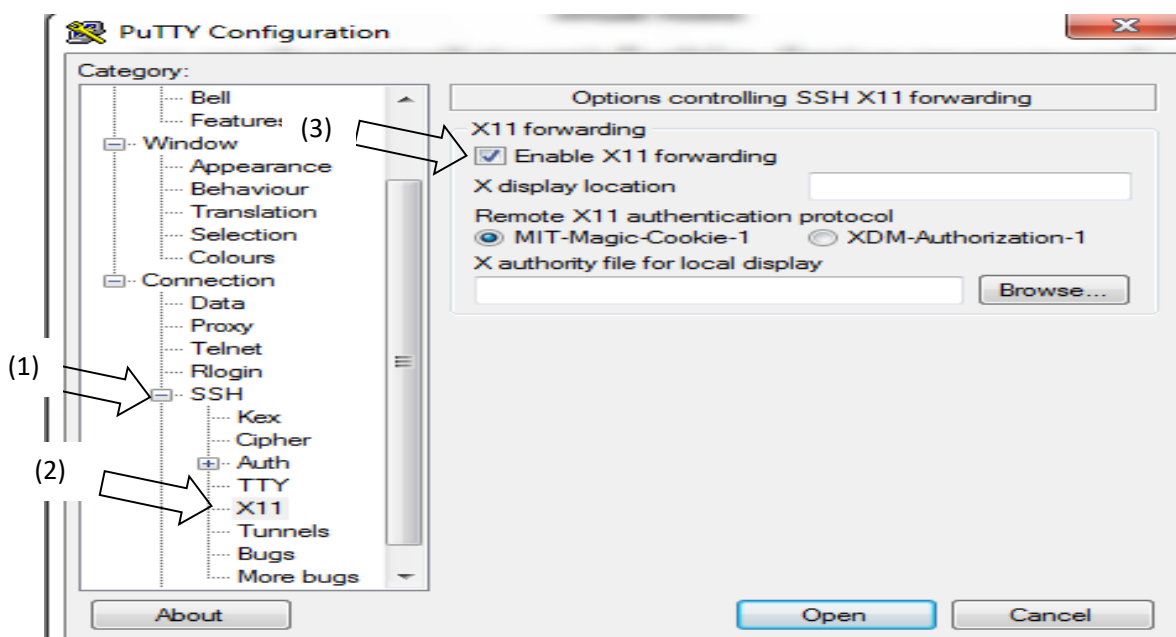
โดยการทำดังนี้ (1) click ที่ รูป windows, (2) click ที่รูป Xming, (3) click ที่รูป, (4) ถ้ามี logo ของ Xming แสดงว่า ขณะนี้ Xming server กำลังทำงานอยู่



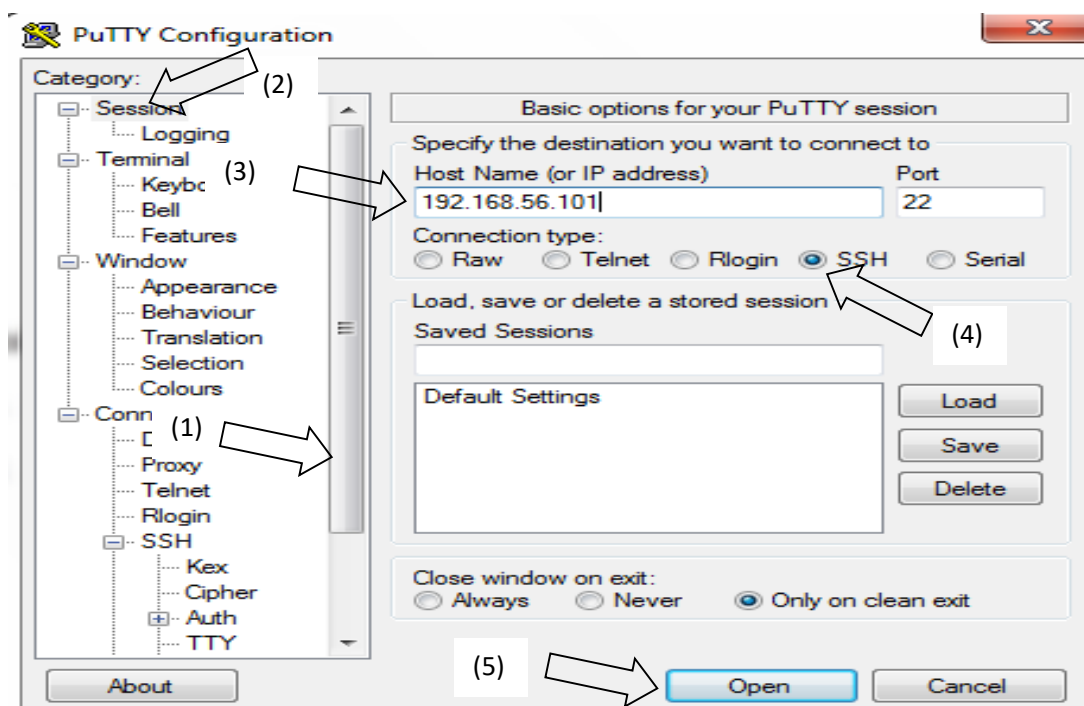


ข. การสร้าง SSH terminal ที่มีการ enable X11 forwarding

ให้เปิดการใช้งาน putty ขึ้นใหม่ บน หน้าจอ desktop, เพื่อเปิดการทำงาน (enable) x11 forwarding โดยทำตามขั้นตอนดังนี้: (1) click ที่ SSH, (2) click ที่ X11, (3) check ที่ Enable X11 forwarding



จากนั้นใช้ (1) ใช้เมาส์เลื่อน ที่แถบเลื่อนหน้าจอลงมา, (2) click ที่ session , (3) พิมพ์ค่า IP address ของ VM คือ 192.168.56.101 (4) เลือก SSH จากนั้น click open



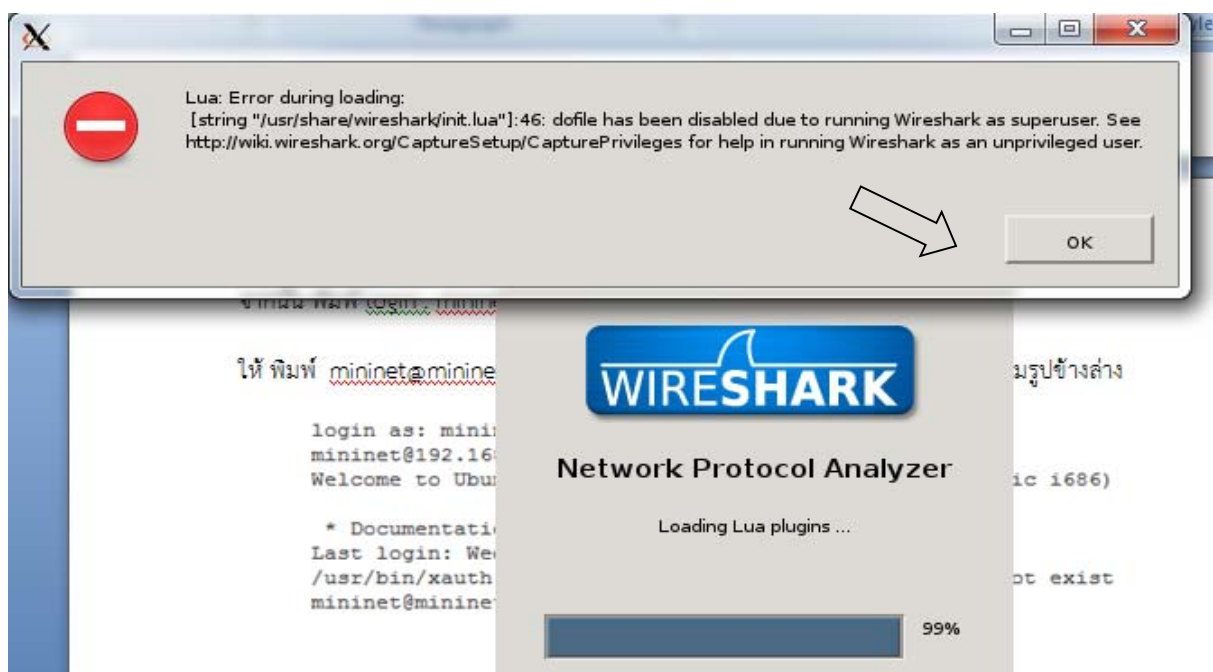
จากนั้น พิมพ์ login : mininet , password: mininet

ให้ พิมพ์ mininet@mininet-vm:~\$ sudo wireshark & จากนั้น กด enter ได้ตามรูปข้างล่าง

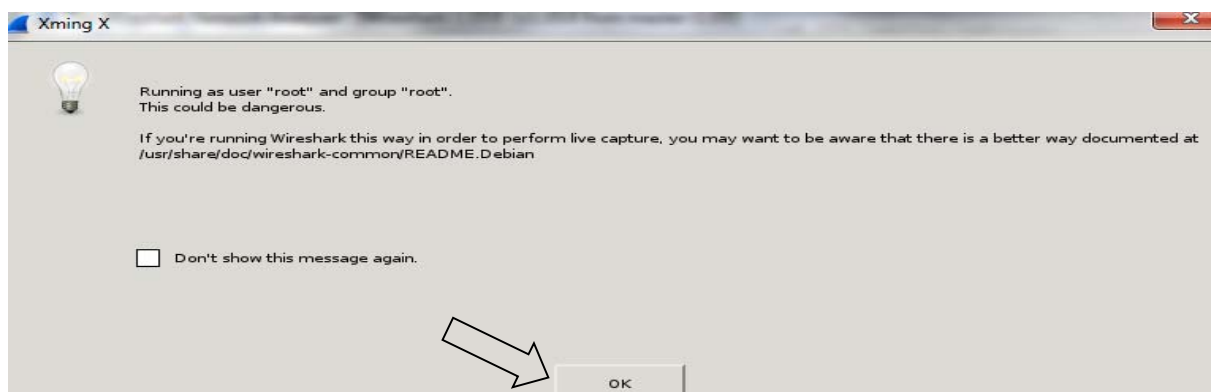
```
login as: mininet
mininet@192.168.56.101's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic i686)

* Documentation:  https://help.ubuntu.com/
Last login: Wed Jan  6 21:27:48 2016
/usr/bin/xauth:  file /home/mininet/.Xauthority does not exist
mininet@mininet-vm:~$ sudo wireshark &
```

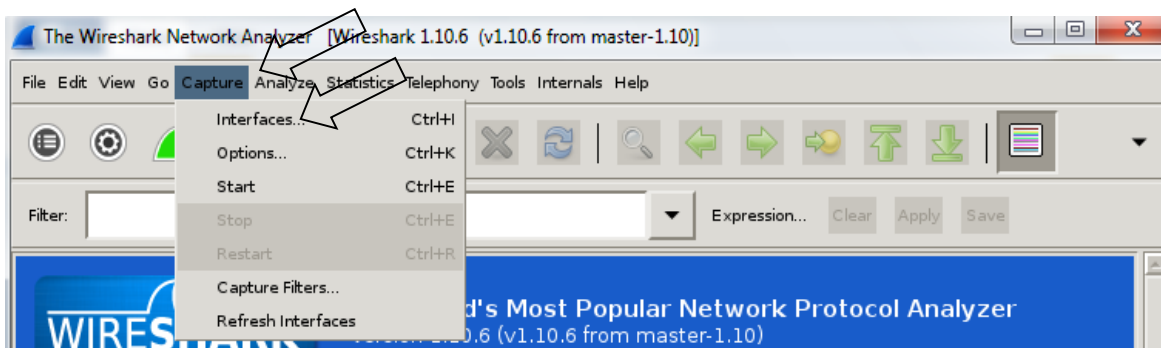
แม้ว่า得有ข้อความ Lua: error during loading ก็ตาม ให้ click ok



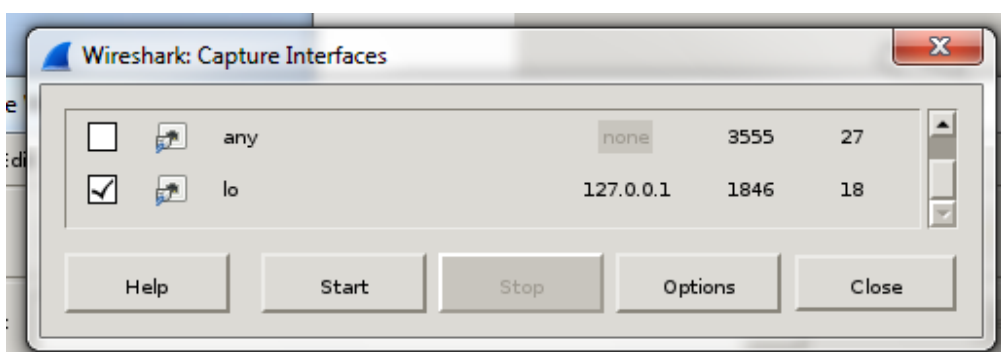
จากนั้นจะได้ข้อความเตือนอีกครั้งให้ click ok



จะได้ ตามรูป ให้ (1) click ที่ capture, (2) เลื่อนเมาส์มาที่click ที่ interface



ใช้เมาส์เลื่อนที่ แถบเลื่อนเพื่อเลือกตรวจวัด(capture) ข่าวสารที่ interface ที่ต้องการ  
ในรูปเป็นตัวอย่างการเลือกวัดที่ lo (loopback ของ VM)

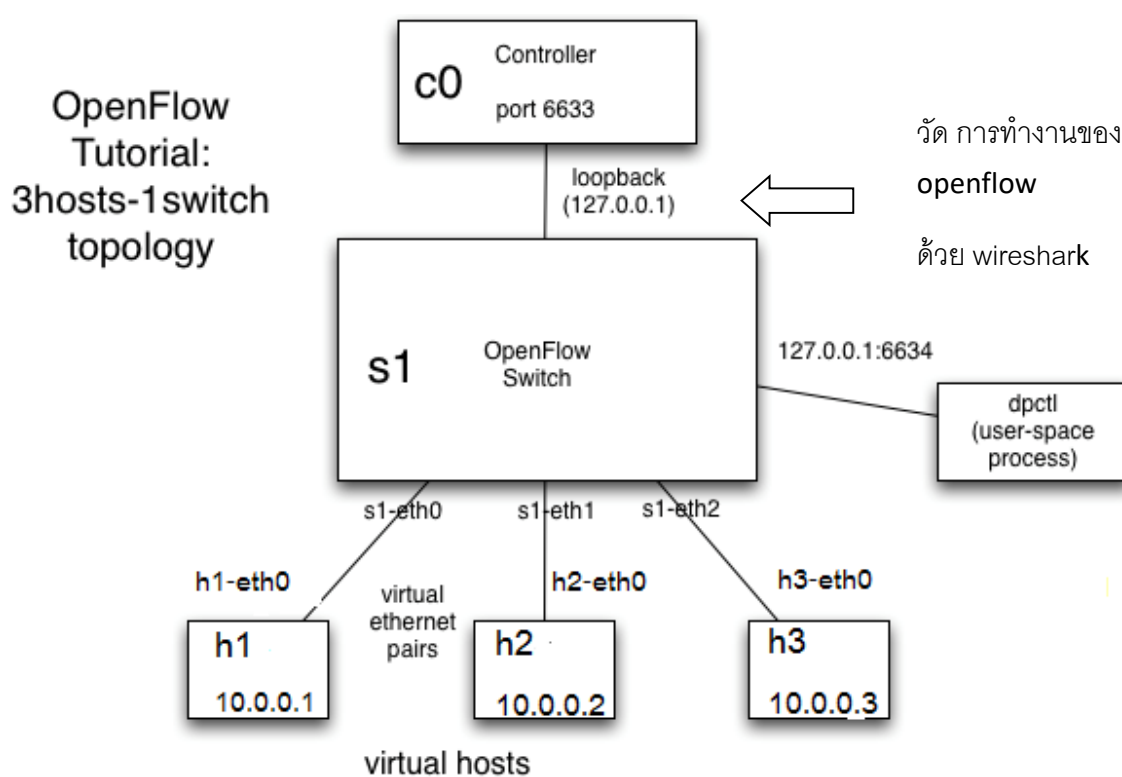


#### 4. แนะนำ openflow

OpenFlow เป็น protocol ที่กล่าวถึงขั้นตอน,รูปแบบ,การรับส่งข้อมูลการเชื่อมต่อแบบมาตรฐาน (open interface) ระหว่าง SDN controller กับ SDN switch สำหรับการควบคุมการทำงานของ switches, routers, และ access points ที่อยู่ระยะไกล ด้วย Flow table ทั้งนี้ด้วยแนวคิดเดิมการควบคุมการส่งผ่าน, หรือการ forward data ของ SDN switch จะอยู่ในตัวอุปกรณ์เหล่านั้น แต่ด้วยแนวโน้มของsdnที่ต้องการแยกการควบคุมการส่งผ่าน (control plane) ของ switch ออกจากตัว switch ที่มีไว้เพื่อรับส่งข้อมูลอย่างเดียว (data plane) โดยทำงานตามFlow table ที่ถูกประมวลผลโดย SDN controller

## 5. ใช้งาน Mininet

ในการศึกษาการทำงานของ OpenFlow จำเป็นต้องสร้างการเชื่อมต่อของอุปกรณ์หลายชนิดที่ประกอบกันขึ้นในโครงข่าย ได้แก่ virtual hosts, links, และ switches เป็นต้น และ เนื่องจาก เราจะสร้างโครงข่าย ที่ไม่ได้ใช้อุปกรณ์จริง แต่เราใช้ simulation software ชื่อ Mininet ถูกจำลองด้วย VirtualBox ทำงานบน Laptop เครื่องเดียว ในที่นี้ Mininet สร้างโครงข่ายเสมือนที่ประกอบด้วย virtual links, virtual hosts, virtual openflow switches.



```
mininet@mininet-vm:~$ sudo mn help บอกรูปแบบการใช้งานคำสั่ง mn
```

ในการศึกษาของเราจะสร้างโครงข่ายเสมือนที่ประกอบด้วย 3 virtual hosts, 1 OpenFlow switch, 1 OpenFlow controller ดังรูปในหน้าที่แล้ว ที่เราสามารถสร้างโครงข่ายเสมือนนี้ ผ่านทางSSH terminal ด้วยคำสั่งดังนี้:

```
|mininet@mininet-vm:~$ sudo mn --topo=single,3 --mac --switch=ovsk --controller=remote
```

อธิบายการใช้คำสั่ง ในแต่ละส่วนย่อยดังนี้

`sudo mn` เริ่มใช้งาน Mininet และ การ run Mininet ต้องใช้ `sudo` เพื่อใช้สิทธิของ root

-- `topo` single, 3 บวก mininet ว่า ให้สร้างการเชื่อมต่อ (topology) ของ switch แบบ single หรือ ชั้นเดียว โดย switch เชื่อมต่อไปยัง 3 virtual hosts

-- `mac` บวก Mininet ให้กำหนด แต่ละ virtual host มีค่า mac address ที่เรียงลำดับกัน และตรงกันกับ IP address ของแต่ละ virtual host เอง

-- `switch ovsk` ใช้ Open vSwitch ใน kernel mode ของ switch

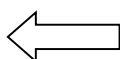
-- `controller remote` เป็นการบอก mininet ว่า OpenFlow switches แต่ละตัว จะติดต่อกับ SDN controller ที่อยู่ภายนอก Mininet

โครงข่ายจำลองที่ถูกสร้างขึ้นด้วย mininet

ยังไม่ถูกเชื่อมต่อกับ controller

ผลการใช้คำสั่งข้างบนได้ผลดังนี้ :

```
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1
*** Starting CLI:
mininet>
```



หน้าจอของ SSH terminal ที่เริ่มต้นด้วย `mininet>` เรียกว่า

Mininet console

## 5.1 คำสั่งเบื้องต้นของ Mininet

เนื่องจากเราจะใช้งาน Mininet ในการทดลองนี้เป็นส่วนใหญ่ จึงนับว่าเป็นประโยชน์ที่จะเรียนรู้คำสั่งใช้งานเฉพาะของ Mininet พอสมควรดังนี้, โดยคำสั่งต่างๆจะถูก run บน Mininet console

ดูรายการของ nodes ที่ถูกสร้างขึ้น, ใช้คำสั่ง nodes

```
mininet> nodes
available nodes are:
c0 h1 h2 h3 s1
```

C0 => OpenFlow controller

h1, h2, h3 => virtual host1, virtual host2, virtual host3

s1=> virtual OpenFlow switch

- ดู รายการของคำสั่ง ที่สามารถใช้งานได้ , ใช้คำสั่ง help

```
mininet> help
```

```
Documented commands (type help <topic>):
```

```
=====  
EOF      gterm   iperfudp nodes      pingpair   py         switch  
dpctl    help    link     noecho     pingpairfull quit       time  
dump     intfs   links    pingall    ports      sh         x  
exit     iperf   net      pingallfull px         source    xterm
```

```
You may also send a command to a node using:
```

```
<node> command {args}
```

```
For example:
```

```
mininet> h1 ifconfig
```

```
The interpreter automatically substitutes IP addresses  
for node names when a node is the first arg, so commands  
like
```

```
mininet> h2 ping h3
```

```
should work.
```

```
Some character-oriented interactive commands require  
noecho:
```

```
mininet> noecho h2 vi foo.py
```

```
However, starting up an xterm/gterm is generally better:
```

```
mininet> xterm h2
```

```
mininet>
```

- ทำการ run คำสั่งเดียวที่ node, ให้ใส่ชื่อ node ไว้หน้าคำสั่ง ตัวอย่างการใช้ เช่น ต้องการทราบ IP address ของ virtual host 1, ให้ใช้คำสั่ง h1 ifconfig

```

mininet> h1 ifconfig
h1-eth0  Link encap:Ethernet  HWaddr 00:00:00:00:00:01
         inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
         inet6 addr: fe80::200:ff:fe00:1/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:0 (0.0 B)  TX bytes:738 (738.0 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:0
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

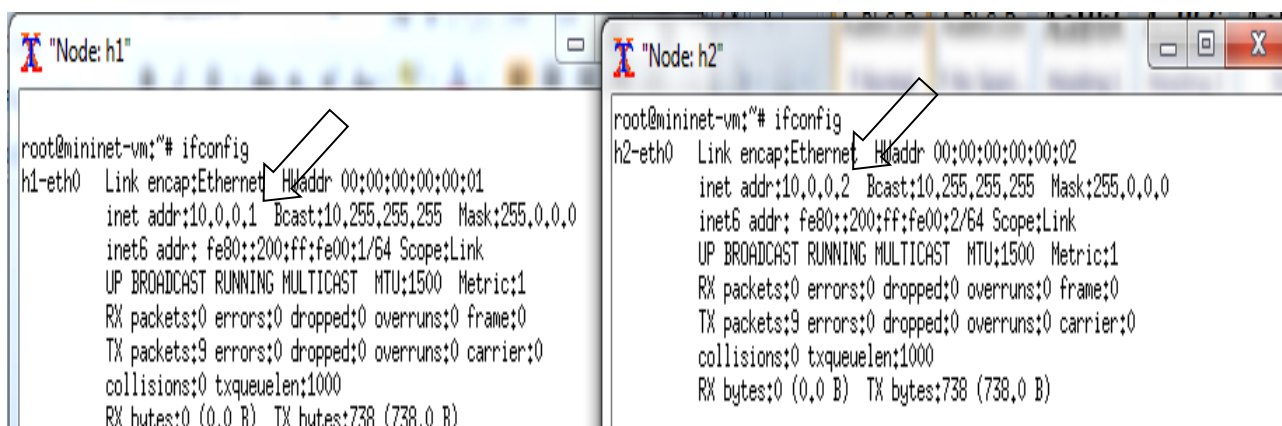
```

- มีทางเลือกให้ใช้ในบางกรณี ที่ต้องการ run คำสั่งโต้ตอบ กับ virtual host เพื่อตรวจหาข้อบกพร่อง ที่ output เราสามารถใช้คำสั่ง xterm กับ virtual host เดียวหรือมากกว่า เช่น xterm h1 h2

(โดยก่อนใช้ คำสั่ง xterm บน SSH terminal จะต้องทำตามข้อ 3.2 ก. เปิดการทำงาน Xming server, และข้อ 3.2 ข. การสร้าง SSH terminal ที่มีการ Enable X11 forwarding)

```
mininet> xterm h1 h2
```

ได้ผลตามรูปข้างล่างคือ มี หน้าจอ ปรากฏขึ้นมา 2 หน้าจอ (เราป้อนค่า h1 h2) เพื่อให้เราใช้คำสั่งบน virtual host คือ h1 และ h2 ได้โดยตรง และหน้าจอทั้งสองนี้เรียกว่า **xterm terminal**



จากตัวอย่างเราต้องการทราบ IP address ของ h1 และ h2 โดยใช้คำสั่ง ifconfig , การเรียกใช้งาน หน้าจอของ h1, h2 จะถูกเรียกใช้ โดย click ที่ รูปของ Xming server, ในที่นี้ท่านสามารถปิดการใช้งานของหน้าจอ h1 และ h2 ได้ เนื่องจากคำสั่งส่วนใหญ่เราจะใช้ผ่าน Mininet console

- ในบางครั้ง ถ้า Mininet ทำงานไม่ถูกต้อง และจำเป็นต้อง restart, ในขั้นแรกให้ท่าน ออกจากการใช้ Mininet (โดยการใช้ exit command หรือ control-D) และพยายามลบสถานะและการประมวลผลด้วยการใช้คำสั่ง sudo mn -c

```
mininet@mininet-vm:~$ sudo mn -c
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd
ovs-controller udpbwtest mnexec ivs 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openfl
owd ovs-controller udpbwtest mnexec ivs 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
*** Removing OVS datapathsovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
```

## 6. ตัวอย่าง การใช้ ovs-ofctl

(ขออ้อถึงขณะนี้เรายังคง เปิดการใช้งาน Mininet บน VM ด้วย putty ผ่าน SSH อยู่หนึ่งหน้าจอ ซึ่งมีการ run คำสั่ง sudo mn -topo=single,3 --mac --switch=ovsk --controller=remote ใช้งานอยู่ตามในข้อ5)

ovs-ofctl เป็น คำสั่ง ที่อำนวยความสะดวกสำหรับสั่ง (configure) ให้ Open vSwitch ทำงานตามที่ต้องการ หรือ สอบถาม สถานะการทำงานของ Open vSwitch และเช่น สามารถสั่งให้ Open vswitchทำงานตามที่ต้องการโดย ส่งผ่านข้อมูลไปที่ตารางการทำงาน หรือ flow table ของ Open Vswitch นอกจากนี้เราจะพบว่า คำสั่งนี้ (ovs-ofctl) มีประโยชน์โดยเฉพาะเอาไว้หาข้อบกพร่อง (debugging) โดยการดู สถานะของ switch port (ช่อง interface), flow state, flow counters, และ ในการทดลองที่จะได้กล่าวต่อไปเราใช้คำสั่งนี้เพื่อสั่ง Open vswitch ให้ทำงานตามที่ต้องการแบบง่ายๆ โดยการเพิ่ม flow ด้วยคำสั่งที่ละบรรทัด



สร้างหน้าจอใหม่สำหรับการใช้งาน Mininet บน VM ด้วย putty ผ่าน SSH (ตามข้อ3.1) และ run คำสั่ง ovs-ofctl show s1 ดังนี้: (ต้องมี sudo นำหน้าคำสั่ง ovs-ofctl มิฉะนั้น จะไม่ได้รับอนุญาตให้ใช้คำสั่งนี้)

เราใช้คำสั่ง ovs-ofctl show s1 ก็เพื่อดูสถานะของ port ของ Open vswitch หรือ ก็คือ S1 นั่นเอง ที่ถูกเราสร้างด้วยคำสั่ง sudo mn --topo single,3 --mac --switch ovsk --controller remote

```
mininet@mininet-vm:~$ sudo ovs-ofctl show s1
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000001
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: OUTPUT SET_VLAN_VID SET_VLAN_PCP STRIP_VLAN SET_DL_SRC SET_DL_DST
SET_NW_SRC SET_NW_DST SET_NW_TOS SET_TP_SRC SET_TP_DST ENQUEUE
 1(s1-eth1): addr:ca:9b:e1:61:8e:7f
   config:      0
   state:       0
   current:     10GB-FD COPPER
   speed: 10000 Mbps now, 0 Mbps max
 2(s1-eth2): addr:36:4e:3e:4b:24:39
   config:      0
   state:       0
   current:     10GB-FD COPPER
   speed: 10000 Mbps now, 0 Mbps max
 3(s1-eth3): addr:c6:a5:cf:1a:e9:f4
   config:      0
   state:       0
   current:     10GB-FD COPPER
   speed: 10000 Mbps now, 0 Mbps max
LOCAL(s1): addr:e6:73:69:ed:ae:ae
 config:      PORT_DOWN
 state:       LINK_DOWN
 speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

โดยคำสั่ง show จะเชื่อมต่อไปยัง switch และ แสดงสถานะ(state) ของ port ของ switch (s1),และค่าสมรรถนะ(capabilities)อื่นๆของ switch (s1)

คำสั่งที่มีประโยชน์อีกคำสั่งคือ ovs-ofctl dump-flows s1 (ต้องมี sudo นำหน้าคำสั่ง ovs-ofctl มิฉะนั้น จะไม่ได้รับอนุญาตให้ใช้คำสั่งนี้)

```
mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s1
NXST FLOW reply (xid=0x4):
```

option ที่เราใส่ คือ dump-flows ก็เพื่อต้องการดูค่าใน flow-table ของ Open Vswitch S1 ในที่นี้ได้ค่าใน flow-table ว่างเปล่า ทั้งนี้เนื่องจากเรายังไม่ได้ สั่งให้ controller ทำงานเลย

**หมายเหตุ:** .กรณีตัวอย่างข้างบน, ovs-ofctl ติดต่อ Open vSwitch ที่อยู่ภายในของVM ดังนั้นการติดต่อก็จะใช้ชื่อ S1 ได้, แต่ในกรณีที่เราจะติดต่อกับเครื่อง OpenFlow Switch ที่มี hardware จริง และกำลังทำงานอยู่ และเราต้องการทราบสถานะของ OpenFlow Switch ดังกล่าวแล้วจะใช้คำสั่งดังนี้:

```
$ ovs-ofctl dump-flows tcp : {IP address} : {port}
```

โดย {IP address} คือ IP address ของ interface ที่ใช้ทำ management ของ switch และ {port}

ก็จะเป็น listening/management port ของ OpenFlow ที่เป็น passive

## 6.1 ทดสอบคำสั่ง Ping

ไปที่ Mininet console ที่ยังคงเปิดการใช้งานอยู่ และให้ทดลองใช้ คำสั่ง ping จาก h1 ไปยัง h2 ดังนั้นที่ Mininet console จะได้ดังนี้:

```
mininet> h1 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2000ms
pipe 3
mininet>
```

สังเกตว่า ชื่อของ host h2 ถูกแทนด้วย IP address 10.0.0.2 อย่างอัตโนมัติ เมื่อ run คำสั่ง ping ใน Mininet console

**คำถาม** การใช้คำสั่ง ping ครั้งนี้ h1 ติดต่อ h2 ได้หรือไม่ ? ทำไม ?

ตามที่ท่านได้ทราบก่อนหน้านี้แล้วว่า flow table ของ switch s1 ว่างเปล่า, นอกจากนี้ switch s1 ก็ยังไม่ได้ถูกเชื่อมต่อไปยัง controller ใดๆ ดังนั้น switch s1 จึงยังไม่ว่าจะต้องทำอะไร เมื่อมี ข้อมูลเข้ามาถึง switch s1 จึงทำให้ การping ครั้งนี้ไม่ประสบความสำเร็จ

ให้เราใช้ ovs-ofctl เพื่อติดตั้ง หรือเพิ่ม flow บน switch S1 ที่จำเป็นด้วยcommand-line, ที่ SSH terminal ใช้คำสั่งดังนี้:

```
mininet@mininet-vm:~$ sudo ovs-ofctl add-flow s1 in_port=1,action=output:2
mininet@mininet-vm:~$ sudo ovs-ofctl add-flow s1 in_port=2,action=output:1
```

ด้วยการใช้คำสั่งข้างบน จะบอกให้ switch ทำการส่งผ่าน (forward) ข้อมูลที่เข้ามาทาง port 1 พาท่อออกไปทาง port 2 และในทำนองเดียวกันเมื่อมี ข้อมูลเข้ามาทาง port 2 แล้ว switch ก็จะพาท่อออกไปทาง port 1, เราสามารถพิสูจน์ความถูกต้องการทำงานของ flow-table ของ switch s1 ได้โดยใช้คำสั่งดังนี้:

```
mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
  cookie=0x0, duration=619.451s, table=0, n_packets=0, n_bytes=0, idle_age=619, in_port=1 actions=output:2
  cookie=0x0, duration=575.683s, table=0, n_packets=0, n_bytes=0, idle_age=575, in_port=2 actions=output:1
```

ไปที่ Mininet console (SSH terminal) และให้ทดลองใช้ คำสั่ง ping จาก h1 ไปยัง h2 อีกครั้ง ดั่งนั้นที่ Mininet console จะได้ดังนี้

```
mininet> h1 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.90 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.092 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.091 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.091/0.696/1.905/0.854 ms
```

**คำถาม** การใช้คำสั่ง ping ครั้งนี้ h1 ติดต่อ h2 ได้หรือไม่ ?

ให้ตรวจสอบ flow-table อีกครั้ง และดูค่าสถิติ สำหรับในแต่ละบรรทัดของ flow, และค่าเหล่านี้เป็นไปตามที่ท่านคาดหวังไว้หรือไม่ โดยอาศัยการ ping

จากนั้นให้ออกจาก Mininet โดยการใส่คำสั่ง

```
mininet> exit
```

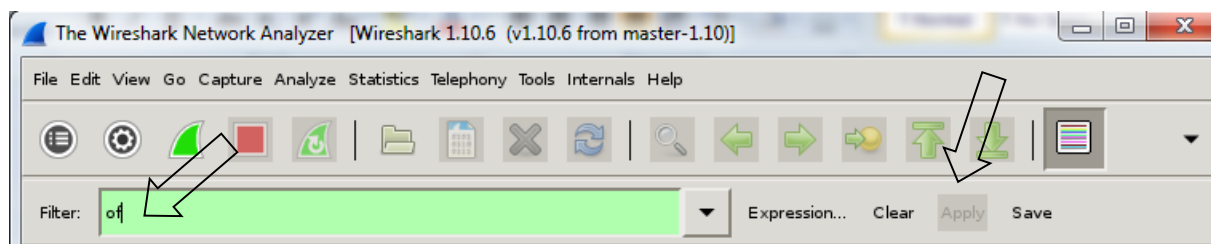
จากนั้นใส่คำสั่ง restart Mininet

```
|mininet@mininet-vm:~$ sudo mn -c
```

เป็นการยกเลิกการสร้าง โครงข่ายจำลอง

## 7. เปิดการใช้งาน Controller และ ใช้ Wireshark ดู ข่าวสารที่รับส่งในช่วงเริ่มการทำงานของ controller.

ให้ทำการเปิดการใช้งานของ wireshark โดยทำตามข้อ 3.2 ทั้งข้อ ก และข, จากนั้นที่ Wireshark ให้ตั้งค่าที่ช่องของ filter โดยการพิมพ์ of (เป็นตัวย่อของ OpenFlow) ตามในรูปเพื่อดูข่าวสารที่ใช้ควบคุมของ OpenFlow ที่รับส่งระหว่าง controller กับ switch



จากนั้น click ที่ apply ตามรูปข้างบนเพื่อทำการ filter ข่าวสารที่ถูกบันทึกทั้งหมด

จากนั้นทำตามข้อ 3.2 ข เพื่อวัดชนิดข่าวสารที่ interface lo, โดยใช้ เม้าส์เลือกที่ capture, interface, จากนั้น วัดที่ interface lo นั่นคือขณะนี้ Wireshark กำลังตรวจจับข่าวสารที่ interface lo ของ VM

ให้เริ่ม การทำงานของ controller อย่างง่าย ด้วยคำสั่ง controller ptcp:6633 & ที่ SSH terminal ,

```
mininet@mininet-vm:~$ controller ptcp:6633 &
[2] 6132
```

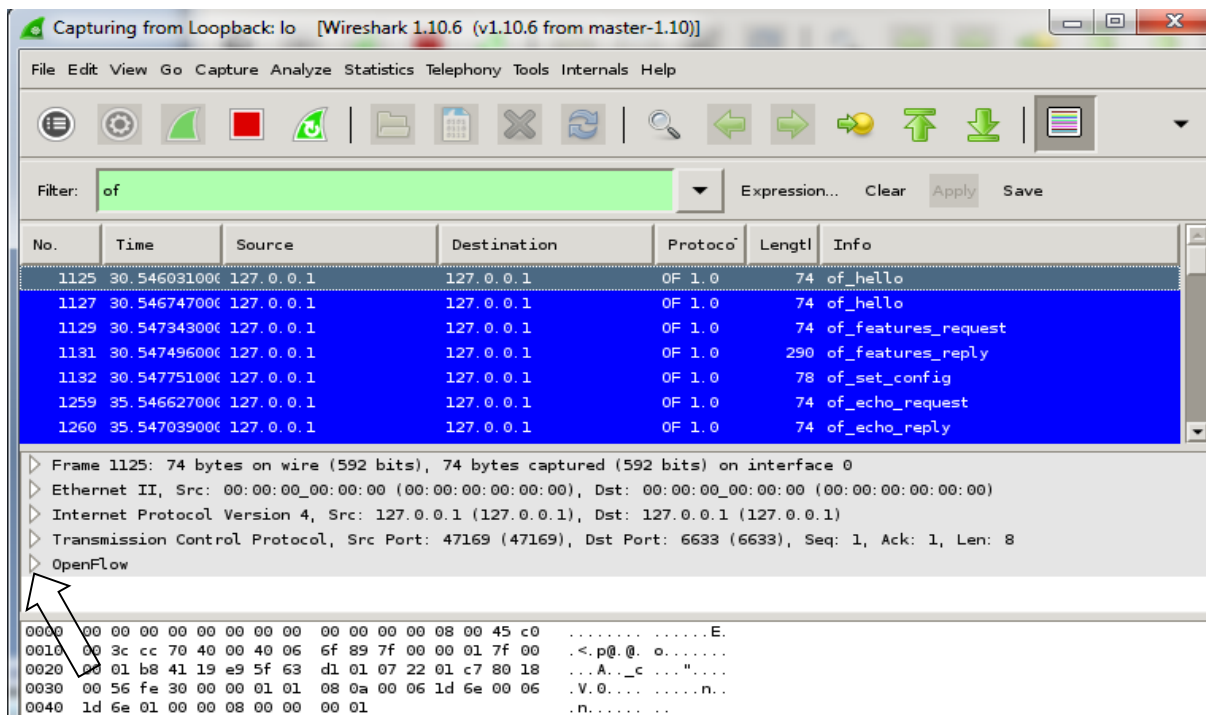
เป็นการ สั่งให้ openflow controller ทำงาน ที่ port 6633 , ip=127.0.0.1,

ดูวิธีใช้ mininet@mininet-vm:~\$ controller --help , ptcp ใช้ passive openflow connection

จากนั้นให้สร้างโครงข่ายจำลองที่เคยสร้างแล้ว (openflow switch 1 ตัว, virtual hosts 3ตัว, และ controller ) ด้วยคำสั่ง sudo mn -topo=single,3 -mac -switch=ovsk -controller=remote , สังเกตว่าโครงข่ายจำลองที่ถูกสร้างขึ้นนั้น สามารถเชื่อมต่อกับ controller ได้สำเร็จ (Adding controller)

```
mininet@mininet-vm:~$ sudo mn --topo=single,3 --mac --switch=ovsk --controller=remote
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

โดยที่ไม่ต้องเพิ่ม flow การทำงานใดๆลงบน flow-table ของ switch ในข้อ 6. (ด้วยคำสั่ง sudo ovs-ofctl add-flow s1 in\_port=1, action=output:2) โดยเราจะเห็น กลุ่มของข่าวสารที่แสดงโดย Wireshark เกิดขึ้น ดังนี้ (เนื่องมาจากการใช้คำสั่ง \$ controller ptcp:6633 & )

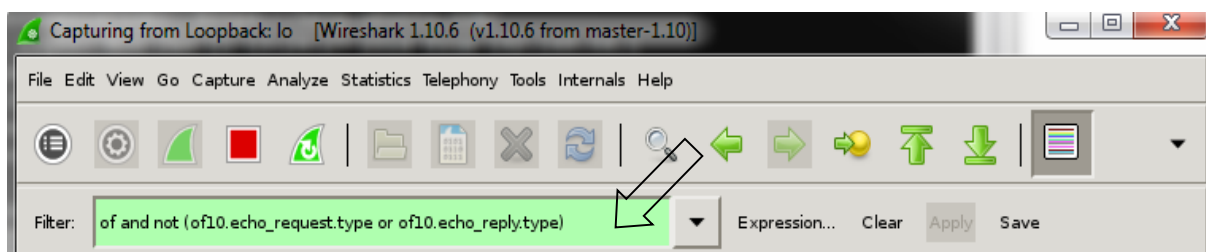


ข่าวสารของ OpenFlow ที่รับส่งระหว่าง controller กับ switch เริ่มจากการแลกเปลี่ยนข่าวสาร Hello , ตามด้วย Feature\_request, Feature\_reply, set\_config, echo\_request และ echo\_reply โดยการ click ที่รูปสามเหลี่ยม ข้างหน้าของ OpenFlow ก็จะแสดงรายละเอียดเพื่อดูข้อมูลของส่วนต่างๆใน OpenFlow เพิ่มเติม เป็นต้น

- ให้ ดูข่าวสารของ OpenFlow จากการใช้คำสั่ง Ping,

ก่อนอื่นที่ช่อง filter ของ Wireshark เราจะกรองเพื่อไม่ให้ Wireshark แสดงข่าวสาร OpenFlow ที่เป็นชนิด of\_echo\_request และ of\_echo\_reply (ซึ่งเป็นข่าวสารของ OpenFlow ที่ใช้ตรวจสอบการมีอยู่ระหว่าง controller และ switch) โดยพิมพ์คำสั่งต่อไปนี้ลงในช่องของ filter

(of and not (of10.echo\_request.type or of10.echo\_reply.type)



จากนั้นให้ลบค่าต่างๆ บน ARP table ของ host h1และ h2 (arp table บอกค่า MAC address ของ host ปลายทางและบอกว่าอยู่ interface ไหน) โดยใช้คำสั่ง ที่ Mininet consoleดังนี้:

```
mininet> h1 ip -s -s neigh flush all
```

```
mininet> h2 ip -s -s neigh flush all
```

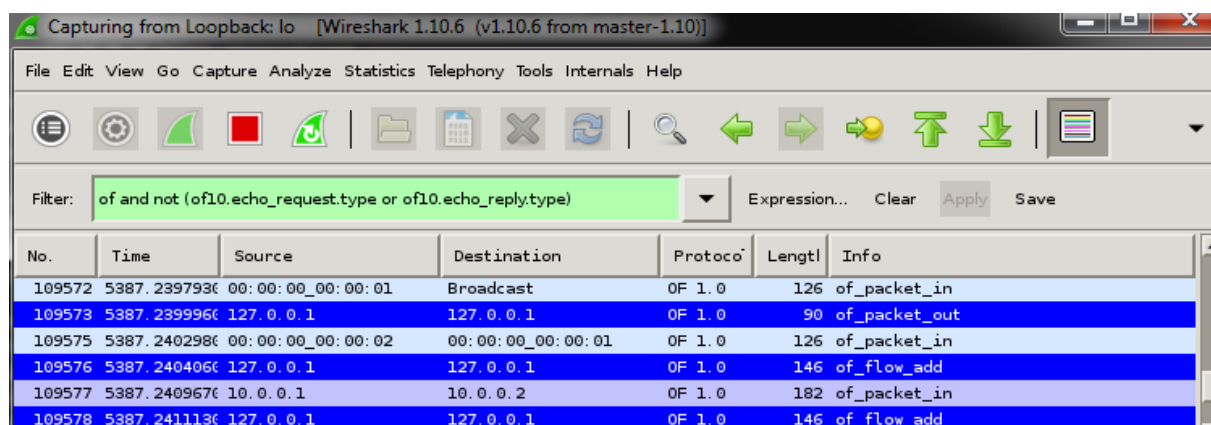
ใช้คำสั่ง ping โดย ping จาก h1 ไปยัง h2 จำนวน 1 ครั้ง, โดยทำที่ Mininet console ดังนี้

```
mininet> h1 ping -c1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=2.15 ms
```

### คำถาม

ท่านคิดว่า การ ping ครั้งนี้ จาก host h1 ไปยัง host h2 ทำไม จึง ping สำเร็จ ?

และเมื่อใช้ Wireshark ตรวจสอบก็จะได้ OpenFlow ชนิดต่างๆ เช่น of\_packet\_in , of\_packet\_out, of\_flow\_add เป็นต้น ตามรูปข้างล่าง



No.	Time	Source	Destination	Protocol	Length	Info
109572	5387.2397936	00:00:00_00:00:01	Broadcast	OF 1.0	126	of_packet_in
109573	5387.2399966	127.0.0.1	127.0.0.1	OF 1.0	90	of_packet_out
109575	5387.2402986	00:00:00_00:00:02	00:00:00_00:00:01	OF 1.0	126	of_packet_in
109576	5387.2404066	127.0.0.1	127.0.0.1	OF 1.0	146	of_flow_add
109577	5387.2409676	10.0.0.1	10.0.0.2	OF 1.0	182	of_packet_in
109578	5387.2411136	127.0.0.1	127.0.0.1	OF 1.0	146	of_flow_add

### คำถาม

1. ให้ใช้ wireshark วัตและดูค่าอย่างละเอียด เพื่อหาว่า ARP\_request, ARP\_reply อยู่ที่ packet ไหน ( ณะ ดูที่ of\_packet\_in, )
2. ดูว่า Openflow Controller สั่งให้ switch S1 ทำงาน ตามที่ controller ต้องการอยู่ที่ packet ไหน ( ณะ ดูที่ of\_packet\_add, )

ให้ออกจาก mininet ด้วย exit และเป็นการยกเลิกการสร้าง โครงข่ายจำลองด้วย sudo mn -c

## 8. การสร้างโครงข่ายเสมือนด้วยMininet ที่ใช้ GUI (Graphical User Interface)

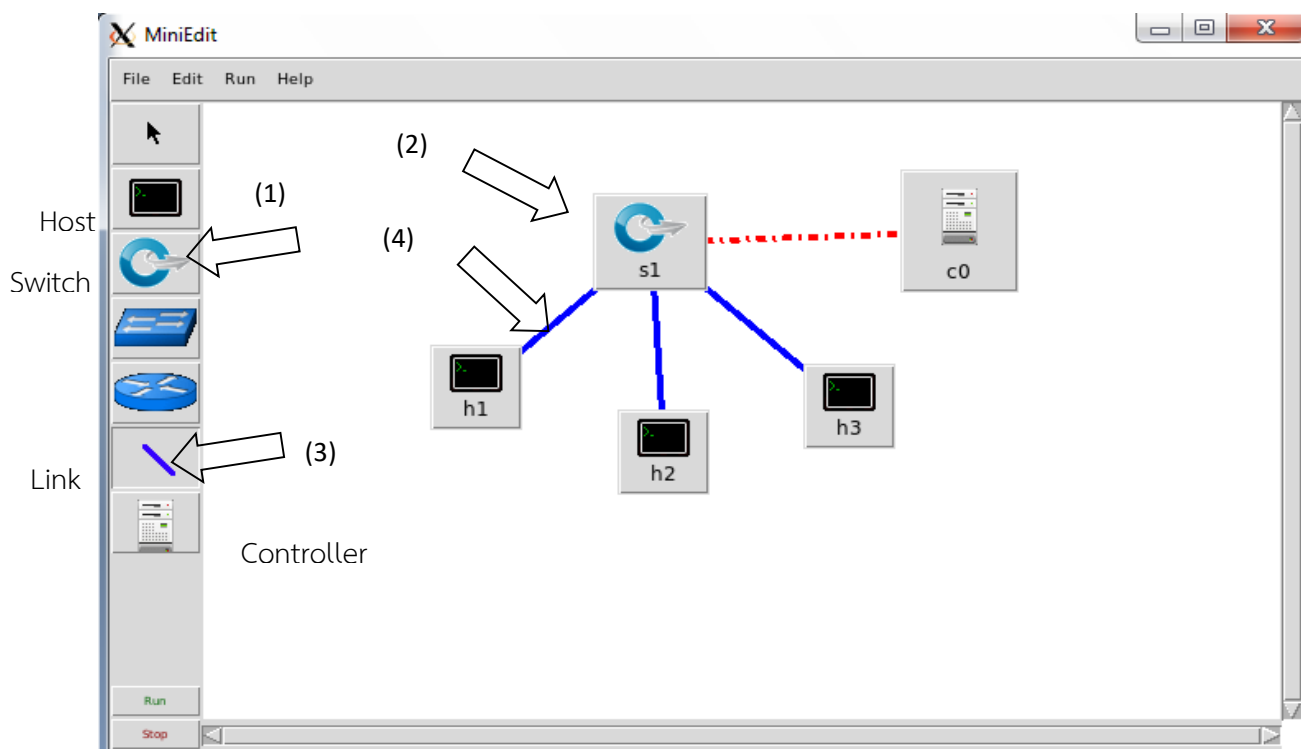
ในที่นี่ตามในข้อ 5. เราได้สร้างโครงข่ายเสมือน ที่ประกอบด้วย 3 virtual hosts, 1 OpenFlow switch, 1 OpenFlow controller ผ่านทางSSH terminal ด้วยคำสั่ง:

```
mininet@mininet-vm:~$ sudo mn --topo single,3 --mac --switch ovsk --controller remote
```

ไปแล้วนั้น ในที่นี่เราจะสร้างโครงข่ายเดียวกัน (ตามในข้อ 5 การใช้ Mininet หน้า 14) แต่ใช้ GUI , โดยที่เราจะต้องทำในข้อ 3.2ก.คือติดตั้งXming serverและทำในข้อ 3.2ข คือการ enable X11 forward บน ssh terminal, ดังนั้นที่ SSH terminal (ที่ได้ enable X11 forwarding) ให้พิมพ์คำสั่งดังนี้:

```
mininet@mininet-vm:~$ sudo ~/mininet/examples/miniedit.py
```

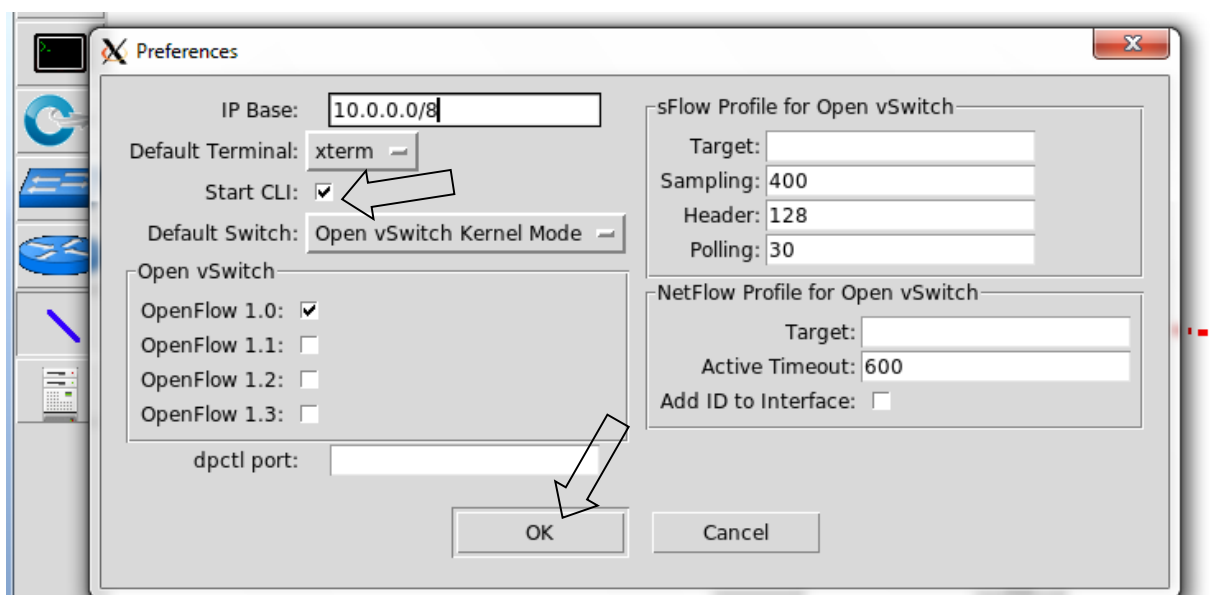
และจะได้ตามรูปข้างล่าง(หน้าจอว่างเปล่า), จากนั้น ให้เราเลือก อุปกรณ์ที่เราต้องการสร้าง ก็เอาเมาส์เลื่อนมาที่รูป s1 จากนั้นให้ click ลงไปที่รูปของ switch s1 (1), จากนั้น ใช้เมาส์เลื่อนมาที่หน้าจอ พื้นที่ว่างทางขวามือ แล้ว click ลงไป ก็จะได้รูป switch s1 ปรากฏขึ้นมา(2), ทำในทำนองเดียวกันนี้ กับอุปกรณ์อื่นๆ (host h1,host h2,host h3, controller c0)จนครบทุกตัว, จากนั้น ใช้เมาส์ มา click ที่ link (3), จากนั้น สมมุติต้องการสร้าง link ระหว่าง S1 กับ h1 ก็เอาเมาส์มา click ที่ s1 โดยกดเมาส์ค้างไว้ แล้วลากเมาส์มาที่ h1 แล้วปล่อยการกดปุ่มของเมาส์ (4) ก็จะได้ link ที่เชื่อมต่อระหว่าง S1 กับ h1 ทำเช่นนี้จนกว่าได้สร้าง link ครบตามรูปข้างล่าง



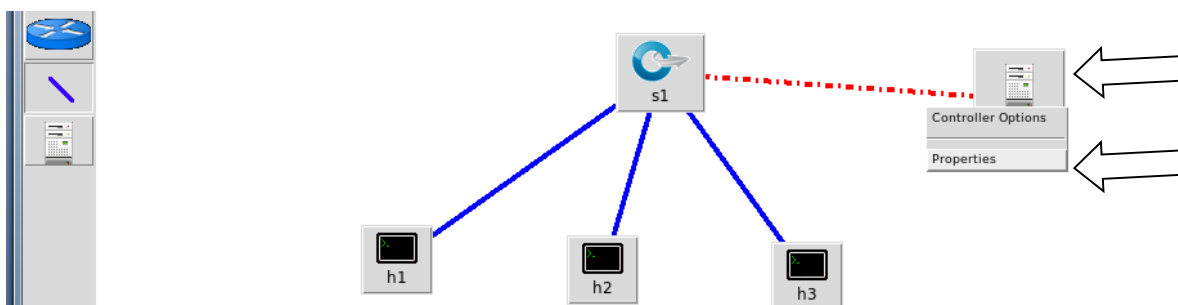
การใช้ MiniEdit GUI เพื่อสร้างโครงข่ายที่ประกอบด้วย Switch, Host, และ controller จะมีการ configure ค่า preference เพิ่มเติม , โดย click ที่ edit แล้ว click ที่ preference



จะได้ตามรูปข้างล่าง , ให้ click เลือก Start CLI แล้ว Click OK, ตามรูป



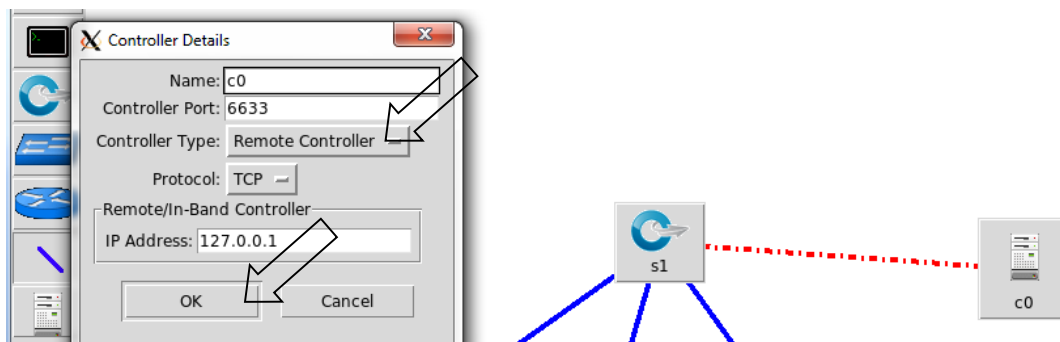
ใช้เมาส์ชี้ไปที่ controller แล้วก็กดเมาส์ค้างไว้ แล้วเลื่อนมาชี้ที่ properties แล้วปล่อย



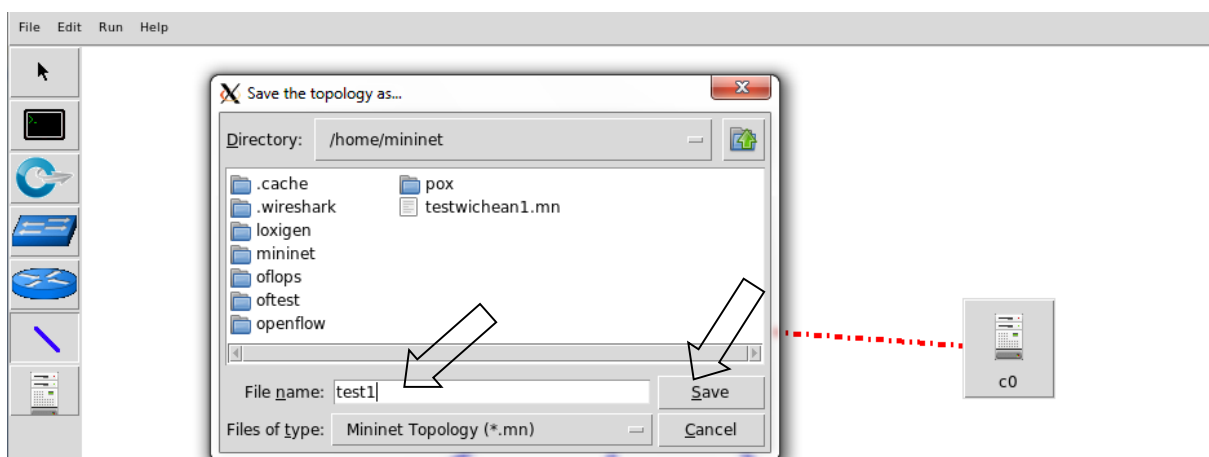
หมายเหตุ หน้าจอของ MiniEdit จะถูกซ่อนอยู่ให้ click ไปที่ Xming server ก็จะได้ หน้าจอ MiniEdit



เลือก controller type เป็น remote controller, จากนั้น click ที่ OK

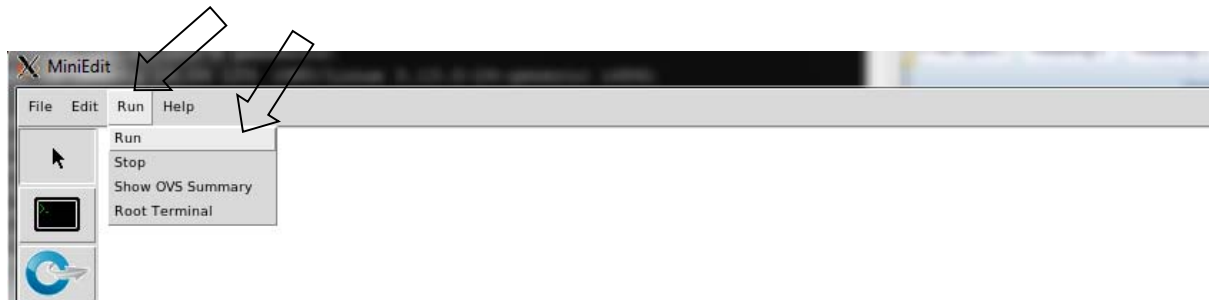


จากนั้นให้มา click ที่ file, click ที่ save จะได้ตามรูปข้างล่างใส่ชื่อไฟล์เช่น ในที่นี้ตั้งชื่อเป็น test1, click ที่ save



- สั่ง ให้โครงข่ายที่เราสร้างขึ้นมา ทำงาน

ให้เลื่อน เม้าส์มา พร้อมกับ click ที่ RUN เลื่อน เม้าส์ลงมาพร้อมกับ click ที่ run อีกครั้ง



จากนั้นให้ เปิดหน้าจอ SSH terminal (ใช้เม้าส์ ไปที่ รูปจอซ้อนกัน 2 เครื่อง ที่ menu bar ว่างสุดเพื่อเลือกจอ SSH terminal ที่มีการ run Mininet) จะได้หน้าจอ Mininet console ตามรูป

```
NOTE: PLEASE REMEMBER TO EXIT THE CLI BEFORE YOU PRESS THE STOP BUTT
ting will prevent MiniEdit from quitting and will prevent you from st
network again during this sessoin.
```

```
*** Starting CLI:
mininet>
```

มาถึงขณะนี้ ที่ OpenFlow Switch ยังไม่ได้เชื่อมต่อไปยัง controller ดังนั้น ถ้า ping จาก h1 ไปยัง h2 จำนวน 1 ครั้ง การ ping ในครั้งนี้ ก็จะไม่สำเร็จ เนื่องจาก switch S1 ไม่รู้ว่าจะทำอะไรเมื่อมี ข้อมูลเข้ามาที่ switch S1

```
mininet> h1 ping -c1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
```

ในขั้นต่อไปจะ สาธิตแนวคิดพื้นฐานของ software-defined networking (SDN) ที่ใช้ SDN controller ชื่อ POX **POX controller สั่งให้ switch S1ทำงาน เป็น Hub**

โดยใช้คำสั่งดังนี้ ที่ SSH terminal (เปิดหน้าจอ putty ใหม่)

```
mininet@mininet-vm:~$ sudo ~/pox/pox.py forwarding.hub
```

เป็นการบอก POX sdn controller ทำงาน โดยสั่งให้ switch S1 ทำงาน เป็น hub, อย่งไรก็ตาม switch อาจใช้เวลาสักครู่ เพื่อเชื่อมต่อไปยัง controller และให้คอยจนกว่า switch S1 เชื่อมกับ controller โดยหน้าจอของ POX บนจอ SSH terminal จะแสดงดังรูป

```
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:forwarding.hub:Hub running.
INFO:core:POX 0.2.0 (carp) is up.
INFO:openflow.of_01:[None 1] closed
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
INFO:forwarding.hub:Hubifying 00-00-00-00-00-01
```

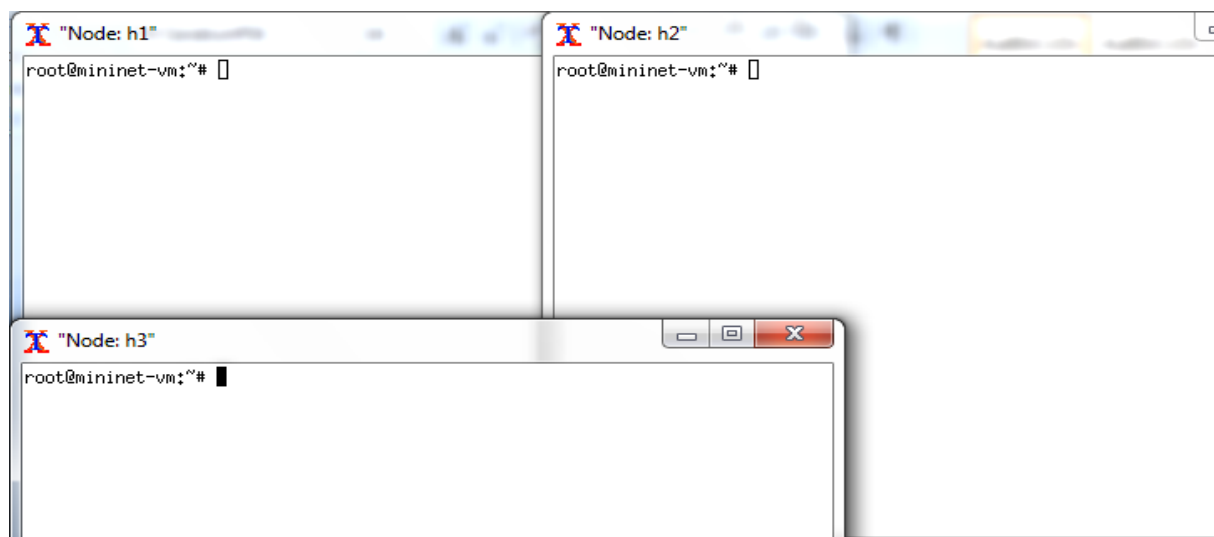


ตรวจสอบการทำงานของ switch s1 (ถูกสั่งให้ทำหน้าที่เป็น hub) ด้วย tcpdump

เราจะตรวจสอบว่า host แต่ละตัวสามารถ ping ถึงกันและกันได้, โดย host ทุกตัวจะมองเห็น ข่าวสารชนิดเดียวกัน ซึ่งก็คือการทำงานของ hub, ในที่นี้เราจะสร้าง terminal ของ host แต่ละตัวด้วยคำสั่งที่ mininet console ตามข้างล่างนี้

```
mininet> xterm h1 h2 h3
```

ใช้เมาส์ชี้ไปที่ รูป Xming Server ที่ tool bar บรรทัดล่างสุดของ windows พร้อม click ไปที่ host แต่ละตัว และจัดให้ทั้ง 3 หน้าจอของ host แสดงผลพร้อมกันตามรูป



ที่หน้าจอของ host h2, ใช้คำสั่ง #tcpdump

ที่หน้าจอของ host h3, ใช้คำสั่ง #tcpdump

ที่หน้าจอของ host h1, ใช้คำสั่ง #ping -c1 10.0.0.2

คำสั่ง ping จะถูกส่งไปยัง controller , ที่จะทำให้ คำสั่ง ping ดังกล่าวถูก flood คือ ถูกส่งออกทุก port ของ switch ได้แก่ port 2, 3 แต่จะไม่ถูกส่งออก port 1 ซึ่งคำสั่ง ping ถูกส่งเข้ามาทาง port นี้ เราจะมองเห็นว่า

ข่าวสารของทั้ง ARP และ ICMP packets เดียวกันนี้ จะปรากฏพร้อมกันบนหน้าจอของ host h2, host h3

ที่สอดคล้องกับคำสั่ง ping ที่เราใช้ อันนี้เป็นการแสดงการทำงานของ hub นั่นคือ hub จะส่ง packets ออกไปยังทุกport ของ hub ยกเว้น port ที่ hub ได้รับ packet เข้ามา

```

X "Node: h1"
root@mininet-vm:~# ping -c1 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data:
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=16.5 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 16.535/16.535/16.535/0.000 ms

```

```

Node: h2
root@mininet-vm:~# tcpdump -xx -n -i h2-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h2-eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
17:41:33.666681 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 2411, seq 1, length
64
    0x0000: 267b cd89 c355 7a27 70b3 3fde 0800 4500
    0x0010: 0054 9cad 4000 4001 89f9 0a00 0001 0a00
    0x0020: 0002 0800 9fd2 096b 0001 4dab 9556 77bc
    0x0030: 0900 0809 0a0b 0c0d 0e0f 1011 1213 1415
    0x0040: 1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
    0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
    0x0060: 3637
17:41:33.666741 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 2411, seq 1, length
64
    0x0000: 7a27 70b3 3fde 267b cd89 c355 0800 4500
    0x0010: 0054 efd2 0000 4001 76d4 0a00 0002 0a00
    0x0020: 0001 0000 a7d2 096b 0001 4dab 9556 77bc
    0x0030: 0900 0809 0a0b 0c0d 0e0f 1011 1213 1415
    0x0040: 1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
    0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
    0x0060: 3637
17:41:38.649206 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
    0x0000: 267b cd89 c355 7a27 70b3 3fde 0806 0001
    0x0010: 0800 0604 0001 7a27 70b3 3fde 0a00 0001
    0x0020: 0000 0000 0000 0a00 0002
17:41:38.649235 ARP, Reply 10.0.0.2 is-at 26:7b:cd:89:c3:55, length 28
    0x0000: 7a27 70b3 3fde 267b cd89 c355 0806 0001
    0x0010: 0800 0604 0002 267b cd89 c355 0a00 0002
    0x0020: 7a27 70b3 3fde 0a00 0001

```

```

Node: h3
root@mininet-vm:~# tcpdump -xx -n -i h3-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h3-eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
17:41:33.666677 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 2411, seq 1, length
64
    0x0000: 267b cd89 c355 7a27 70b3 3fde 0800 4500
    0x0010: 0054 9cad 4000 4001 89f9 0a00 0001 0a00
    0x0020: 0002 0800 9fd2 096b 0001 4dab 9556 77bc
    0x0030: 0900 0809 0a0b 0c0d 0e0f 1011 1213 1415
    0x0040: 1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
    0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
    0x0060: 3637
17:41:33.669220 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 2411, seq 1, length
64
    0x0000: 7a27 70b3 3fde 267b cd89 c355 0800 4500
    0x0010: 0054 efd2 0000 4001 76d4 0a00 0002 0a00
    0x0020: 0001 0000 a7d2 096b 0001 4dab 9556 77bc
    0x0030: 0900 0809 0a0b 0c0d 0e0f 1011 1213 1415
    0x0040: 1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
    0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
    0x0060: 3637
17:41:38.649204 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
    0x0000: 267b cd89 c355 7a27 70b3 3fde 0806 0001
    0x0010: 0800 0604 0001 7a27 70b3 3fde 0a00 0001
    0x0020: 0000 0000 0000 0a00 0002
17:41:38.650248 ARP, Reply 10.0.0.2 is-at 26:7b:cd:89:c3:55, length 28
    0x0000: 7a27 70b3 3fde 267b cd89 c355 0806 0001
    0x0010: 0800 0604 0002 267b cd89 c355 0a00 0002
    0x0020: 7a27 70b3 3fde 0a00 0001

```

ตรวจสอบประสิทธิภาพการทำงานของ controller ที่ถูกกำหนดให้ทำหน้าที่เป็น Hub

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

ใช้คำสั่ง iperf เพื่อทดสอบสมรรถนะการส่งข้อมูลของ switch s1

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h2 and h1
*** Results: ['1.49 Gbits/sec', '1.49 Gbits/sec']
```

ต่อไปจะให้ POX controller ทำหน้าที่เป็น Layer 2 learning switch โดยดำเนินการดังนี้:

จาก ที่หน้าจอเดิมของ POX controller ที่ทำหน้าที่เป็น Hub ตามรูปข้างล่าง

```
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
DEBUG:core:POX 0.2.0 (carp) going up...
DEBUG:core:Running on CPython (2.7.6/Mar 22 2014 22:59:38)
DEBUG:core:Platform is Linux-3.13.0-24-generic-i686-with-Ubuntu-14.04-trusty
INFO:core:POX 0.2.0 (carp) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
DEBUG:misc.of tutorial:Controlling [00-00-00-00-00-01 1]
```

ให้พิมพ์ `ctrl + C` จะได้หน้าจอตามรูปข้างล่าง

```
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
DEBUG:misc.of tutorial:Controlling [00-00-00-00-00-01 1]
^CINFO:core:Going down...
INFO:openflow.of_01:[00-00-00-00-00-01 1] disconnected
INFO:core:Down.
mininet@mininet-vm:~$
```

POX sdn controller สั่งให้ switch S1 ทำงาน เป็น Layer 2 learning switch

มาที่หน้าจอ putty ที่โดยใช้คำสั่งดังนี้ ที่ SSH terminal

```
mininet@mininet-vm:~$ sudo ~/pox/pox.py forwarding.l2_learning
```

อย่างไรก็ตาม switch อาจใช้เวลาสักครู่ เพื่อเชื่อมต่อไปยัง controller และให้คอยจนกว่า switch S1 เชื่อมกับ controller โดยหน้าจอของ POX บนจอ SSH terminal จะแสดงดังรูป

```
mininet@mininet-vm:~$ sudo ~/pox/pox.py forwarding.l2_learning
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:core:POX 0.2.0 (carp) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
```

ตรวจสอบประสิทธิภาพการทำงานของ POX controller ที่ถูกกำหนดให้ทำหน้าที่เป็น layer 2 learning switch

```
mininet> pingall
*** Ping: testing ping reachability
h3 -> h2 h1
h2 -> h3 h1
h1 -> h3 h2
*** Results: 0% dropped (6/6 received)
```

คำถามให้ตรวจสอบด้วยว่า switch S1 มีการ flood ข้อมูลหรือไม่ เมื่อ switch s1 ถูกสั่งด้วย POX SDN controller ให้ทำงานเป็น Layer 2 Learning Switch

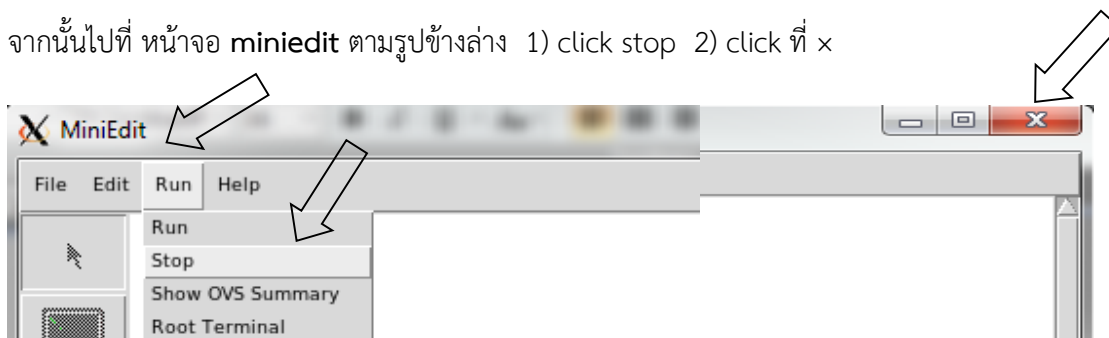
ใช้คำสั่ง iperf เพื่อทดสอบสมรรถนะการส่งข้อมูลของ switch s1

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h2 and h1
*** Results: ['5.96 Gbits/sec', '5.96 Gbits/sec']
```

มาที่ หน้าจอ putty ให้ออกจาก mininet

```
mininet> exit
```

จากนั้นไปที่ หน้าจอ miniedit ตามรูปข้างล่าง 1) click stop 2) click ที่ x



ที่หน้าจอ putty สั่งหยุดการทำงานของ mininet server

```
mininet@mininet-vm:~$ sudo shutdown -h now
```

คำถาม ด้วยโครงข่ายเสมือนที่ถูกสร้างขึ้นด้วยคำสั่ง “sudo mn -topo=single,3 -mac -switch=ovsk - controller=remote“ เชื่อมต่อกับ controller 3 ชนิด tcp:6633, POX ทำหน้าที่เป็น Hub ,และ POX ทำหน้าที่เป็น layer 2 learning switch ท่านคิดว่า controller ชนิดใดที่ทำให้ switch s1 ทำงานมีประสิทธิภาพมากที่สุด และทำไมจึงเป็นเช่นนั้น

### เอกสารอ้างอิง

1. <http://mininet.org/> แนะนำการติดตั้ง พร้อมตัวอย่าง การใช้งานเริ่มต้น
2. <https://github.com/mininet/mininet/wiki/Introduction-to-Mininet> แนะนำ Mininet
3. <https://github.com/mininet/openflow-tutorial/wiki> แนะนำ openflow-tutorial
4. N. McKeown, T. Anderson, H. Balakrishnan, G. ParulKar, L. Peterson, J. Rexford, S. Shenker, J. Turner, “ OpenFlow: Enabling Innovation in the Campus Network”, ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, p. 6, 2008.