SDN Lab1

TA: 王彥錚
Lab: ED817
Email: luke010708@gmail.com

Outline

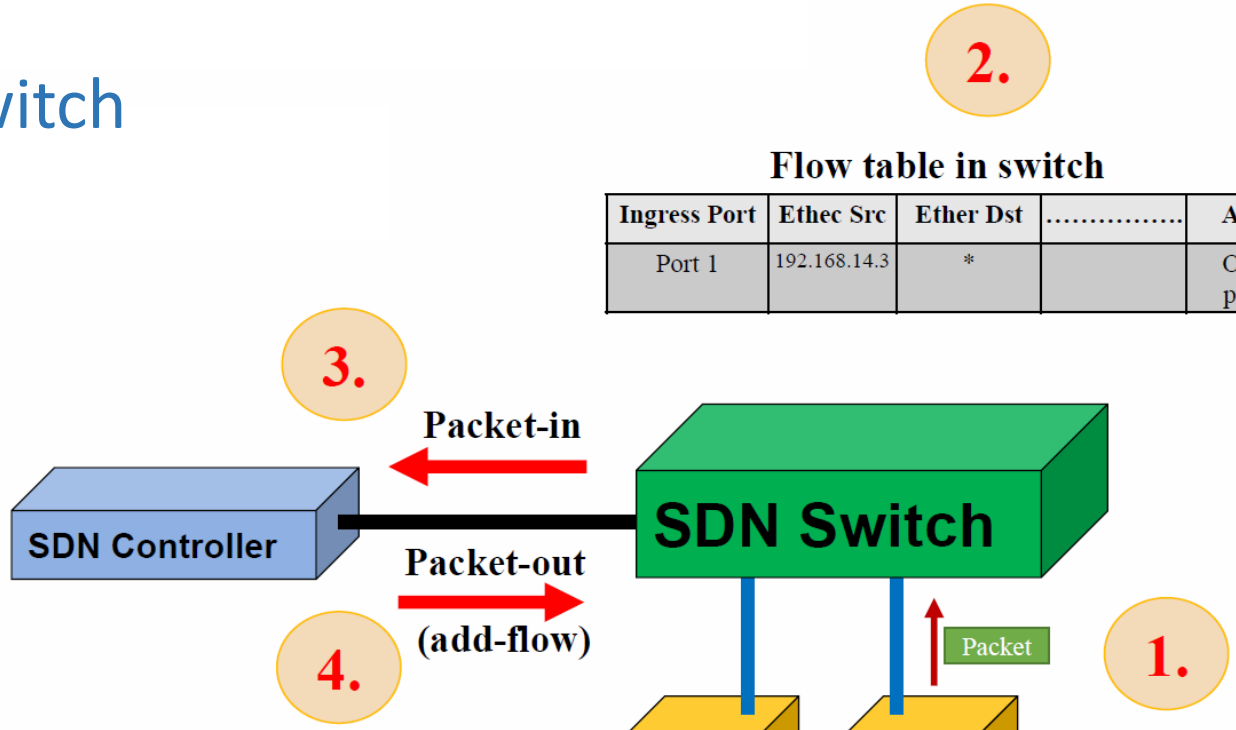
- Goals
- Objectives
- Lab Content
- Instructions
- Reference

Goals

- Learn how to use Virtual Machine and be familiar with Ubuntu Linux
- Learn basic ideas of Software Defined Network (SDN) and SDN switches
- Learn how to use Mininet to create a simple network topology
- Learn how to construct SDN controller Ryu and use it to control flows on the created topology
- Learn how to create network topology in Mininet
- Learn how to SDN controller Ryu to monitor the created network system

Objectives

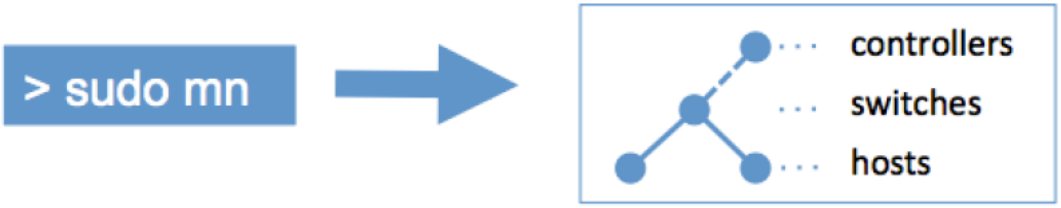
- SDN switch



1. Server 1 wants to send a packet to server 2.
2. First, SDN switch will check the flow table.
3. If there is no match in flow table, switch will forward packet to controller (packet-in)
4. Controller will decide the action and send packet back to switch (packet-out)

Objectives

- Mininet
  - It creates a realistic virtual network, running real kernel, switch and application code
  - It runs a collection of end-hosts, switches, routers, and links on a single Linux kernel
  - The created switches are OpenFlow-enabled



Objectives

- SDN controller RYU
  - RYU supports the OpenFlow1.0, 1.2, 1.3 and 1.4
  - RYU can work in conjunction with OpenStack for cloud computing
  - Written in Python

Lab Content

- Demo 1
  - Step 1: Create a VM which runs Ubuntu
  - Step 2: Install Mininet
  - Step 3: Install Ryu
  - Step 4: Run Mininet and Ryu to emulate a simple SDN network system
  - Step 5: Study SDN controller's sample code

Lab Content

- Demo 2
  - Step 1: Create a specific topology network system in Mininet
  - Step 2: Modify the SDN controller code based on “simple_switch_13.py” to make your controller be able to monitor the switches
  - Step 3: Show the Layer 2 address table of the switches

Demo 1 Instructions

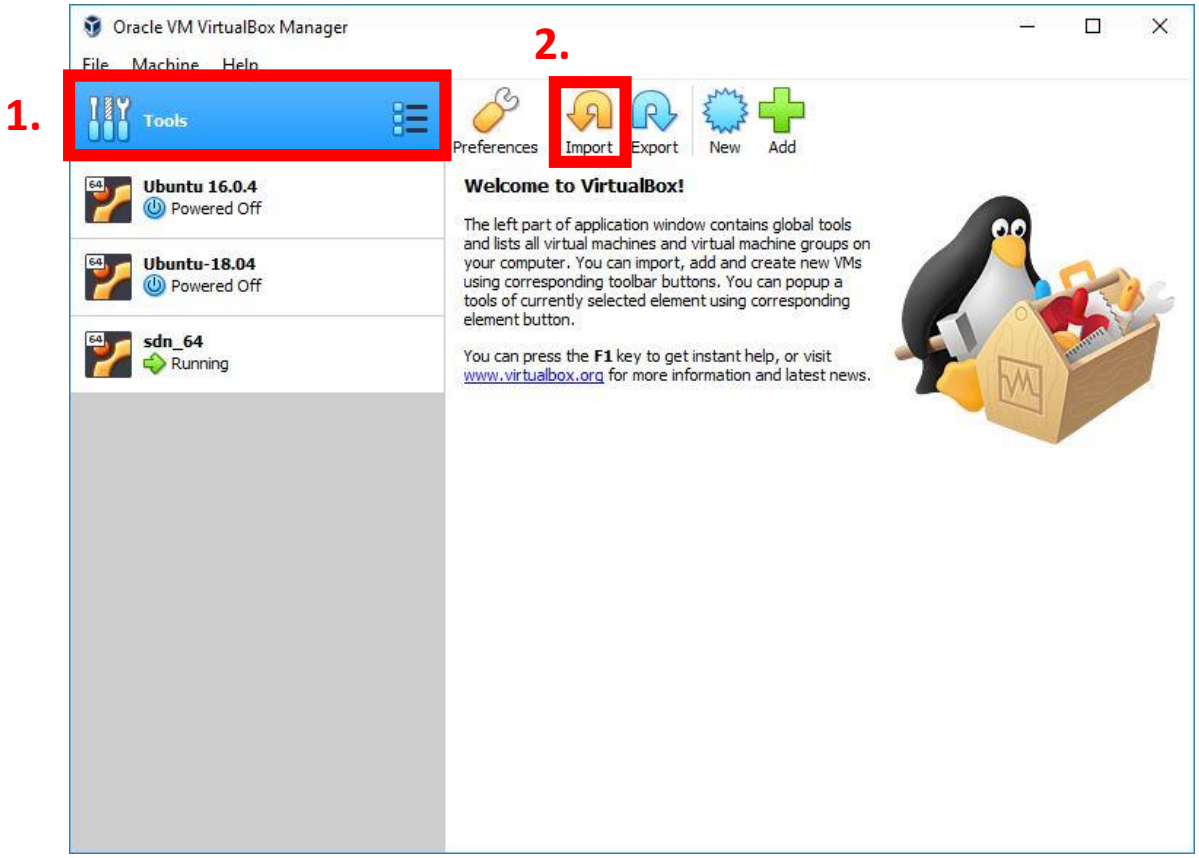
- Step 1: Create a VM which runs Ubuntu
  - Download the latest version of Virtual Box
  - Download the latest version of Ubuntu
  - Install them step-by-step
  - For created VM at ED713

    user name: sdn

    password: mininet
  - Need to wait several minutes for system configuration at first time startup

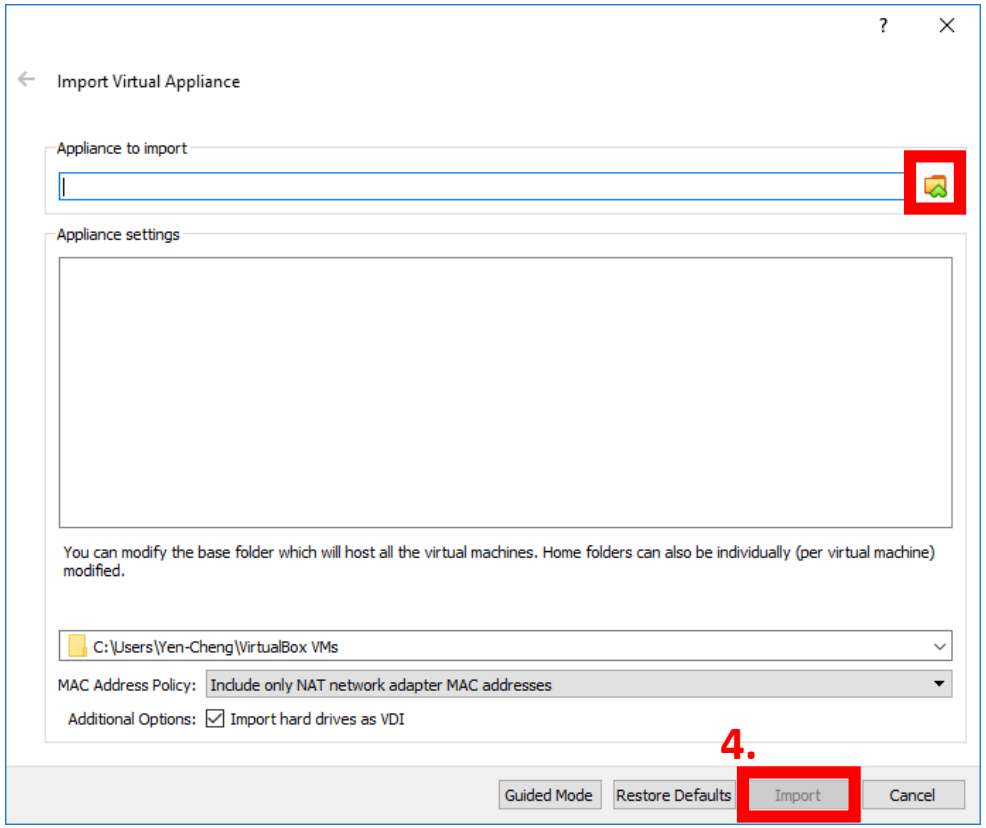
Demo 1 Instructions

- Step 1: Create a VM which runs Ubuntu



Demo 1 Instructions

- Step 1: Create a VM which runs Ubuntu



3. Find sdn_64.ova

Demo 1 Instructions

- Step 2: Install Mininet
  - Install git

    sudo apt-get install -y git
  - Download mininet by git

    sudo git clone git://github.com/mininet/mininet
  - Install mininet (it will take 3~7 minutes)

    cd mininet/util

    ./install.sh -a
  - Test

    sudo mn --test pingall

Demo 1 Instructions

- Step 2: Install Mininet
  - You should see the following screen for a successful install

```
sdn@ubuntu:~/mininet/util$ sudo mn --test pingall
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 5.614 seconds
sdn@ubuntu:~/mininet/util$
```

Demo 1 Instructions

- Step 3: Install Ryu
  - Install the following packages

    sudo apt-get install -y python3-pip python-dev build-essential

    sudo pip3 install --upgrade pip

    sudo pip3 install --upgrade six

    sudo apt-get install -y python-eventlet python-routes

    sudo apt-get install -y python-webob python-paramiko

    pip3 install --upgrade setuptools
  - Download and install Ryu

    sudo git clone git://github.com/osrg/ryu.git

    cd ryu

    sudo pip3 install .
  - Test

    ryu-manager
  - Press “Ctrl+C” to leave ryu-manger

Demo 1 Instructions

- Step 3: Install Ryu
  - If you can see this output, then Ryu is installed

```
sdn@ubuntu:~/mininet/util$ ryu-manager
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Demo 1 Instructions

- Step 4: Run Mininet and Ryu to emulate a simple SDN network system
  - Run mininet and create a tree topology with depth=3

    sudo mn --controller remote,ip=127.0.0.1 --topo tree,depth=3

```
sdn@ubuntu:~/mininet/util$ ryu-manager
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
^Csdn@ubuntu:~/mininet/util$ sudo mn --controller remote,ip=127.0.0.1 --topo tree,depth=3
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6) (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet>
```

Demo 1 Instructions

- Step 4: Run Mininet and Ryu to emulate a simple SDN network system
  - Open another new terminal and go to Ryu installation folder
    - Get the Ryu installation directory:

      pip3 show ryu
    - Go to the specified directory, e.g.:

      cd /usr/local/lib/python3.5/dist-packages

```
tim@VBox1:~$ pip3 show ryu
Name: ryu
Version: 4.34
Summary: Component-based Software-defined Networking Framework
Home-page: http://osrg.github.io/ryu/
Author: Ryu project team
Author-email: ryu-devel@lists.sourceforge.net
License: Apache License 2.0
Location: /usr/local/lib/python3.5/dist-packages
Requires: six, msgpack, tinyrpc, webob, routes, oslo.config, ovs, netaddr, eventlet
Required-by:
tim@VBox1:~$ cd /usr/local/lib/python3.5/dist-packages
tim@VBox1:/usr/local/lib/python3.5/dist-packages$
```

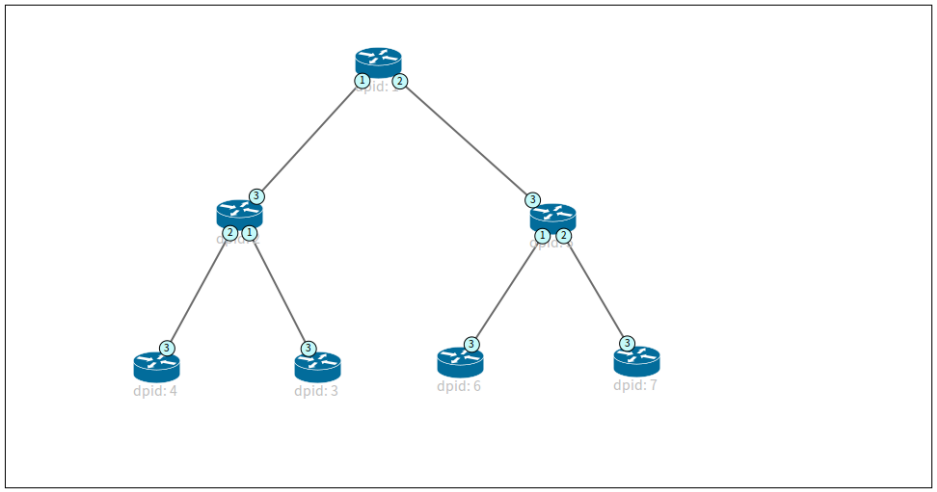
Demo 1 Instructions

- Step 4: Run Mininet and Ryu to emulate a simple SDN network system
  - Run the Ryu topology viewer

PYTHONPATH=. ryu run --observe-links ryu/app/gui_topology/gui_topology.py

  - Open the web browser

http://127.0.0.1:8080



Demo 1 Instructions

- Step 4: Run Mininet and Ryu to emulate a simple SDN network system
  - Press “Ctrl+C” to stop the topology viewer
  - Enter following command to make switches supports OF 1.3 (in this topology we have 7 switches s1~s7)

```
sudo ovs-vsctl set bridge s1 protocols=OpenFlow13
sudo ovs-vsctl set bridge s2 protocols=OpenFlow13
sudo ovs-vsctl set bridge s3 protocols=OpenFlow13
sudo ovs-vsctl set bridge s4 protocols=OpenFlow13
sudo ovs-vsctl set bridge s5 protocols=OpenFlow13
sudo ovs-vsctl set bridge s6 protocols=OpenFlow13
sudo ovs-vsctl set bridge s7 protocols=OpenFlow13
```

Demo 1 Instructions

- Step 4: Run Mininet and Ryu to emulate a simple SDN network system
  - Run controller sample code: simple_switch_13.py

    ryu-manager ryu/app/simple_switch_13.py
  - *Go back to terminal with Mininet*

    pingall

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
mininet>
```

Demo 1 Instructions

- Step 5: Study SDN controller's sample code
  - Open simple_switch_13.py
  - Study the sample code with reference, Ryubook.pdf (Chap.2)
  - TA will check Step 1 to Step 4 and ask some simple questions about the sample code

Demo 2 Instructions

- Step 1: Create a specific topology in Mininet
  - Find the simple custom topology script in mininet at " ~/mininet/custom/topo-2sw-2host.py "
  - There are some useful commands:

    addHost, addSwitch, addLink
  - Ex: "self.addLink(sw1,sw2,bw=10,loss=10)" means add a link with a bandwidth of 10 Mbps, and 10% packet loss rate

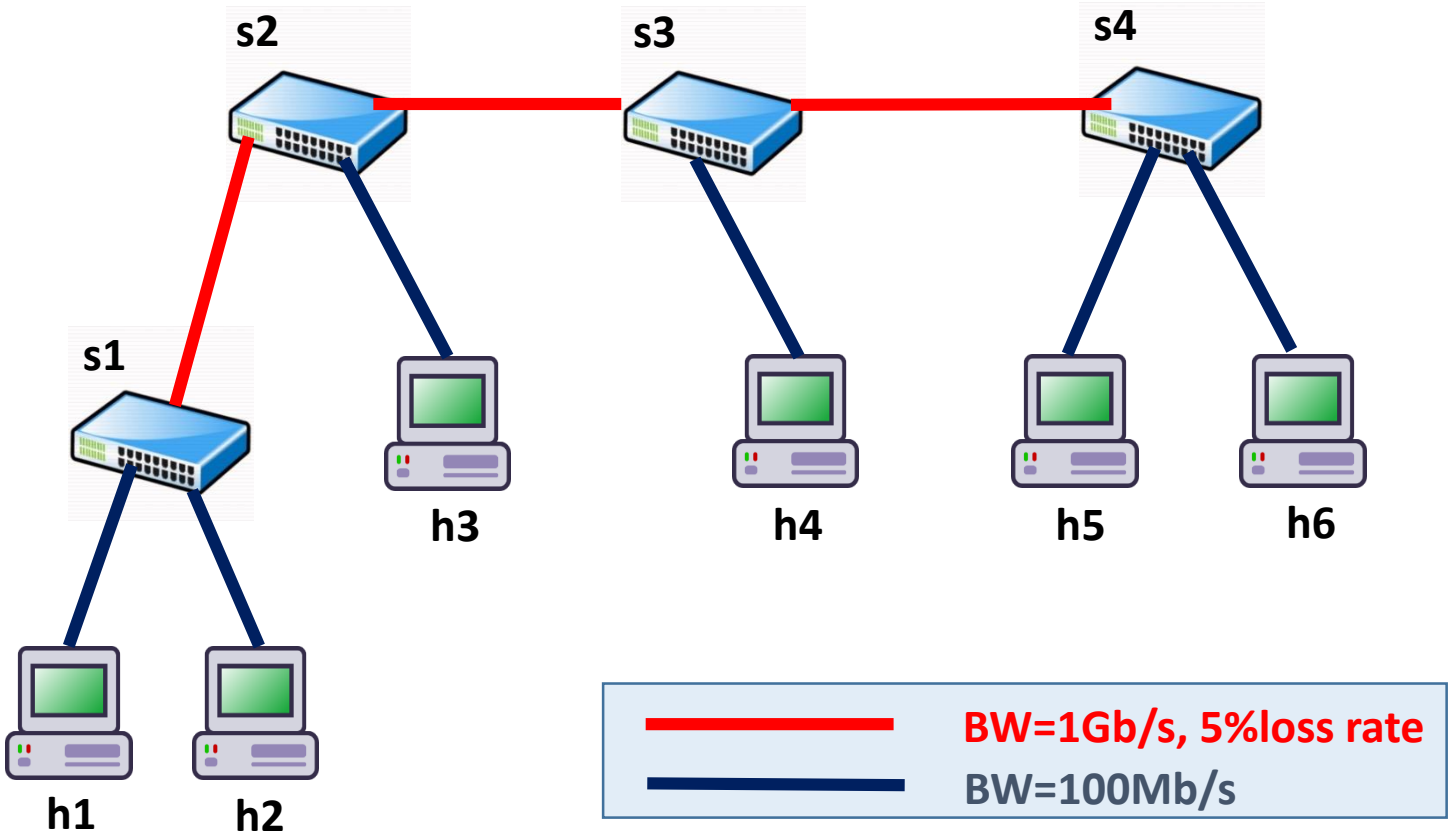
Demo 2 Instructions

- Step 1: Create a specific topology in Mininet
  - Understand the sample script and write your own topology script
  - "--custom" means use custom topology
  - "--topo" means use topology "mytopo" from the dictionary "topos" in the script
  - "--link" means use traffic control link
  - Use the following command to create your topology:

sudo mn --topo mytopo --custom ~/mininet/custom/yourscript.py --controller remote --switch default,protocols=OpenFlow13 --link=tc

Demo 2 Instructions

- Step 1: Create a specific topology in Mininet



Demo 2 Instructions

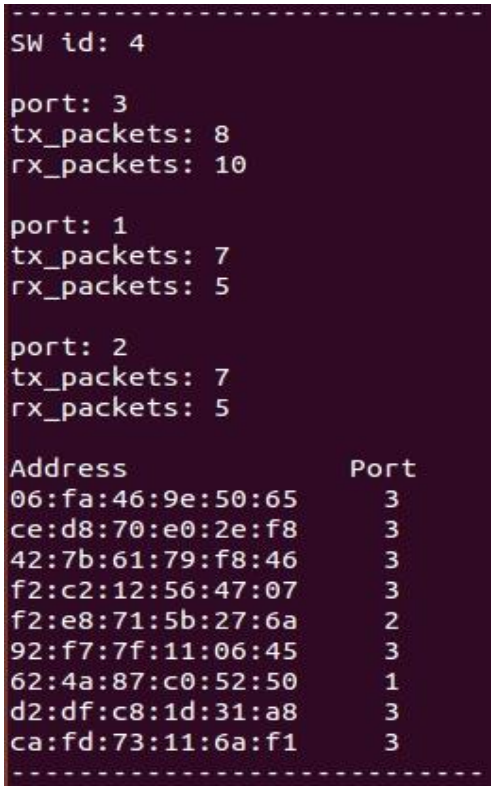
- Step 2: Modify the SDN controller code
  - Create a thread to monitor the traffic of all the switches every 5 seconds [1]
  - You will have to use "OFPPortStatsRequest( )", "OFPPortStatsReply( )" to get the switch information [2]
  - Run your code by the command:

    "ryu-manager yourcode.py"

Reference :
[1] Chapter 3 of Ryubook: http://osrg.github.io/ryu-book/en/Ryubook.pdf
[2] http://ryu.readthedocs.org/en/latest/ofproto_v1_3_ref.html#multipart-messages

Demo 2 Instructions

- Step 2: Modify the SDN controller code
  - Print the address table and monitor information of all switches every 5 seconds including:
    1. Switch IDs
    2. Port numbers
    3. Number of transmitted and received packets of each port
    4. Layer 2 address table of the switch (Hint: In simple_switch_13.py)



Reference

- VMware player 6.0.7: https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/6_0|PLAYER-607
- Ubuntu: https://www.ubuntu-tw.org/modules/tinyd0/
- Mininet: http://mininet.org/
- Ryu: http://osrg.github.io/ryu/
- Ryu book: http://osrg.github.io/ryu-book/en/Ryubook.pdf