Lab No: 05
Lab Report Name : Software Defined Network (SDN Mininet with Ryu Controller)
Name : Shuvo Biswas
ID : IT-16014

Objectives :

The objective of the lab 5 is to:
 Install and use traffic generators as powerful tools for testing network performance.
 Install and configure SDN Controller
 Install and understand how the mininet simulator works
 Implement and run basic examples for understanding the role of the controller and how it interact with mininet

Theory :

2.1. Traffic Generator:

What is iPerf?: iPerf is a tool for active measurements of the maximum achievable bandwidth on IP networks. It supports tuning of various parameters related to timing, buffers and protocols (TCP, UDP, SCTP with IPv4 and IPv6). For each test it reports the bandwidth, loss, and other parameters.

2.2. Software Defined Networking:

Software-defined networking was pioneered between 2008 and 2011 by work done at Stanford University and the Nicira Company.

that by separating control of network functions from hardware devices, administrators acquire more power to route and direct traffic in response to changing requirements.

Software-Defined vs. Traditional Networking: The key difference between traditional and software-defined networking is how SDNs handle data packets. In a traditional network, the way a switch handles an incoming data packet is written into its firmware. Most switches — particularly those used in commercial data centers rather than enterprise environments — respond to and route all packets the same way. SDN provides admins with granular control over the way switches handle data, giving them the ability to automatically prioritize or block certain types of packets. This, in turn, allows for greater efficiency without the need to invest in expensive, application-specific network switches

Benefits of Software-Defined Networking:

There are several benefits to the more advanced level of control afforded by implementing SND in a multi-tenant network environment:

- ***Automation:*** SND allows for automation of complex operational tasks that make networks faster, more efficient and easier to manage.
- ***Increased uptime:*** SDN has proven effective in reducing deployment and configuration errors that can lead to service disruptions.
- ***Less drain on resources:*** SDN gives administrators control over how their routers and switches operate from a single, virtual workflow. This frees up key staff to focus on more important tasks.
- ***Better visibility:*** With SDN, system administrator's gain improved visibility into overall network function, allowing them to allocate resources more effectively.
- ***Cost savings:*** SND can lead to significant overall costs savings. It also reduces the amount of spending required on infrastructure by allowing data centers to get the most use of their existing devices.

Methodology :

TIP: For getting extra space in your USB-please use the following tips:

- Empty the trash
- Delete the Android related staff
- Delete the extras for other courses
- Delete the already installed package sources

Install iperf :

1. Open the *Synaptic Package Manager* (Navigator ->System-> *Synaptic Package Manager*)
2. Setup the proxy:

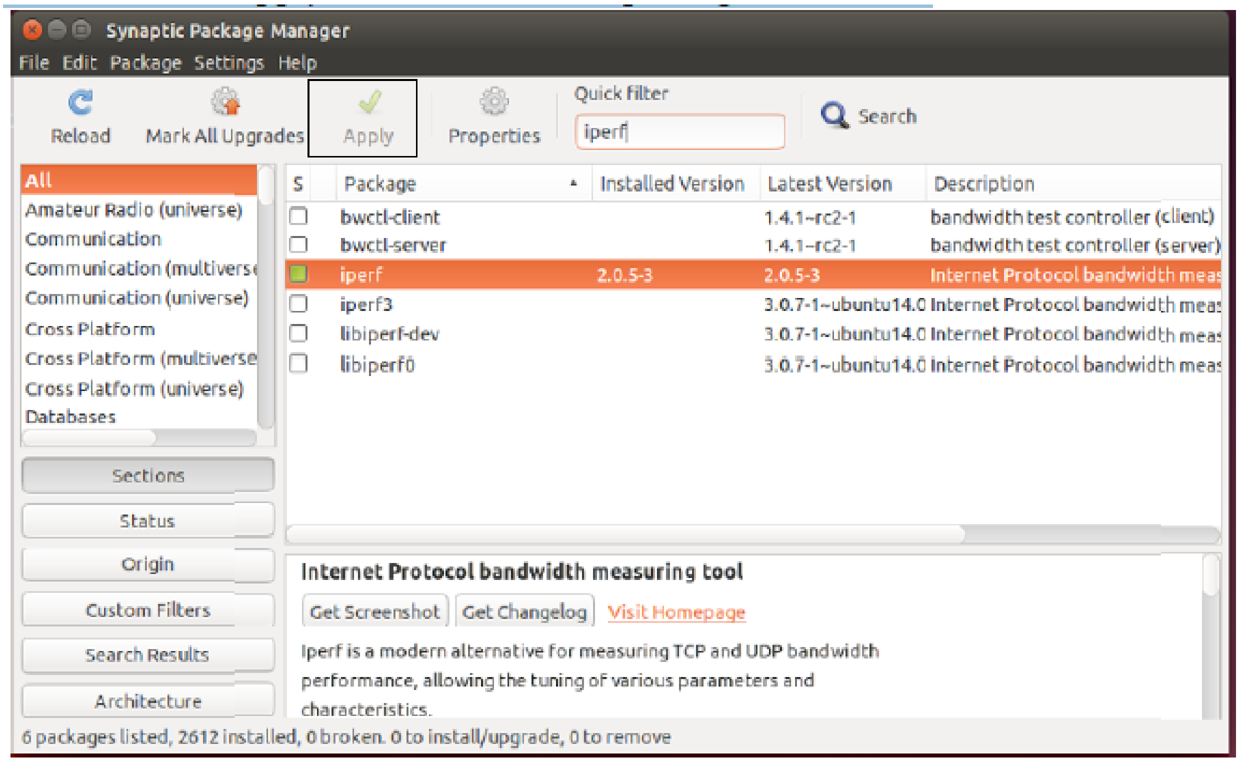
o Click on *settings-> Preference -> Network* o Click on *manual proxy configuration*

o HTT and FTP Proxy: proxy.rmit.edu.au Port: 8080

3. Search for *Quick filter `iperf`*

4. Click on *Mark for installation*

5. Then click on *Apply* and wait until the package is installed



Quick filter: iperf

S	Package	Installed Version	Latest Version	Description
□	bwctl-client		1.4.1~rc2-1	bandwidth test controller (client)
□	bwctl-server		1.4.1~rc2-1	bandwidth test controller (server)
🟩	iperf	2.0.5-3	2.0.5-3	Internet Protocol bandwidth meas
□	iperf3		3.0.7-1~ubuntu14.0	Internet Protocol bandwidth meas
□	libiperf-dev		3.0.7-1~ubuntu14.0	Internet Protocol bandwidth meas
□	libiperf0		3.0.7-1~ubuntu14.0	Internet Protocol bandwidth meas

Internet Protocol bandwidth measuring tool

Iperf is a modern alternative for measuring TCP and UDP bandwidth performance, allowing the tuning of various parameters and characteristics.

6 packages listed, 2612 installed, 0 broken. 0 to install/upgrade, 0 to remove

Install mininet

1. In *Synaptic Package*

2. Search for *Quick filter `mininet`*

3. Click on *Mark for installation*

4. Then click on *Apply* and wait until the package is installed

Install Controller

OVS controller:

1. In *Synaptic Package*

2. Search for *Quick filter `openvswitch-controller`*

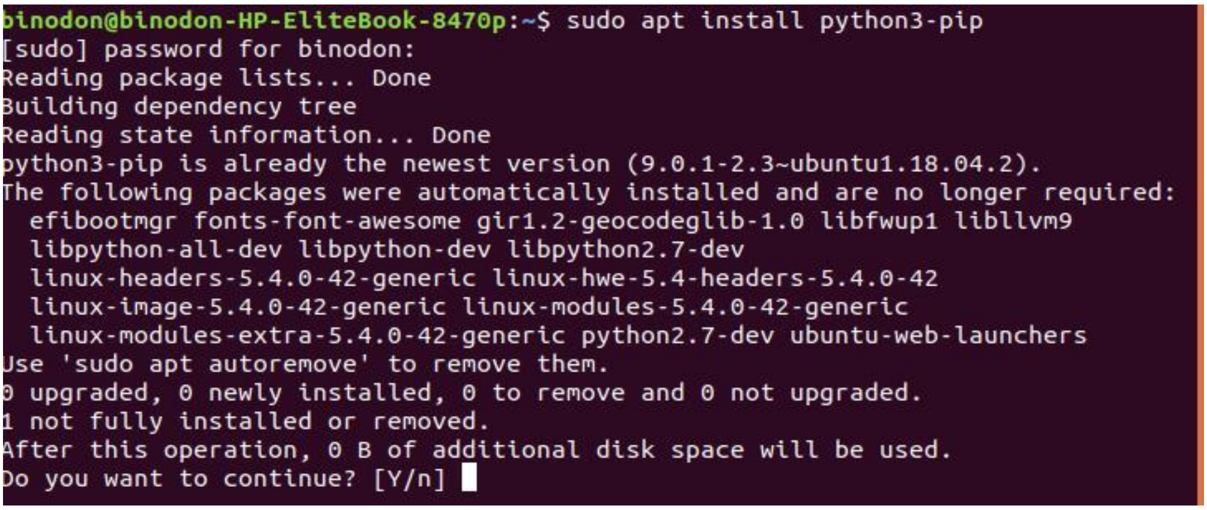
3. Click on *Mark for installation*

4. Then click on *Apply* and wait until the package is installed

Install Ryu Controller

1.To install Ryu Controller Firstly,We need to install Python in our system

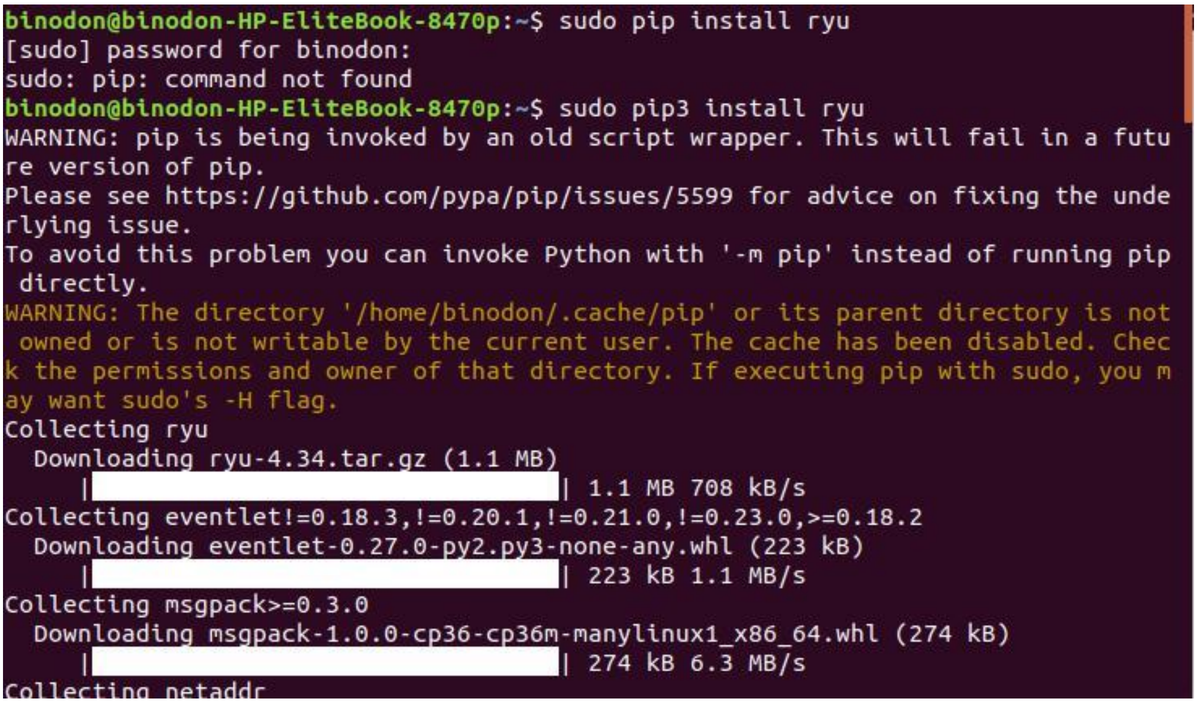
$ sudo apt install python3-pip



```
binodon@binodon-HP-EliteBook-8470p:~$ sudo apt install python3-pip
[sudo] password for binodon:
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-pip is already the newest version (9.0.1-2.3~ubuntu1.18.04.2).
The following packages were automatically installed and are no longer required:
  efibootmgr fonts-font-awesome gir1.2-geocodeglib-1.0 libfwup1 libllvm9
  libpython-all-dev libpython-dev libpython2.7-dev
  linux-headers-5.4.0-42-generic linux-hwe-5.4-headers-5.4.0-42
  linux-image-5.4.0-42-generic linux-modules-5.4.0-42-generic
  linux-modules-extra-5.4.0-42-generic python2.7-dev ubuntu-web-launchers
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
1 not fully installed or removed.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] █
```

2.Now we may install Ryu Controller by following instruction

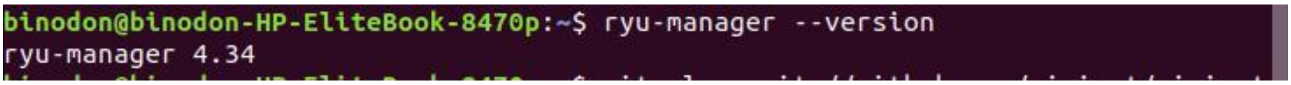
$ sudo pip3 install ryu



```
binodon@binodon-HP-EliteBook-8470p:~$ sudo pip install ryu
[sudo] password for binodon:
sudo: pip: command not found
binodon@binodon-HP-EliteBook-8470p:~$ sudo pip3 install ryu
WARNING: pip is being invoked by an old script wrapper. This will fail in a futu
re version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the unde
rlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running pip
 directly.
WARNING: The directory '/home/binodon/.cache/pip' or its parent directory is not
 owned or is not writable by the current user. The cache has been disabled. Chec
k the permissions and owner of that directory. If executing pip with sudo, you m
ay want sudo's -H flag.
Collecting ryu
  Downloading ryu-4.34.tar.gz (1.1 MB)
     |████████████████████████████████| 1.1 MB 708 kB/s
Collecting eventlet!=0.18.3,!=0.20.1,!=0.21.0,!=0.23.0,>=0.18.2
  Downloading eventlet-0.27.0-py2.py3-none-any.whl (223 kB)
     |████████████████████████████████| 223 kB 1.1 MB/s
Collecting msgpack>=0.3.0
  Downloading msgpack-1.0.0-cp36-cp36m-manylinux1_x86_64.whl (274 kB)
     |████████████████████████████████| 274 kB 6.3 MB/s
Collecting netaddr
```

3.After Completing installation Now We can check the version:

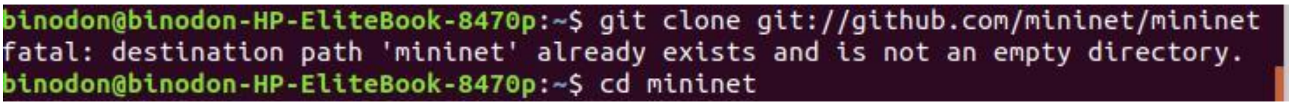
$ ryu-manager --version



```
binodon@binodon-HP-EliteBook-8470p:~$ ryu-manager --version
ryu-manager 4.34
```

Install Mininet:

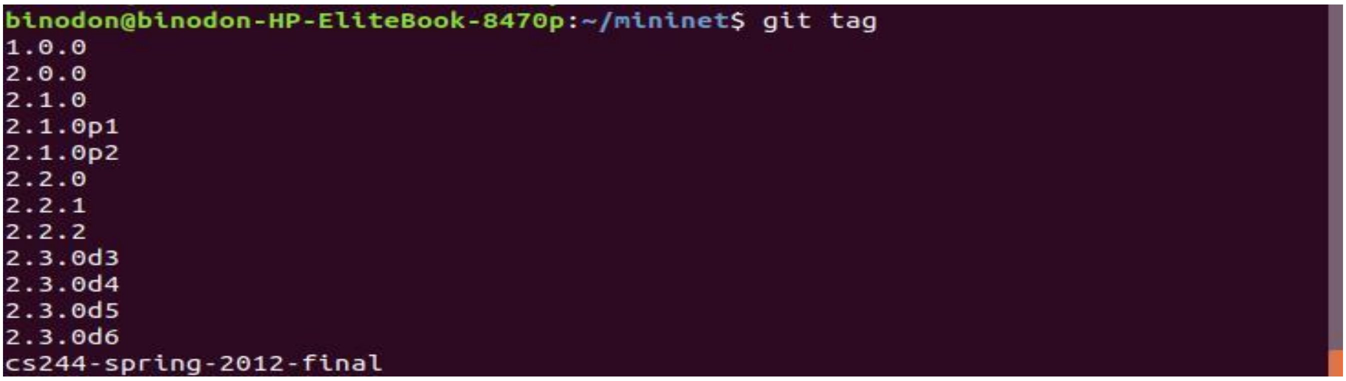
1.$ git clone git://github.com/mininet/mininet



```
binodon@binodon-HP-EliteBook-8470p:~$ git clone git://github.com/mininet/mininet
fatal: destination path 'mininet' already exists and is not an empty directory.
binodon@binodon-HP-EliteBook-8470p:~$ cd mininet
```

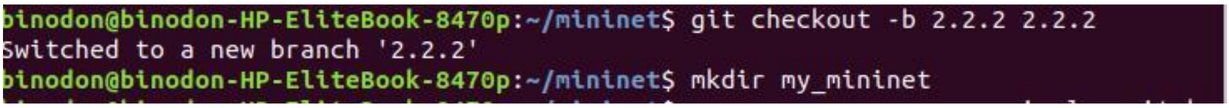
Mininet Already exists in my system

2.$ git tag



```
binodon@binodon-HP-EliteBook-8470p:~/mininet$ git tag
1.0.0
2.0.0
2.1.0
2.1.0p1
2.1.0p2
2.2.0
2.2.1
2.2.2
2.3.0d3
2.3.0d4
2.3.0d5
2.3.0d6
cs244-spring-2012-final
```

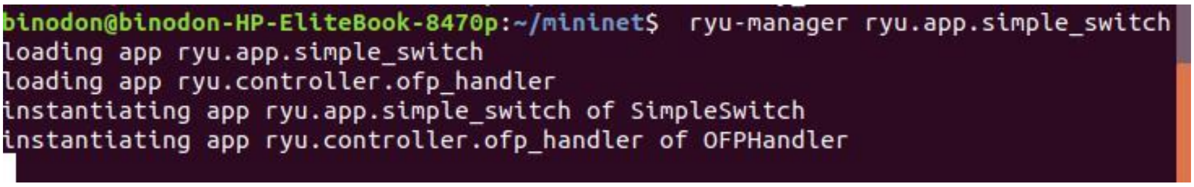
3.$ git checkout -b 2.2.2 2.2.2



```
binodon@binodon-HP-EliteBook-8470p:~/mininet$ git checkout -b 2.2.2 2.2.2
Switched to a new branch '2.2.2'
binodon@binodon-HP-EliteBook-8470p:~/mininet$ mkdir my_mininet
```

Run Ryu Controller

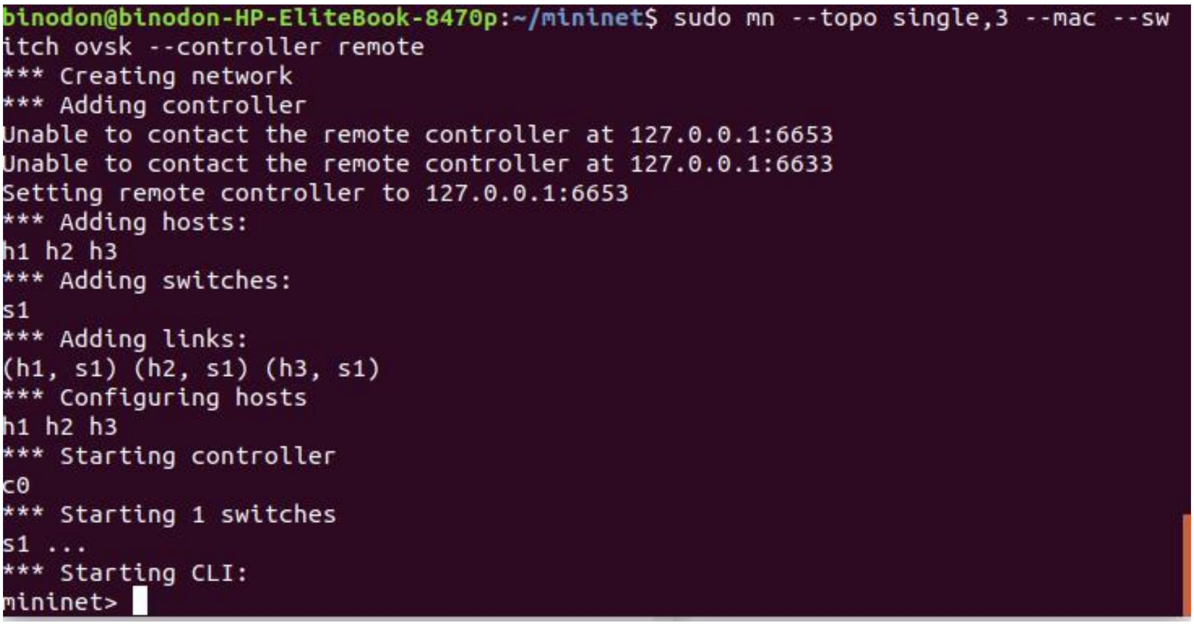
$ ryu-manager ryu.app.simple_switch



```
binodon@binodon-HP-EliteBook-8470p:~/mininet$  ryu-manager ryu.app.simple_switch
loading app ryu.app.simple_switch
loading app ryu.controller.ofp_handler
instantiating app ryu.app.simple_switch of SimpleSwitch
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Run Mininet topology

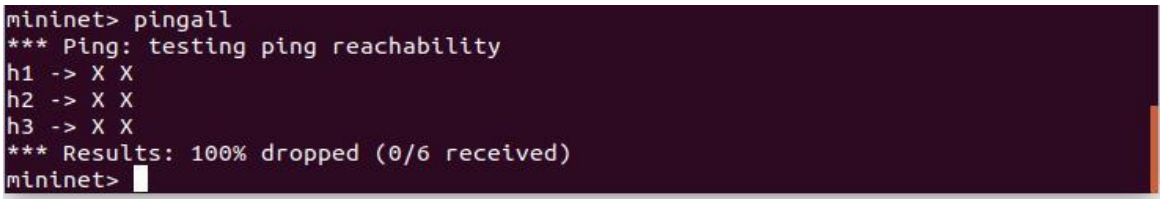
$ sudo mn --topo single,3 --mac --switch ovsk --controller remote



```
binodon@binodon-HP-EliteBook-8470p:~/mininet$ sudo mn --topo single,3 --mac --sw
itch ovsk --controller remote
*** 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
*** 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>
```

Application:

mininet> pingall



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X X
h3 -> X X
*** Results: 100% dropped (0/6 received)
mininet>
```

Conclusion:

In this lab we will learn basically in Software-defined networking most switches — particularly those used in commercial data centers rather than enterprise environments — respond to and route all packets the same way. SDN provides admins with granular control over the way switches handle data, giving them the ability to automatically prioritize or block certain types of packets.