Lab 9 (Week 10)
SDN Lab using Ryu and Mininet

CAN201

Dr. Gordon Boateng

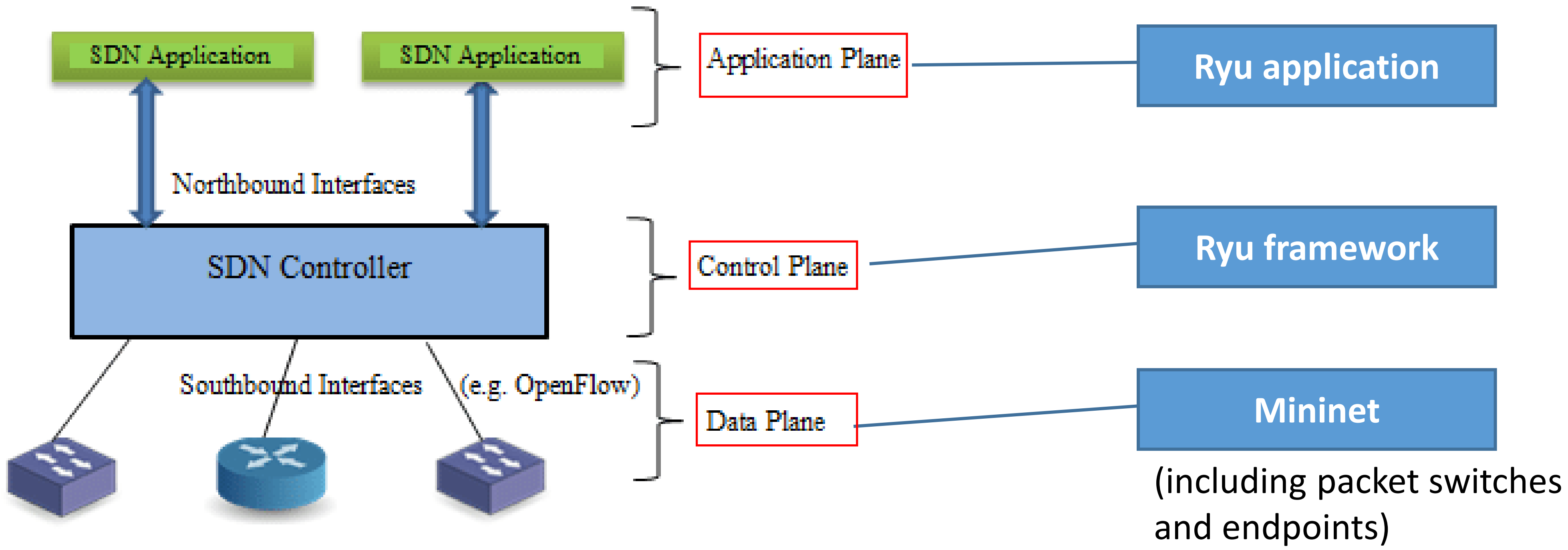
Dr. Fei Cheng

Outline

- Recap the SDN architecture
- SDN Controller - Ryu
- Integrating Ryu with Mininet
- Hands-on Practice

Recap the SDN architecture

Design **Implementation**

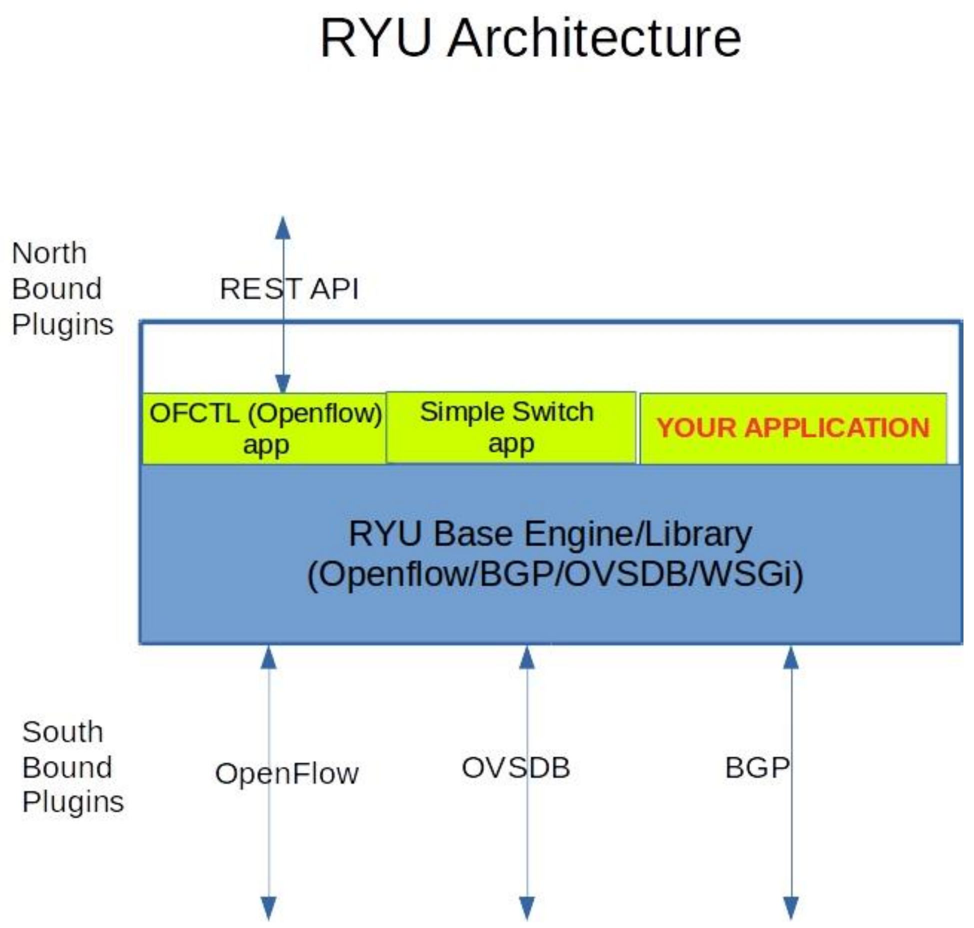


Outline

- Recap the SDN architecture
- SDN Controller - Ryu
- Integrating Ryu with Mininet
- Hands-on Practice

SDN Controller - Ryu

1. “Ryu” means “flow” in Japanese.
2. Ryu supports OpenFlow 1.0 – 1.5
3. Python program
4. Well-documented



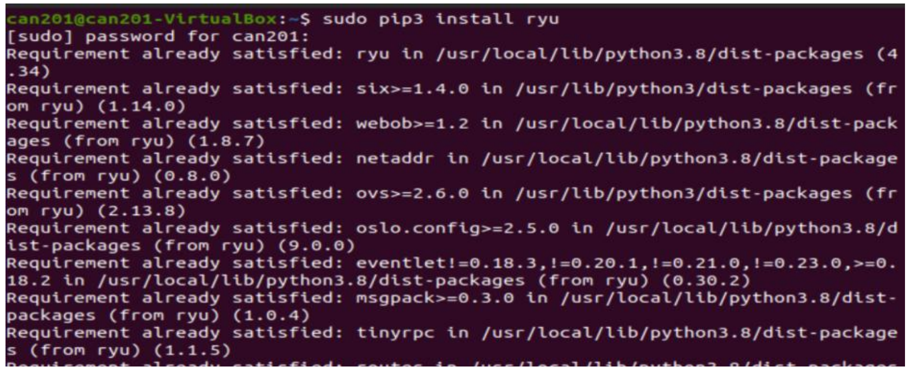
How to install Ryu

- Install Ryu (The given Ubuntu OVA file has Ryu installed; you do not need to install it by yourself. Also, if you are using the desktop in our Linux Lab, you do not need to install it.)

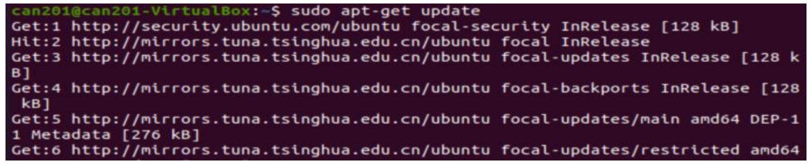
```
sudo apt-get update
```

```
sudo apt-get install python3 python3-pip xterm iperf hping3 net-tools wireshark apache2-utils curl
```

```
sudo pip3 install ryu
```



```
can201@can201-VirtualBox:~$ sudo pip3 install ryu
[sudo] password for can201:
Requirement already satisfied: ryu in /usr/local/lib/python3.8/dist-packages (4
.34)
Requirement already satisfied: six>=1.4.0 in /usr/lib/python3/dist-packages (fr
om ryu) (1.14.0)
Requirement already satisfied: webob>=1.2 in /usr/local/lib/python3.8/dist-pack
ages (from ryu) (1.8.7)
Requirement already satisfied: netaddr in /usr/local/lib/python3.8/dist-package
s (from ryu) (0.8.0)
Requirement already satisfied: ovs>=2.6.0 in /usr/lib/python3/dist-packages (fr
om ryu) (2.13.8)
Requirement already satisfied: oslo.config>=2.5.0 in /usr/local/lib/python3.8/d
ist-packages (from ryu) (9.0.0)
Requirement already satisfied: eventlet!=0.18.3,!=0.20.1,!=0.21.0,!=0.23.0,>=0.
18.2 in /usr/local/lib/python3.8/dist-packages (from ryu) (0.30.2)
Requirement already satisfied: msgpack>=0.3.0 in /usr/local/lib/python3.8/dist-
packages (from ryu) (1.0.4)
Requirement already satisfied: tinyrpc in /usr/local/lib/python3.8/dist-package
s (from ryu) (1.1.5)
```



```
can201@can201-VirtualBox:~$ sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Hit:2 http://mirrors.tuna.tsinghua.edu.cn/ubuntu focal InRelease
Get:3 http://mirrors.tuna.tsinghua.edu.cn/ubuntu focal-updates InRelease [128 k
B]
Get:4 http://mirrors.tuna.tsinghua.edu.cn/ubuntu focal-backports InRelease [128
 kB]
Get:5 http://mirrors.tuna.tsinghua.edu.cn/ubuntu focal-updates/main amd64 DEP-1
1 Metadata [276 kB]
Get:6 http://mirrors.tuna.tsinghua.edu.cn/ubuntu focal-updates/restricted amd64
```

How to install Ryu

- In addition, if you prefer to install Ryu from the source code:

```
#git clone https://github.com/faucetsdn/ryu.git
#cd ryu
#pip3 install .  (do not omit the ‘dot’ at the end)
```

More info refer to:
https://ryu.readthedocs.io/en/latest/getting_started.html

Test the installation of Ryu

- Test Python3

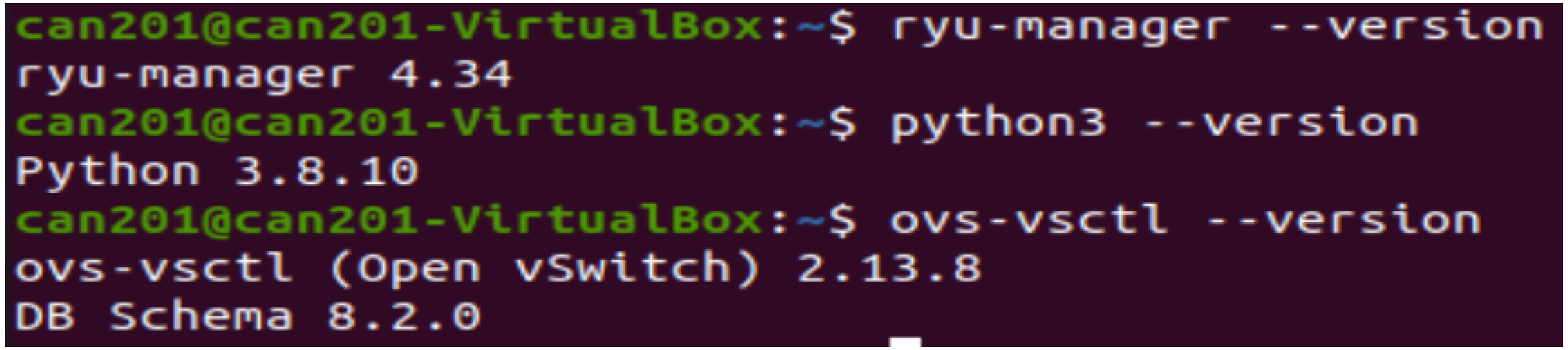
  python3 --version

- Test Open vSwitch

  ovs-vsctl --version

- Test Ryu

  ryu-manager --version

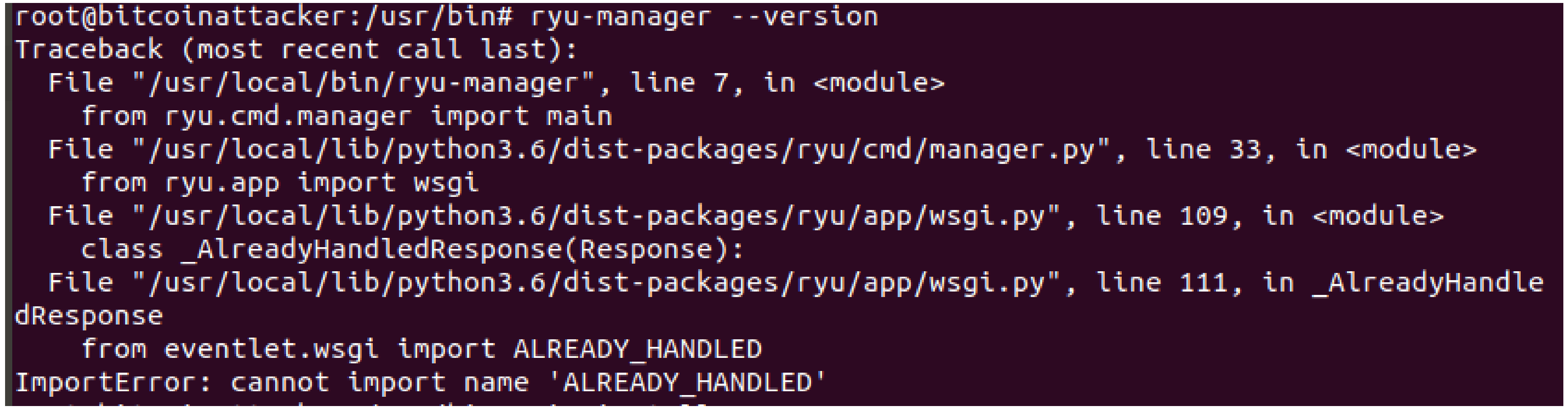


```
can201@can201-VirtualBox:~$ ryu-manager --version
ryu-manager 4.34
can201@can201-VirtualBox:~$ python3 --version
Python 3.8.10
can201@can201-VirtualBox:~$ ovs-vsctl --version
ovs-vsctl (Open vSwitch) 2.13.8
DB Schema 8.2.0
```

Test the installation of Ryu

- Test Ryu (if error occurred)

    ryu-manager --version



```
root@bitcoinattacker:/usr/bin# ryu-manager --version
Traceback (most recent call last):
  File "/usr/local/bin/ryu-manager", line 7, in <module>
    from ryu.cmd.manager import main
  File "/usr/local/lib/python3.6/dist-packages/ryu/cmd/manager.py", line 33, in <module>
    from ryu.app import wsgi
  File "/usr/local/lib/python3.6/dist-packages/ryu/app/wsgi.py", line 109, in <module>
    class _AlreadyHandledResponse(Response):
  File "/usr/local/lib/python3.6/dist-packages/ryu/app/wsgi.py", line 111, in _AlreadyHandle
dResponse
    from eventlet.wsgi import ALREADY_HANDLED
ImportError: cannot import name 'ALREADY_HANDLED'
```

Solution:

    pip3 install eventlet==0.30.2

How to run a Ryu application

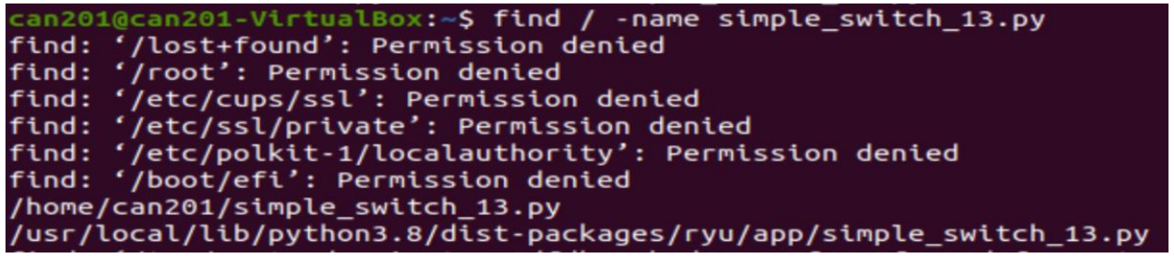
- If you create your own app file, like yourRyuApp.py, you can run it by:

  #ryu-manager yourRyuApp.py

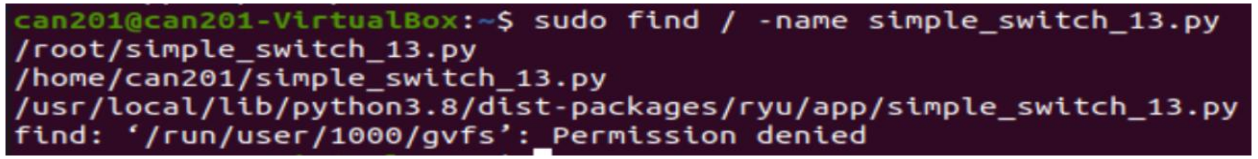
- You could also run the Ryu building app templates, like the simply_switch_13:

  #ryu-manager ryu.app.simple_switch_13

This one is actually a python file you can find it using the following way



```
can201@can201-VirtualBox:~$ find / -name simple_switch_13.py
find: '/lost+found': Permission denied
find: '/root': Permission denied
find: '/etc/cups/ssl': Permission denied
find: '/etc/ssl/private': Permission denied
find: '/etc/polkit-1/localauthority': Permission denied
find: '/boot/efi': Permission denied
/home/can201/simple_switch_13.py
/usr/local/lib/python3.8/dist-packages/ryu/app/simple_switch_13.py
```



```
can201@can201-VirtualBox:~$ sudo find / -name simple_switch_13.py
/root/simple_switch_13.py
/home/can201/simple_switch_13.py
/usr/local/lib/python3.8/dist-packages/ryu/app/simple_switch_13.py
find: '/run/user/1000/gvfs': Permission denied
```

- You can also run the simple_switch_13 python file by:

  #ryu-manager simple_switch_13.py

How to write ryu application, pls refer to
https://ryu.readthedocs.io/en/latest/writing_ryu_app.html

Outline

- Recap the SDN architecture
- SDN Controller - Ryu
- Integrating Ryu with Mininet
- Hands-on Practice

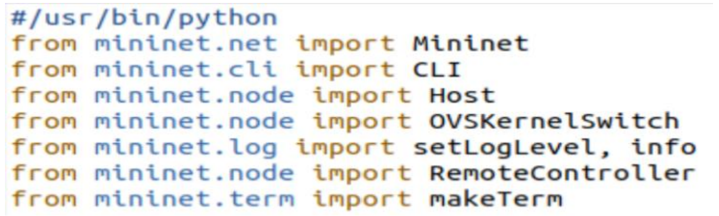
Integrating Ryu with Mininet

Create a gedit/vim (name it lab9) file and type the following lines of code:

1. Integrate a (remote) controller into a network emulated by Mininet:

   import the remote controller module

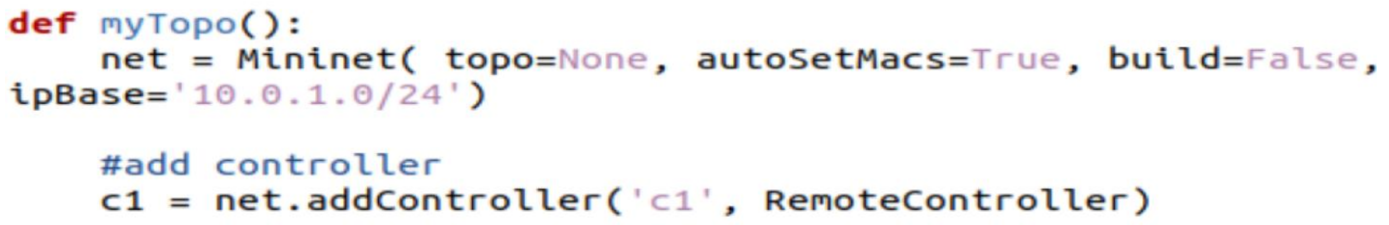
   from mininet.node import RemoteController



```
#/usr/bin/python
from mininet.net import Mininet
from mininet.cli import CLI
from mininet.node import Host
from mininet.node import OVSKernelSwitch
from mininet.log import setLogLevel, info
from mininet.node import RemoteController
from mininet.term import makeTerm
```

2. Add the controller to the net

   c1 = net.addController('c1', RemoteController)



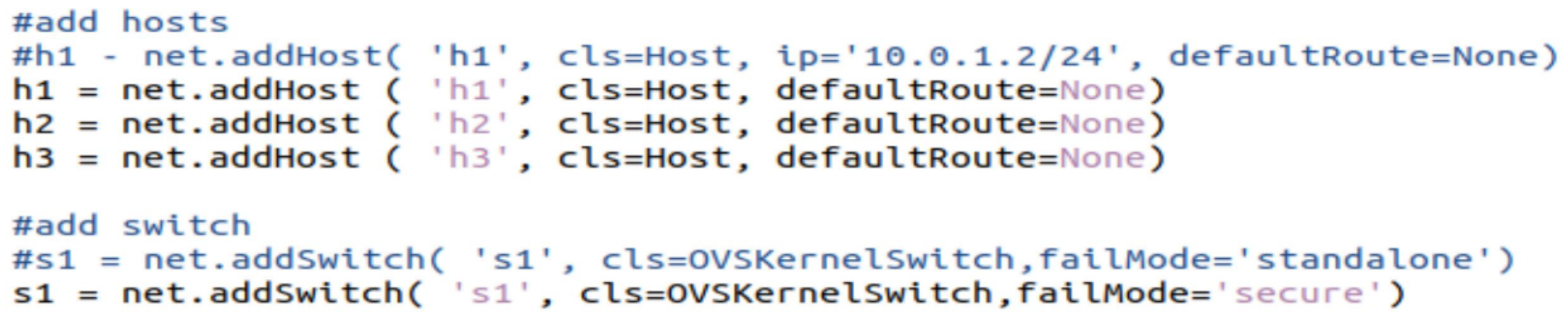
```
def myTopo():
    net = Mininet( topo=None, autoSetMacs=True, build=False,
ipBase='10.0.1.0/24')

    #add controller
    c1 = net.addController('c1', RemoteController)
```

Integrating Ryu with Mininet

3. Add hosts and a switch. Change the legacy switch to SDN packet switch:

    In mininet, we just set ‘secure’ (rather than ‘standalone’) for the failMode

        s1 = net.addSwitch('s1', cls=OVSKernelSwitch, failMode='secure')



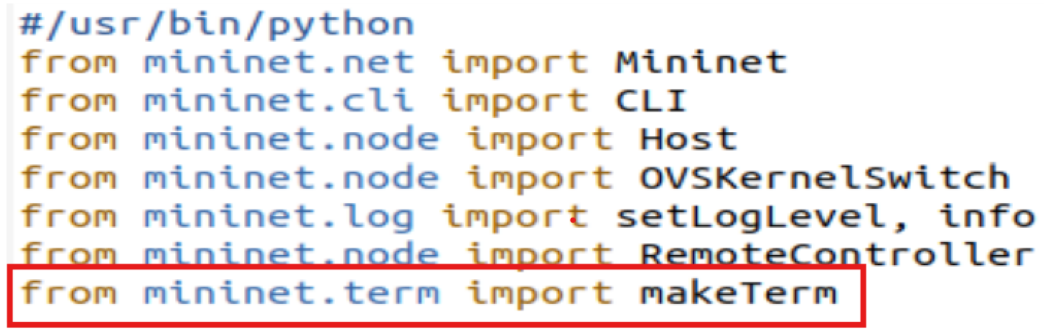
```
#add hosts
#h1 - net.addHost( 'h1', cls=Host, ip='10.0.1.2/24', defaultRoute=None)
h1 = net.addHost ( 'h1', cls=Host, defaultRoute=None)
h2 = net.addHost ( 'h2', cls=Host, defaultRoute=None)
h3 = net.addHost ( 'h3', cls=Host, defaultRoute=None)

#add switch
#s1 = net.addSwitch( 's1', cls=OVSKernelSwitch,failMode='standalone')
s1 = net.addSwitch( 's1', cls=OVSKernelSwitch,failMode='secure')
```

Integrating Ryu with Mininet

- For the sake of interaction convenience, we make the nodes’ xterm pop up automatically

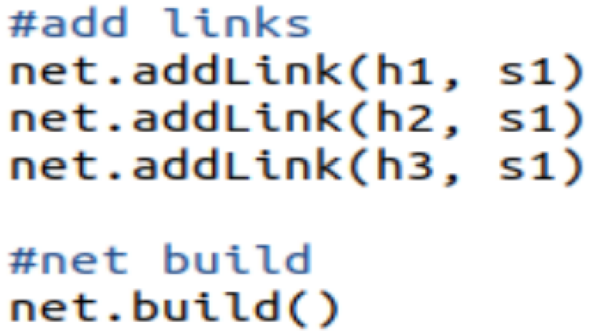
  import makeTerm module

  from mininet.term import makeTerm



```
#/usr/bin/python
from mininet.net import Mininet
from mininet.cli import CLI
from mininet.node import Host
from mininet.node import OVSKernelSwitch
from mininet.log import setLogLevel, info
from mininet.node import RemoteController
from mininet.term import makeTerm
```

4. Add links and build the network



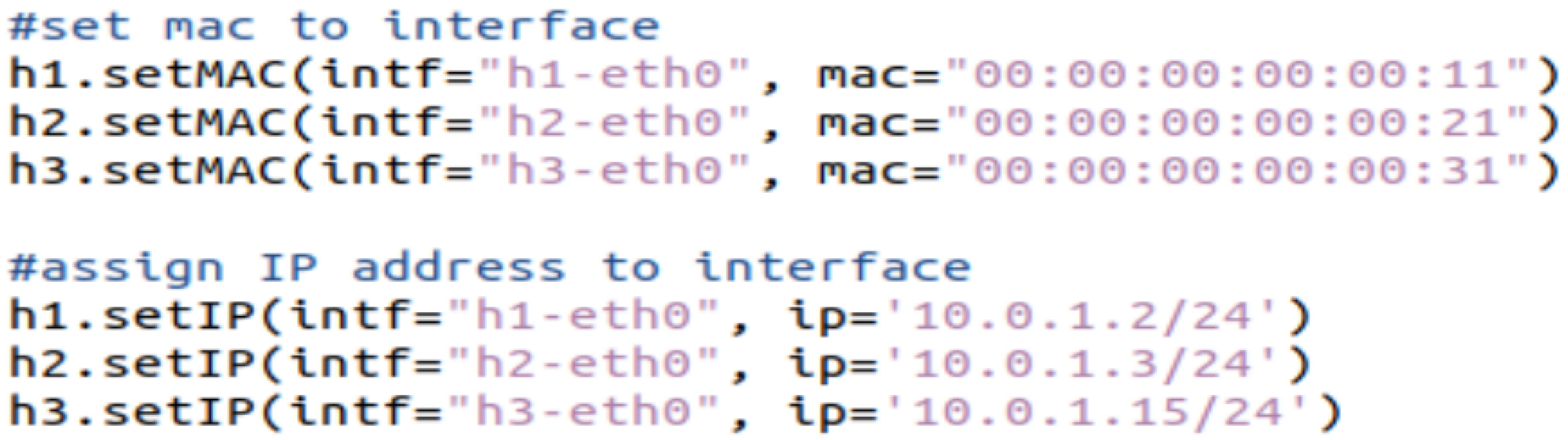
```
#add links
net.addLink(h1, s1)
net.addLink(h2, s1)
net.addLink(h3, s1)

#net build
net.build()
```

Integrating Ryu with Mininet

5. Set MAC to the interfaces in Mininet, using setMAC(), e.g.,:

    h1.setMAC(intf="h1-eth0", mac="00:00:00:00:00:11")

6. Assign IP addresses to the interfaces

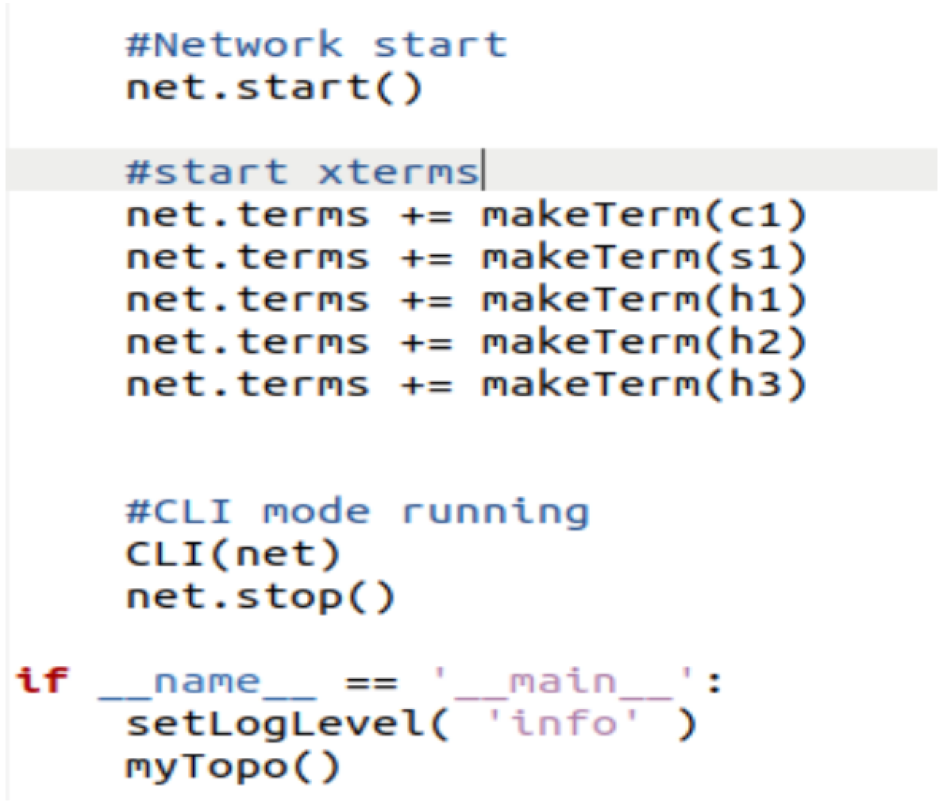


```
#set mac to interface
h1.setMAC(intf="h1-eth0", mac="00:00:00:00:00:11")
h2.setMAC(intf="h2-eth0", mac="00:00:00:00:00:21")
h3.setMAC(intf="h3-eth0", mac="00:00:00:00:00:31")

#assign IP address to interface
h1.setIP(intf="h1-eth0", ip='10.0.1.2/24')
h2.setIP(intf="h2-eth0", ip='10.0.1.3/24')
h3.setIP(intf="h3-eth0", ip='10.0.1.15/24')
```

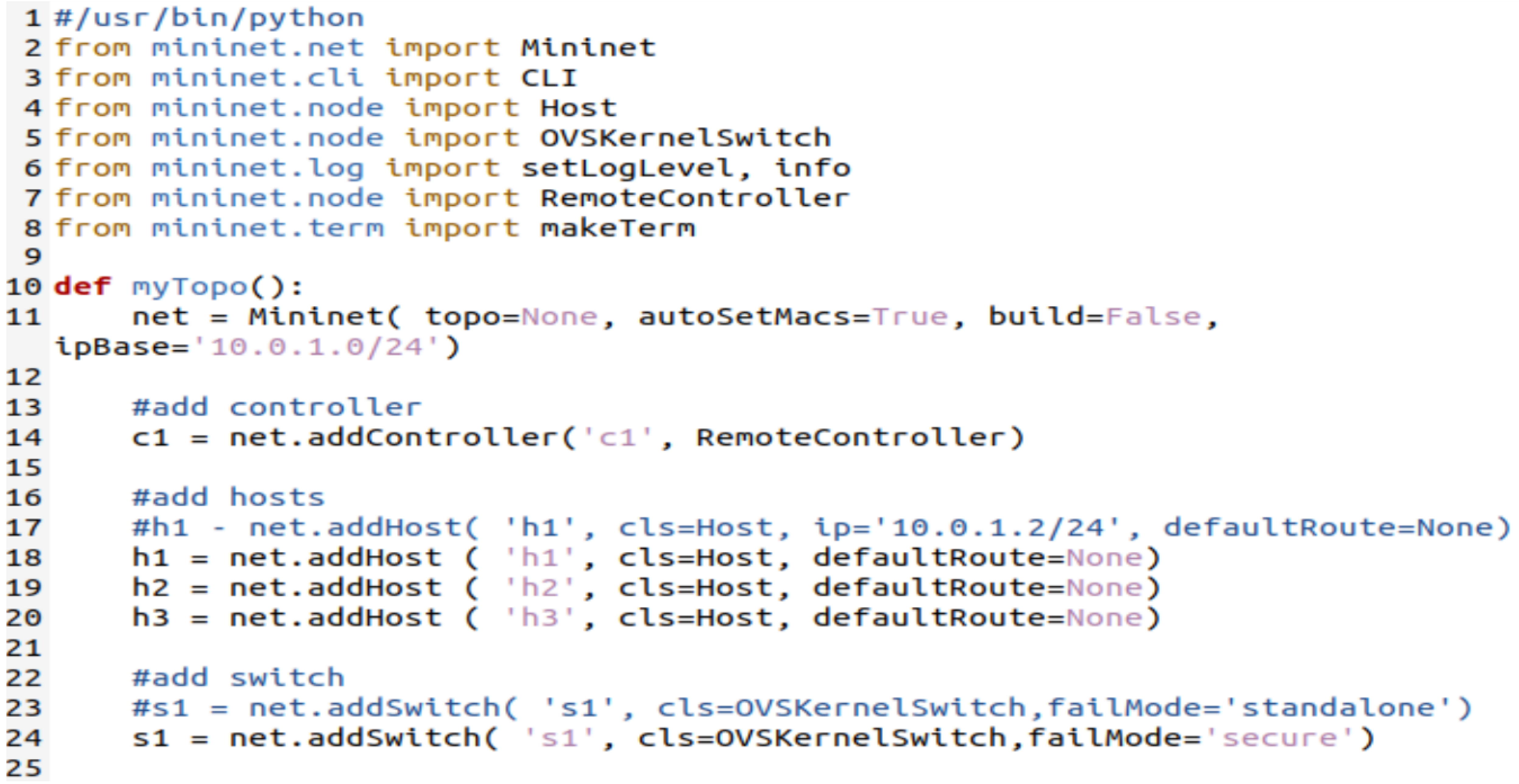
Integrating Ryu with Mininet

7. Add terms for each entity by using:

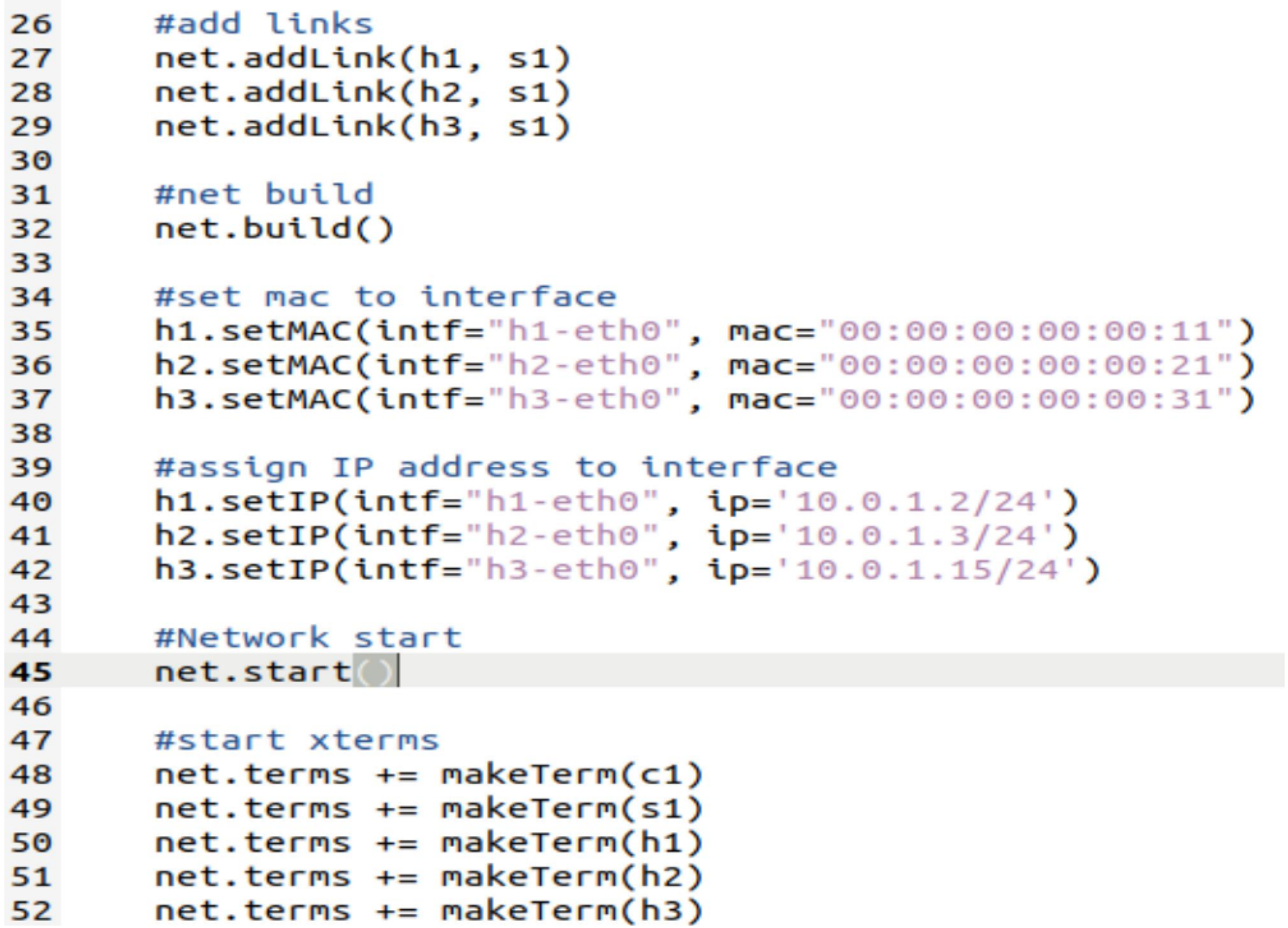
net.terms += makeTerm(h1)



```
    #Network start
    net.start()

    #start xterms
    net.terms += makeTerm(c1)
    net.terms += makeTerm(s1)
    net.terms += makeTerm(h1)
    net.terms += makeTerm(h2)
    net.terms += makeTerm(h3)


    #CLI mode running
    CLI(net)
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    myTopo()
```



```python
#/usr/bin/python
from mininet.net import Mininet
from mininet.cli import CLI
from mininet.node import Host
from mininet.node import OVSKernelSwitch
from mininet.log import setLogLevel, info
from mininet.node import RemoteController
from mininet.term import makeTerm

def myTopo():
    net = Mininet( topo=None, autoSetMacs=True, build=False, ipBase='10.0.1.0/24')

    #add controller
    c1 = net.addController('c1', RemoteController)

    #add hosts
    #h1 - net.addHost( 'h1', cls=Host, ip='10.0.1.2/24', defaultRoute=None)
    h1 = net.addHost ( 'h1', cls=Host, defaultRoute=None)
    h2 = net.addHost ( 'h2', cls=Host, defaultRoute=None)
    h3 = net.addHost ( 'h3', cls=Host, defaultRoute=None)

    #add switch
    #s1 = net.addSwitch( 's1', cls=OVSKernelSwitch,failMode='standalone')
    s1 = net.addSwitch( 's1', cls=OVSKernelSwitch,failMode='secure')

```



```
    #add links
    net.addLink(h1, s1)
    net.addLink(h2, s1)
    net.addLink(h3, s1)

    #net build
    net.build()

    #set mac to interface
    h1.setMAC(intf="h1-eth0", mac="00:00:00:00:00:11")
    h2.setMAC(intf="h2-eth0", mac="00:00:00:00:00:21")
    h3.setMAC(intf="h3-eth0", mac="00:00:00:00:00:31")

    #assign IP address to interface
    h1.setIP(intf="h1-eth0", ip='10.0.1.2/24')
    h2.setIP(intf="h2-eth0", ip='10.0.1.3/24')
    h3.setIP(intf="h3-eth0", ip='10.0.1.15/24')

    #Network start
    net.start()

    #start xterms
    net.terms += makeTerm(c1)
    net.terms += makeTerm(s1)
    net.terms += makeTerm(h1)
    net.terms += makeTerm(h2)
    net.terms += makeTerm(h3)
```

```python


    #CLI mode running
    CLI(net)
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    myTopo()

```

Integrating Ryu with Mininet

How to run the new mininet network topology

1. Run the mininet topology file (let’s call it lab9.py):

```
sudo python3 lab9.py
```

2. Then, in the terminal of c1, run the controller app “simple_switch_13.py” (meaning you should have this py file in advance under the same folder of lab9.py):

```
sudo ryu-manager simple_switch_13.py
```

If the simple_switch_13.py file is not in the same folder, copy it to the folder using:

`sudo find / -name simple_switch_13.py` and then copy the line that begins with/usr/local/lib……. Use the command `cp /user/local/lib….` to copy the file.

After that, check whether it is successfully copied by navigating the folder using `ls`

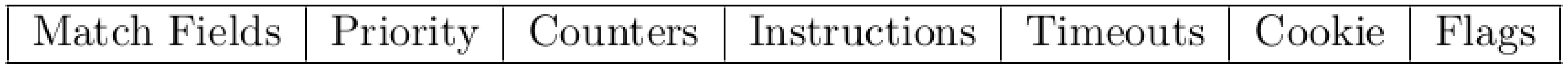
3. Further, in the terminal of s1, you can check the flow table:

```
sudo ovs-ofctl -O OpenFlow13 dump-flows s1
```

Integrating Ryu with Mininet

OVS flow table

```
$ sudo ovs-ofctl -O OpenFlow13 dump-flows s1

cookie=0x0, duration=3.933s, table=0, n_packets=238752, n_bytes=11069190464,
priority=1,tcp,nw_src=192.168.1.2,nw_dst=192.168.1.1,tp_src=37304,tp_dst=5001 actions=output:"s1-eth1"

cookie=0x0, duration=3.906s, table=0, n_packets=192421, n_bytes=12699810,
priority=1,tcp,nw_src=192.168.1.1,nw_dst=192.168.1.2,tp_src=5001,tp_dst=37304 actions=output:"s1-eth2"

cookie=0x0, duration=31.495s, table=0, n_packets=43, n_bytes=4309, priority=0 actions=CONTROLLER:65535
```



Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Table 1: Main components of a flow entry in a flow table.

Integrating Ryu with Mininet

- **match fields:** to match against packets. These consist of the ingress port and packet headers, and optionally other pipeline fields such as metadata specified by a previous table.
- **priority:** matching precedence of the flow entry.
- **counters:** updated when packets are matched.
- **instructions:** to modify the action set or pipeline processing.
- **timeouts:** maximum amount of time or idle time before flow is expired by the switch.
- **cookie:** opaque data value chosen by the controller. May be used by the controller to filter flow entries affected by flow statistics, flow modification and flow deletion requests. Not used when processing packets.
- **flags:** flags alter the way flow entries are managed, for example the flag OFPFF_SEND_FLOW_REM triggers flow removed messages for that flow entry.
- **The flow entry that wildcards all match fields (all fields omitted) and has priority equal to 0 is called the table-miss flow entry. The table-miss flow entry must support at least sending packets to the controller using the CONTROLLER reserved port.**

Demo

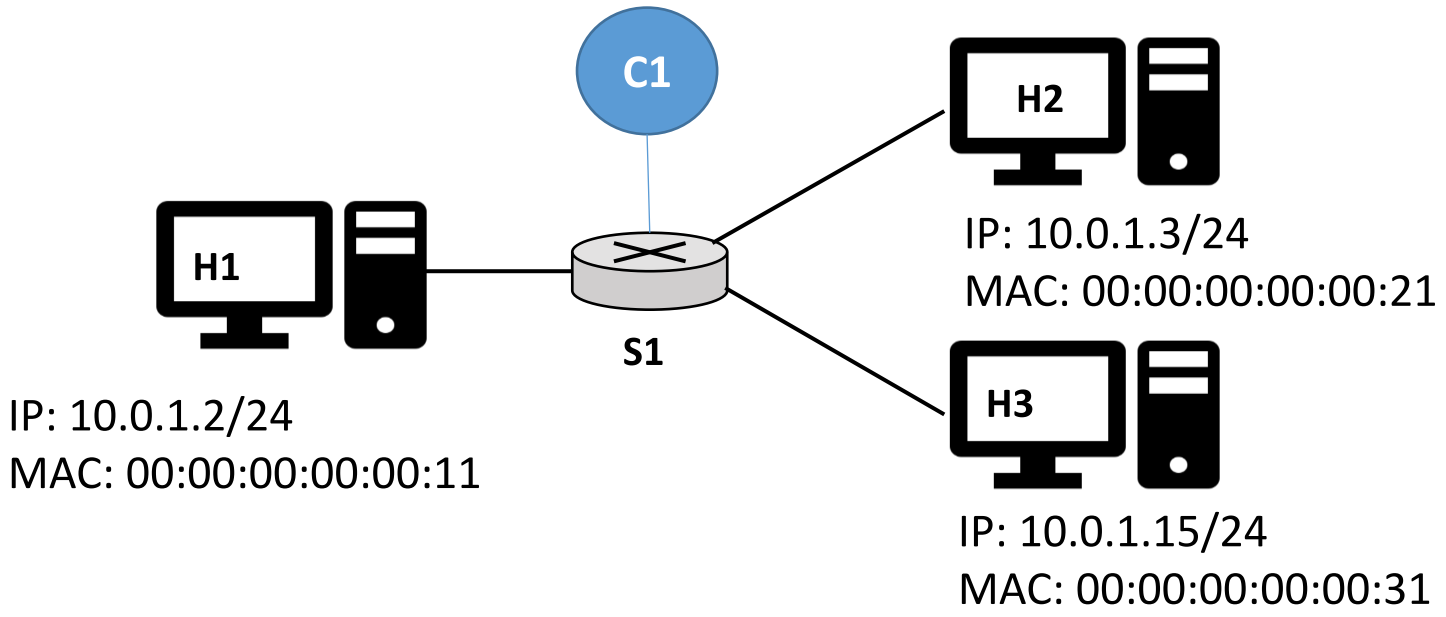
https://box.xjtlu.edu.cn/f/e048b64c19274f48a16b/

Outline

- Recap the SDN architecture
- SDN Controller - Ryu
- Integrating Ryu with Mininet
- Hands-on Practice

Hands-on Practice

1. From the following figure, develop the controller app Python program called lab9ex1.py



1. Create lab9ex1.py and simple_switch_13.py, and run them.
2. Ping h3 from h1, analyze the OpenFlow table entries, and record the following: i) number of bytes of the sending and receiving packets, ii) number of packets of the sending and receiving entries.

Show the results to the TA (Do not modify lab9.py).

Hands-on Practice

2. Using the existing lab 9 mininet topology file lab9.py, create a new topology file named lab9ex2.py. Your task is to extend the network by doing the following:

i) Add 6 more hosts, name them h4, h5, h6, h7, h8, h9. Assign each host a unique IP address in the 10.0.1.0/24 network. Assign each a distinct MAC address, following the pattern already used.

ii) Add two more switches, name them s2 and s3. Use SDN mode.

iii) Create a three-switch interconnected network. Then, attach hosts to the switches in any reasonable distribution. Do not modify the controller. All switches must be connected to the controller automatically.

iv) Generate Xterm windows for all hosts.

v) Run the Mininet topology file, controller application, and dump the OpenFlow table on s1. Ping h9 from h4 and h1 from h2. Analyze the OpenFlow table entries. What is the difference between the two ping results?

vi) Do s1 and s2 exchange OpenFlow messages with each other like h1 and h2 do? Use the flow table to justify your answer. **NB:** You can use sudo ovs-vsctl show to check all switch connections