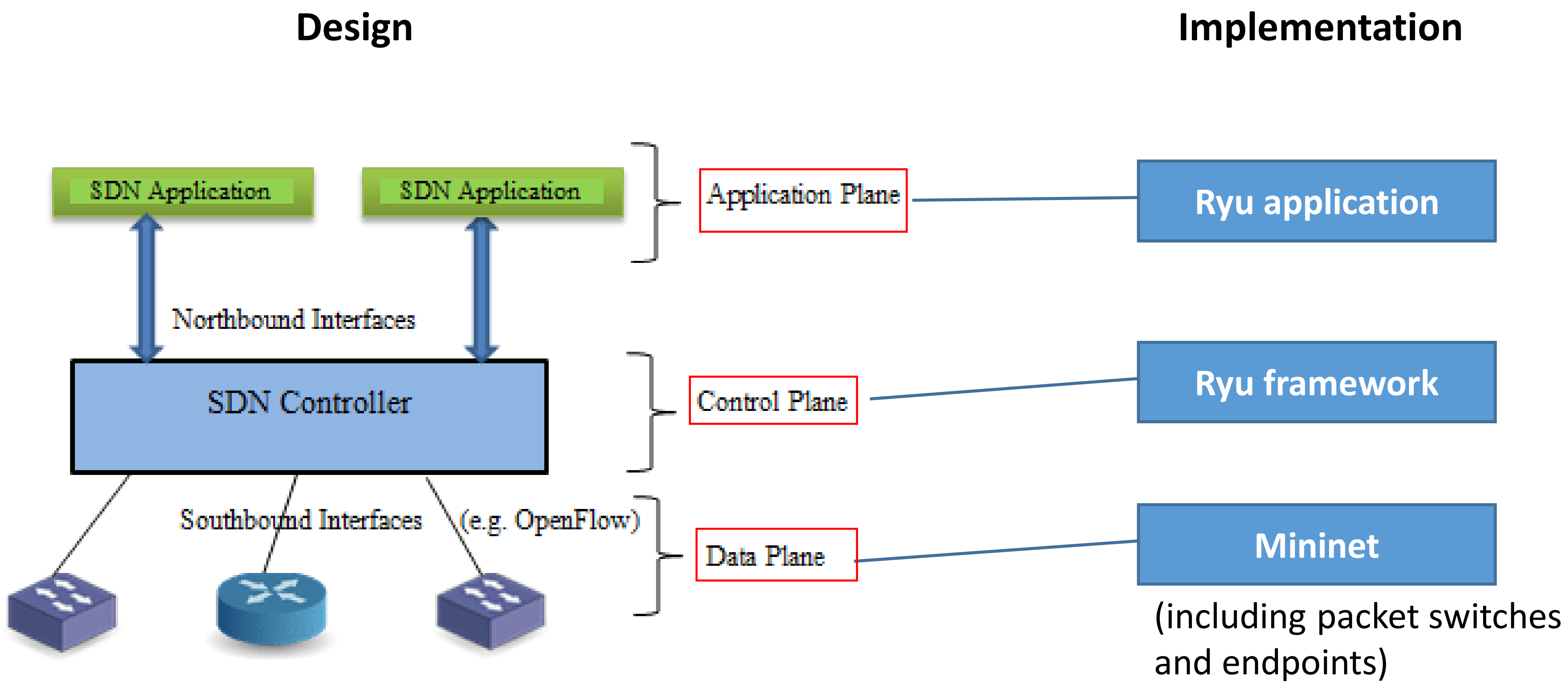
Lab 9 (Week 10) SDN Lab using Ryu and Mininet

CAN201

Dr. Wenjun Fan

Recap the SDN architecture

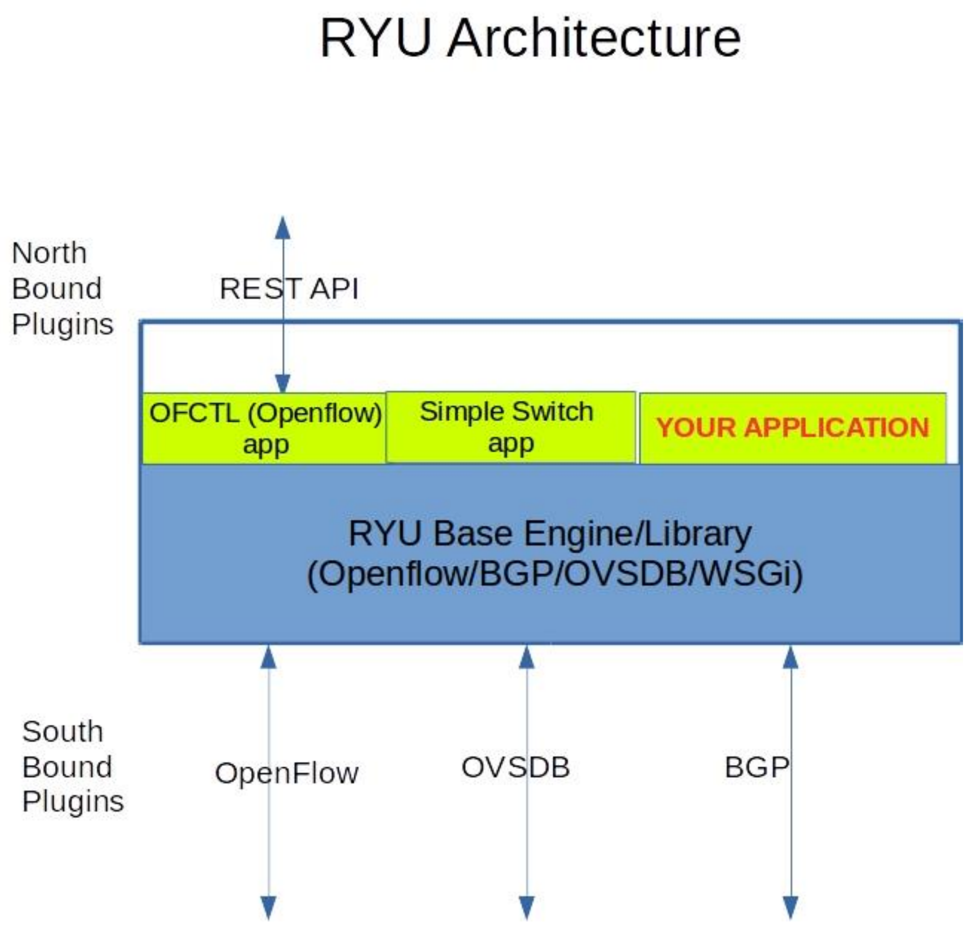


Outline

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

Ryu controller

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 installs Ryu, so you don’t need to install it by yourselves. Also, if you are using the desktop in our Linux Lab, you don’t need to install it neither.)

```
sudo apt-get update

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

sudo pip3 install ryu
```

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 .  (don’t 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

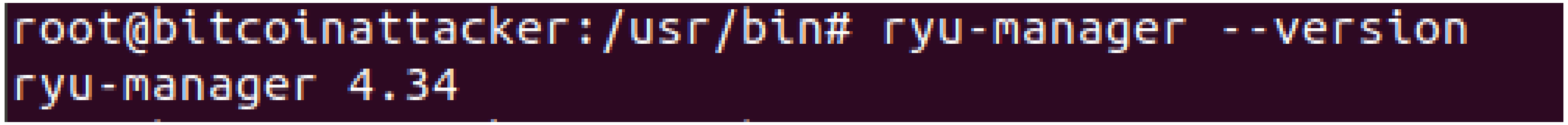
  python3 --version

- Test Open vSwitch

  ovs-vsctl --version

- Test Ryu

  ryu-manager --version

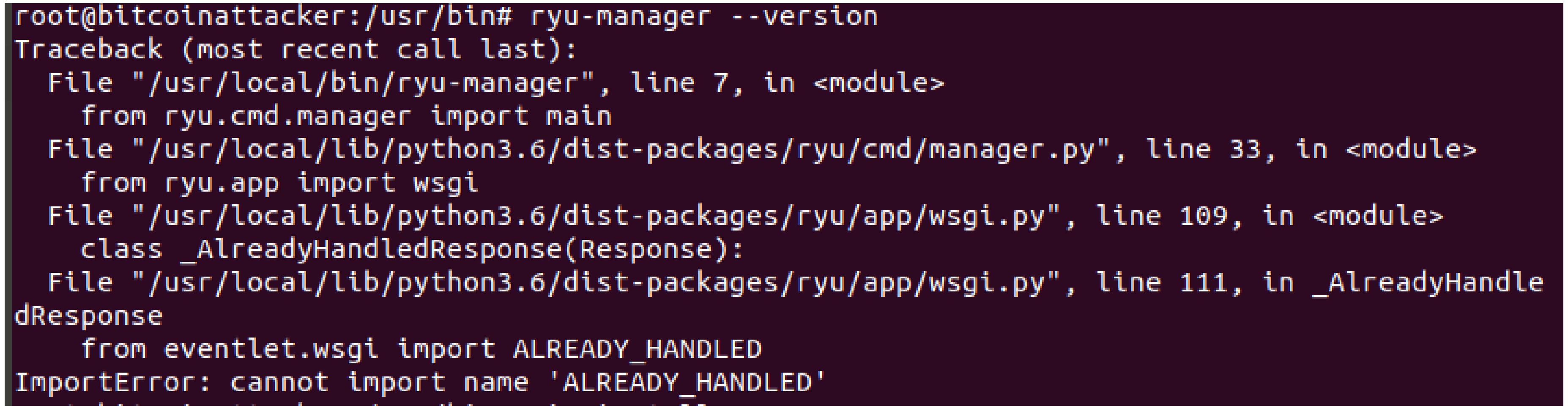


```
root@bitcoinattacker:/usr/bin# ryu-manager --version
ryu-manager 4.34
```

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

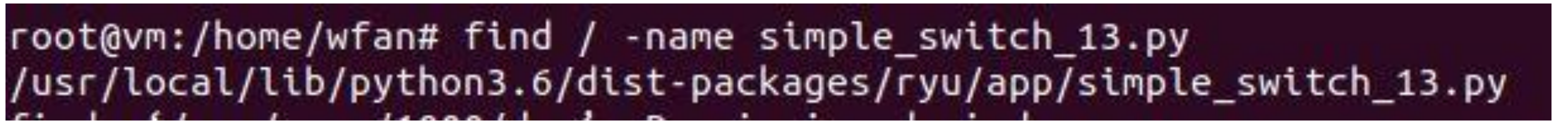
How to run a Ryu application

- 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 buildin 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



```
root@vm:/home/wfan# find / -name simple_switch_13.py
/usr/local/lib/python3.6/dist-packages/ryu/app/simple_switch_13.py
```

So, 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

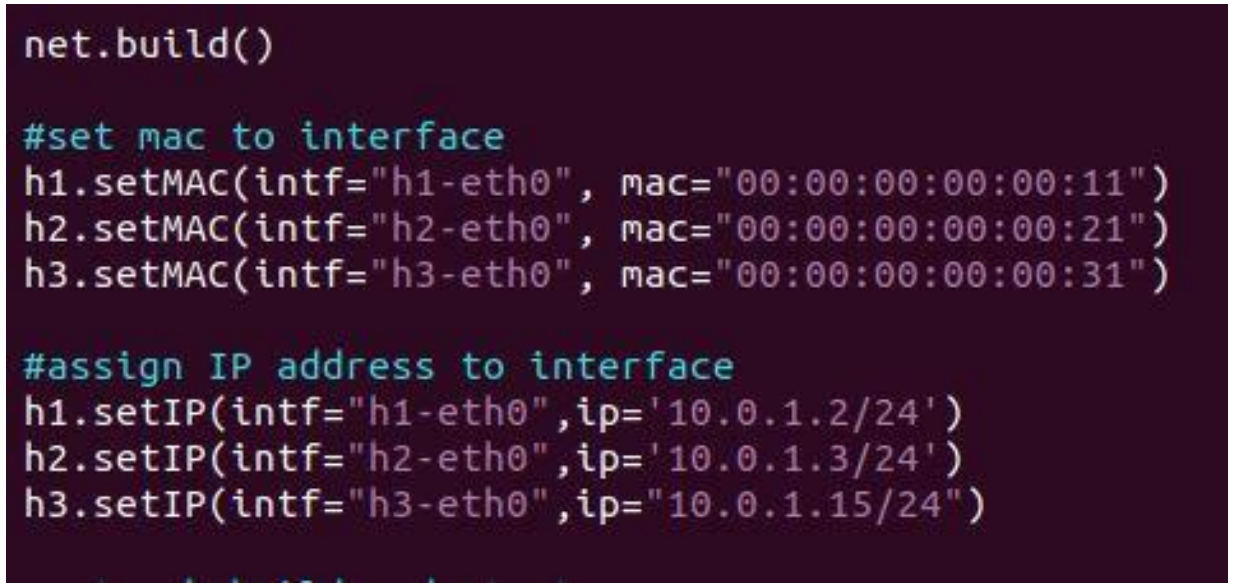
Outline

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

Integrate Ryu with Mininet

- Let’s first finish how to set MAC to the interface in Mininet, just use setMAC(), e.g.,:

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



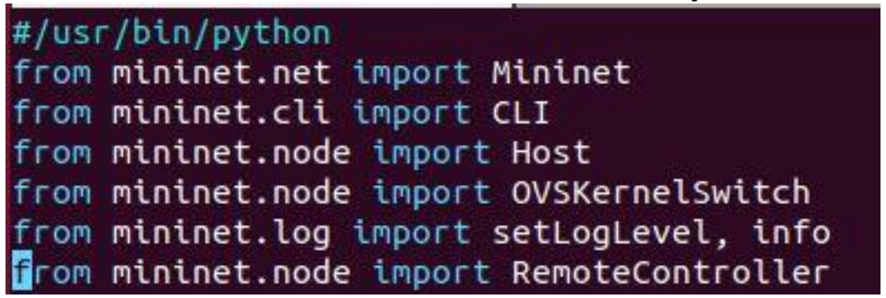
```
net.build()

#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")
```

Integrate Ryu with Mininet

- Integrate a (remote) controller to a network emulated by Mininet:

1. import the remote controller module

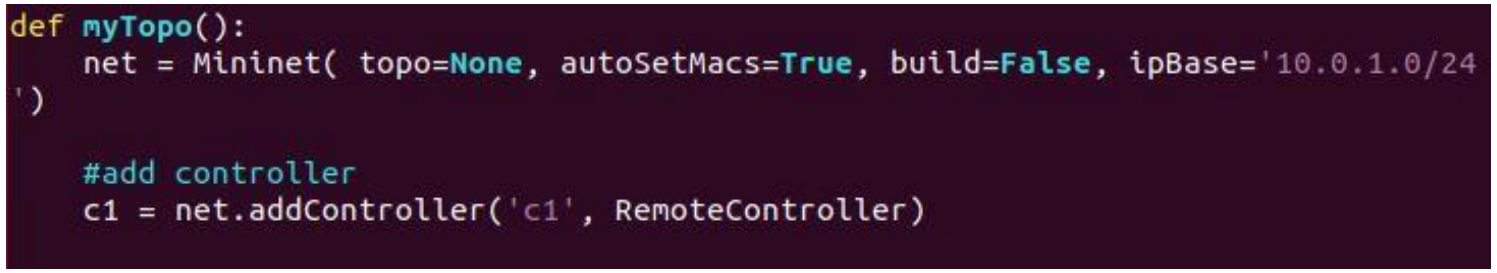
   from mininet.node import RemoteController



```
#/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
```

2. add the controller to the net

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



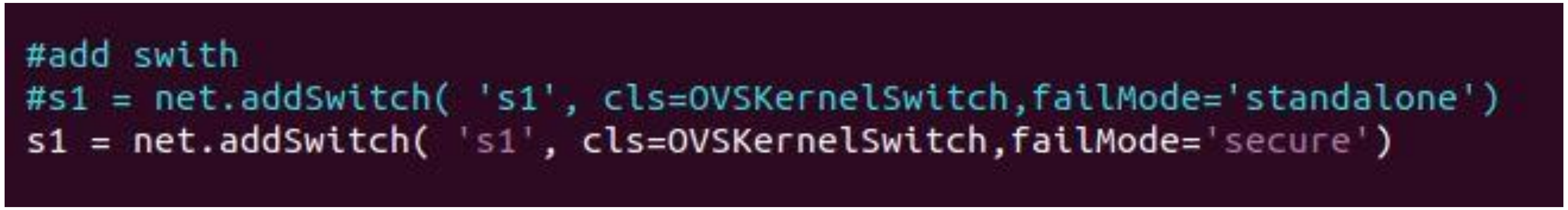
```
def myTopo():
    net = Mininet( topo=None, autoSetMacs=True, build=False, ipBase='10.0.1.0/24
')

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

Integrate Ryu with Mininet

- 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')



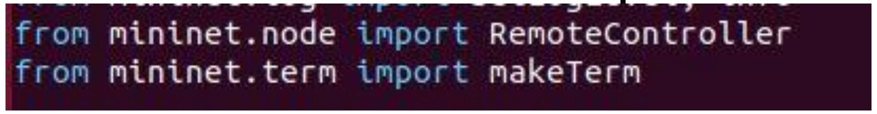
```
#add swith
#s1 = net.addSwitch( 's1', cls=OVSKernelSwitch,failMode='standalone')
s1 = net.addSwitch( 's1', cls=OVSKernelSwitch,failMode='secure')
```

Integrate Ryu with Mininet

- For the sake of interaction convenience, let’s make the node’s xterm showing up automatically

1. import makeTerm module

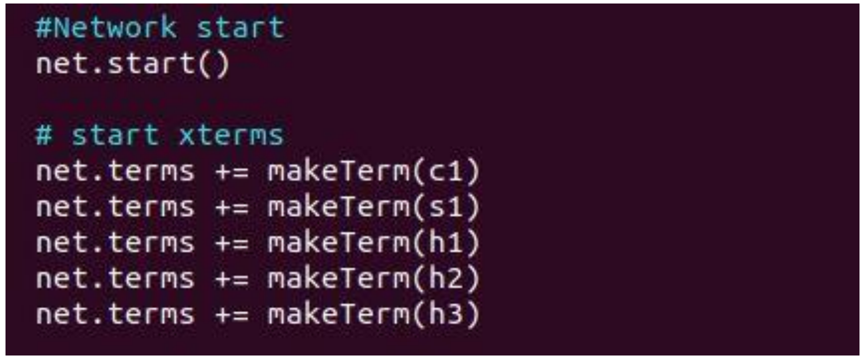
   from mininet.term import makeTerm



```
from mininet.node import RemoteController
from mininet.term import makeTerm
```

2. 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)
```

Integrate Ryu with Mininet

How to run the new mininet network topology

1. Run the mininet topology file (let's call it myTopo_lab10.py):

```
$sudo python3 myTopo_lab10.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 myTopo_lab10.py):

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

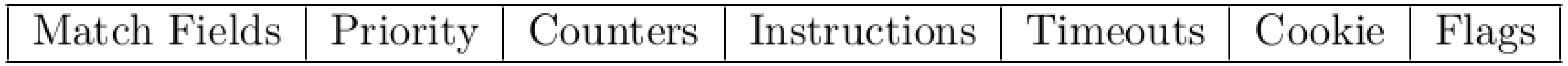
3. Further, in the terminal of s1, you can check the flow table:

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

Integrate 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

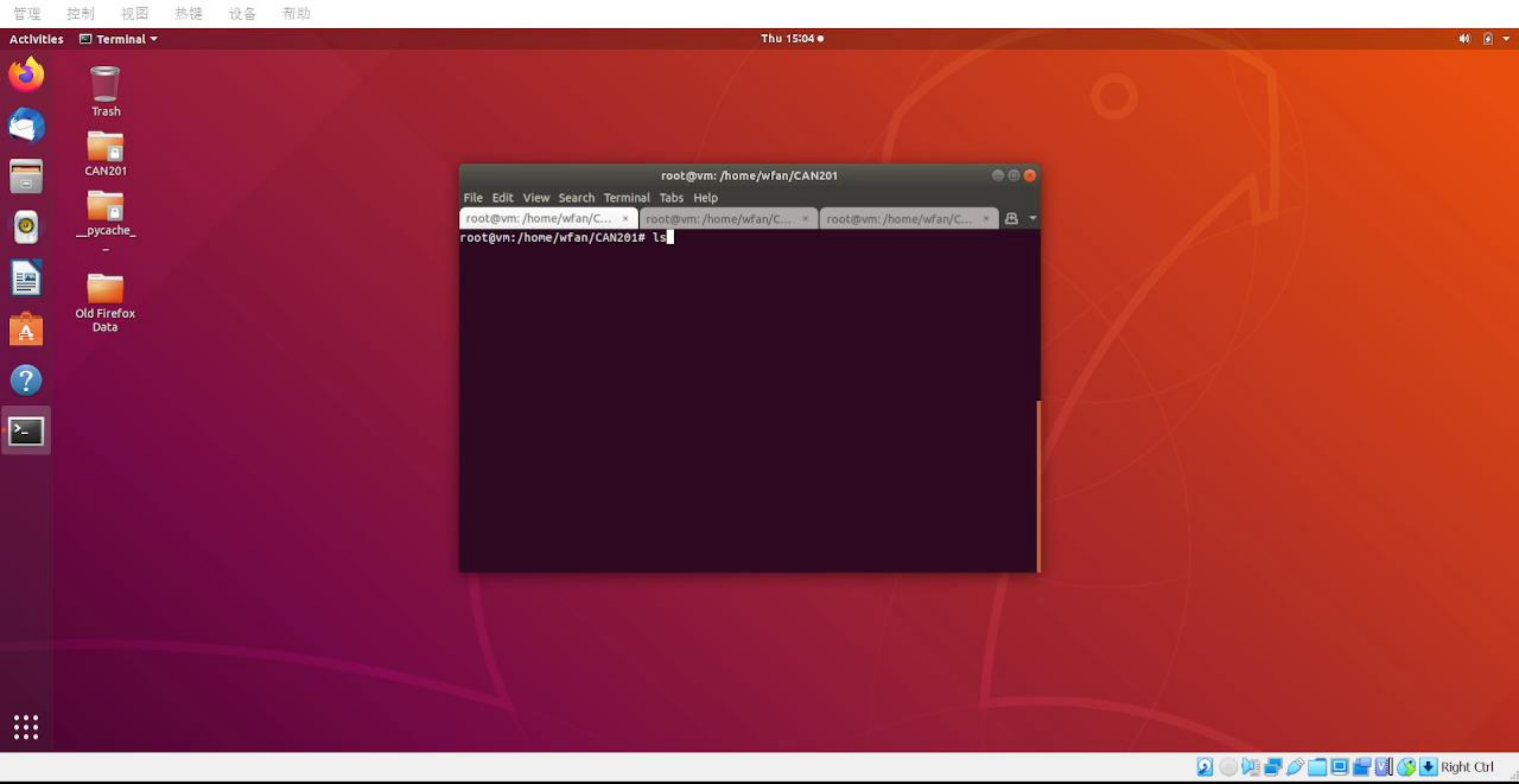
Table 1: Main components of a flow entry in a flow table.

Integrate 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

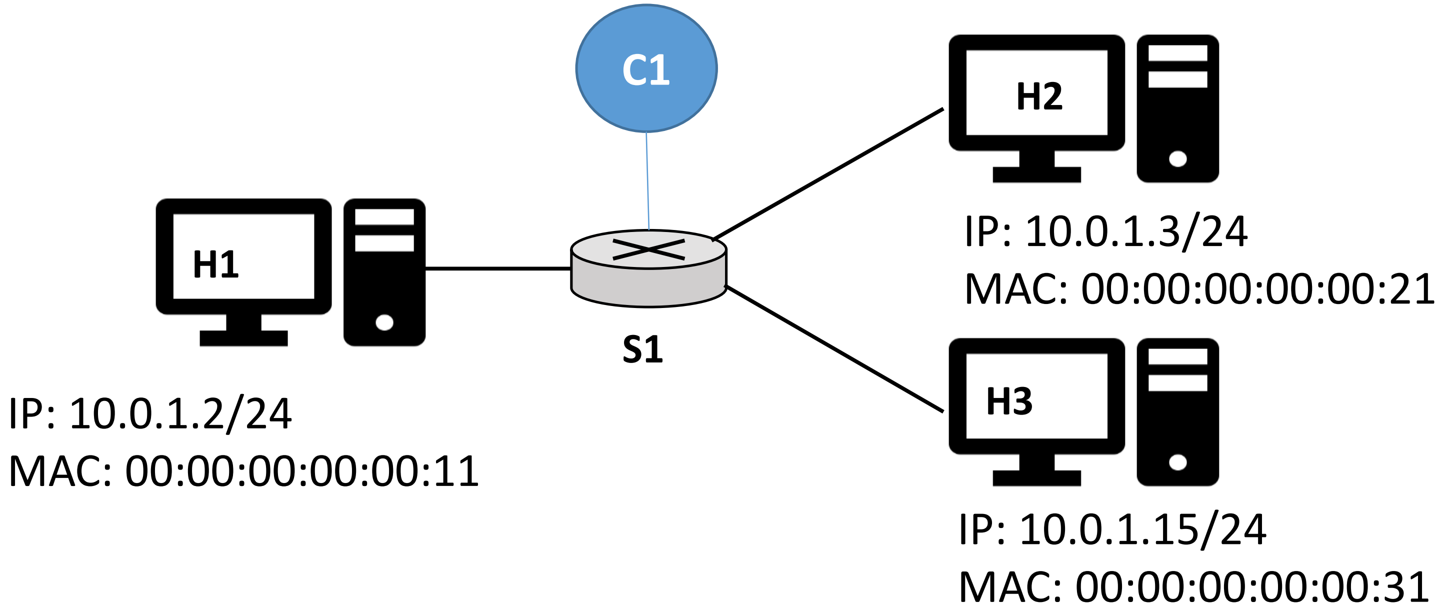
https://box.xjtlu.edu.cn/f/ecfe56940d474633970a/



```
root@vm:/home/wfan/CAN201# ls
```

Hands-on Practice

- In terms of the following figure, develop the controller app python program called myTopo_Lab10.py



- To create myTopo_Lab10.py and simple_switch_13.py, run them and show the results to TA.