Computer Networks
@CS.NCTU

Lab. 3: Route Configuration

Deadline : Dec 20, 2019. 23:59

Objectives

In this lab, we are going to write a Python program with Ryu SDN framework to build a simple software-defined network and compare the differences between two forwarding rules.

1. Learn how to build a simple software-defined networking with Ryu SDN framework
2. Learn how to add forwarding rules into each OpenFlow switch

TODO

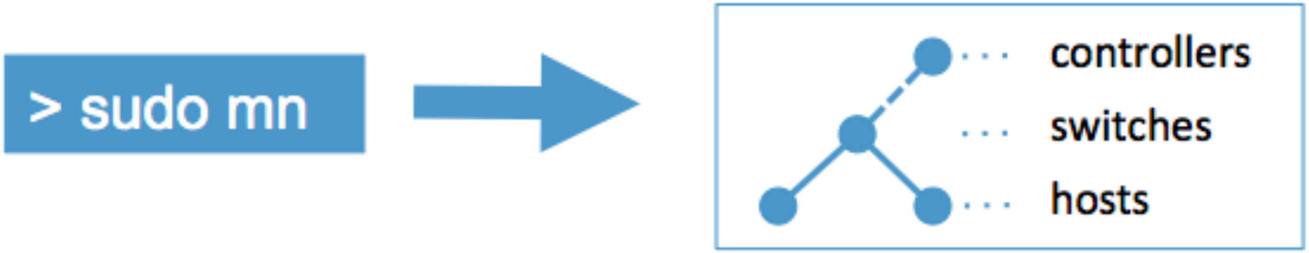
1. Modify topo.py according to the topology we provide
2. Create controller.py from the example code SimpleController.py, and modify controller.py to set the forwarding rule
3. Measure the networks by using iperf

Search **"[TODO]"** in this slide and codes to figure out where you should modify

Overview

Mininet

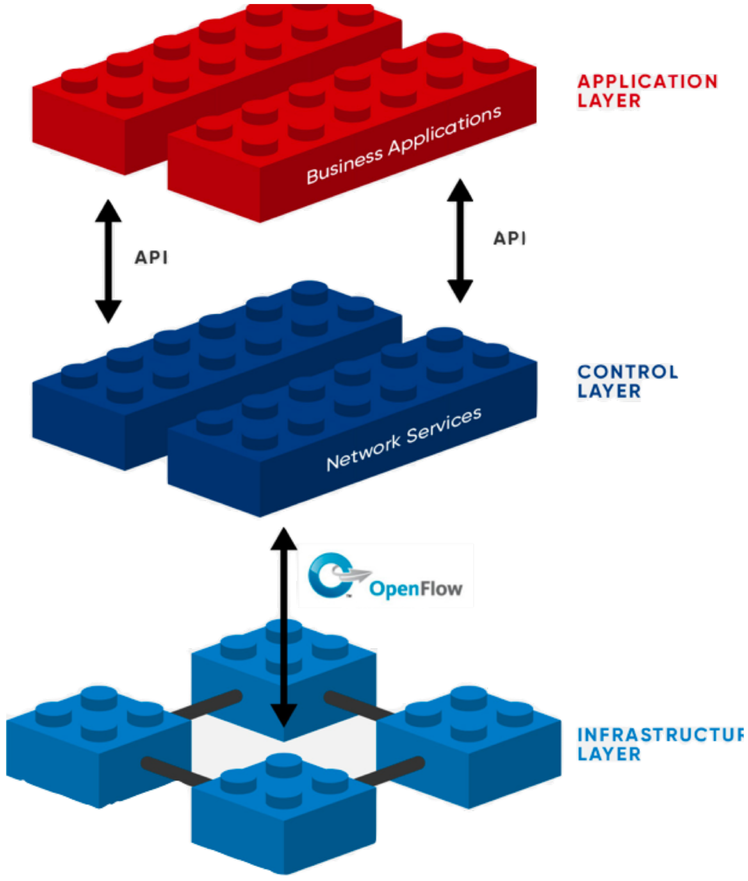
- Mininet is a network emulator
  - Overview of Mininet - http://mininet.org/overview/
  - We have provided you a container that has installed Mininet
- Create a realistic virtual network, running real kernel, switch and application code, on a single machine (VM, cloud or native)
- Run a collection of end-hosts, switches, routers, and links on a single Linux kernel.



```
> sudo mn
```

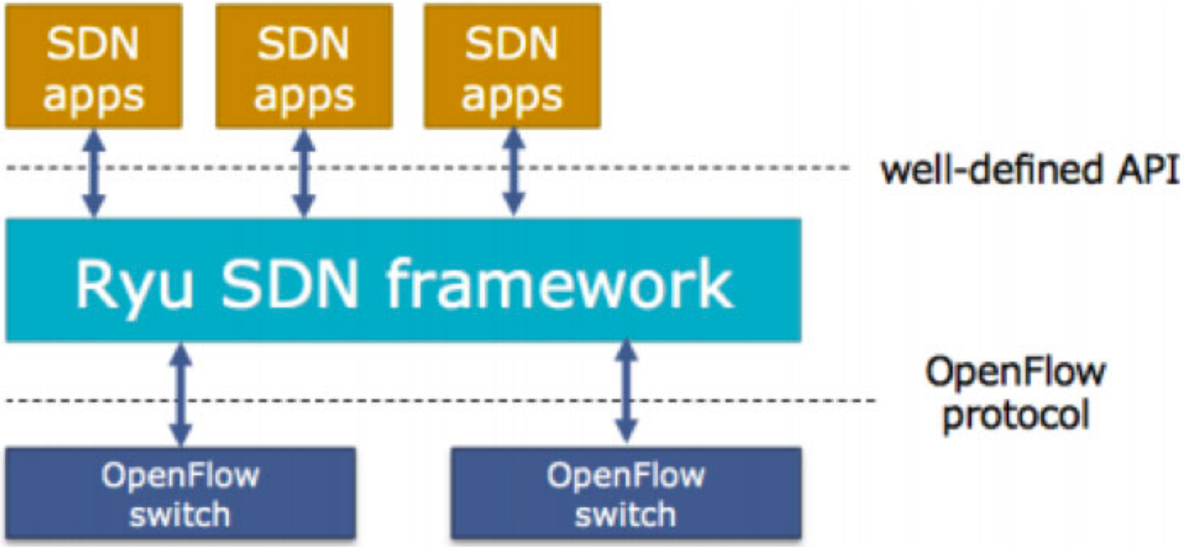
Software-Defined Networking (SDN)

- Software-defined = Programmable
  - Dynamic
  - Manageable
  - Cost-effective
  - Adaptable
- The OpenFlow protocol is a foundational element for building SDN

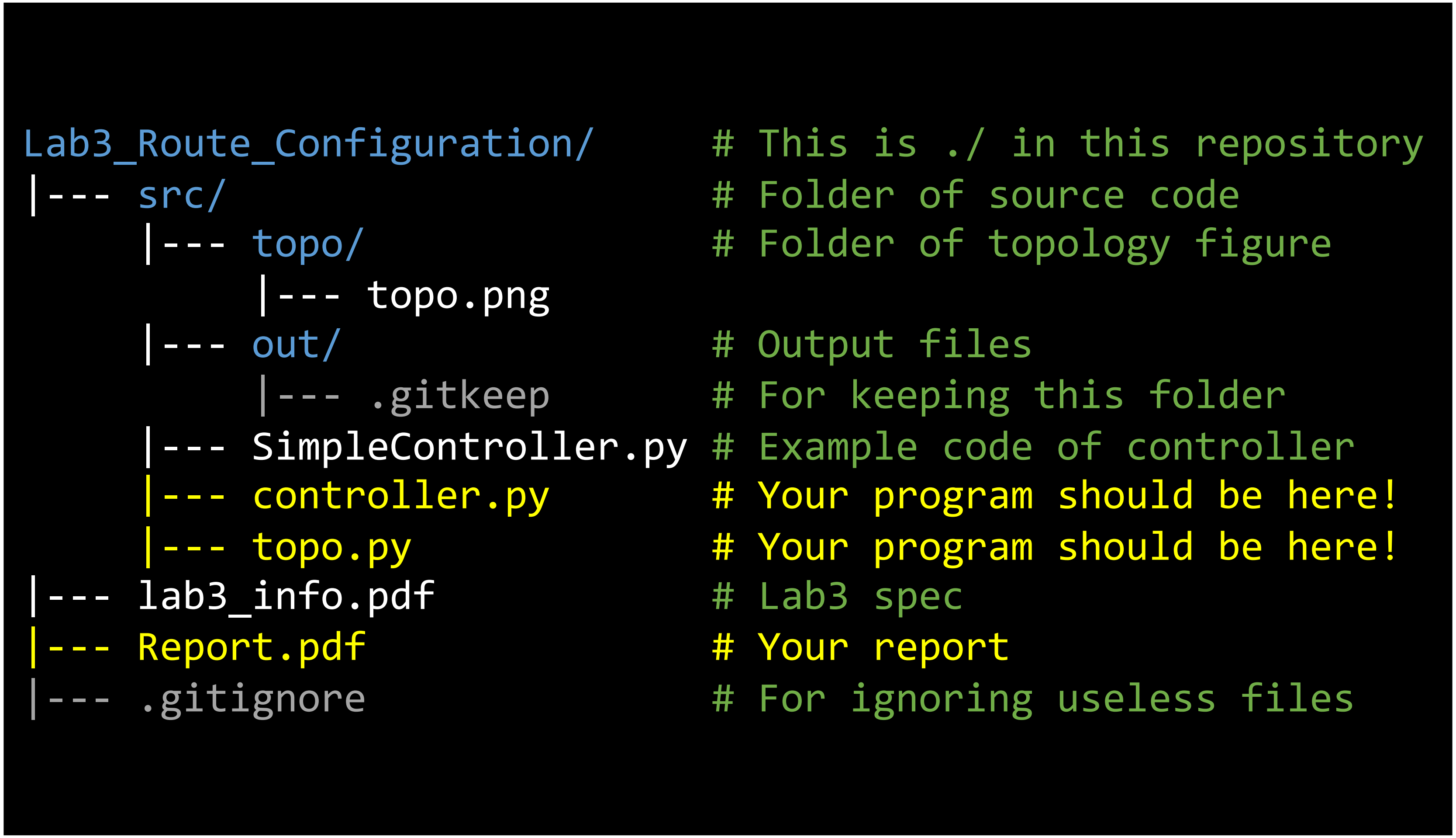


Ryu SDN

- Ryu is a component-based software defined networking framework
  - Support various protocols for managing network devices, such as OpenFlow, etc
  - We have provided you a container that has installed Ryu



File Structure



```
Lab3_Route_Configuration/     # This is ./ in this repository
|--- src/                     # Folder of source code
     |--- topo/               # Folder of topology figure
          |--- topo.png
     |--- out/                # Output files
          |--- .gitkeep       # For keeping this folder
     |--- SimpleController.py # Example code of controller
     |--- controller.py       # Your program should be here!
     |--- topo.py             # Your program should be here!
|--- lab3_info.pdf            # Lab3 spec
|--- Report.pdf               # Your report
|--- .gitignore               # For ignoring useless files
```

Tasks

Tasks

1. Environment Setup
2. Example of Ryu SDN
3. Mininet Topology (modify topo.py)
4. Ryu Controller (modify controller.py)
5. Measurement
6. Report

Task 1. Environment Setup

- **Step 1. Join this lab on GitHub Classroom**
  - Click the following link to join this lab
    - https://classroom.github.com/a/qiRwd3rR
  - Go to our GitHub group to see your repository
    - https://github.com/nctucn

Task 1. Environment Setup (cont.)

- Step 2. Login to your container using SSH
  - **For Windows**
    - Open PieTTY and connect to your container
      - IP address: `140.113.195.69`
      - Port: port list (Different from last 2 labs !!!)
    - Login as root

```
Login: root
Password: cn2019
```

  - **For Windows, MacOS and Ubuntu**
    - Use terminal to connect to the Docker

```
$ ssh root@140.113.195.69 -p xxxxx
Password: cn2019
```

Task 1. Environment Setup (cont.)

Step 2. Change the password (in container)

- Change the password of container instead of using the default one

```
$ passwd
Enter new UNIX password:
Retype new UNIX password:
```

- You will see the following message if succeeded

```
passwd: password updated successfully
```

- Please remember your own password

Task 1. Environment Setup (cont.)

- **Step 3. Get GitHub repository (in container)**
  - Download required files from GitHub

```
$ git clone
https://github.com/chenyang14/Lab3_Route_Configuration.git
```

  - Get and set repository for global options

```
$ cd Lab3_Route_Configuration/
$ git config --global user.name "<NAME>"
$ git config --global user.email "<EMAIL>"
```

  - Set a new remote URL to your repository

```
$ git remote set-url origin
https://github.com/nctucn/lab3-<GITHUB_ID>.git
```

  - Push your repository to GitHub

```
$ git push origin master
```

Task 1. Environment Setup (cont.)

- **Step 4. Run Mininet for testing**
  - After logging to your container, you may meet the following error when running Mininet

```
# Run Mininet for testing
$ [sudo] mn
......
*** Error connecting to ovs-db with ovs-vsctl
Make sure that Open vSwitch is installed, that ovsdb-
server is running, and that
"ovs-vsctl show" works correctly.
You may wish to try "service openvswitch-switch start".
```

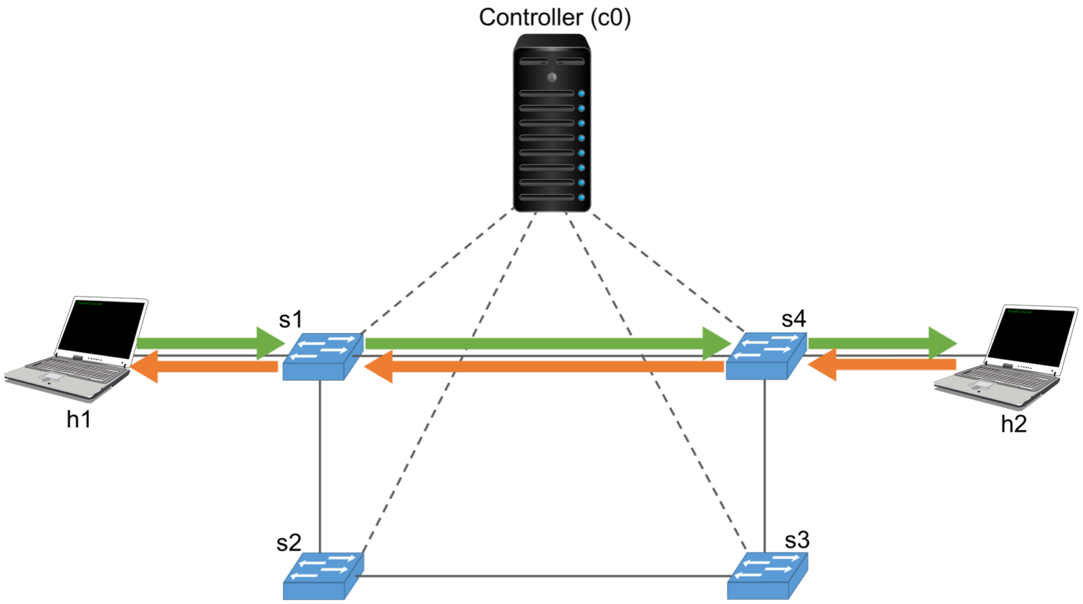
  - **Solution**

```
# Start the service of Open vSwitch
$ [sudo] service openvswitch-switch start
# Run Mininet again!
$ [sudo] mn
```

Task 2. Example of Ryu SDN

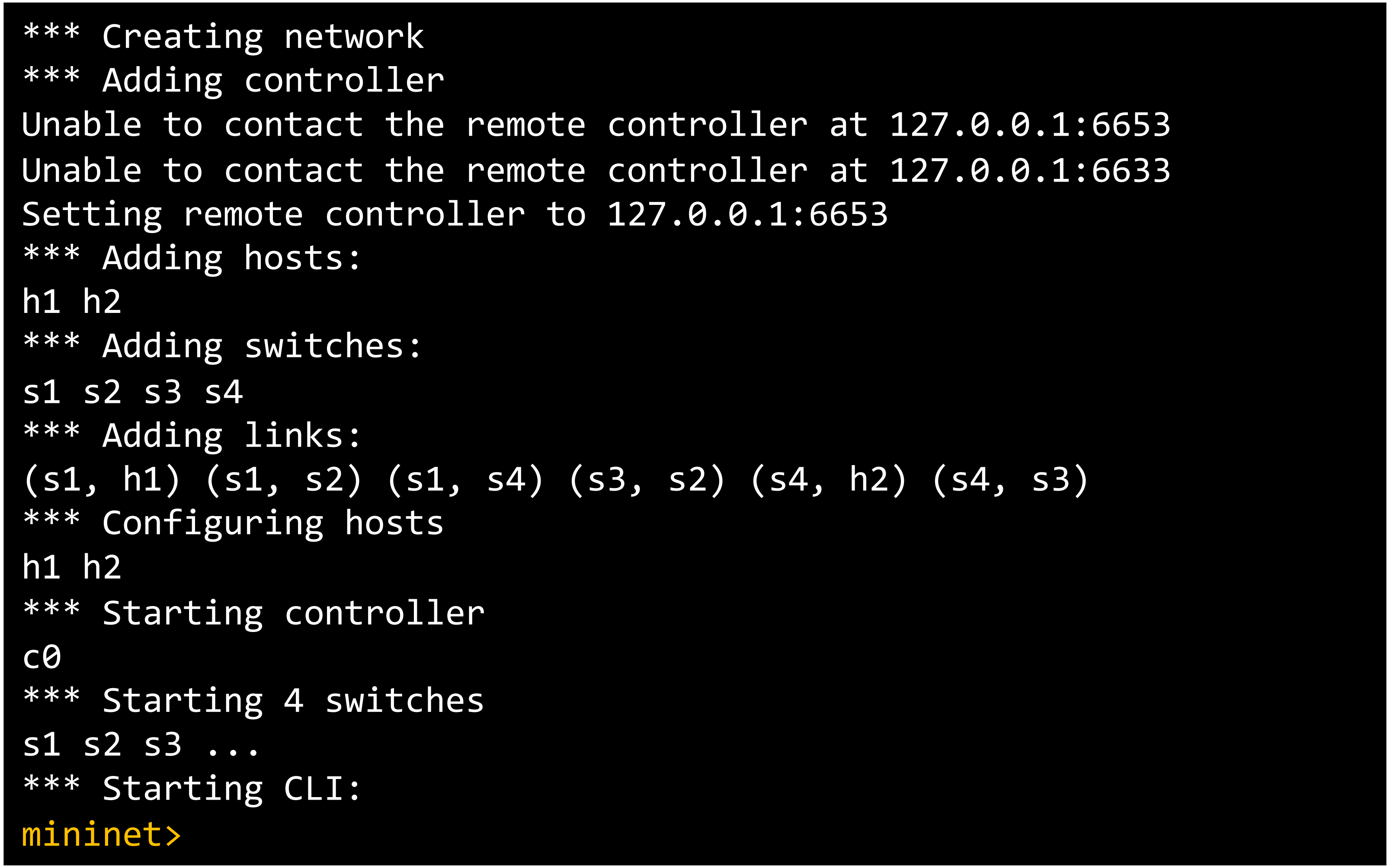
- **Step 1. Run Mininet topology**
  - Run topo.py in one terminal first

```
# Change the directory into
# /root/Lab3_Route_Configuration/src/
$ cd /root/Lab3_Route_Configuration/src/
# Run the topo.py with Mininet
$ [sudo] mn --custom topo.py --topo topo --link tc
--controller remote
```



Task 2. Example of Ryu SDN (cont.)

- The result after running topo.py



```
*** 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
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(s1, h1) (s1, s2) (s1, s4) (s3, s2) (s4, h2) (s4, s3)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

Task 2. Example of Ryu SDN (cont.)

- **Troubleshooting 1**
  - The following error may occur when you run topo.py or Mininet’s program

```
# Run the topo.py with Mininet
$ [sudo] mn --custom topo.py --topo topo --link tc
--controller remote
*** Creating network
......
Exception: Error creating interface pair (s1-eth1,s2-
eth1): RTNETLINK answers: File exists
```

  - **Solution:**

```
# If Mininet crashes for some reason, clean it up!
$ [sudo] mn -c
```

Task 2. Example of Ryu SDN (cont.)

- **Troubleshooting 2**
  - The following message which won’t affect in this lab may show when you run topo.py or your Mininet’s program

```
# Run the example code (topo.py)
$ [sudo] mn --custom topo.py --topo topo --link tc
--controller remote
*** Error setting resource limits. Mininet's
performance may be affected.
*** Creating network
*** Adding controller
......
```

Task 2. Example of Ryu SDN (cont.)

- **Step 2. Run Ryu manager with controller**
  - Run SimpleController.py in another terminal

```
# Change the directory into
# /root/Lab3_Route_Configuration/src/
$ cd /root/Lab3_Route_Configuration/src/
# Run the SimpleController.py with Ryu manager
$ [sudo] ryu-manager SimpleController.py --observe-
links
loading app controller.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app controller.py of SimpleController
instantiating app ryu.topology.switches of Switches
instantiating app ryu.controller.ofp_handler of
OFPHandler
```

Task 2. Example of Ryu SDN (cont.)

- **Step 3. How to leave the Ryu controller?**
  - Leave topo.py in one terminal first

```
# Leave the Mininet CLI
mininet> exit
```

  - Then, leave SimpleController.py in another terminal

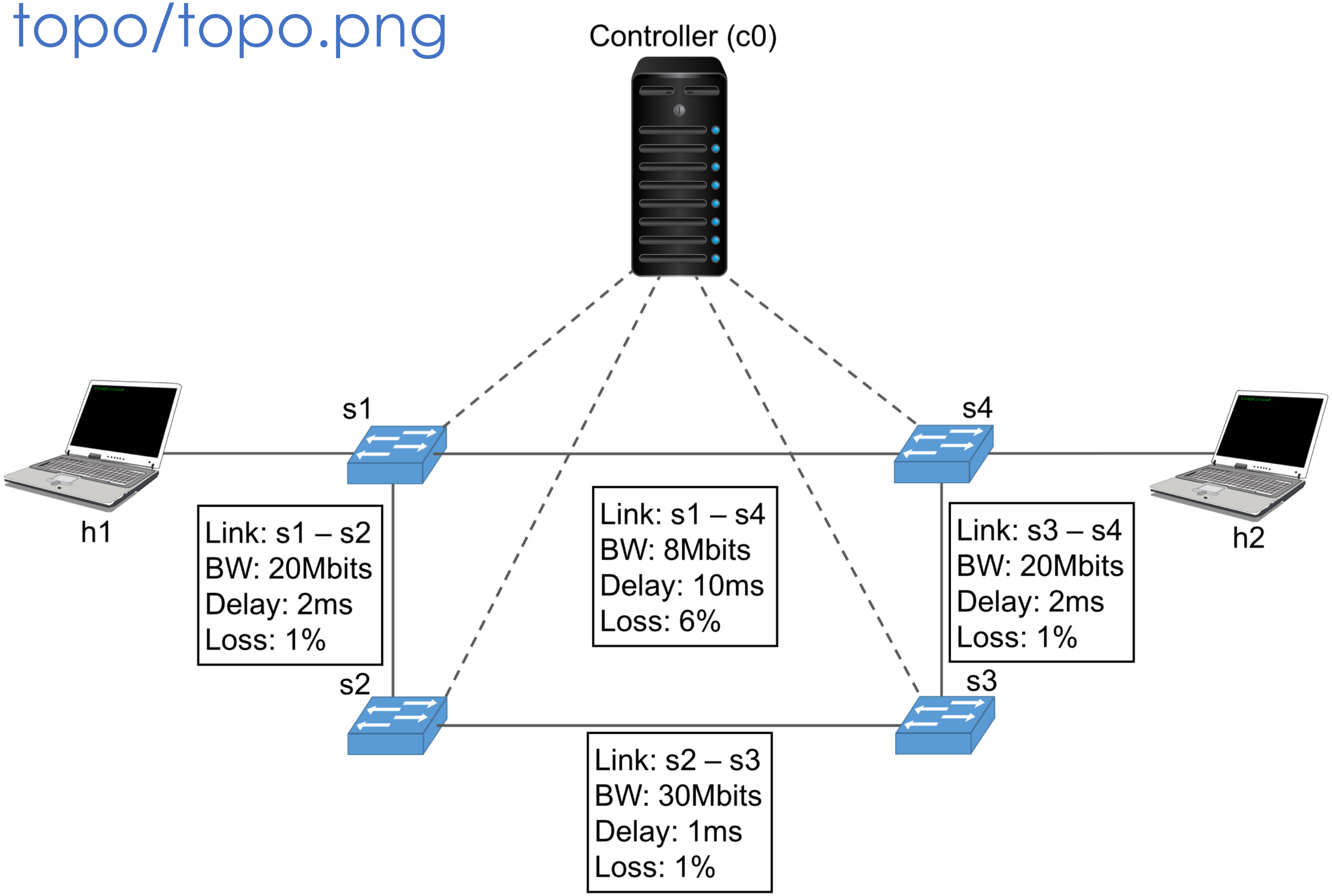
```
# Leave and stop the controller process
Ctrl-c
# Make sure “RTNETLINK” is clean indeed
$ mn -c
```

Task 3. Mininet Topology (cont.)

- **Step 1. Build the topology via Mininet**
  - **[TODO]** Modify topo.py to add the constraints (e.g., bandwidth, delay, and loss rate) according to /Lab3_Route_Configuration/src/topo/topo.png
  - You don’t need to set the bandwidth, delay, or loss rate if the figure don’t specify.

Task 3. Mininet Topology (cont.)

- Topology of /Lab3_Route_Configuration/src/topo/topo.png



Link: s1 – s2
BW: 20Mbits
Delay: 2ms
Loss: 1%

Link: s1 – s4
BW: 8Mbits
Delay: 10ms
Loss: 6%

Link: s3 – s4
BW: 20Mbits
Delay: 2ms
Loss: 1%

Link: s2 – s3
BW: 30Mbits
Delay: 1ms
Loss: 1%

Task 3. Mininet Topology (cont.)

- **Step 2. Run Mininet topology and controller**
  - Run topo.py in one terminal first

```
# Run the topo.py with Mininet
$ [sudo] mn --custom topo.py --topo topo --link tc
--controller remote
......
mininet>
```

  - Then, run SimpleController.py in another terminal

```
# Run the SimpleController.py with Ryu manager
$ [sudo] ryu-manager SimpleController.py –observe-links
loading app controller.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app controller.py of SimpleController
instantiating app ryu.topology.switches of Switches
instantiating app ryu.controller.ofp_handler of
OFPHandler
```

Task 3. Mininet Topology (cont.)

- **Troubleshooting 3**
  - The following message means your controller’s program has some error

```
$ ryu-manager SimpleController.py –observe-links
loading app SimpleController.py
Traceback (most recent call last):
  File "/usr/local/bin/ryu-manager", line 9, in
......
ImportError: No module named SimpleController.py
```

  - The following message means your topology’s program has some error

```
$ [sudo] mn --custom topo.py --topo topo --link tc
--controller remote
-------------------------------------------------------
Caught exception. Cleaning up...
SyntaxError: invalid syntax (topo.py, line 19)
```

Task 3. Mininet Topology (cont.)

- **Troubleshooting 4**
  - You can ping each link respectively by using the following command in the Mininet's CLI mode

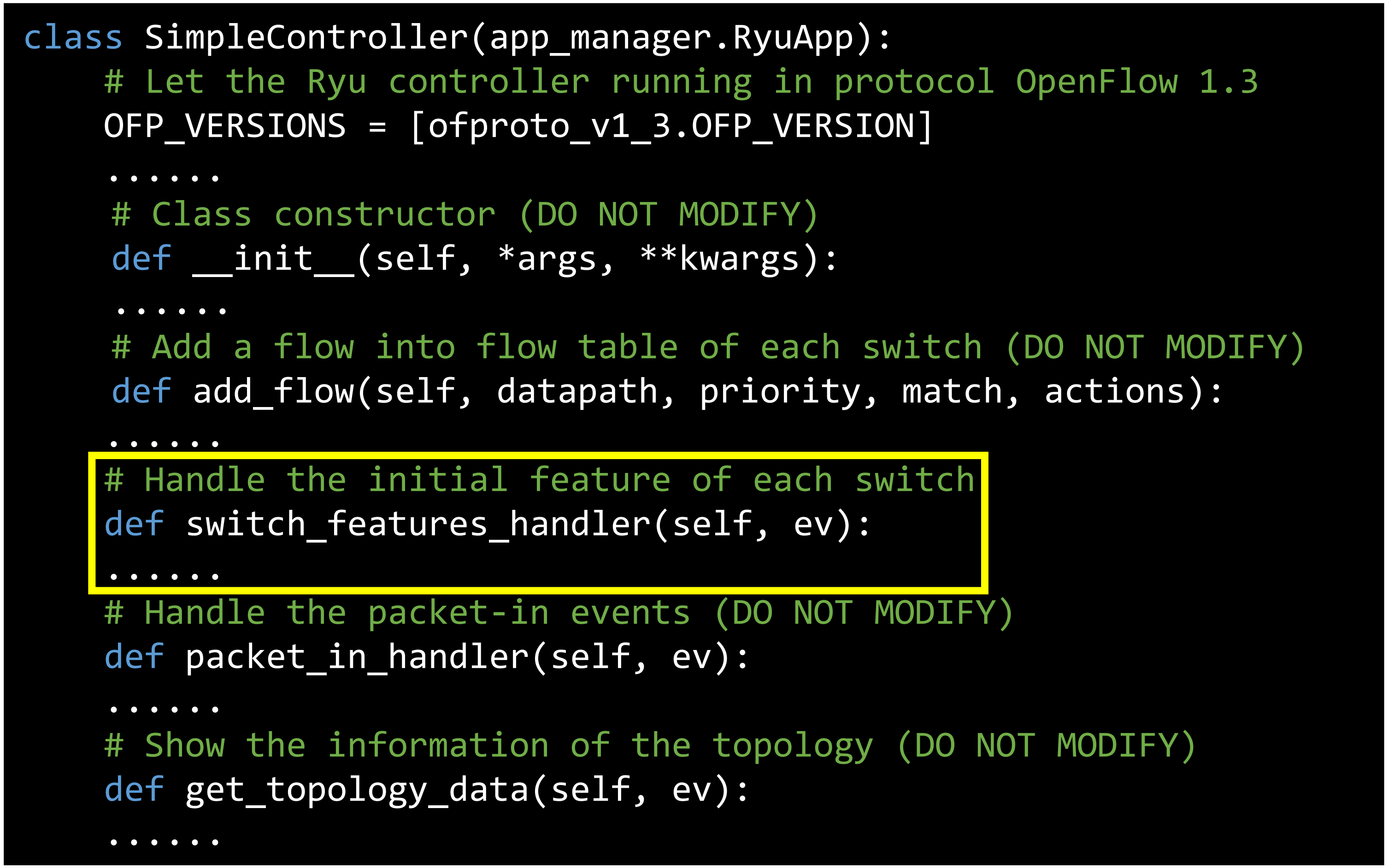
```
# Example of testing the connectivity between h1 and h2
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=31.8 ms
......
```

  - Please refer to the **Troubleshooting 1** for solving the following error when running your program

```
Exception: Error creating interface pair (s1-eth1,s2-eth1): RTNETLINK answers: File exists
```

Task 4. Ryu Controller

- **Step 1. Trace the code of Ryu controller**
  - Trace the example code SimpleController.py



```
class SimpleController(app_manager.RyuApp):
    # Let the Ryu controller running in protocol OpenFlow 1.3
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]
    ......
    # Class constructor (DO NOT MODIFY)
    def __init__(self, *args, **kwargs):
    ......
    # Add a flow into flow table of each switch (DO NOT MODIFY)
    def add_flow(self, datapath, priority, match, actions):
    ......
    # Handle the initial feature of each switch
    def switch_features_handler(self, ev):
    ......
    # Handle the packet-in events (DO NOT MODIFY)
    def packet_in_handler(self, ev):
    ......
    # Show the information of the topology (DO NOT MODIFY)
    def get_topology_data(self, ev):
    ......
```

Task 4. Ryu Controller

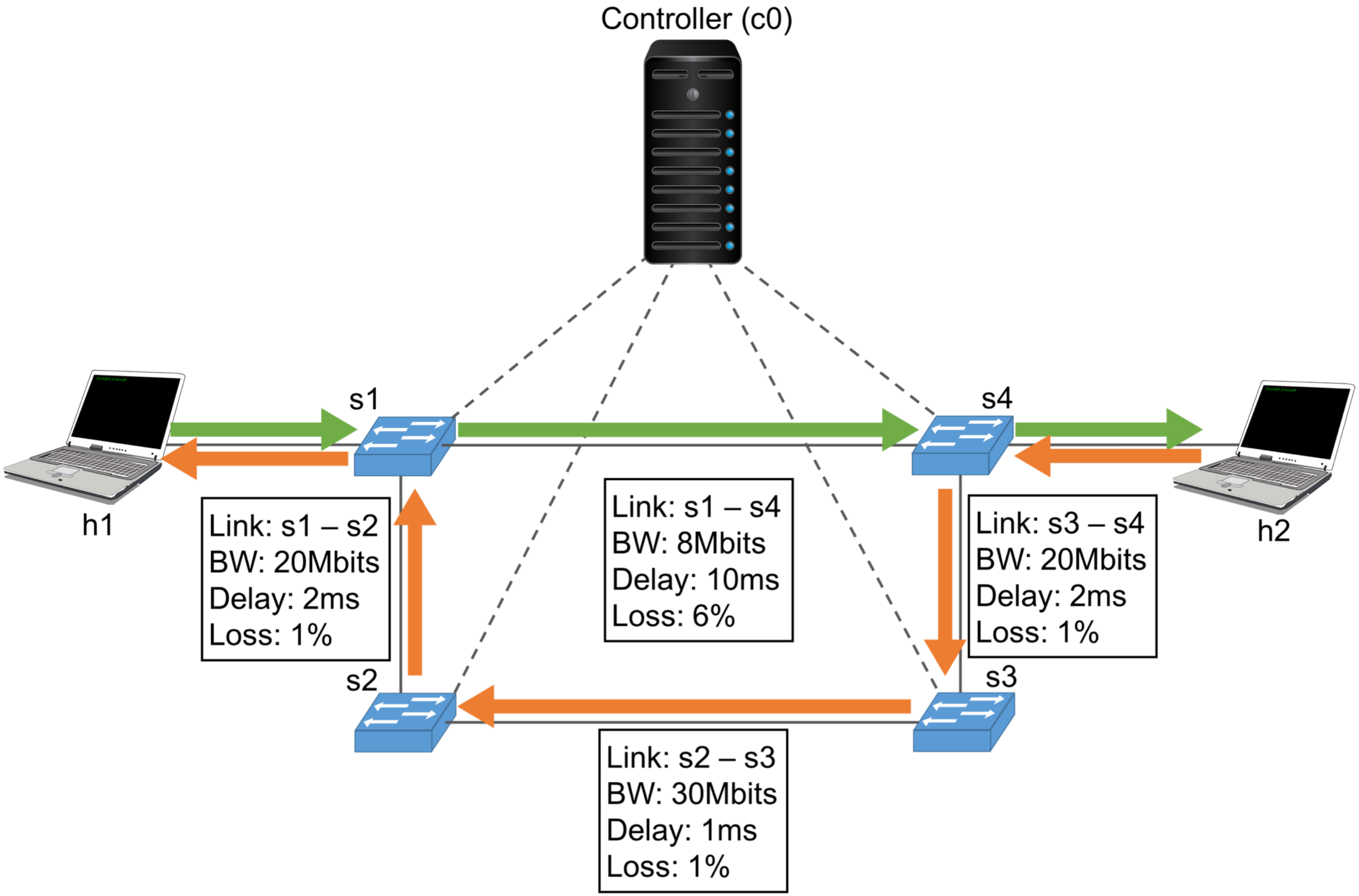
- **Step 2. Write another Ryu controller**
  - Duplicate the example code SimpleController.py and name it controller.py

```
# Make sure the current directory is
# /root/Lab3_Route_Configuration/src/
$ cp SimpleController.py controller.py
```

  - Follow the the forwarding rules in the next slide and modify controller.py
  - **[TODO]** You ONLY need to modify the method `switch_features_handler(self, ev)`

Task 4. Ryu Controller

- Step 3. Define forwarding rules (controller.py)



Task 5. Measurement

- **Step 1. Run topology with SimpleController.py**
  - Run topo.py in one terminal first

```
# Run the topo.py with Mininet
$ [sudo] mn --custom topo.py --topo topo --link tc
--controller remote
```

  - Then, run SimpleController.py in another terminal

```
# Run the SimpleController.py with Ryu manager
$ [sudo] ryu-manager SimpleController.py –observe-links
loading app SimpleController.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app SimpleController.py of
SimpleController
instantiating app ryu.topology.switches of Switches
instantiating app ryu.controller.ofp_handler of
OFPHandler
```

Task 5. Measurement

Step 2. Measure the bandwidth

- Use the following iPerf commands to measure the bandwidth in your network with **SimpleController**
- [TODO] Screenshot the output of iperf

```
# Run in the iPerf command in Mininet CLI
mininet> h1 iperf -s -u -i 1 –p 5566 > ./out/result1 &
mininet> h2 iperf -c 10.0.0.1 -u –i 1 –p 5566
```

- Leave topo.py in one terminal first

```
# Leave the Mininet CLI
mininet> exit
```

- Then, leave SimpleController.py in another terminal

```
# Leave controller process and clean “RTNETLINK”
Ctrl-c
$ mn -c
```

Task 5. Measurement

Step 3. Run topology with controller.py

- Run topo.py in one terminal first

```
# Run the topo.py with Mininet
$ [sudo] mn --custom topo.py --topo topo --link tc
--controller remote
```

- Then, run controller.py in another terminal

```
# Run the controller.py with Ryu manager
$ [sudo] ryu-manager controller.py –observe-links
loading app controller.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app controller.py of SimpleController
instantiating app ryu.topology.switches of Switches
instantiating app ryu.controller.ofp_handler of
OFPHandler
```

Task 5. Measurement

- **Step 4. Measure the bandwidth**
  - Use the following iPerf commands to measure the bandwidth in your network with **controller.py**
  - [TODO] Screenshot the output of iperf

```
# Run in the iPerf command in Mininet CLI
mininet> h1 iperf -s -u -i 1 –p 5566 > ./out/result2 &
mininet> h2 iperf -c 10.0.0.1 -u –i 1 –p 5566
```

  - Leave topo.py in one terminal first

```
# Leave the Mininet CLI
mininet> exit
```

  - Then, leave controller.py in another terminal

```
# Leave controller process and clean “RTNETLINK”
Ctrl-c
$ mn -c
```

Task 6. Report

- **Your Report.pdf must include**
  - **Execution**
    - Steps for finishing this lab and how to run your program.
      - Not just copy the content from this slide
    - What is the meaning of the executing command (both Mininet and Ryu controller)?
      - mn …
      - ryu-manager …
    - Screenshot the result of using iPerf command (both SimpleController.py and controller.py)
  - **Discussion (Q&A)**
    - Answer the questions on next page

Task 6. Report

- **Discussion**
  1. Describe the differences between packet-in and packet-out in detail.
  2. What is “table-miss” in SDN?
  3. Why is “`(app_manager.RyuApp)`” adding after the declaration of class in controller.py?
  4. What is the meaning of “`datapath`” in controller.py?
  5. Why need to set “`ip_proto=17`”and “`eth_type=0x0800`” in the flow entry?
  6. Compare the differences between the iPerf results of SimpleController.py and controller.py. Which forwarding rule is better? Why?

Submission

- **Submit your works to your GitHub repository**

```
# In container folder: Lab3_Route_Configuration/
# Add all files into staging area
$ git add .
# Commit your files
$ git commit -m "YOUR OWN COMMIT MESSAGE"
# Push your files to remote
$ git push origin master
```

  - Go to our GitHub group to check your repository successfully updates
    - https://github.com/nctucn

Submission

- **Push your works to GitHub repository (nctucn)**
  - **Trace files** (./src/out/)
    - result1
    - result2
  - **Python code** (./src/)
    - topo.py
    - controller.py
  - **Report** (./)
    - Report.pdf

- **No need to submit to new E3**

Grading Policy

- Deadline – Dec 20, 2019. 23:59
- Python program and result correctness– 50 %
  - topo.py
  - controller.py
- Report – 50 %

References

- **Ryu SDN**
  - English
    - Ryubook Documentation
    - Ryubook [PDF]
    - Ryu 4.30 Documentation
    - Ryu Controller Tutorial
    - OpenFlow 1.3 Switch Specification
  - Chinese
    - Ryubook 說明文件
    - GitHub - Ryu Controller 教學專案
    - Ryu SDN 指南 – Pengfei Ni
    - OpenFlow 通訊協定