Computer Networks
@CS.NCTU

Lab. 2: Route Configuration

Instructor: Kate Lin

Deadline: 2022.01.09 23:59

Objectives

In this lab, we are going to write three Python programs with **Ryu** *SDN framework* to build a simple software-defined network and compare the differences between three 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
3. Learn how to add a new flow rule to find a (optimal) path

TODO

1. Modify topo.py according to the topology we provide
2. Modify the example code SimpleController.py.

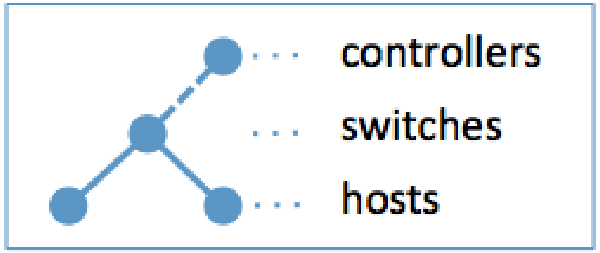
Search **“[TODO]”** in this slide and codes to figure out where you should modify

Overview

Mininet

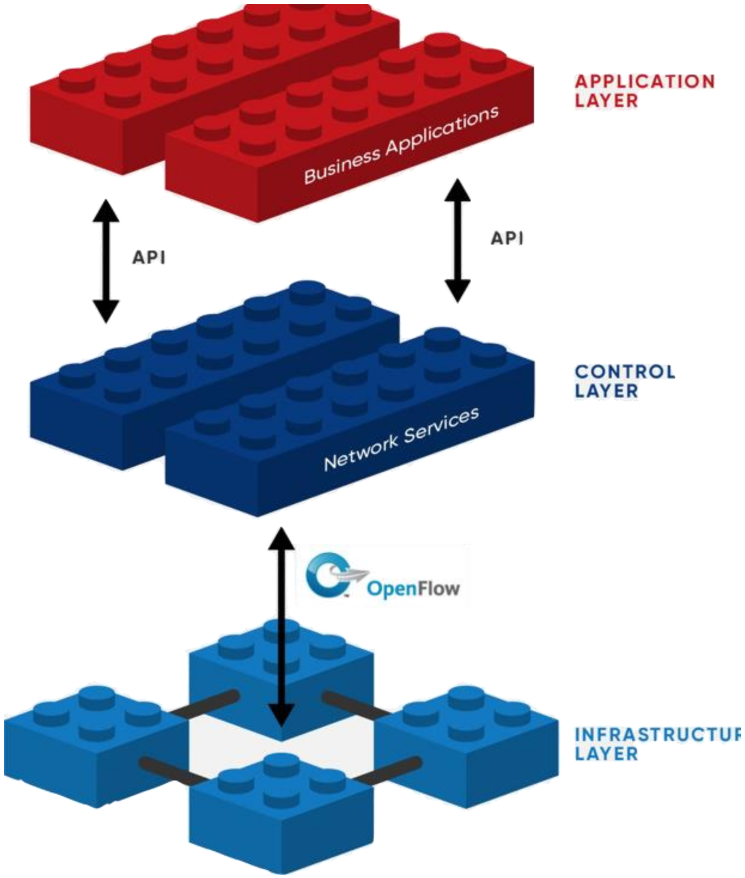
- Mininet is a network emulator
  - Overview of Mininet - http://mininet.org/overview/
  - We have provided you a VM that has installed Mininet
- Create a realistic virtual network, running real kernel, switch and application code, on a single machine (VM, cloud or native)
- Run a collection of end-hosts,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

```
> sudo mn
```



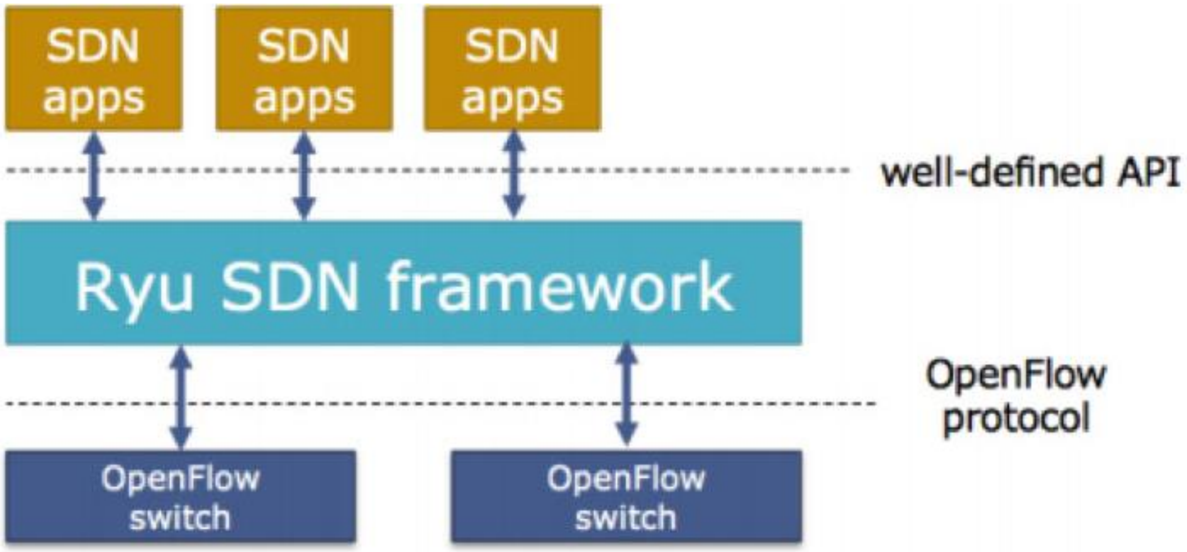
Software-Defined Networking (SDN)

- 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

```
lab2-<GITHUB_ID>/                 # This is ./ in this repository
|--- src/                         # Folder of source code
     |--- topo/                   # Folder of topology figure
          |--- topo.png
     |--- out/                    # Output files
     |--- SimpleController.py     # Example code of controller
     |--- controller1.py          # Your program should be here!
     |--- controller2.py          # Your program should be here!
     |--- topo.py                 # Your program should be here!
     |--- AdaptiveController.py   # Your program should be here!
|--- Report.pdf                   # Your report
```

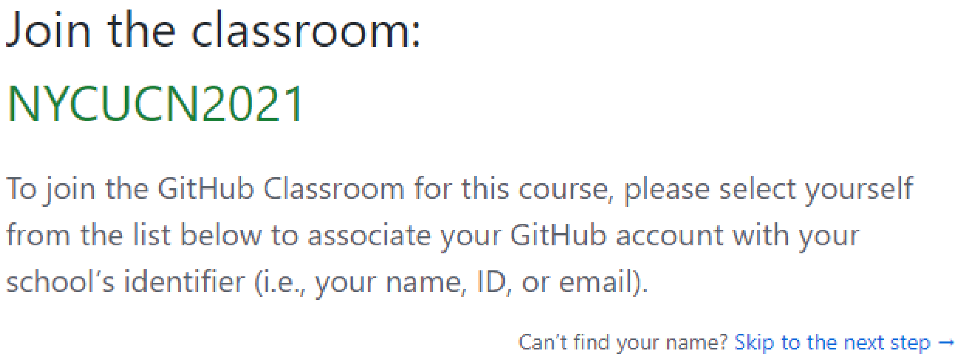
Tasks

Tasks

1. Environment Setup
2. Example of Ryu SDN
3. Mininet Topology
   - modify topo.py
4. Ryu Controller
   - modify controller1.py and controller2.py
5. Measurement
6. Flow-Removed Events
   - modify AdaptiveController.py
7. Report

Task 1. Environment Setup

- **Step 1. Join the GitHub Classroom Lab2**
  - Click the following link to join this lab
    - GitHub Classroom Lab2
  - Skip to the next step when you see this



Join the classroom:

NYCUCN2021

To join the GitHub Classroom for this course, please select yourself from the list below to associate your GitHub account with your school's identifier (i.e., your name, ID, or email).

Can't find your name? Skip to the next step →

  - Go to our GitHub group to see your repository
    - https://github.com/NYCUCN/lab2-<GITHUB_ID>
  - You will have an initial repository we prepared

Task 1. Environment Setup (cont.)

- **Step2. Install Oracle VM VirtualBox**
  - Oracle VM VirtualBox – Downloads

- **Step3. Download TA’s .ova file and import it into your Oracle VM VirtualBox**
  - lab2
  - Password: cn2021
  - How To Use OVA Files with VirtualBox (alphr.com)

Task 1. Environment Setup (cont.)

- **Step4. Download required files from Github**

```
$ git clone https://github.com/NYCUCN2021/lab2-<GITHUB_ID>.git
```

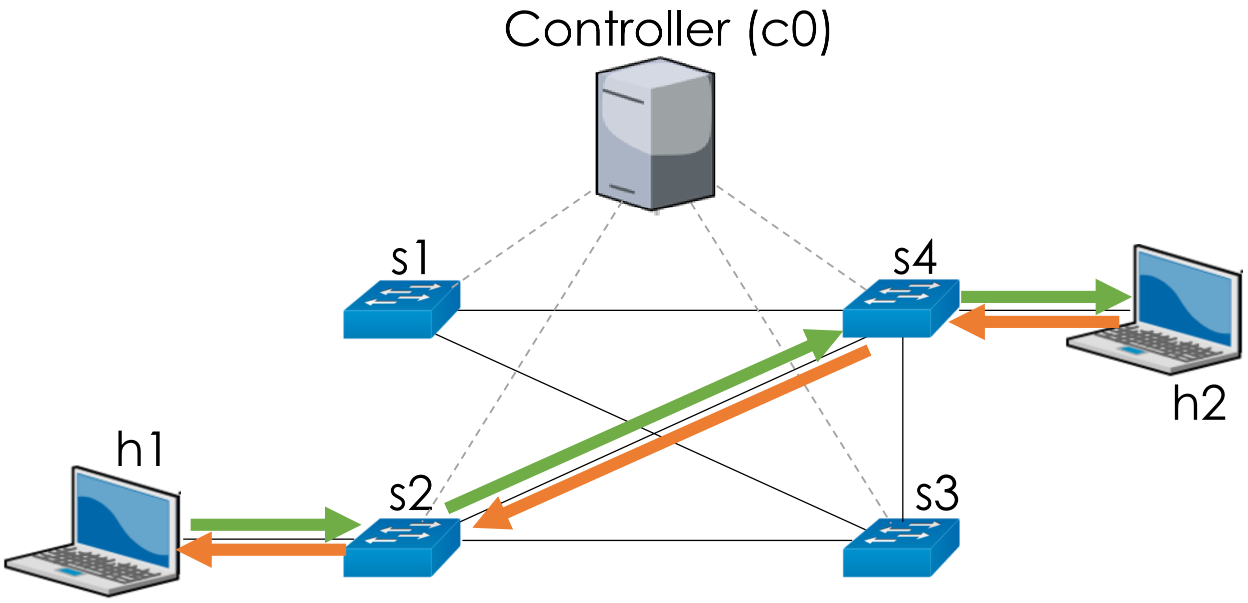
- **Step5. Get and set repository for global options**

```
$ cd lab2-<GITHUB_ID>/
$ git config --global user.name "<NAME>"
$ git config --global user.email "<EMAIL>"
```

Task 2. Example of Ryu SDN

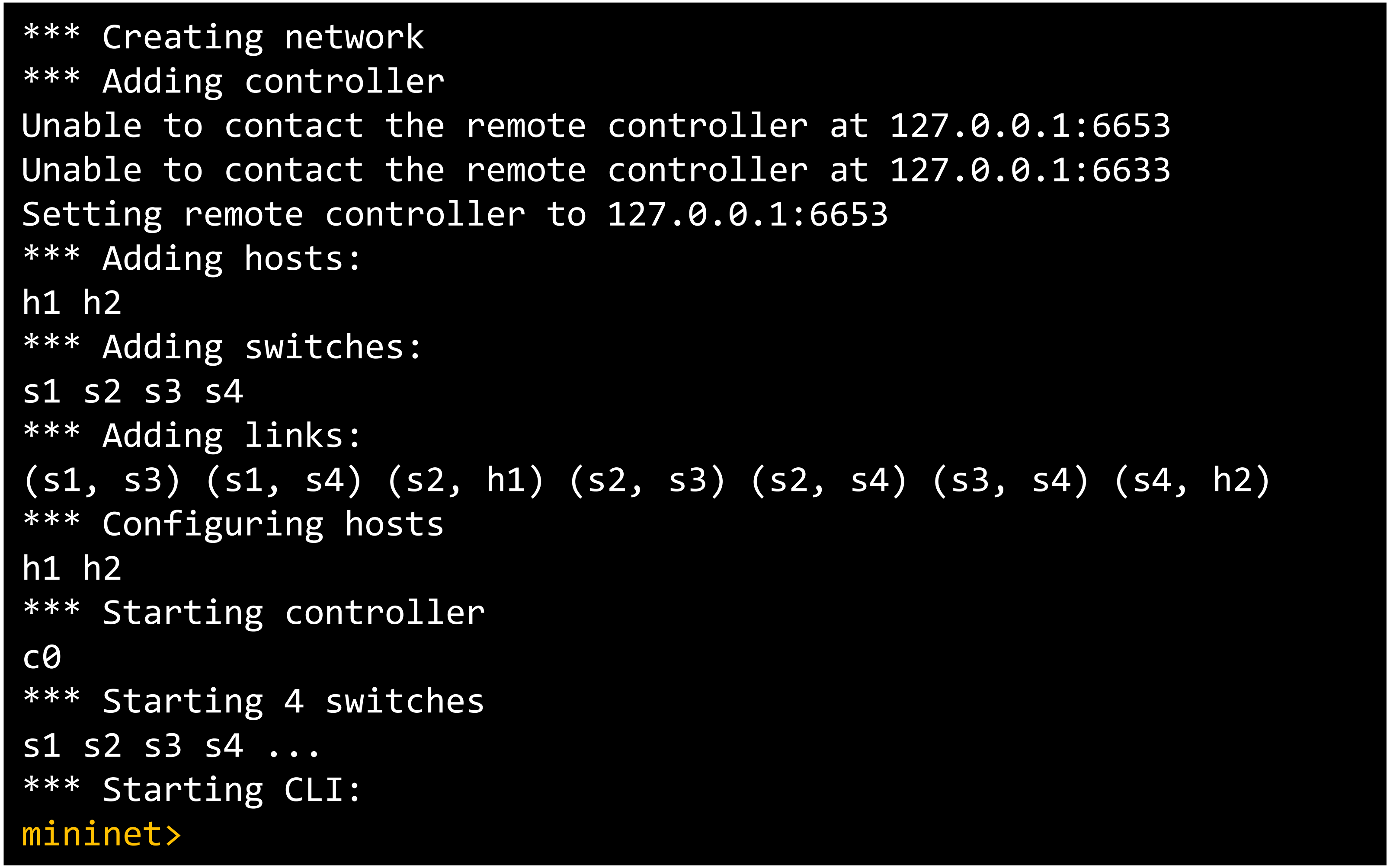
- **Step 1. Run Mininet topology**
  - Run topo.py in one terminal first

```
# Change the directory into /root/lab2-<GITHUB_ID>/src/
$ cd /root/lab2-<GITHUB_ID>/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, s3) (s1, s4) (s2, h1) (s2, s3) (s2, s4) (s3, s4) (s4, h2)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** 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 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.)

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

```
# Change the directory into /root/lab2-<GITHUB_ID>/src/
$ cd /root/lab2-<GITHUB_ID>/src/
# 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 2. Example of Ryu SDN (cont.)

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

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

  - Then, leave SimpleController.py in another terminal

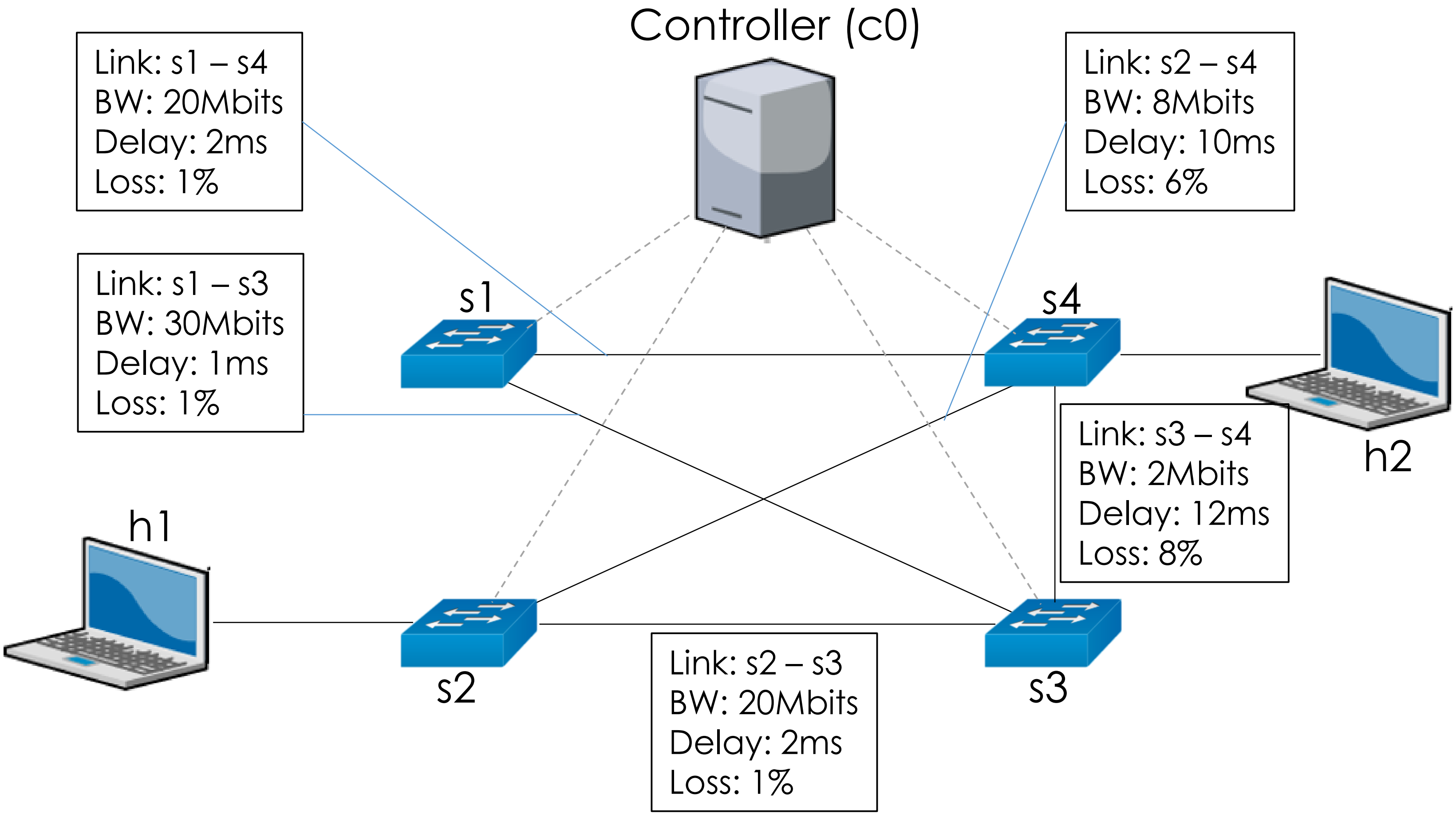
```
# Leave and stop the controller process
Ctrl-z
# Make sure “RTNETLINK” is clean indeed
$ mn -c
```

Task 3. Mininet Topology

- **Step 1. Build the topology via Mininet**
  - **[TODO]** Modify topo.py to add constraints (e.g., bandwidth, delay and loss rate) according to /lab2-<GITHUB_ID>/src/topo/topo.png
  - You don’t need to set the bandwidth, delay or loss rate if the figure doesn’t specify

Task 3. Mininet Topology (cont.)

- Topology of /lab2-<GITHUB_ID>/src/topo/topo.png



Link: s1 – s4
BW: 20Mbits
Delay: 2ms
Loss: 1%

Link: s1 – s3
BW: 30Mbits
Delay: 1ms
Loss: 1%

Link: s2 – s4
BW: 8Mbits
Delay: 10ms
Loss: 6%

Link: s3 – s4
BW: 2Mbits
Delay: 12ms
Loss: 8%

Link: s2 – s3
BW: 20Mbits
Delay: 2ms
Loss: 1%

Task 3. Mininet Topology (cont.)

Step 2. Run Mininet topology and controller

- Run topo.py in one terminal first

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

- Then, run SimpleController.py in another terminal

```
# Run 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 3. Mininet Topology (cont.)

- **Troubleshooting 2**
  - The following message means your controller’s program has some error

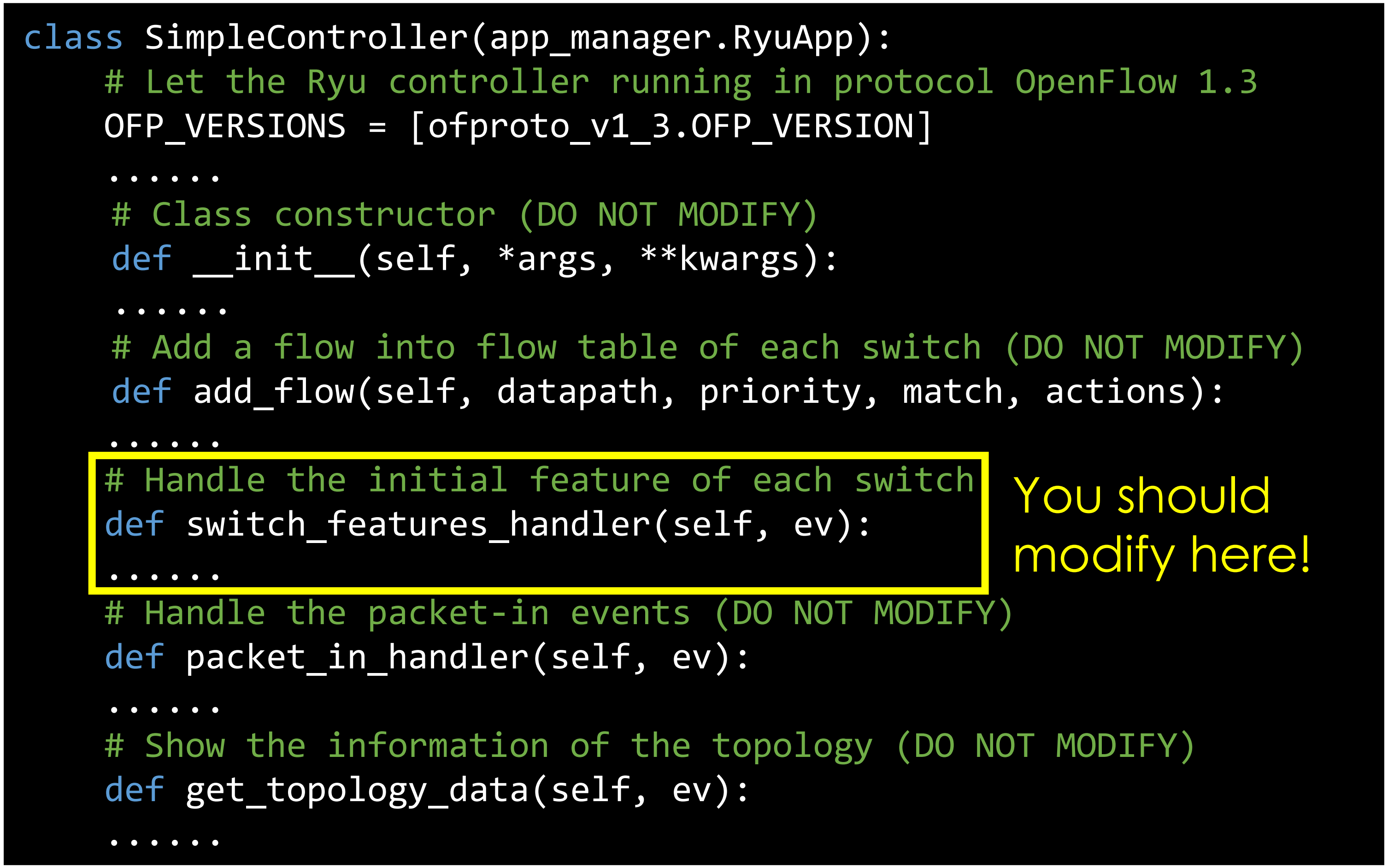
```
$ ryu-manager SimpleController.py --observe-links
loading app SimpleController.py
Traceback (most recent call last):
  File “/usr/local/bin/ryu-manager”, line 9, in
......
ImportError: No module named SimpleController.py
```

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

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

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):
    ......
```

You should modify here!

Task 4. Ryu Controller

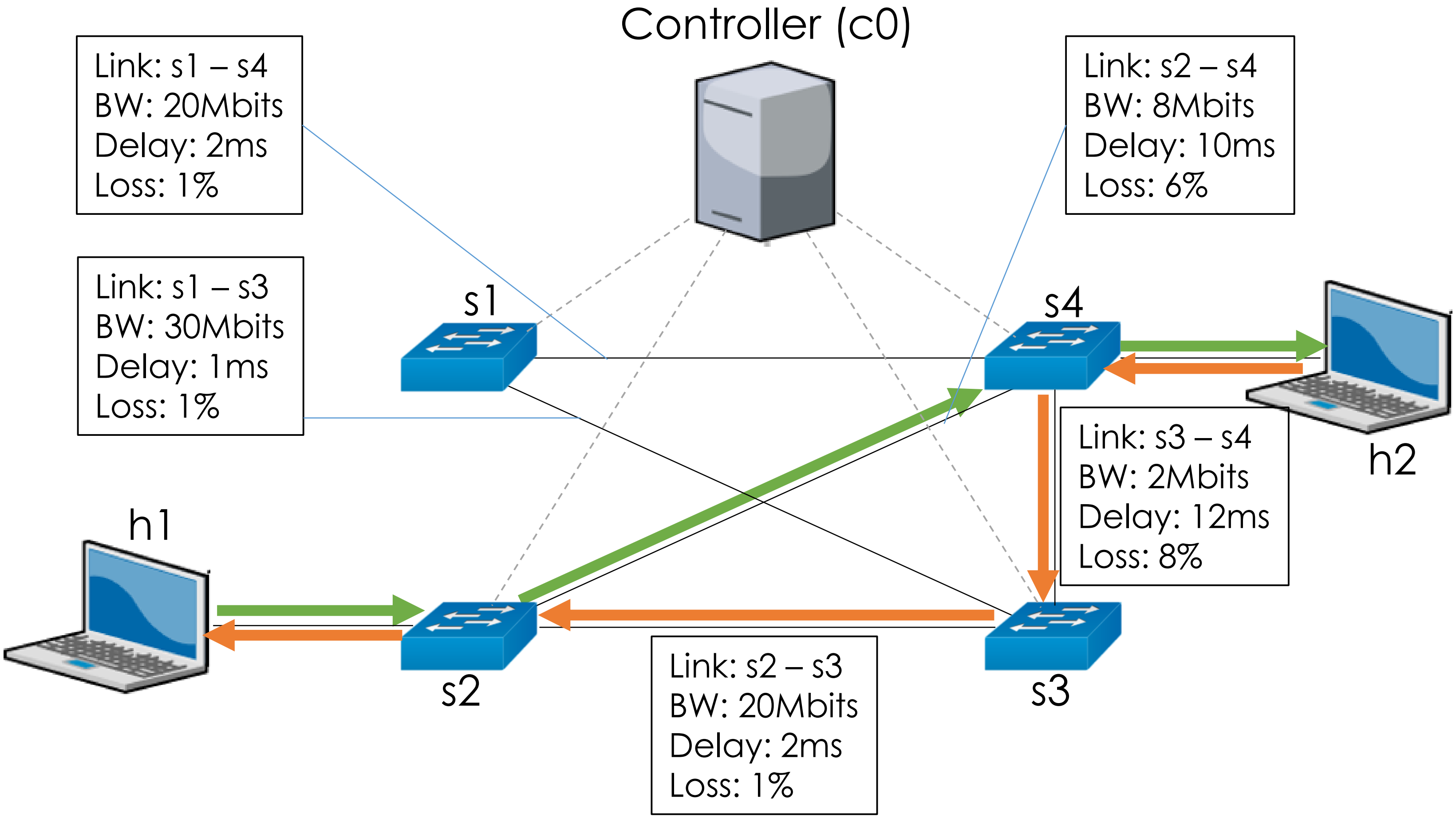
- **Step 2. Write another Ryu controller**
  - Duplicate the example code SimpleController.py and name it controller1.py

```
# Make sure the current directory is
# /root/lab2-<GITHUB_ID>/src/
$ cp SimpleController.py controller1.py
```

  - Follow the the forwarding rules in the next slide and modify controller1.py
  - **[TODO]** Modify the function `switch_features_handler(self, ev)` `to define new flow rules`

Task 4. Ryu Controller (cont.)

- **Step 3. Define forwarding rules** (controller1.py)



Link: s1 – s4
BW: 20Mbits
Delay: 2ms
Loss: 1%

Link: s1 – s3
BW: 30Mbits
Delay: 1ms
Loss: 1%

Link: s2 – s4
BW: 8Mbits
Delay: 10ms
Loss: 6%

Link: s3 – s4
BW: 2Mbits
Delay: 12ms
Loss: 8%

Link: s2 – s3
BW: 20Mbits
Delay: 2ms
Loss: 1%

Task 4. Ryu Controller

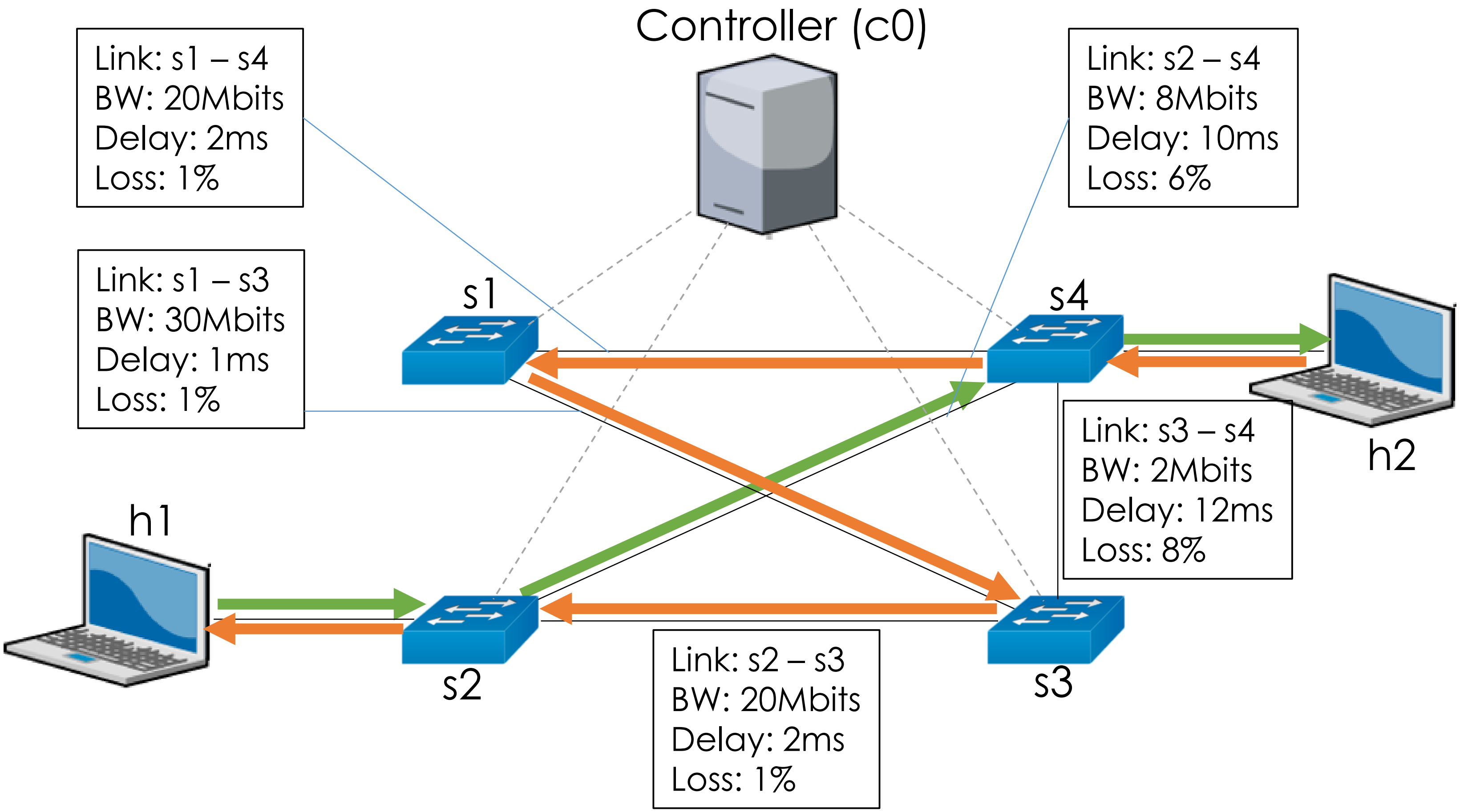
- **Step 4. Write another Ryu controller**
  - Duplicate the example code SimpleController.py and name it controller2.py

```
# Make sure the current directory is
# /root/lab2-<GITHUB_ID>/src/
$ cp SimpleController.py controller2.py
```

  - Follow the the forwarding rules in the next slide and modify controller2.py
  - **[TODO]** You ONLY need to modify the function `switch_features_handler(self, ev)`

Task 4. Ryu Controller (cont.)

- **Step 5. Define forwarding rules** (controller2.py)



Link: s1 – s4
BW: 20Mbits
Delay: 2ms
Loss: 1%

Link: s1 – s3
BW: 30Mbits
Delay: 1ms
Loss: 1%

Link: s2 – s4
BW: 8Mbits
Delay: 10ms
Loss: 6%

Link: s3 – s4
BW: 2Mbits
Delay: 12ms
Loss: 8%

Link: s2 – s3
BW: 20Mbits
Delay: 2ms
Loss: 1%

Task 5. Measurement

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

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

  - Then, run SimpleController.py in another terminal

```
# Run 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 (cont.)

- **Step 2-1. Ping**
  - Use the following ping command to make sure that ICMP and APR packets can reach the destination
  - Stop ping by "ctrl-c" once the ping command received a response from h2

```
# Run the ping command in Mininet CLI
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=1828 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=10.1 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=10.0 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=10.0 ms
^C
```

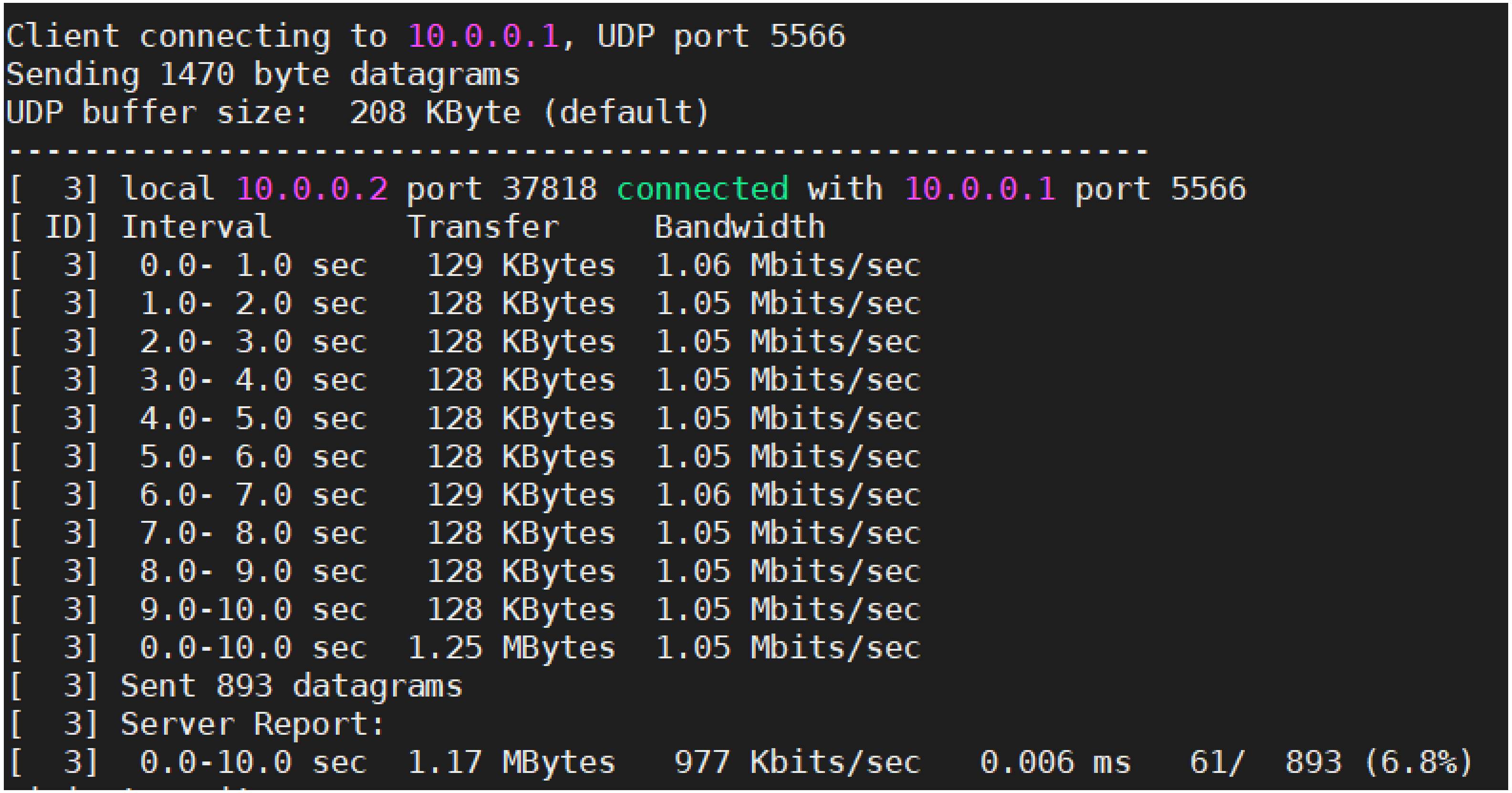
Task 5. Measurement (cont.)

- **Step 2-2. Measure the bandwidth**
  - Use the following iPerf commands to measure the bandwidth in your network with **SimpleController.py**
  - Remember to create the folder "out/"
  - [TODO] Take a screenshot of the output of iperf

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

Task 5. Measurement (cont.)

- **Example screenshot of the output of iperf**



```
Client connecting to 10.0.0.1, UDP port 5566
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.2 port 37818 connected with 10.0.0.1 port 5566
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  6.0- 7.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.17 MBytes   977 Kbits/sec   0.006 ms   61/  893 (6.8%)
```

Task 5. Measurement (cont.)

- **Step 2-3. Check the number of packets**
  - The controller will output the number of packets which the forwarding rules on switch 2 (s2) matched every 10 seconds
  - [TODO] Take a screenshot of the output until there is no more change in the number of packets (iperf command finished)

```
# In the terminal runnung the controller
switch 2: count 0 packets
switch 2: count 526 packets
switch 2: count 832 packets
switch 2: count 832 packets
```

(just an example)

Task 5. Measurement (cont.)

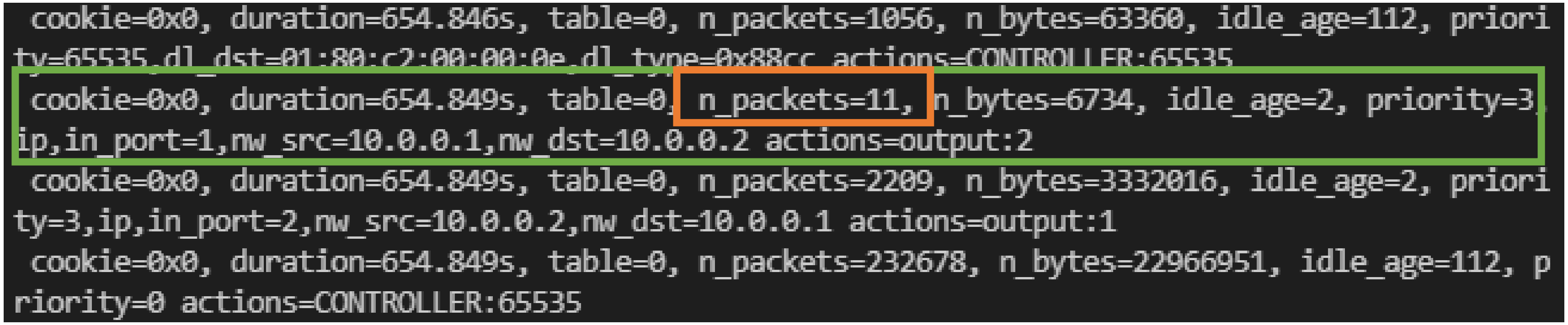
- **Step 2-4. Dump flow rules**
  - Output the forwarding rules on switch 2 (s2) using the following command

```
$ sh ovs-ofctl dump-flows s2
```

  - [TODO] Take a screenshot of the output
    - Mark the forwarding rules of priority=3
    - Mark the number of packets which the forwarding rules of priority=3 on switch 2 (s2) matched

Task 5. Measurement (cont.)

- Example screenshot of the output of “dump-flows”



```
 cookie=0x0, duration=654.846s, table=0, n_packets=1056, n_bytes=63360, idle_age=112, priori
ty=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=654.849s, table=0, n_packets=11, n_bytes=6734, idle_age=2, priority=3,
ip,in_port=1,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=654.849s, table=0, n_packets=2209, n_bytes=3332016, idle_age=2, priori
ty=3,ip,in_port=2,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=654.849s, table=0, n_packets=232678, n_bytes=22966951, idle_age=112, p
riority=0 actions=CONTROLLER:65535
```

Task 5. Measurement (cont.)

Step 3. Run topology with controller1.py

- Run topo.py in one terminal first

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

- Then, run controller1.py in another terminal

```
# Run controller1.py with Ryu manager
$ [sudo] ryu-manager controller1.py --observe-links
loading app controller1.py
loading app ryu.controller.ofp_handler
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app ryu.topology.switches of Switches
instantiating app controller1.py of SimpleController
```

Task 5. Measurement (cont.)

- **Step 4-1. Ping**
  - Use the following ping command to make sure that ICMP and APR packets can reach the destination
  - Stop ping by "ctrl-c" once the ping command received a response from h2

```
# Run the ping command in Mininet CLI
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=1828 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=10.1 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=10.0 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=10.0 ms
^C
```

Task 5. Measurement (cont.)

Step 4-2. Measure the bandwidth

- Use the following iPerf commands to measure the bandwidth in your network with **controller1.py**
- [TODO] Take a screenshot of the output of iperf

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

Task 5. Measurement (cont.)

- **Step 4-3. Check the number of packets**
  - The controller will output the number of packets which the forwarding rules on switch 2 (s2) matched every 10 seconds
  - **[TODO]** Take a screenshot of the output until there is no more change in the number of packets (iperf command finished)

Task 5. Measurement (cont.)

- **Step 4-4. Dump flow rules**
  - Output the forwarding rules on switch 2 (s2) using the following command in in Mininet

```
$ sh ovs-ofctl dump-flows s2
```

  - **[TODO]** Take a screenshot of the output
    - Mark the forwarding rules of priority=3
    - Mark the number of packets which the forwarding rules of priority=3 on switch 2 (s2) matched

Task 5. Measurement (cont.)

Step 5. Run topology with controller2.py

- Run topo.py in one terminal first

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

- Then, run controller2.py in another terminal

```
# Run controller2.py with Ryu manager
$ [sudo] ryu-manager controller2.py --observe-links
loading app controller2.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app controller2.py of SimpleController
instantiating app ryu.topology.switches of Switches
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Task 5. Measurement (cont.)

- **Step 6-1. Ping**
  - Use the following ping command to make sure that ICMP and APR packets can reach the destination
  - Stop ping by "ctrl-c" once the ping command received a response from h2

```
# Run the ping command in Mininet CLI
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=1828 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=10.1 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=10.0 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=10.0 ms
^C
```

Task 5. Measurement (cont.)

Step 6-2. Measure the bandwidth

- Use the following iPerf commands to measure the bandwidth in your network with **controller2.py**
- [TODO] Take a screenshot of the output of iperf

```
# Run the iPerf command in Mininet CLI
mininet> h1 iperf -s -u -i 1 –p 5566 > ./out/result3 &
mininet> h2 iperf -c 10.0.0.1 -u
```

Task 5. Measurement (cont.)

- **Step 6-3. Check the number of packets**
  - The controller will output the number of packets which the forwarding rules on switch 2 (s2) matched every 10 seconds
  - **[TODO]** Take a screenshot of the output until there is no more change in the number of packets (iperf command finished)

Task 5. Measurement (cont.)

- **Step 6-4. Dump flow rules**
  - Output the forwarding rules on switch 2 (s2) using the following command in Mininet

```
$ sh ovs-ofctl dump-flows s2
```

  - **[TODO]** Take a screenshot of the output
    - Mark the forwarding rules of priority=3
    - Mark the number of packets which the forwarding rules of priority=3 on switch 2 (s2) matched

Task 6. Flow-Removed Event

- Duplicate the example code SimpleController.py and name it AdaptiveController.py
- **[TODO]**
  - AdaptiveController.py works as follows
    - Measure the bandwidth of the three "orange" paths as mentioned (hint: each measuring for 5 seconds)
    - Detected Flow-Removed event and add new flow entry
    - Finally config the best path and output the path number (controller1.py: path 1, controller2.py: path 2, SimpleController.py: path3)
- [Hint 1] Initially run iperf in a long period of time before the controller is launched
- [Hint 2] n_packets in the flow rule is related to bandwidth

Task 7. Report

- **Your Report.pdf must include**
  - **Execution**

    Part1 : Run Mininet and Ryu controller

    - Steps for running mininet and Ryu controller to ping successfully from host to host.
      - Not just copy the content from this slide
    - What is the meaning of the executing command (both Mininet and Ryu controller)?
      - Mn, Ryu-manager …
    - Screenshots

    Part2 : (on next page)

Task 7. Report

- **Execution**

  Part2 : Handling flow-removed events

  - Explain your code as detailed as possible, should including
    - The flow rule you can find (original, different or optimized path)
    - How’s it working

  Part3 : **Problems encountered**

  - Problems you met while doing this lab
  - Any advices

Task 7. Report (cont.)

- **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 SimpleController.py?
  4. What is the meaning of "`datapath`" in SimpleController.py?
  5. Why need to set "`eth_type=0x0800`" in the flow entry?
  6. Compare the differences between the iPerf results of SimpleController.py, controller1.py and controller2.py. Which forwarding rule is better? Why?

- **You can write your report in English or Chinese**

Submission

- Submit your works to your GitHub repository

```
# In container folder: lab2-<GITHUB_ID>/
# Add files into staging area
$ git add <file>
# Commit your files
$ git commit -m "YOUR OWN COMMIT MESSAGE"
# Push your files to remote
$ git push origin master
```

Submission

- **Push your works to GitHub repository (nctucn)**
  - **Trace files** (./src/out)
    - result1
    - result2
    - result3
  - **Python code** (./src)
    - topo.py
    - controller1.py
    - controller2.py
    - AdaptiveController.py (if any)
  - **Report** (./)
    - Report.pdf

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

Grading Policy

- **Deadline – Jan. 9, 2022. 23:59**
- **Grade**
  - **Python program and result correctness – 60%**
    - **topo.py, controller1.py, controller2.py, and AdaptiveController.py**
  - **Report – 40%**
- Late Policy
  - (Your score) * 0.8^D, where D is the number of days over due
- Cheating Policy
  - Academic integrity: Homework must be your own – cheaters share the score
  - Both the cheaters and the students who aided the cheater equally share the score

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