Computer Networks
@CS.NCTU

Lab. 2: Network Topology with Mininet

Due Dec. 01 (Sun) 23:59

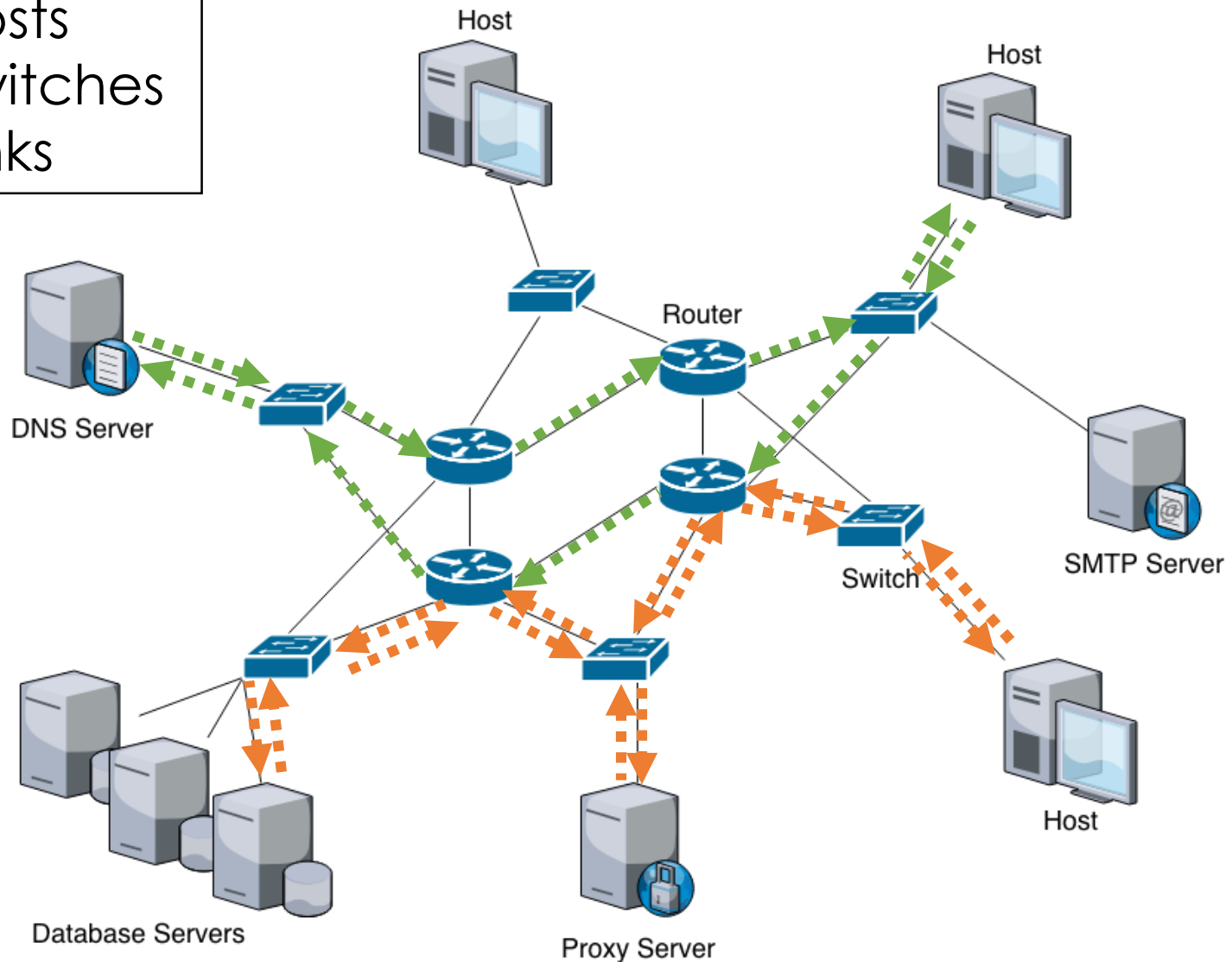
Objectives

In this lab, we are going to write a Python program which can generate a network topology for Mininet and use iPerf to measure the bandwidth of a path in this topology

1. Learn how to create a network topology for Mininet
2. Learn how to measure the bandwidth by using iPerf in Mininet

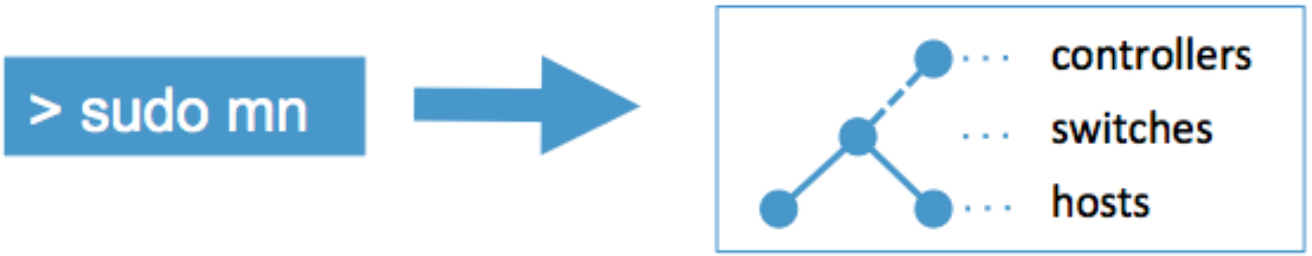
What is a Network Topology?

- Hosts
- Switches
- Links

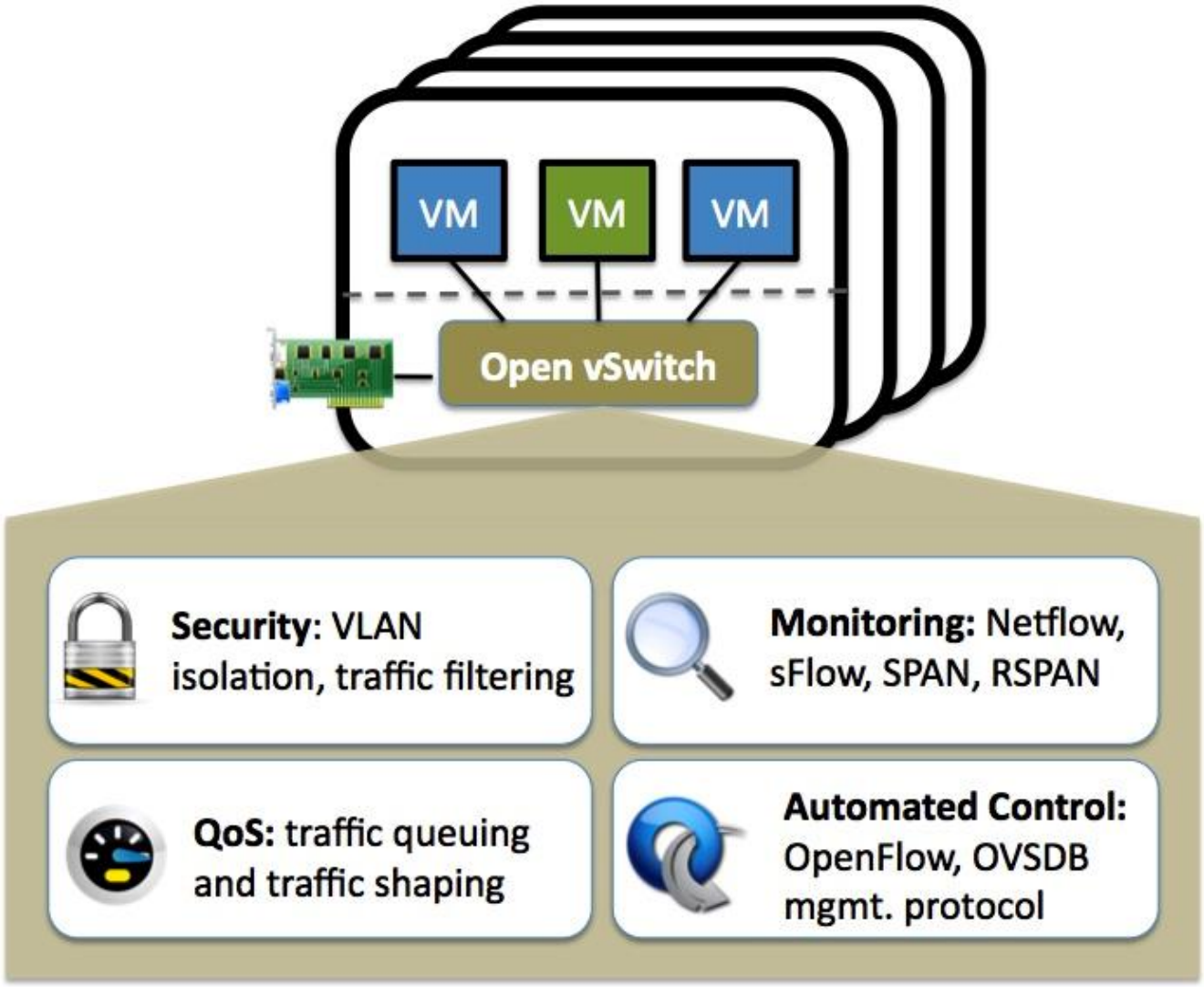


Mininet

- Mininet is a network emulator
  - Overview of Mininet - http://mininet.org/overview/
- 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.



Open vSwitch (OvS)



Why Mininet?

- Fast and easily
- Create custom topologies
- Run real programs
- Customize packet forwarding
- Support OpenFlow and software-defined network (SDN)

Mininet CLI (Command-Line Interface)

- Start a minimal topology and enter the CLI

```
$ sudo mn
mininet> help
```

- Show the information of every nodes

```
mininet> nodes
```

- Show every links of all nodes

```
mininet> links
```

- Show the network topology

```
mininet> net
```

- Show all ports on every switches

```
mininet> ports
```

Mininet CLI (Command-Line Interface)

- Show all network interfaces

```
mininet> intfs
```

- Dump information about all nodes

```
mininet> dump
```

- Test the connectivity of all hosts

```
mininet> pingall
```

- Test TCP connection of two hosts with iPerf

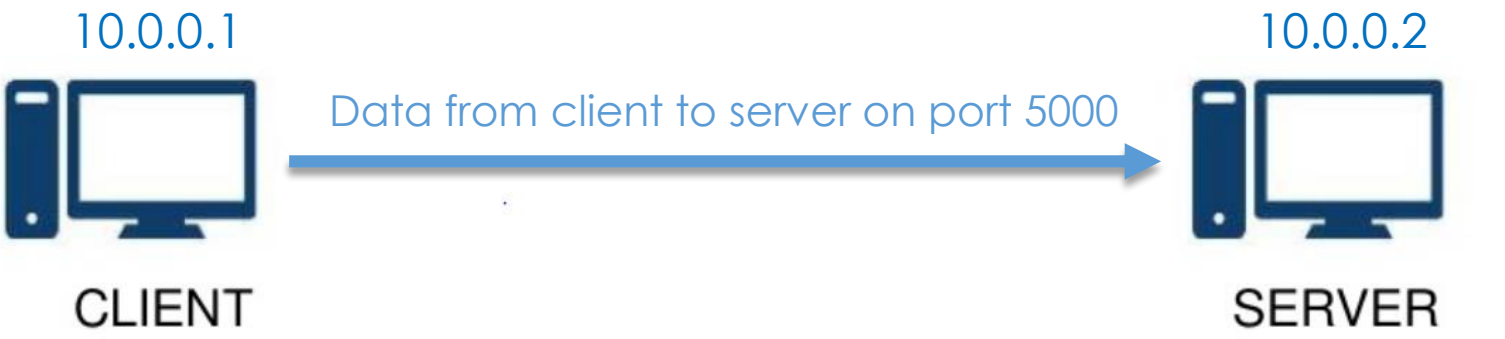
```
mininet> iperf
```

- Leave the Mininet's CLI mode

```
mininet> exit
```

iPerf

- iPerf is a tool for active measurements of the maximum achievable bandwidth on IP networks
- Support tuning of various parameters related to timing, buffers and protocols (TCP, UDP, SCTP with IPv4 and IPv6)



Lab2 Tasks

File Structure

```
Lab2_Network_Topology/          # This is ./ in this repository
|--- src/                       # Folder of source code
     |--- topo/                 # The figure of topology
          |--- topo0.png
          |--- topo1.png
          |--- topo2.png
     |--- expect/               # Expected result using iPerf
          |--- topo0
          |--- topo1
          |--- topo2
     |--- out/                  # Output files
          |--- .gitkeep         # For keeping this folder
     |--- example.py            # Example code of using Mininet
     |--- MyTopo.py             # The program you should modify
|--- Report.pdf                 # Your report of Lab2
|--- .gitignore                 # For ignoring useless files
```

TODO

1. We will give you a Python code (example.py) that includes an example network topology of Mininet
2. We will get you a figure illustrating a new topology you should generate
3. Refer to code from example.py and write your Python code (MyTopo.py) to generate this topology
4. Use iperf to measure the performance of this topology

Tasks

1. Environment Setup
2. Example of Mininet
3. Topology Generator [modify `MyTopo.py`]
4. Measurement
5. 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/E9d6YtLR
  - 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
    - 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 3. Install Mininet**
  - Install Mininet **in your container first** (important)

```
# Type this command in container
$ /mininet/util/install.sh -a
```

  - The installation process takes about 15 minutes
  - When installation finished, you would see these messages in the end

```
...
make[1]: Entering directory '/root/oflops/doc'
make[1]: Nothing to be done for 'install'.
make[1]: Leaving directory '/root/oflops/doc'
Enjoy Mininet!
```

Task 1. Environment Setup (cont.)

- **Step 4. Get GitHub repository (in container)**
  - Download required files from GitHub

```
$ git clone
https://github.com/chenyang14/Lab2_Network_Topology.git
```

  - Get and set repository for global options

```
$ cd Lab2_Network_Topology/
$ git config --global user.name "<NAME>"
$ git config --global user.email "<EMAIL>"
```

  - Set a new remote URL to your repository

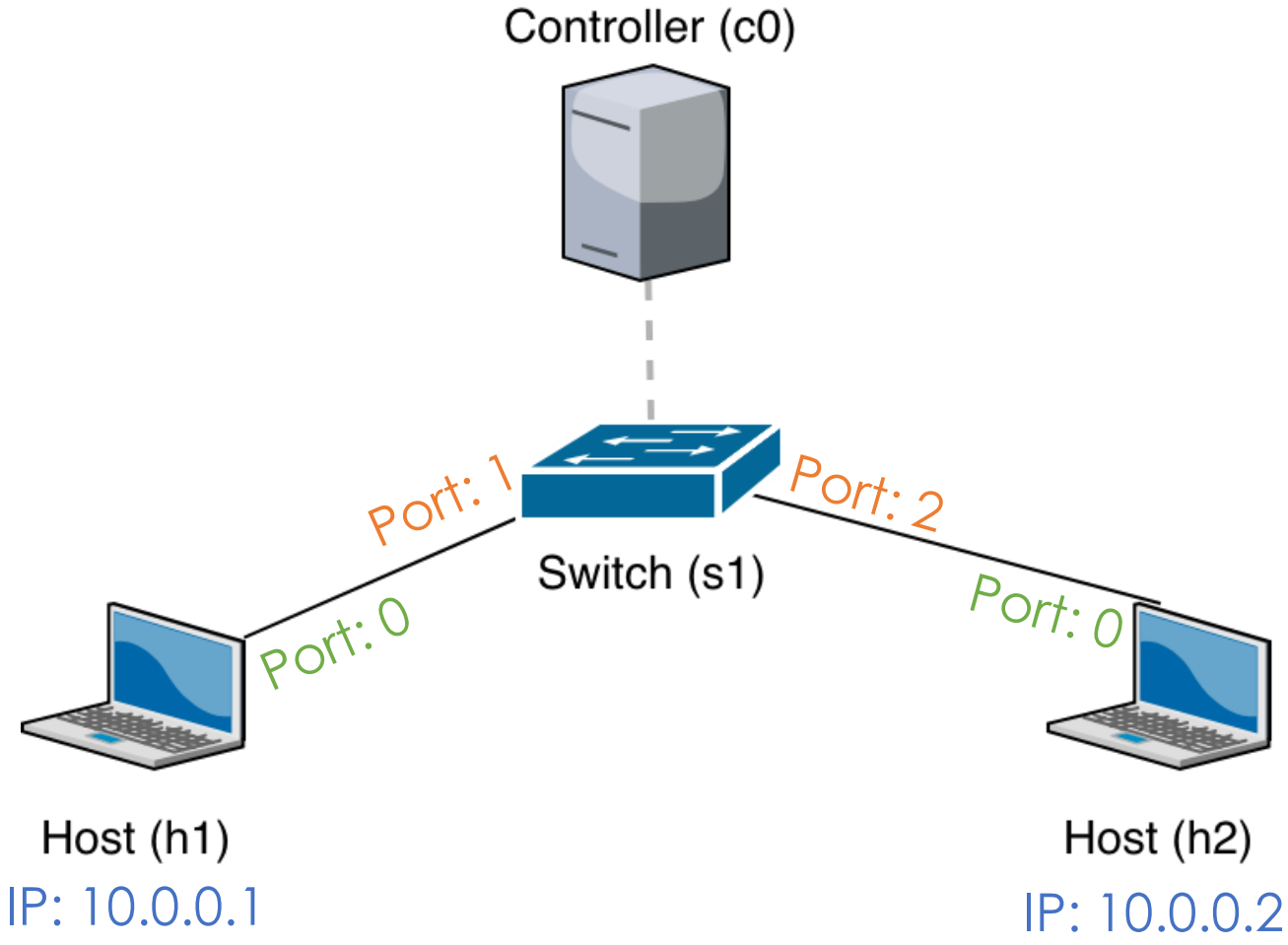
```
$ git remote set-url origin
https://github.com/nctucn/lab2-<GITHUB_ID>.git
```

  - Push your repository to GitHub

```
$ git push origin master
```

Task 2. Example of Mininet

- Network topology of example.py



Task 2. Example of Mininet (cont.)

- example.py – create topology and use **pingall** check connection between hosts
  - Topology: 1 switch with 2 hosts

```
class SingleSwitchTopo(Topo):
    def build(self, n = 2):
        # Add a switch to a topology
        switch = self.addSwitch('s1')
        # Add the host and link to a topology
        for h in range(n):
            # Add a host to a topology
            host = self.addHost('h%s' % (h + 1))
            # Add a bidirectional link to a topology
            self.addLink(host, switch, bw=10, delay='5ms')
```

Task 2. Example of Mininet (cont.)

- Run the example code we provided

```
# Do these in your container
# Change the directory into /Lab2_Network_Topology/src/
$ cd ~/Lab2_Network_Topology/src/
# Run the example code (example.py)
$ python example.py
```

- Just ignore the following message if you see it after executing example.py

```
*** Error setting resource limits. Mininet's performance
may be affected.
```

Task 2. Example of Mininet (cont.)

- The result after running example code

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit 5ms delay 0.00000%
loss) (10.00Mbit 5ms delay
0.00000% loss) (h1, s1)
(10.00Mbit 5ms delay 0.00000%
loss) (10.00Mbit 5ms delay
0.00000% loss) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
C0
```

```
*** Starting 1 switches
s1 ... (10.00Mbit 5ms delay 0.00000%
loss) (10.00Mbit 5ms delay 0.00000%
loss)
Testing network connectivity
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2
received)
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
```

Task 2. Example of Mininet (cont.)

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

```
# Change directory into ~/Lab2_Network_Topology/src/
# Run the example code (example.py)
$ python example.py
*** 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 3. Topology Generator

- **Step 1. Find the topology you should generate**
  - Please divide the last digit of your student ID by 3 to get the remainder
  - Find the figure you should generate in folder /Lab2_Network_Topology/src/topo/
  - For example, student ID "0716001" should implement topo1.png

Remainder	Topology figure
0	topo0.png
1	topo1.png
2	topo2.png

Task 3. Topology Generator (cont.)

- **Step 2. Generate the topology via Mininet**
  - **[TODO]** Modify the python program MyTopo.py to generate a network topology for Mininet
    - Create hosts and switches
    - Construct links
    - Configure link bandwidth, delay, and loss rate
  - You can refer to the example.py and make sure you really understand each line of code

Task 3. Topology Generator (cont.)

- **Other requirements**
  - **[TODO]** Dump every hosts' connections in your program

```
# Remember to import the following module first!
from mininet.util import dumpNodeConnections
# Dump every hosts' and switches' connections
dumpNodeConnections(net.hosts)
dumpNodeConnections(net.switches)
```

  - **[TODO]** Enter in the Mininet's CLI mode in your program

```
# Remember to import the following module first!
from mininet.cli import CLI
# Start CLI mode while executing
CLI(net)
```

Task 3. Topology Generator (cont.)

- **Troubleshooting 2**
  - 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

```
# Run the example code (example.py)
$ python example.py
*** Creating network
......
Exception: Error creating interface pair (s1-eth1,s2-
eth1): RTNETLINK answers: File exists
```

Task 4. Measurement

- **In Mininet CLI mode**, using pingall to check the link connections of the topology you built
  - **[TODO]** Screenshot the result of pingall

```
mininet> pingall
```

  - The expected result (should be 0% dropped)

```
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
...
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
```

  - The number of hosts may be different, it depends on which topology you built

Task 4. Measurement

- **In Mininet CLI mode**, using the following iperf commands to measure the topology in **TCP**
  - **[TODO]** Screenshot the result of iPerf command
  - For topo0.png

```
mininet> h8 iperf -s -i 1 > ./out/result.txt &
mininet> h1 iperf -c 10.0.0.8 –i 1
```

  - For topo1.png

```
mininet> h8 iperf -s -i 1 > ./out/result.txt &
mininet> h4 iperf -c 10.0.0.8 –i 1
```

  - For topo2.png

```
mininet> h4 iperf -s -i 1 > ./out/result.txt &
mininet> h1 iperf -c 10.0.0.4 –i 1
```

  - The above commands will dump the result of iPerf's measurement into the file ./out/result.txt

Task 4. Measurement (cont.)

- The expected result from the topo0.png

```
mininet> h8 iperf -s -i 1 > ./out/result.txt &
mininet> h1 iperf -c 10.0.0.8 –i 1
------------------------------------------------------------
Client connecting to 10.0.0.8, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.1 port 50528 connected with 10.0.0.8 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  3.00 MBytes  25.2 Mbits/sec
[  3]  1.0- 2.0 sec  3.12 MBytes  26.2 Mbits/sec
......
[  3]  9.0-10.0 sec  2.25 MBytes  18.9 Mbits/sec
[  3]  0.0-10.2 sec  27.8 MBytes  22.7 Mbits/sec
```

You will get the results that close to these numbers.
And remember to take the screenshot

Task 4. Measurement (cont.)

- The expected result from the topo1.png

```
mininet> h8 iperf -s -i 1 > ./out/result.txt &
mininet> h4 iperf -c 10.0.0.8 –i 1
------------------------------------------------------------
Client connecting to 10.0.0.8, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.4 port 51906 connected with 10.0.0.8 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  2.50 MBytes  21.0 Mbits/sec
[  3]  1.0- 2.0 sec  2.25 MBytes  18.9 Mbits/sec
......
[  3]  9.0-10.0 sec  1.88 MBytes  15.7 Mbits/sec
[  3]  0.0-10.4 sec  22.6 MBytes  18.2 Mbits/sec
```

You will get the results that close to these numbers.
And remember to take the screenshot

Task 4. Measurement (cont.)

- The expected result from the topo2.png

```
mininet> h4 iperf -s -i 1 > ./out/result.txt &
mininet> h1 iperf -c 10.0.0.4 –i 1
------------------------------------------------------------
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.1 port 41028 connected with 10.0.0.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  1.88 MBytes  15.7 Mbits/sec
[  3]  1.0- 2.0 sec  2.00 MBytes  16.8 Mbits/sec
......
[  3]  9.0-10.0 sec  1.50 MBytes  12.6 Mbits/sec
[  3]  0.0-10.7 sec  17.5 MBytes  13.7 Mbits/sec
```

You will get the results that close to these numbers.
And remember to take the screenshot

Task 5. Report

- **Your Report.pdf must include**
  - **Measurement results**
    - Screenshot the result of **pingall** command
    - Screenshot the result of **iperf** command
  - **Description**
    - Describe how you finish this work in detail
    - Describe the meaning of iPerf command you used
      - Explain what these argument (-c, -s , -i) mean?
      - If we want to perform iperf UDP test instead of TCP test, which argument should be added?

Submission

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

```
# In container folder: Lab2_Network_Topology/
# 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/)
    - Result.txt
  - **Python code** (./src/)
    - MyTopo.py
  - **Report** (./)
    - Report.pdf

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

Grading Policy

- Deadline - Dec, 01 2019. 23:59
- Python program – 70 %
- Report – 30 %

Grading Policy (cont.)

- **Late Policy** (follow syllabus)

$$(\textbf{Your score}) \times \mathbf{0.8}^{\boldsymbol{D}},$$

where D is the number of days over due

- **Cheating Policy** (follow syllabus)
  - Academic integrity
  - Homework must be your own – cheaters share the score
  - Both the cheaters and the students who aided the cheater will be held responsible for the cheating

References

- **Mininet**
  - English
    - Mininet Walkthrough
    - Introduction to Mininet
    - Mininet Python API Reference Manual
    - A Beginner's Guide to Mininet
  - Chinese
    - GitHub/OSE-Lab - 熟悉如何使用 Mininet
    - 菸酒生的記事本 – Mininet 筆記
    - Hwchiu Learning Note – 手把手打造仿 mininet 網路
    - 阿寬的實驗室 – Mininet 指令介紹
    - Mininet 學習指南

References (cont.)

- **Others**
  - iPerf User Documentation
  - Vim Tutorial – Tutorialspoint
  - 鳥哥的 Linux 私房菜 – 第九章、vim 程式編輯器