Computer Networks
@CS.NCTU

Lab. 2: route configuration in Mininet and PoX controller

Outline

- **Mininet and PoX Controller**
- Preparation
- Assignment
- Reference

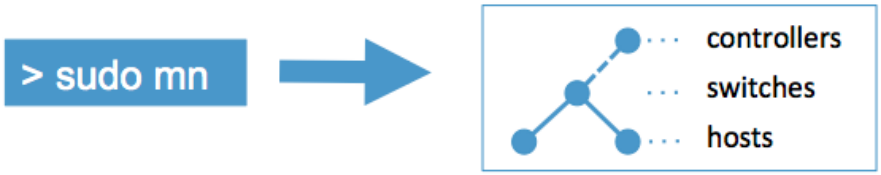
Mininet

- http://mininet.org/
- Overview: http://mininet.org/overview/
- Download and Install: http://mininet.org/download/

- Network emulator which creates a network of virtual hosts, switches, controllers, and links
- Run on Linux
- Configuration script written in Python

Mininet

- Mininet creates a **realistic virtual network**, running **real kernel, switch and application code**, on a single machine (VM, cloud or native), in seconds, with a single command



- Mininet is also a great way to develop, share, and experiment with **OpenFlow** and **Software-Defined Networking** systems

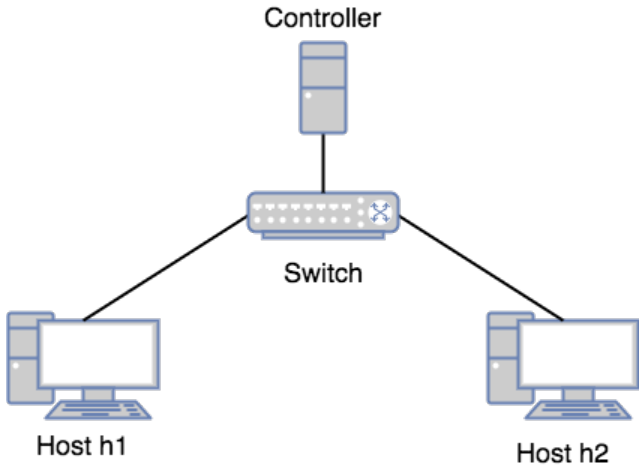
Mininet Notes

- Introduction to Mininet: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet
- Python API reference manual: http://mininet.org/api/annotated.html
- Chinese notes
  - http://it-life.wyx-ccy.com/2016/04/mininet-for-ubuntu-linux-1510.html
  - https://wiki.kshuang.xyz/doku.php/ccis_lab:sdn:mininet:mininet_install
  - http://hwchiu.com/setup-mininet-like-environment.html
  - https://seannets.wordpress.com/2016/04/19/%E5%AD%B8%E7%BF%92sdn-%E6%BA%96%E5%82%99%E5%B7%A5%E4%BD%9C-%E5%AE%89%E8%A3%9Dmininet-and-ryu/
  - https://laszlo.tw/?p=81
  - http://www.cs.nchu.edu.tw/~snmlab/CloudMgnt201409/Lab3.html

Mininet Introduction

- Interact with Hosts and Switches
  - Start a minimal topology and enter the CLI

```
$ sudo mn
```



Mininet Introduction

- Interact with Hosts and Switches
  - Display Mininet CLI commands:

```
Mininet> help
```

  - Display nodes:

```
Mininet> nodes
```

  - Display links:

```
Mininet> net
```

  - Dump information about all nodes:

```
Mininet> dump
```

Outline

- Mininet and PoX Controller
- **Preparation**
- Assignment
- Reference

Environment Setup

- Using Your Own Ubuntu Machine or VM
  - Install Needed Tools

```
$ sudo apt-get update
$ sudo apt-get install -y git vim-nox python-
setuptools python-all-dev flex bison traceroute
```

  - Install **Mininet**

```
$ cd ~
$ git clone git://github.com/mininet/mininet
$ cd mininet
$ ./util/install.sh -fnv
```

Environment Setup (cont.)

- Using Your Own Ubuntu Machine or VM
  - Install **POX**

```
$ cd ~
$ git clone http://github.com/noxrepo/pox
```

    - Two important folders under **./pox**
      - **./pox/pox** – The core of the controller
      - **./pox/ext** – Place the custom controller program

  - Install **ltprotocol**

```
$ cd ~
$ git clone
git://github.com/dound/ltprotocol.git
$ cd ltprotocol
$ sudo python setup.py install
```

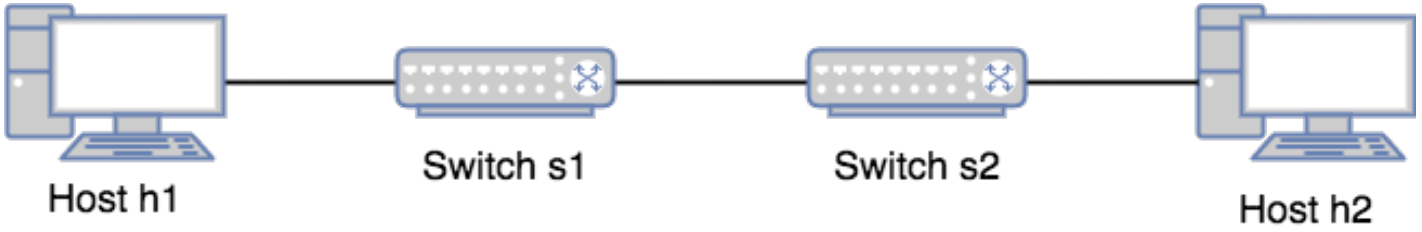
Environment Setup (cont.)

- Using Your Own Ubuntu Machine or VM
  - Install **iperf**

```
$ sudo apt-get update

# For Ubuntu 64-bit Debian 64 bits / Mint 64 bits (AMD 64)
$ wget https://iperf.fr/download/ubuntu/libiperf0_3.1.3-1_amd64.deb
$ wget https://iperf.fr/download/ubuntu/iperf3_3.1.3-1_amd64.deb
$ sudo dpkg -i libiperf0_3.1.3-1_amd64.deb iperf3_3.1.3-1_amd64.deb

# For Ubuntu 32 bits / Debian 32 bits / Mint 32 bits (i386) :
$ wget https://iperf.fr/download/ubuntu/libiperf0_3.1.3-1_i386.deb
$ wget https://iperf.fr/download/ubuntu/iperf3_3.1.3-1_i386.deb
$ sudo dpkg -i libiperf0_3.1.3-1_i386.deb iperf3_3.1.3-1_i386.deb

# Remove iPerf
$ sudo apt-get remove iperf3 libiperf0
```

Sample Code

- sample.py, sample_controller.py
- Simple topology
  - Two hosts – **h1**, **h2**
  - Two switches – **s1**, **s2**



- h1 sends traffic to h2 along the path (h1, S1, S2, h2)
- Use ***iperf*** to generate UDP traffic
  - Server at h2: $iperf -s -u -i 1
  - Client at h1: $ iperf -c *h2-IP* -u -i 1

Sample Code (cont.)

- You can run the sample code on Mininet
  - Move the `sample_controller.py` into the folder `./pox/ext`
  - Run the `sample_controller.py` on POX and don't close this terminal

```
$ cd ./pox
$ ./pox.py sample_controller
```

  - Open another terminal and run the `sample.py` on Mininet

```
# Change the directory into the program sample.py
# Change to the executable mode of sample.py (Optional)
$ sudo chmod +x sample.py
$ sudo ./sample.py
```

Sample Code (cont.)

- After running the `sample.py`, it will print some messages on the terminal

```
*** Creating network
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2
*** Adding links:
(1.00Mbit 10ms delay 0.00000% loss) (1.00Mbit 10ms delay
0.00000% loss) (h1, s1) (1.00Mbit 10ms delay 0.00000% loss)
(1.00Mbit 10ms delay 0.00000% loss) (s1, s2) (1.00Mbit 10ms
delay 0.00000% loss) (1.00Mbit 10ms delay 0.00000% loss) (s2,
h2)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(1.00Mbit 10ms delay 0.00000% loss) (1.00Mbit 10ms
delay 0.00000% loss) (1.00Mbit 10ms delay 0.00000% loss)
(1.00Mbit 10ms delay 0.00000% loss)
*** Dumping host connections
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth2
```

```
*** Testing bandwidth between h1 and h2
------------------------------------------------------------
Client connecting to 10.0.0.1, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.2 port 36526 connected with 10.0.0.1 port
5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  11.9 MBytes  10.0 Mbits/sec
[  3] Sent 8505 datagrams
[  3] WARNING: did not receive ack of last datagram after 10
tries.
*** Output the iperf results
1: ------------------------------------------------------------
2: Server listening on UDP port 5001
3: Receiving 1470 byte datagrams
4: UDP buffer size:  208 KByte (default)


......
```

Sample Code (cont.)

- Moreover, you can use Mininet and ping the host **h2** on host **h1**
  - Make the **line 40 – 50** into comments in the `sample.py`
  - Types `CLI(net)` after the above comments

```
40  """print "*** Testing bandwidth between h1 and h2"
41  h1.cmd('iperf -s -u -i 1')
42  print h2.cmd('iperf -c 10.0.0.1 -u -b 10m -t 10')
43  h1.cmd('kill %iperf')
44
45  print "*** Output the iperf results"
46  file = open('results')
47  line_num = 1
48  for line in file.readlines():
49      print "%d: %s" %(line_num, line.strip())
50      line_num += 1"""
51
52  CLI(net)
```

Sample Code (cont.)

- Follow the steps mentioned (slide p.8) to run the program

```
*** Creating network
*** Adding hosts:h1 h2
*** Adding switches:s1 s2
*** Adding links:
(1.00Mbit 10ms delay 0.00000% loss) (1.00Mbit 10ms delay 0.00000% loss) (h1, s1) (1.00Mbit 10ms
delay 0.00000% loss) (1.00Mbit 10ms delay 0.00000% loss) (s1, s2) (1.00Mbit 10ms delay 0.00000%
loss) (1.00Mbit 10ms delay 0.00000% loss) (s2, h2)
*** Configuring hosts
h1 h2
Unable to contact the remote controller at 127.0.0.1:6633
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(1.00Mbit 10ms delay 0.00000% loss) (1.00Mbit 10ms delay 0.00000% loss) (1.00Mbit 10ms
delay 0.00000% loss) (1.00Mbit 10ms delay 0.00000% loss)
Dumping host connections
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth2
*** Starting CLI:
mininet>
```

Sample Code (cont.)

- Ping the host **h2** on **h1**

```
# Ping with setting up the number of ICMP echo requests to send
mininet> h1 ping -c 10 h2
# Ping with setting up the TTL value
mininet> h1 ping -i 0.1 h2
```

- If success, it will return the following results

```
ING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=81.3 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=80.5 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=80.7 ms
...
```

CLI(net)

- Simple command-line interface to talk to nodes
  - Add “CLI(net)” in your python code
  - Leave the command-line mode by typing “exist”
  - The Mininet may end after you exist the commend-line mode

Error Handling

- The following error may occur when you runs the program `sample.py`

```
$ sudo ./sample.py
*** Creating network

......

Exception: Error creating interface pair (s1-eth2,s2-
eth1): RTNETLINK answers: File exists
```

  - Solution

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

Outline

- Mininet and PoX Controller
- Preparation
- **Assignment**
- Reference

What You Will Learn from the Lab?

1. Learn how to set up a network topology in MiniNet

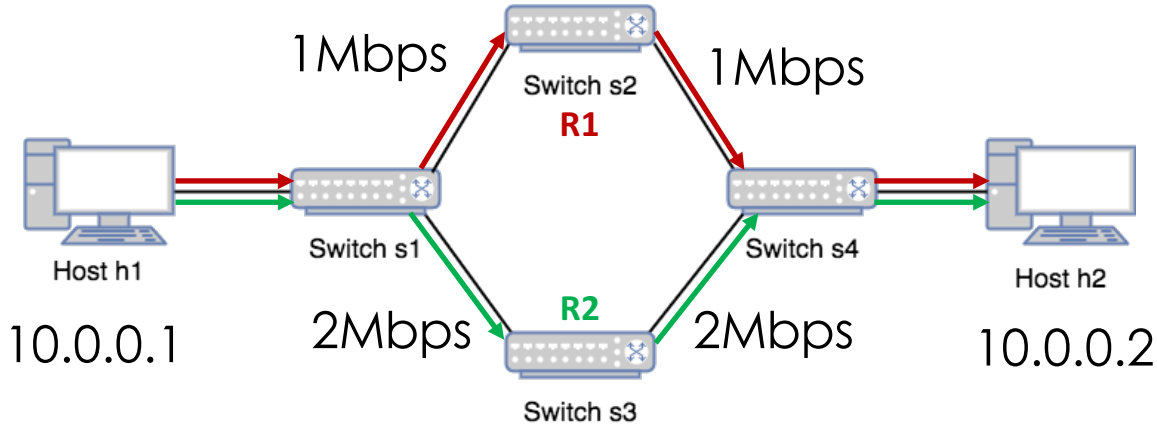
2. Learn how to set up the packet loss ratio of each link

3. Learn how to re-configure the routing table of each switch at a particular time

4. Learn how to estimate the achievable throughput of different routing configurations

TODO

- Build the following topology and set up the two forwarding rules **R1** and **R2**
  - **R1**: **`h1-s1-s2-s4-h2`**
  - **R2**: **`h1-s1-s3-s4-h2`**
- Set the packet loss rate of switch `s2` and `s3` to 50% and 10%, respectively
- Set the bandwidth of `s1-s2`, `s2-s4`, `s1-s3` and `s3-s4` to 1, 1, 2 and 2, respectively



TODO (cont.)

- After setting up the rules, you need to change between two paths for every 5 seconds
- Uses *iPerf* command to output the achievable throughput
  - **h2** – **Server**, listen UDP packets every 1 second and output results
    → $iperf -s -u -i 1
  - **h1** – **Client**, send UDP packets every 1 second
    → $ iperf -c *h1-IP* -u -i 1

Hints

- In your topology, your need to set the loss probability of link **`s1-s2`** and **`s1-s3`**

- Remember to modify **the event handler** when the connection is up!

Hints

- Be careful to connect each node **in correct port** to enable bidirectional forwarding
  - In _handle_PacketIn function, try to add the following code to switch **s1** and switch **s4**

```
a = packet.find('arp')
if a and a.protodst == "10.0.0.2":
    msg = of.ofp_packet_out(data = event.ofp)
    msg.actions.append(of.ofp_action_output(port = 2))
    event.connection.send(msg)
```

  - Use CLI(net) to switch to the commend-line mode. Then, use the following commend to check the port number

```
Mininet> net
s1-eth1 <> s2-eth2
```

POX: Set Up Timer

- Perform a task every few seconds (reference)
  - In your python code, `from pox.lib.recoco import Timer`
  - Timer class is designed to handle – executing a piece of code at a single or recurring time in the future
  - Recurring timer

```
# Simulate a long road trip
from pox.lib.recoco import Timer

we_are_there = False

def are_we_there_yet ():
  if we_are_there: return False # Cancels timer (see selfStoppable)
  print "Are we there yet?"

Timer(30, are_we_there_yet, recurring = True)
```

Outline

- Mininet and PoX Controller
- Preparation
- Assignment
- **Reference**

Output

- python files
  - ID_lab2.py
  - ID_lab2_controller.py
- Report (ID_lab2.pdf) including
  - A short summary explaining your implementation
  - A figure shows the throughput over time
- Submit to E3 by Dec. 28, 23:59
  - Delay policy: see syllabus

References

- Python documentation
  - Python 2.7.14 documentation
  - Python 3.6.3 documentation
- Mininet Walkthrough
- POX Controller
- POX Wiki
- iPerf user docs