

Lab 1

ONOS and Mininet Installation

Environment Setup & Basic Operation





Deadline 2024/09/25 (WED) 23:59

Email: sdnta@win.cs.nycu.edu.tw



Overview

- Lab1 homework provide 4 files in E3.

 2024-LAB1.pdf	Adobe Acrobat 文件
 env_setup.sh	SH 來源檔案
 sample.py	Python 來源檔案
 SDN_Environment_Setup.pdf	Adobe Acrobat 文件



Outline

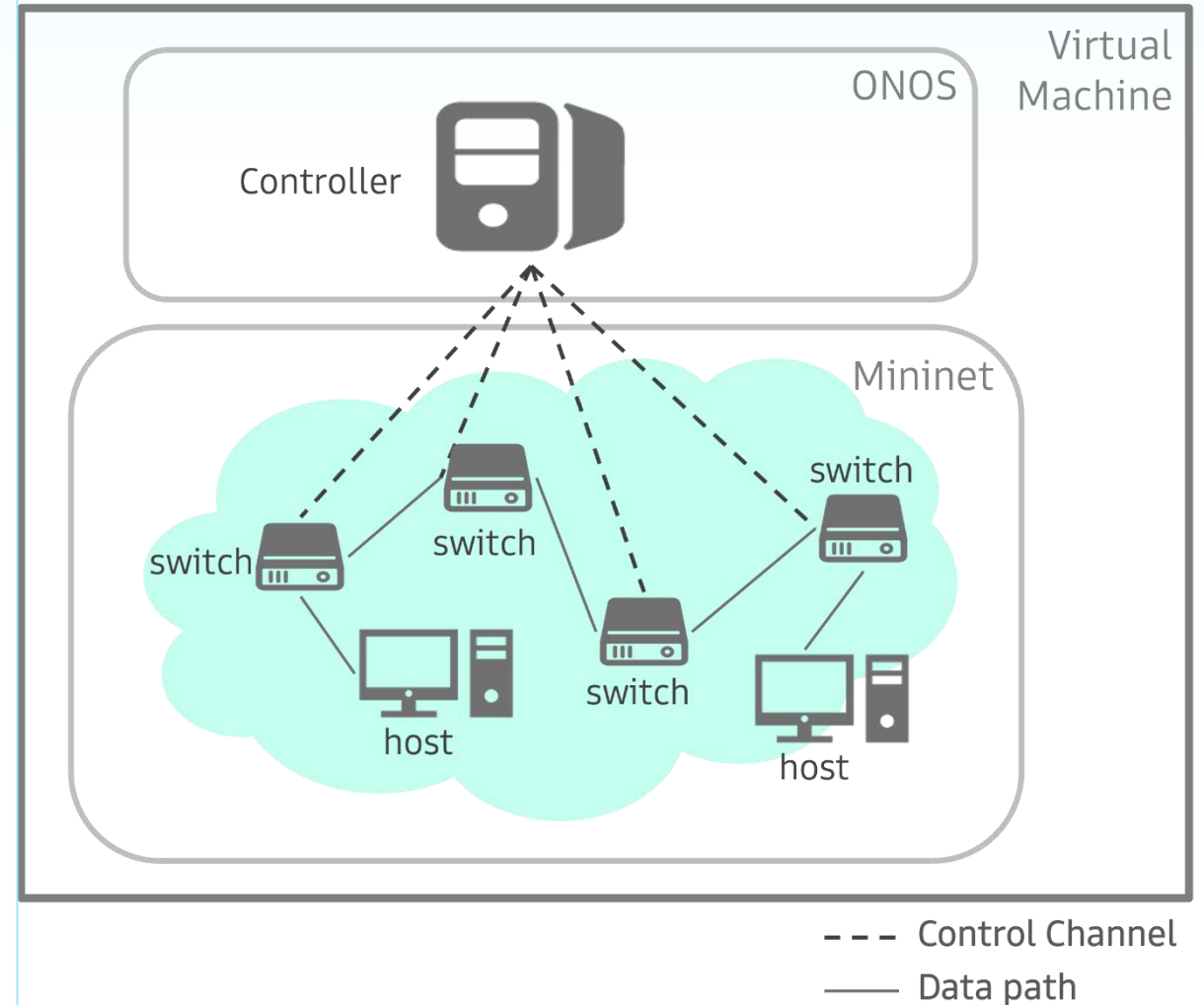
- Environment Introduce & Setup
 - Overview introduction
 - VirtualBox, Bazel, ONOS, Mininet and OVS Installation
- Building virtual network
 - Build ONOS
 - Activate control plane function
 - Create a topology controlled with Mininet
- Lab Requirements
 - Part 1: Answer Questions
 - Part 2: Write a Custom Topology
 - Part 3: Statically Assign Hosts IP Address In Mininet



Overview

To emulate an SDN network we need:

- controller for control
 - Mini-topology with switches
 - Switches with appropriate protocols
- Controller connect with switch through control channel
 - Packets go through data path from host to host





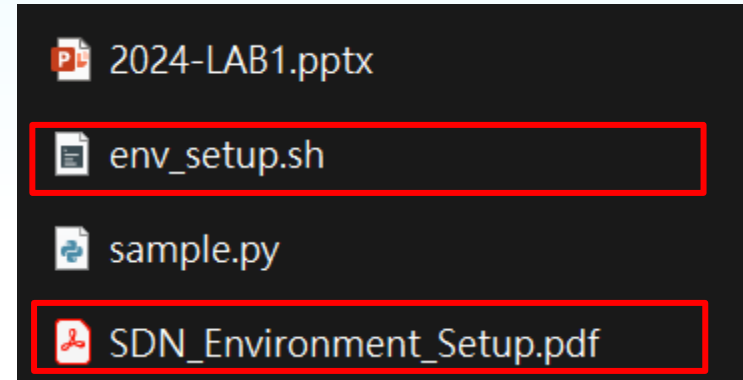
VirtualBox, Bazel, ONOS, Mininet and OVS Installation

- **Bazel:** Free and open-source software tool for “automation of building and testing software.”
- **Open Network Operating System (ONOS):** Open source SDN network controller.
- **Mininet:** a software emulator for prototyping a large network on a single machine.
- **Open vSwitch (OVS):**
 - an open-source implementation of a distributed virtual multilayer switch.
 - Provides a switching stack for hardware virtualization environments while supporting multiple protocols and standards used in computer networks



VirtualBox, Bazel, ONOS, Mininet and OVS Installation

- Installation:
 - Follow **SDN_Environment_Setup.pdf**
 - Use TA-provided **env_setup.sh**
- Mac M series chips, which are based on ARM CPUs, may not be able to successfully create the desired environment.
 - Therefore, we recommend using resources based on x86 CPUs in your labs or cloud platforms such as Azure, AWS, and GCP as a solution to this issue.





SDN development environment (4/4)

- If the installation complete, you'll see finish message

```
Make[1]: Leaving directory '/home/demo/ovs'
Start registering Open vSwitch service...
Created symlink /etc/systemd/system/multi-user.target.wants/ovs.service → /etc/s
ystemd/system/ovs.service.
Finished the installation of Open vSwitch.
*****
* The environment setup finished successfully! *
*****
demo@SDN-NFV:~/Desktop$
```

- Run the command

```
demo@SDN-NFV:~$ source ~/.bashrc
```

- After this, you have finished your environment setup



Outline

- Environment Introduce & Setup
- Building Virtual network
 - Build ONOS
 - ONOS CLI
 - ONOS GUI
 - Activate Control plane function
 - Method1 : Via ONOS CLI
 - Method2 : Via ONOS GUI
 - Create a topology controlled with Mininet
 - Method 1: Built-in Topology
 - Method 2: Custom Topology
- Lab Requirements



Build ONOS

Start ONOS in localhost

```
demo@SDN-NFV:~$ cd $ONOS_ROOT
demo@SDN-NFV:~/onos$ bazel run onos-local -- clean debug
# option 'clean' to delete all previous running status
# option 'debug' to enable remote debugging (port 5005)
```

```
demo@SDN-NFV:~/onos$ bazel run onos-local -- clean debug
INFO: Analyzed target //:onos-local (0 packages loaded, 0 targets configured).
INFO: Found 1 target...
Target //:onos-local_current-jdk up-to-date:
  bazel-bin/onos-runner_current-jdk
INFO: Elapsed time: 0.486s, Critical Path: 0.00s
INFO: 0 processes.
INFO: Build completed successfully, 1 total action
INFO: Build completed successfully, 1 total action
Killing ONOS server...
Using JDK in /tmp/onos-2.2.0-jdk...
```

```
ConfigurationEvent: pid=org.onosproject.net.intent.impl.IntentCleanup) | OpenFlowRuleProvider | 203 - org.onosproject.onos-p
ConfigurationEvent: pid=org.onosproject.net.intent.impl.IntentCleanup) | OpenFlowRuleProvider | 203 - org.onosproject.onos-p
se
AtomixClusterStore | 192 - org.onosproject.onos-core-primitives - 2.2.0 | Updated node 127.0.0.1 state to READY
```



ONOS CLI

- Bring up another new terminal and enter ONOS CLI

```
demo@SDN-NFV:~/onos$ tools/test/bin/onos localhost
```

```
demo@SDN-NFV:~/onos$ tools/test/bin/onos localhost
Welcome to Open Network Operating System (ONOS)!

  ONOS

Documentation: wiki.onosproject.org
Tutorials:    tutorials.onosproject.org
Mailing lists: lists.onosproject.org

Come help out! Find out how at: contribute.onosproject.org

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'logout' to exit ONOS session.

demo@root > |
```

- Reference: [ONOS CLI command](#)

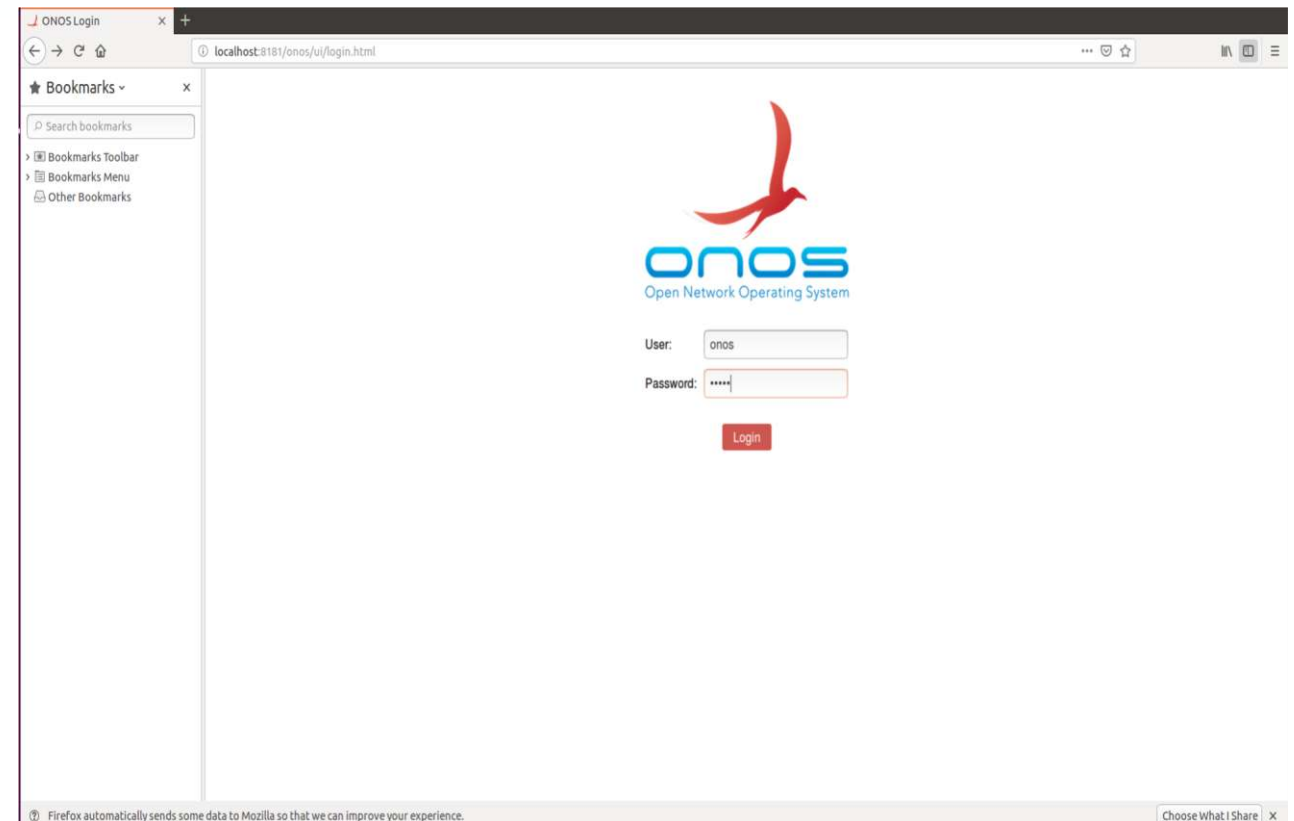


ONOS Web GUI

- Open web browser (e.g. Firefox)
visit <http://localhost:8181/onos/ui>

User/Password: onos/rocks

- Reference: [ONOS GUI tutorial](#)





Outline

- Environment Introduce & Setup
- Building Virtual network
 - Build ONOS
 - ONOS CLI
 - ONOS GUI
 - Activate Control plane function
 - Method1 : Via ONOS CLI
 - Method2 : Via ONOS GUI
 - Create a topology controlled with Mininet
 - Method 1: Built-in Topology
 - Method 2: Custom Topology
- Lab Requirements



Cli show APPs

- You can check all applications that is installed on the ONOSs

```
onos> apps -s  
# Show all apps in short
```

```
demo@root > apps -s 23:14:08  
3 org.onosproject.portloadbalancer 2.7.0 Port Load Balance Service  
4 org.onosproject.mcast 2.7.0 Multicast traffic control  
5 org.onosproject.tunnel 2.7.0 Tunnel Subsystem  
* 6 org.onosproject.optical-model 2.7.0 Optical Network Model  
* 7 org.onosproject.openflow-base 2.7.0 OpenFlow Base Provider  
* 8 org.onosproject.lldpprovider 2.7.0 LLDP Link Provider  
* 9 org.onosproject.hostprovider 2.7.0 Host Location Provider  
10 org.onosproject.route-service 2.7.0 Route Service Server  
11 org.onosproject.simplefabric 2.7.0 SONA SimpleFabric  
12 org.onosproject.ovsdb-base 2.7.0 OVSDb Provider  
13 org.onosproject.drivers.ovsdb 2.7.0 Generic OVSDb Drivers  
14 org.onosproject.k8s-node 2.7.0 Kubernetes Node Application  
15 org.onosproject.k8s-networking 2.7.0 Kubernetes Networking Application  
16 org.onosproject.linkdiscovery 2.7.0 Link Discovery Provider  
17 org.onosproject.faultmanagement 2.7.0 Fault Management  
18 org.onosproject.netconf 2.7.0 NETCONF Provider  
19 org.onosproject.drivers.netconf 2.7.0 Generic NETCONF Drivers  
20 org.onosproject.drivers.ciena.c5162 2.7.0 Ciena 5162 Drivers  
21 org.onosproject.gui 2.7.0 ONOS Legacy GUI  
22 org.onosproject.messaging-perf 2.7.0 Messaging Performance Test  
23 org.onosproject.events 2.7.0 Event History  
24 org.onosproject.influxdbmetrics 2.7.0 InfluxDB Report and Query  
25 org.onosproject.protocols.grpc 2.7.0 gRPC Protocol Subsystem  
26 org.onosproject.protocols.gnmi 2.7.0 gNMI Protocol Subsystem  
27 org.onosproject.generaldeviceprovider 2.7.0 General Device Provider  
28 org.onosproject.protocols.gnoi 2.7.0 gNOI Protocol Subsystem  
29 org.onosproject.drivers.gnoi 2.7.0 gNOI Drivers  
30 org.onosproject.yang 2.7.0 YANG Compiler and Runtime  
* 31 org.onosproject.drivers 2.7.0 Default Drivers  
32 org.onosproject.drivers.optical 2.7.0 Basic Optical Drivers  
33 org.onosproject.models.common 2.7.0 Common YANG Models  
34 org.onosproject.models.ciena.waveserverai 2.7.0 Ciena Waveserver AI YANG Models  
35 org.onosproject.drivers.ciena.waveserverai 2.7.0 Ciena Waveserver AI Drivers  
36 org.onosproject.network-troubleshoot 2.7.0 Network Troubleshooter  
37 org.onosproject.dhcp 2.7.0 DHCP Server  
* 38 org.onosproject.openflow 2.7.0 OpenFlow Provider Suite  
39 org.onosproject.ovsdbhostprovider 2.7.0 OVSDb host Provider  
40 org.onosproject.ovsdb 2.7.0 OVSDb Southbound Meta  
41 org.onosproject.workflow 2.7.0 Workflow  
42 org.onosproject.workflow.ofoverlay 2.7.0 Openflow overlay  
43 org.onosproject.openstacknode 2.7.0 OpenStack Node Bootstrap  
44 org.onosproject.openstacknetworking 2.7.0 OpenStack Networking Application
```




Activate basic ONOS APPS via CLI

```
onos> apps -a -s # Show activated apps only
```

```
demo@root > apps -a -s 02:39:00
* 6 org.onosproject.optical-model 2.7.0 Optical Network Model
* 7 org.onosproject.openflow-base 2.7.0 OpenFlow Base Provider
* 8 org.onosproject.lldpprovider 2.7.0 LLDP Link Provider
* 9 org.onosproject.hostprovider 2.7.0 Host Location Provider
* 31 org.onosproject.drivers 2.7.0 Default Drivers
* 38 org.onosproject.openflow 2.7.0 OpenFlow Provider Suite
* 168 org.onosproject.gui2 2.7.0 ONOS GUI2
```

```
onos> app activate <name> # activate onos app
```

```
onos> app deactivate <name> # deactivate onos app
```



```
demo@root > app activate org.onosproject.openflow 02:39:21
Activated org.onosproject.openflow
demo@root > app activate org.onosproject.fwd 02:40:10
Activated org.onosproject.fwd
```

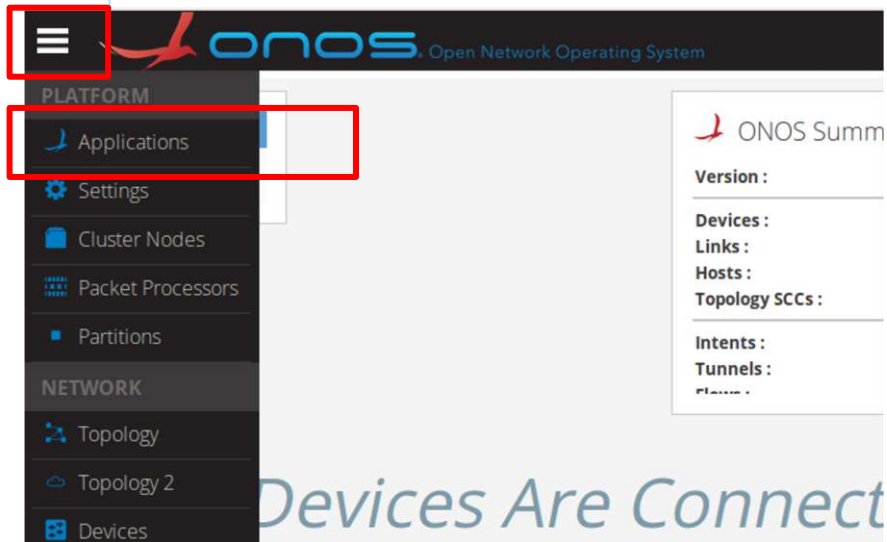
```
onos> app --help # display command help message
```



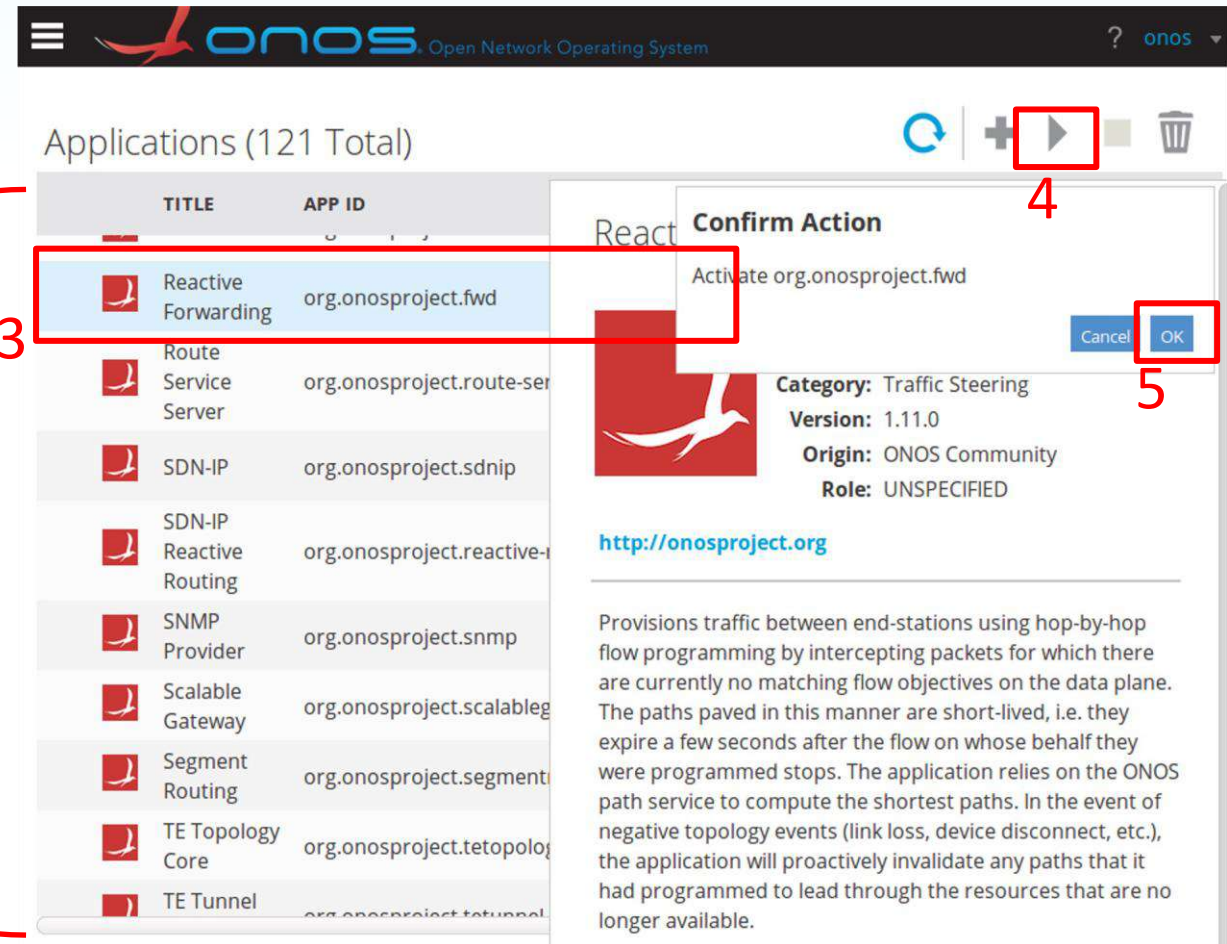
Activate basic ONOS APPS via GUI

• Via ONOS GUI

1. Click 
2. Choose “Applications”
3. Choose “Reactive Forwarding” from APPs list
4. Click 
5. Click “OK”



APPs
list





Outline

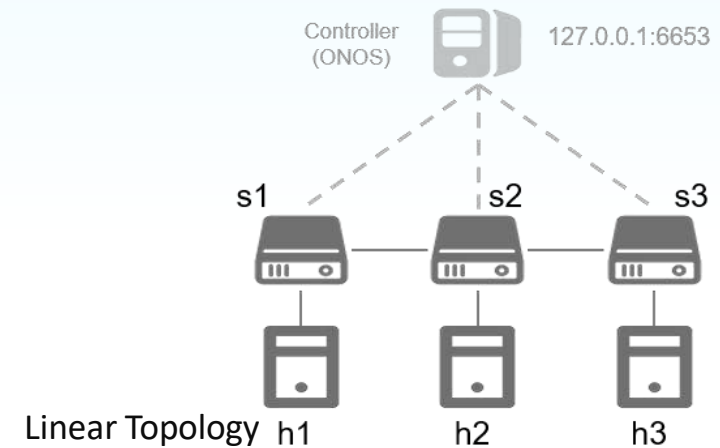
- Environment Introduce & Setup
- Building Virtual network
 - Build ONOS
 - ONOS CLI
 - ONOS GUI
 - Activate Control plane function
 - Method1 : Via ONOS CLI
 - Method2 : Via ONOS GUI
 - Create a topology controlled with Mininet
 - Method 1: Built-in Topology
 - Method 2: Custom Topology
- Lab Requirements



Build-in Topology in Mininet

- Five Built-in topologies:

- Minimal
Also called "Default"
- Single
- Linear
- Torus
- Tree



```
$ sudo mn --topo=linear,3 --controller=remote,127.0.0.1:6653 \
--switch=ovs,protocols=OpenFlow14
```

- Command for Mininet : mn [Options]

- switch: chose switch interface
- controller: add the controller
- topo: specifies the topology
- custom: read custom classes parameter from .py file

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> █
```



Clear your Experiment Environment

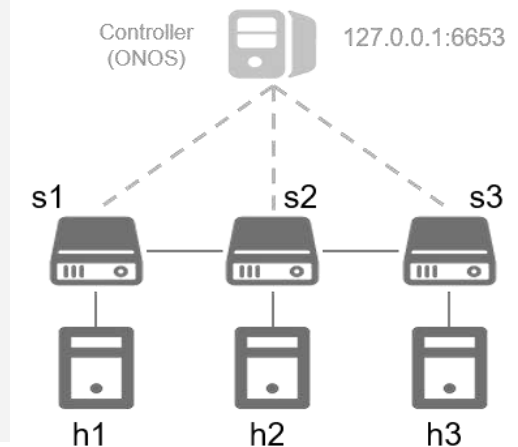
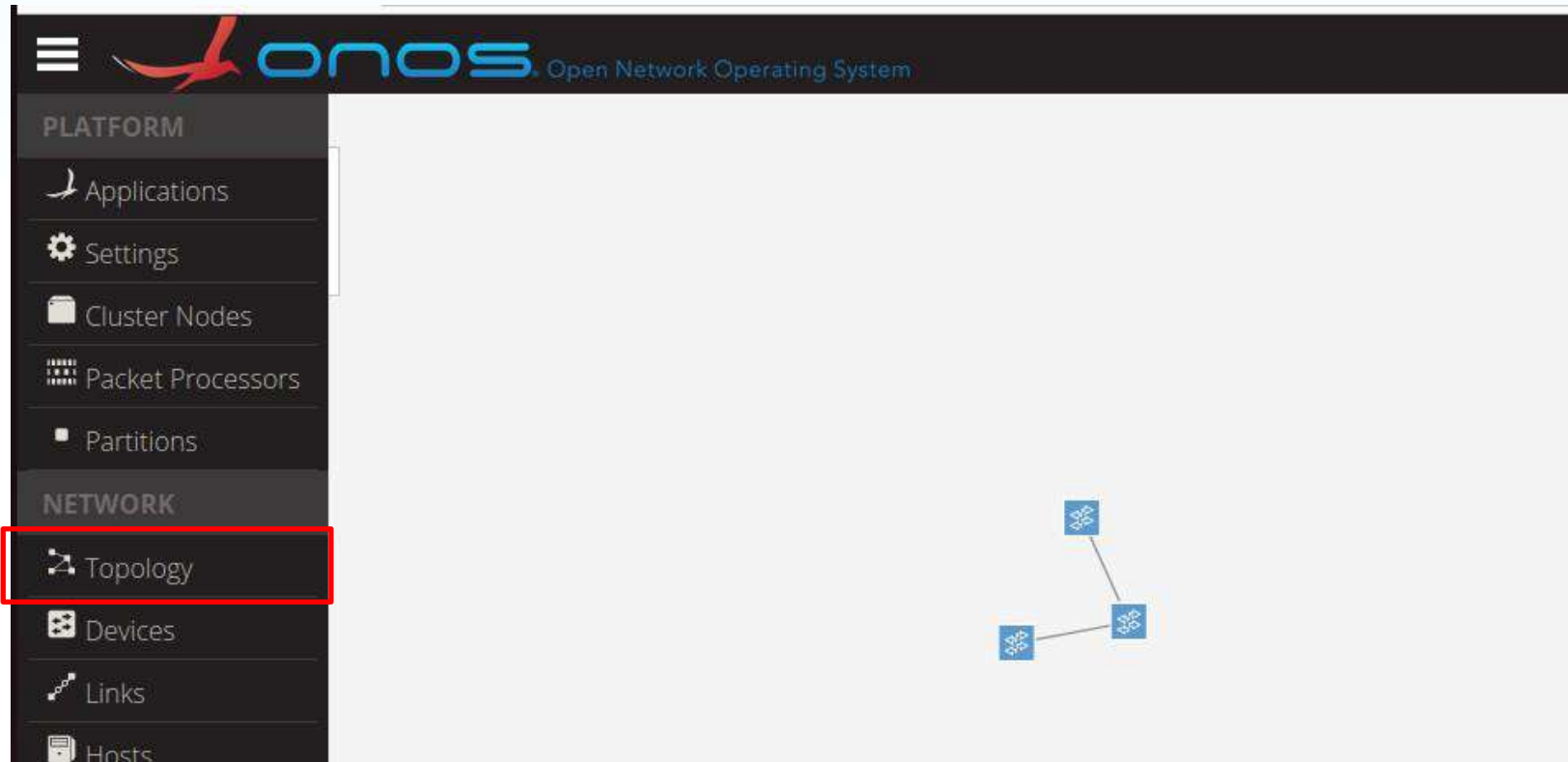
- Note: Make sure to clean up the environment of Mininet after every time you exit Mininet CLI

```
mininet> exit
$ sudo mn -c          # exit mininet cli
                      # clean and exit
                      # A "cleanup" command to get rid of junk (interfaces,
                      # processes, files in /tmp,etc.) which might be left around by
                      # Mininet or Linux.
```



Check Topology on ONOS GUI

- After building topology with mininet, you can view Topology on ONOS GUI



- Question: why can't see hosts?
 - Switches spontaneously connect to Controller, but hosts do not.
 - Controller knows the existence of switches, but not hosts.



Make hosts appear in ONOS GUI

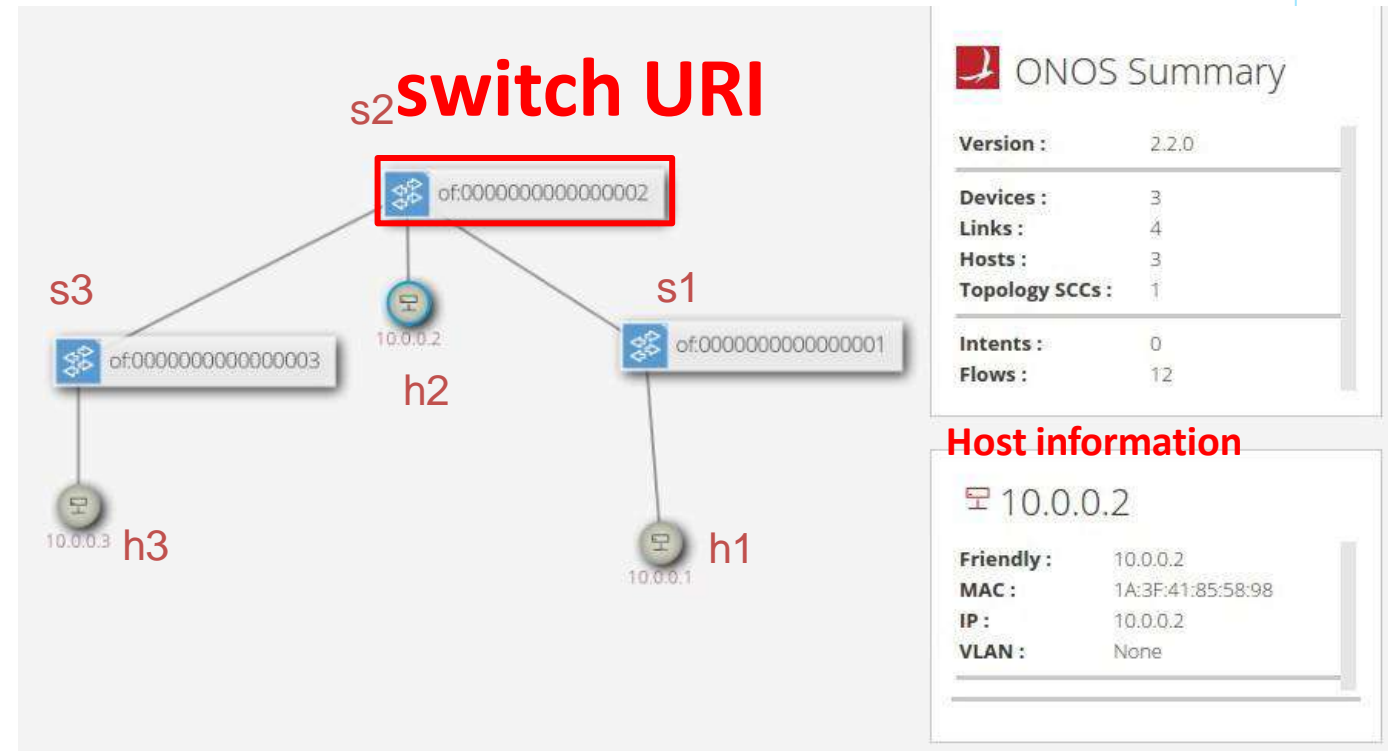
1. First, use “pingall” with Mininet CLI

```
mininet> pingall      # ping between all hosts
```

```
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

2. Hotkeys on GUI

- “h” to show host Information
- “l” to show switch URIs





Create a custom topology

1. Prepare a Python script to create a custom topology.

```
from mininet.topo import Topo

class MyTopo( Topo ):
    def __init__( self ):
        Topo.__init__( self )

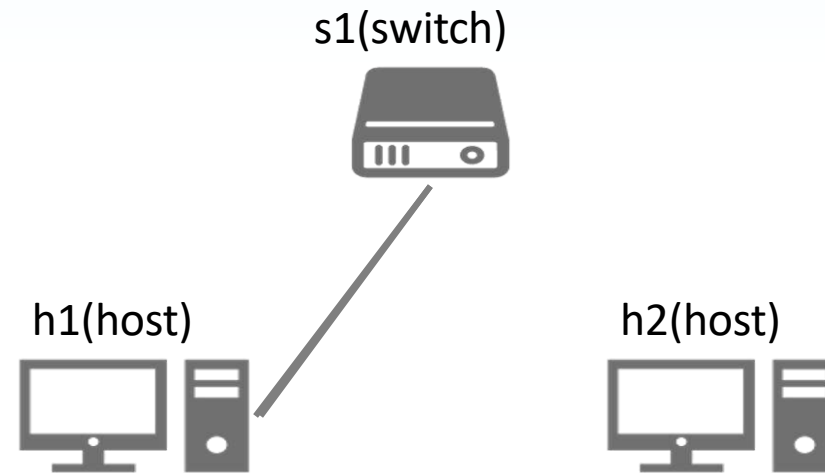
        # Add hosts
        h1 = self.addHost( 'h1' )
        h2 = self.addHost( 'h2' )

        # Add switches
        s1 = self.addSwitch( 's1' )

        # Add links
        self.addLink( h1, s1 )
        self.addLink( h2, s1 )

topos = { 'mytopo': MyTopo }
```

sample.py



2. Run Mininet with options “custom”, “topo”, “controller”, and “switch”

```
$ sudo mn --custom=sample.py --topo=mytopo \
--controller=remote,ip=127.0.0.1,port=6653 \
--switch=ovs,protocols=OpenFlow14
```



Topology Dictionary

- sample.py for the custom topology.

```
from mininet.topo import Topo

class MyTopo( Topo ):
    def __init__( self ):
        Topo.__init__( self )

        # Add hosts
        h1 = self.addHost( 'h1' )
        h2 = self.addHost( 'h2' )

        # Add switches
        s1 = self.addSwitch( 's1' )

        # Add links
        self.addLink( h1, s1 )
        self.addLink( h2, s1 )

        topos = { 'mytopo': MyTopo }
```

Topology Dictionary

```
topos = { 'mytopo': MyTopo }
```

- It is a Python datatype : Dictionary = { key : value }
- "topos" is a reserved word in mininet

```
$ sudo mn --custom=sample.py --topo=mytopo \
--controller=remote,ip=127.0.0.1,port=6653 \
--switch=ovs,protocols=OpenFlow14
```

- Other functions for creating topology
- Topo example



References

- Basic ONOS tutorial
 - <https://wiki.onosproject.org/display/ONOS/Basic+ONOS+Tutorial>
- ONOS GUI:
 - <https://wiki.onosproject.org/display/ONOS/The+ONOS+Web+GUI>
- ONOS CLI:
 - <https://wiki.onosproject.org/display/ONOS/The+ONOS+CLI>
- Mininet intro:
 - <https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#creating>
- Mininet Python API :
 - <http://mininet.org/api/annotated.html>
- Topo example:
 - <https://github.com/mininet/mininet/tree/master/examples>
- Manpage for Linux command
 - netstat: <http://manpages.ubuntu.com/manpages/trusty/man8/netstat.8.html>
 - mn: <http://manpages.ubuntu.com/manpages/bionic/man1/mn.1.html>



Outline

- Environment Introduce & Setup
- Building virtual internet
- Lab Requirements
 - Part1: Answer Questions (40%)
 - Part2: Create a Custom Topology (50%)
 - Part3: Statically Assign Hosts IP Address IP in Mininet (10%)



Part1 : Answer Questions

Activate ONOS APPS

1. When ONOS activate “org.onosproject.openflow,” what APPs does it activate?
2. After we activate ONOS and run P.17 Mininet command, will H1 ping H2 successfully? Why or why not?

Hint: Please refer to the reference “Basic ONOS Tutorial” on p.23

Observe ONOS listening port with terminal command “netstat”

3. Which TCP port does the controller listen to the OpenFlow connection request from the switch? (Take screenshot and explain your answer.)
4. In question 3, which APP enables the controller to listen on the TCP port?

Hint: Observe the Network connection

1. Bring up and enter a new terminal
2. Deactivate/activate apps and use “netstat: in the new terminal to observe network connection

```
$ netstat -nlpt
```

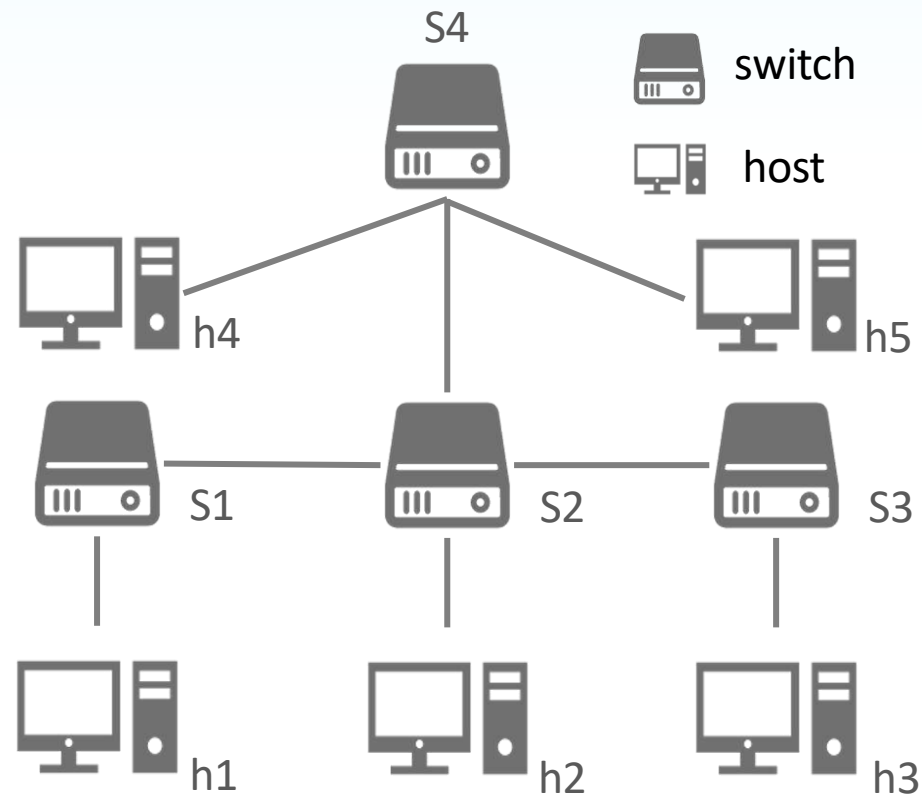
```
#check out command detail with command
```

```
$man netstat
```



Part2 : Create a custom Topology

- Write a Python script to build the following topology:



- Run your Python script and use command “pingall”.
- Then take a screenshot of topology on GUI.



Naming Conventions for part 2

- Naming conventions in your python script
 - a. Name of Python script: `lab1_part2_<studentID>.py`
 - b. Name of topology class: `Lab1_Topo_<studentID>`
 - c. Name of dictionary's key: `topo_part2_<studentID>`

➤ Command to execute your script:

```
$ sudo mn --custom=lab1_part2_<studentID>.py \  
--topo=topo_part2_<studentID> \  
--controller=remote,ip=127.0.0.1:6653 \  
--switch=ovs,protocols=OpenFlow14
```



Part3 : Statically assign Hosts IP Address in Mininet (1)

- Reuse the topology in part 2
- By default, Mininet automatically assigns an IP address and a subnet mask to each host interface
(i.e. 10.0.0.1/8, 10.0.0.2/8, 10.0.0.3/8)

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=11188>
<Host h2: h2-eth0:10.0.0.2 pid=11190>
```

```
mininet> h1 ifconfig
h1-eth0  Link encap:Ethernet  HWaddr ae:c2:c4:b8:d3:ac
         inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
         inet6 addr: fe80::acc2:c4ff:feb8:d3ac/64  Scope:Link
```



Part3 : Statically assign Hosts IP Address in Mininet (2)

- Format for manual assignment of host IP address:
 - 192.168.0.0/27
 - netmask 255.255.255.224

Host	IP Address
h1	192.168.0.1
h2	192.168.0.2
	...

- **Statically assign IP addresses with Python** and hand in the Python script you've edited
- Start mn with your Python script and take screenshots with command “dump” and “ifconfig” for all host.



Naming Conventions for part 3

- Naming conventions in your python script
 - a. Name of Python script: `lab1_part3_<studentID>.py`
 - b. Name of topology class: `Lab1_Topo_<studentID>`
 - c. Name of dictionary's key: `topo_part3_<studentID>`

➤ Note: Command to execute your script:

```
$ sudo mn --custom=lab1_part3_<studentID>.py \  
--topo=topo_part3_<studentID> \  
--controller=remote,ip=127.0.0.1:6653 \  
--switch=ovs,protocols=OpenFlow14
```



Naming Conventions & Submission

- Files

- Two Python scripts:
 - lab1_part2_<studentID>.py
 - lab1_part3_<studentID>.py
- A report: lab1_<studentID>.pdf
 1. Part 1: Answers to those four questions
 2. Part 2: Take screenshots and explain what you've done
 3. Part 3: Take screenshots and explain what you've done
 4. What you've learned or solved

- Submission

- Upload two Python scripts and a report in a directory lab1_<studentID>/
- Zip Python scripts and the report into a zip file
 - Named: lab1_<studentID>.zip
- Wrong file name or format will result in 10 points deduction
- **Deduction 20% for late submission in one week. Won't accept submission over 1 week**



About help!

- **For lab problem, ask at e3 forum**
 - Ask at the e3 forum
 - TAs will help to clarify Lab contents instead of giving answers!
 - Please describe your questions with sufficient context,
 - , e.g., Environment setup, Input/Output, Screenshots, ...
- **For personal problem mail to sdnta@win.cs.nycu.edu.tw**
 - You have special problem and you can't meet the deadline
 - You got weird score with project
- **No Fixed TA hour**



Q & A