

# Project 1

## ONOS and Mininet Installation

### Environment Setup & Basic Operation

Deadline 2023/10/5 (WED) 23:59

Email [sdnta@win.cs.nctu.edu.tw](mailto:sdnta@win.cs.nctu.edu.tw)

# Outline

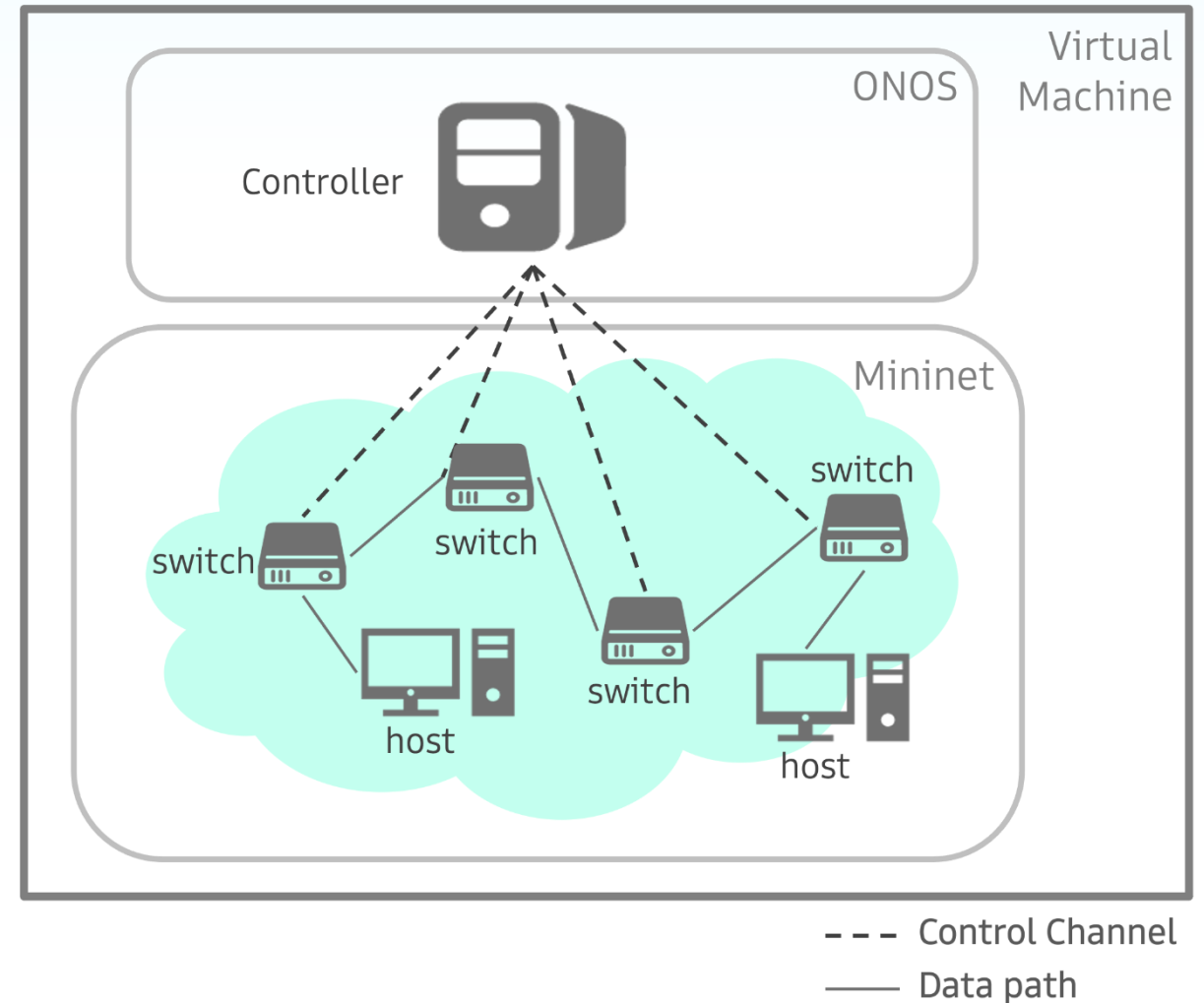
---

- 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
- Project Requirements
  - Part 1: Answer Questions
  - Part 2: Write a Custom Topology
  - Part 3: Statically Assign Hosts IP Address In Mininet

# Overview

If we want to emulate a network we will need

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



# VirtualBox, Bazel, ONOS, Mininet and OVS Installation

- Bazel: Free software tool for “automation of building and testing of software.”
- Open Network Operating System (ONOS): Open source network controller for SDN.
- 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
- Installation:
  - Follow [SDN\\_Environment\\_Setup.pdf](#)
  - Use [TA-provided env\\_setup.sh](#)

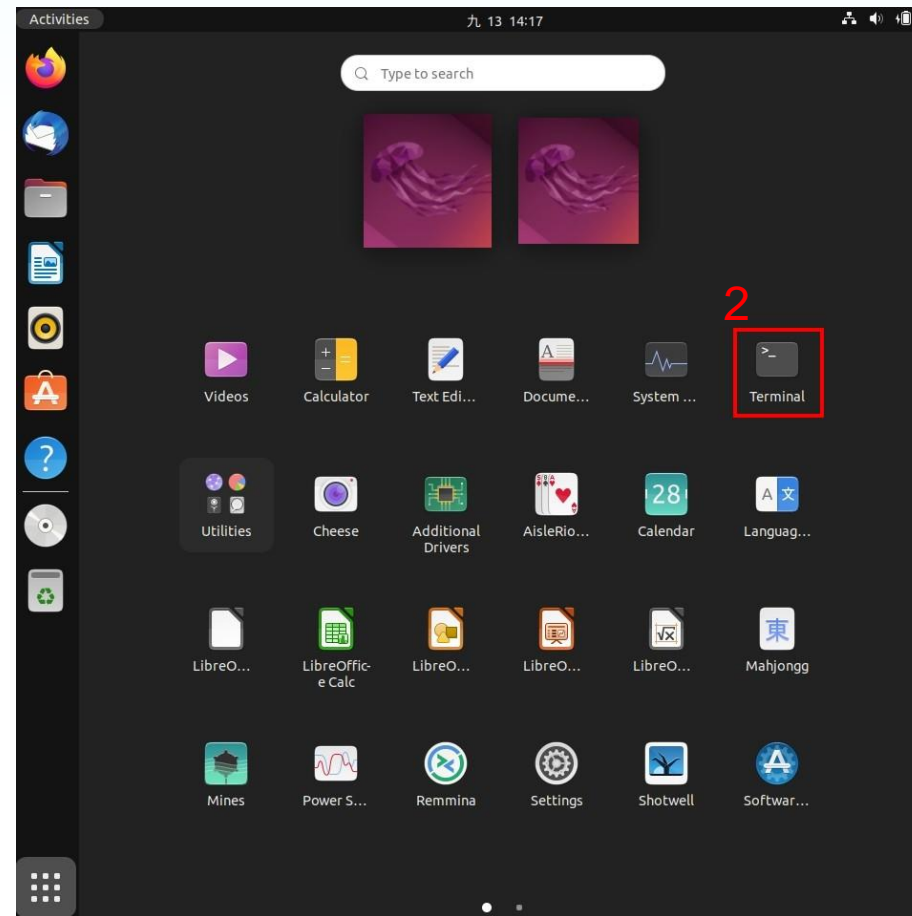
# Outline

---

- 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
- Project Requirements

# Build ONOS

- Open a new terminal



# 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)
```

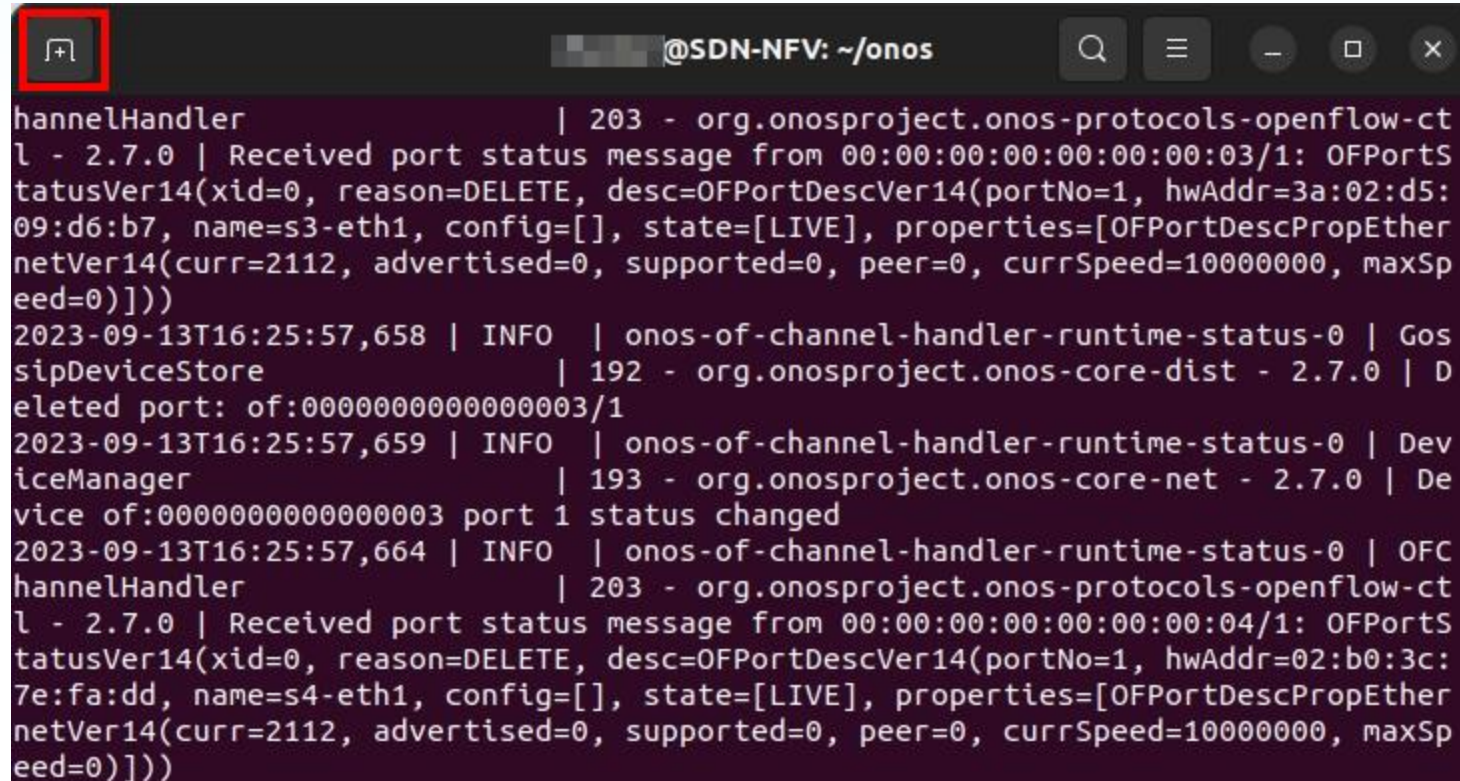
```
demo@SDN-NFV:~/onos$ bazel run onos-local -- clean debug
INFO: Analyzed target //:onos-local (0 packages loaded, 0 targets configured).
INFO: Found 1 target...
Target //:onos-local_current-jdk up-to-date:
  bazel-bin/onos-runner_current-jdk
INFO: Elapsed time: 0.486s, Critical Path: 0.00s
INFO: 0 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-provid
ConfigurationEvent: pid=org.onosproject.net.intent.impl.IntentCleanup) | OpenFlowRuleProvider | 203 - org.onosproject.onos-provid
se
AtomixClusterStore | 192 - org.onosproject.onos-core-primitives - 2.2.0 | Updated node 127.0.0.1 state to READY
```



# ONOS CLI

- Open another terminal tab



The screenshot shows a terminal window titled "@SDN-NFV: ~/onos". In the top-left corner, a button with a plus sign and a terminal icon is highlighted with a red box, indicating how to open a new terminal tab. The terminal displays log messages from the ONOS system, including channel handler status updates and port status messages.

```
channelHandler | 203 - org.onosproject.onos-protocols-openflow-ct
l - 2.7.0 | Received port status message from 00:00:00:00:00:00:03/1: OFPorts
tatusVer14(xid=0, reason=DELETE, desc=OFPortDescVer14(portNo=1, hwAddr=3a:02:d5:
09:d6:b7, name=s3-eth1, config=[], state=[LIVE], properties=[OFPortDescPropEther
netVer14(curr=2112, advertised=0, supported=0, peer=0, currSpeed=10000000, maxSp
eed=0)))
2023-09-13T16:25:57,658 | INFO | onos-of-channel-handler-runtime-status-0 | Gos
sipDeviceStore | 192 - org.onosproject.onos-core-dist - 2.7.0 | D
eleted port: of:0000000000000003/1
2023-09-13T16:25:57,659 | INFO | onos-of-channel-handler-runtime-status-0 | Dev
iceManager | 193 - org.onosproject.onos-core-net - 2.7.0 | De
vice of:0000000000000003 port 1 status changed
2023-09-13T16:25:57,664 | INFO | onos-of-channel-handler-runtime-status-0 | OFC
channelHandler | 203 - org.onosproject.onos-protocols-openflow-ct
l - 2.7.0 | Received port status message from 00:00:00:00:00:00:04/1: OFPorts
tatusVer14(xid=0, reason=DELETE, desc=OFPortDescVer14(portNo=1, hwAddr=02:b0:3c:
7e:fa:dd, name=s4-eth1, config=[], state=[LIVE], properties=[OFPortDescPropEther
netVer14(curr=2112, advertised=0, supported=0, peer=0, currSpeed=10000000, maxSp
eed=0)))
```



# ONOS CLI

- Enter ONOS CLI

```
demo@SDN-NFV:~/onos$ 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 > █
```

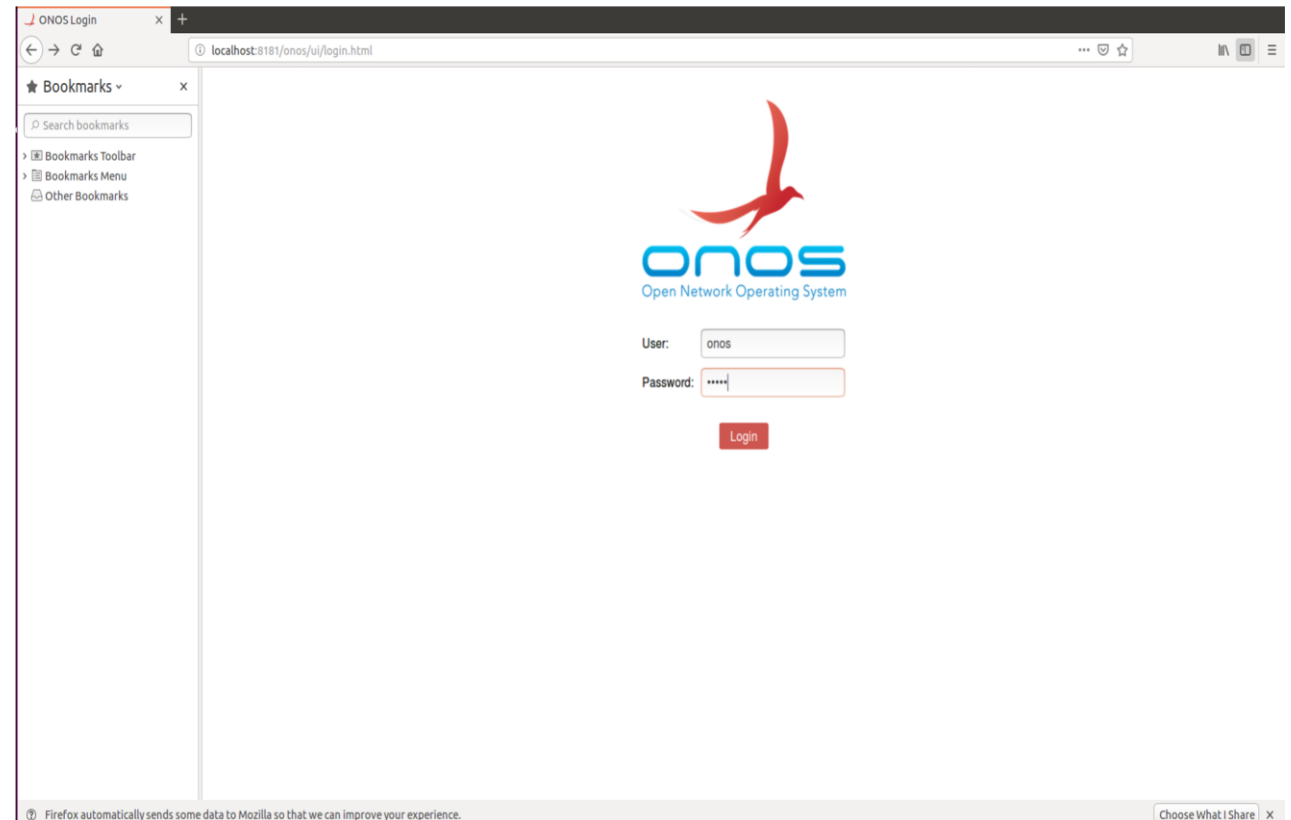
- Reference: [ONOS CLI command](#)

# 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 basic ONOS APPs
    - Method1: Via ONOS CLI
    - Method2: Via ONOS GUI
  - Create a topology controlled with Mininet
    - Method 1: Built-in Topology
    - Method 2: Custom Topology
- Project Requirements

# CLI shows APPs

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

```
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

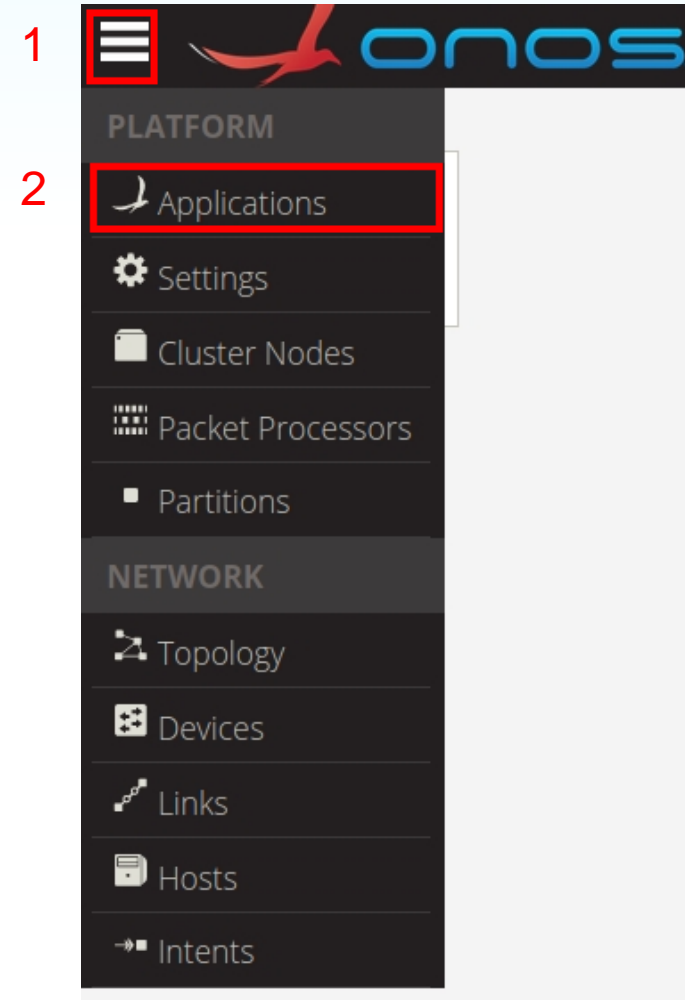
onos> app deactivate <name> # deactivate onos app

```
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

- Open the Application list in ONOS GUI
  1. Click 
  2. Choose “Applications”



# Activate basic ONOS APPs via GUI

## ■ Activate apps

1. Choose the app which you want to activate from APPs list
2. Click
3. Click "OK"

The screenshot displays the ONOS Open Network Operating System GUI. At the top, the header shows the ONOS logo and the text 'Open Network Operating System'. Below the header, there is a section titled 'Applications (121 Total)' with a table listing various applications. A red box labeled '1' highlights the 'Reactive Forwarding' application, which has the APP ID 'org.onosproject.fwd'. To the right of the table, a 'Confirm Action' dialog box is open, showing the details for the selected application. A red box labeled '2' highlights the 'Activate' button (a play icon) in the top right of the dialog. Another red box labeled '3' highlights the 'OK' button in the bottom right of the dialog. The dialog also displays the application's category ('Traffic Steering'), version ('1.11.0'), origin ('ONOS Community'), and role ('UNSPECIFIED'). A description of the application's functionality is provided at the bottom of the dialog.

Applications (121 Total)

| TITLE                   | APP ID                     |
|-------------------------|----------------------------|
| Reactive Forwarding     | org.onosproject.fwd        |
| Route Service Server    | org.onosproject.route-ser  |
| SDN-IP                  | org.onosproject.sdnip      |
| SDN-IP Reactive Routing | org.onosproject.reactive-r |
| SNMP Provider           | org.onosproject.snmp       |
| Scalable Gateway        | org.onosproject.scalableg  |
| Segment Routing         | org.onosproject.segment    |
| TE Topology Core        | org.onosproject.tetopolog  |
| TE Tunnel               | org.onosproject.tetunnel   |

**Confirm Action**  
Activate org.onosproject.fwd

**Category:** Traffic Steering  
**Version:** 1.11.0  
**Origin:** ONOS Community  
**Role:** UNSPECIFIED

<http://onosproject.org>

Provisions traffic between end-stations using hop-by-hop flow programming by intercepting packets for which there are currently no matching flow objectives on the data plane. The paths paved in this manner are short-lived, i.e. they expire a few seconds after the flow on whose behalf they were programmed stops. The application relies on the ONOS path service to compute the shortest paths. In the event of negative topology events (link loss, device disconnect, etc.), the application will proactively invalidate any paths that it had programmed to lead through the resources that are no longer available.



# Outline

---

- Environment Introduce & Setup
- Building virtual internet
  - Build ONOS
    - ONOS CLI
    - ONOS GUI
  - Activate basic ONOS APPs
    - Method1: Via ONOS CLI
    - Method2: Via ONOS GUI
  - Create a topology controlled with Mininet
    - Method 1: Built-in Topology
    - Method 2: Custom Topology
- Project Requirements

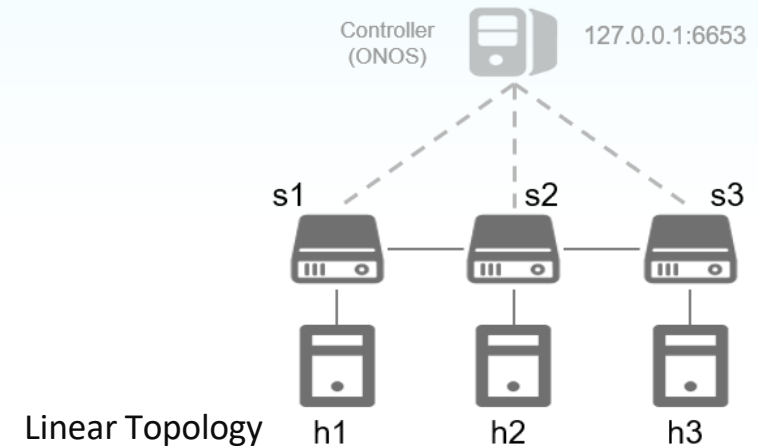
# Build-in Topology in Mininet

## ■ 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

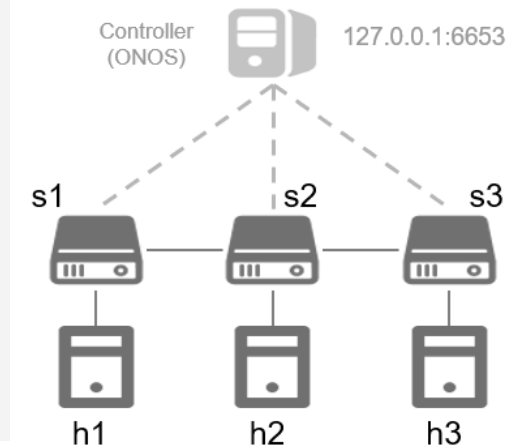
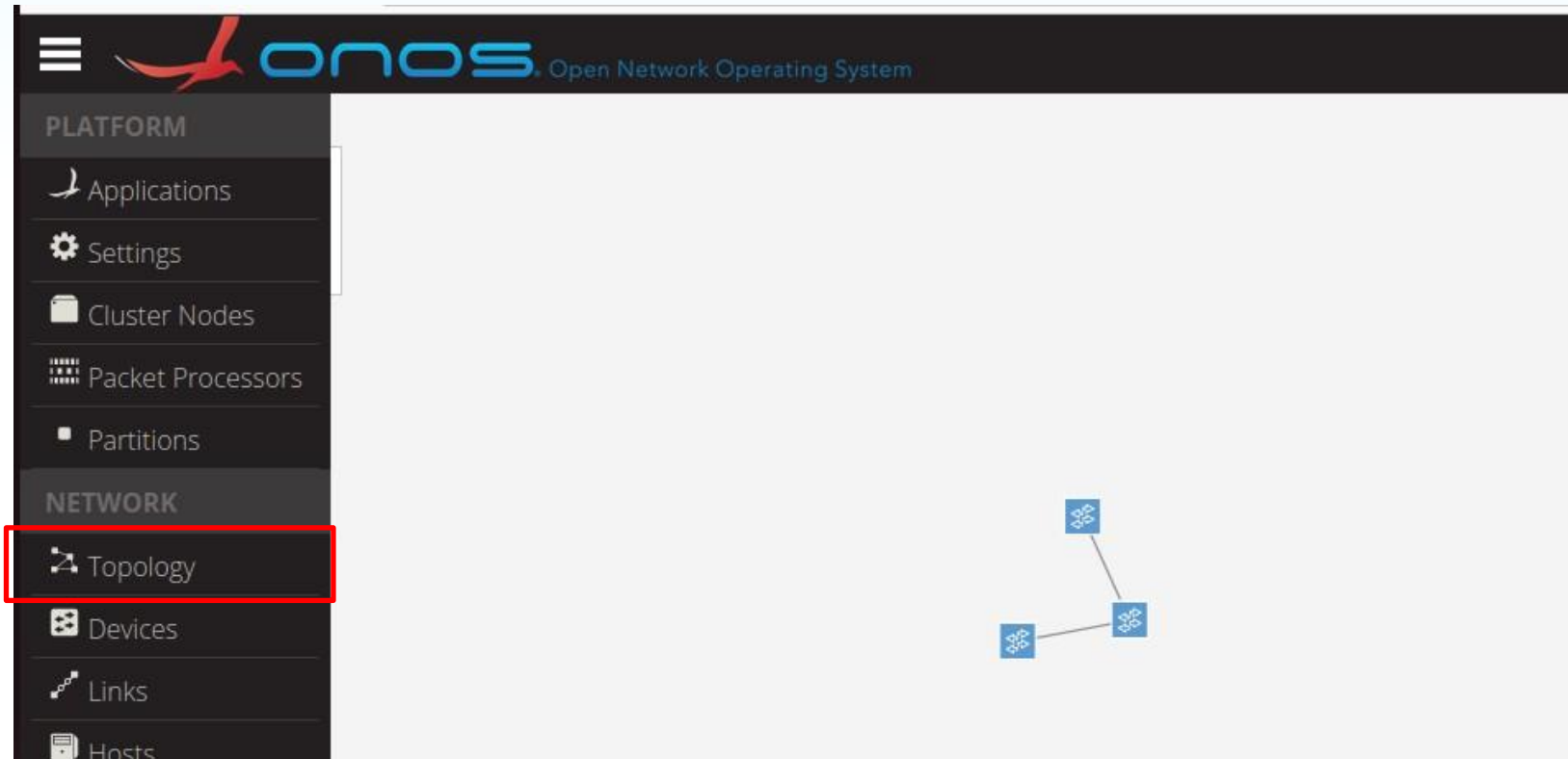
- **Note:** Make sure to clean up the environment of Mininet after every time you exit Mininet CLI

```
mininet> exit          # exit mininet cli
$ sudo mn -c          #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

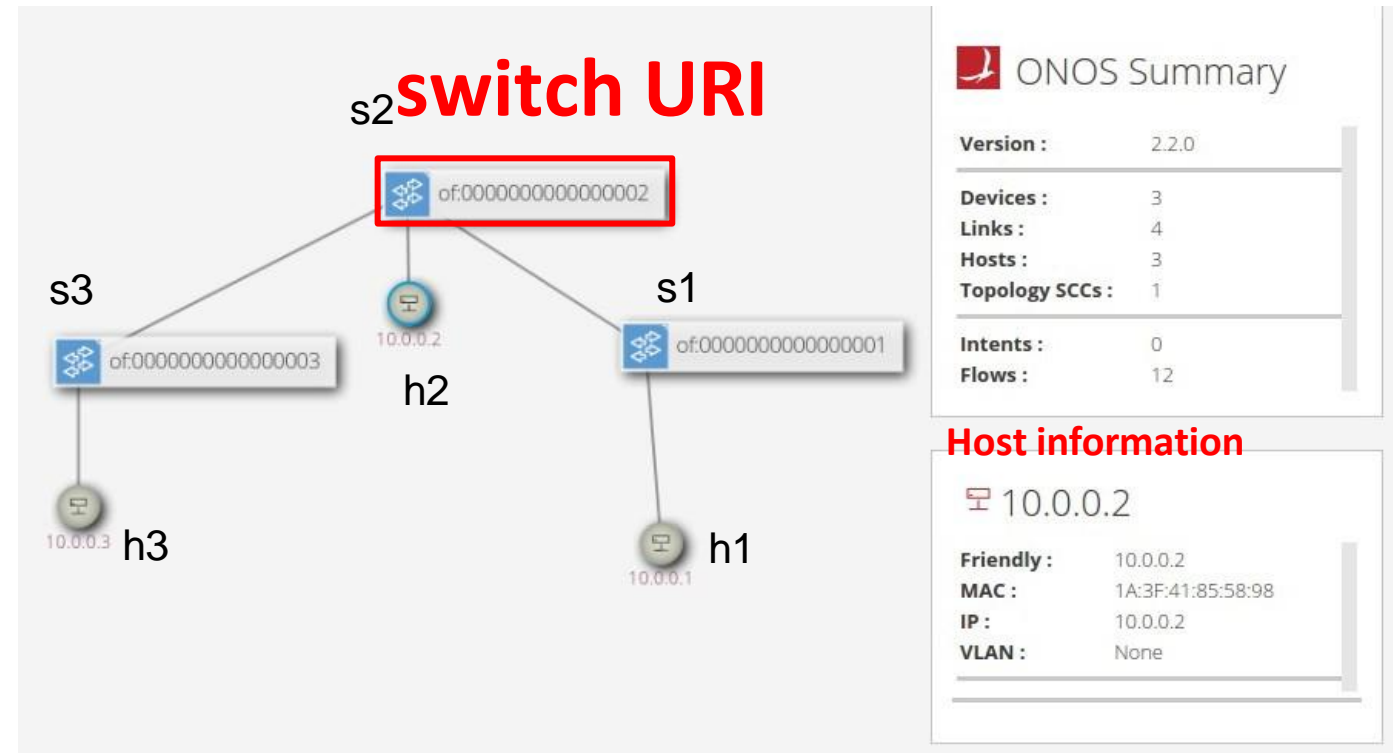
1. First, use “pingall” with Mininet CLI

```
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 hosts
- “l” to show switch URI



# Create a custom topology

## 1. Specify topology in Python script

```
from mininet.topo import Topo

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

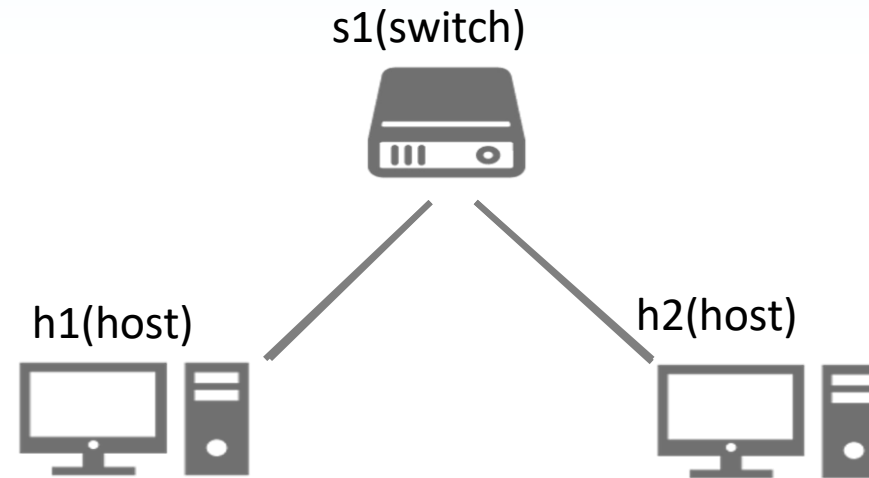
        # Add hosts
        h1 = self.addHost( 'h1' )
        h2 = self.addHost( 'h2' )

        # 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

- Recall: create a custom topology specified in sample.py

```
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 }
```

```
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
- Project Requirements
  - Part1: Answer Questions (40%)
  - Part2: Create a Custom Topology (50%)
  - Part3: Statically Assign Hosts IP Address IP in Mininet (10%)

# Preparation

1. Close all the exist processes(terminals) Hint: Please refer to p.17 to clear the Mininet environment completely
2. Restart ONOS

```
demo@SDN-NFV:~$ cd $ONOS_ROOT
demo@SDN-NFV:~/onos$ bazel run onos-local -- clean debug
```

3. Enter ONOS CLI

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

4. Check the activating apps are the same as below (It doesn't matter if the sequence numbers are not the same as those in your terminal)

```
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
```

# Part1 : Answer Questions

## Activate ONOS APPs

1. When ONOS activates “org.onosproject.openflow,” what are the APPs which it also activates?
2. After activating ONOS and running the commands on P.17 and P.20. Will H1 ping H2 successfully? Why or why not?

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

## Observe listening port with terminal command “netstat”

3. Which TCP port the controller listens for the OpenFlow connection request from the switch? screenshot
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

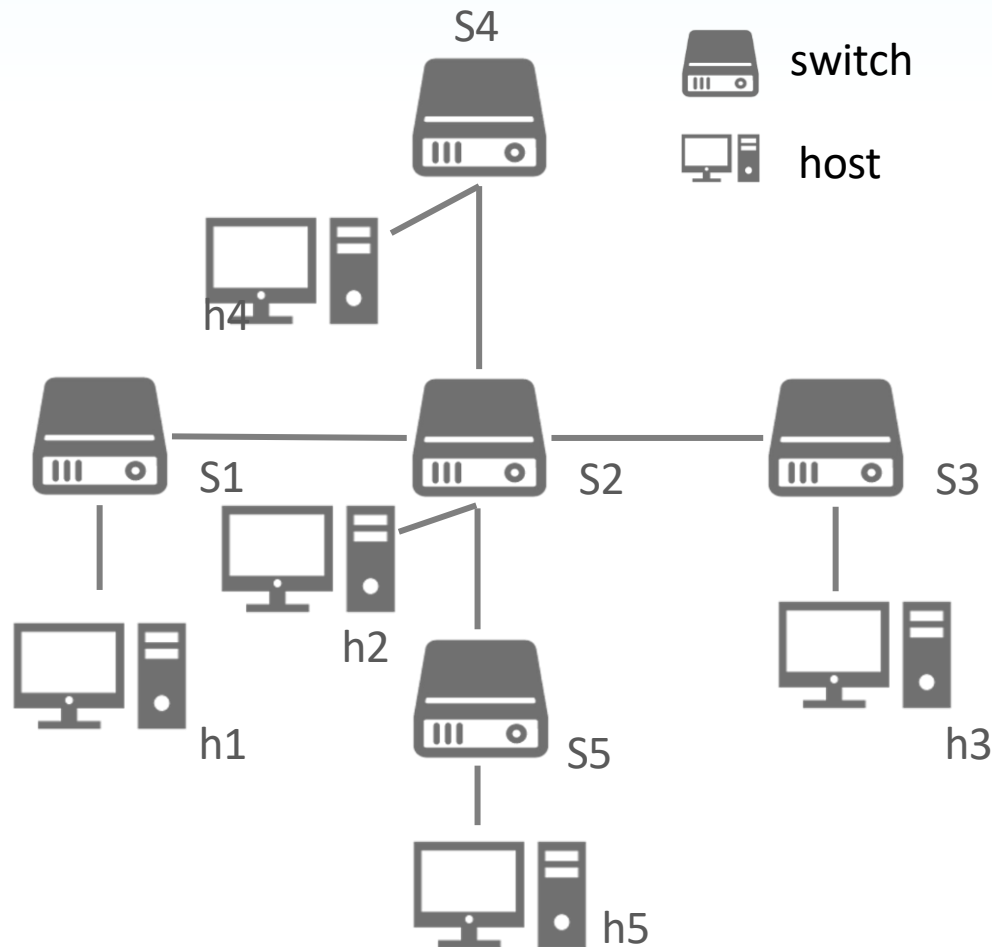
2. Deactivate/activate apps and use “netstat: in the new terminal to observe network connection

```
$ 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: `project1_part2_<studentID>.py`
  - b. Name of topology class: `Project1_Topo_<studentID>`
  - c. Name of dictionary's key: `topo_part2_<studentID>`
- Command to execute your script:

```
$ sudo mn --custom=project1_part2_<studentID>.py \
--topo=topo_part2_<studentID> \
--controller=remote,ip=127.0.0.1:6653
```

## 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: `project1_part3_<studentID>.py`
  - b. Name of topology class: `Project1_Topo_<studentID>`
  - c. Name of dictionary's key: `topo_part3_<studentID>`
- Note: Command to execute your script:

```
$ sudo mn --custom=project1_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:
  - `project1_part2_<studentID>.py`
  - `project1_part3_<studentID>.py`
- A report: `project1_<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

- Put two Python scripts and report in a directory `project1_<studentID>`
- Zip Python scripts and the report into a zip file
  - Named: `project1_<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.nctu.edu.tw](mailto:sdnta@win.cs.nctu.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