

Lab 2

OpenFlow Protocol Observation and Flow Rule Installation

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

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



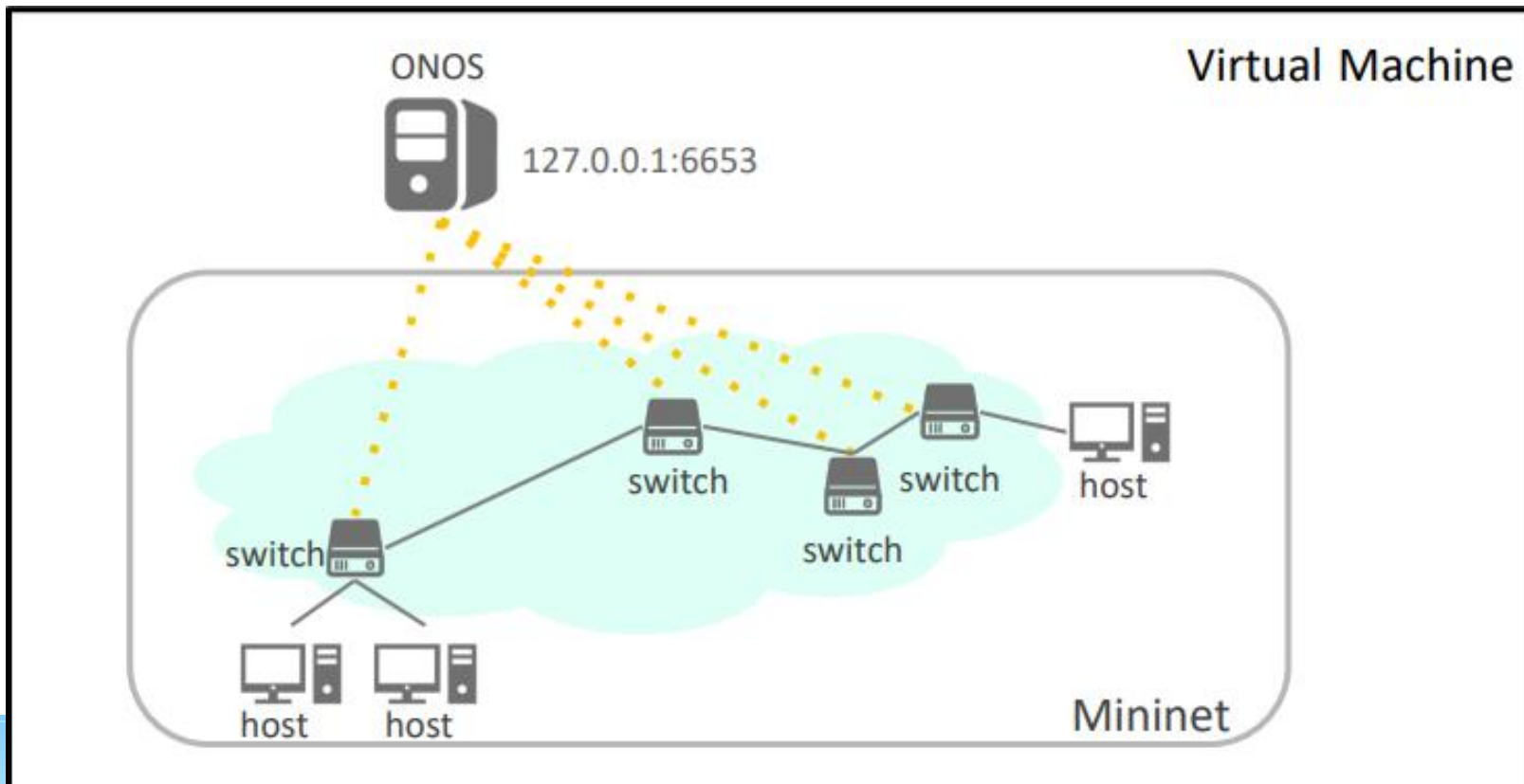
Outline

- OpenFlow Messages
 - Monitor Traffic between ONOS & Switches
 - OpenFlow Message Observation
- Install/Delete Flow Rules
 - Rest, JSON file, and Curl introduction
 - ONOS and Topology Setup
 - Method 1: via Command “curl”
 - Method 2: via ONOS Web GUI
- Lab 2 Requirements
 - Part 1: Answer Questions
 - Part 2: Install Flow Rules
 - Part 3: Create Broadcast Storm
 - Part 4: Trace ReactiveForwarding



OpenFlow Protocol

- OpenFlow is a Software Defined Network (SDN) control protocol
- ONOS SDN controller uses OpenFlow messages to communicate with OVS switches.
 - Hello, Packet-in/out, Flow Install/Remove, etc.





Wireshark Installation

- Wireshark is an open-source and widely-used network packet analyzer
 - Can capture packets on any specified interface

- Installation steps:

1. Update package information

```
$ sudo apt update          # update all packages information
```

2. Install Wireshark

```
$ sudo apt install wireshark -y
```

- Start Wireshark

```
$ sudo wireshark
```

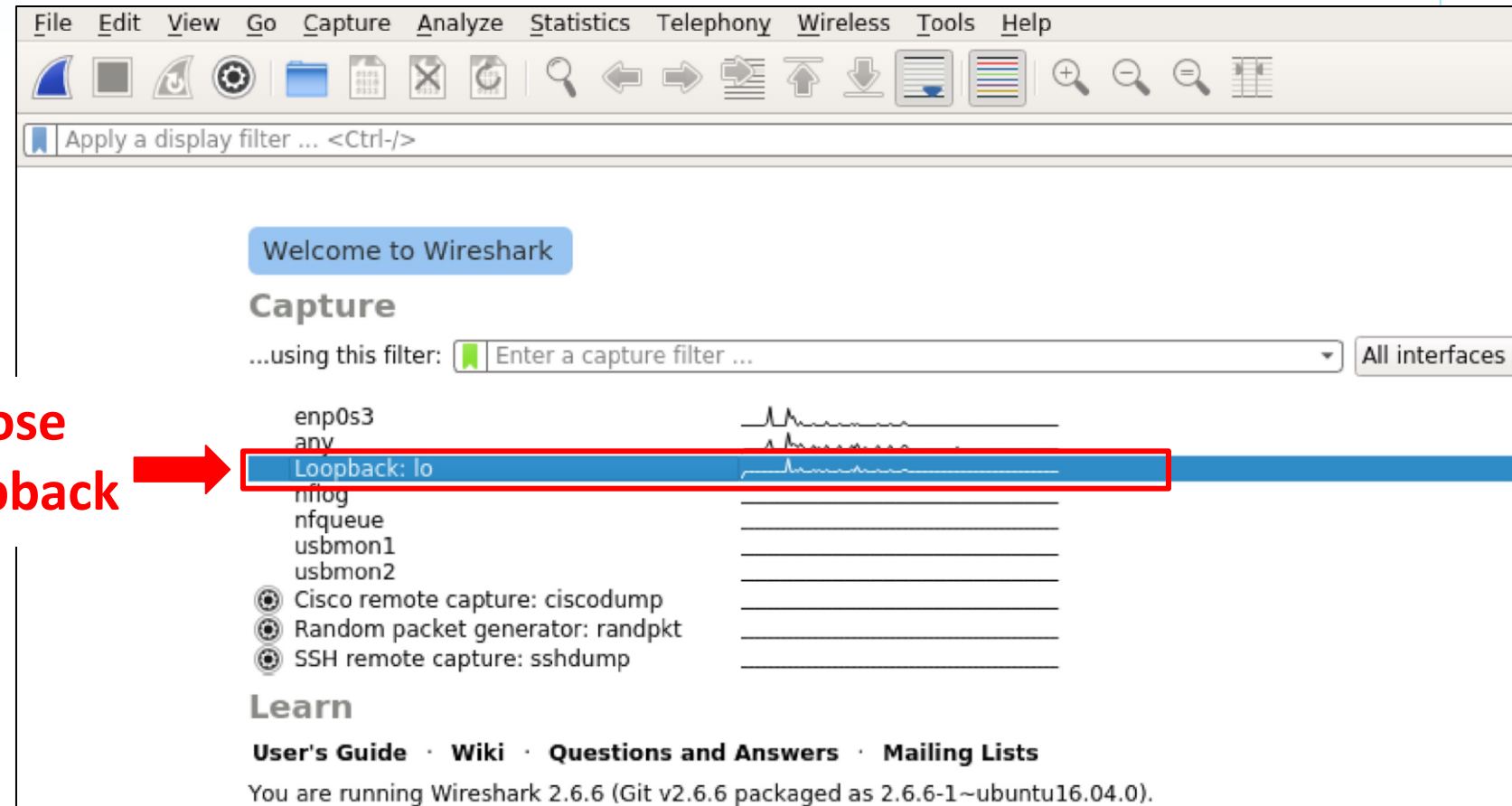




Capture Packets in Wireshark

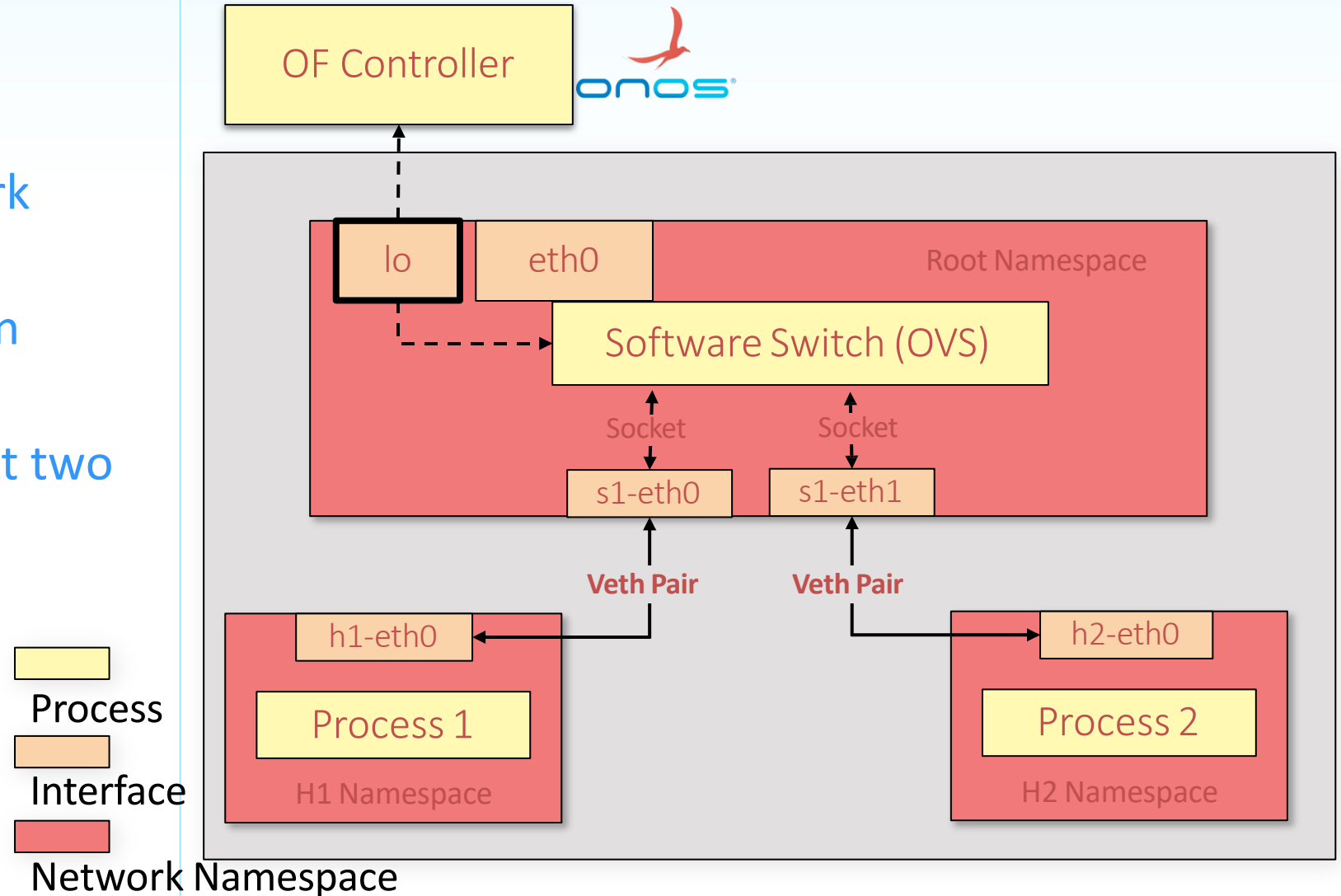
- Both ONOS and Mininet run locally in VM
 - We can capture packets on the Loopback (lo) interface
- Loopback:
Route data streams back to the source without intentional processing or modification.

**Choose
Loopback**



Mininet and Network Namespace

- Mininet uses network namespace to emulate networks
- OVS runs in root network namespace
- Each host runs in its own network namespace
- Use Veth pair to connect two networks of different namespaces





Sending OpenFlow Messages

1. Start ONOS

2. Activate ReactiveForwarding

```
onos> apps -a -s          # (optional) check activated application
onos> app activate fwd     # activate ReactiveForwarding
```

3. Start Mininet with default topology

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

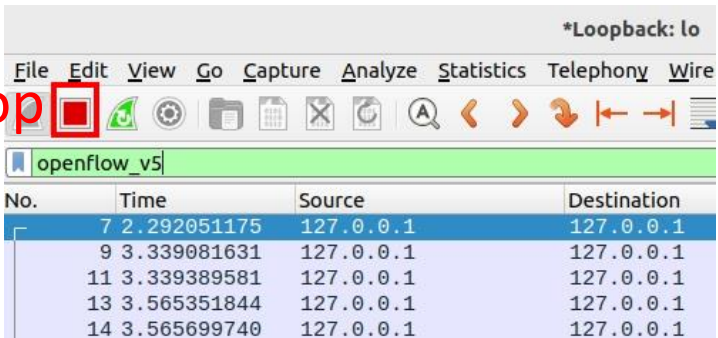
4. H1 ping H2 in Mininet

```
mininet> h1 ping h2 -c 5          # send five ICMP echo_request packets
```

5. Exit Mininet when ping terminates


6. Stop capturing packets in Wireshark

7. Observe captured OpenFlow packets

Stop 

No.	Time	Source	Destination
7	2.292051175	127.0.0.1	127.0.0.1
9	3.339081631	127.0.0.1	127.0.0.1
11	3.339389581	127.0.0.1	127.0.0.1
13	3.565351844	127.0.0.1	127.0.0.1
14	3.565699740	127.0.0.1	127.0.0.1



- 
- The screenshot shows the Wireshark interface. The packet list pane on the left has a red box around the entry 'openflow_v5'. The packet details pane on the right shows the 'Ethernet II' section with the destination MAC address '08:00:27:00:00:00'.

-
- The screenshot shows the Wireshark interface with a packet list on the left and a packet details pane on the right. The packet list shows a packet of type OFPT_PACKET_OUT. The packet details pane shows the details of the OFPT_PACKET_OUT packet, including the Transaction ID, Buffer ID, In port, and Actions. A red box highlights the packet details pane, and a red arrow points to the 'Apply as Filter' button in the packet list pane.
- Packet List:
- | No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------|-----------|-------------|----------|--------|-----------------------|
| 335 | 9.900829544 | 127.0.0.1 | 127.0.0.1 | OpenFlow | 245 | Type: OFPT_PACKET_OUT |
| 337 | 9.901126392 | 127.0.0.1 | 127.0.0.1 | OpenFlow | 245 | Type: OFPT_PACKET_OUT |
| 340 | 9.901371527 | 127.0.0.1 | 127.0.0.1 | OpenFlow | 245 | Type: OFPT_PACKET_OUT |
| 341 | 0.002242270 | 127.0.0.1 | 127.0.0.1 | OpenFlow | 245 | Type: OFPT_PACKET_OUT |
- Packet Details:
- ```

Frame 340: 245 bytes on wire (1960 bits), 245 bytes captured (1960 bits) on interface 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 6653, Dst Port: 43640, Seq: 4808, Ack: 19327, Len: 179
OpenFlow 1.4
 Version: 1.4 (0x05)
 Type: OFPT_PACKET_OUT (13)
 Length: 179
 Transaction ID: 164
 Buffer ID: OFF_NO_BUFFER (429496729)
 In port: OFPP_CONTROLLER (429496729)
 Actions length: 16
 Pad: 00000000000000
 Action
 Data

```
- Packet List Actions:
- | Action               | Shortcut            |
|----------------------|---------------------|
| Collapse Subtrees    | Shift+Left          |
| Expand All           | Ctrl+Right          |
| Collapse All         | Ctrl+Left           |
| Apply as Column      | Ctrl+Shift+I        |
| Apply as Filter      | Selected            |
| Prepare a Filter     | Not Selected        |
| Conversation Filter  | ...and Selected     |
| Colorize with Filter | ...or Selected      |
| Follow               | ...and not Selected |





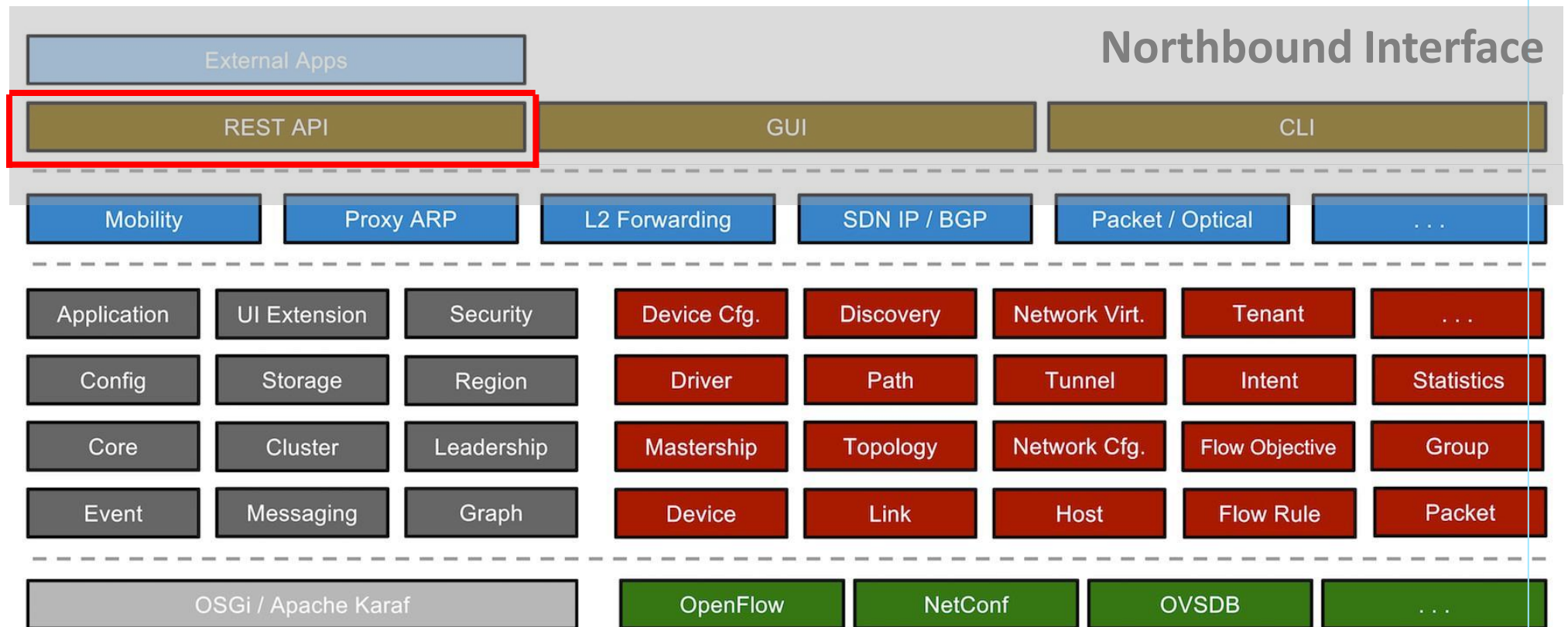
# Outline

- OpenFlow Messages
- Install/Delete Flow Rules
  - Rest, JSON file, and Curl introduction
  - ONOS and Topology Setup
  - Method 1: via Command “curl”
  - Method 2: via ONOS Web GUI
- Lab 2 Requirements



# ONOS Northbound Interface

- Northbound Interface of ONOS is the interface that interacts with programmers
- REST is a **software architectural style** for Web services
- We will use the REST API to install/delete flow rules





# Create a JSON file of flow rules

flows1.json

- What is a JSON (JavaScript Object Notation)?
  - JSON
    - An open standard file format and data interchange format
    - Uses human-readable text to store and transmit data objects consisting of
      - attribute–value pairs
      - arrays
      - other serializable values
- E.g., flows1.json: JSON file for a flow rule
- Hint:
  - Priority of preinstalled flow rules: 40000
  - The priorities of flow rules **MUST** be 40001 to 65535
- Ref: Flow Rule Criteria & Instructions

```
{
 "priority": 50000,
 "timeout": 0,
 "isPermanent": true,
 "selector": {
 "criteria": [
 {
 "type": "IN_PORT",
 "port": 1
 }
]
 },
 "treatment": {
 "instructions": [
 {
 "type": "OUTPUT",
 "port": 2
 }
]
 }
}
```

# JSON File: Match Fields (criteria)

flows1.json

```
{
 "priority": 50000,
 "timeout": 0,
 "isPermanent": true,
 "selector": {
 "criteria": [
 {
 "type": "IN_PORT",
 "port": 1
 }
]
 },
 "treatment": {
 "instructions": [
 {
 "type": "OUTPUT",
 "port": 2
 }
]
 }
}
```

- Match fields may have dependency; please refer to OpenFlow spec v1.4.0.

```
"selector": {
 "criteria": [
 {
 "type": "IN_PORT",
 "port": 1
 },
 { ... },
 ...
]
},
```

# JSON File: Actions (instructions)

flows1.json

```
{
 "priority": 50000,
 "timeout": 0,
 "isPermanent": true,
 "selector": {
 "criteria": [
 {
 "type": "IN_PORT",
 "port": 1
 }
]
 },
 "treatment": {
 "instructions": [
 {
 "type": "OUTPUT",
 "port": 2
 }
]
 }
}
```

```
"treatment": {
 "instructions": [
 {
 "type": "OUTPUT",
 "port": 2
 },
 { ... },
 ...
]
}
```



# Curl – Command Tool for Transferring Data with URL

- Command format

```
curl [options] [URL...]
```

- Transferring data with URL

```
$ curl -u <user:password> -X <method-type> -H <header> -d <data> [URL...]
```

```
option "-X" specifies a HTTP request method
```

```
option "-H" includes extra header in the HTTP request
```

```
option "-d" sends specified data in a POST request
```

```
URL (Uniform Resource Locator)
```

- "<data>" can be a file name with prefix "@"

```
$ curl -u <user:password> -X <method-type> -H <header> -d @<filename> [URL...]
```

- References:

- ["request methods" in HTTP](#)
- [Manpage for command "curl"](#)

## Review:

- user: onos
- password: rocks



# Outline

- OpenFlow Messages
- Install/Delete Flow Rules
  - Rest, JSON file, and Curl introduction
  - **ONOS and Topology Setup**
  - Method 1: via Command “curl”
  - Method 2: via ONOS Web GUI
- Project 2 Requirements



# ONOS & Topology Setup

## 1. Restart ONOS

1. <ctrl+c> in the ONOS log panel to shutdown the ONOS instance
2. Start ONOS

```
demo@SDN-NFV:~/onos$ ok clean
ok is an alias of command "bazel run onos-local -- "
```

## 2. Deactivate ReactiveForwarding APP

```
onos> app deactivate fwd # deactivate ReactiveForwarding
```

## 3. Start Mininet with default (minimal) topology

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

## 4. Make sure that two hosts CAN NOT ping each other

```
mininet> h1 ping h2
```

```
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
From 10.0.0.1 icmp_seq=4 Destination Host Unreachable
```





# Outline

- OpenFlow Messages
- Install/Delete Flow Rules
  - Rest, JSON file, and Curl introduction
  - ONOS and Topology Setup
  - **Method 1: via Command “curl”**
  - Method 2: via ONOS Web GUI
- Lab 2 Requirements



# Upload JSON File to ONOS

- Install flow rules on ONOS with JSON file (flows1.json)

```
$ curl -u onos:rocks -X POST \
> -H 'Content-Type: application/json' \
> -d @flow1.json \
> 'http://localhost:8181/onos/v1/flows/of:000000000000000001'
```

*#Recall command from p.15*  
\$ curl -u <user:password>  
-X <method-type>  
-H <header>  
-d @<filename>  
[URL...]

## Device ID

- DeviceID MUST be the URI shown in the ONOS web GUI
- DeviceID can be set by either ONOS or user specified topology file (i.e. \*.py)





of:000000000000000001

URI : of:000000000000000001  
Vendor : Nicira, Inc.  
H/W Version : Open vSwitch



# Check whether the flow rule is installed (1/2)

1. Go to ONOS web GUI (<http://localhost:8181/onos/ui>)
2. Left click on  . Then, the panel of switch info will pop out
3. Left click on  .

2. Left click



3. Left click



Flow rules in switch

| STATE | PACKETS | DURATION | FLOW PRIORITY | TABLE NAME | SELECTOR      | TREATMENT                            | APP NAME |
|-------|---------|----------|---------------|------------|---------------|--------------------------------------|----------|
| Added | 0       | 36       | 50000         | 0          | IN_PORT:1     | imm[OUTPUT:2], cleared:false         | *rest    |
| Added | 0       | 960      | 40000         | 0          | ETH_TYPE:bddp | imm[OUTPUT:CONTROLLER], cleared:true | *core    |
| Added | 0       | 960      | 40000         | 0          | ETH_TYPE:lldp | imm[OUTPUT:CONTROLLER], cleared:true | *core    |
| Added | 12      | 960      | 40000         | 0          | ETH_TYPE:arp  | imm[OUTPUT:CONTROLLER], cleared:true | *core    |



# Check whether the flow rule is installed (2/2)

| SELECTOR    |         |            | TREATMENT                    |            |               | APP NAME                             |          |  |
|-------------|---------|------------|------------------------------|------------|---------------|--------------------------------------|----------|--|
| IN_PORT:1   |         |            | imm[OUTPUT:2], cleared:false |            |               | *rest                                |          |  |
| STATE       | PACKETS | DURATION ▲ | FLOW PRIORITY                | TABLE NAME | SELECTOR      | TREATMENT                            | APP NAME |  |
| Pending Add | 0       | 0          | 50000                        | 0          | IN_PORT:1     | imm[OUTPUT:2], cleared:false         | *rest    |  |
| Added       | 0       | 10,485     | 40000                        | 0          | ETH_TYPE:bddp | imm[OUTPUT:CONTROLLER], cleared:true | *core    |  |
| Added       | 0       | 10,485     | 40000                        | 0          | ETH_TYPE:arp  | imm[OUTPUT:CONTROLLER], cleared:true | *core    |  |
| Added       | 0       | 10,485     | 40000                        | 0          | ETH_TYPE:lldp | imm[OUTPUT:CONTROLLER], cleared:true | *core    |  |

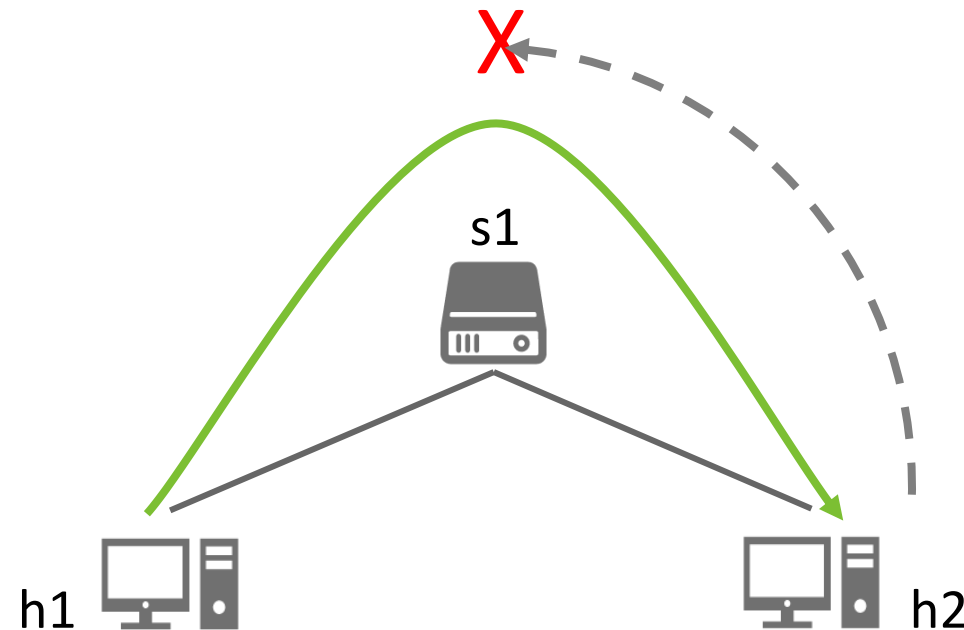
- Flow Rule States:

- **PENDING\_ADD** - ONOS has received a request from the application to install the flow rule, but that flow has NOT yet been observed on the device.
- **ADDED** - Once the flow rule subsystem observes the flow on the device, it will change to this state.



# Why Hosts Still Cannot Ping Each Other?

- Because we have only installed a flow rule for one direction
  - s1 can forward packets from h1 to h2
  - But, s1 CANNOT forward packets from h2 to h1
    - By default, s1 drops a packet if the packet does not match any flow rule
      - i.e. **table-miss**



# Delete Flow Rules (1/2)

- Use URL to find flow rule IDs **in a switch**

Ex. <http://localhost:8181/onos/v1/flows/of:000000000000000001>

```
▼ flows:
 ▶ 0: {}
 ▼ 1:
 groupId: 0
 state: "ADDED"
 life: 515
 liveType: "UNKNOWN"
 lastSeen: 1695031088914
 packets: 9
 bytes: 378
 id: "49539596291667367"
 appId: "org.onosproject.rest"
 priority: 50000
 timeout: 0
 isPermanent: true
 deviceId: "of:000000000000000001"
 tableId: 0
 tableName: "0"
 ▶ treatment: {}
 ▶ selector: {}
 ▶ 2: {}
 ▶ 3: {}
```

```
id: "49539596291667367"
appId: "org.onosproject.rest"
```

- ID of the flow we just added is 49539596291667367

- Alternatively, we could use “curl” to get flow information

```
$ curl -u onos:rocks -X GET -H 'Accept: application/json' \
'http://localhost:8181/onos/v1/flows/of:000000000000000001'
```



## Delete Flow Rules (2/2)

- Then, delete the flow rule with flowID: 49539596291667367

```
$ curl -u onos:rocks \
-X DELETE \
-H 'Accept: application/json' \
'http://localhost:8181/onos/v1/flows/of:000000000000000001\
/49539596291667367'
```

**Flow ID**

**Device ID**



# Outline

- OpenFlow Messages
- **Install/Delete Flow Rules**
  - Rest, JSON file, and Curl introduction
  - ONOS and Topology Setup
  - Method 1: via Command “curl”
  - **Method 2: via ONOS Web GUI**
- Lab 2 Requirements





# REST API on ONOS Web GUI

- Browse <http://localhost:8181/onos/v1/docs>

## ONOS Core REST API

ONOS Core REST API

|                                                         |           |                 |                   |
|---------------------------------------------------------|-----------|-----------------|-------------------|
| docs : REST API documentation                           | Show/Hide | List Operations | Expand Operations |
| applications : Manage inventory of applications         | Show/Hide | List Operations | Expand Operations |
| cluster : Manage cluster of ONOS instances              | Show/Hide | List Operations | Expand Operations |
| configuration : Manage component configurations         | Show/Hide | List Operations | Expand Operations |
| keys : Query and Manage Device Keys                     | Show/Hide | List Operations | Expand Operations |
| devices : Manage inventory of infrastructure devices    | Show/Hide | List Operations | Expand Operations |
| diagnostics : Provides stream of diagnostic information | Show/Hide | List Operations | Expand Operations |
| nextobjectives : Get Flow objective next list           | Show/Hide | List Operations | Expand Operations |
| flowobjectives : Manage flow objectives                 | Show/Hide | List Operations | Expand Operations |
| <b>flows</b> : Query and program flow rules             | Show/Hide | List Operations | Expand Operations |
| groups : Query and program group rules                  | Show/Hide | List Operations | Expand Operations |
| hosts : Manage inventory of end-station hosts           | Show/Hide | List Operations | Expand Operations |



# Using Web GUI to Install Flow Rule (1/2)

- Fill out required fields ("appld" could be arbitrary string)

Select

[POST] /flows/{deviceId}

- Corresponding curl command

```
$ curl -u onos:rocks \
-X POST \
-H 'Content-Type: application/json' \
-d @flows1.json \
'http://localhost:8181/onos/v1/flows/of:0000000000000001'
```

Type In

deviceId  
appld  
JSON file

flows : Query and program flow rules

| Method | URL                        | Description                                 |
|--------|----------------------------|---------------------------------------------|
| GET    | /flows/table/{tableId}     | Gets all flow entries for a table           |
| DELETE | /flows/application/{appld} | Removes flow rules by application ID        |
| GET    | /flows/application/{appld} | Gets flow rules generated by an application |
| DELETE | /flows                     | Removes a batch of flow rules               |
| GET    | /flows                     | Gets all flow entries                       |
| POST   | /flows                     | Creates new flow rules                      |
| DELETE | /flows/{deviceId}/{flowId} | Removes flow rule                           |
| GET    | /flows/{deviceId}/{flowId} | Gets flow rules                             |
| GET    | /flows/pending             | Gets all pending flow entries               |
| GET    | /flows/{deviceId}          | Gets flow entries of a device               |
| POST   | /flows/{deviceId}          | Creates new flow rule                       |

Implementation Notes

Creates and installs a new flow rule for the specified device.  
Flow rule criteria and instruction description: <https://wiki.onosproject.org/display/ONOS/Flow+Rules>

Parameters

| Parameter | Value                                                                                                                                                                      | Description            | Parameter Type | Data Type |               |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|----------------|-----------|---------------|
| deviceId  | of:0000000000000001                                                                                                                                                        | device identifier      | path           | string    |               |
| appId     | app                                                                                                                                                                        | application identifier | query          | string    |               |
| stream    | {<br>"priority": 40000,<br>"timeout": 0,<br>"isPermanent": true,<br>"deviceId":<br>"of:0000000000000001",<br>"treatment": {<br>"instructions": [<br>{<br>"type": "OUTPUT", | flow rule JSON         | body           | Model     | Example Value |

Parameter content type: application/json



# Using Web GUI to Install Flow Rule (2/2)

- Click “Try it out!”
  - Web will pass the JSON stream to ONOS
  - Status code 201 represents HTTP Request is granted

appId: app

stream: {"type": "IN\_PORT", "port": "1"}

Parameter content type: application/json

Response Messages

| HTTP Status Code | Reason               | Response Model | Headers |
|------------------|----------------------|----------------|---------|
| 200              | successful operation |                |         |

default: Unexpected error

**Try it out!** [Hide Response](#)

**Click “Try it out!”**

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
 "priority": 40000, \
 "timeout": 0, \
 "isPermanent": true, \
 "deviceId": "of:0000000000000001", \
 "treatment": { \
 "instructions": [\
 { \
 "type": "OUTPUT", \
 "port": "2" \
 } \
] \
 }, \
 "selector": { \
 "criteria": [\
 { \
 "type": "IN_PORT", \
 "port": "1" \
 } \
] \
 } \
}' http://127.0.0.1:8181/onos/v1/flows/of%3A0000000000000001?appId=app'
```

HTTP response replied by ONOS

Response Body

no content

Response Code

**201 status code**

Response Headers

```
{
 "content-length": "0",
 "location": "http://127.0.0.1:8181/onos/v1/flows/of:0000000000",
 "server": "Jetty(9.4.18.v20190429)",
 "content-type": null
}
```

- In case of “curl”, use “-i” option to include HTTP Response header in the output



# Delete Flow Rule via ONOS Web GUI (1/2)

## Response Code

201

## Response Headers

```
{
 "content-length": "0",
 "location": "http://localhost:8181/onos/v1/flows/of:000000000000000001/49821071268378023",
 "server": "Jetty(9.4.28.v20200408)",
 "content-type": null
}
```

**flowId**



# Delete Flow Rule via ONOS Web GUI (2/2)

- Same procedure as installing flow rules

Select

**[DELETE]** /flows/{deviceId}/{flowId}



**flows** : Query and program flow rules Show/Hide List Operations Expand Operations

|        |                            |                                             |
|--------|----------------------------|---------------------------------------------|
| GET    | /flows/table/{tableId}     | Gets all flow entries for a table           |
| DELETE | /flows/application/{appId} | Removes flow rules by application ID        |
| GET    | /flows/application/{appId} | Gets flow rules generated by an application |
| DELETE | /flows                     | Removes a batch of flow rules               |
| GET    | /flows                     | Gets all flow entries                       |
| POST   | /flows                     | Creates new flow rules                      |
| DELETE | /flows/{deviceId}/{flowId} | Removes flow rule                           |
| GET    | /flows/{deviceId}/{flowId} | Gets flow rules                             |
| GET    | /flows/pending             | Gets all pending flow entries               |
| GET    | /flows/{deviceId}          | Gets flow entries of a device               |
| POST   | /flows/{deviceId}          | Creates new flow rule                       |

**Implementation Notes**  
Creates and installs a new flow rule for the specified device.  
Flow rule criteria and instruction description: <https://wiki.onosproject.org/display/ONOS/Flow+Rules>

**Parameters**

| Parameter | Value                                                                                                                           | Description            | Parameter Type | Data Type |
|-----------|---------------------------------------------------------------------------------------------------------------------------------|------------------------|----------------|-----------|
| deviceId  | of:0000000000000001                                                                                                             | device identifier      | path           | string    |
| appId     | app                                                                                                                             | application identifier | query          | string    |
| stream    | <pre>{   "priority": 40000,   "timeout": 0,   "isPermanent": true,   "deviceId": "of:0000000000000001",   "treatment": { </pre> | flow rule JSON         | body           | Model     |

Parameter content type: application/json

**Example Value**

```
{
 "priority": 40000,
 "timeout": 0,
 "isPermanent": true,
 "deviceId": "of:0000000000000001",
 "treatment": {
 "instructions": [
 {
 "type": "OUTPUT",

```



# Outline

- OpenFlow Messages
- Install/ Delete Flow Rules
- Lab 2 Requirements
  - Part 1: Answer Questions (15%)
  - Part 2: Install Flow Rules (15%)
  - Part 3: Create Broadcast Storm (20%)
  - Part 4: Trace ReactiveForwarding (10%)
  - Project 2 Demo(40%)



# Part 1: Answer Questions (1/2)

## ● Preparation:

please refer to the commands on p.8

1. Start capturing packets on the loopback interface (lo) with Wireshark.
2. Create a default topology.
3. Activate “org.onosproject.fwd”.
4. Execute command “h1 ping h2 -c 5” in Mininet CLI.
5. Observe the flow rules shown in the GUI.
6. Exit Mininet and stop capturing packets **when the forwarding rules disappear**.

## ● Questions:

1. How many **OpenFlow headers** with type “OFPT\_FLOW\_MOD” and command “OFPFC\_ADD” are there among all the packets?
2. What are **the match fields** and the corresponding **actions** in each “OFPT\_FLOW\_MOD” message?
3. What are the **Idle Timeout values** for all flow rules on s1 in **GUI**?

Report format

Ex: There are x distinct “OFPT\_FLOW\_MOD” headers during the experiment.

| Match fields | actions       | Timeout values |
|--------------|---------------|----------------|
| IN_PORT=1    | Output port=4 | 0              |
| .....        |               |                |



# Part 1: Answer Questions (2/2)

## ● Hints

- A single OpenFlow packet may contain multiple OpenFlow message headers
- Only count the number of **distinct** OpenFlow **headers**
  - If match fields of two headers are the same, just count once
- Value of timeout can be zero
- There will be an `Unknown` type, please refer to ONOS's [Source Code](#) to find what it is

```
▼ OpenFlow 1.4
 Version: 1.4 (0x05)
 Type: OFPT_FLOW_MOD (14)
 Length: 96
 Transaction ID: 7
 Cookie: 0x00010000021b41dc
 Cookie mask: 0x0000000000000000
 Table ID: 0
 Command: OFPFC_ADD (0)
 Idle timeout: 0
 Hard timeout: 0
 Priority: 5
 Buffer ID: OFP_NO_BUFFER (4294967295)
 Out port: OFPP_ANY (4294967295)
 Out group: OFPG_ANY (4294967295)
 ▶ Flags: 0x0001
 Importance: 0
 ▼ Match
 Type: OFPMT_OXM (1)
 Length: 10
 ▼ OXM field
 Class: OFPXMC_OPENFLOW_BASIC (0x8000)
 0000 101. = Field: OFPXMT_OFB_ETH_TYPE (5)
 0 = Has mask: False
 Length: 2
 Value: IPv4 (0x0800)
 Pad: 000000000000
```

```
Type: OFPT_FLOW_MOD (14)
Length: 96
Transaction ID: 7
Cookie: 0x00010000021b41dc
Cookie mask: 0x0000000000000000
Table ID: 0
Command: OFPFC_ADD (0)
```

```
▼ Match
 Type: OFPMT_OXM (1)
 Length: 10
 ▼ OXM field
 Class: OFPXMC_OPENFLOW_BASIC (0x8000)
 0000 101. = Field: OFPXMT_OFB_ETH_TYPE (5)
 0 = Has mask: False
 Length: 2
 Value: IPv4 (0x0800)
 Pad: 000000000000
```



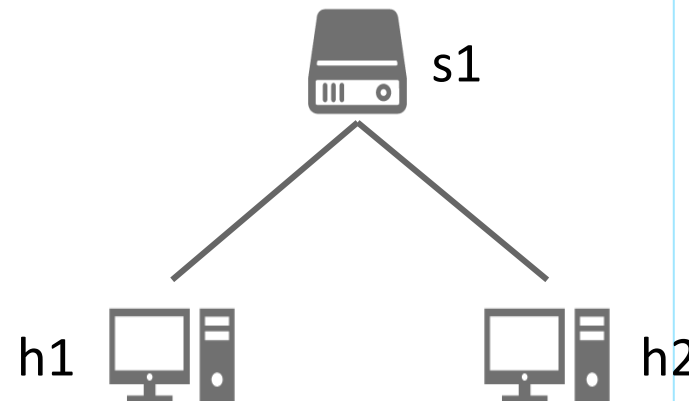


## Part 2: Install Flow Rules (1/3)

- Please deactivate all the Apps, **except the following Apps** initially activated:

- “org.onosproject.hostprovider”
- “org.onosproject.lldpprovider”
- “org.onosproject.optical-model”
- “org.onosproject.openflow-base”
- “org.onosproject.openflow”
- “org.onosproject.drivers”
- “org.onosproject.gui2”

- Use the following topology (i.e. h1-s1-h2):



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



## Part 2: Install Flow Rules (2/3)

1. Install **one** flow rule to forward ARP packets
  - Match Fields
    - Ethernet type (ARP)
  - Actions
    - Forwarding ARP packets to all port **in one instruction**
- Take **screenshot** to verify the flow rule you installed

```
mininet> h1 arping h2 # send ARP request
```

```
mininet> h1 arping h2
ARPING 10.0.0.2
42 bytes from 5e:c0:96:3f:8e:ab (10.0.0.2): index=0 time=4.134 msec
42 bytes from 5e:c0:96:3f:8e:ab (10.0.0.2): index=1 time=6.226 usec
42 bytes from 5e:c0:96:3f:8e:ab (10.0.0.2): index=2 time=5.839 usec
42 bytes from 5e:c0:96:3f:8e:ab (10.0.0.2): index=3 time=3.860 usec
42 bytes from 5e:c0:96:3f:8e:ab (10.0.0.2): index=4 time=3.991 usec
42 bytes from 5e:c0:96:3f:8e:ab (10.0.0.2): index=5 time=7.219 usec
```



## Part 2: Install Flow Rules (3/3)

2. Install **two** flow rules to forward IPv4 packets
  - Match Fields
    - IPv4 destination address and other required dependencies
  - Actions
    - Forwarding IPv4 packets to the right host
- Take screenshot to verify the flow rules you installed

```
mininet> h1 ping h2 # send ICMP request
```

```
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=0.339 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.056 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.057 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.064 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.055 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.054 ms
```

*Hint: Match fields may have dependency; please refer to OpenFlow spec v1.4.0.*



## Part 3: Create Topology with Broadcast Storm

- Steps:

- Create a topology that may cause a “Broadcast Storm”.
- Install flow rules on switches.
- Send packets from one host to another host.
- Observe link status of the network and the CPUs utilization of VM

- **Do NOT** activate any other APPs, except for those initially activated by ONOS
- Describe what you have observed and explain why the broadcast storm occurred
- **Take screenshot** of CPU’s utilization
- Hand in Topology file (\*.py) and flow rule files (\*.json)

Hint: ONOS would initially install several flow rules.



## Part 4: Trace ReactiveForwarding

- Activate only “org.onosproject.fwd” and other initially activated APPs.
- Use Mininet default topology and let h1 ping h2.
- Observe what happens in control and data planes
  - From the time when h1 pings h2 until h2 receives the first ICMP request
  - Write down every operation made by control and data planes
  - Please refer to the ONOS ReactiveForwarding application
  - [Source Code](#)
- Describe what you observed step by step in report
- **Don't** need to take screenshot

Hint: Tracing Source code with Wireshark to finish this Part



# Submission (1/2)

- Files:
  - A report: **lab2\_<studentID>.pdf**
    1. Part 1: Answers to those three questions in **p.31** format
    2. Part 2: Take screenshots of arping/ping result
    3. Part 3: Take screenshots and answer the question
    4. Part 4: Write down what you have observed step by step
    5. What you've learned or solved
  - JSON files for installing flow rules in part 2 and part 3
    - Please follow naming convention
  - A Python script for creating topology in part 3



# Naming Convention

- Use the following convention to name the files created in both part 2 and part 3.
  1. Python script for the topology: `topo_<studentID>.py`
  2. JSON files for flow rules: `flows_s<i>-<j>_<studentID>.json`
    - “i” is the switch number
    - “j” is the flow rule number, starting from 1, on a switch.

e.g.

| File Name                            | Meaning                       |
|--------------------------------------|-------------------------------|
| <code>flows_s1-1_0748787.json</code> | #1 flow rule to install on s1 |
| <code>flows_s1-2_0748787.json</code> | #2 flow rule to install on s1 |
| <code>flows_s2-1_0748787.json</code> | #1 flow rule to install on s2 |



## Submission (2/2)

- Directory structure:
  - Create root folder: **lab2\_<studentID>**
  - In root folder, create part2 and part3 folders and move files (i.e. \*.json, \*.py) into the corresponding folders

e.g.

```
project2_0748787/
├── part2
│ ├── flows_s1-1_0748787.json
│ ├── flows_s1-2_0748787.json
│ └── flows_s1-3_0748787.json
├── part3
│ ├── flows_s1-1_0748787.json
│ ├── flows_s1-2_0748787.json
│ ├── flows_s2-1_0748787.json
│ ├── flows_s3-1_0748787.json
│ └── topo_0748787.py
└── project2_0748787.pdf
```

- Zip root folder: **lab2\_<studentID>.zip**
- Wrong file name or format will result in 10 points deduction





## Lab 2 Demo

- The demo dates will be in the week after lab 2 deadline.
- Demo question will show when demo start.
- The questions involve modification of the code and the content related to the lecture and the lab



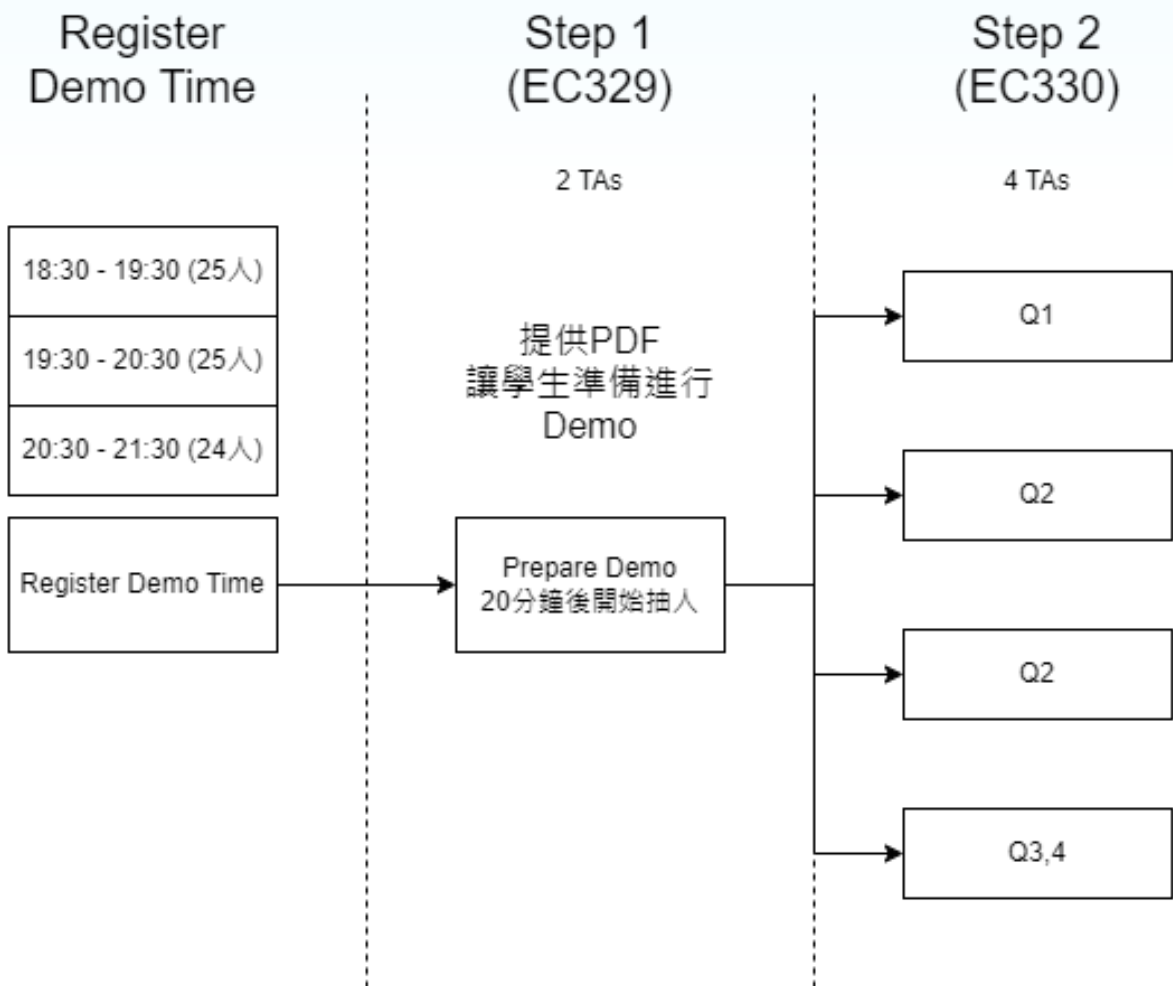
# Lab Assign, Due, and Demo Time

|               |                                                                         | Assign Time | Due Time             | Demo Time                    |
|---------------|-------------------------------------------------------------------------|-------------|----------------------|------------------------------|
| Lab 1         | ONOS and Mininet Installation                                           | 09/05       | 09/25 (WED)<br>23:59 | -                            |
| Lab 2         | OpenFlow Protocol Observation & Flow Rule installation                  | 09/12       | 09/25 (WED)<br>23:59 | 09/30 (MON)<br>18:30 – 21:30 |
| Lab 3         | ONOS Application Development: SDN-enabled Learning Bridge and ARP Proxy | 09/26       | 10/16 (WED)<br>23:59 | 10/21 (MON)<br>18:30 – 21:30 |
| Lab 4         | Meter Table and Bandwidth Control                                       | 10/17       | 11/06 (WED)<br>23:59 | 11/11 (MON)<br>18:30 – 21:30 |
| Lab 5         | Network Function Virtualization: Software Router and Containerization   | 11/07       | 11/27 (WED)<br>23:59 | 12/02 (MON)<br>18:30 – 21:30 |
| Final Project | SDN network as vRouter (Group project)                                  | 11/28       | 待定                   | 待定                           |



# Lab 2 Demo Flow Chart

## ● Lab 2 Demo Flow Chart





# Lab 2 Demo Time-reserved Table

- Lab 2 Demo Time-reserved Table
  - URL : <https://anurl.app/xhDBIb>
  - Date of completion: now ~ 9/25 (WED) 23:59

| Lab 2 Demo 時段登記          |    |    |    |    |    |    |    |    |    |    |
|--------------------------|----|----|----|----|----|----|----|----|----|----|
| 日期: 2024/09/30           | 學號 | 姓名 | 學號 | 姓名 | 學號 | 姓名 | 學號 | 姓名 | 學號 | 姓名 |
| 時段 1<br>18:30 - 19:30    |    |    |    |    |    |    |    |    |    |    |
|                          |    |    |    |    |    |    |    |    |    |    |
|                          |    |    |    |    |    |    |    |    |    |    |
| 時段 2<br>19:30 - 20:30    |    |    |    |    |    |    |    |    |    |    |
|                          |    |    |    |    |    |    |    |    |    |    |
|                          |    |    |    |    |    |    |    |    |    |    |
| 時段 3<br>20:30 - 21:30    |    |    |    |    |    |    |    |    |    |    |
|                          |    |    |    |    |    |    |    |    |    |    |
|                          |    |    |    |    |    |    |    |    |    |    |
| Ex: 313551001    Ex: 王小明 |    |    |    |    |    |    |    |    |    |    |



[Lab 2 Demo Time-reserved Table](https://anurl.app/xhDBIb)



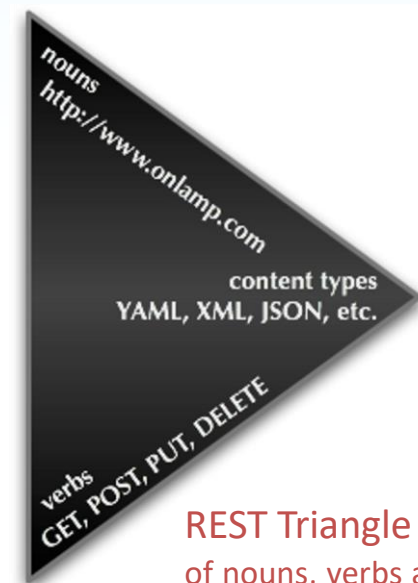
# References

- OpenFlow spec v1.4.0
  - <https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf>
- Wireshark wiki
  - <https://wiki.wireshark.org/Home>
- ONOS REST API
  - <https://wiki.onosproject.org/display/ONOS/Appendix+B%3A+REST+API>
- JSON Format for Installing Flow Rules
  - <https://wiki.onosproject.org/display/ONOS/Flow+Rules>



# Appendix—REST (REpresentational State Transfer)

- REST is a **software architectural style** for creating Web services
- Architectural constraints:
  - Client-server architecture
  - Stateless
  - Cacheable
  - Uniform interface
  - Layered system
- Allow us to access and manipulate web resources
  - Commonly we use HTTP method
    - Payload could be formatted in HTML, XML, JSON



REST Triangle  
of nouns, verbs and content types

Source: Soul & Shell Blog



# 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](mailto: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 hours**



# Q & A