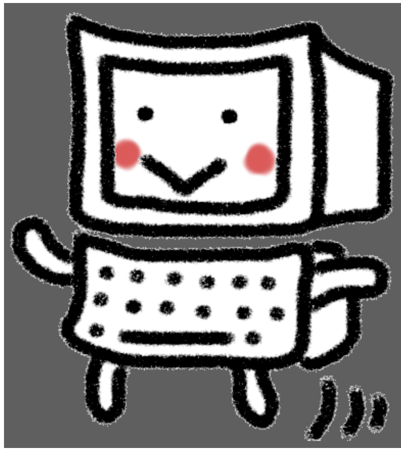


# Project 2

*OpenFlow Protocol Observation &*
*Flow Rule Installation*

Date : 2020/03/16 (MON) 18:30

Deadline: 2020/04/05 (SUN) 23:59

# Outline

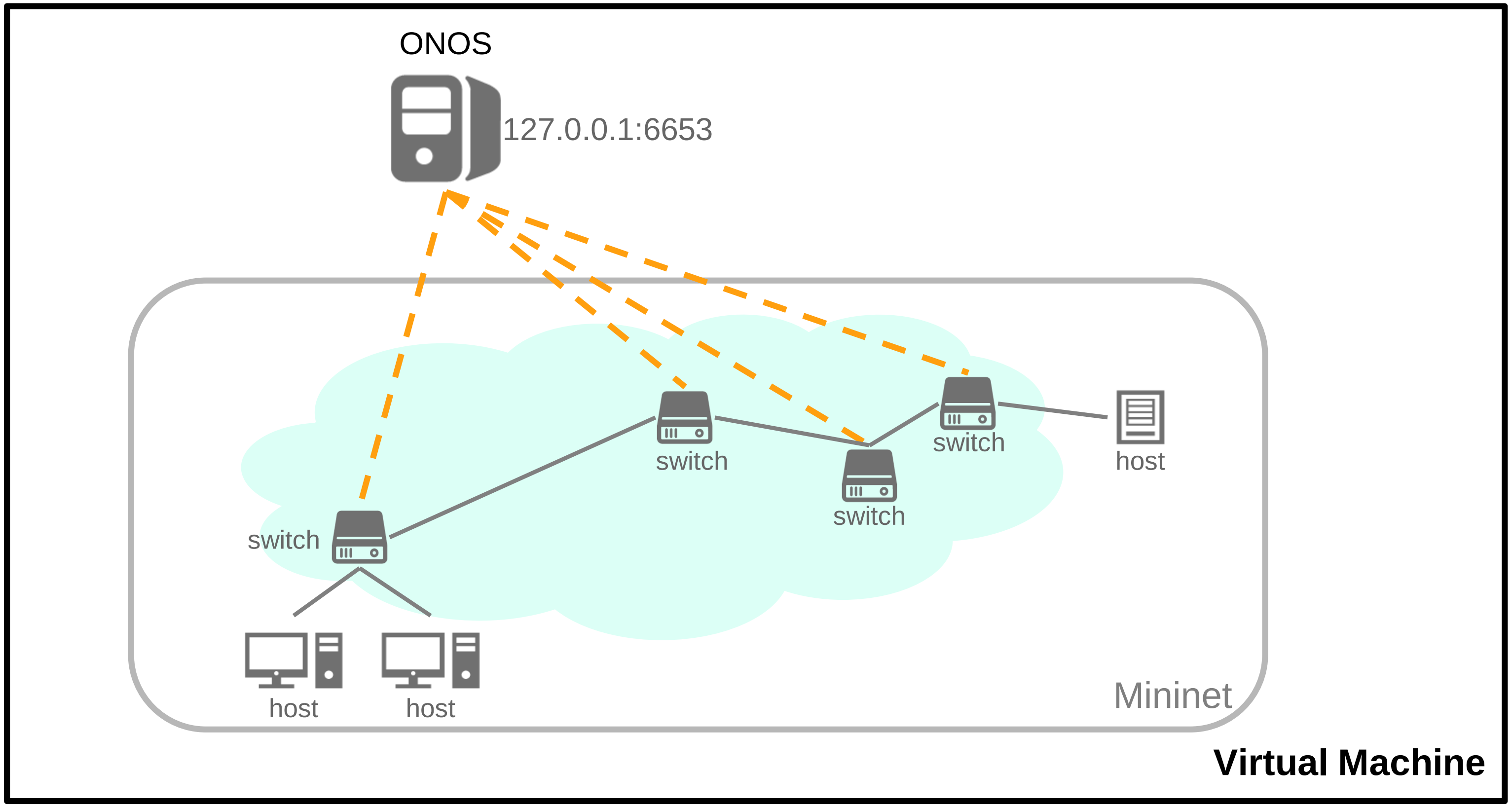
- ❑ **Observe OpenFlow Messages**
  - ■ Monitor traffic between ONOS & Switches
  - ■ OpenFlow Message Observation
- ❑ **Install/ Delete Flow Rules**
  - ■ REST & curl
  - ■ ONOS & Topology Setup
  - ■ [Method 1] Via Command “curl”
  - ■ [Method 2] Via ONOS Web GUI
- ❑ **Project 2 Requirements**
  - ■ Answer Questions (30%)
  - ■ Install Flow rules (40%)
  - ■ Create Topology with Broadcast Storm(30%)
  - ■ Bonus

# Outline

- **Observe OpenFlow Messages**
  - Monitor traffic between ONOS & Switches
  - OpenFlow Message Observation
- **Install/ Delete Flow Rules**
  - REST & curl
  - ONOS & Topology Setup
  - [Method 1] Via Command “curl”
  - [Method 2] Via ONOS Web GUI
- **Project 2 Requirements**
  - Answer Questions (30%)
  - Install Flow rules (40%)
  - Create Topology with Broadcast Storm(30%)
  - Bonus

# Capturing Openflow Messages

- How to capture and observe Openflow Messages?
  - Openflow messages exchanged between Controller and OVS
  - Install Wireshark on the VM running ONOS and Mininet



# Wireshark Installation

❑ Wireshark

- An open-source and widely-used network packet analyzer

❑ Wireshark Installation

- Update package info first
  - Use apt (i.e. Advanced Packaging Tools) command in Ubuntu

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

- Then install Wireshark

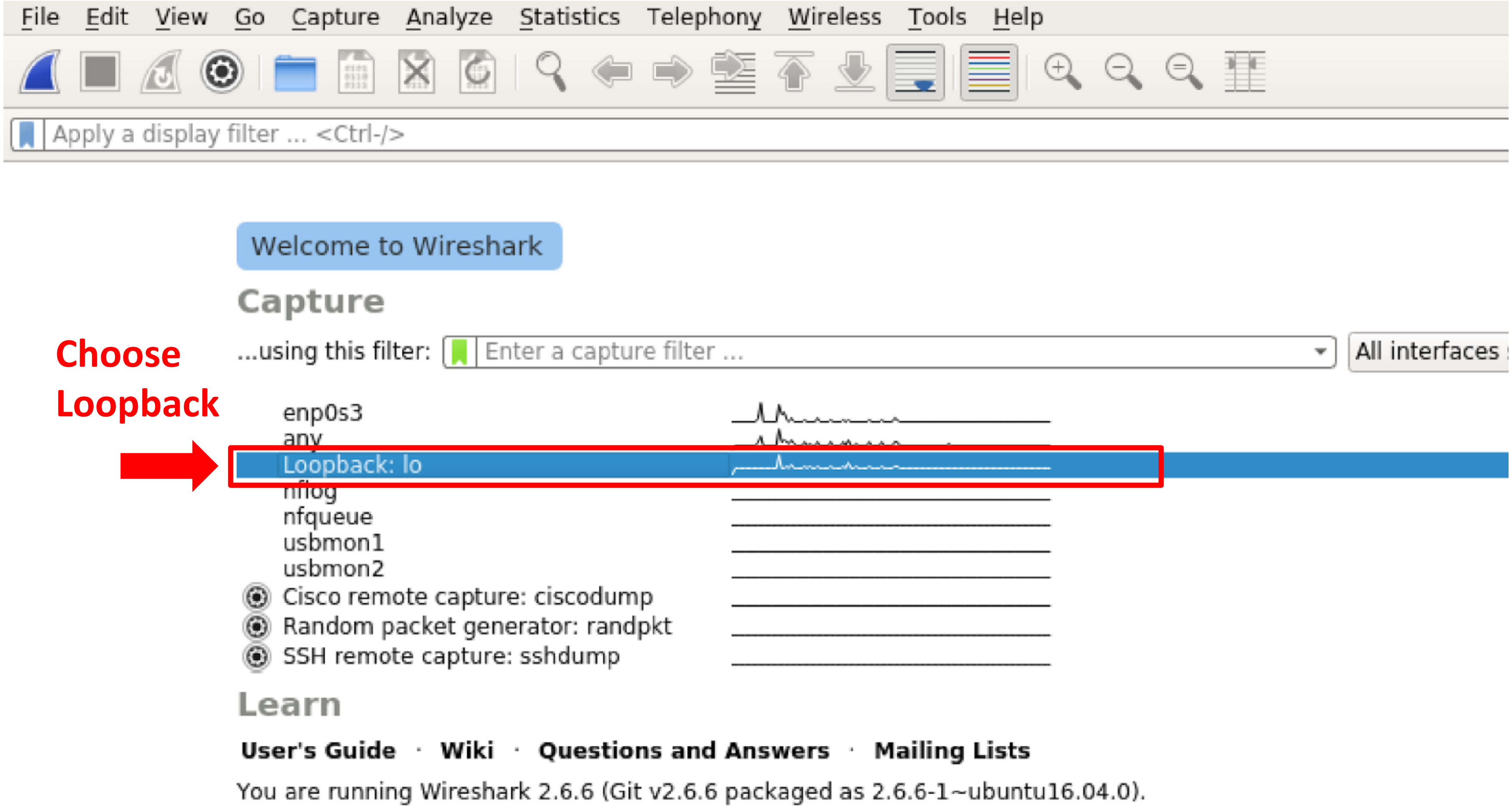
```
$ sudo apt install wireshark
```

❑ Start Wireshark

```
$ sudo wireshark           # start wireshark
```

# How to Capture Packets In Wireshark

- ❑ Both ONOS and Mininet runs on localhost of VM in our setup
- ❑ Capture packets on the Loopback (lo) interface

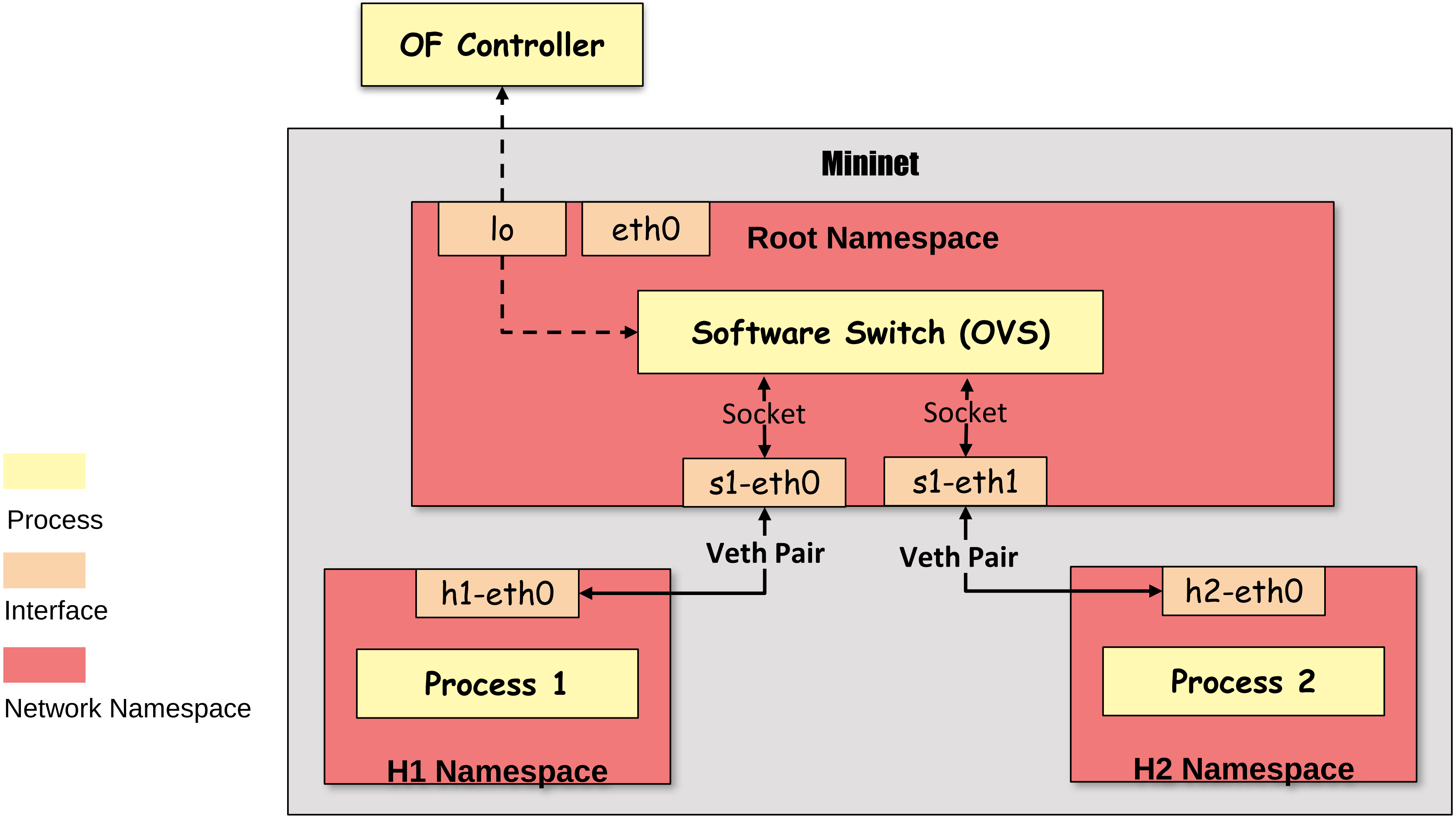


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces
enp0s3
any
Loopback: lo
nflog
nfqueue
usbmon1
usbmon2
Cisco remote capture: ciscodump
Random packet generator: randpkt
SSH remote capture: sshdump
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
You are running Wireshark 2.6.6 (Git v2.6.6 packaged as 2.6.6-1~ubuntu16.04.0).

Choose Loopback

# Mininet utilizes network namespace

- Mininet utilizes network namespace to emulate networks
  - OVS runs in the root network namespace



# Outline

- ❑ **Observe OpenFlow Messages**
  - Monitor traffic between ONOS & Switches
  - OpenFlow Message Observation
- **Install/ Delete Flow Rules**
  - REST & curl
  - ONOS & Topology Setup
  - [Method 1] Via Command "curl"
  - [Method 2] Via ONOS Web GUI
- **Project 2 Requirements**
  - Answer Questions (30%)
  - Install Flow rules (40%)
  - Create Topology with Broadcast Storm(30%)
  - Bonus

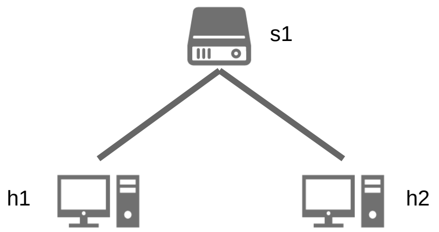
# Start ONOS and Capture OpenFlow Messages

1. Start ONOS
2. Activate ReactiveForwarding in ONOS CLI

```
onos> apps -a -s      # optionally check activated application
onos> app activate fwd      # activate ReactiveForwarding
```

3. Start Mininet with default (minimal) topology

```
$ sudo mn --controller=remote,127.0.0.1:6653
```



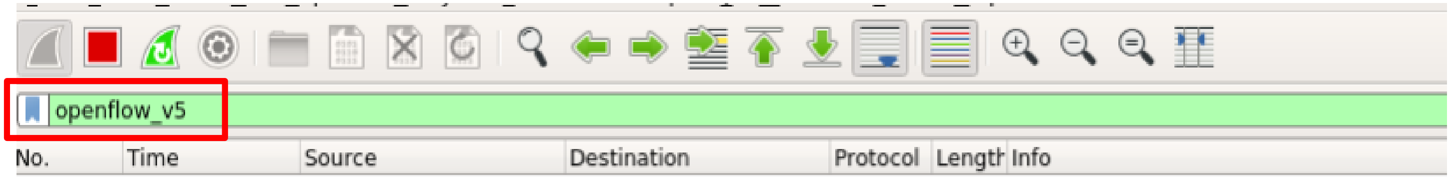
4. Ping a host in Mininet

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

5. Exit Mininet and stop capturing packets in Wireshark when ping terminates
6. Observe captured OpenFlow packets

# Filter Captured Packets

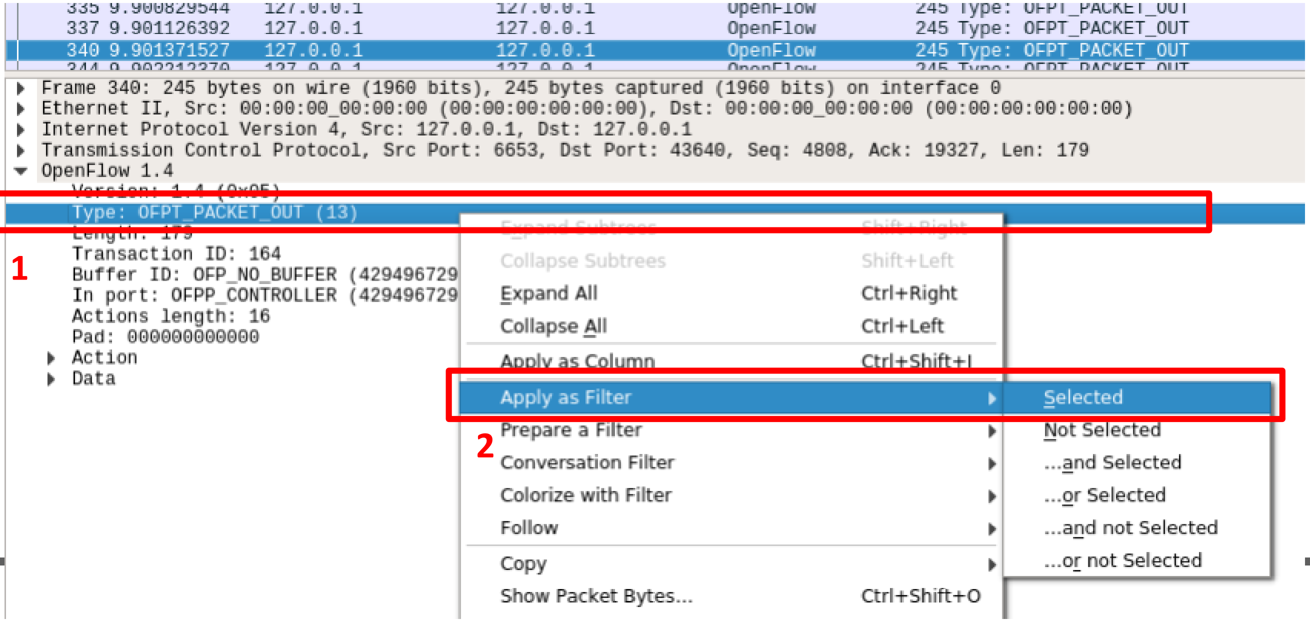
- Use keyword “openflow_v5” to filter **OpenFlow v1.4.0** packets
  - ONOS v2.2.0 uses Openflow v1.4.0



openflow_v5

No. Time Source Destination Protocol Length Info

- Alternatively, apply filter in the following steps:
  1. Right click on the packet header field which you want to apply as filter
  2. Choose “Apply as Filter” and click “Selected”
  3. Wireshark will immediately filter out all the relevant packets



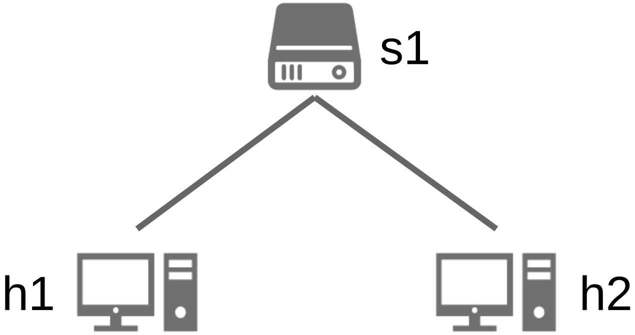
```
335 9.900829544   127.0.0.1        127.0.0.1        OpenFlow     245 Type: OFPT_PACKET_OUT
337 9.901126392   127.0.0.1        127.0.0.1        OpenFlow     245 Type: OFPT_PACKET_OUT
340 9.901371527   127.0.0.1        127.0.0.1        OpenFlow     245 Type: OFPT_PACKET_OUT
344 9.902212270   127.0.0.1        127.0.0.1        OpenFlow     245 Type: OFPT_PACKET_OUT
```

```
▸ 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: OFP_NO_BUFFER (429496729
    In port: OFPP_CONTROLLER (429496729
    Actions length: 16
    Pad: 000000000000
  ▸ Action
  ▸ Data
```

# Minimal Topology in Mininet

- Command "mn" builds a default topology with a switch and two hosts connected

```
$ sudo mn --controller=remote,127.0.0.1:6653
```



- Manpage for command “mn”
    - http://manpages.ubuntu.com/manpages/bionic/man1/mn.1.html

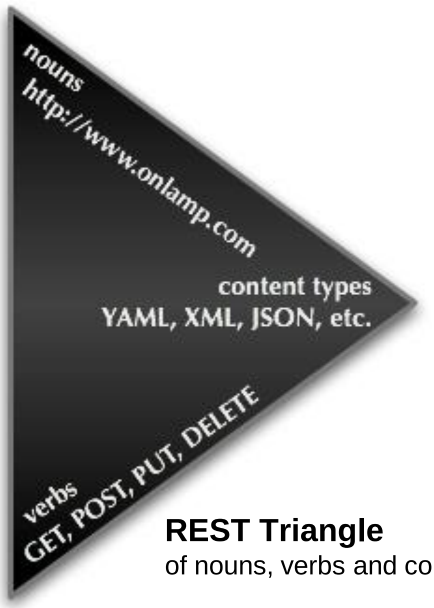
# Outline

- ❑ Observe OpenFlow Messages
  - ■ Monitor traffic between ONOS & Switches
  - ■ OpenFlow Message Observation
- ❑ **Install/ Delete Flow Rules**
  - ■ REST & curl
  - ■ ONOS & Topology Setup
  - ■ [Method 1] Via Command “curl”
  - ■ [Method 2] Via ONOS Web GUI
- ■ **Project 2 Requirements**
  - ■ Answer Questions (30%)
  - ■ Install Flow rules (40%)
  - ■ Create Topology with Broadcast Storm(30%)
  - ■ Bonus

# 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



**REST Triangle**
of nouns, verbs and content types

*Source: Soul & Shell Blog*

- Allow us to access and manipulate web resources
  - Commonly we use HTTP method
    - Payload could be formatted in HTML, XML, JSON

# Curl – Command Tool For Transferring Data

- ❑ Format of "curl"

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

- ❑ Transferring data with URL

```
$ curl –u <user:password> -X <command> -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 prefixed with `@`

```
$ curl –u <user:password> -X <command> -H <header> -d @<file> [URL...]
```

- ❑ Manpage for Linux command "curl"
  - http://manpages.ubuntu.com/manpages/xenial/man1/curl.1.html

# Outline

- ❑ **Observe OpenFlow Messages**
  - ■ Monitor traffic between ONOS & Switches
  - ■ OpenFlow Message Observation
- ❑ **Install/ Delete Flow Rules**
  - ■ REST & curl
  - ■ ONOS & Topology Setup
  - ■ [Method 1] Via Command "curl"
  - ■ [Method 2] Via ONOS Web GUI
- ■ **Project 2 Requirements**
  - ■ Answer Questions (30%)
  - ■ Install Flow rules (40%)
  - ■ Create Topology with Broadcast Storm(30%)
  - ■ Bonus

# ONOS & Topology Setup

❑ Restart ONOS

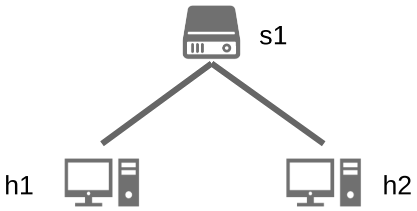
1. <ctrl+c> in the ONOS log panel to shutdown ONOS server
2. 
```
demo@SDN-NFV:~/onos$ ok clean
    # ok is an alias of command “bazel run onos-local -- ”
```

❑ Deactivate Reactiveforwarding APP

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

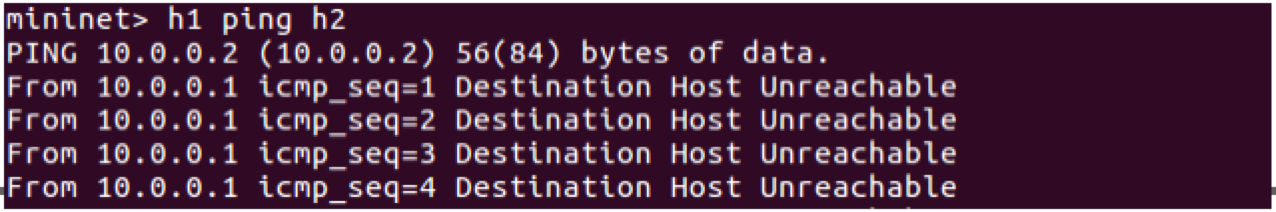
❑ Start Mininet with default topology (minimal)

```
$ sudo mn --controller=remote,127.0.0.1:6653
```



❑ Check 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

- ❑ **Observe OpenFlow Messages**
  - ■ Monitor traffic between ONOS & Switches
  - ■ OpenFlow Message Observation
- ❑ **Install/ Delete Flow Rules**
  - ■ REST & curl
  - ■ ONOS & Topology Setup
  - ■ [Method 1] Via Command “curl”
  - ■ [Method 2] Via ONOS Web GUI
- ■ **Project 2 Requirements**
  - ■ Answer Questions (30%)
  - ■ Install Flow rules (40%)
  - ■ Create Topology with Broadcast Storm(30%)
  - ■ Bonus

# Create a JSON file of flow rules

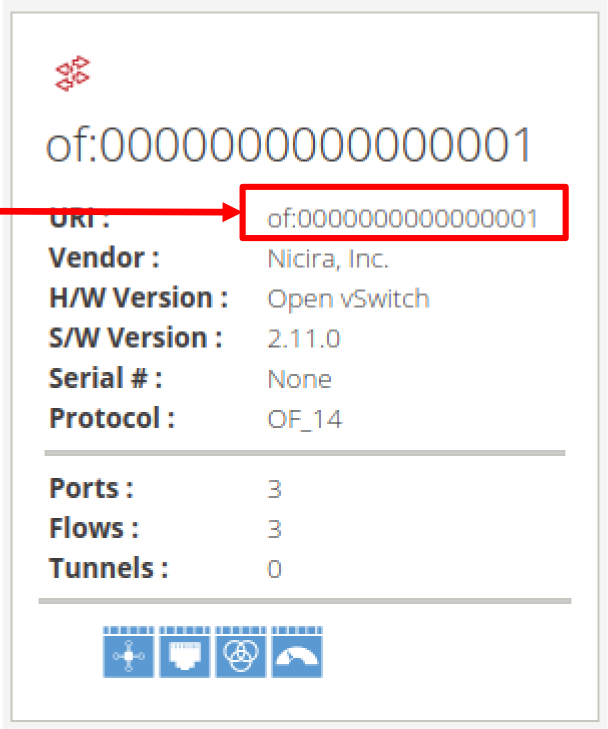
```
{
   "priority": 50000,
   "timeout": 0,
   "isPermanent": true,
   "deviceId": "of:0000000000000001",
   "treatment": {
     "instructions": [
       {
         "type": "OUTPUT",
         "port": "2"
       }
     ]
   },
   "selector": {
     "criteria": [
       {
         "type": "IN_PORT",
         "port": "1"
       }
     ]
   }
 }
```

**flows1.json**

# JSON file: Device ID

```
{
   "priority": 50000,
   "timeout": 0,
   "isPermanent": true,
   "deviceId": "of:0000000000000001",
   "treatment": {
     "instructions": [
       {
         "type": "OUTPUT",
         "port": "2"
       }
     ]
   },
   "selector": {
     "criteria": [
       {
         "type": "IN_PORT",
         "port": "1"
       }
     ]
   }
 }
```

**flows1.json**



of:0000000000000001

**URI :** of:0000000000000001
**Vendor :** Nicira, Inc.
**H/W Version :** Open vSwitch
**S/W Version :** 2.11.0
**Serial # :** None
**Protocol :** OF_14

**Ports :** 3
**Flows :** 3
**Tunnels :** 0

- DeviceID **MUST be** the URI, shown in the ONOS web GUI of the target OF switch
- DeviceID is set by either ONOS or user specified topology file

# JSON File: Match Field of Flow Rule

```
{
   "priority": 50000,
   "timeout": 0,
   "isPermanent": true,
   "deviceId": "of:0000000000000001"
   "treatment": {
     "instructions": [
       {
          "type": "OUTPUT",
          "port": "2"
       }
     ]
   },
   "selector": {
     "criteria": [
       {
          "type": "IN_PORT",
          "port": "1"
       }
     ]
   }
}
```

**flows1.json**

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

# JSON File: Action Field of Flow Rule

```
{
   "priority": 50000,
   "timeout": 0,
   "isPermanent": true,
   "deviceId": "of:0000000000000001",
   "treatment": {
     "instructions": [
       {
         "type": "OUTPUT",
         "port": "2"
       }
     ]
   },
   "selector": {
     "criteria": [
       {
         "type": "IN_PORT",
         "port": "1"
       }
     ]
   }
 }
```

**flows1.json**

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

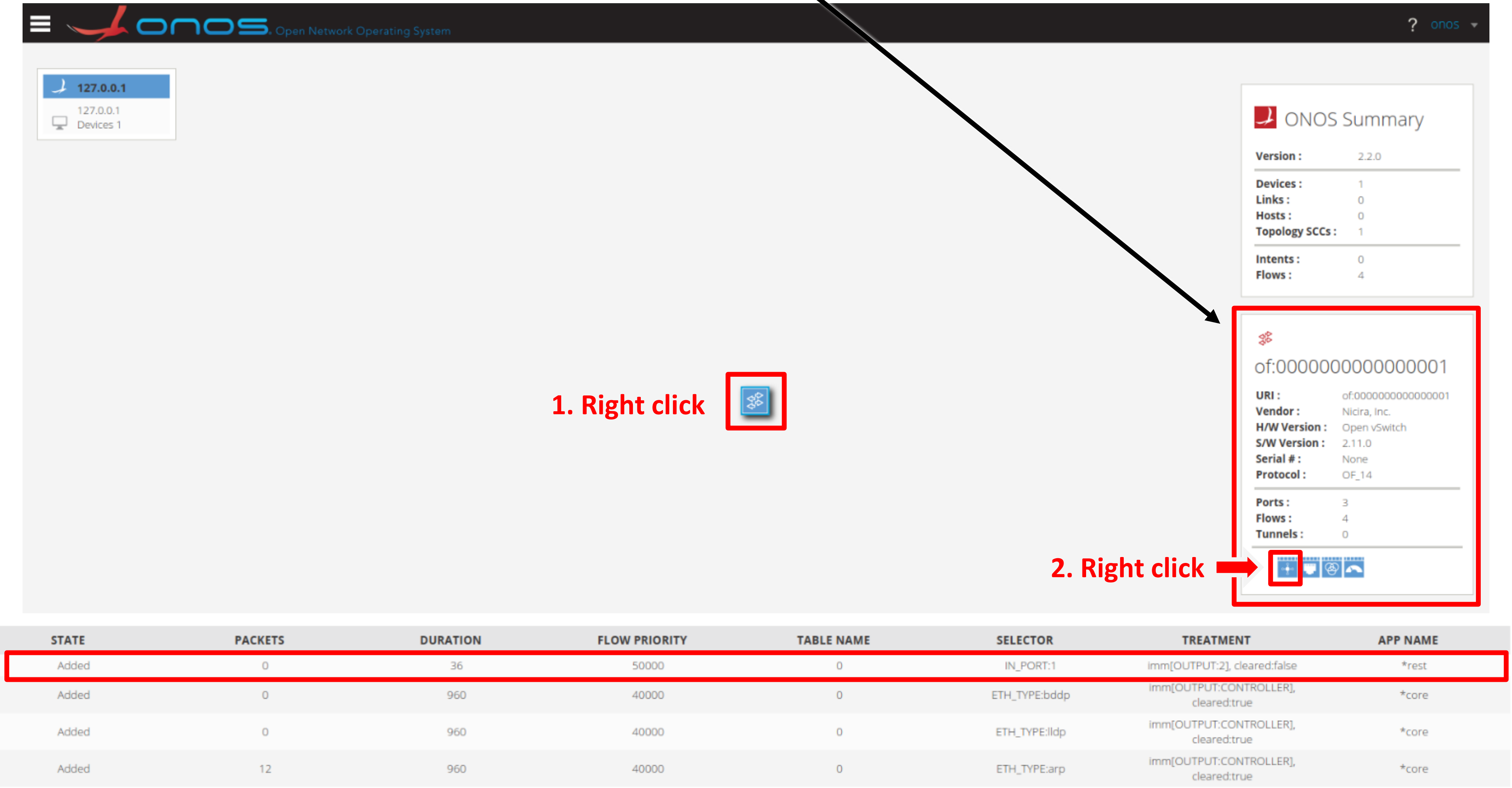
# Upload JSON File to ONOS

❑ Install flow rules on ONOS with JSON file say “flows1.json”

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

# Check whether the flow rule is installed 1/2

1. Left click on . Then, the panel of switch info will pop out
2. Left click on



ONOS Summary

Version : 2.2.0

Devices : 1
Links : 0
Hosts : 0
Topology SCCs : 1

Intents : 0
Flows : 4

of:0000000000000001

URI : of:0000000000000001
Vendor : Nicira, Inc.
H/W Version : Open vSwitch
S/W Version : 2.11.0
Serial # : None
Protocol : OF_14

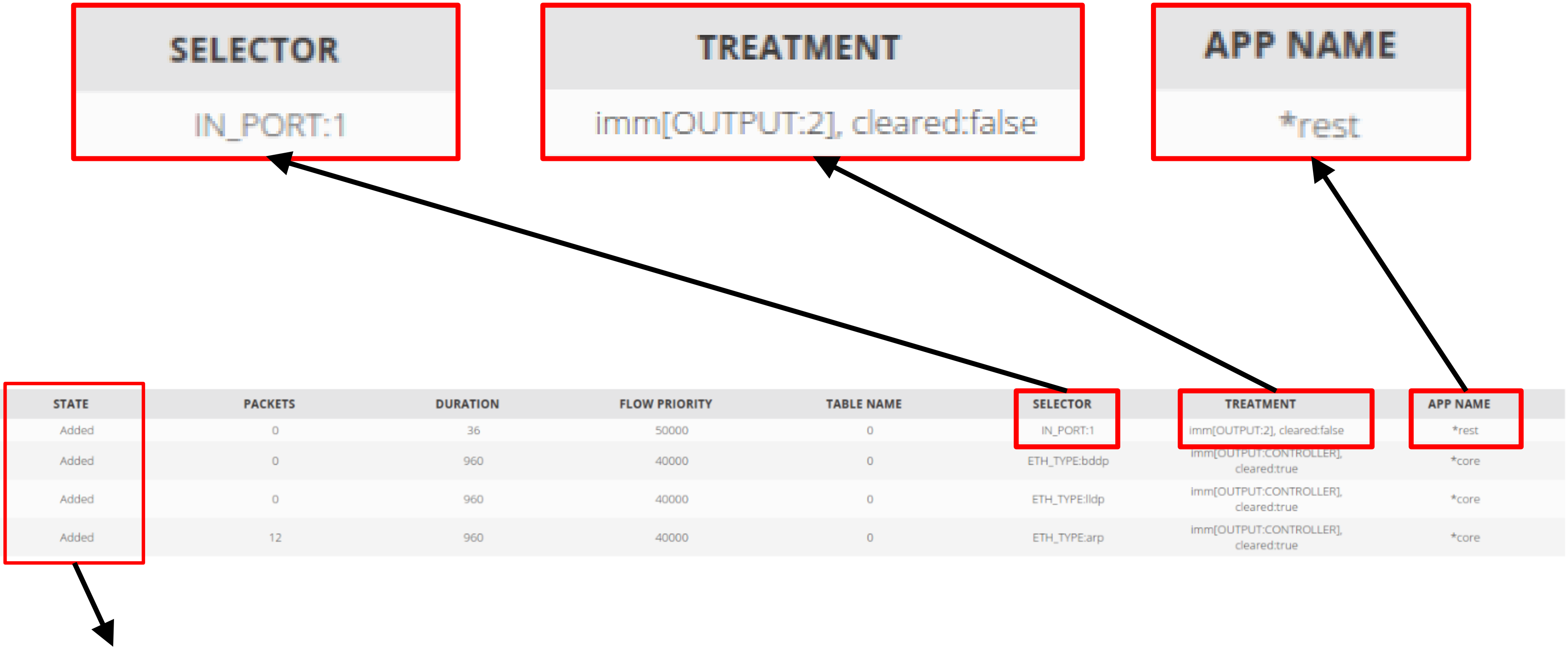
Ports : 3
Flows : 4
Tunnels : 0

1. Right click

2. Right click

| 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 |
|---|
| IN_PORT:1 |

| TREATMENT |
|---|
| imm[OUTPUT:2], cleared:false |

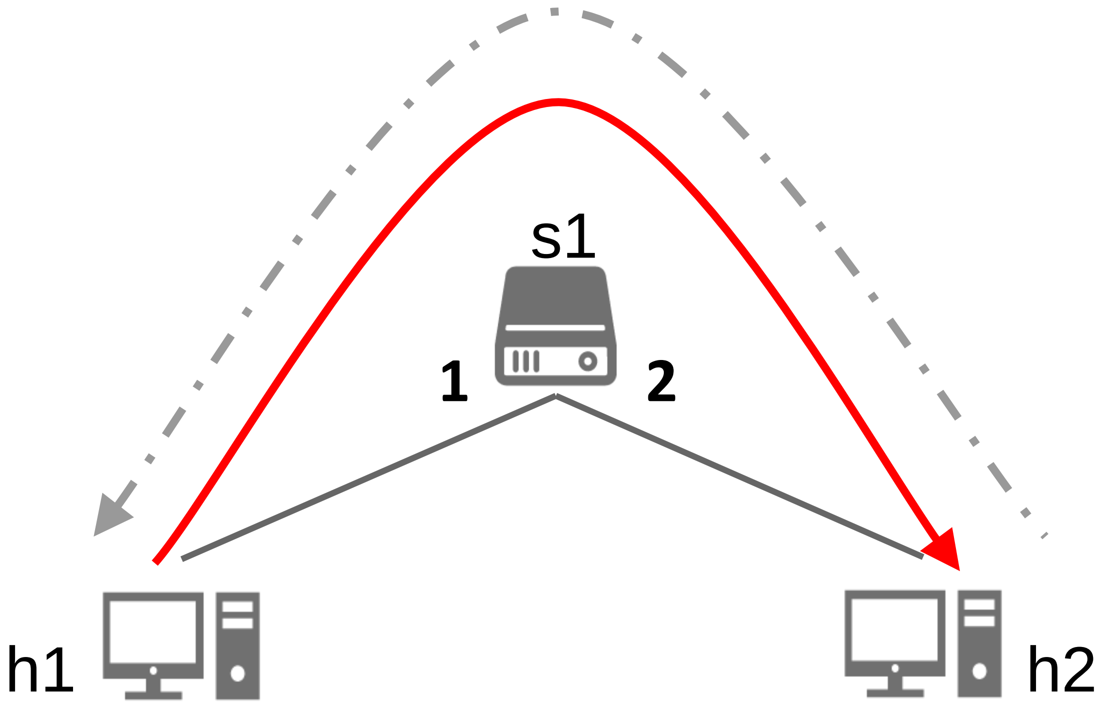
| APP NAME |
|---|
| *rest |

| 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 |

- Flow Rule States:
  - **PENDING_ADD** – this indicates that 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 transition to this state.

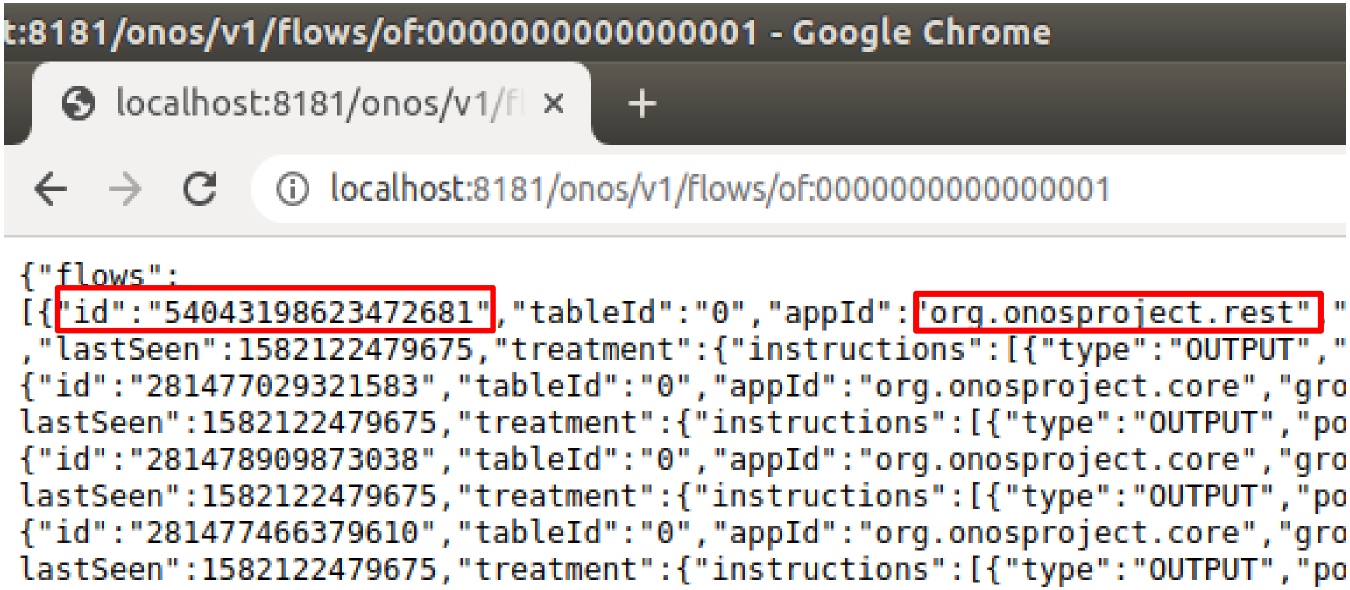
# Why Hosts Still Can’t Ping Each Other?

- Because we have installed flow rule for one direction only
  - S1 will forward packets from h1 to h2
  - But, s1 will not forward packets from h2 to h1
    - S1 drops a packet if the packet does not match any flow rule



# Delete Flow Rules – Find Flow ID

- ❑ Use URL to find the flowID on switch
  - ✔ Ex. http://localhost:8181/onos/v1/flows/of:0000000000000001


t:8181/onos/v1/flows/of:0000000000000001 - Google Chrome
localhost:8181/onos/v1/fl ×
+
localhost:8181/onos/v1/flows/of:0000000000000001

```
{"flows":
[{"id":"54043198623472681","tableId":"0","appId":"org.onosproject.rest","
,"lastSeen":1582122479675,"treatment":{"instructions":[{"type":"OUTPUT","
{"id":"281477029321583","tableId":"0","appId":"org.onosproject.core","gro
lastSeen":1582122479675,"treatment":{"instructions":[{"type":"OUTPUT","po
{"id":"281478909873038","tableId":"0","appId":"org.onosproject.core","gro
lastSeen":1582122479675,"treatment":{"instructions":[{"type":"OUTPUT","po
{"id":"281477466379610","tableId":"0","appId":"org.onosproject.core","gro
lastSeen":1582122479675,"treatment":{"instructions":[{"type":"OUTPUT","po
```

- ■ flowID of the flow we just added is **54043198623472681**

- ❑ 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:0000000000000001’
```

# Delete Flow Rules – Delete via curl

- Then, delete the flow rule with flowID **54043198623472681**

```
$ curl -u onos:rocks -X DELETE -H ‘Accept: application/json’ \
> ‘http://localhost:8181/onos/v1/flows/of:0000000000000001 \
> /54043198623472681’
```

Device ID

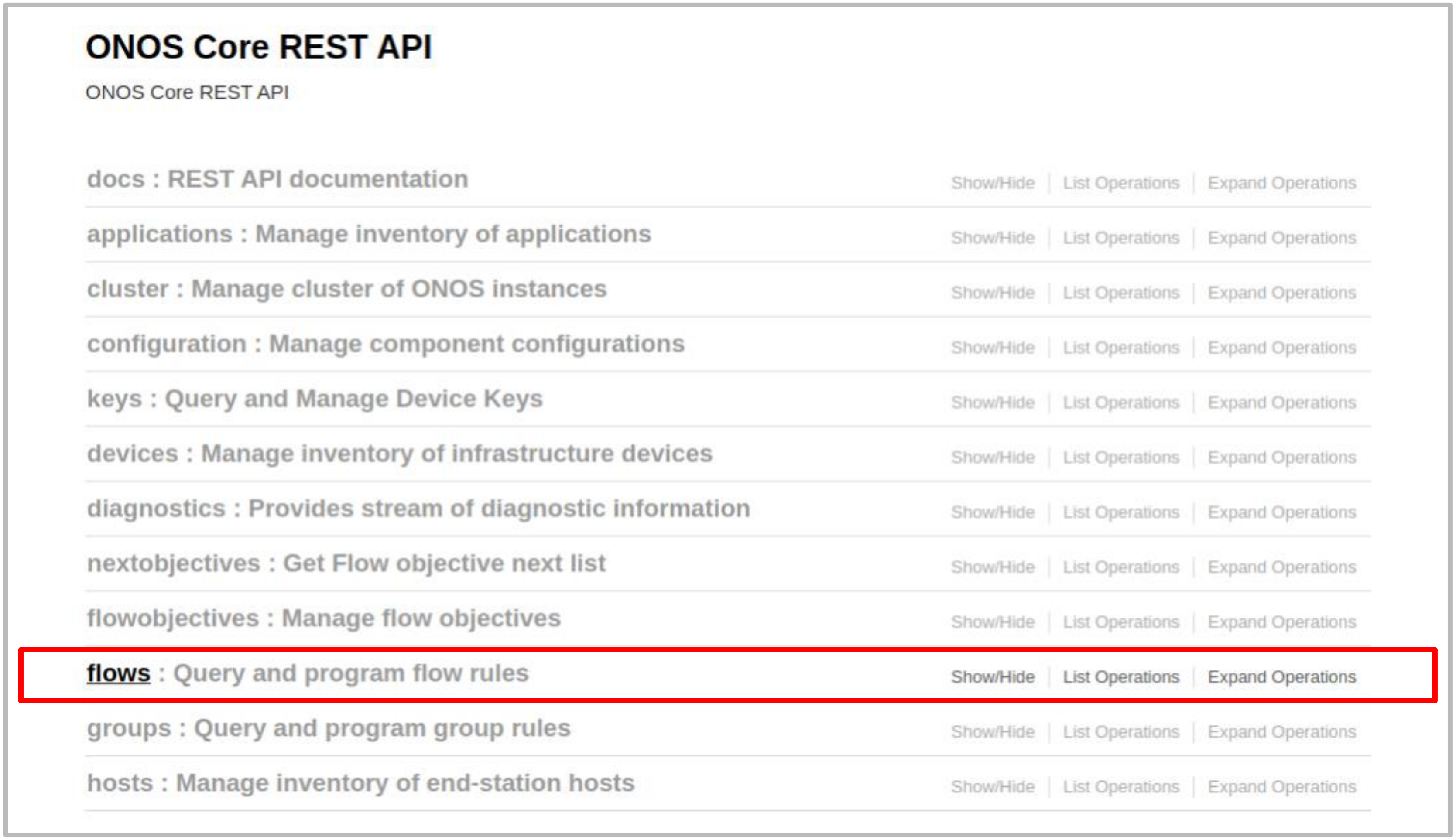
Flow ID

# Outline

- ❑ **Observe OpenFlow Messages**
  - ■ Monitor traffic between ONOS & Switches
  - ■ OpenFlow Message Observation
- ❑ **Install/ Delete Flow Rules**
  - ■ REST & curl
  - ■ ONOS & Topology Setup
  - ■ [Method 1] Via Command "curl"
  - ■ [Method 2] Via ONOS Web GUI
- ■ **Project 2 Requirements**
  - ■ Answer Questions (30%)
  - ■ Install Flow rules (40%)
  - ■ Create Topology with Broadcast Storm(30%)
  - ■ Bonus

# REST API on ONOS Web GUI

- Browse http://127.0.0.1:8181/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

flows : 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 REST API to Install Flow Rule on ONOS GUI

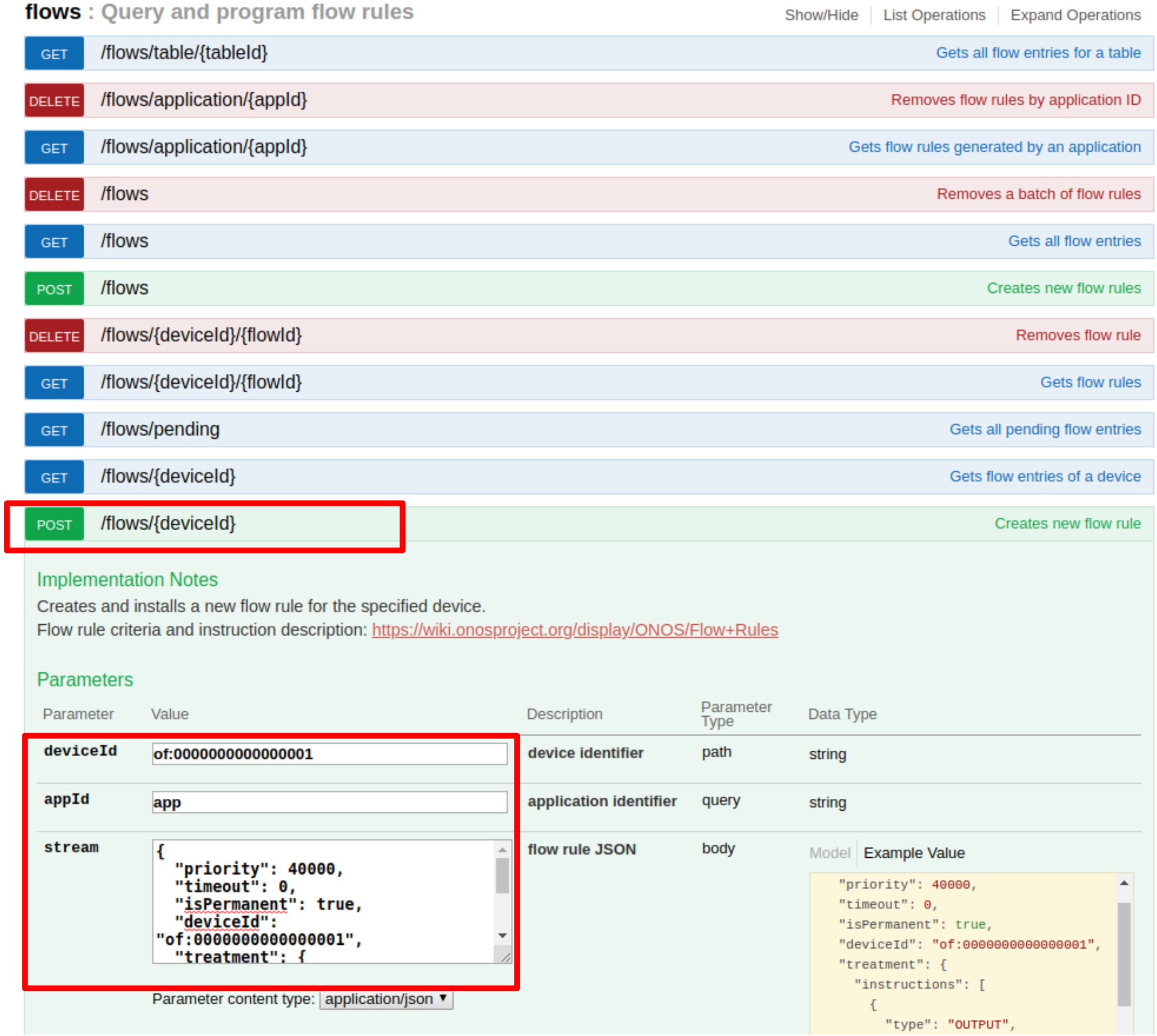
- ❑ Fill out required fields ("appID" could be arbitrary string)

**Select**

POST /flows/{deviceId}

**Type In**

deviceId, appid, stream (JSON file of flow rule)



**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 | (see below) | flow rule JSON | body | Model \| Example Value |

```
{
  "priority": 40000,
  "timeout": 0,
  "isPermanent": true,
  "deviceId":
"of:0000000000000001",
  "treatment": {
```

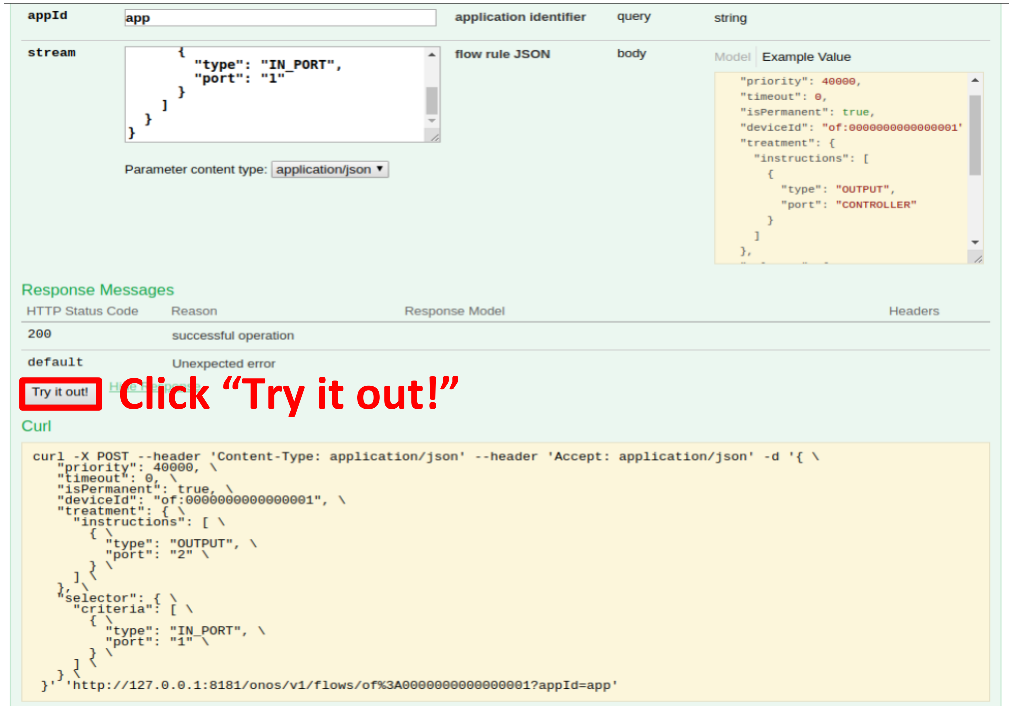
Parameter content type: application/json ▾

Example Value:

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

# Transfer Flow Rule via REST API on ONOS GUI

- Click “Try it out!”
  - It’ll automatically pass the JSON stream as a parameter to ONOS REST API
  - HTTP Status Code 201 represent HTTP Request granted
    - In case of “curl”, use “-i” option to include HTTP Response headers in the output



| | | | | |
|---|---|---|---|---|
| appId | app | application identifier | query | string |
| stream | | flow rule JSON | body | Model / Example Value |

```
        {
          "type": "IN_PORT",
          "port": "1"
        }
      ]
    }
}
```

Parameter content type: application/json ▾

```
  "priority": 40000,
  "timeout": 0,
  "isPermanent": true,
  "deviceId": "of:0000000000000001'
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "CONTROLLER"
      }
    ]
  },
```

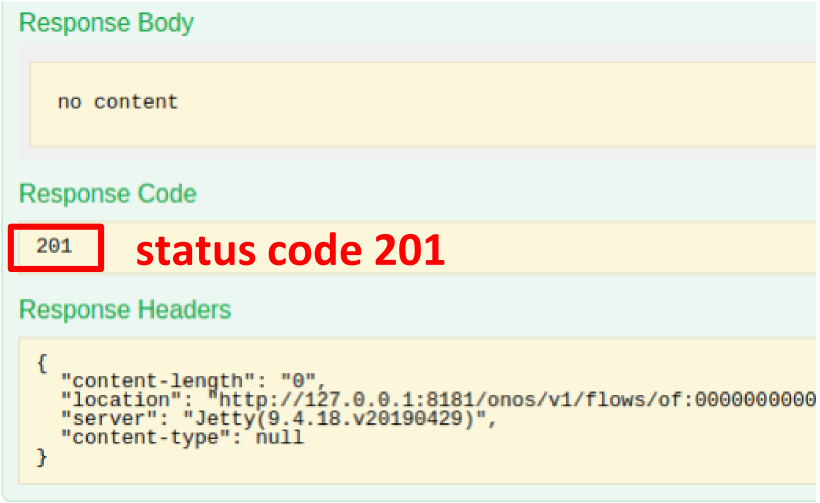
Response Messages

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

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'
```



Response Body

```
no content
```

Response Code

```
201
```

status code 201

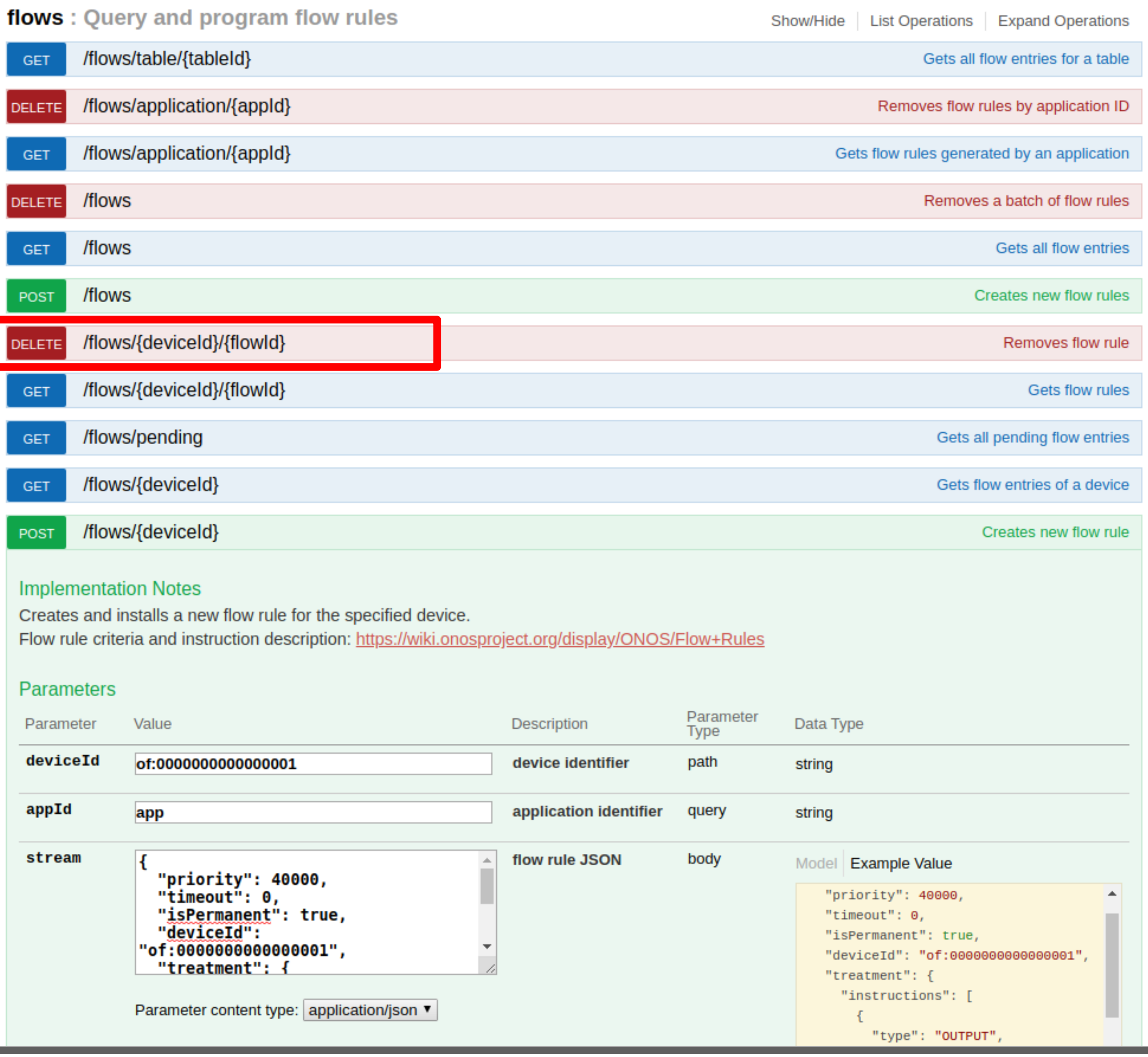
Response Headers

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

HTTP response replied by ONOS

# Delete Flow Rule Via ONOS Web GUI

- ❑ Same procedure as installing flow rules



**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 | (see below) | flow rule JSON | body | Model \| Example Value |

```
{
  "priority": 40000,
  "timeout": 0,
  "isPermanent": true,
  "deviceId":
"of:0000000000000001",
  "treatment": {
```

Parameter content type: application/json ▾

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

I. Answer Questions (30%)
II. Install Flow Rules (40%)
III. Create Topology with Broadcast Storm (30%)
IV. Bonus

# Project 2 Requirements

# Outline

- ❑ Observe OpenFlow Messages
  - Monitor traffic between ONOS & Switches
  - OpenFlow Message Observation
- ❑ **Install/ Delete Flow Rules**
  - REST & curl
  - ONOS & Topology Setup
  - [Method 1] Via Command “curl”
  - [Method 2] Via ONOS Web GUI
- **Project 2 Requirements**
  - Answer Questions (30%)
  - Install Flow rules (40%)
  - Create Topology with Broadcast Storm(30%)
  - Bonus

# Part 1: Answer Questions

- Preparation:
  - Start capturing packets on the loopback interface with Wireshark.
  - Create a topology mentioned before (i.e. h1-s1-h2).
  - Remember to activate “org.onosproject.fwd”.
  - Then, execute command “h1 ping h2  -c 5” in Mininet CLI.
  - When ping terminates, exit Mininet and stop capturing packets.

- Please answer the following questions:
  1. How many OpenFlow **headers** of type “OFPT_FLOW_MOD” are there among all the packets?

     Hint: More than one OpenFlow header may exist in a single packet.

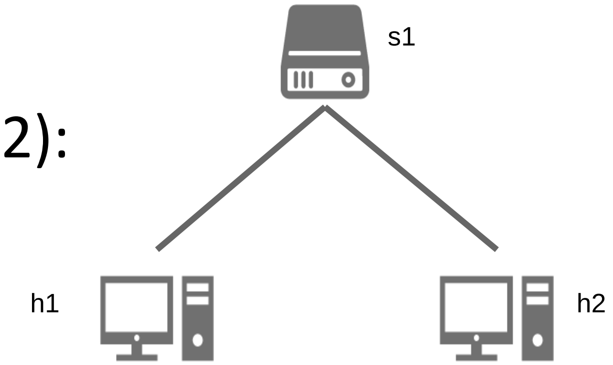
  2. Flow Rules
     a. What are the **matching fields** and the corresponding **actions** in each of “OFPT_FLOW_MOD” messages?
     b. What are the values of the **priority** fields of all “OFPT_FLOW_MOD” messages?

# Outline

- ❑ Observe OpenFlow Messages
  - ■ Monitor traffic between ONOS & Switches
  - ■ OpenFlow Message Observation
- ❑ **Install/ Delete Flow Rules**
  - ■ REST & curl
  - ■ ONOS & Topology Setup
  - ■ [Method 1] Via Command “curl”
  - ■ [Method 2] Via ONOS Web GUI
- ■ **Project 2 Requirements**
  - ■ Answer Questions (30%)
  - ■ Install Flow rules (40%)
  - ■ Create Topology with Broadcast Storm(30%)
  - ■ Bonus

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

- ❑ Please deactivate all the apps, **except those** initially activated.
  “org.onosproject.hostprovider”,
  “org.onosproject.lldpprovider”,
  “org.onosproject.optical-model”,
  “org.onosproject.openflow-base”,
  “org.onosproject.openflow”,
  “org.onosproject.drivers”
  and “org.onosproject.gui2”.
- ❑ Use the following topology (i.e. h1-s1-h2):



- ❑ Hand in all your flow rule files (.json) when you submit

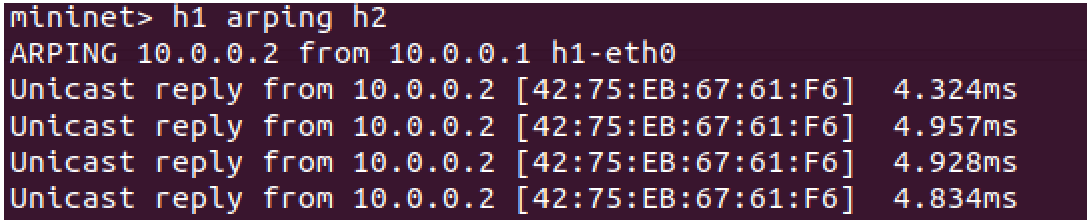
Note: Host1 should be able to ping host2 if the following flow rules are correctly installed

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

- Install following flow rules to forward ARP packets
  - Matching fields
    - Ethernet type (ARP)
  - Actions
    - Output from port, forwarding ARP packets to hosts

- Verify the flow rules your installed

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



```
mininet> h1 arping h2
ARPING 10.0.0.2 from 10.0.0.1 h1-eth0
Unicast reply from 10.0.0.2 [42:75:EB:67:61:F6]  4.324ms
Unicast reply from 10.0.0.2 [42:75:EB:67:61:F6]  4.957ms
Unicast reply from 10.0.0.2 [42:75:EB:67:61:F6]  4.928ms
Unicast reply from 10.0.0.2 [42:75:EB:67:61:F6]  4.834ms
```

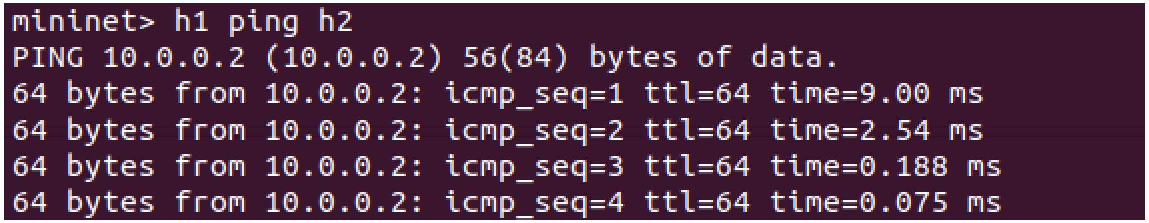
Hint: The priority of this flow rule MUST be higher than the flow rule initially installed (>40000), and less than 65535.

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

- □ Install flow rules to forward IPv4 packets
  - ■ Matching fields
    - IPv4 destination address
  - ■ Actions
    - Output from port, forwarding IPv4 packets to hosts

- □ Verify the flow rules your 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=9.00 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2.54 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.188 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.075 ms
```

Hint:

1. Switch may remove flow rules installed previously after a period of time.
2. Match fields may have dependency, please refer to OpenFlow spec v1.4.0.

# Outline

- ❑ Observe OpenFlow Messages
  - ■ Monitor traffic between ONOS & Switches
  - ■ OpenFlow Message Observation
- ❑ **Install/ Delete Flow Rules**
  - ■ REST & curl
  - ■ ONOS & Topology Setup
  - ■ [Method 1] Via Command “curl”
  - ■ [Method 2] Via ONOS Web GUI
- ■ **Project 2 Requirements**
  - ■ Answer Questions (30%)
  - ■ Install Flow rules (40%)
  - ■ Create Topology with Broadcast Storm (30%)
  - ■ Bonus

# Part 3: Create Topology with Broadcast Storm

❑ Create a **“Broadcast Storm”** phenomenon

1. Create a topology that may cause a **“Broadcast Storm”**.
2. Install flow rules on switches of the network.
3. Send packets from a host to another host.
4. Observe statuses of links of the network and the CPUs utilization of VM

- Describe what you have observed and explain why the broadcast storm occurred.
- Do NOT activate any other APPs, except for those initially activated by ONOS
- Hand in both Topology file (.py) and flow rule files (.json)

Hint: ONOS would initially install several flow rules.

# 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 |
|---|---|
| flows_s1-1_0748787.json | #1 flow rule to install on s1 |
| flows_s1-2_0748787.json | #2 flow rule to install on s1 |
| flows_s2-1_0748787.json | #1 flow rule to install on s2 |

# Bonus

- Activate only “org.onosproject.fwd” and other initially activated APPs.
- Please describe what happens in the data and control planes during the period when a host pings another host and until it receives the reply.
  - Include the mechanism that generates the packet in this period.
  - Please write down the operations made by both data plane or control planes.
- Please refer to the ONOS ReactiveForwarding application
  - https://github.com/opennetworkinglab/onos/blob/onos-2.2/apps/fwd/src/main/java/org/onosproject/fwd/ReactiveForwarding.java

Report

# Report Submission

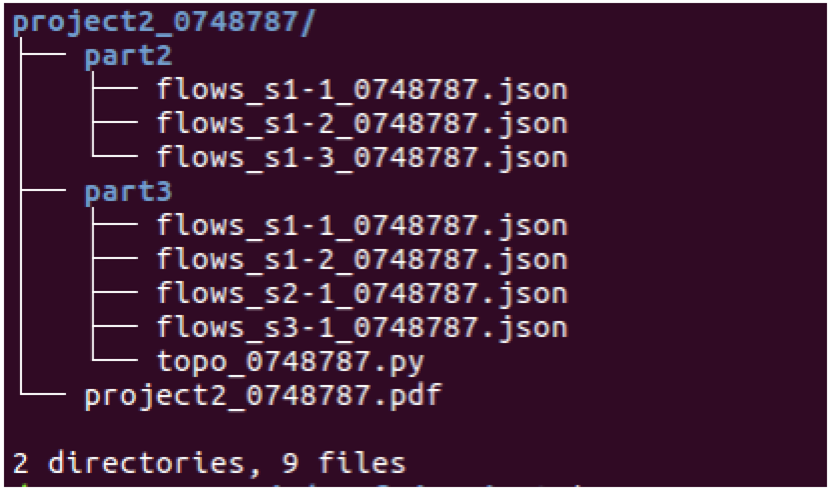
# Report Submission (1/2)

- Files:
  - A report: **project2_<studentID>.pdf**
    - Part 1: Answers to the part 1 questions
    - Part 2, Part 3 & Bonus:
      Take screenshots of your procedure and also explain in detail
    - Also write down what you’ve learned or solved
  - Several JSON files created, with correct naming convention, in both part 2 and part 3
  - A Python script for creating topology in part 3

# Report Submission (2/2)

- Submit:
  - Create folder: **project2_<studentID>**
  - In project2_<studentID>, create part2 and part3 directory and place files (i.e. .json, .py) into the corresponding directory

    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

2 directories, 9 files
```

  - Zip all the files into a zip file: **project2_<studentID>.zip**
  - Incorrect naming convention or format subjects to not scoring

Q & A

# Thank you

# References

- OpenFlow spec v1.4.0
  - https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf
- 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