



行動寬頻尖端技術課程推廣計畫

LAB 3: ONOS Controller with Mininet



國立臺北科技大學

National Taipei University of Technology

Oct. 26, 2017

葉又銘、江建勳

Lab 3/10



Outline

- 實驗目標與步驟
 - 瞭解 controller 與 switch 之間的關係，並且能夠控制封包的傳遞的方向。
- 實驗背景知識介紹
 - Open Network Operating System (ONOS)
 - REST API
- 實驗步驟
 1. 安裝 ONOS
 2. 連接 ONOS 及 Mininet
 3. 由 ONOS REST API 將 flow rule 設置至 OVS
- 實驗考核: 基於 REST API 設置 flow rule

實驗背景知識介紹 1: ONOS

ONOS

- **Open Network Operating System (ONOS)** 是由 Linux 基金會託管的一個開源社區。
 - 該項目的目標是為可擴展性、高性能和高可用性設計的通信服務提供商創建一個軟件定義網絡 (SDN) 操作系統。

Northbound Abstraction:

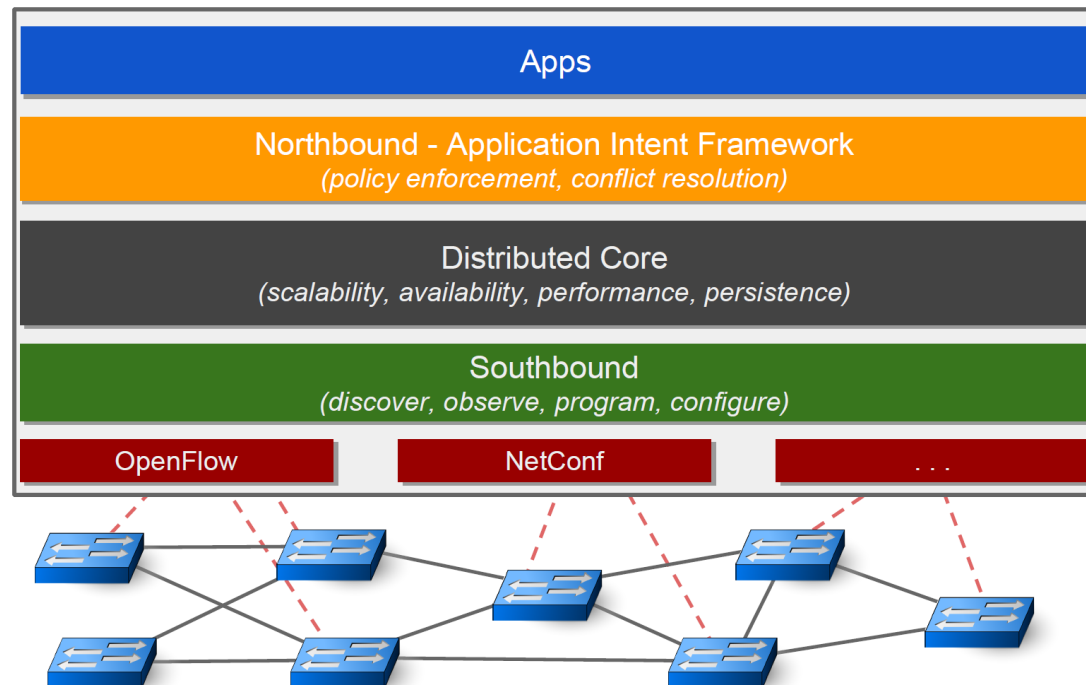
- network graph
- application intents

Core:

- distributed
- protocol independent

Southbound Abstraction:

- generalized OpenFlow
- pluggable & extensible



實驗背景知識介紹 2: REST API

REST 設計風格

- Representational State Transfer (REST) 是 Roy Thomas Fielding 博士於 2000 年在博士論文中提出的一種全球資訊網軟體架構風格
 - 目的是便於不同軟體 / 程序在網絡（例如：網際網路）中互相傳遞信息。
- 在 REST 中的資源 (Resource) 代表整個網路上的資源。
 - 網路上提供了各式各樣的資源，而網路上的資源由 Uniform Resource Identifier (URI，統一資源標識符) 來提供。
- Client 透過 URI 來獲取資源的具體象徵 (Representational)，並且使應用程式轉變其狀態 (以 Browser 而言，取得 HTML、CSS、JavaScript ... 來產生畫面)
 - 隨著不斷取得資源的具體象徵，Client 端不斷地改變其狀態，這樣不斷的反覆 (iterations) 過程就是所謂的 Representational State Transfer。

{ REST }

REST API

- 符合REST設計風格的Web API稱為RESTful API。它對資源(Resource)有以下三個方面的定義：

- 直觀簡短的資源地址：URI，比如：http://example.com/resources/。
- 傳輸的資源：Web服務接受與返回的網際網路媒體類型，比如：JSON，XML，YAML等。
- 對資源的操作：Web服務在該資源上所支持的一系列請求方法（比如：POST，GET，PUT或DELETE等）。

- HTTP請求方法(Request Method)在RESTful API中的典型應用：

資源	GET	PUT	POST	DELETE
一組資源的URI，比如 http://example.com/resources/	列出URI，以及該資源組中每個資源的詳細資訊（後者可選）。	使用給定的一組資源替換當前整組資源。	在本組資源中創建/追加一個新的資源。該操作往往返回新資源的URL。	刪除整組資源。
單個資源的URI，比如 http://example.com/resources/142	獲取指定的資源的詳細資訊，格式可以自選一個合適的網絡媒體類型（比如：XML、JSON等）	替換/創建指定的資源。並將其追加到相應的資源組中。	把指定的資源當做一個資源組，並在其下創建/追加一個新的元素，使其隸屬於當前資源。	刪除指定的元素。

RESTful

實驗步驟 1: 安裝 ONOS

安裝環境設置

在安裝 ONOS 之前，有些套件需要先準備好：

- 首先建立兩個資料夾
 - `mkdir Downloads Applications`
 - `cd Downloads`
 - ✓ 移動到Downloads
- 下載所需要的套件
 - `wget http://archive.apache.org/dist/karaf/3.0.5/apache-karaf-3.0.5.tar.gz`
 - `wget http://archive.apache.org/dist/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz`
- 解壓縮到Applications
 - `tar -zxvf apache-karaf-3.0.5.tar.gz -C ../Applications/`
 - `tar -zxvf apache-maven-3.3.9-bin.tar.gz -C ../Applications/`

*資料夾名稱務必一樣,解壓縮完回家目錄(cd)繼續下一步驟

安裝環境設置

- 接下來安裝 ONOS 使用的JAVA-8
 - `sudo apt-get install software-properties-common -y`
 - `sudo add-apt-repository ppa:webupd8team/java -y`
 - `sudo apt-get update`
 - `sudo apt-get install oracle-java8-installer oracle-java8-set-default -y`

注意安裝時假如碰到組態設置提示請選擇“YES”

- 完成安裝後要確認JAVA有沒有被引入的Linux系統內
 - `env | grep JAVA_HOME`
- 如果沒有回覆任何指令,請手動加入
 - `export JAVA_HOME=/usr/lib/jvm/java-8-oracle`

並再次確認是否安裝成功。

註：ONOS API 使用 JAVA 開發為主

安裝 ONOS

- 請先安裝 git
 - `sudo apt-get install git`
- 下載套件包
 - `git clone https://gerrit.onosproject.org/onos`
- 下載完畢後移動至資料夾內執行以下指令
 - `cd onos`
 - `git checkout 1.5.0`
 - ✓ 我們使用的版本為 1.5.0
- ONOS 的環境變數設置
 - `export ONOS_ROOT=~/.onos`
 - `source $ONOS_ROOT/tools/dev/bash_profile`
- 進行安裝
 - `mvn clean install`
 - ✓ 安裝過程需十五分鐘左右

安裝成功畫面

```
[INFO] onos-app-cordvtn ..... SUCCESS [ 3.038 s]
[INFO] onos-app-mfwd ..... SUCCESS [ 1.049 s]
[INFO] onos-app-igmp ..... SUCCESS [ 0.943 s]
[INFO] onos-app-pim ..... SUCCESS [ 1.136 s]
[INFO] onos-app-mlb ..... SUCCESS [ 0.884 s]
[INFO] onos-app-pp ..... SUCCESS [ 1.080 s]
[INFO] onos-app-drivermatrix ..... SUCCESS [ 0.859 s]
[INFO] onos-app-cpman ..... SUCCESS [ 9.343 s]
[INFO] onos-events ..... SUCCESS [ 0.946 s]
[INFO] onos-app-vrouter ..... SUCCESS [ 0.921 s]
[INFO] onos-app-cord-mcast ..... SUCCESS [ 3.030 s]
[INFO] onos-app-vpls ..... SUCCESS [ 3.761 s]
[INFO] onos-app-openstacknode ..... SUCCESS [ 1.316 s]
[INFO] onos-app-openstacknetworking ..... SUCCESS [ 0.411 s]
[INFO] onos-app-openstacknetworking-api ..... SUCCESS [ 0.679 s]
[INFO] onos-app-openstackswitching ..... SUCCESS [ 1.228 s]
[INFO] onos-app-openstackrouting ..... SUCCESS [ 1.107 s]
[INFO] onos-app-openstackinterface-app ..... SUCCESS [ 1.149 s]
[INFO] onos-app-openstacknetworking-web ..... SUCCESS [ 1.086 s]
[INFO] onos-app-openstacknetworking-app ..... SUCCESS [ 0.475 s]
[INFO] onos-incubator-core ..... SUCCESS [ 0.714 s]
[INFO] onos-incubator-rpc ..... SUCCESS [ 2.797 s]
[INFO] onos-incubator-rpc-grpc ..... SUCCESS [ 38.816 s]
[INFO] onos-features ..... SUCCESS [ 9.535 s]
[INFO] onos-archetypes ..... SUCCESS [ 0.130 s]
[INFO] onos-api-archetype ..... SUCCESS [ 15.653 s]
[INFO] onos-bundle-archetype ..... SUCCESS [ 0.108 s]
[INFO] onos-cli-archetype ..... SUCCESS [ 0.087 s]
[INFO] onos-rest-archetype ..... SUCCESS [ 0.036 s]
[INFO] onos-ui-archetype ..... SUCCESS [ 0.104 s]
[INFO] onos-uitab-archetype ..... SUCCESS [ 0.093 s]
[INFO] onos-uitopo-archetype ..... SUCCESS [ 0.098 s]
[INFO] onos-branding ..... SUCCESS [ 0.988 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 17:19 min
[INFO] Finished at: 2017-09-06T19:01:48+08:00
[INFO] Final Memory: 310M/741M
[INFO] -----
```

ONOS 參數設置

- 首先加入onos的參數
 - export ONOS_IP=自己interface的ip
 - ✓ 使用 ifconfig 查看
 - export ONOS_APPS=drivers,openflow,proxyarp,mobility,fwd
- 接著輸入指令
 - ok clean

```
lab@lab-VirtualBox:~/onos$ ok clean
Existing ONOS Karaf uses version different from 1.5.0; forcing clean install...
Removing existing ONOS Karaf, apps, and config directories...
Unpacking /home/lab/Downloads/apache-karaf-3.0.5.tar.gz to /home/lab/Applications...
Adding ONOS feature repository...
Adding ONOS boot features webconsole,onos-api,onos-core,onos-incubator,onos-cli,onos-rest,onos-gui...
Branding as ONOS...
Creating local cluster configs for IP 10.0.2.15...
Copying package configs...
Staging builtin apps...
Customizing apps to be auto-activated: drivers,openflow,proxyarp,mobility,fwd...
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 'system:shutdown' or 'logout' to shutdown ONOS.
```


實驗步驟二：連接 ONOS 及 Mininet

Step1. 開啟 ONOS controller

首先先開一個終端機，並開啟ONOS controller

```
lab@lab-VirtualBox:~/onos$ ok clean
Creating local cluster configs for IP 127.0.0.1...
Staging builtin apps...
Customizing apps to be auto-activated: drivers,openflow,fwd,proxyarp,mobility...
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 'system:shutdown' or 'logout' to shutdown ONOS.

onos> |
```

Step2. 開啟 Mininet

- 再開另一個新的終端機 (Terminal) 並執行 Mininet

```
lab@lab-VirtualBox: ~  
lab@lab-VirtualBox:~$ sudo mn --topo linear,5 --mac --switch ovsk,protocols=Open  
Flow13 --controller remote --arp  
[sudo] password for lab:  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2 h3 h4 h5  
*** Adding switches:  
s1 s2 s3 s4 s5  
*** Adding links:  
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (s2, s1) (s3, s2) (s4, s3) (s5, s4)  
  
*** Configuring hosts  
h1 h2 h3 h4 h5  
*** Starting controller  
c0  
*** Starting 5 switches  
s1 s2 s3 s4 s5 ...  
*** Starting CLI:  
mininet> 
```

Step2. 開啟 Mininet

- `sudo mn --topo linear,5 --mac --switch ovsk, protocols=OpenFlow13 --controller remote --arp`
 - ✓ “--topo linear,5”：製作一個範例拓樸，使用線型的方式，製作五個節點
 - ✓ “--mac”：mac 使用內建的方式編排
 - ✓ “--switch ovsk”：使用 OpenVSwitch
 - ✓ “protocols=OpenFlow13”：限定 Openflow 為 1.3 版本，預設為1.0
 - ✓ “--controller remote”：使用外部的 controller 來控制 Mininet

註：相對於 LAB 1，此處增加了 OpenVSwitch 的設置，包含 OpenFlow 版本設置與 controller 控制方式

- `sudo mn --controller=remote,ip=127.0.0.1 --topo=linear,3 --switch ovsk,protocols=OpenFlow13`

註：可指定 controller ip=127.0.0.1

Step3. 測試 Mininet

- 使用 pingall 的指令來檢查自己設定的拓樸

```
Starting CLI.  
mininet> pingall  
*** Ping: testing ping reachability  
h1 -> h2 h3 h4 h5  
h2 -> h1 h3 h4 h5  
h3 -> h1 h2 h4 h5  
h4 -> h1 h2 h3 h5  
h5 -> h1 h2 h3 h4  
*** Results: 0% dropped (20/20 received)  
mininet> █
```

- 共有5個節點,彼此都可以互相通訊

Step4. 在 ONOS 上確認有無擷取設備

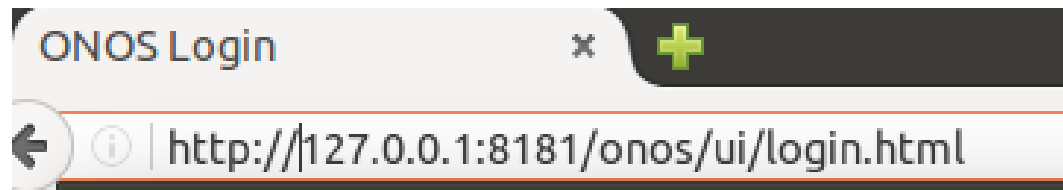
- 切換至 ONOS 的終端機上操作，輸入 hosts

```
onos> hosts
id=00:00:00:00:00:01/-1, mac=00:00:00:00:00:01, location=of:0000000000000001/1, vlan=-1, ip(s)=[]
id=00:00:00:00:00:02/-1, mac=00:00:00:00:00:02, location=of:0000000000000002/1, vlan=-1, ip(s)=[]
id=00:00:00:00:00:03/-1, mac=00:00:00:00:00:03, location=of:0000000000000003/1, vlan=-1, ip(s)=[]
id=00:00:00:00:00:04/-1, mac=00:00:00:00:00:04, location=of:0000000000000004/1, vlan=-1, ip(s)=[]
id=00:00:00:00:00:05/-1, mac=00:00:00:00:00:05, location=of:0000000000000005/1, vlan=-1, ip(s)=[]
onos> █
```

- 發現在 ONOS 上也有找到剛剛 Mininet 上的節點。

Step5. 利用瀏覽器操作ONOS

- 開啟瀏覽器在網址列輸入



- 可以看見登入畫面
 - 帳號和密碼都是:karaf



ONOS GUI 介面

The screenshot shows the ONOS GUI interface. The browser address bar displays `localhost:8181/onos/ui/index.html#/topo`. The main header includes the ONOS logo, the text "Open Network Operating System", and a "logout" link. A red box highlights the menu icon in the top left corner. A blue arrow points from a box labeled "其他細節資訊" (Other detailed information) to this menu icon. The left sidebar contains a list of navigation items under "Platform" and "Network" categories. A red box highlights the "Topology" item. A blue arrow points from a box labeled "其他可顯示的資訊 (Host, flow...)" (Other information that can be displayed (Host, flow...)) to this item. The main content area displays a network topology diagram with five nodes labeled 10.0.0.1 through 10.0.0.5. A red box highlights this diagram. A blue arrow points from a box labeled "Controller管理的拓撲" (Topology managed by Controller) to this diagram. On the right side, a red box highlights the "ONOS Summary" panel, which displays the following information:

ONOS Summary	
Version :	1.5.0
Devices :	5
Links :	8
Hosts :	5
Topology SCCs :	1
Intents :	0
Tunnels :	0
Flows :	25

A blue arrow points from a box labeled "Controller基本資訊" (Basic information of Controller) to this summary panel. At the bottom left, a red box highlights a toolbar with various icons for different views and actions.

實驗步驟三：由 ONOS REST API 將 flow rule 設置至 OVS

開啟 ONOS

- 基礎功能結束後，要開始來替 LAB1 的拓樸建置 Flow rule 了!

```
lab@lab-VirtualBox:~/onos$ ok clean
Creating local cluster configs for IP 127.0.0.1...
Staging builtin apps...
Customizing apps to be auto-activated: drivers,openflow,fwd,proxyarp,mobility...
Welcome to Open Network Operating System (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 'system:shutdown' or 'logout' to shutdown ONOS.

onos> 
```


開啟 Mininet

- 用mininet開啟自製拓樸

```
lab@lab-VirtualBox:~/mininet/custom$ sudo mn --custom topo-2sw-2host.py --topo mytopo --mac --controller=remote
```

topo-3sw-3host.py

- 自製拓樸請參考 LAB 1

- pingall

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

- 一開始會能夠互 ping 是因為 ONOS Controller 開啟封包自動轉傳的應用程式 fwd

ONOS 拓撲查詢

- 利用ONOS查詢有哪些設備 devices

```
onos> devices
id=of:0000000000000001, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.0.2, serial=None, managementAddress=127.0.0.1, protocol=OF_10, channelId=127.0.0.1:57232
id=of:0000000000000002, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.0.2, serial=None, managementAddress=127.0.0.1, protocol=OF_10, channelId=127.0.0.1:57230
id=of:0000000000000003, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.0.2, serial=None, managementAddress=127.0.0.1, protocol=OF_10, channelId=127.0.0.1:57228
```

- 利用ONOS查詢有哪些線路連接 links

```
onos> links
src=of:0000000000000001/3, dst=of:0000000000000002/2, type=DIRECT, state=ACTIVE, expected=false
src=of:0000000000000003/2, dst=of:0000000000000001/1, type=DIRECT, state=ACTIVE, expected=false
src=of:0000000000000002/1, dst=of:0000000000000003/3, type=DIRECT, state=ACTIVE, expected=false
src=of:0000000000000002/2, dst=of:0000000000000001/3, type=DIRECT, state=ACTIVE, expected=false
src=of:0000000000000003/3, dst=of:0000000000000002/1, type=DIRECT, state=ACTIVE, expected=false
src=of:0000000000000001/1, dst=of:0000000000000003/2, type=DIRECT, state=ACTIVE, expected=false
```

連結的設備
ID/Port

ONOS 關掉自動轉傳服務

- 此處我們要把 ONOS 上的自動轉傳服務先關掉，之後再利用 REST API 將 flow rule 安裝至OVS上。
 - 關掉自動轉傳服務

```
onos> app deactivate org.onosproject.fwd
```

- pingall 結果

```
mininet> pingall  
*** Ping: testing ping reachability  
h1 -> X X  
h2 -> X X  
h3 -> X X  
*** Results: 100% dropped (0/6 received)  
mininet> █
```

REST API for flow rule installation

- 開啟一個瀏覽器在網址列打上
 - [http://\(ControllerIP:port\)/onos/v1/docs](http://(ControllerIP:port)/onos/v1/docs)
 - ✓ Controller IP: 根據一開始 export 的 IP
 - ✓ ONOS Controller port: 8181
 - 輸入後會需要輸入帳號密碼
 - ✓ 均為 karaf

REST API 使用

- 基本上ONOS的功能都能使用REST來控制

ONOS Core REST API

Core APIs for external interactions with various ONOS subsystems.

applications : Manage inventory of applications	Show/Hide	List Operations	Expand Operations
cluster : Manage cluster of ONOS instances	Show/Hide	List Operations	Expand Operations
config : Inject devices, ports, links and end-station hosts	Show/Hide	List Operations	Expand Operations
configuration : Manage component configurations	Show/Hide	List Operations	Expand Operations
devices : Manage inventory of infrastructure		List Operations	Expand Operations
docs : REST API documentation		List Operations	Expand Operations
flowobjectives : Manage flow objectives		List Operations	Expand Operations
flows : Query and program flow rules		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
intents : Query, submit and withdraw network intents	Show/Hide	List Operations	Expand Operations
keys : Query and Manage Device Keys	Show/Hide	List Operations	Expand Operations
links : Manage inventory of infrastructure links	Show/Hide	List Operations	Expand Operations

今天要利用REST來下Flow,幫助h1能夠ping到h2

REST API: flows

- 選擇POST flows

flows : Query and program flow rules

Show/Hide

List Operations

Expand Operations

GET

/flows

Get all flow entries

DELETE

/flows/{deviceId}/{flowId}

Remove flow rule

GET

/flows/{deviceId}/{flowId}

Get flow rule

GET

/flows/{deviceId}

Get flow entries of a device

POST

/flows/{deviceId}

Create new flow rule

REST API: flows

POST `/flows/{deviceId}` [Create new flow rule](#)

Implementation Notes
Creates and installs a new flow rule for the specified device.
Instructions description: <https://wiki.onosproject.org/display/ONOS>
Criteria description: <https://wiki.onosproject.org/display/ONOS/Flo>

Parameters

Parameter	Value	Description	Parameter Type	Data Type
deviceId	(required)	device identifier	path	string
stream	(required)	flow rule JSON	body	

Parameter content type: `application/json`

Model | **Model Schema**

```
{
  "priority": 400000,
  "timeout": 0,
  "isPermanent": true,
  "deviceId": "of:0000000000000001",
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "CONTROLLER"
      }
    ]
  }
}
```

Click to set as parameter value

HTTP Status Code Reason Response Model Headers

針對哪一個設備下flow

Flow的架構區域

點選後會在
stream上產生範本

REST API: flows

- Stream的範本

- 範本為把Input (port:2) 對應到 output (port:1)
- 內容是把 port:2 的資料能夠導向 port:1，並且加入兩個條件 ETH_DST (mac 目的地) 及 ETH_SRC (mac來源端)，只要符合這樣的封包就轉傳至 port1

```
{
  "priority": 1234,
  "timeout": 0,
  "isPermanent": true,
  "deviceId": "of:0000000000000001",
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "3"
      }
    ]
  },
  "selector": {
    "criteria": [
      {
        "type": "IN_PORT",
        "port": "2"
      },
      {
        "type": "ETH_DST",
        "mac": "00:00:00:00:00:02"
      },
      {
        "type": "ETH_SRC",
        "mac": "00:00:00:00:00:01"
      }
    ]
  }
}
```

Priority: Flow的優先權
Timeout: 逾時時間
isPermanent: 是否永久
deviceId: 設備編號
Selector: 選擇目標
Treatment: 對應目標

REST API: flows

- 填寫好了之後，按下Try it out，就可以上傳flow到controller。

POST

/flows/{deviceId}

Create new flow rule

Implementation Notes

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

Parameters

Parameter	Value	Description	Parameter Type	Data Type
deviceId	of:0000000000000001	device identifier	path	string
stream	<div><div><pre>{ "priority": 400000, "timeout": 0, "isPermanent": true, "deviceId": "of:0000000000000001", "treatment": { "instructions": [{ "type": "OUTPUT", "port": "CONTROLLER" }] } }</pre></div><div>Parameter content type: application/json</div></div> <div>flow rule JSON</div> <div>body</div> <div><div>Model</div><div>Model Schema</div><div><pre>{ "priority": 400000, "timeout": 0, "isPermanent": true, "deviceId": "of:0000000000000001", "treatment": { "instructions": [{ "type": "OUTPUT", "port": "CONTROLLER" }] } }</pre></div><div>Click to set as parameter value</div></div>			

Response Messages

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

Try it out!

REST API: flows

- 按下try it 後，下方會有URL：

Request URL

```
http://127.0.0.1:8181/onos/v1/flows/of%3A000000000000000001
```

- 開新瀏覽器，貼上這段URL可以看見這個設備的flow。

REST API: flows

- 可以看見剛剛自己下的flow細節

註：請注意每個人的拓撲Port都不一樣，要好設定好自己的Port以免發生轉傳不了的意外

```
▼ 2:
  id: "21673574388798441"
  tableId: 0
  appId: "org.onosproject.rest"
  groupId: 0
  priority: 1234
  timeout: 0
  isPermanent: true
  deviceId: "of:000000000000000001"
  state: "ADDED"
  life: 5
  packets: 0
  bytes: 0
  lastSeen: 1506734163599
  treatment:
    ▼ instructions:
      ▼ 0:
        type: "OUTPUT"
        port: "3"
        deferred:
      ▼ selector:
        ▼ criteria:
          ▼ 0:
            type: "IN_PORT"
            port: 2
          ▼ 1:
            type: "ETH_DST"
            mac: "00:00:00:00:00:02"
          ▼ 2:
            type: "ETH_SRC"
            mac: "00:00:00:00:00:01"
```

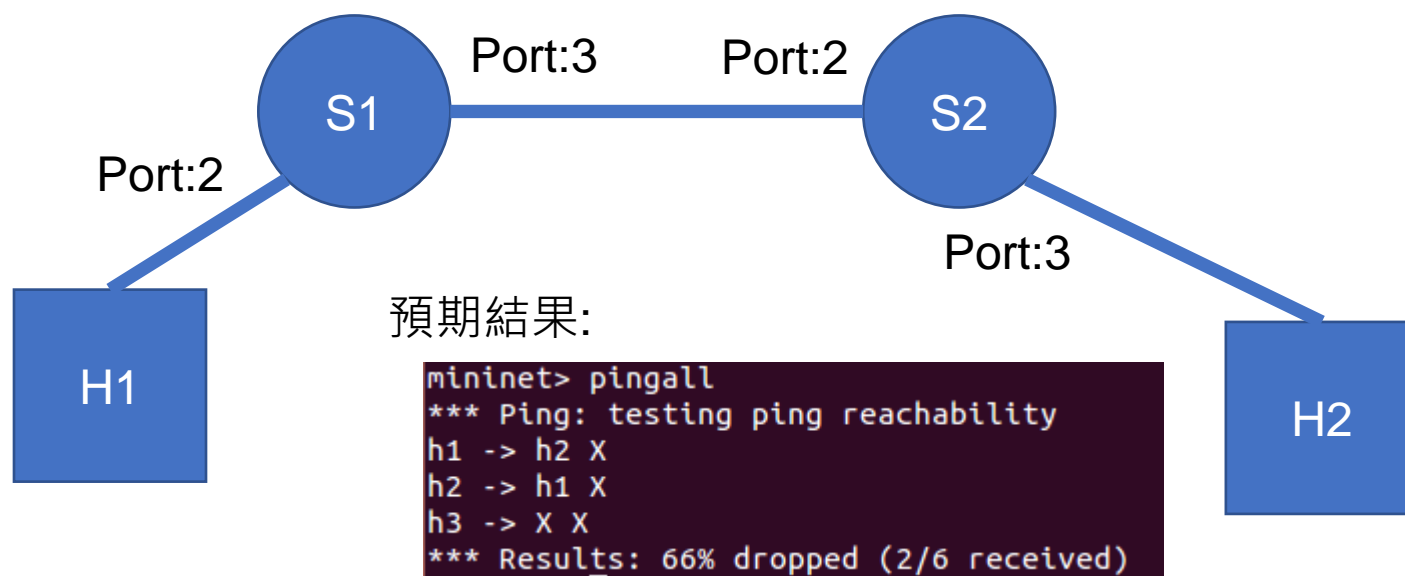
實驗考核

Lab210



LAB 3 實驗考核

- 請利用剛才的REST API 幫助 H1 Ping H2
- 提示:
 - 1. 剛剛是針對 S1 下 flow，但是封包要抵達 H2 還會經過 S2，所以也需要對 S2 下 flow
 - 2. PING 的封包是需要雙向的溝通，Ping 過去 & 回來的 flow entry 都需要有。



Thanks for your attention!

Lab210

TAIPEI
TECH