

下世代Network Slicing模組設計

實驗單元：Mininet搭配Controller 使用OpenFlow模擬SDN網路環境

國立中山大學 資訊工程系
授課教師：李宗南教授
教材編撰：曾國維

目錄

CONTENTS

- 01 實驗目標
- 02 實驗設備
- 03 SDN簡介與基本架構
- 04 軟體介紹
- 05 安裝及執行
- 06 實驗要求

01 實驗目標



實驗目標

- 實驗目標1：修課學生得以了解 SDN 的基本觀念及架構
- 實驗目標2：修課學生得以理解 Mininet的網路環境模擬以及SDN控制器 Ryu的使用以及SDN應用程式Postman的使用。
- 實驗目標3：修課學生得以完成軟體定義網路環境進行實驗，及驗證 OpenFlow執行。

02 實驗設備



實驗設備

- 硬體:
 - 電腦：Ubuntu作業系統16.04
- 軟體
 - Mininet: 2.3.0d4
 - Ryu: ryu-manager 4.30
 - Postman REST Client
 - gcc編譯器
 - vi/vim文字編輯器

03 SDN簡介與基本架構



SDN簡介與基本架構

- 一種新的網路架構。利用OpenFlow協定，將路由器及交換器傳輸封包中的控制與內容分離，以軟體方式實作。
 - 控制平面 (Control plane)
 - 資料平面 (Data plane)
- 這個架構可以讓網路管理員，在不更動硬體裝置的前提下，以中央控制方式，用程式重新規劃網路，為控制網路流量提供了新的方法，也提供了核心網路及應用創新的良好平台。

SDN簡介與基本架構

SDN 架構



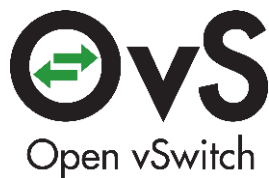
POSTMAN



OPEN
DAYLIGHT

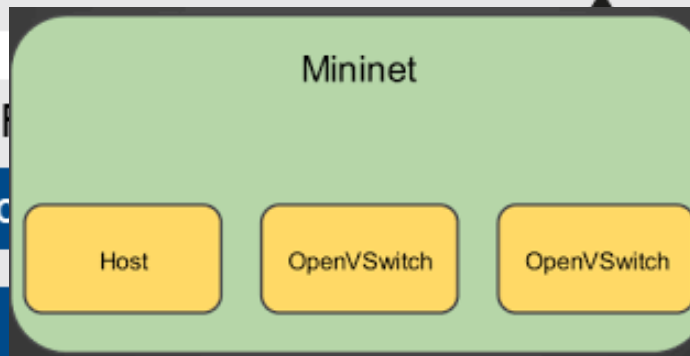


CONTROL LAYER



INFRASTRUCTURE LAYER

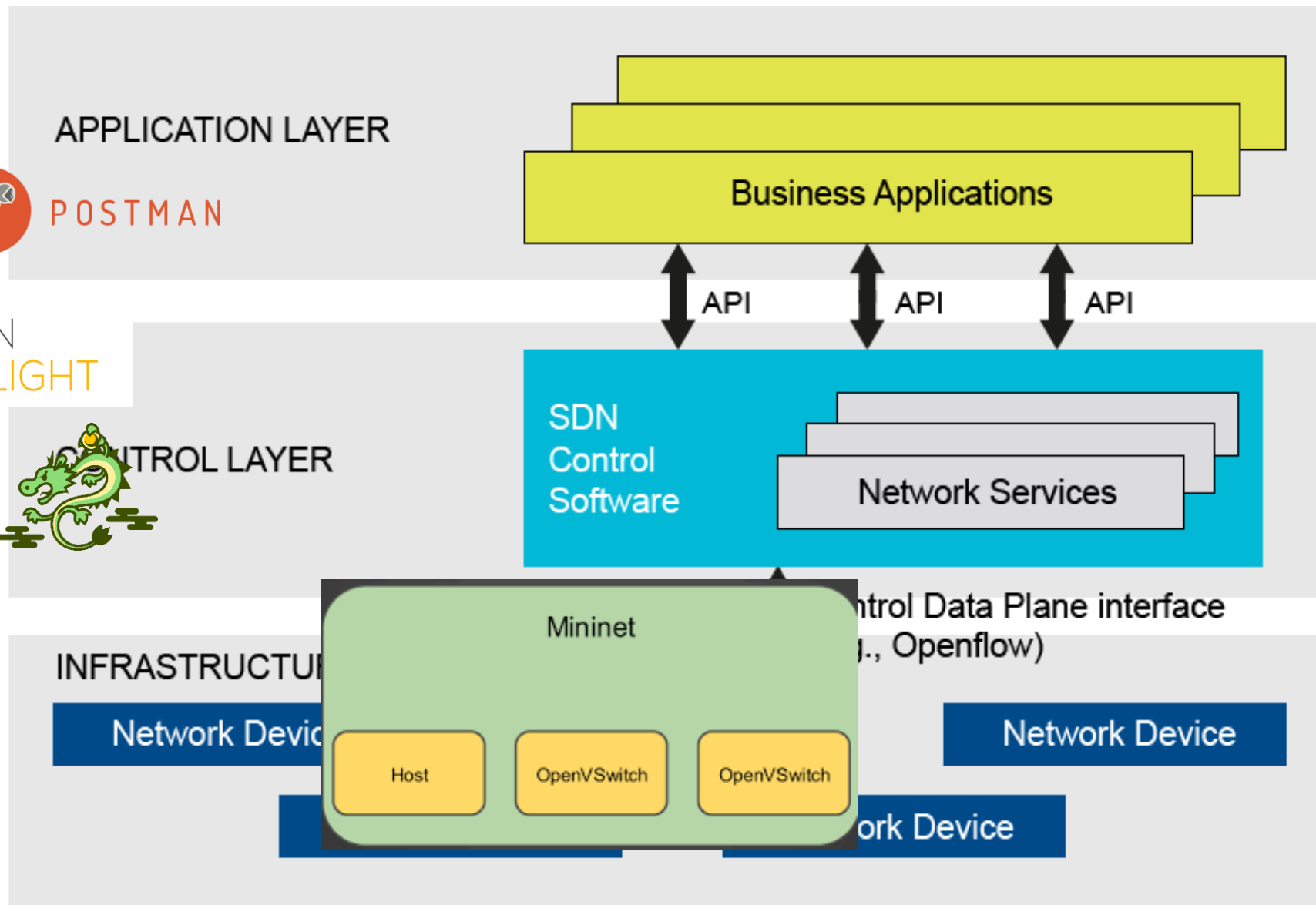
Network Device



Control Data Plane interface
(e.g., Openflow)

Network Device

Network Device



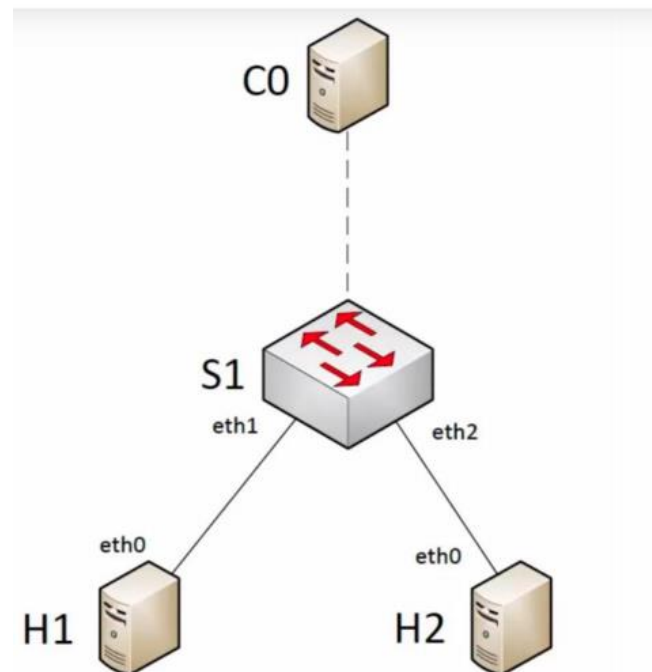
source: http://www.cc.ntu.edu.tw/chinese/epaper/0029/20140620_2908.html

04 軟體介紹



軟體介紹-Mininet(1)

- Mininet是一個可以透過一些虛擬終端機、路由器、交換器等連接創建虛擬網路拓樸的平台
- 兼容運行不同系統上 (windows\linux\Mac OS) 可以輕易的在自己的個人電腦中創作支援SDN的區域網路，造出的虛擬host以真實電腦般發送封包，以驗證實驗方法
- Mininet自帶switch、host、controller，同時，在mininet上可以安裝OpenvSwitch以及多種控制器 (NOX\POX\RYU\Floodlight\OpenDaylight等)



軟體介紹-Mininet(2)

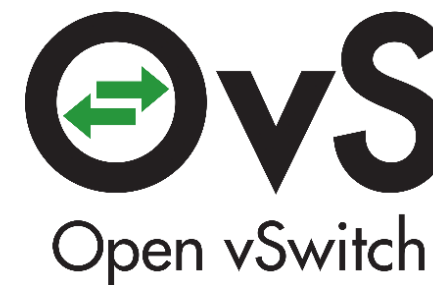
建立拓譜主要分成2種方式：

- 1.利用終端機指令下參數，建立拓譜。
- 2.撰寫python檔直接利用.py檔撰寫程式指令以及建立拓譜。

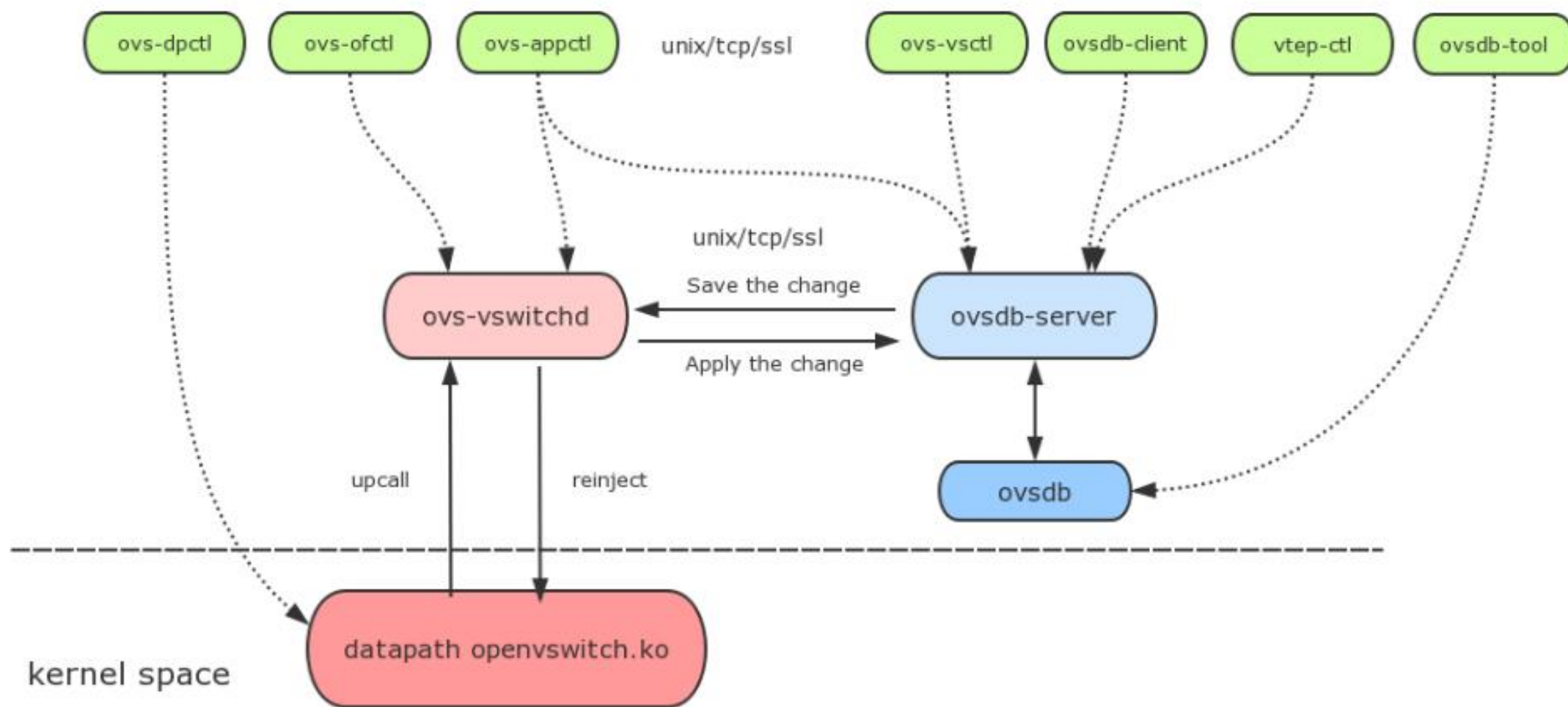
MININET

軟體介紹-OpenvSwitch(1)

- Open Networking Foundation(ONF)的開源計畫，顧名思義是一個 virtual switch，它的目的是讓大規模網路透過可編程化來進行擴展及管理
- 可用於定義路徑，切割網域，QoS或是流量監控，同時支持標準的管理接口服務和各項協議(sFlow,NetFlow,OpenFlow等)。
- Mininet中以OVS作為作為實現虛擬Switch與虛擬Host間連線

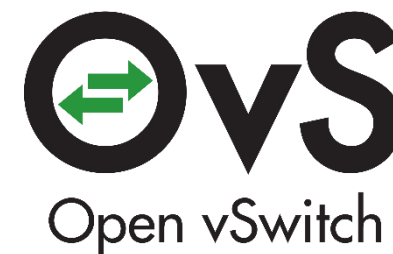


軟體介紹-OpenvSwitch(2)



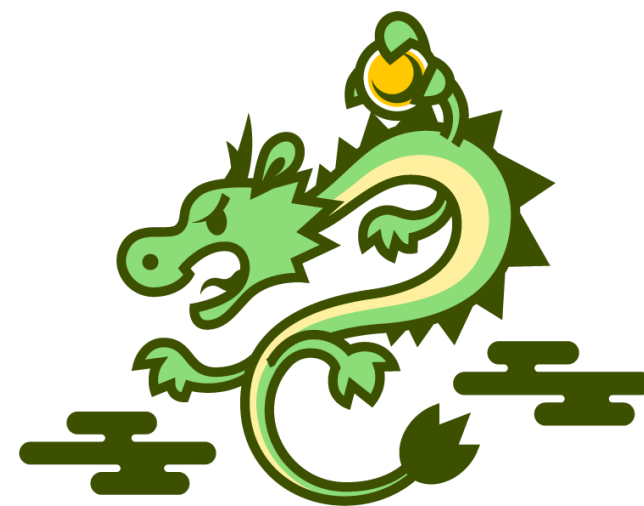
軟體介紹-OpenvSwitch(3)

1. ovs-dpctl : 管理 ovs datapath, 大部分資訊都是透過 netlink 反應。
2. ovs-vsctl : 對 ovssdb 操作, bridge, interface 新增, 刪除, 查詢, 操作指令。
3. ovs-ofctl : openflow switch 管理工具, 可以操作與 openflow 相關的設定。
4. ovs-appctl : ovs-vswitchd 的管理工具, 可以跟 ovs-vswitchd 程序溝通, ex: ofproto/trace 可用來追蹤封包 flow。
5. ovs-vswitchd : 主要模組, 實現內核 datapath upcall 處理以及 ofproto 查表, 同時是 dpdk datapath 處理程序。



軟體介紹-Ryu

- Ryu is a component-based software defined networking framework.
- Ryu 是來自於日本 NTT 所開發以及設計，針對 SDN 的控制器開發框架 (Framework)
- 包含了 OpenFlow (以及其他部分協定) Controller 的功能
- 使用Python進行開發



05 安裝及執行



- Mininet
- Postman REST
- Mininet + Ryu - Task1, Task2, Task3
- Mininet + OpenDaylight – Task4

安裝及執行-Mininet

- \$ git clone git://github.com/mininet/mininet
- \$ cd mininet
- \$ util/install.sh -a
- \$ sudo mn

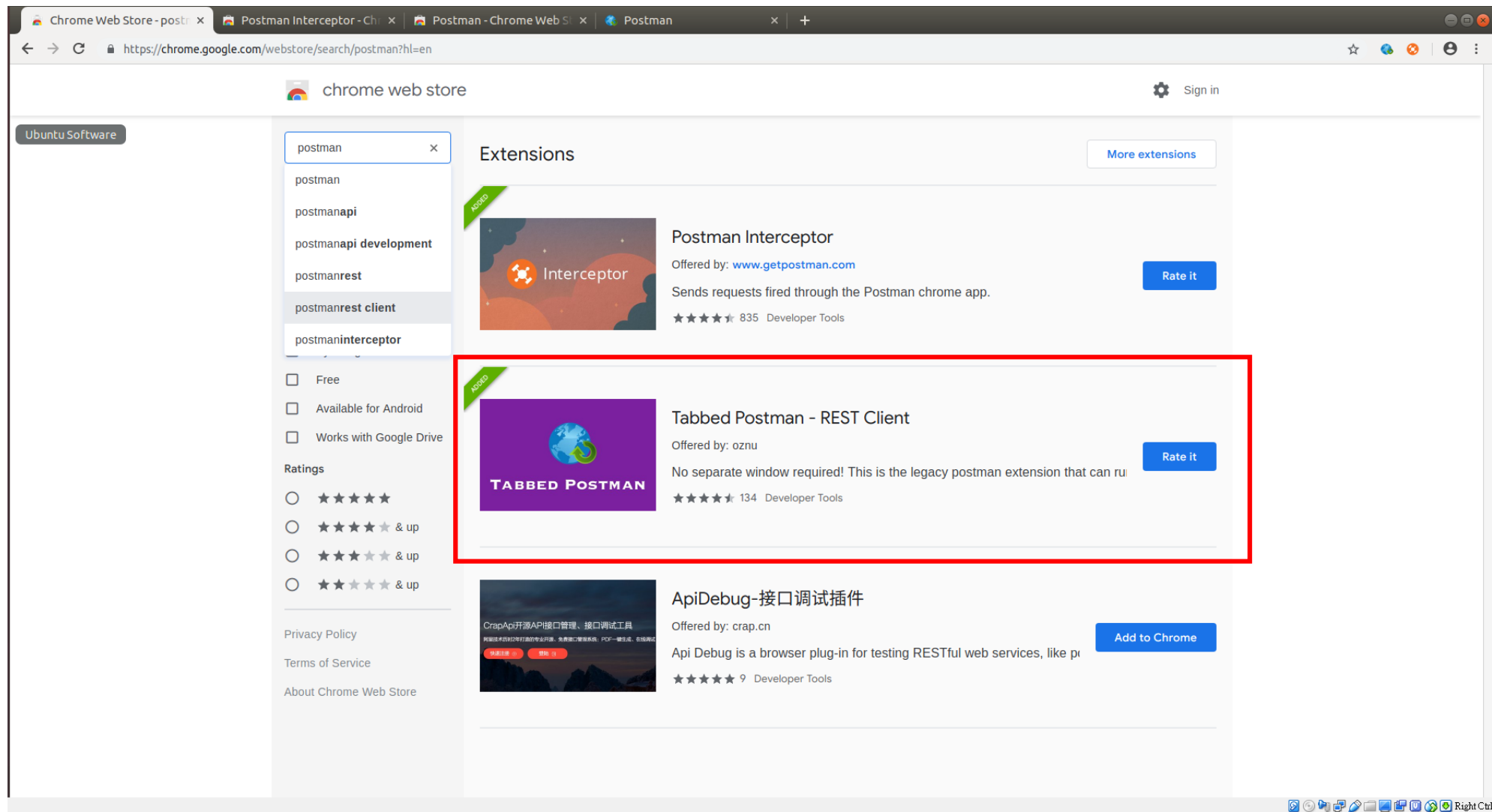
```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> _
```

安裝及執行-Mininet Dump OVS Flows

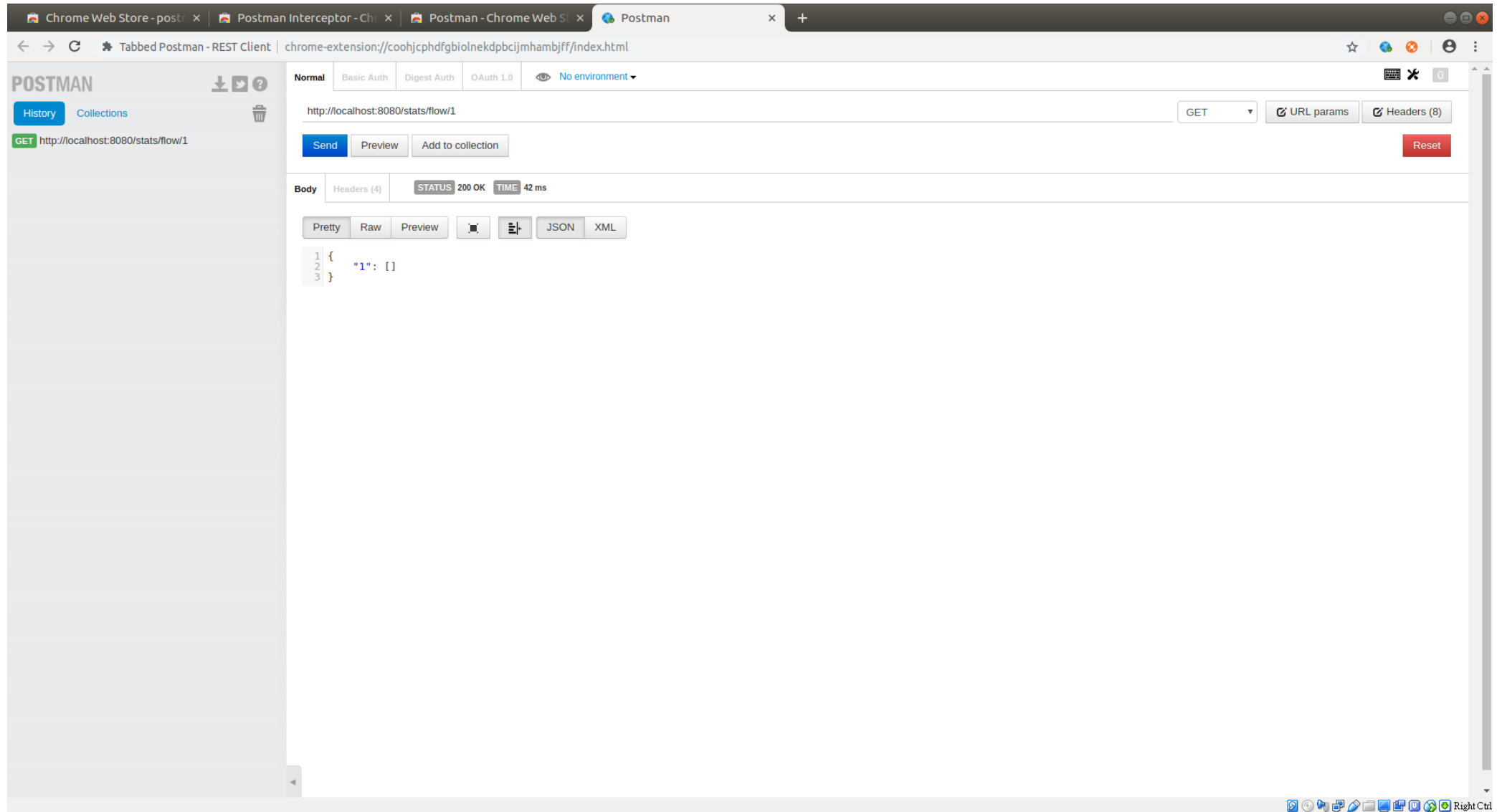
- Dump Flows
 - \$ sh ovs-ofctl -O openflow13 dump-flows s1
- Dump Hidden Flows
 - \$ sh ovs-appctl bridge/dump-flows s1

```
mininet> sh ovs-ofctl -O openflow13 dump-flows s1
cookie=0x1, duration=30.901s, table=0, n_packets=5, n_bytes=378, hard_timeout=2000, send_flow_rem priority=11114,dl_dst=00:00:00:00:00:01 actions=output:"s1-eth1"
cookie=0x1, duration=25.683s, table=0, n_packets=4, n_bytes=336, hard_timeout=2000, send_flow_rem priority=11113,dl_dst=00:00:00:00:00:02 actions=output:"s1-eth2"
cookie=0x1, duration=23.584s, table=0, n_packets=1, n_bytes=42, hard_timeout=2000, send_flow_rem priority=11111,arp,arp_tpa=10.0.0.2 actions=output:"s1-eth2"
mininet> sh ovs-appctl bridge/dump-flows s1
duration=47s, n_packets=5, n_bytes=378, priority=11114,dl_dst=00:00:00:00:00:01,actions=output:1
duration=42s, n_packets=4, n_bytes=336, priority=11113,dl_dst=00:00:00:00:00:02,actions=output:2
duration=40s, n_packets=1, n_bytes=42, priority=11111,arp,arp_tpa=10.0.0.2,actions=output:2
table_id=254, duration=77s, n_packets=0, n_bytes=0, priority=2,recirc_id=0,actions=drop
table_id=254, duration=77s, n_packets=0, n_bytes=0, priority=0,reg0=0x1,actions=controller(reason=)
table_id=254, duration=77s, n_packets=18, n_bytes=1452, priority=0,reg0=0x2,actions=drop
table_id=254, duration=77s, n_packets=0, n_bytes=0, priority=0,reg0=0x3,actions=drop
mininet> █
```

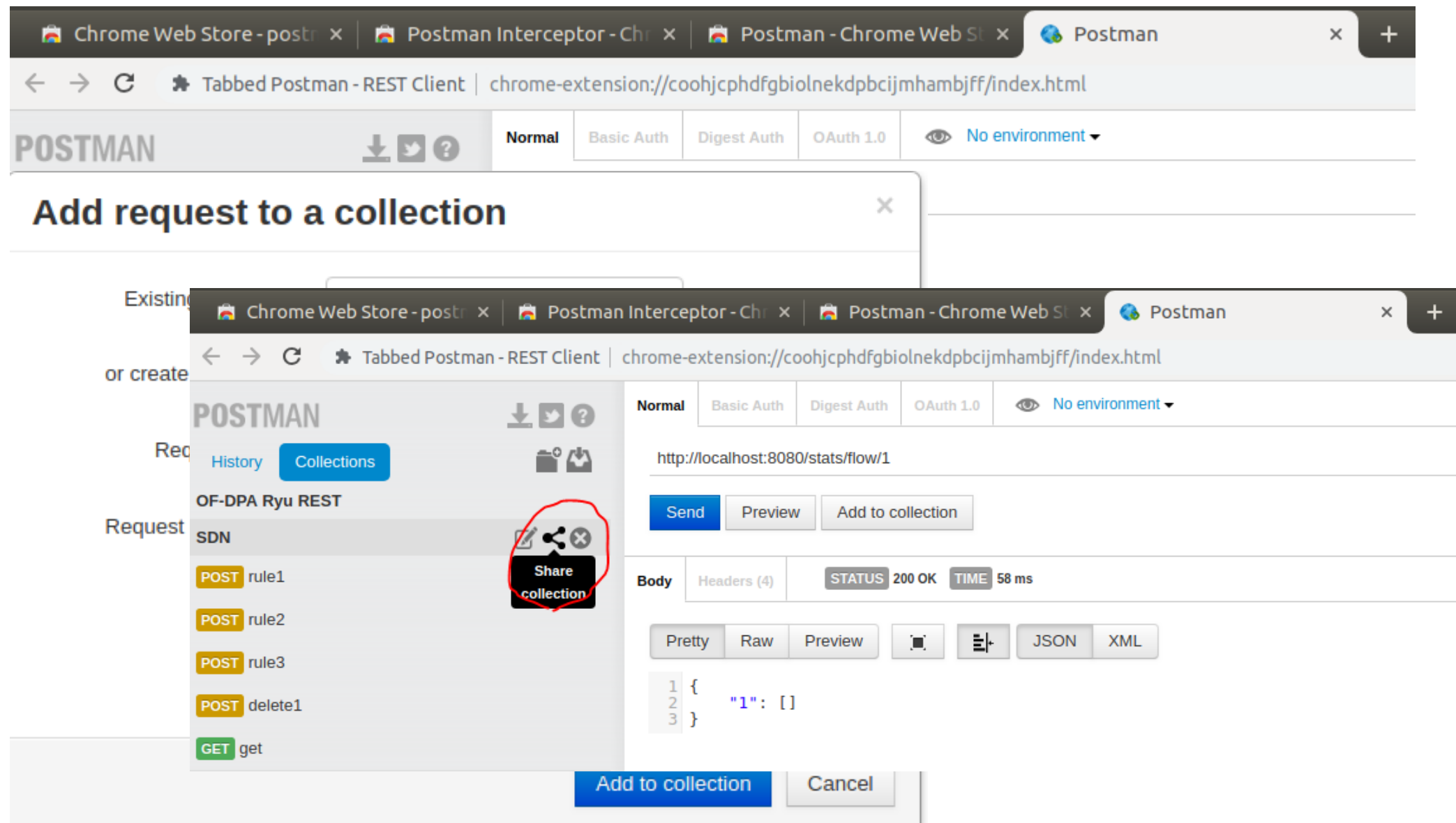
安裝及執行-Postman REST



安裝及執行-Postman REST



安裝及執行-Postman Get Collections

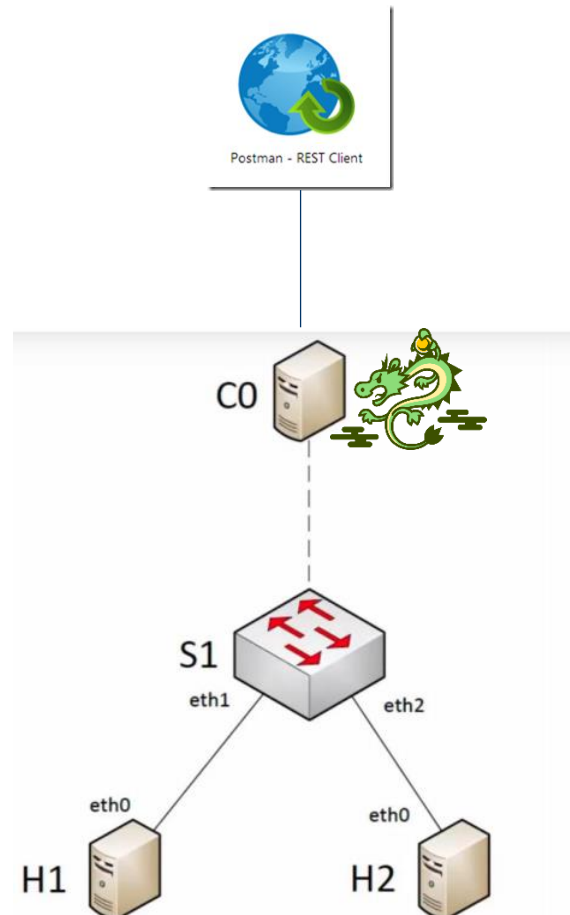


安裝及執行-Ryu

- `$ sudo apt-get update`
- `$ sudo apt-get install libxml2-dev libxslt1-dev python-pip python-eventlet python-routes python-webob python-paramiko`
- `$ sudo pip install msgpack-python eventlet==0.15.2`
- `$ sudo pip install six --upgrade`
- `$ sudo pip install oslo.config q --upgrade`
- `$ sudo pip install ryu`

Task1 – Mininet + Ryu

- Let h1 and h2 be able to access each other via **Postman REST**. (pingall)



Task1 – Mininet + Ryu

- Ryu
- `$ ryu-manager --verbose ryu.app.ofctl_rest`
- Mininet
- `$ sudo mn --mac --switch ovsk,protocols=OpenFlow13 --controller remote,ip=127.0.0.1,port=6633`

Task1 – Mininet + Ryu


- Add 3 Flows
- Flow 1 : Arp Requery Query:h1(10.0.0.2) to h2(10.0.0.2)
- Flow 2 : h2 mac addr(00:00:00: 00:00:02) ->output to port 2
- Flow 3 : h1 mac addr(00:00:00: 00:00:01) ->output to port 1

Task1 – Mininet + Ryu

```
1 {
2   "dpid": {{dpid}},
3   "cookie": 1,
4   "cookie_mask": 1,
5   "table_id": 0,
6   #"idle_timeout": 30,
7   "hard_timeout": 200,
8   "priority": 11111,
9   "flags": 1,
10  "match":{
11    "dl_type":2054,
12    "arp_tpa":"10.0.0.2"
13  },
14  "actions":[
15    {
16      "type":"OUTPUT",
17      "port":2
18    }
19  ]
20 }
```

```
1 {
2   "dpid": {{dpid}},
3   "cookie": 1,
4   "cookie_mask": 1,
5   "table_id": 0,
6   #"idle_timeout": 30,
7   "hard_timeout": 200,
8   "priority": 11109,
9   "flags": 1,
10  "match":{
11    "dl_dst":"00:00:00:00:00:01"
12  },
13  "actions":[
14    {
15      "type":"OUTPUT",
16      "port":1
17    }
18  ]
19 }
```

```
1 {
2   "dpid": {{dpid}},
3   "cookie": 1,
4   "cookie_mask": 1,
5   "table_id": 0,
6   #"idle_timeout": 30,
7   "hard_timeout": 200,
8   "priority": 11110,
9   "flags": 1,
10  "match":{
11    "dl_dst":"00:00:00:00:00:02"
12  },
13  "actions":[
14    {
15      "type":"OUTPUT",
16      "port":2
17    }
18  ]
19 }
```



M3Lab Since 2010

Task1 – Mininet + Ryu

```
ciis@ciis-VirtualBox: ~  
File Edit View Search Terminal Tabs Help  
ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x  
CONSUMES EventOFPEchoRequest  
CONSUMES EventOFPEchoReply  
BRICK RestStatsApi  
CONSUMES EventOFPPDescStatsReply  
CONSUMES EventOFPPQueueStatsReply  
CONSUMES EventOFPPFlowStatsReply  
CONSUMES EventOFPPPortDescStatsReply  
CONSUMES EventOFPPMeterConfigStatsReply  
CONSUMES EventOFPPQueueGetConfigReply  
CONSUMES EventOFPPGroupFeaturesStatsReply  
CONSUMES EventOFPPGroupStatsReply  
CONSUMES EventOFPPTableStatsReply  
CONSUMES EventOFPPMeterFeaturesStatsReply  
CONSUMES EventOFPPAggregateStatsReply  
CONSUMES EventOFPPSwitchFeatures  
CONSUMES EventOFPPQueueDescStatsReply  
CONSUMES EventOFPPMeterStatsReply  
CONSUMES EventOFPPPortStatsReply  
CONSUMES EventOFPPRoleReply  
CONSUMES EventOFPPTableFeaturesStatsReply  
CONSUMES EventOFPPGroupDescStatsReply  
CONSUMES EventOFPPStatsReply  
(8133) wsgi starting up on http://0.0.0.0:8080  
  
ciis@ciis-VirtualBox: ~$ sudo mn --mac --switch ovsk,protocols=OpenFlow13 --controller remote,ip=127.0.0.1,port=6633  
[sudo] password for ciis:  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1)  
*** Configuring hosts  
h1 h2  
*** Starting controller  
c0  
*** Starting 1 switches  
s1 ...  
*** Starting CLI:  
mininet> pingall  
*** Ping: testing ping reachability  
h1 -> X  
h2 -> X  
*** Results: 100% dropped (0/2 received)  
mininet>
```

Task1 – Mininet + Ryu

The screenshot displays the Tabbed Postman - REST Client interface. The left sidebar shows a collection named "OF-DPA Ryu REST" with a "GET" method selected for the endpoint "http://localhost:8080/stats/flow/1". The main panel shows the response body in JSON format, which is a list of two flow statistics objects. The status is "200 OK" and the time taken is "77 ms".

The JSON response body is as follows:

```
{
  "1": {
    "actions": [
      "OUTPUT:2"
    ],
    "idle timeout": 0,
    "cookie": 1,
    "packet count": 30,
    "hard timeout": 2000,
    "byte count": 2660,
    "duration_sec": 392,
    "duration_nsec": 359000000,
    "priority": 11113,
    "length": 88,
    "flags": 1,
    "table_id": 0,
    "match": {
      "dl_dst": "00:00:00:00:00:02"
    }
  },
  "2": {
    "actions": [
      "OUTPUT:1"
    ],
    "idle timeout": 0,
    "cookie": 1,
    "packet count": 31,
    "hard timeout": 2000,
    "byte count": 2702,
    "duration_sec": 385,
    "duration_nsec": 255000000,
    "priority": 11114,
    "length": 88,
    "flags": 1,
    "table_id": 0,
    "match": {
      "dl_dst": "00:00:00:00:00:01"
    }
  }
}
```

An inset terminal window shows the following output:

```
ciis@ciis-VirtualBox: ~
File Edit View Search Terminal Tabs Help
ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x
13 packets transmitted, 13 received, 0% packet loss, time 12266ms
rtt min/avg/max/mdev = 0.040/0.078/0.398/0.093 ms
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=0.041 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.076 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.047 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.104 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.081 ms
^C
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5108ms
rtt min/avg/max/mdev = 0.041/0.069/0.104/0.023 ms
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=0.542 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.072 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.059 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.059/0.224/0.542/0.225 ms
mininet>
```

Task2 – Mininet + Ryu

- Develop an app which **allows controller to flood every packet** that comes to it.
- Ryu
- `$ ryu-manager --verbose ryu.app.ofctl_rest ryu.app.mysw_basic`
- Path
- `$ /usr/local/lib/python2.7/dist-packages/ryu/app/`
- Mininet
- `$ sudo mn --mac --switch ovsk,protocols=OpenFlow13 --controller remote,ip=127.0.0.1,port=6633`

Task2 – Mininet + Ryu

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_3
class L2Switch(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]
    def __init__(self, *args, **kwargs):
        super(L2Switch, self).__init__(*args, **kwargs)
    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
    def switch_features_handler(self, ev):
        self.logger.info("***** Add Default Flow *****")
        dp = ev.msg.datapath
        ofp = dp.ofproto
        parser = dp.ofproto_parser
        match = parser.OFPMatch()
        actions = [parser.OFPACTIONOutput(ofp.OFPP_CONTROLLER, ofp.OFPCML_NO_BUFFER)]
        inst = [parser.OFPIInstructionActions(ofp.OFPIT_APPLY_ACTIONS, actions)]
        mod = parser.OFPFlowMod(datapath=dp, priority=0, match=match, instructions=inst)
        dp.send_msg(mod)
    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def packet_in_handler(self, ev):
        msg = ev.msg
        dp = msg.datapath
        ofp = dp.ofproto
        parser = dp.ofproto_parser
        self.logger.debug("***** Debug *****")
        # output port
        data = None
        if msg.buffer_id == ofp.OFP_NO_BUFFER:
            data = msg.data
        actions = [parser.OFPACTIONOutput(ofp.OFPP_FLOOD)]
        out = parser.OFPPacketOut(datapath=dp, buffer_id=msg.buffer_id, in_port=msg.match['in_port'], actions=actions, data=data)
        dp.send_msg(out)
```


Task2 – Mininet + Ryu

```
ciis@ciis-VirtualBox: ~  
File Edit View Search Terminal Tabs Help  
ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x ciis  
ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x  
CONSUMES EventOFPPacketIn  
CONSUMES EventOFPSwitchFeatures  
BRICK RestStatsApi  
CONSUMES EventOFPTableFeaturesStatsReply  
CONSUMES EventOFPPStatsReply  
CONSUMES EventOFPPFlowStatsReply  
CONSUMES EventOFPMeterFeaturesStatsReply  
CONSUMES EventOFPAggregateStatsReply  
CONSUMES EventOFPPortDescStatsReply  
CONSUMES EventOFPPQueueStatsReply  
CONSUMES EventOFPPGroupDescStatsReply  
CONSUMES EventOFPMeterConfigStatsReply  
CONSUMES EventOFPPGroupStatsReply  
CONSUMES EventOFPPQueueGetConfigReply  
CONSUMES EventOFPPGroupFeaturesStatsReply  
CONSUMES EventOFPPPortStatsReply  
CONSUMES EventOFPTableStatsReply  
CONSUMES EventOFPPDescStatsReply  
CONSUMES EventOFPMeterStatsReply  
CONSUMES EventOFPSwitchFeatures  
CONSUMES EventOFPPQueueDescStatsReply  
CONSUMES EventOFPPRoleReply  
(8389) wsgi starting up on http://0.0.0.0:8080  
  
ciis@ciis-VirtualBox: ~$  
ciis@ciis-VirtualBox: ~$ sudo mn --mac --switch ovsk,protocols=OpenFlow13 --contr  
oller remote,ip=127.0.0.1,port=6633  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1)  
*** Configuring hosts  
h1 h2  
*** Starting controller  
c0  
*** Starting 1 switches  
s1 ...  
*** Starting CLI:  
mininet> pingall  
*** Ping: testing ping reachability  
h1 -> h2  
h2 -> h1  
*** Results: 0% dropped (2/2 received)  
mininet>
```

Task2 – Mininet + Ryu

The screenshot displays the Postman REST client interface and a terminal window. The Postman interface shows a GET request to `http://localhost:8080/stats/flow/1` with a status of 200 OK and a response time of 53 ms. The response body is shown in JSON format, with the `actions` field circled in red, containing the value `"OUTPUT:CONTROLLER"`.

The terminal window shows the output of the REST client, including the request and response details. The response is a 200 OK status with a response time of 0.008934 seconds. The response body is a JSON object with the following fields: `actions`, `idle_timeout`, `cookie`, `packet_count`, `hard_timeout`, `byte_count`, `duration_sec`, `duration_nsec`, `priority`, `length`, `flags`, `table_id`, and `match`.

```

{
  "1": [
    {
      "actions": [
        "OUTPUT:CONTROLLER"
      ],
      "idle_timeout": 0,
      "cookie": 0,
      "packet_count": 26,
      "hard_timeout": 0,
      "byte_count": 2124,
      "duration_sec": 18,
      "duration_nsec": 76000000,
      "priority": 0,
      "length": 80,
      "flags": 0,
      "table_id": 0,
      "match": {}
    }
  ]
}

```

Task3 – Mininet + Ryu

- Develop an app which **allows controller to add a flow.**
 - Match : everything
 - Action : Flood
- Ryu
- `$ ryu-manager --verbose ryu.app.ofctl_rest ryu.app.mysw_flow`
- Path
- `$ /usr/local/lib/python2.7/dist-packages/ryu/app/`
- Mininet
- `$ sudo mn --mac --switch ovsk,protocols=OpenFlow13 --controller remote,ip=127.0.0.1,port=6633`

Task3 – Mininet + Ryu

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_3

class L2Switch(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]
    def __init__(self, *args, **kwargs):
        super(L2Switch, self).__init__(*args, **kwargs)
    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
    def switch_features_handler(self, ev):
        self.logger.info("***** Add Default Flow *****")
        dp = ev.msg.datapath
        ofp = dp.ofproto
        parser = dp.ofproto_parser
        match = parser.OFPMatch()
        actions = [parser.OFPActionOutput(ofp.OFPP_CONTROLLER, ofp.OFPCML_NO_BUFFER)]
        inst = [parser.OFPInstructionActions(ofp.OFPIT_APPLY_ACTIONS, actions)]
        mod = parser.OFPFlowMod(datapath=dp, priority=0, match=match, instructions=inst)
        dp.send_msg(mod)
    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def packet_in_handler(self, ev):
        msg = ev.msg
        dp = msg.datapath
        ofp = dp.ofproto
        parser = dp.ofproto_parser

        self.logger.debug("***** Add FLOOD Flow *****")
        match = parser.OFPMatch()
        actions = [parser.OFPActionOutput(ofp.OFPP_FLOOD)]
        inst = [parser.OFPInstructionActions(ofp.OFPIT_APPLY_ACTIONS, actions)]
        mod = parser.OFPFlowMod(datapath=dp, priority=2048, match=match, instructions=inst)
        dp.send_msg(mod)
        ## output port
        #data = None
        #if msg.buffer_id == ofp.OFP_NO_BUFFER:
        #    data = msg.data
        #actions = [parser.OFPActionOutput(ofp.OFPP_FLOOD)]
        #out = parser.OFPFloodPacketOut(datapath=dp, buffer_id=msg.buffer_id, in_port=msg.match['in_port'], actions=actions, data=data)
        #dp.send_msg(out)
```

Task3 – Mininet + Ryu

```
ciis@ciis-VirtualBox: ~
File Edit View Search Terminal Tabs Help
ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x
CONSUMES EventOFPPacketIn
CONSUMES EventOFPSwitchFeatures
BRICK RestStatsApi
CONSUMES EventOFPDdescStatsReply
CONSUMES EventOFPGroupFeaturesStatsReply
CONSUMES EventOFPMeterFeaturesStatsReply
CONSUMES EventOFPPQueueStatsReply
CONSUMES EventOFPGroupDescStatsReply
CONSUMES EventOFPPortDescStatsReply
CONSUMES EventOFPMeterConfigStatsReply
CONSUMES EventOFPPQueueGetConfigReply
CONSUMES EventOFPPStatsReply
CONSUMES EventOFPPRoleReply
CONSUMES EventOFPPGroupStatsReply
CONSUMES EventOFPPTableStatsReply
CONSUMES EventOFPSwitchFeatures
CONSUMES EventOFPPQueueDescStatsReply
CONSUMES EventOFPPFlowStatsReply
CONSUMES EventOFPMeterStatsReply
CONSUMES EventOFPPPortStatsReply
CONSUMES EventOFPPAggregateStatsReply
CONSUMES EventOFPPTableFeaturesStatsReply
(8697) wsgi starting up on http://0.0.0.0:8080

ciis@ciis-VirtualBox: ~$
ciis@ciis-VirtualBox: ~$ sudo mn --mac --switch ovsk,protocols=OpenFlow13 --controller remote,ip=127.0.0.1,port=6633
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

Task3 – Mininet + Ryu

The image shows a web browser with Postman Interceptor and Postman tabs. The Postman interface displays a REST client for 'OF-DPA Ryu REST' with a collection of endpoints. The 'GET /stats/flow/1' endpoint is selected, showing a 200 OK status and a JSON response. The response body is highlighted with a red circle, showing a list of flow statistics. A second red circle highlights the 'actions' field in the response, which is set to 'OUTPUT:CONTROLLER'.

On the right, a terminal window shows the output of a Mininet script. The script includes commands for setting up the Ryu controller, adding default and flood flows, and sending a message to the datapath. The terminal output shows the successful execution of these commands, including the addition of a default flow and a flood flow, and the successful execution of the 'GET /stats/flow/1' request.

Postman REST Client Details:

- Endpoint: `http://localhost:8080/stats/flow/1`
- Method: `GET`
- Status: `200 OK`
- Time: `81 ms`

JSON Response Body:

```
{
  "1": {
    "actions": [
      "OUTPUT:CONTROLLER"
    ],
    "idle_timeout": 0,
    "cookie": 0,
    "packet_count": 1,
    "hard_timeout": 0,
    "byte_count": 110,
    "duration_nsec": 43,
    "priority": 0,
    "length": 80,
    "flags": 0,
    "table_id": 0,
    "match": {}
  },
  "actions": [
    "OUTPUT:FLOOD"
  ],
  "idle_timeout": 0,
  "cookie": 0,
  "packet_count": 22,
  "hard_timeout": 0,
  "byte_count": 1676,
  "duration_nsec": 43,
  "priority": 2048,
  "length": 80,
  "flags": 0,
  "table_id": 0,
  "match": {}
}
```

Terminal Output:

```
ciis@ciis-VirtualBox: ~
File Edit View Search Terminal Tabs Help
ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x ciis@ciis-VirtualBox: ~ x
EVENT ofp_event->dpset EventOFPSwitchFeatures
EVENT ofp_event->L2Switch EventOFPSwitchFeatures
switch features ev version=0x4,msg_type=0x6,msg_len=0x20,xid=0x813ed7ec,OFPSwitc
hFeatures(auxiliary_id=0,capabilities=79,datapath_id=1,n_buffers=0,n_tables=254)
***** Add Default Flow *****
move onto main mode
EVENT ofp_event->dpset EventOFPPortStatus
DPSET: register datapath <ryu.controller.controller.Datapath object at 0x7f094d9
93b90>
EVENT ofp_event->L2Switch EventOFPPacketIn
***** Add FLOOD Flow *****
EVENT ofp_event->dpset EventOFPPortStatus
DPSET: A port was modified.(datapath id = 0000000000000001, port number = 2)
EVENT ofp_event->dpset EventOFPPortStatus
DPSET: A port was modified.(datapath id = 0000000000000001, port number = 1)
(8697) accepted ('127.0.0.1', 50792)
Sending message with xid(813ed7f0) to datapath(0000000000000001): version=None,m
sg_type=None,msg_len=None,xid=0x813ed7f0,OFPPFlowStatsRequest(cookie=0,cookie_mas
k=0,flags=0,match=OFPMatch(oxm_fields={}),out_group=4294967295,out_port=42949672
95,table_id=255,type=1)
EVENT ofp_event->RestStatsApi EventOFPPFlowStatsReply
127.0.0.1 - - [20/Dec/2018 05:29:53] "GET /stats/flow/1 HTTP/1.1" 200 624 0.0501
56
```

06 實驗要求



- Task1
 - 請將Postman下達的flow集成一Collection並儲存，並也將實驗結果一併截圖繳交
- Task2
 - 請在程式碼中加入學號並在controller中印出，並將實驗結果截圖繳交
- Task3
 - 請在程式碼中加入學號並在controller中印出，並將實驗結果截圖繳交

[1]軟體定義網路 (SDN)

<https://www.xinguard.com/content.aspx?id=34>

[2] Mininet+OpenDaylight

[http://140.117.164.12/data/SDN_NFV_class\(v2_2017\)/SDN_Lab1.pdf](http://140.117.164.12/data/SDN_NFV_class(v2_2017)/SDN_Lab1.pdf)

[3] Mininet+Ryu

[http://140.117.164.12/data/SDN_NFV_class\(v2_2017\)/SDN_Lab4.pdf](http://140.117.164.12/data/SDN_NFV_class(v2_2017)/SDN_Lab4.pdf)

[4] OpenFlow 通訊協定

https://osrg.github.io/ryu-book/zh_tw/html/openflow_protocol.html