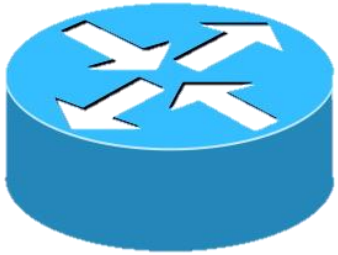


Ryu Controller

Speaker: Yi-Ta Chen

Advisor: Tsung-Nan Lin

Date: 20180430

Outline

- Ryu Controller 簡介
- Lab2-1:L3 routing
- Lab2-2:Firewall

Ryu Controller 介紹

- Ryu Controller
  - Operations Support System (OSS)
    - computer systems used by telecommunications service providers to manage their networks
  - Ryu SDN Framework founded by NTT
    - Python library
    - Apache v2 license
  - 支援多種管理網路設備的協定
    - OpenFlow, Netconf, OF-config etc.
  - 支援OpenFlow 版本 1.0~1.5

Ryu Controller 介紹

- Ryu Controller
  - 多種SDN應用與函式庫
    - Simple_switch, firewall, router
    - LACP, STP
    - RestAPI, RPC
  - GUI拓樸視覺模組
    - Topology discovery
    - Flow entry management
  - 支援其他平台
    - OpenStack
    - Zookeeper

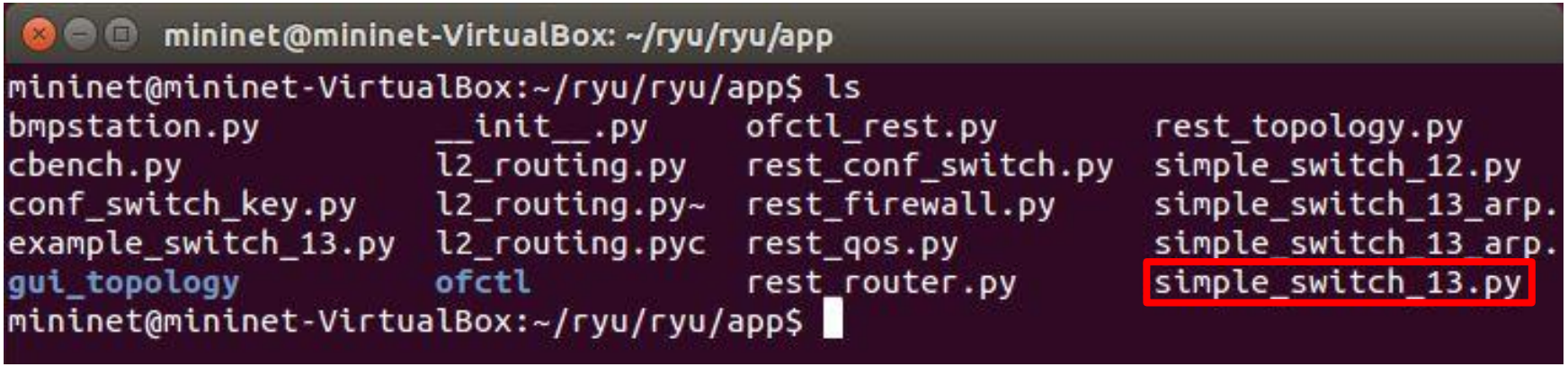
Ryu Controller 介紹

- 選用Ryu作為Controller的產品
  - Pica8, Broadcom等
  - Open Source switch (OvS, CPQD)
- Ryu優點
  - 輕巧、快速開發
  - OpenFlow協定支援完整
  - 支援軟/硬體交換機

http://www.slideshare.net/apnic/ryu-sdn-framework?qid=b548fb40-1f9d-477c-ad35-2b9c85f47358&v=default&b=&from_search=1

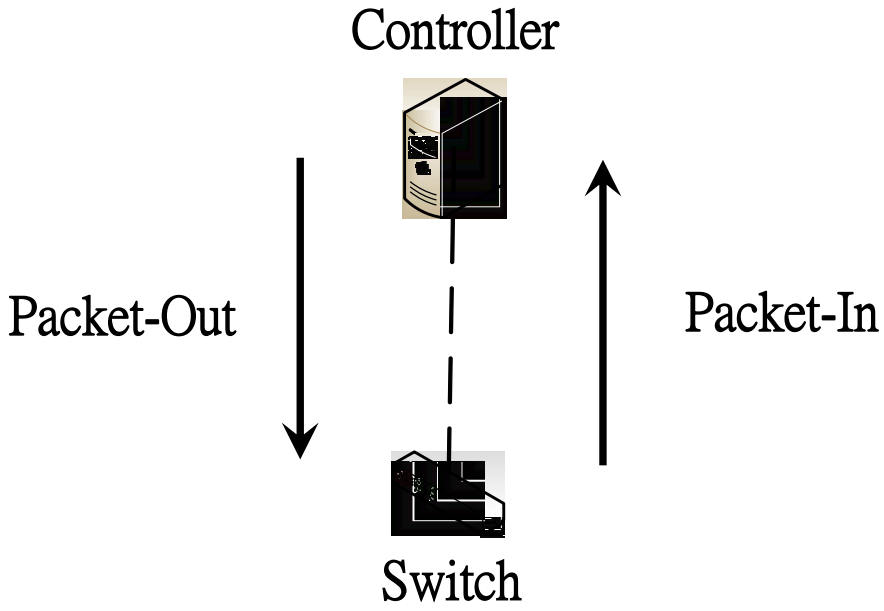
Ryu(simple_switch)

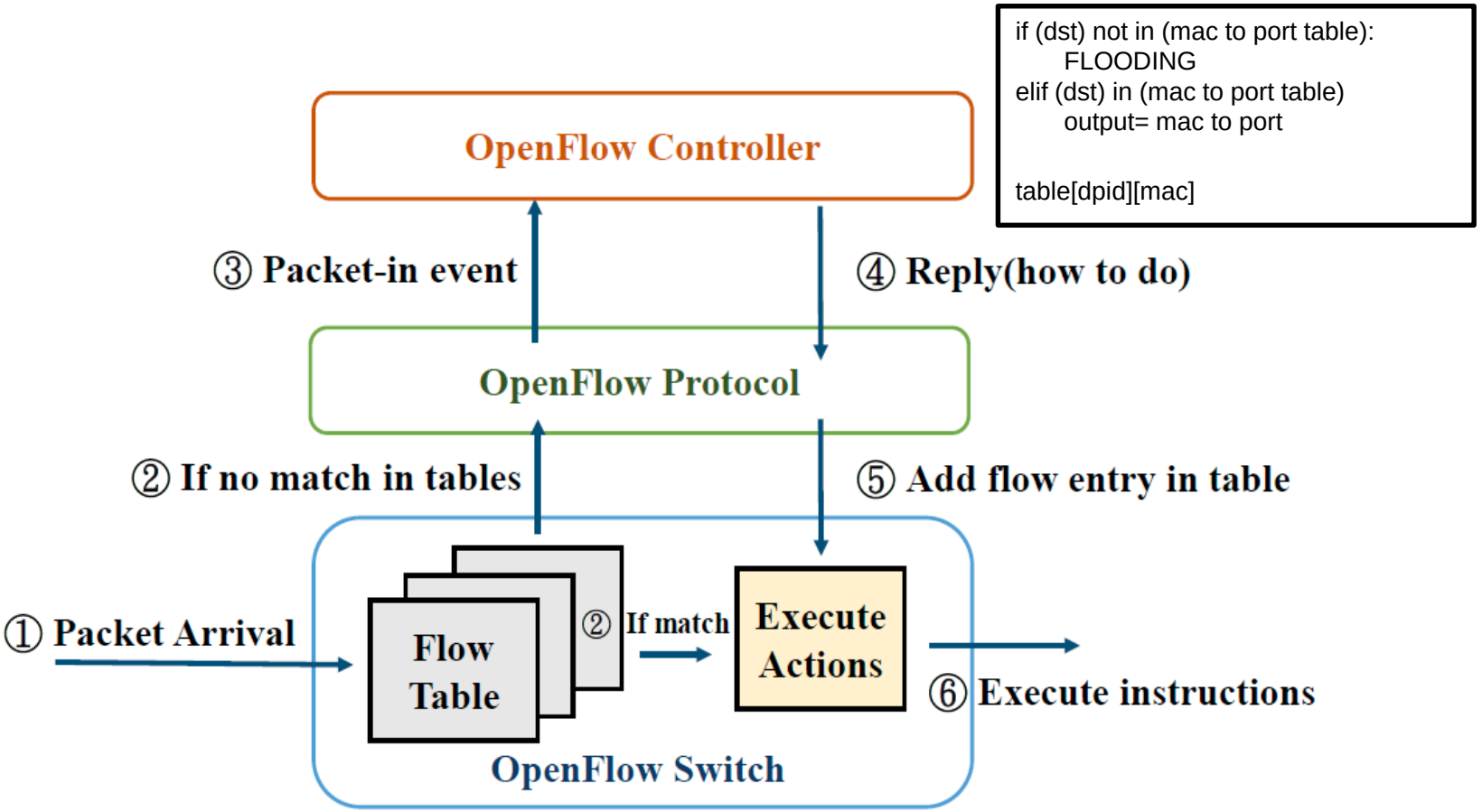
- Ryu example
  - 路徑: home/ryu/ryu/app
- 本實驗透過修改simple_switch_13.py達到實驗之目的。



Ryu(simple_switch)

- 本實驗Controller主要功用
  - 學習switch底下的host MAC 位址，並記錄在 MAC 位址表
    - 利用Packet-In message
  - 根據MAC 位址表，決定封包的routing
    - 利用Packet-Out message
  - 對於未指定目地位址的封包，則執行 Flooding




if (dst) not in (mac to port table):
FLOODING
elif (dst) in (mac to port table)
output= mac to port
table[dpid][mac]
OpenFlow Controller
③ Packet-in event
④ Reply(how to do)
OpenFlow Protocol
② If no match in tables
⑤ Add flow entry in table
① Packet Arrival
Flow Table
②
If match
Execute Actions
⑥ Execute instructions
OpenFlow Switch

Ryu(simple_switch)

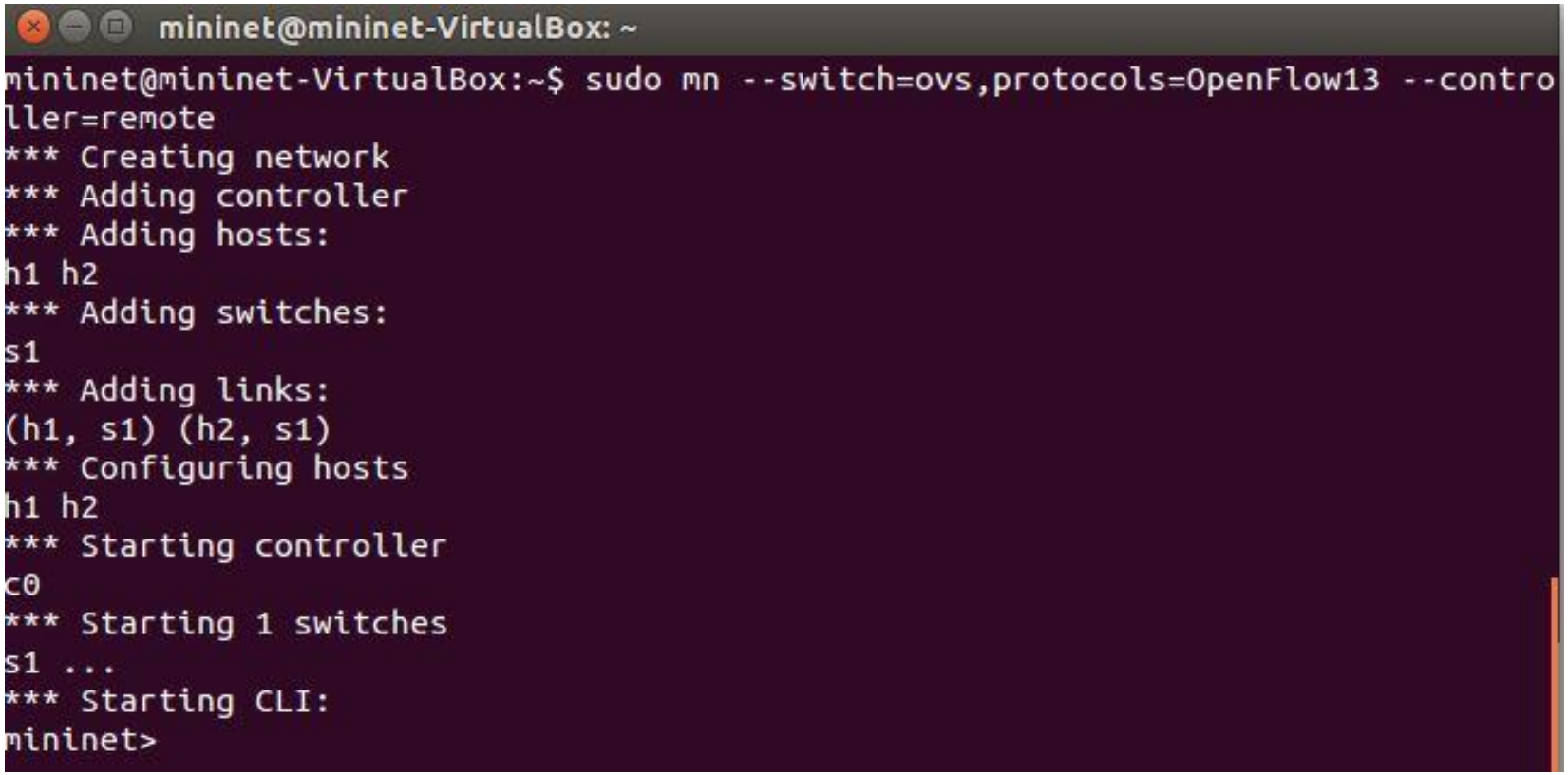
- 開啟終端機，執行Ryu應用程式
- 指令: ryu-manager ryu/ryu/app/simple_switch_13.py

- 若看到此畫面表示Ryu正在等待交換機的連接

```
mininet@mininet-VirtualBox: ~
mininet@mininet-VirtualBox:~$ ryu-manager ryu/ryu/app/simple_switch_13.py
loading app ryu/ryu/app/simple_switch_13.py
loading app ryu.controller.ofp_handler
instantiating app ryu/ryu/app/simple_switch_13.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Ryu(simple_switch)

- 開啟另一終端機
- 指令: sudo mn --switch=ovs,protocols=OpenFlow13 --controller=remote



Ryu(simple_switch)

- 回到Ryu Controller終端機
- 畫面上會看到Ryu不斷收到Packet-In封包，
  即表示控制器與交換機連接成功

```
packet in 1 96:b6:19:81:a8:3c 33:33:ff:81:a8:3c 1
packet in 1 a6:bf:88:74:81:41 33:33:ff:0b:b3:c0 4294967294
packet in 1 62:45:23:4c:1e:18 33:33:00:00:00:16 2
packet in 1 62:45:23:4c:1e:18 33:33:ff:4c:1e:18 2
packet in 1 a6:bf:88:74:81:41 33:33:00:00:00:16 4294967294
packet in 1 96:b6:19:81:a8:3c 33:33:00:00:00:16 1
packet in 1 96:b6:19:81:a8:3c 33:33:00:00:00:02 1
packet in 1 a6:bf:88:74:81:41 33:33:00:00:00:16 4294967294
packet in 1 a6:bf:88:74:81:41 33:33:00:00:00:02 4294967294
packet in 1 62:45:23:4c:1e:18 33:33:00:00:00:02 2
packet in 1 62:45:23:4c:1e:18 33:33:00:00:00:16 2
packet in 1 a6:bf:88:74:81:41 33:33:00:00:00:16 4294967294
packet in 1 96:b6:19:81:a8:3c 33:33:00:00:00:16 1
packet in 1 62:45:23:4c:1e:18 33:33:00:00:00:16 2
```

Ryu (Event handler)

規則為 ryu.controller.ofp_event.EventOFP + <OpenFlow訊息名稱>

Ex:

Packet_in 事件

```
@set_ev_cls(ofp_event.EventOFPPacketIn,
MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
    msg = ev.msg
    datapath = msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser
```

Related API reference:
https://ryu.readthedocs.io/en/latest/ofproto_v1_3_ref.html#asynchronous-messages

Ryu(simple_switch)

- 以下說明simple_switch_13.py中的程式碼
- Ryu與交換機完成握手協議後，會執行以下的函式。

```
@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
def switch_features_handler(self, ev):
    datapath = ev.msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser
```

  - ev.msg: 儲存OpenFlow 訊息類別的實體對應事件
  - msg.datapath: 儲存 OpenFlow 交換器的實體
  - datapath: 處理 OpenFlow 交換器重要的訊息

Ryu(simple_switch)

- 以下函式會在握手協議完成之後，
  新增一個Table-miss Flow Entry
  - 本節實驗修改此函式

```
def switch_features_handler(self, ev):

    # ...

    # install table-miss flow Entry
    #
    # We specify NO BUFFER to max_len of the output action due to
    # OVS bug. At this moment, if we specify a lesser number, e.g.,
    # 128, OVS will send Packet-In with invalid buffer_id and
    # truncated packet data. In that case, we cannot output packets
    # correctly.
    match = parser.OFPMatch()
    actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
                                      ofproto.OFPCML_NO_BUFFER)]
    self.add_flow(datapath, 0, match, actions)
```

Ryu(simple_switch)

- 加入規則的函式

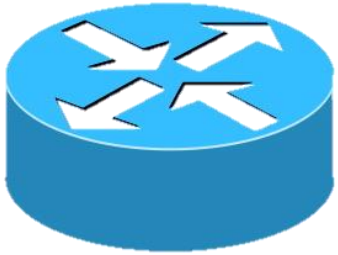
```
def add_flow(self, datapath, priority, match, actions):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                         actions)]

    mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                            match=match, instructions=inst)
    datapath.send_msg(mod)
```

  - datapath.send_msg(): 將規則送到指定交換機上

Ryu(simple_switch)

- 最後為接收交換機所發送的Packet-In封包處理函式

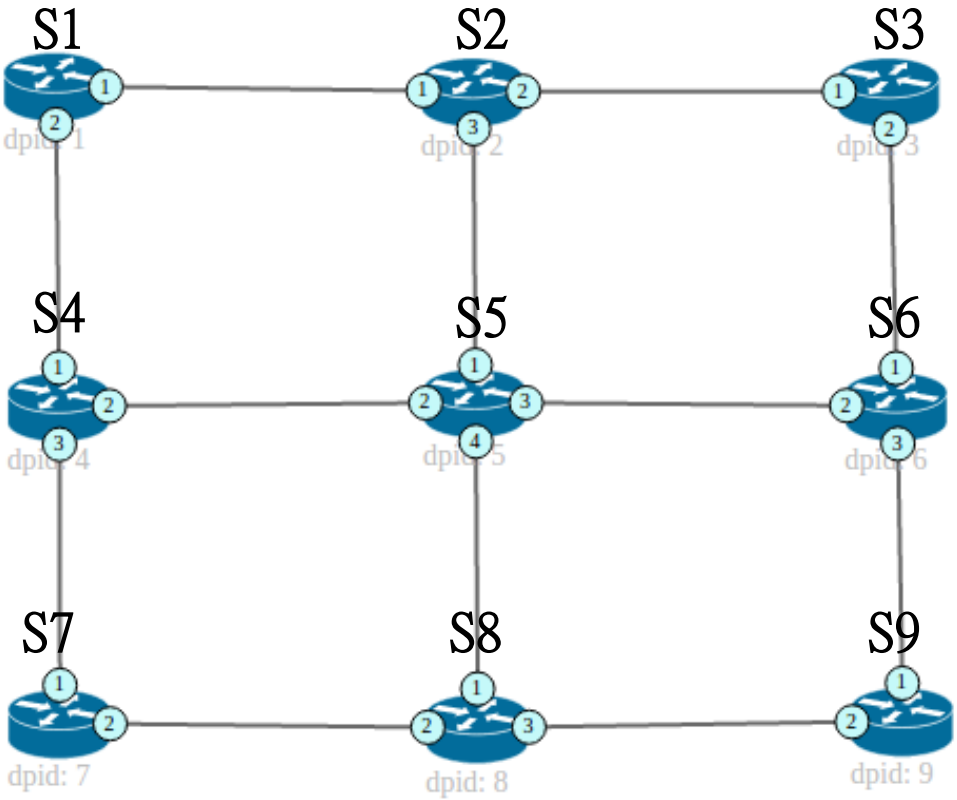
```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
    msg = ev.msg
    datapath = msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser
```



Lab 2-1: L3 routing

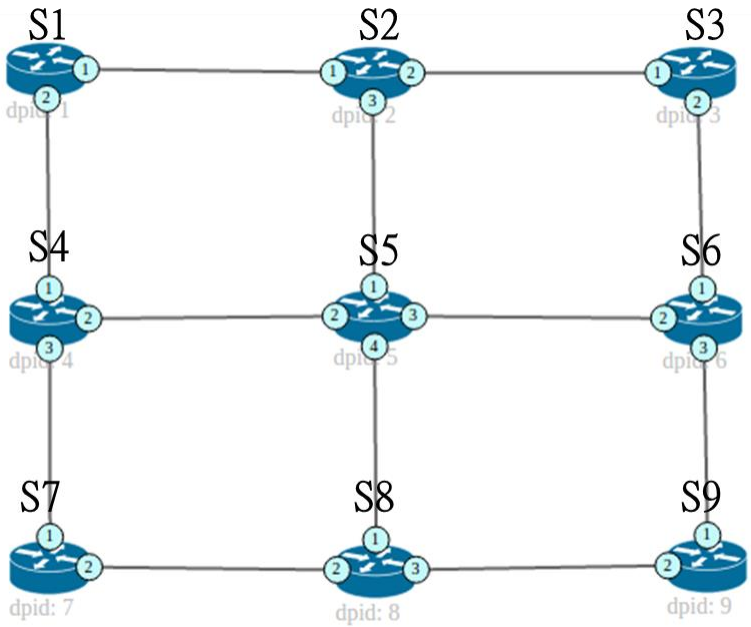
Lab 2-1: L3 routing

- 採用自訂拓樸(路徑:mininet/custom/grid_3x3.py)
  - 3x3 grid拓樸



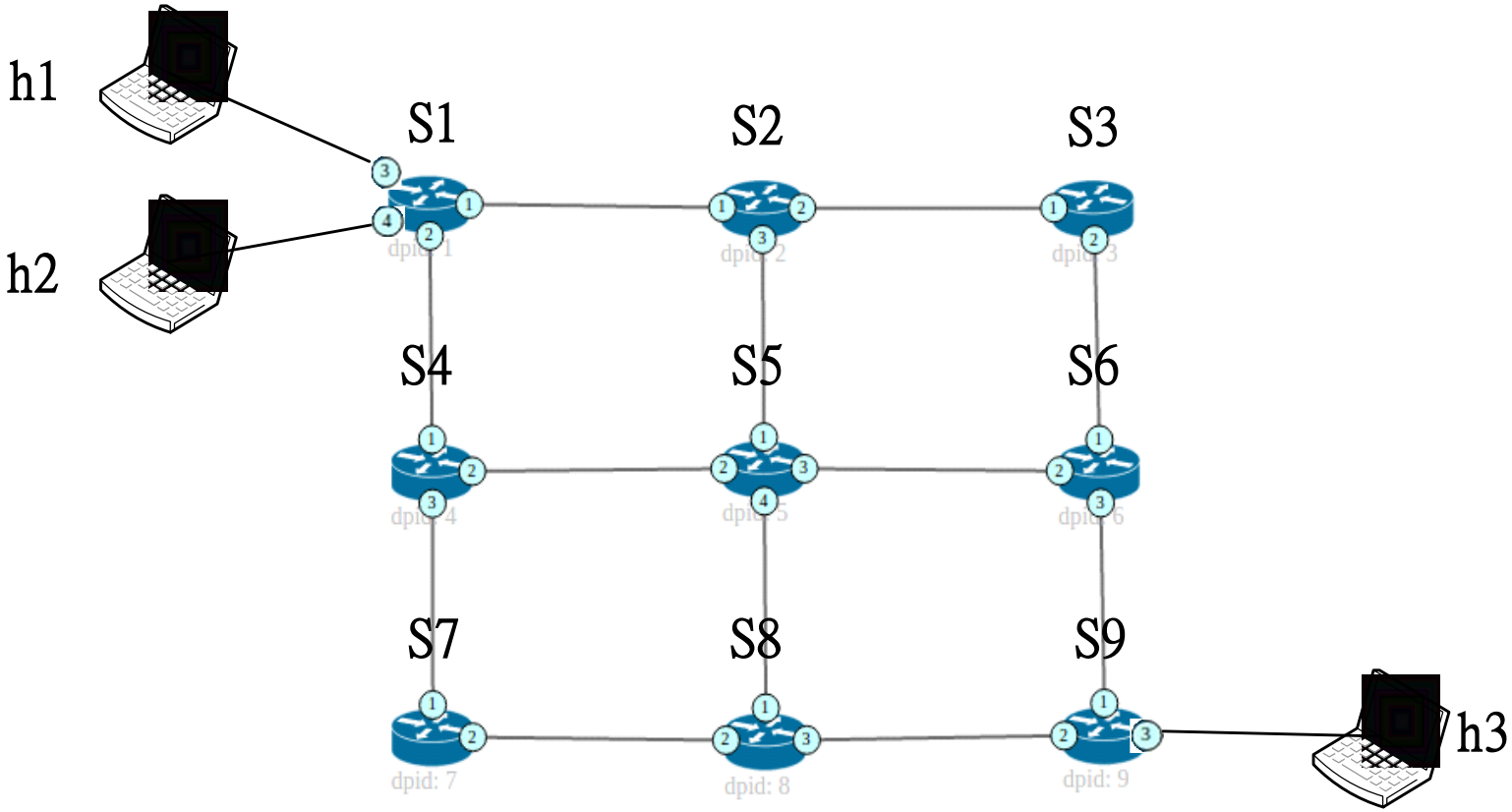
Lab 2-1: L3 routing

- 設定arp table
  - xterm h1
    - arp -s 10.0.0.2 00:00:00:00:00:02
    - arp -s 10.0.0.3 00:00:00:00:00:03
  - xterm h2
    - arp -s 10.0.0.1 00:00:00:00:00:01
    - arp -s 10.0.0.3 00:00:00:00:00:03
  - xterm h3
    - arp -s 10.0.0.2 00:00:00:00:00:02
    - arp -s 10.0.0.1 00:00:00:00:00:01



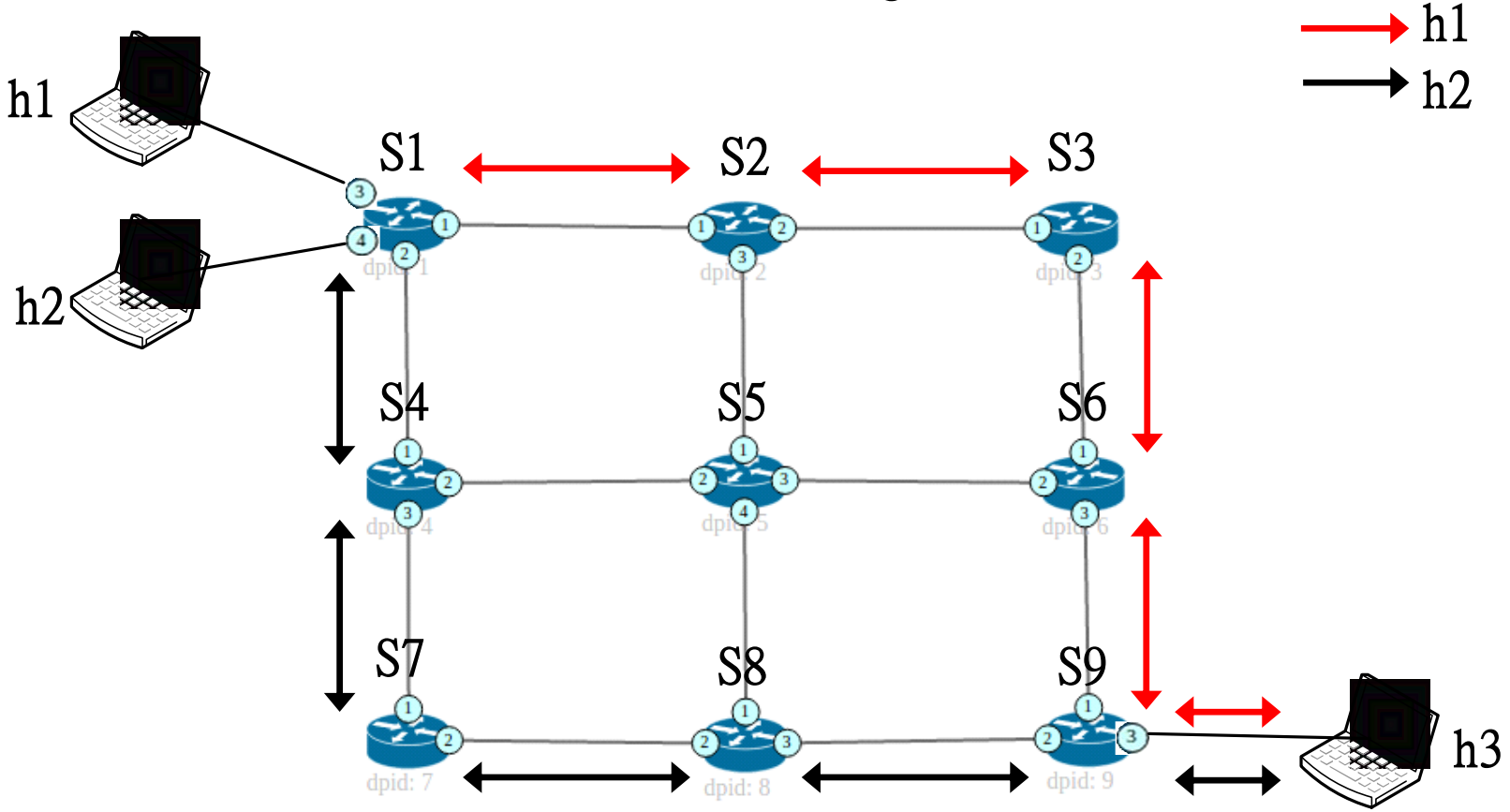
Lab 2-1: L3 routing

- 使用Ryu控制器對switch下路由規則。
- 在拓樸上有三台host分別為h1,h2和h3



Lab 2-1: L3 routing

- 以IP位址作為match的條件達成L3 routing。



Lab 2-1: L3 routing

- Step 1: 進入到Ryu的app資料夾底下，
  - 複製l2_routing.py檔案
  - 重新命名 l3_routing.py
  - 修改以下部分

```
class l3Routing(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(l3Routing, self).__init__(*args, **kwargs)
        self.mac_to_port = {}

    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
    def switch_features_handler(self, ev):
        datapath = ev.msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
        dpid = datapath.id
```

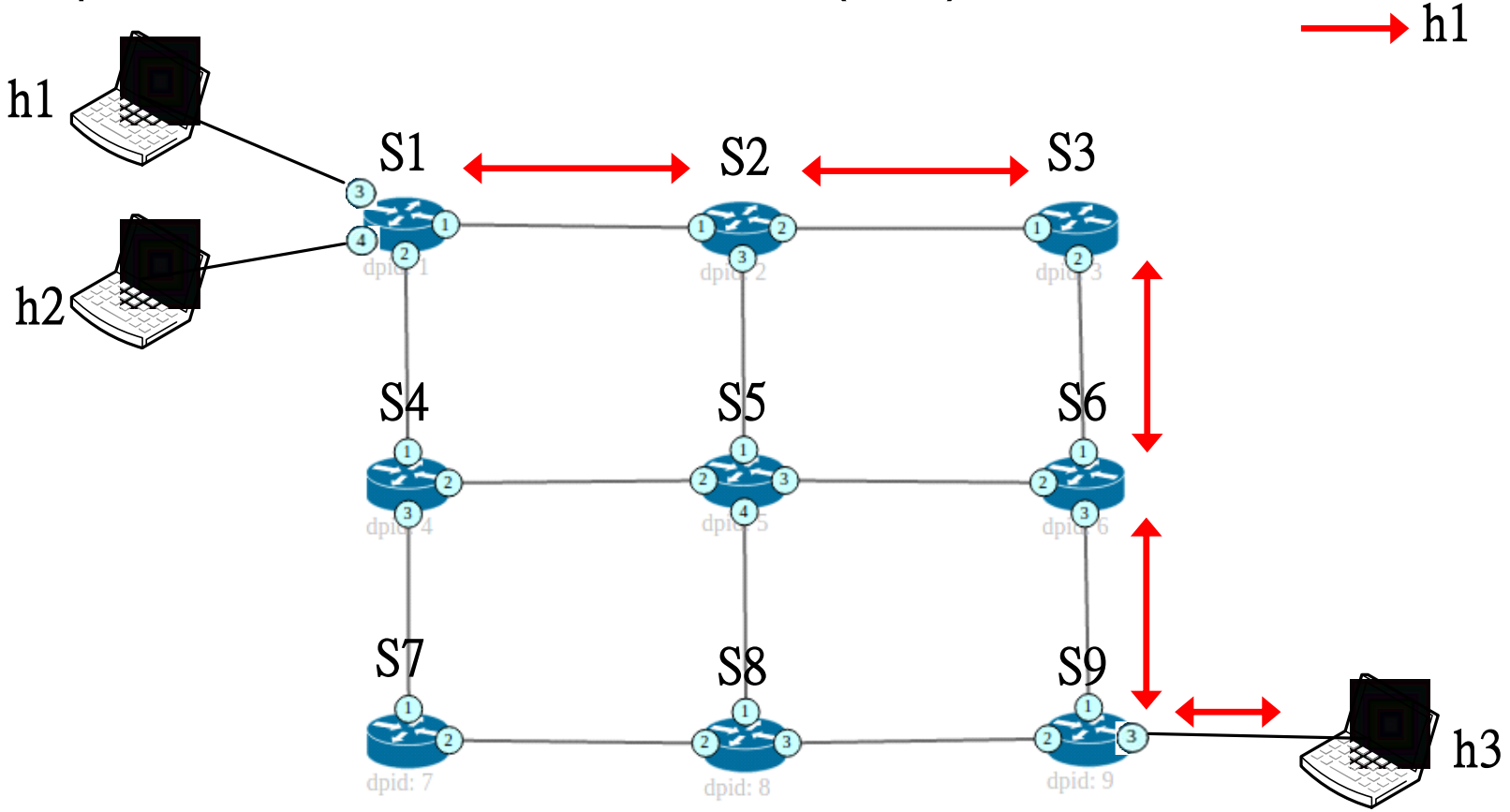
Lab 2-1: L3 routing

- Step 2:在握手協議完成之後加入路由規則。

```
class l3Routing(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(l3Routing, self).__init__(*args, **kwargs)
        self.mac_to_port = {}

    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
    def switch_features_handler(self, ev):
        datapath = ev.msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
        dpid = datapath.id
```

Lab 2-1: L3 routing

- Step 3: 首先先完成h1->h3的路徑設定(紅色)



Lab 2-1: L3 routing

- Step 3: 對S1,S2,S3,S6,S9加入路由規則。

- S1的規則:

```
if dpid == 1:
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.1', ipv4_dst = '10.0.0.3', eth_type=0x800)
            , [parser.OFPActionOutput(1)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.3', ipv4_dst = '10.0.0.1', eth_type=0x800)
            , [parser.OFPActionOutput(3)])
```

Lab 2-1: L3 routing

- S2的規則:

```
elif dpid == 2:
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.1', ipv4_dst = '10.0.0.3', eth_type=0x800)
            , [parser.OFPActionOutput(2)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.3', ipv4_dst = '10.0.0.1', eth_type=0x800)
            , [parser.OFPActionOutput(1)])
```

- S3的規則:

```
elif dpid == 3:
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.1', ipv4_dst = '10.0.0.3', eth_type=0x800)
            , [parser.OFPActionOutput(2)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.3', ipv4_dst = '10.0.0.1', eth_type=0x800)
            , [parser.OFPActionOutput(1)])
```

Lab 2-1: L3 routing

- S6的規則:

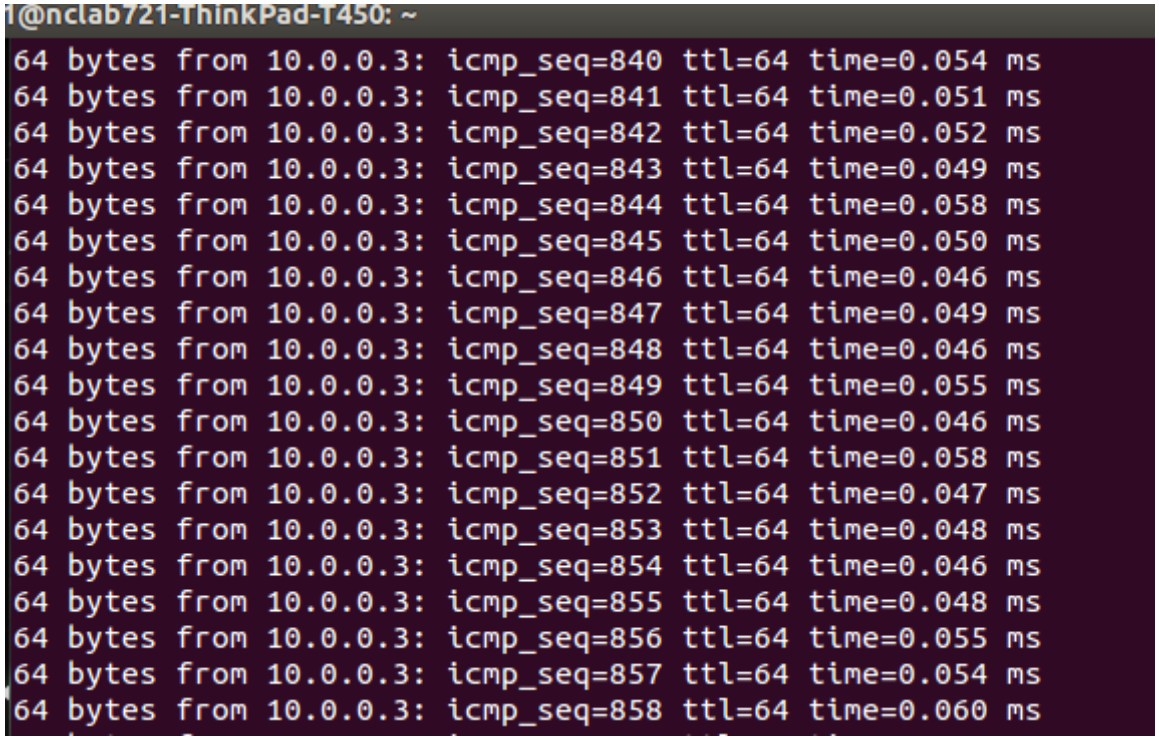
```
elif dpid == 6:
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.1', ipv4_dst = '10.0.0.3', eth_type=0x800)
            , [parser.OFPActionOutput(3)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.3', ipv4_dst = '10.0.0.1', eth_type=0x800)
            , [parser.OFPActionOutput(1)])
```

- S9的規則:

```
elif dpid == 9:
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.1', ipv4_dst = '10.0.0.3', eth_type=0x800)
            , [parser.OFPActionOutput(3)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.3', ipv4_dst = '10.0.0.1', eth_type=0x800)
            , [parser.OFPActionOutput(1)])
```

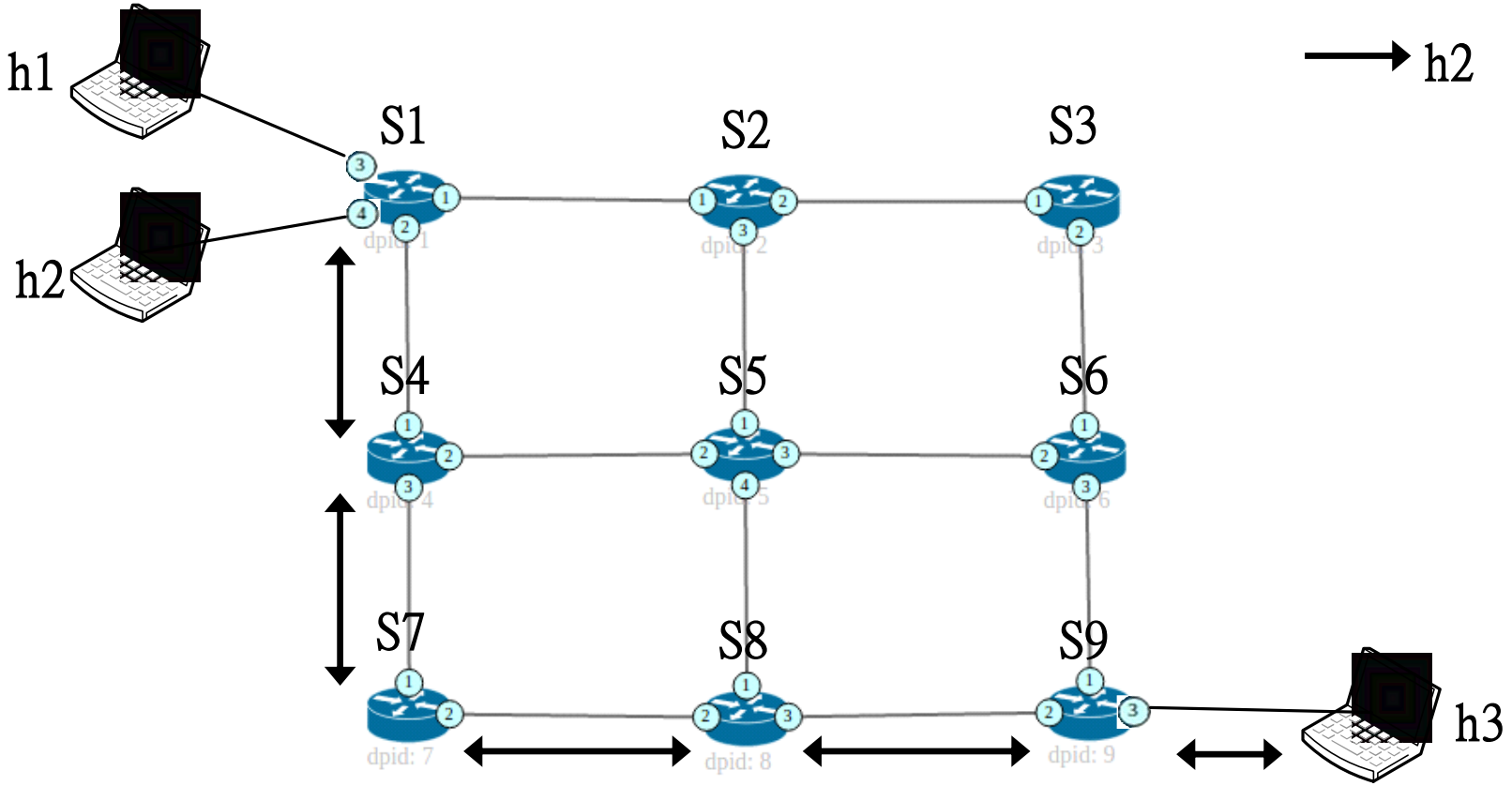
Lab 2-1: L3 routing

- 完成設定後查看各個switch裡的規則
  - ovs-ofctl -O OpenFlow13 dump-flows [switch name]
  - 執行指令:ryu-manager ryu/ryu/app/l3_routing.py
- 使用h1 ping h3測試



Lab 2-1: L3 routing

- Step 4: 完成h2<->h3的路徑設定(黑色)



Lab 2-1: L3 routing

- Step 4: 對S1,S4,S7,S8,S9加入路由規則。

- S1的規則:

```
self.add_flow(datapath, 1,
        parser.OFPMatch(ipv4_src = '10.0.0.2', ipv4_dst = '10.0.0.3', eth_type=0x800)
        , [parser.OFPActionOutput(2)])
self.add_flow(datapath, 1,
        parser.OFPMatch(ipv4_src = '10.0.0.3', ipv4_dst = '10.0.0.2', eth_type=0x800)
        , [parser.OFPActionOutput(4)])
```

Lab 2-1: L3 routing

- S4的規則:

```
elif dpid == 4:
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.2', ipv4_dst = '10.0.0.3', eth_type=0x800)
            , [parser.OFPActionOutput(3)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.3', ipv4_dst = '10.0.0.2', eth_type=0x800)
            , [parser.OFPActionOutput(1)])
```

- S7的規則:

```
elif dpid == 7:
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.2', ipv4_dst = '10.0.0.3', eth_type=0x800)
            , [parser.OFPActionOutput(2)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.3', ipv4_dst = '10.0.0.2', eth_type=0x800)
            , [parser.OFPActionOutput(1)])
```

Lab 2-1: L3 routing

- S8的規則:

```
elif dpid == 8:
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.2', ipv4_dst = '10.0.0.3', eth_type=0x800)
            , [parser.OFPActionOutput(3)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(ipv4_src = '10.0.0.3', ipv4_dst = '10.0.0.2', eth_type=0x800)
            , [parser.OFPActionOutput(2)])
```

- S9的規則:

```
self.add_flow(datapath, 1,
        parser.OFPMatch(ipv4_src = '10.0.0.2', ipv4_dst = '10.0.0.3', eth_type=0x800)
        , [parser.OFPActionOutput(3)])
self.add_flow(datapath, 1,
        parser.OFPMatch(ipv4_src = '10.0.0.3', ipv4_dst = '10.0.0.2', eth_type=0x800)
```

Lab 2-1: L3 routing

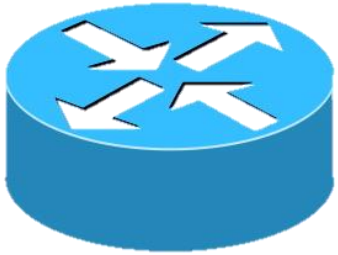
- 完成設定後查看各個switch裡的規則
  - ovs-ofctl –O OpenFlow13 dump-flows [switch name]
  - 執行指令:ryu-manager ryu/ryu/app/l3_routing.py
- 使用h2 ping h3或h1 ping h3測試

```
l@nclab721-ThinkPad-T450: ~
64 bytes from 10.0.0.3: icmp_seq=150 ttl=64 time=0.074 ms
64 bytes from 10.0.0.3: icmp_seq=151 ttl=64 time=0.082 ms
64 bytes from 10.0.0.3: icmp_seq=152 ttl=64 time=0.072 ms
64 bytes from 10.0.0.3: icmp_seq=153 ttl=64 time=0.074 ms
64 bytes from 10.0.0.3: icmp_seq=154 ttl=64 time=0.086 ms
64 bytes from 10.0.0.3: icmp_seq=155 ttl=64 time=0.071 ms
64 bytes from 10.0.0.3: icmp_seq=156 ttl=64 time=0.078 ms
64 bytes from 10.0.0.3: icmp_seq=157 ttl=64 time=0.097 ms
64 bytes from 10.0.0.3: icmp_seq=158 ttl=64 time=0.084 ms
64 bytes from 10.0.0.3: icmp_seq=159 ttl=64 time=0.074 ms
64 bytes from 10.0.0.3: icmp_seq=160 ttl=64 time=0.094 ms
64 bytes from 10.0.0.3: icmp_seq=161 ttl=64 time=0.066 ms
64 bytes from 10.0.0.3: icmp_seq=162 ttl=64 time=0.078 ms
64 bytes from 10.0.0.3: icmp_seq=163 ttl=64 time=0.086 ms
64 bytes from 10.0.0.3: icmp_seq=164 ttl=64 time=0.082 ms
64 bytes from 10.0.0.3: icmp_seq=165 ttl=64 time=0.079 ms
```

Lab 2-1: L3 routing

- 輸入pingall測試
  - h1->h3與h2->h3與h3->h1 h2是通的
  - 但是h1->h2是不通的，因為我們沒有下規則給h1與h2。

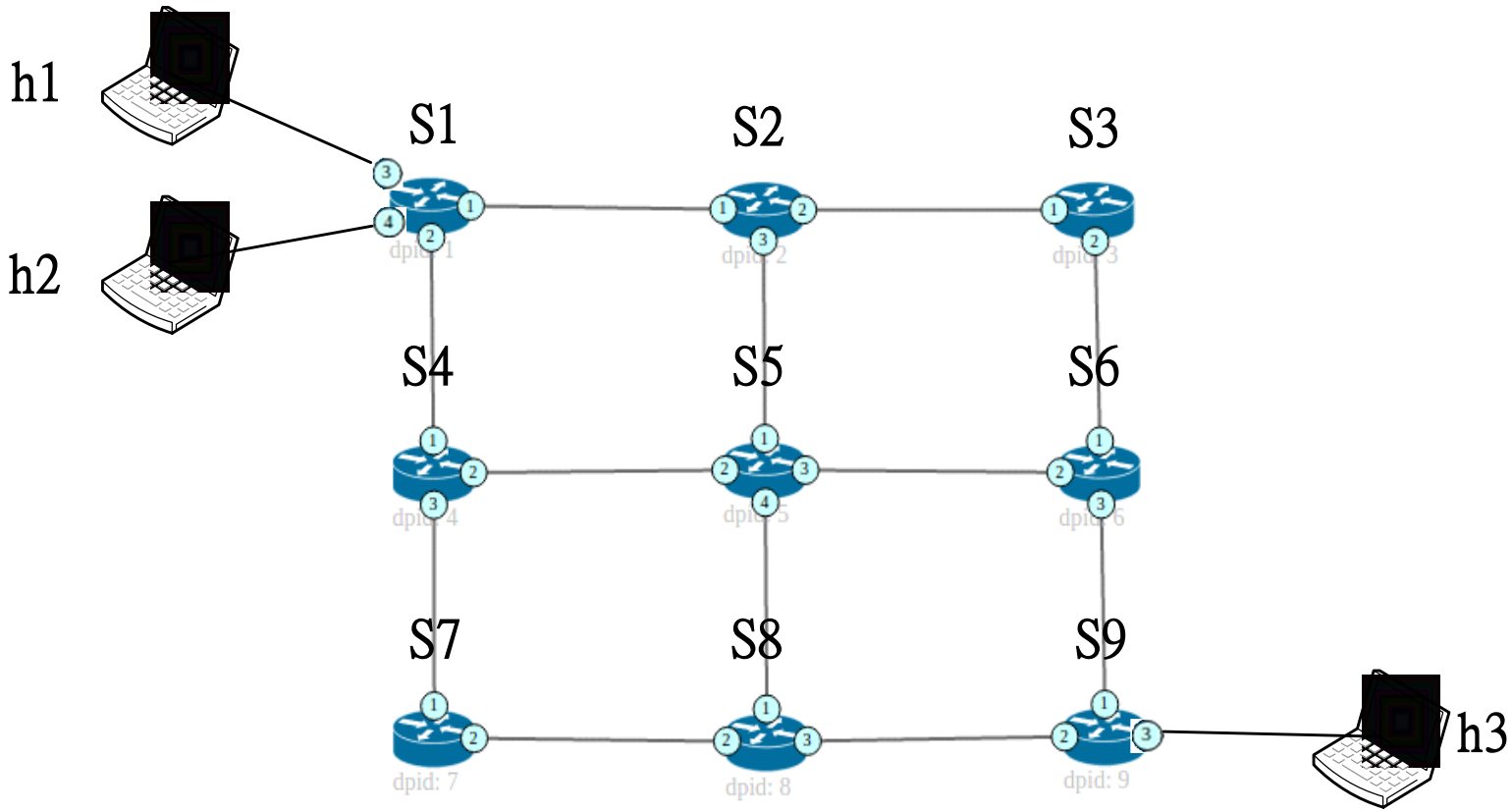
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3
h2 -> X h3
h3 -> h1 h2
*** Results: 33% dropped (4/6 received)
mininet>
```



Lab 2-2: Firewall

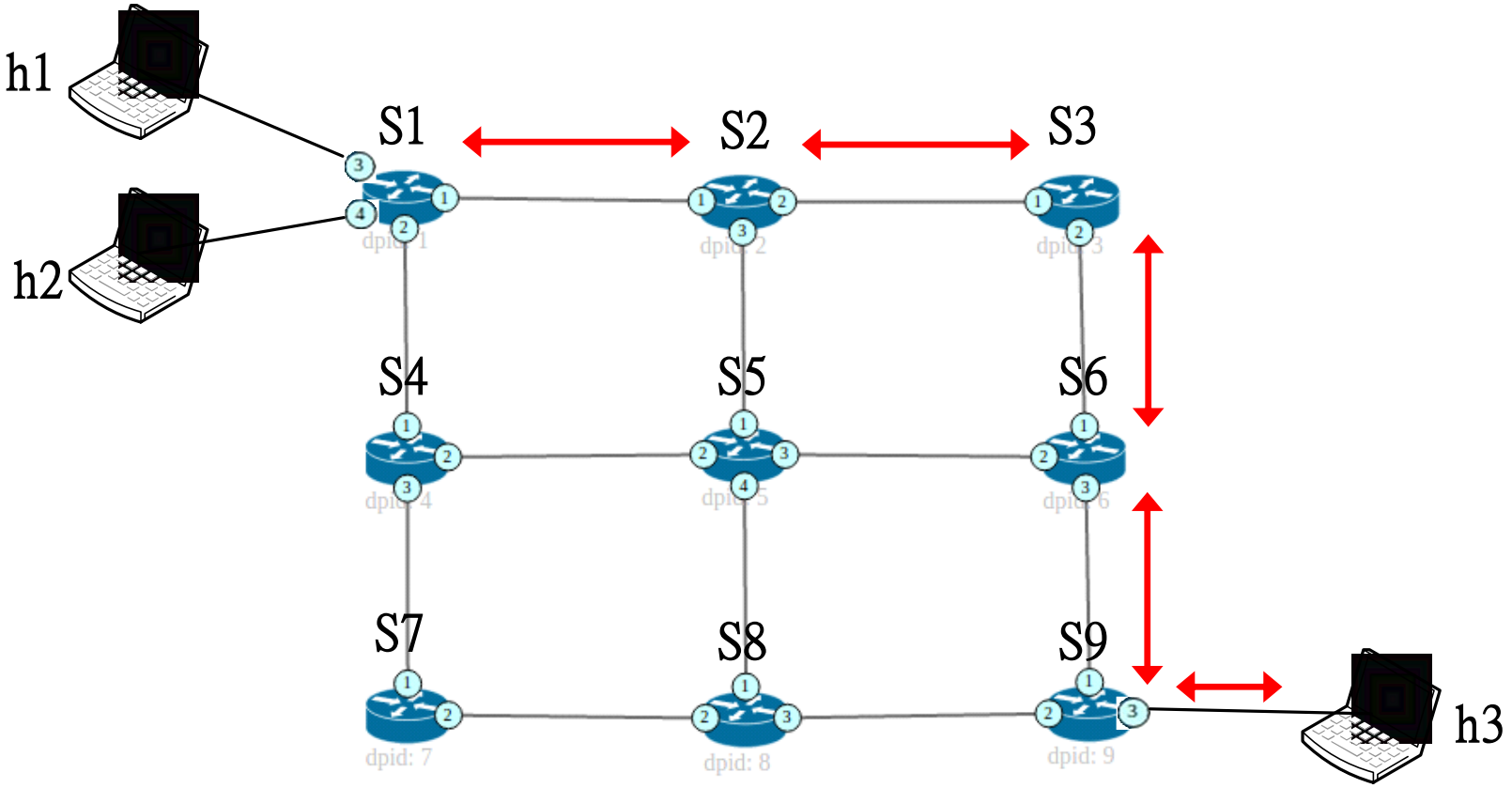
Lab 2-2: Firewall

- 目的:讓h2的封包無法通過S1



Lab 2-2: Firewall

- Step 1: 首先輸入規則讓h1,h2都能夠ping到h3，路徑如下



Lab 2-2: Firewall

- Step 1: 對S1,S2,S3,S6,S9加入路由規則。
- S1的規則:

```
if dpid == 1:
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:03')
            , [parser.OFPActionOutput(1)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:01')
            , [parser.OFPActionOutput(3)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:02')
            , [parser.OFPActionOutput(4)])
```

- S2的規則:

```
elif dpid == 2:
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:03')
            , [parser.OFPActionOutput(2)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:01')
            , [parser.OFPActionOutput(1)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:02')
            , [parser.OFPActionOutput(1)])
```

Lab 2-2: Firewall

- S3的規則:

```
elif dpid == 3:
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:03')
            , [parser.OFPActionOutput(2)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:01')
            , [parser.OFPActionOutput(1)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:02')
            , [parser.OFPActionOutput(1)])
```

- S6的規則:

```
elif dpid == 6:
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:03')
            , [parser.OFPActionOutput(3)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:01')
            , [parser.OFPActionOutput(1)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:02')
            , [parser.OFPActionOutput(1)])
```

Lab 2-2: Firewall

- S9的規則:

```
elif dpid == 9:
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:03')
            , [parser.OFPActionOutput(3)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:01')
            , [parser.OFPActionOutput(1)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:02')
            , [parser.OFPActionOutput(1)])
```

- Step2: 輸入pingall測試

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

Lab 2-2: Firewall

- Step 3: 輸入路由規則讓S1丟棄h2的封包。
- 在S1的match條件裡
  - 修改host2的路由規則如下

```
elif dpid == 1:
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:03')
            , [parser.OFPActionOutput(2)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:01')
            , [parser.OFPActionOutput(1)])
    self.add_flow(datapath, 1,
            parser.OFPMatch(eth_dst = '00:00:00:00:00:02')
            , [])
```

Lab 2-2: Firewall

- Step4:再輸入一次pingall測試
  - h2無法ping到h3。
  - h2亦無法跟其他的host溝通。

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3
h2 -> X X
h3 -> h1 X
*** Results: 66% dropped (2/6 received)
```

Thank you