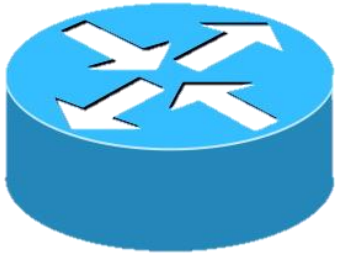


Mininet Labs

Speaker: Yi-Ta Chen

Advisor: Tsung-Nan Lin

Date: 20180423

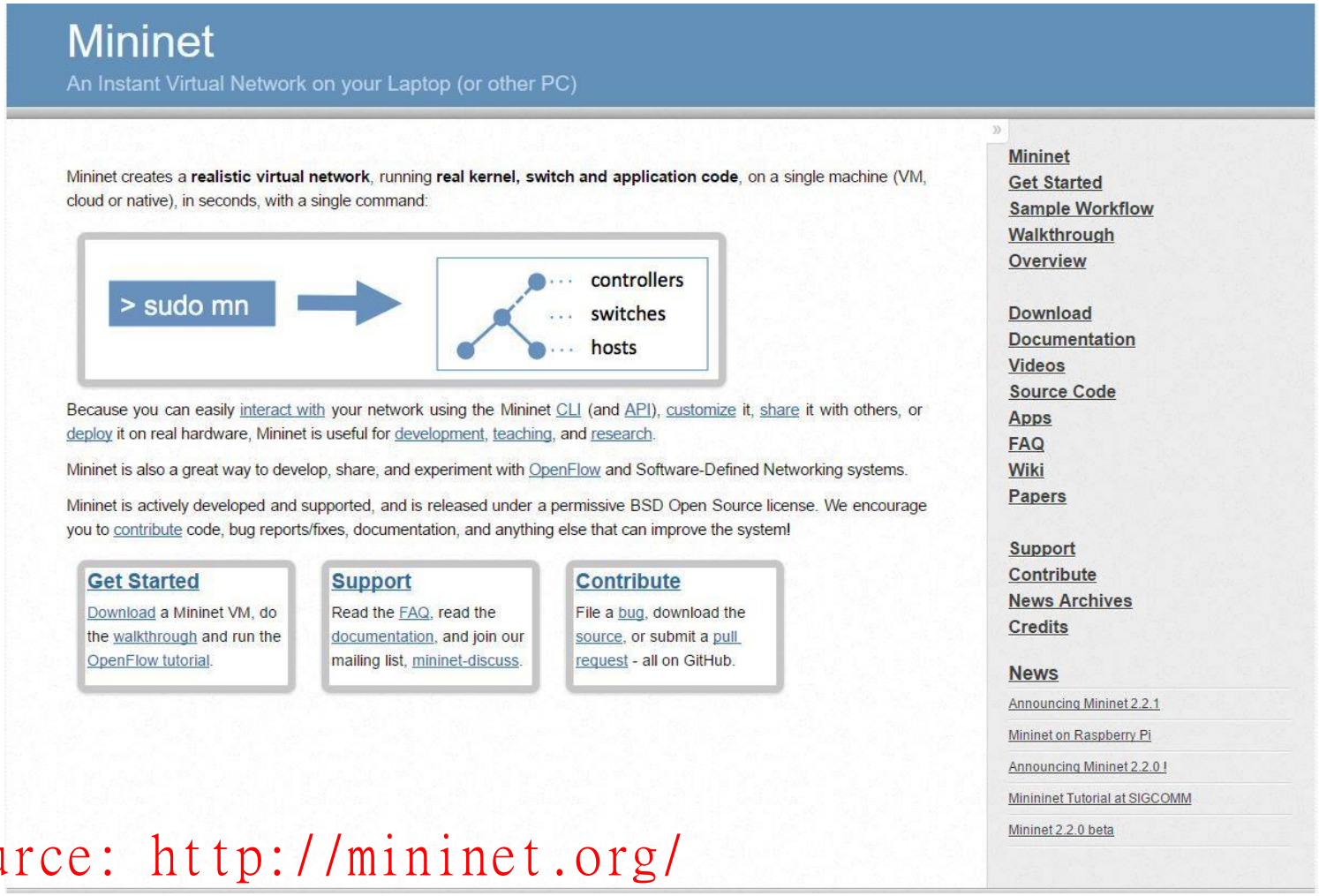
Outline

- Mininet 介紹
  - Mininet 安裝
  - Mininet 產生網路
  - Mininet 預設/自訂拓樸
  - Mininet 內建 GUI介面
- Lab1-1: L2 routing以MAC為match條件
- Lab1-2: L2 routing以IP為match條件
- Lab1-3: Firewall
- 附錄
  - Mininet 常用指令
  - Open vSwitch常用指令
  - Python語言教學影片

Mininet 介紹

- Mininet
  - 一個模擬網路的平台
  - 可以模擬各種網路環境
    - Ethernet, SDN
    - 模擬傳統switch, router
    - 模擬OpenFlow switch (Open vSwitch)
  - controller有Ryu或POX可選擇
    - 預設為Ryu
  - 可與真實的網路環境相連

Mininet 安裝

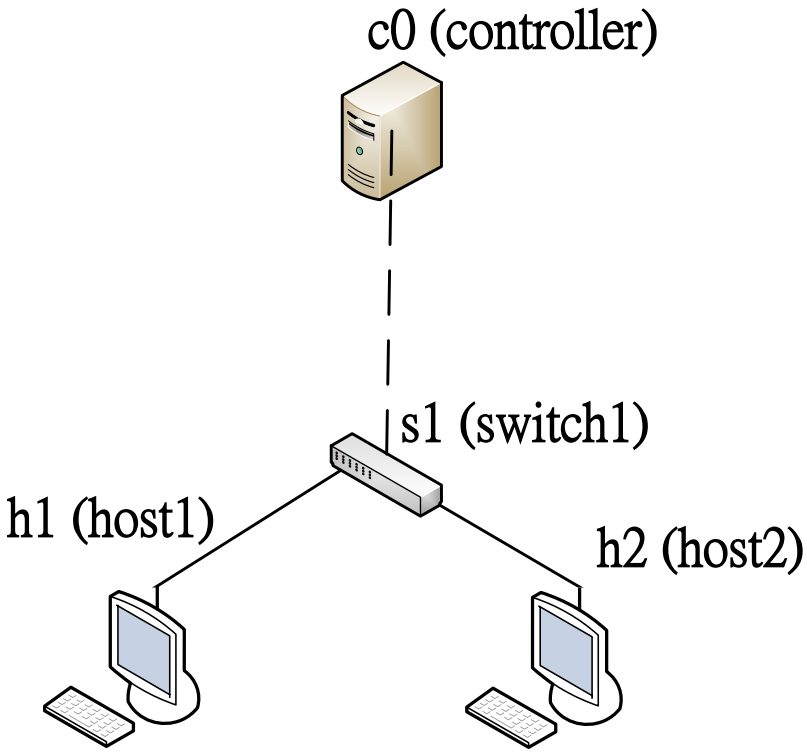


Source: http://mininet.org/

Mininet 安裝 – 匯入VM

- 輸入sudo mn (此為預設拓譜架構)

```
1@nclab721-ThinkPad-T450: ~
nclab721@nclab721-ThinkPad-T450:~$ sudo mn
[sudo] password for nclab721:
*** 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>
```



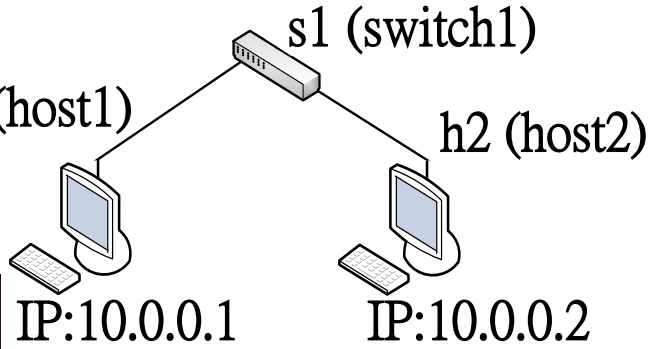
Mininet 產生網路

- h1 ping h2:確認h1和h2的裝置連線狀況

```
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=3.59 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.144 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.031 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.031 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.032 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.034 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.036 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.036 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.049 ms
```

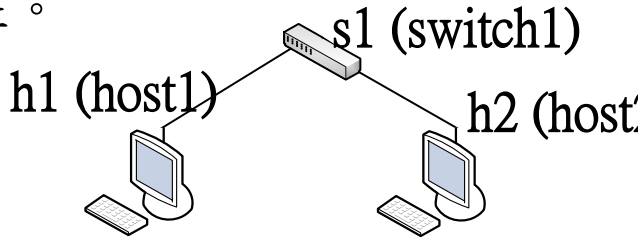
- pingall:確認拓譜網路的連線

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
```



Mininet 產生網路 – h1 ping h2

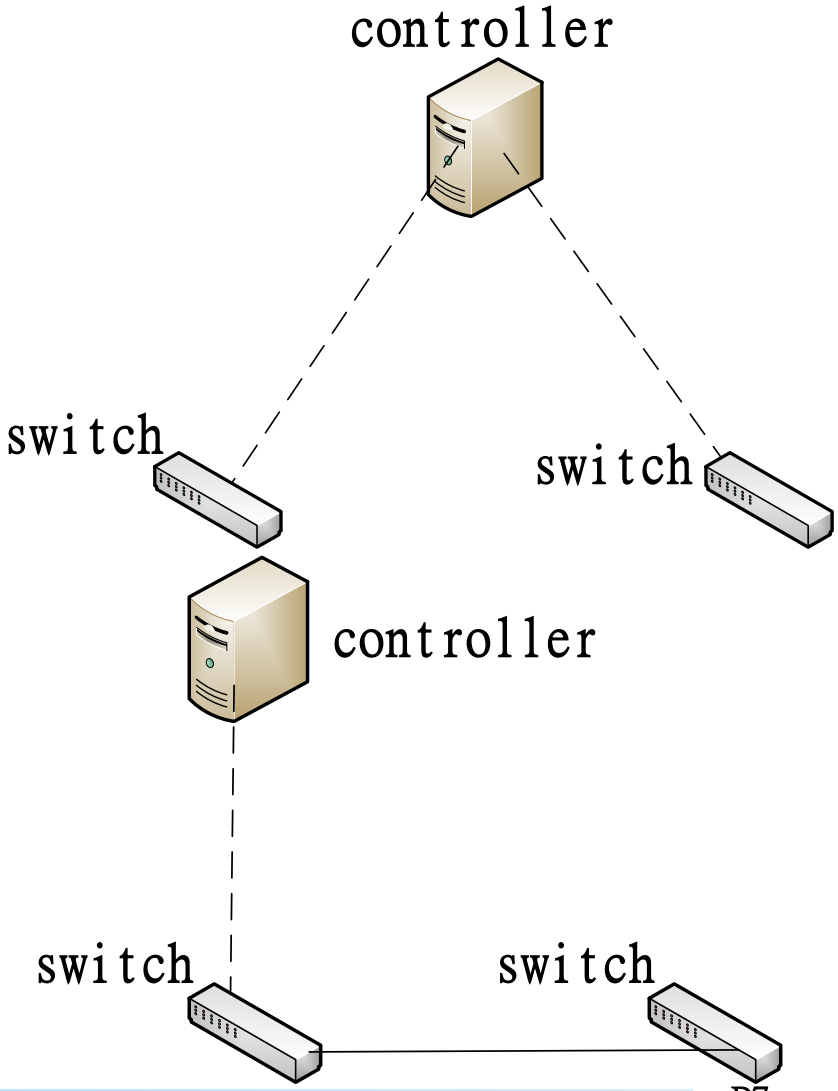
- Step1: ARP request
  h1不知道h2的MAC位址，所以ICMP echo request發送之前，h1會先利用ARP request的廣播封包詢問h2的MAC位址。
- Step2: ARP reply
  h2使用ARP reply回覆h1的要求。
- Step3: ICMP echo request
  現在h1知道h2的MAC位址，因此發送echo request給h2。
- Step4: ICMP echo reply
  h2 此時也知道h1的MAC位址，因此發送echo reply給h1。
- ARP: Address Resolution Protocol用於將IP位址對應到可在本機網路中辨識的實體電腦位址
- ICMP: Internet Control Message Protocol用於TCP/IP網絡中發送控制消息，提供可能發生在通信環境中的各種問題反饋



Mininet 產生網路 –連接Controller與Switch

1. Out-of-band
   - 簡單: 與交換機直接連線
   - 安全: 獨立的控制網路
   - Mininet預設為Out-of-band

2. In-band
   - 控制訊息與一般封包共存於相同網路
   - Switch不需專屬的port與Controller連接



Mininet 預設拓樸

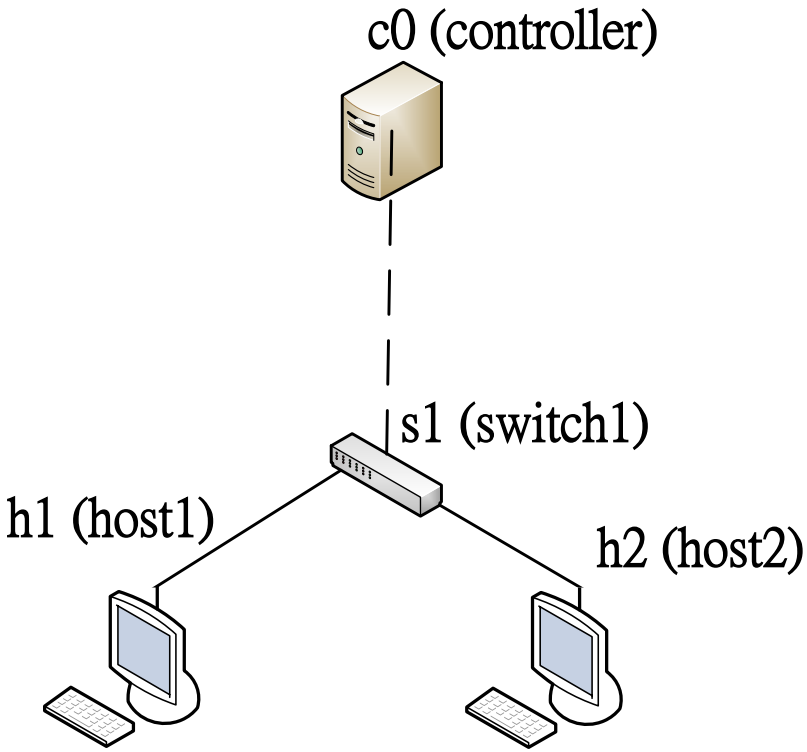
- 可以使用的預設拓樸
  - 顯示可使用的預設拓譜指令: sudo mn –h
  - linear, minimal, reversed, single, torus, tree

```
--topo=TOPO            linear|minimal|reversed|single|torus|tree[,param=value
                       ...] linear=LinearTopo torus=TorusTopo tree=TreeTopo
                       single=SingleSwitchTopo
                       reversed=SingleSwitchReversedTopo minimal=MinimalTopo
```

Mininet 預設拓樸

- 輸入sudo mn

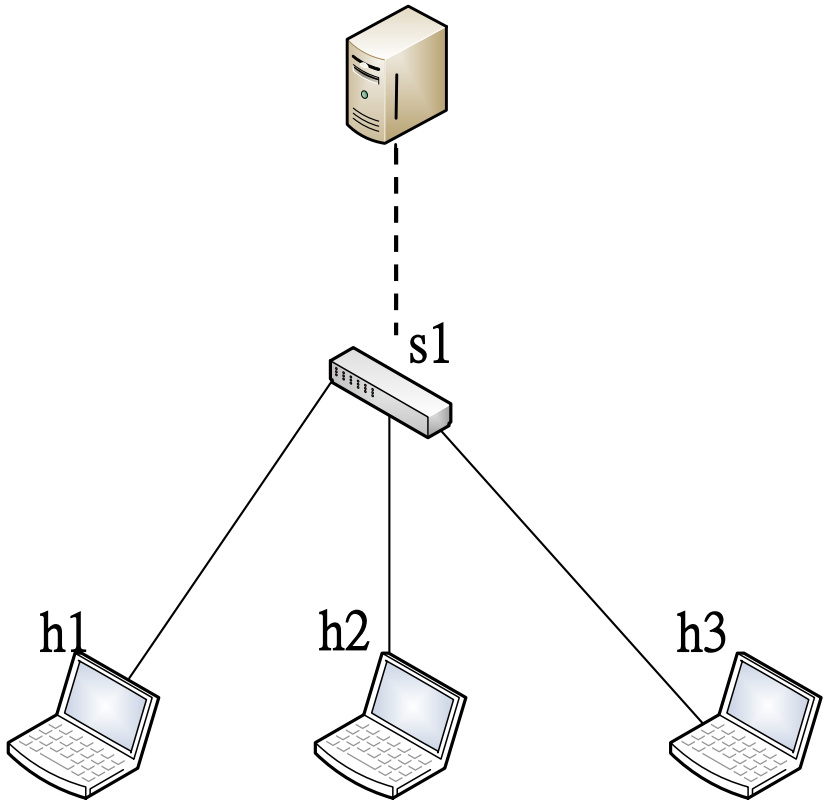
```
1@nclab721-ThinkPad-T450: ~
nclab721@nclab721-ThinkPad-T450:~$ sudo mn
[sudo] password for nclab721:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```



Mininet 預設拓樸 – 單一Switch

- 指令: sudo mn --topo=single,k
  - k為host個數 (h1,h2,h3…)。
- 輸入sudo mn --topo=single,3

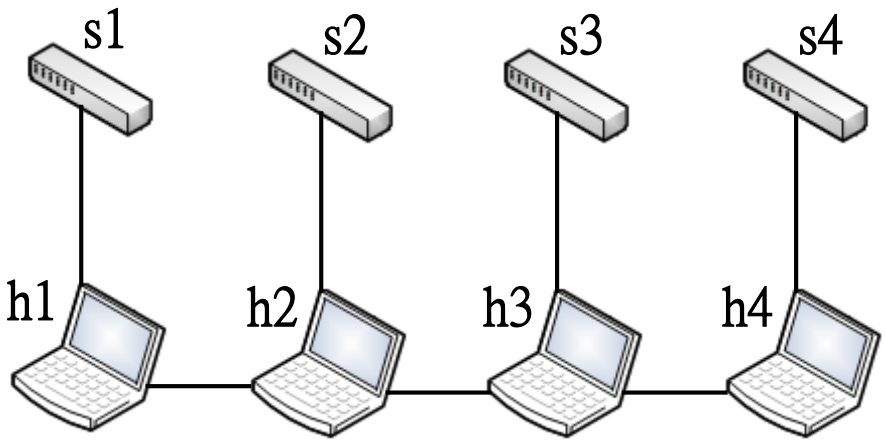
```
mininet@mininet-VirtualBox:~$ sudo mn --topo=single,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```



Mininet 預設拓樸 – simple linear topology

- 指令: sudo mn --topo=linear,n
  - n為Switch個數(s1,s2,s3,s4…), 每一Switch各有一台host。
- 輸入sudo mn --topo=linear,4

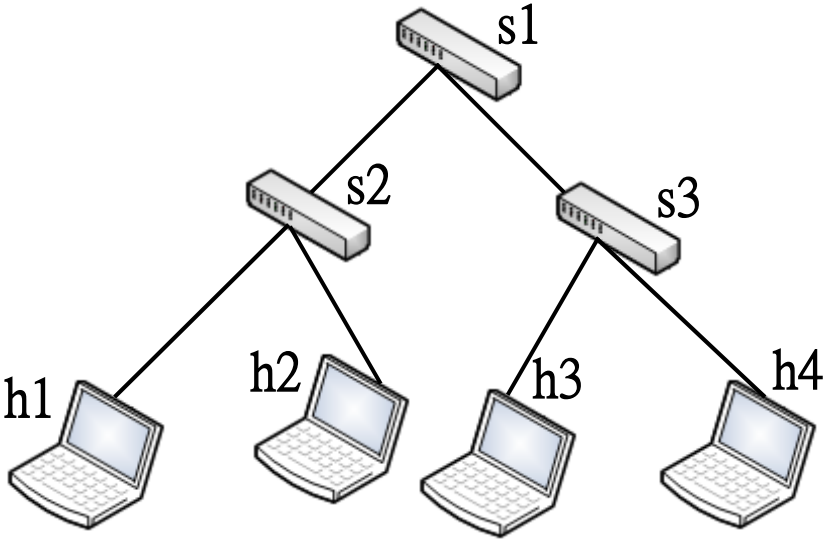
```
mininet@mininet-VirtualBox:~$ sudo mn --topo=linear,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s2, s1) (s3, s2)
(s4, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
```



Mininet 預設拓樸 – tree-like topology

- 指令: sudo mn --topo=tree,d,f
  - d=depth樹深(switch 深度), f=fanout為每個節點的分支數。
- 輸入sudo mn --topo=tree,2,2

```
mininet@mininet-VirtualBox:~$ sudo mn --topo=tree,2,2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
```



Mininet 自訂拓樸

- 使用python將Mininet的API寫成拓樸檔案

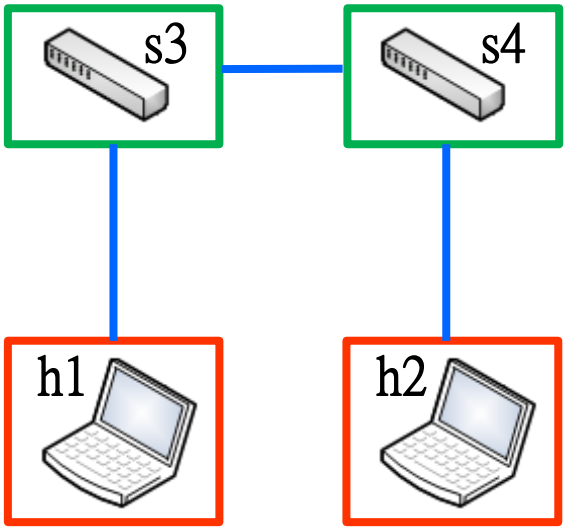
```
11  from mininet.topo import Topo
12
13  class MyTopo( Topo ):
14      "Simple topology example."
15
16      def __init__( self ):
17          "Create custom topo."
18
19          # Initialize topology
20          Topo.__init__( self )
21
22          # Add hosts and switches
23          leftHost = self.addHost( 'h1' )
24          rightHost = self.addHost( 'h2' )
25          leftSwitch = self.addSwitch( 's3' )
26          rightSwitch = self.addSwitch( 's4' )
27
28          # Add links
29          self.addLink( leftHost, leftSwitch )
30          self.addLink( leftSwitch, rightSwitch )
31          self.addLink( rightSwitch, rightHost )
32
33
34  topos = { 'mytopo': ( lambda: MyTopo() ) }
```

- 預設拓樸檔案
  路徑: Mininet/custom
  檔名: topo-2sw-2host.py

- 指令:
  sudo mn --custom 路徑/檔案名稱.py --topo mytopo --controller=none

Mininet 自訂拓樸

- topo-2sw-2host.py




```python
from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )

        # Add links
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Mininet 內建 GUI介面

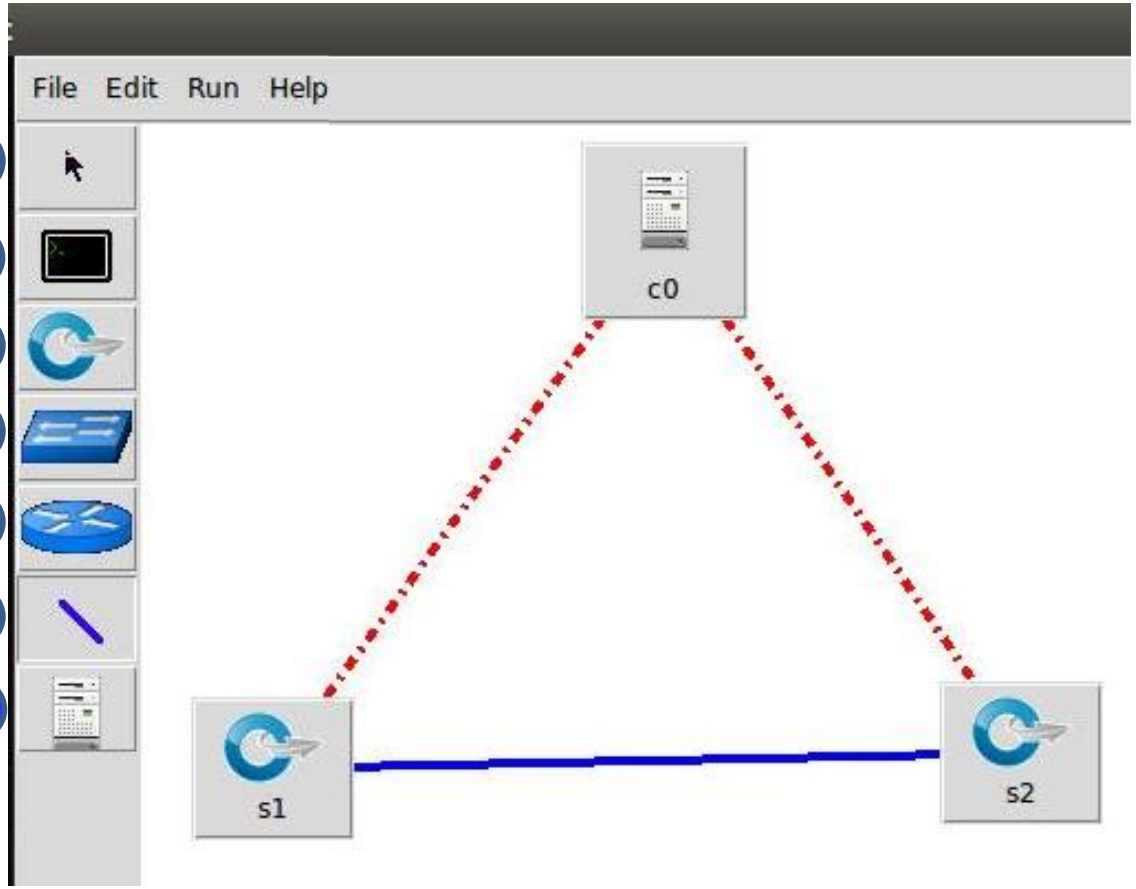
- Mininet提供GUI介面
  - 進入/mininet/example資料夾
  - 指令: sudo python miniedit.py

```
1@nclab721-ThinkPad-T450: ~/mininet/examples
nclab721@nclab721-ThinkPad-T450:~$ cd mininet/
nclab721@nclab721-ThinkPad-T450:~/mininet$ ls
bin  build  CONTRIBUTORS  custom  debian  dist  doc  examples  INSTALL  LICENSE  Makef
nclab721@nclab721-ThinkPad-T450:~/mininet$ cd examples/
nclab721@nclab721-ThinkPad-T450:~/mininet/examples$ ls
baresshd.py     cluster.py         controllers.py  hwintf.py       linearbandwidth.py
bind.py         clusterSanity.py   controlnet.py   __init__.py     linuxrouter.py
clustercli.py   consoles.py        cpu.py          intfoptions.py  miniedit.py
clusterdemo.py  controllers2.py    emptynet.py     limit.py        mobility.py
nclab721@nclab721-ThinkPad-T450:~/mininet/examples$ sudo python miniedit.py
[sudo] password for nclab721:
MiniEdit running against Mininet 2.2.1
topo=none
```

Mininet 內建 GUI介面

- 左邊的列表中由上到下分別為

1. 指標
2. host
3. OpenFlow 交換機
4. 傳統交換機
5. 路由器
6. 線路
7. 控制器



Other command in Mininet

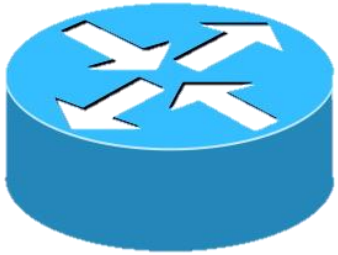
```
        host = self.addHost('h%s' % i, cpu=.5/k)
        switch = self.addSwitch('s%s' % i)
        # 10 Mbps, 5ms delay, 1% loss, 1000 packet queue
        self.addLink( host, switch, bw=10, delay='5ms', loss=1, max_queue_size=1000, use_htb=True)
        if lastSwitch:
            self.addLink(switch, lastSwitch, bw=10, delay='5ms', loss=1, max_queue_size=1000,
use_htb=True)
        lastSwitch = switch

def perfTest():
    "Create network and run simple performance test"
    topo = LinearTopo(k=4)
    net = Mininet(topo=topo,
                  host=CPULimitedHost, link=TCLink)
    net.start()
    print "Dumping host connections"
    dumpNodeConnections(net.hosts)
    print "Testing network connectivity"
    net.pingAll()
    print "Testing bandwidth between h1 and h4"
    h1, h4 = net.get('h1', 'h4')
    net.iperf((h1, h4))
```

每個主機所能分配到的系統 CPU 資源為 50%/k 的資源量

建立主機與交換機的連線，並設置其頻寬為 10Mbps、延遲為 5ms、最大封包數為 1000 個封包其中會有 1 個損失發生，並啟動分層設置(use_htb=True)

導入 CPULimitedHost 與 TCLink 限制於網路拓樸中

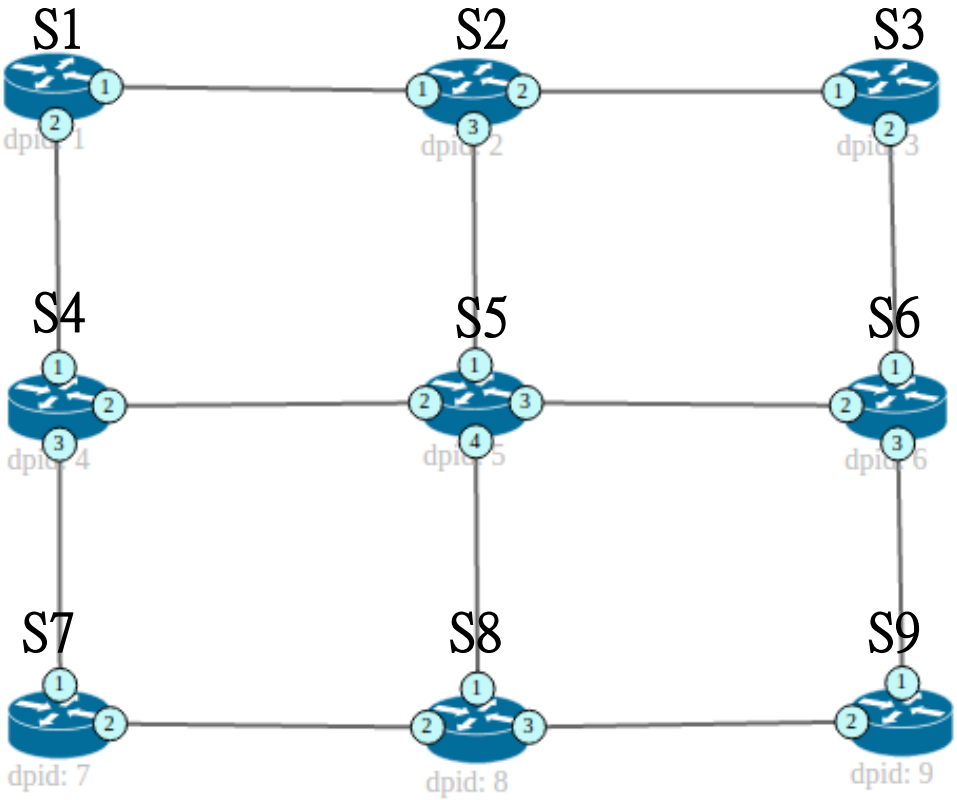
h1 到 h4 的網路連線資訊



Lab 1-1: L2 Routing by MAC Address

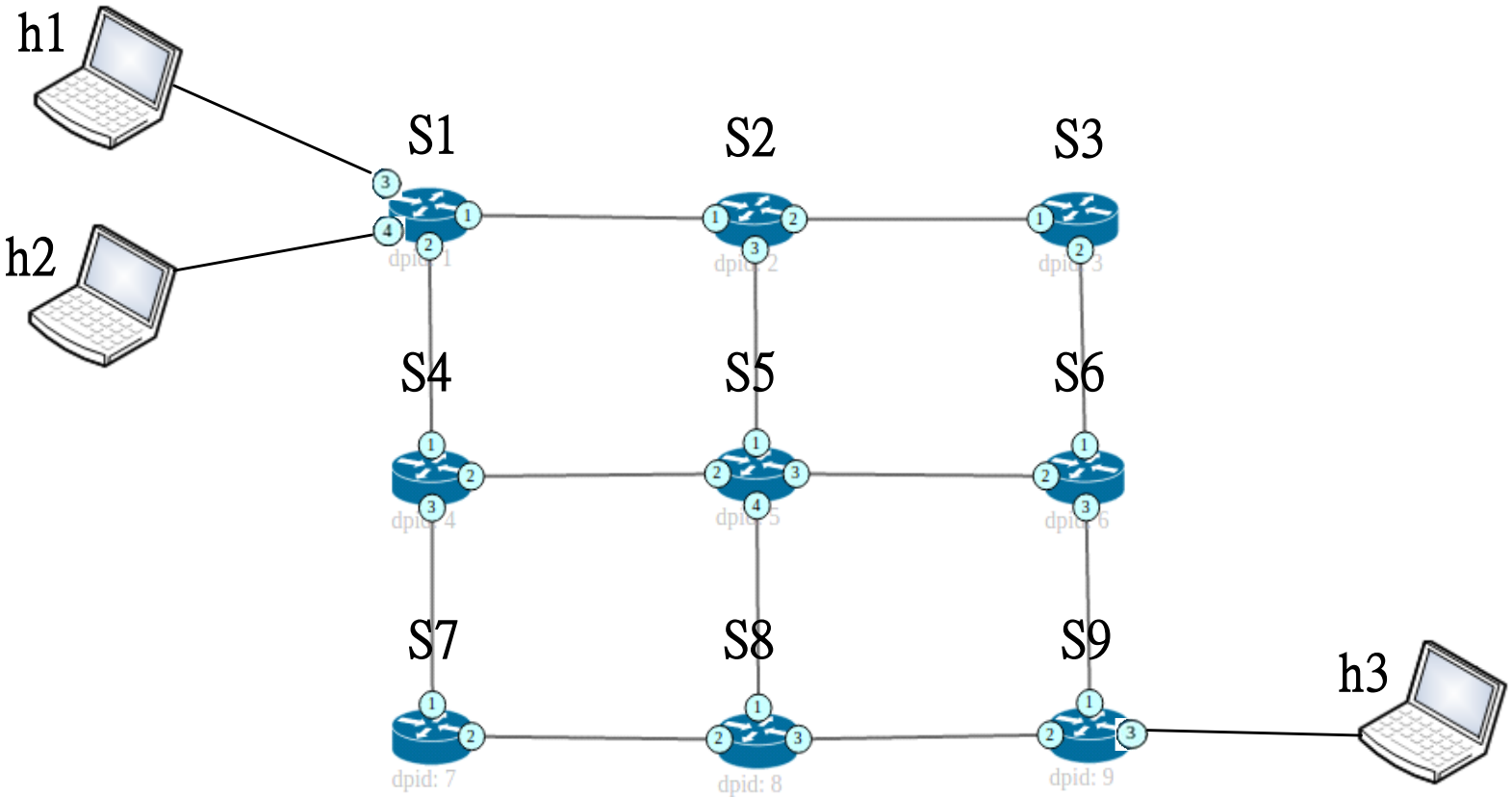
Lab 1-1: L2 routing以MAC為match條件

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



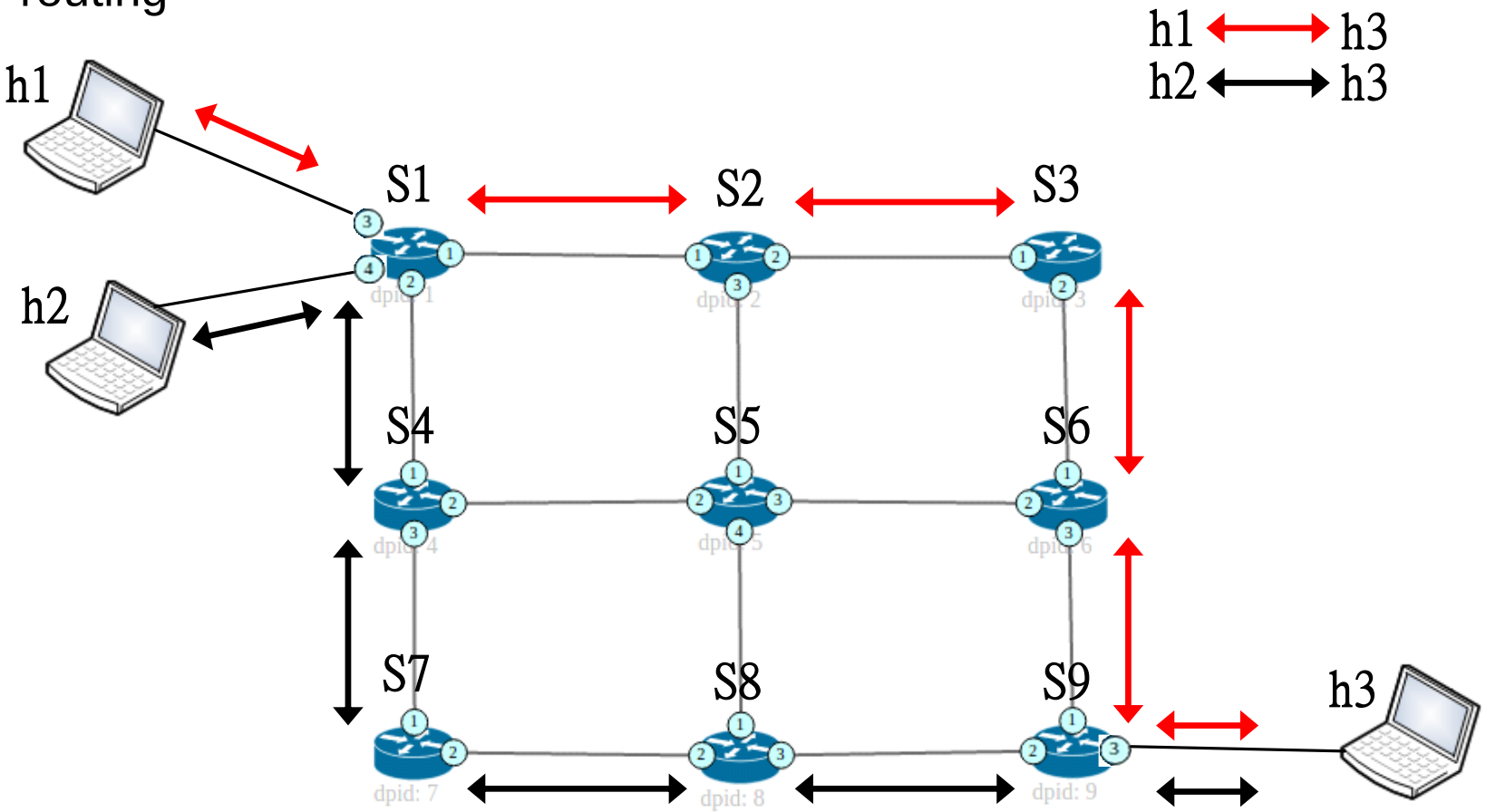
Lab 1-1: L2 routing以MAC為match條件

- 在拓樸網路上加入三台host分別為h1,h2和h3
- 以command line的方式決定switch的路由規則



Lab 1-1: L2 routing以MAC為match條件

- 以MAC位址作為match的條件並以comand line的路由規則達成L2 routing。



Lab 1-1: L2 routing以MAC為match條件

- Step 1:使用自訂拓樸建立grid 3x3的網路環境。
  - 進入到mininet 的custom資料夾底下: cd mininet/custom
  - 輸入以下指令:
    sudo mn --custom grid_3x3.py --topo=mytopo --mac --controller=ryu

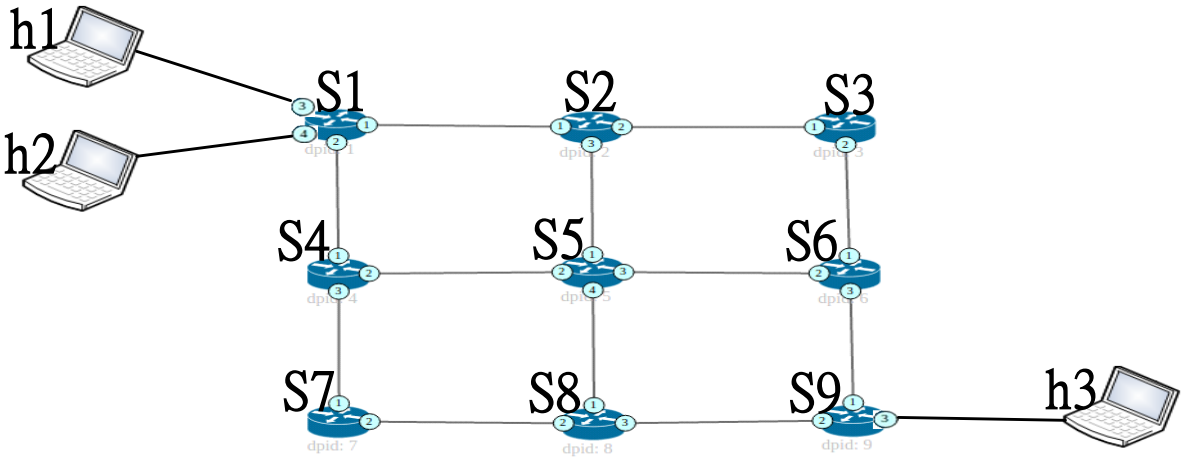
描述拓譜環境

以MAC作為match條件

```
mininet@mininet-VirtualBox:~/mininet/custom$ sudo mn --custom grid_3x3.py --topo
=mytopo --mac --controller=ryu
*** Creating network
*** Adding controller
warning: no Ryu modules specified; running simple_switch only
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9
*** Adding links:
(s1, h1) (s1, h2) (s1, s2) (s1, s4) (s2, s3) (s2, s5) (s3, s6) (s4, s5) (s4, s7)
 (s5, s6) (s5, s8) (s6, s9) (s7, s8) (s8, s9) (s9, h3)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 9 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 ...
*** Starting CLI:
```

Lab 1-1: L2 routing以MAC為match條件

- 檢查拓樸狀態
  - 指令: net




```
mininet> net
h1 h1-eth0:s1-eth3
h2 h2-eth0:s1-eth4
h3 h3-eth0:s9-eth3
s1 lo:  s1-eth1:s2-eth1 s1-eth2:s4-eth1 s1-eth3:h1-eth0 s1-eth4:h2-eth0
s2 lo:  s2-eth1:s1-eth1 s2-eth2:s3-eth1 s2-eth3:s5-eth1
s3 lo:  s3-eth1:s2-eth2 s3-eth2:s6-eth1
s4 lo:  s4-eth1:s1-eth2 s4-eth2:s5-eth2 s4-eth3:s7-eth1
s5 lo:  s5-eth1:s2-eth3 s5-eth2:s4-eth2 s5-eth3:s6-eth2 s5-eth4:s8-eth1
s6 lo:  s6-eth1:s3-eth2 s6-eth2:s5-eth3 s6-eth3:s9-eth1
s7 lo:  s7-eth1:s4-eth3 s7-eth2:s8-eth2
s8 lo:  s8-eth1:s5-eth4 s8-eth2:s7-eth2 s8-eth3:s9-eth2
s9 lo:  s9-eth1:s6-eth3 s9-eth2:s8-eth3 s9-eth3:h3-eth0
c0
```

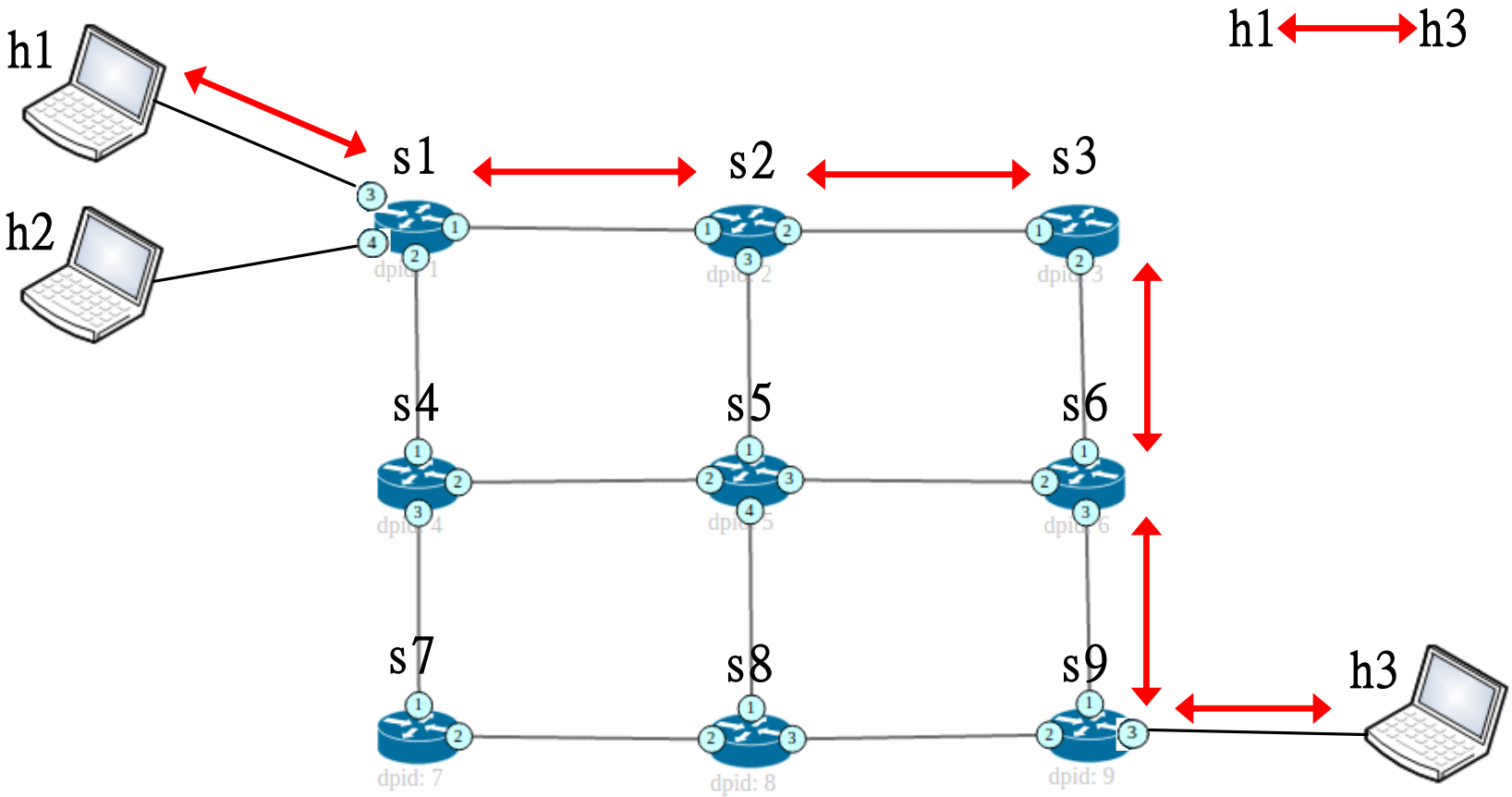
Lab 1-1: L2 routing以MAC為match條件

- 測試
  - 在終端機裡輸入指令: h1 ping h3
    - (預設是不會停止的狀態，按下ctrl+c停止)
  - 使用h1 ping h3或h2 ping h3 時應該是不通的狀況，因為switch裡並沒有任何規則。

```
mininet> h1 ping h3
PING 10.0.0.3 (10.0.0.3) 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
From 10.0.0.1 icmp_seq=5 Destination Host Unreachable
From 10.0.0.1 icmp_seq=6 Destination Host Unreachable
From 10.0.0.1 icmp_seq=7 Destination Host Unreachable
From 10.0.0.1 icmp_seq=8 Destination Host Unreachable
From 10.0.0.1 icmp_seq=9 Destination Host Unreachable
From 10.0.0.1 icmp_seq=10 Destination Host Unreachable
From 10.0.0.1 icmp_seq=11 Destination Host Unreachable
From 10.0.0.1 icmp_seq=12 Destination Host Unreachable
```

Lab 1-1: L2 routing以MAC為match條件

- step 2和step 3執行目的: 進入到switch內設定路由規則
  - h1<->h3的路徑設定(紅色) 。

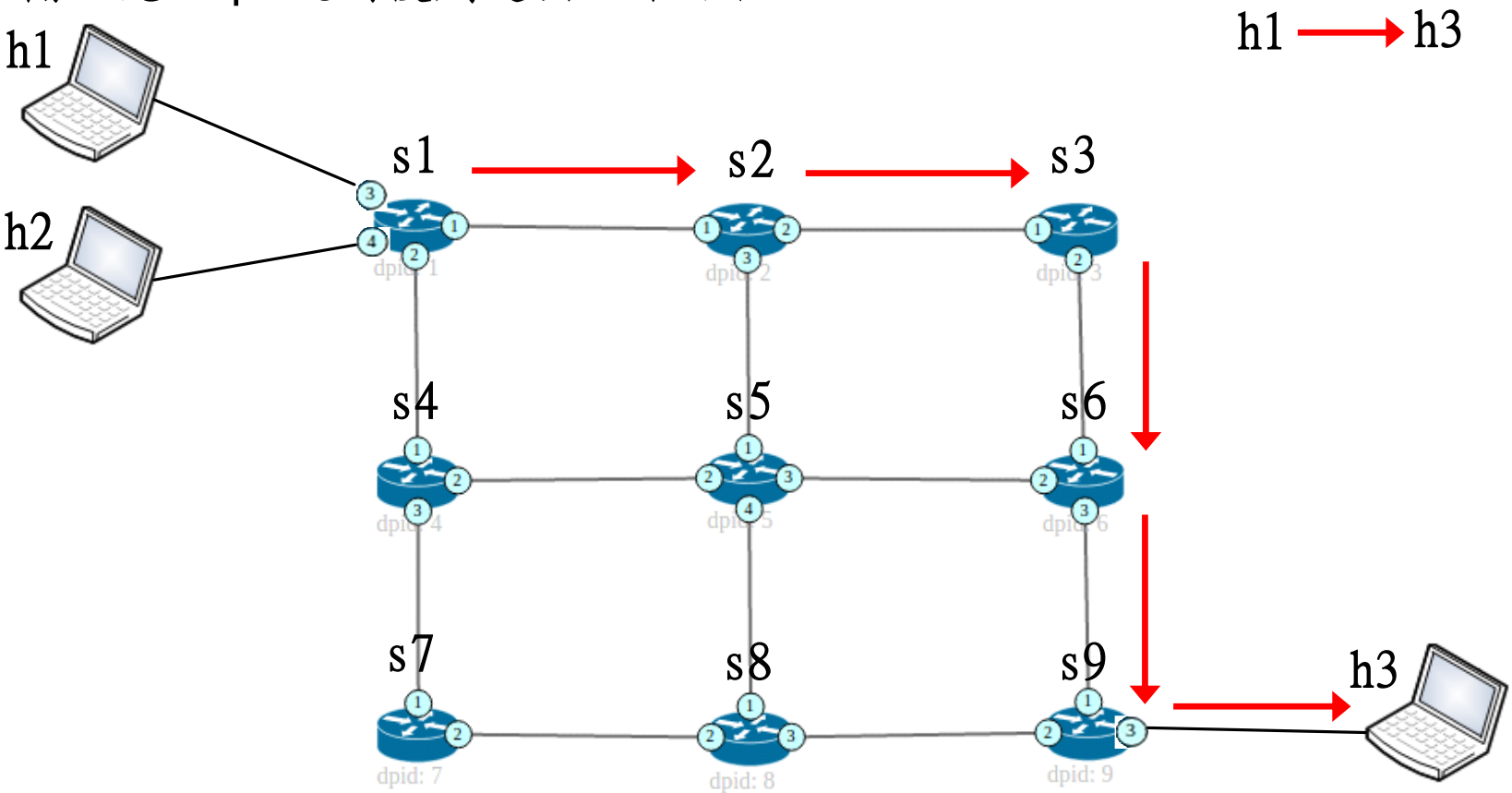


Lab 1-1: L2 routing以MAC為match條件

- step 2: 對s1,s2,s3,s6,s9輸入host1到host3的路徑規則，當來源和目的的mac位置符合時，會往預先決定好的port輸出
  - 在mininet輸入 xterm s1
  - 在s1的終端機輸入下列規則
  - s1的規則:
    ovs-ofctl add-flow s1 “dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03,actions=output:1”
  - s2的規則:
    ovs-ofctl add-flow s2 “dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03,actions=output:2“
  - s3的規則:
    ovs-ofctl add-flow s3 “dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03,actions=output:2“
  - s6的規則:
    ovs-ofctl add-flow s6 “dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03,actions=output:3“
  - s9的規則:
    ovs-ofctl add-flow s9 “dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03,actions=output:3”

Lab 1-1: L2 routing以MAC為match條件

- 輸入完step2規則後創建出以下路徑

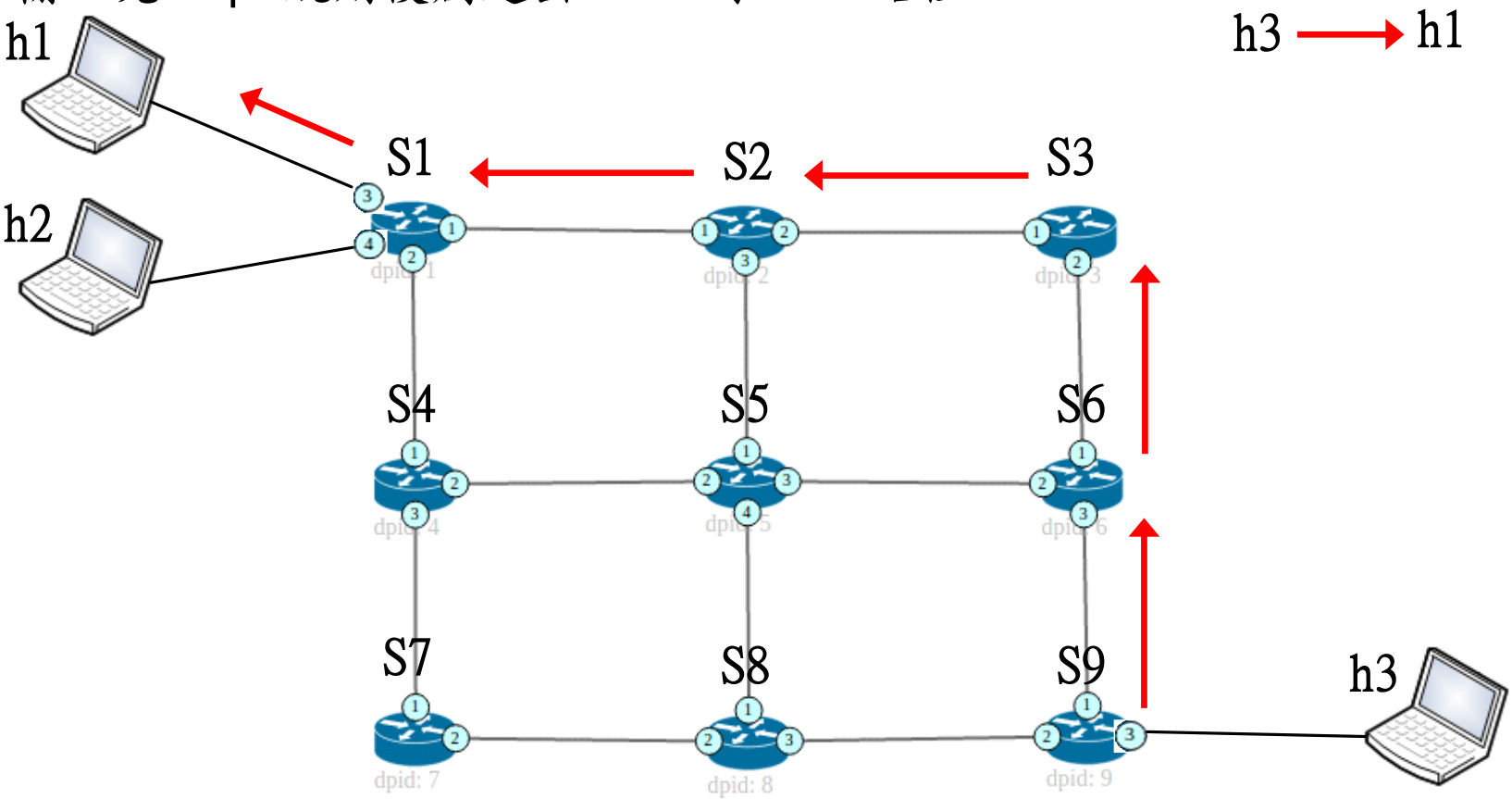


Lab 1-1: L2 routing以MAC為match條件

- step 3:由於測試指令用ping，因此需要host3到host1的規則
  - 在mininet輸入 xterm s1
  - 在s1的終端機輸入下列規則
  - s1的規則:
    ovs-ofctl add-flow s1 “dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:01,actions=output:3”
  - s2的規則:
    ovs-ofctl add-flow s2 “dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:01,actions=output:1”
  - s3的規則:
    ovs-ofctl add-flow s3 “dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:01,actions=output:1”
  - s6的規則:
    ovs-ofctl add-flow s6 “dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:01,actions=output:1”
  - s9的規則:
    ovs-ofctl add-flow s9 “dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:01,actions=output:1”

Lab 1-1: L2 routing以MAC為match條件

- 輸入完step3規則後創建出host3到host1路徑



Lab 1-1: L2 routing以MAC為match條件

- 完成設定後查看各個switch裡的規則
  - 在mininet輸入 xterm s1

```
mininet> xterm s1
```

  - 在s1的終端機輸入 ovs-ofctl dump-flows s1

```
root@mininet-VirtualBox:~/mininet/custom# ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=100.067s, table=0, n_packets=0, n_bytes=0, idle_age=100, d
l_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:01 actions=output:3
 cookie=0x0, duration=476.52s, table=0, n_packets=0, n_bytes=0, idle_age=476, dl
_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03 actions=output:1
```

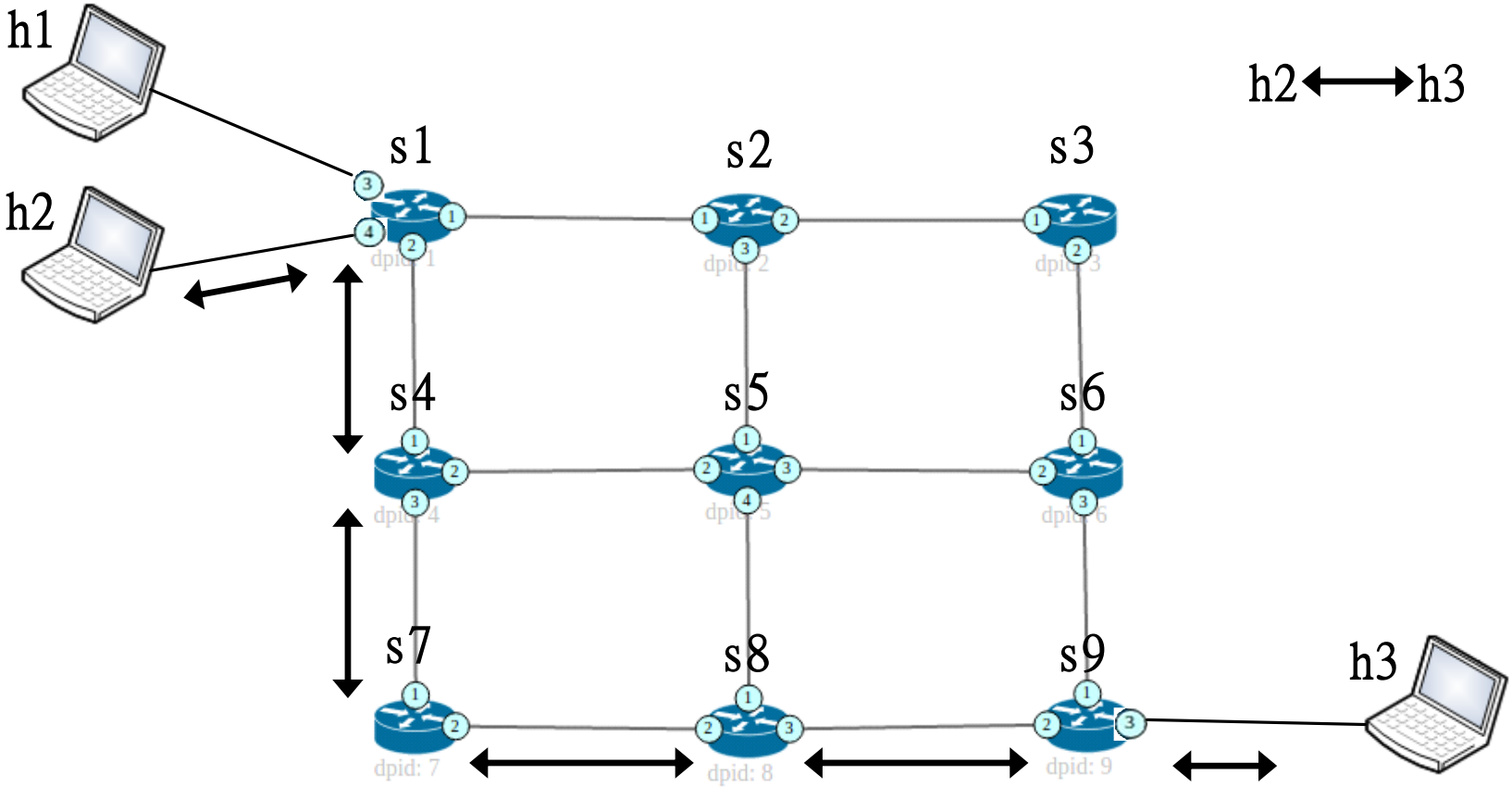
Lab 1-1: L2 routing以MAC為match條件

- 使用h1 ping h3測試

```
@nclab721-ThinkPad-T450: ~
64 bytes from 10.0.0.3: icmp_seq=840 ttl=64 time=0.054 ms
64 bytes from 10.0.0.3: icmp_seq=841 ttl=64 time=0.051 ms
64 bytes from 10.0.0.3: icmp_seq=842 ttl=64 time=0.052 ms
64 bytes from 10.0.0.3: icmp_seq=843 ttl=64 time=0.049 ms
64 bytes from 10.0.0.3: icmp_seq=844 ttl=64 time=0.058 ms
64 bytes from 10.0.0.3: icmp_seq=845 ttl=64 time=0.050 ms
64 bytes from 10.0.0.3: icmp_seq=846 ttl=64 time=0.046 ms
64 bytes from 10.0.0.3: icmp_seq=847 ttl=64 time=0.049 ms
64 bytes from 10.0.0.3: icmp_seq=848 ttl=64 time=0.046 ms
64 bytes from 10.0.0.3: icmp_seq=849 ttl=64 time=0.055 ms
64 bytes from 10.0.0.3: icmp_seq=850 ttl=64 time=0.046 ms
64 bytes from 10.0.0.3: icmp_seq=851 ttl=64 time=0.058 ms
64 bytes from 10.0.0.3: icmp_seq=852 ttl=64 time=0.047 ms
64 bytes from 10.0.0.3: icmp_seq=853 ttl=64 time=0.048 ms
64 bytes from 10.0.0.3: icmp_seq=854 ttl=64 time=0.046 ms
64 bytes from 10.0.0.3: icmp_seq=855 ttl=64 time=0.048 ms
64 bytes from 10.0.0.3: icmp_seq=856 ttl=64 time=0.055 ms
64 bytes from 10.0.0.3: icmp_seq=857 ttl=64 time=0.054 ms
64 bytes from 10.0.0.3: icmp_seq=858 ttl=64 time=0.060 ms
```

Lab 1-1: L2 routing以MAC為match條件

- step 4和step 5: 完成h2<->h3的路徑設定(黑色)

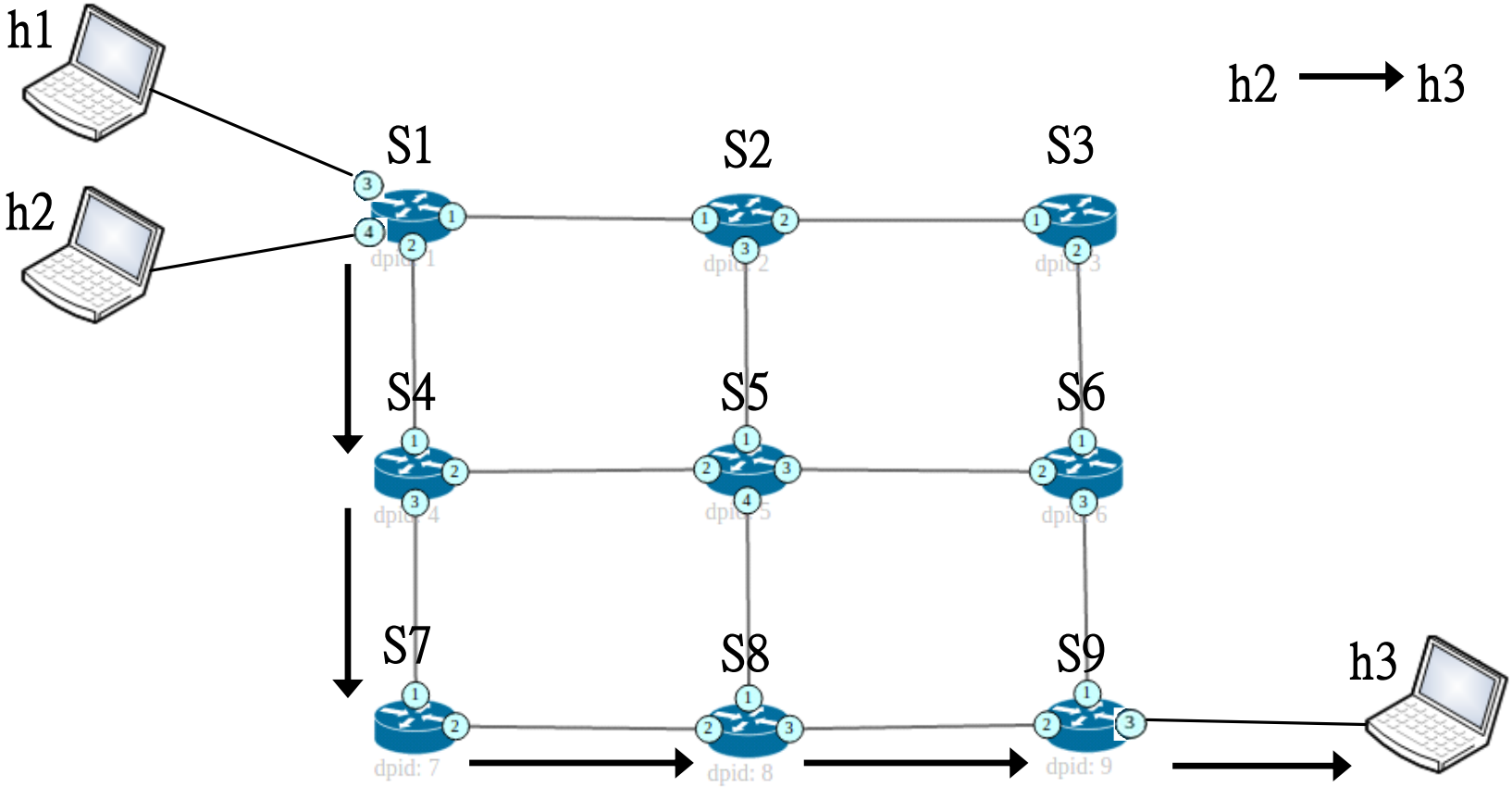


Lab 1-1: L2 routing以MAC為match條件

- step 4: 需要對s1,s4,s7,s8,s9輸入host2到host3的規則
  - 在mininet輸入 xterm s1
  - 在s1的終端機輸入下列規則
  - s1的規則:
    ovs-ofctl add-flow s1 “dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03,actions=output:2”
  - s4的規則:
    ovs-ofctl add-flow s4 “dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03,actions=output:3”
  - s7的規則:
    ovs-ofctl add-flow s7 “dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03,actions=output:2”
  - s8的規則:
    ovs-ofctl add-flow s8 “dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03,actions=output:3”
  - s9的規則:
    ovs-ofctl add-flow s9 “dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03,actions=output:3”

Lab 1-1: L2 routing以MAC為match條件

- 輸入完規則後創建出以下路徑

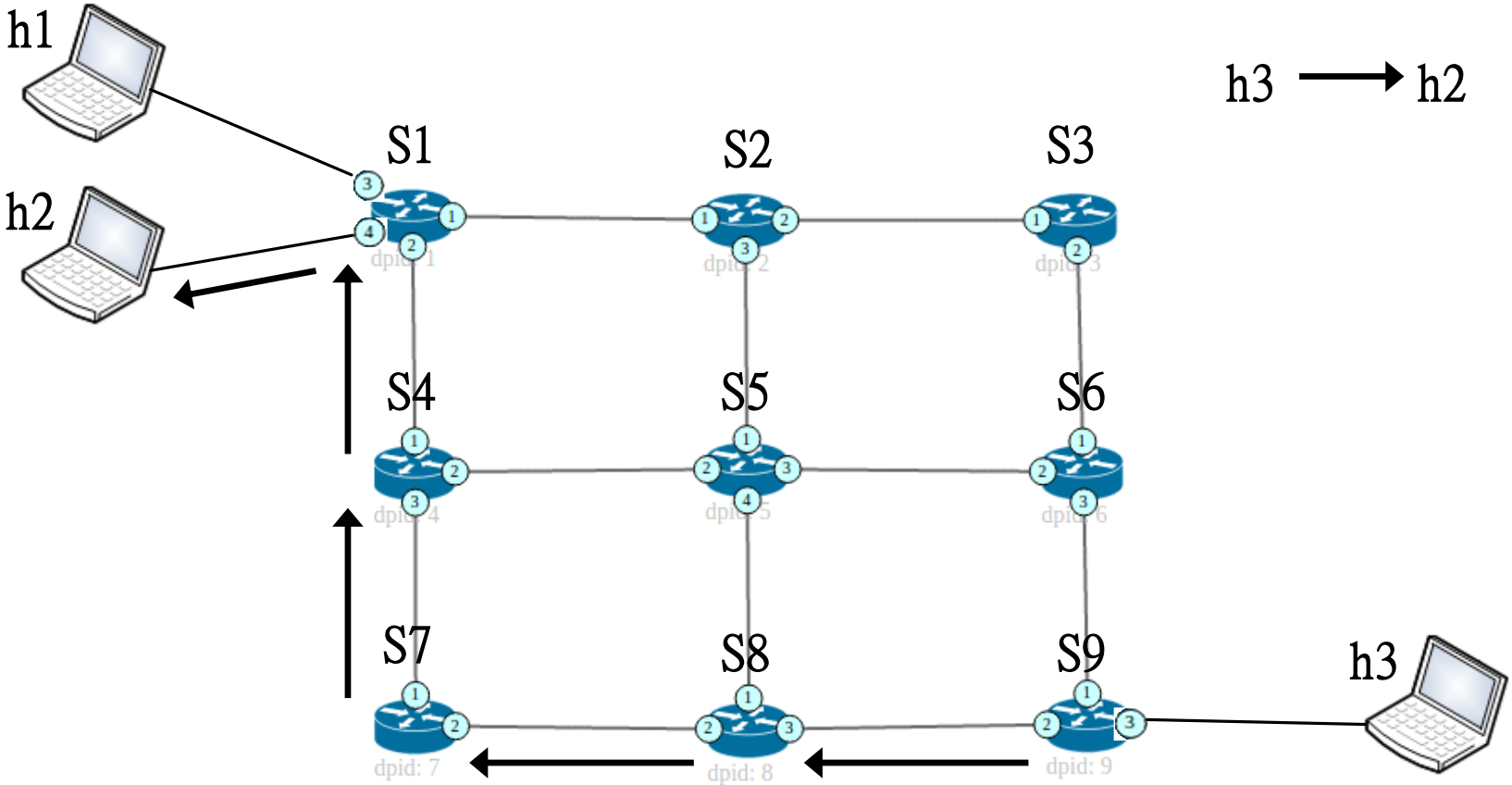


Lab 1-1: L2 routing以MAC為match條件

- step 5:由於測試指令用ping，因此需要host3到host2的規則
  - 在mininet輸入 xterm s1
  - 在s1的終端機輸入下列規則
  - s1的規則:
    ovs-ofctl add-flow s1 “dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02,actions=output:4”
  - s4的規則:
    ovs-ofctl add-flow s4 “dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02,actions=output:1”
  - s7的規則:
    ovs-ofctl add-flow s7 “dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02,actions=output:1”
  - s8的規則:
    ovs-ofctl add-flow s8 “dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02,actions=output:2”
  - s9的規則:
    ovs-ofctl add-flow s9 “dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02,actions=output:2”

Lab 1-1: L2 routing以MAC為match條件

- 輸入完規則後創建出反向路徑



Lab 1-1: L2 routing以MAC為match條件

- 完成設定後查看各個switch裡的規則
  - 在mininet輸入 xterm s1

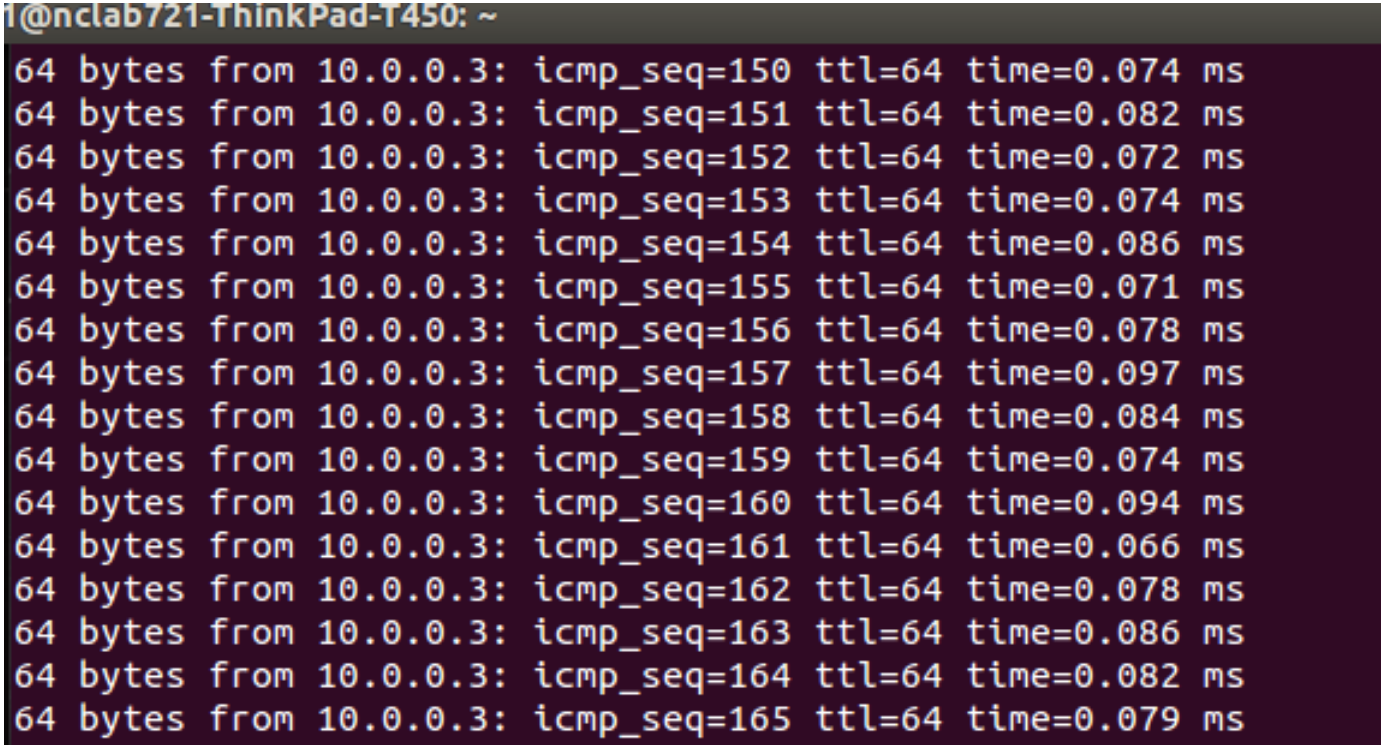
```
mininet> xterm s1
```

  - 在s1的終端機輸入 ovs-ofctl dump-flows s1

```
root@mininet-VirtualBox:~/mininet/custom# ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=80.682s, table=0, n_packets=0, n_bytes=0, idle_age=80, dl_
src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02 actions=output:4
 cookie=0x0, duration=53.245s, table=0, n_packets=0, n_bytes=0, idle_age=53, dl_
src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03 actions=output:2
```

Lab 1-1: L2 routing以MAC為match條件

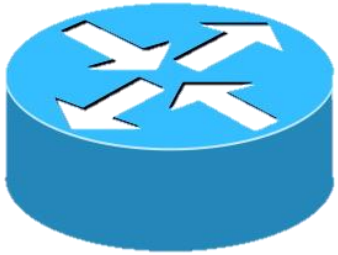
- 使用h2 ping h3或h1 ping h3測試



Lab 1-1: L2 routing以MAC為match條件

- 測試輸入pingall
  - 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。

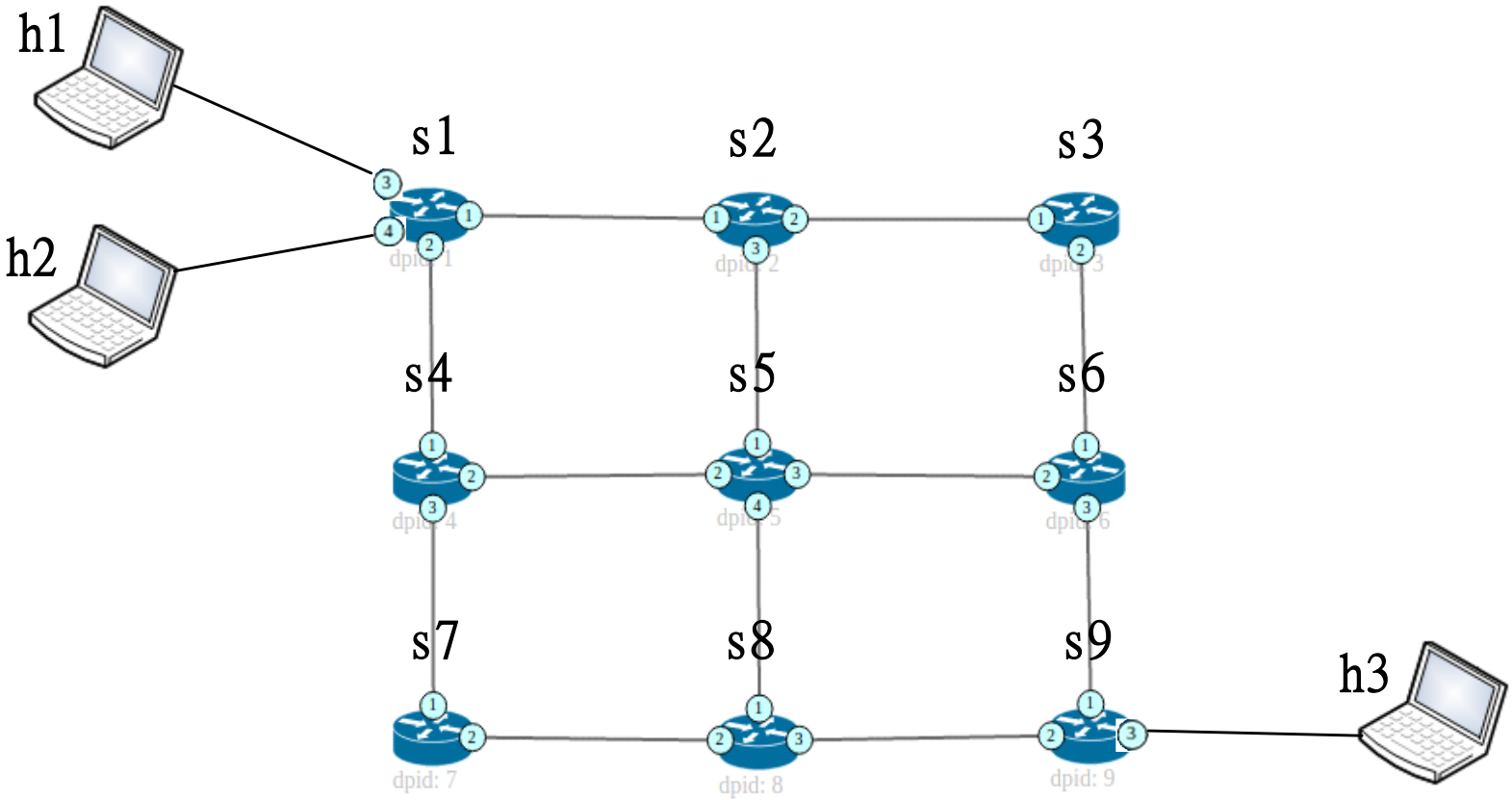
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3
h2 -> X h3
h3 -> h1 h2
*** Results: 33% dropped (4/6 received)
mininet>
```



Lab 1-2: Firewall

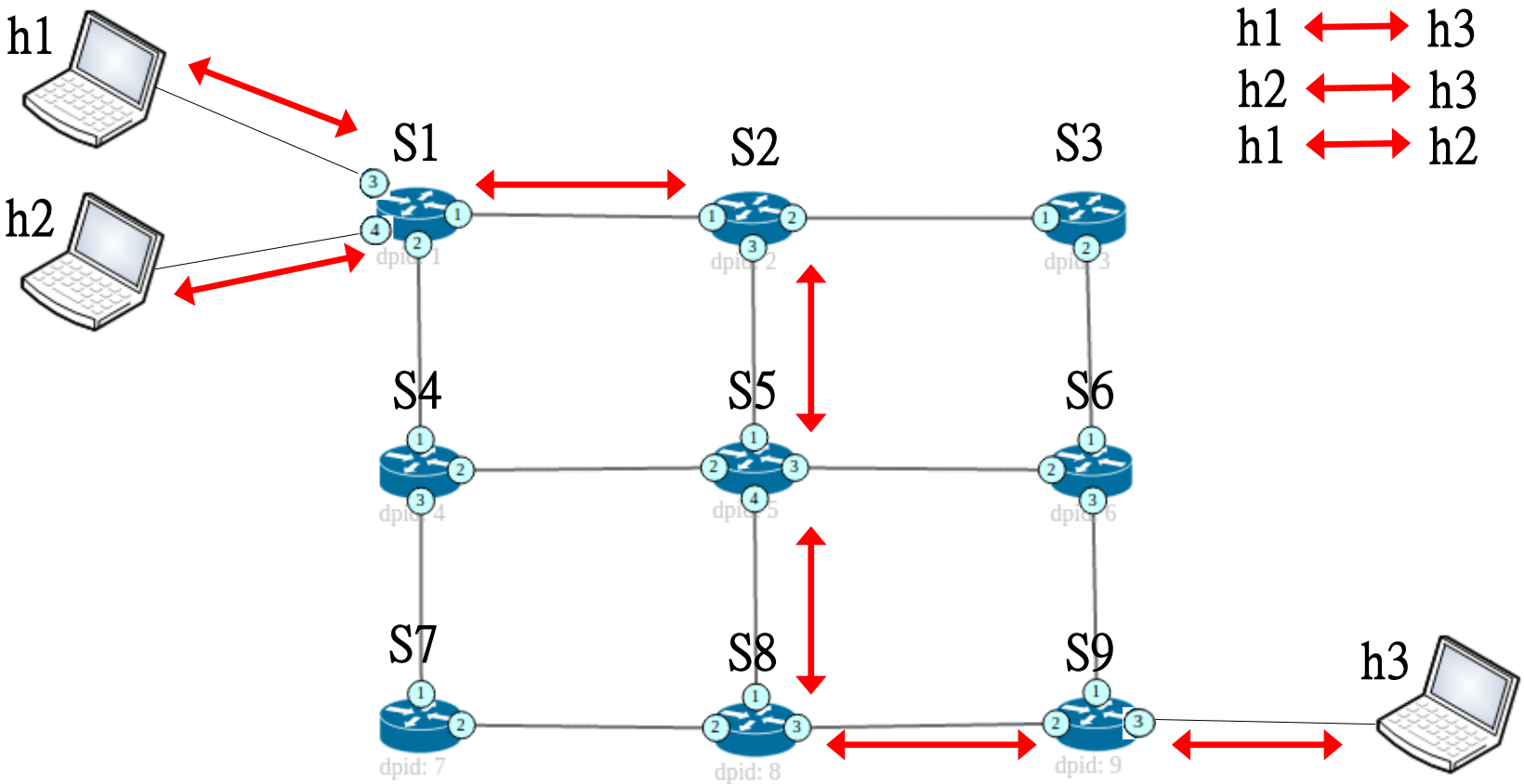
Lab 1-2: Firewall

- 目的: 讓h2的封包無法通過s1



Lab 1-2: Firewall

- step1: 首先輸入規則讓h1,h2都能夠ping到h3，路徑如下



Lab 1-2: Firewall

- step1:對s1,s2,s3,s6,s9輸入規則。
  - s1的規則:

    ovs-ofctl add-flow s1 “priority=32760,dl_dst=00:00:00:00:00:03,actions=output:1”
    ovs-ofctl add-flow s1 “priority=32760,dl_dst=00:00:00:00:00:01,actions=output:3”
    ovs-ofctl add-flow s1 “priority=32760,dl_dst=00:00:00:00:00:02,actions=output:4”
  - s2的規則:

    ovs-ofctl add-flow s2 “priority=32760,dl_dst=00:00:00:00:00:03,actions=output:2”
    ovs-ofctl add-flow s2 “priority=32760,dl_dst=00:00:00:00:00:01,actions=output:1”
    ovs-ofctl add-flow s2 “priority=32760,dl_dst=00:00:00:00:00:02,actions=output:1”
  - s3的規則:

    ovs-ofctl add-flow s3 “priority=32760,dl_dst=00:00:00:00:00:03,actions=output:2”
    ovs-ofctl add-flow s3 “priority=32760,dl_dst=00:00:00:00:00:01,actions=output:1”
    ovs-ofctl add-flow s3 “priority=32760,dl_dst=00:00:00:00:00:02,actions=output:1”

Lab 1-2: Firewall

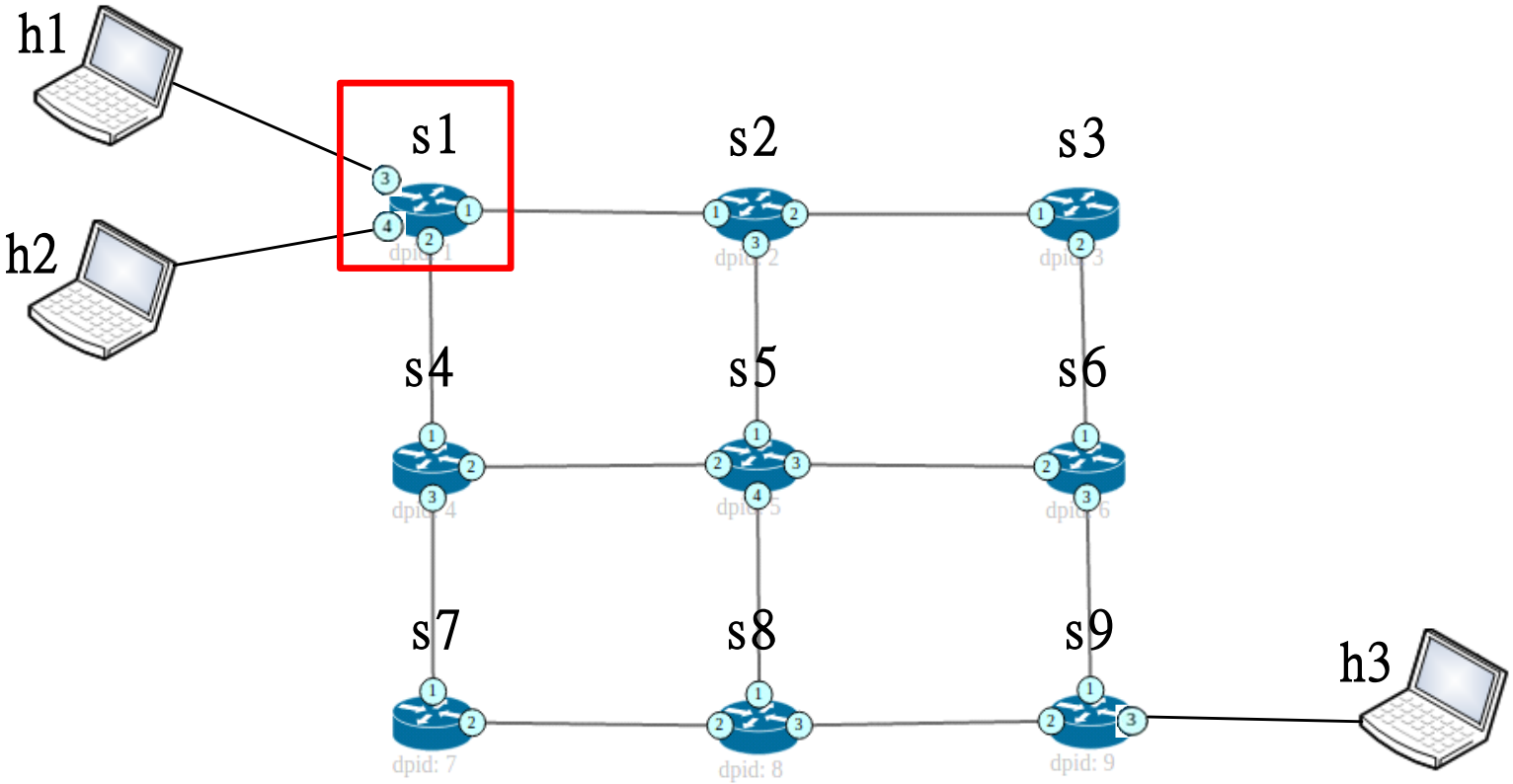
- s6的規則:
  ovs-ofctl add-flow s6 “priority=32760,dl_dst=00:00:00:00:00:03,actions=output:3”
  ovs-ofctl add-flow s6 “priority=32760,dl_dst=00:00:00:00:00:01,actions=output:1”
  ovs-ofctl add-flow s6 “priority=32760,dl_dst=00:00:00:00:00:02,actions=output:1”
- s9的規則:
  ovs-ofctl add-flow s9 “priority=32760,dl_dst=00:00:00:00:00:03,actions=output:3”
  ovs-ofctl add-flow s9 “priority=32760,dl_dst=00:00:00:00:00:01,actions=output:1”
  ovs-ofctl add-flow s9 “priority=32760,dl_dst=00:00:00:00:00:02,actions=output:1”

❖ step2:輸入pingall測試互相都ping的通

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

Lab 1-2: Firewall

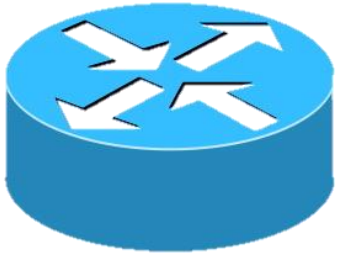
- step3: 開始加入防火牆，輸入路由規則讓s1丟棄h2的封包。
  - 在s1加入以下這條規則
    ovs-ofctl add-flow s1 “priority=32767,dl_src=00:00:00:00:00:02,actions=drop”



Lab 1-2: Firewall

- step4: 再輸入一次pingall
  - 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)
```



Lab 1-3: Create Custom Topology

Create a 4x4 mesh Topology

- 執行指令:sudo mn --custom custom/ 4x4MeshTopo.py --topo=mytopo

附錄

- 附錄
  - Mininet 常用指令
  - Open vSwitch常用指令
  - Python語言教學影片

Mininet 常用指令

- sudo mn -h
  列出所有指令及其教學
- nodes
  - 顯示nodes列表(所有controller, host和switch)
- net
  - 顯示所有link列表
- dump
  - 顯示所有node詳細資訊
- exit
  - 離開mininet CLI
- sudo mn –c
  - 清除網路拓樸

Mininet 常用指令

- dpctl show
  - 顯示所有 virtual switch狀態
- dpctl --version
  - 顯示switch支援的openflow版本
- dpctl dump-flows
  - 顯示switch所有的flow entries
- dpctl dump-ports
  - 顯示switch所有port的流量統計
- [host 1] ping [host 2]
  - 發送ping封包

Open vSwitch常用指令

- ovs-ofctl dump-flows [switch]
  - 列出指定switch的所有規則
- ovs-ofctl add-flow switch “match,action”
  - 加入規則到指定的switch
- ovs-ofctl del-flow switch “match”
  - 刪除指定switch符合match條件的規則
- xterm [switch]
  - 指定switch的終端機視窗

Python語言教學影片

- Tutorial video
  - http://youtu.be/N4mEzFDjqtA

- Official tutorial
  - https://docs.python.org/2/tutorial/
  - https://docs.python.org/3/tutorial/

Thank you