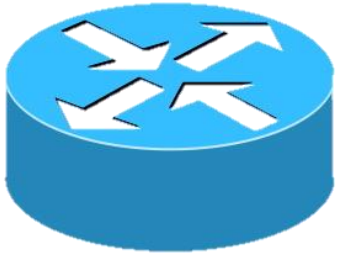


Department of Electronics Engineering, NTU

Traffic Monitor

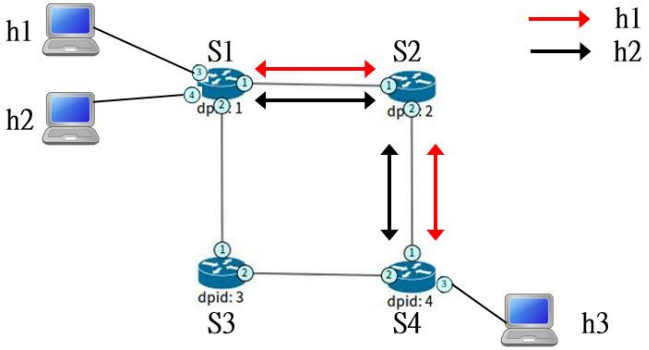
Speaker: Yi-Ta Chen

Advisor: Tsung-Nan Lin

Date: 20180521

Lab 4-1: Traffic Monitor

- Step 1: 修改SDNApplication.py
  - 為了確認拓樸上的交換器都可以被監控到，在初始化時新增一字典變數(紅框)，
  - 並在握手協議完成後加入以下程式(藍框)，將拓樸上的交換器紀錄下來




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

    def __init__(self, *args, **kwargs):
        super(SDNApplication, self).__init__(*args, **kwargs)
        self.datapaths = {}
        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
        self.datapaths[dpid] = datapath
```

Lab 4-1: Traffic Monitor

- Step 2: 監控是持續進行的，因此必須建立一個執行緒，定期取得交換器上的統計資料。
  - ryu.lib.hub為執行緒的類別。
  - 下圖使用hub.spawn()建立執行緒，並搭配_monitor自訂函式(下一頁)。

```
from ryu.lib.packet import ether_types
from ryu.lib import hub

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

    def __init__(self, *args, **kwargs):
        super(SDNApplication, self).__init__(*args, **kwargs)
        self.datapaths = {}
        self.monitor_thread = hub.spawn(self._monitor)
```

Lab 4-1: Traffic Monitor

- Step 3: 讓執行緒固定時間持續執行。
  - _monitor()確保執行緒可以在每隔3秒向交換器發送request取得統計資訊。

```
def _monitor(self):
    while True:
        for dp in self.datapaths.values():
            self._request_stats(dp)
        hub.sleep(3)
```

- Step 4: _request_stats()
  - 使用OFPPortStatsRequest函式取得交換器的相關資訊。
  - 下圖使用OFPP_ANY取得所有的連接埠統計資料。

```
def _request_stats(self, datapath):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    req = parser.OFPPortStatsRequest(datapath, 0, ofproto.OFPP_ANY)
    datapath.send_msg(req)
```

Lab 4-1: Traffic Monitor

- Step 5: 接收交換器所回覆的訊息
  - 在初始化時新增一字典變數，如下圖
  - 並搭配_port_stats_reply_handler()函式(下一頁)

```
def __init__(self, *args, **kwargs):
    super(SDNapplication, self).__init__(*args, **kwargs)
    self.mac_to_port = {}
    self.datapaths = {}
    self.monitor_thread = hub.spawn(self._monitor)
    self.bandwidth = {}
```

Lab 4-1: Traffic Monitor

- Step 6: _port_stats_reply_handler()
  - body會列出在OFPPortStats中的資料列表，
    包括連接埠號、接收的封包數量、位元數量等等資訊。

```
@set_ev_cls(ofp_event.EventOFPPortStatsReply, MAIN_DISPATCHER)
def _port_stats_reply_handler(self, ev):
    body = ev.msg.body
    parser = ev.msg.datapath.ofproto_parser
    self.logger.info('datapath         port     '
                     'rx-pkts  rx-bytes '
                     'tx-pkts  tx-bytes bandwidth')
    self.logger.info('---------------- -------- '
                     '-------- -------- '
                     '-------- -------- --------')
    for stat in sorted(body):
        if stat.port_no < 5:
            index = str(ev.msg.datapath.id) + '-' + str(stat.port_no)
            if index not in self.bandwidth:
                self.bandwidth[index] = 0
            transfer_bytes = stat.rx_bytes + stat.tx_bytes
            speed = (transfer_bytes - self.bandwidth[index]) / 3
            self.logger.info('%016x %8x %8d %8d %8d %8d %8d',
                             ev.msg.datapath.id, stat.port_no,
                             stat.rx_packets, stat.rx_bytes,
                             stat.tx_packets, stat.tx_bytes, speed)

            self.bandwidth[index] = transfer_bytes
```

Lab 4-1: Traffic Monitor

- Step 7: 執行Ryu控制器與Mininet，進行流量監控。
  - 由於沒有任何流量，使用頻寬為零。

datapath	port	rx-pkts	rx-bytes	tx-pkts	tx-bytes	bandwidth
0000000000000003	1	21	3141	21	3141	0
0000000000000003	2	21	3141	21	3141	0
datapath	port	rx-pkts	rx-bytes	tx-pkts	tx-bytes	bandwidth
0000000000000001	1	21	3141	21	3141	0
0000000000000001	2	21	3141	21	3141	0
0000000000000001	3	9	738	21	3141	0
0000000000000001	4	9	738	21	3141	0
datapath	port	rx-pkts	rx-bytes	tx-pkts	tx-bytes	bandwidth
0000000000000004	1	21	3141	21	3141	0
0000000000000004	2	21	3141	21	3141	0
0000000000000004	3	9	738	21	3141	0
datapath	port	rx-pkts	rx-bytes	tx-pkts	tx-bytes	bandwidth
0000000000000002	1	21	3141	21	3141	0
0000000000000002	2	21	3141	21	3141	0

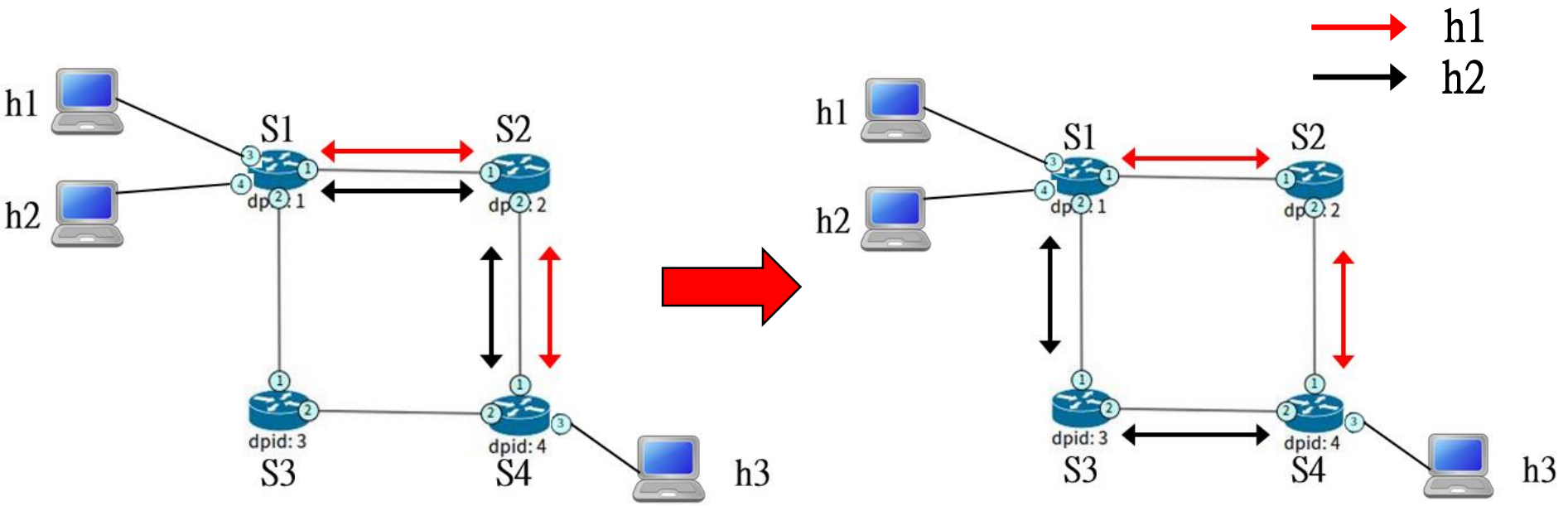
Lab 4-1: Traffic Monitor

- Step 8: sudo mn --topo=tree,2 --mac –switch ovsk,protocols=OpenFlow13 --controller=remote
- 使用Ping程式測試觀察是否有監控到流量。

```
datapath           port     rx-pkts  rx-bytes tx-pkts  tx-bytes bandwidth
------------------ -------- -------- -------- -------- -------- --------
0000000000000002          1    13480   571821    40277  1697295    40488
0000000000000002          2    40277  1697295    13479   571779    40474
datapath           port     rx-pkts  rx-bytes tx-pkts  tx-bytes bandwidth
------------------ -------- -------- -------- -------- -------- --------
0000000000000003          1    14165   597511    14167   597595    20790
0000000000000003          2    14167   597595    14165   597511    20790
datapath           port     rx-pkts  rx-bytes tx-pkts  tx-bytes bandwidth
------------------ -------- -------- -------- -------- -------- --------
0000000000000001          1    42508  1791165    14225   603279    41790
0000000000000001          2    14167   597595    14165   597511    20790
0000000000000001          3       70     6632    56651  2385171    41790
0000000000000001          4        9      738    28308  1191517    20790
datapath           port     rx-pkts  rx-bytes tx-pkts  tx-bytes bandwidth
------------------ -------- -------- -------- -------- -------- --------
0000000000000004          1    14225   603279    42509  1791207    41804
0000000000000004          2    14165   597511    14167   597595    20790
0000000000000004          3    28354  1194512    28367  1197243    41804
```

Lab 4-2: Dynamic Routing

- 當網路中某連接埠正處於高流量的狀態，必須根據目前網路狀態動態調整路由，避免網路壅塞導致傳輸延遲甚至掉封包情形。
- 情境: 當S1之連接埠1的使用頻寬超過10Mbps時，
  則將h2->h3的路徑調整成S1->S3->S4。



Lab 4-5: Dynamic Routing

- Step 1: 修改SDN_Application.py
  - 設定頻寬門檻值10 Mbps(10485760 bytes)

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

    def __init__(self, *args, **kwargs):
        super(SDNApplication, self).__init__(*args, **kwargs)
        self.threshold = 10485760
        self.datapaths = {}
```

Lab 4-5: Dynamic Routing

- Step 2:判斷S1之連接埠1使用頻寬是否超過10Mbps
  - 在port_stats_reply_handler()函式底下加入if判斷式。

```
for stat in sorted(body):
    if stat.port_no < 5:
        index = str(ev.msg.datapath.id) + '-' + str(stat.port_no)
        if index not in self.bandwidth:
            self.bandwidth[index] = 0
        transfer_bytes = stat.rx_bytes + stat.tx_bytes
        speed = (transfer_bytes - self.bandwidth[index]) / 3
        self.logger.info('%016x %8x %8d %8d %8d %8d %8d',
                         ev.msg.datapath.id, stat.port_no,
                         stat.rx_packets, stat.rx_bytes,
                         stat.tx_packets, stat.tx_bytes, speed)

        self.bandwidth[index] = transfer_bytes

        if  speed > self.threshold and index == '1-1':
```

Lab 4-5: Dynamic Routing

- Step 3: 在if判斷式裡分別對S1, S3, S4加入規則
  - 調整h2->h3路徑為S1->S3->S4
  - 當網路壅塞時，分流傳輸流量。

```
if  speed > self.threshold and index == '1-1':
    self.add_flow(self.datapaths[1], 5,
        parser.OFPMatch(eth_src = '00:00:00:00:00:02')
        , [parser.OFPActionOutput(2)])
    self.add_flow(self.datapaths[3], 5,
        parser.OFPMatch(eth_src = '00:00:00:00:00:02')
        , [parser.OFPActionOutput(2)])
    self.add_flow(self.datapaths[3], 5,
        parser.OFPMatch(eth_dst = '00:00:00:00:00:02')
        , [parser.OFPActionOutput(1)])
    self.add_flow(self.datapaths[4], 5,
        parser.OFPMatch(eth_dst = '00:00:00:00:00:02')
        , [parser.OFPActionOutput(2)])
```

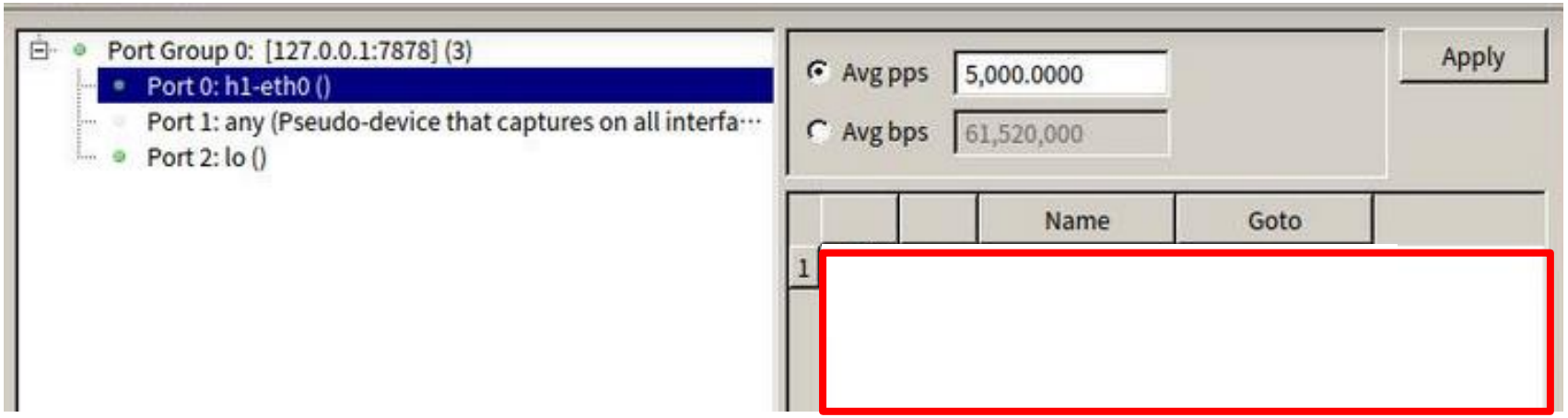
Lab 4-5: Dynamic Routing

- Step 4: 執行Ryu控制器與Mininet，進行流量監控。
  - 沒有任何流量，使用頻寬為零。
  - 注意: 控制器與Mininet不要終止，繼續以下步驟。

datapath	port	rx-pkts	rx-bytes	tx-pkts	tx-bytes	bandwidth
0000000000000003	1	21	3141	21	3141	0
0000000000000003	2	21	3141	21	3141	0
datapath	port	rx-pkts	rx-bytes	tx-pkts	tx-bytes	bandwidth
0000000000000001	1	21	3141	21	3141	0
0000000000000001	2	21	3141	21	3141	0
0000000000000001	3	9	738	21	3141	0
0000000000000001	4	9	738	21	3141	0
datapath	port	rx-pkts	rx-bytes	tx-pkts	tx-bytes	bandwidth
0000000000000004	1	21	3141	21	3141	0
0000000000000004	2	21	3141	21	3141	0
0000000000000004	3	9	738	21	3141	0
datapath	port	rx-pkts	rx-bytes	tx-pkts	tx-bytes	bandwidth
0000000000000002	1	21	3141	21	3141	0
0000000000000002	2	21	3141	21	3141	0

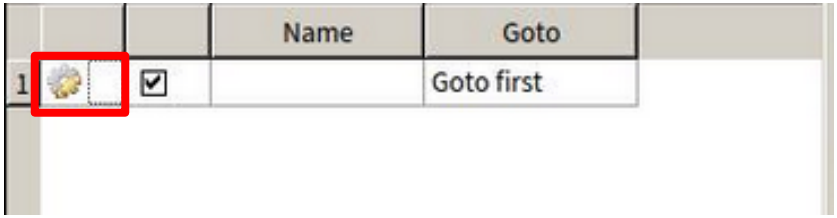
Lab 4-5: Dynamic Routing

- Step 5: 產生traffic
  - 在mininet CLI底下輸入xterm h1，
    並在h1 CLI底下輸入sudo ostinato &。
  - 選擇h1-eth0，並在空白處(紅框)按下滑鼠右鍵點選New Stream。

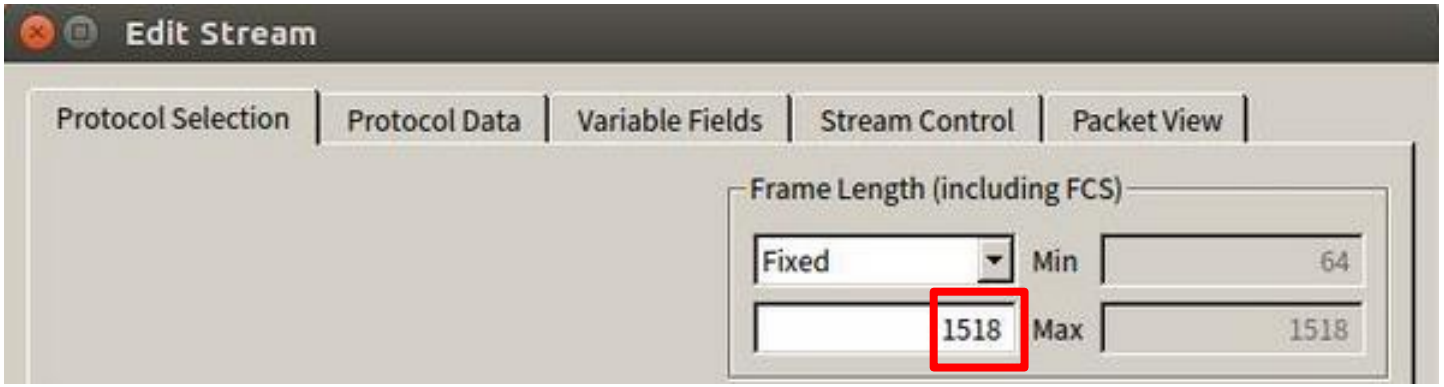


Lab 4-5: Dynamic Routing

- Step 6: 將Next模式設定為Goto first模式後，
  按下設定鍵(紅框)。

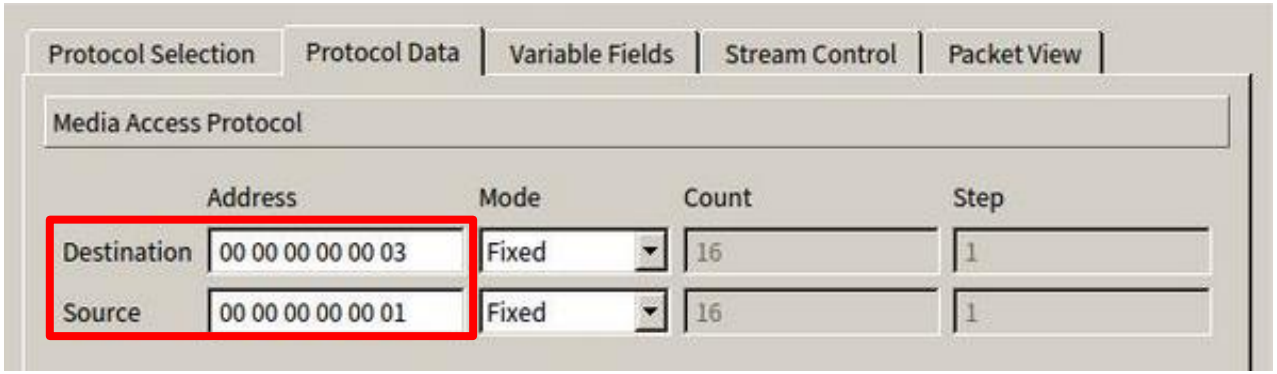


- Step 7: 將封包大小設定為1518 bytes。

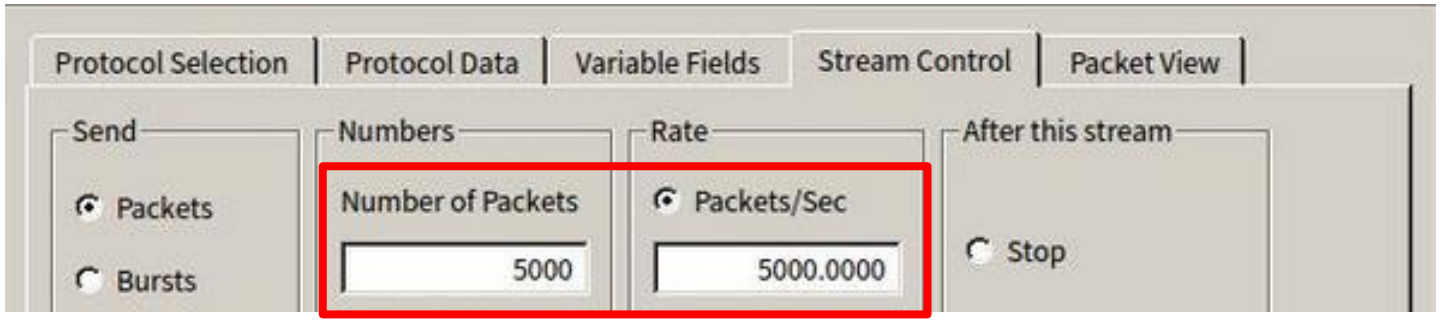


Lab 4-5: Dynamic Routing

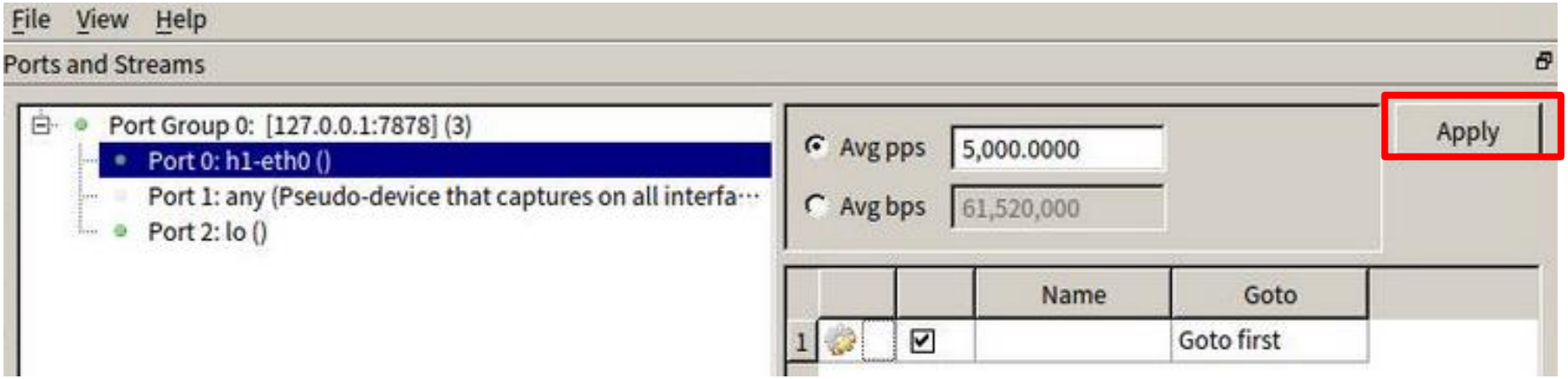
- Step 8: 將目的端MAC位址設為h3，
  來源端MAC位址設為h1。



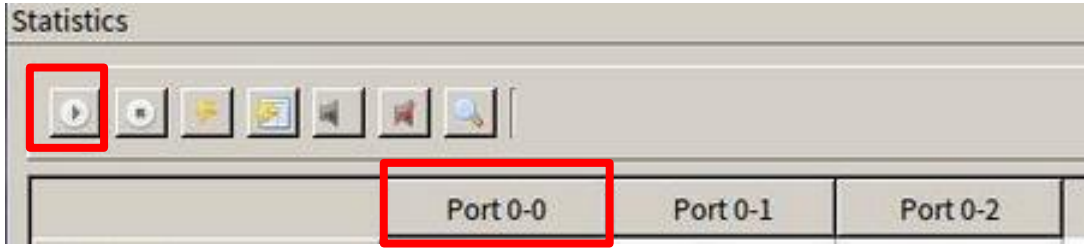
- Step 9: 設定每秒送出5000個封包，
  並按下右下角OK完成設定。



Lab 4-5: Dynamic Routing



- Step 11: 按下Port 0-0後，再按下發送鍵。



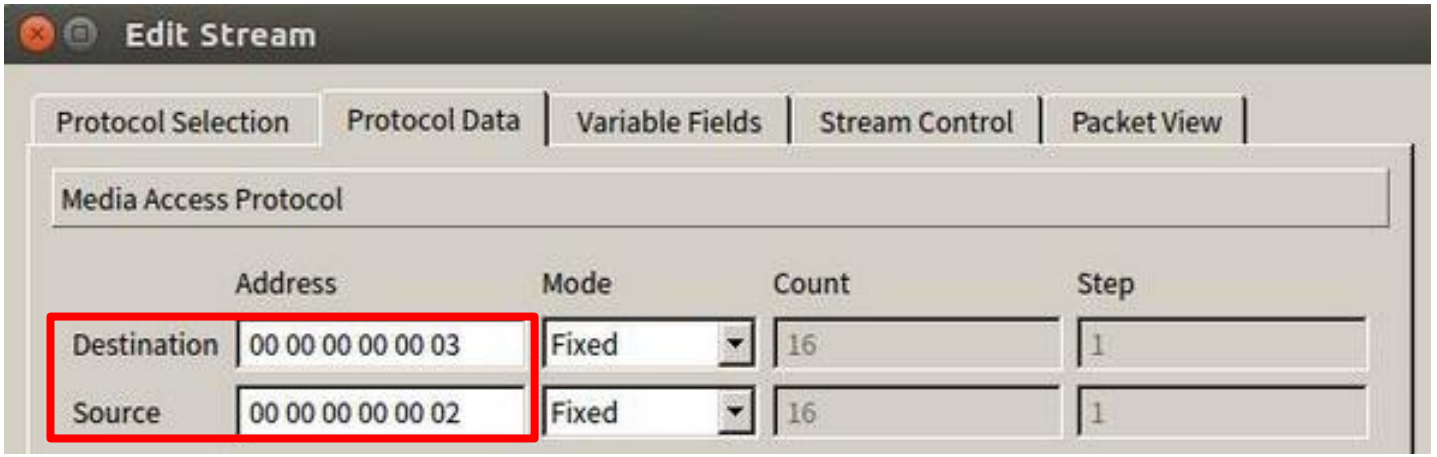
Lab 4-5: Dynamic Routing

- Step 12: 觀察控制器視窗
  - 可以發現由h1->h3的路徑S1->S3->S4產生出大約7 Mbps的流量(尚未超過門檻值 10Mbps)。

datapath	port	rx-pkts	rx-bytes	tx-pkts	tx-bytes	bandwidth
0000000000000002	1	61927	93726203	23	3547	7568990
0000000000000002	2	23	3547	61927	93726203	7568486
datapath	port	rx-pkts	rx-bytes	tx-pkts	tx-bytes	bandwidth
0000000000000001	1	23	3547	76926	116434689	7570000
0000000000000001	2	23	3547	23	3547	0
0000000000000001	3	76912	116431880	23	3547	7570000
0000000000000001	4	9	738	23	3547	0
datapath	port	rx-pkts	rx-bytes	tx-pkts	tx-bytes	bandwidth
0000000000000004	1	76927	116436203	23	3547	7569495
0000000000000004	2	23	3547	23	3547	0
0000000000000004	3	9	738	76928	116437717	7570000
datapath	port	rx-pkts	rx-bytes	tx-pkts	tx-bytes	bandwidth
0000000000000003	1	23	3547	23	3547	0
0000000000000003	2	23	3547	23	3547	0

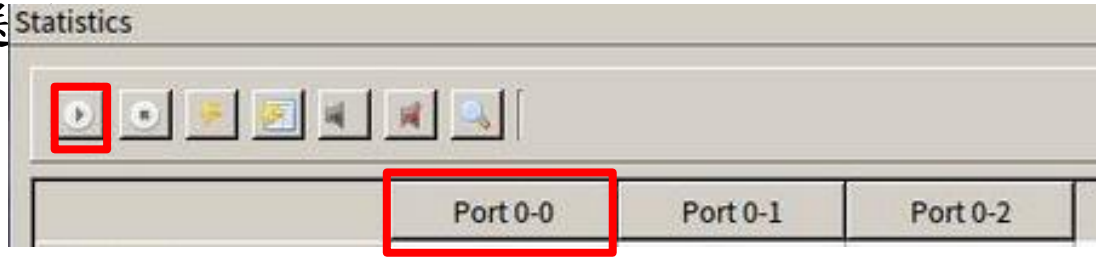
Lab 4-5: Dynamic Routing

- Step 13: 產生超過門檻值的流量
  - 在Mininet CLI底下輸入xterm h2，
    並在h2 CLI底下輸入sudo ostinato &。
  - 依照在h1新增Stream的步驟，
    同樣新增一個與h1相同大小的流量。
  - 注意: 來源端MAC位址須設為h2的MAC位址。



Lab 4-5: Dynamic Routing

- Step 14: 同樣按下Apply鍵後，按下Port 0-0，再按下發送



- Step 15: 觀察控制器視窗
  - 可以發現S1連接埠1瞬間產生了超過門檻值10 Mbps的流量(h1->h3加上h2->h3的流量)。

datapath	port	rx-pkts	rx-bytes	tx-pkts	tx-bytes	bandwidth
0000000000000001	1	23	3547	81226	122944889	11699687
0000000000000001	2	23	3547	23	3547	0
0000000000000001	3	8196	12394420	23	3547	4131201
0000000000000001	4	73026	110548476	23	3547	7568486

Lab 4-5: Dynamic Routing

- Step 16:觀察控制器視窗 (續)
  - 大約在3秒後(monitor的設定時間)，流量又回到原本大概7 Mbps左右，同時S3有流量產生。

```
datapath           port     rx-pkts  rx-bytes tx-pkts  tx-bytes bandwidth
----------------- -------- -------- -------- -------- -------- --------
0000000000000001         1       23     3547    96242 145679113  7578074
0000000000000001         2       23     3547    14981 22649959  7548804
0000000000000001         3    23184 35086252       23     3547  7563944
0000000000000001         4    88012 133237280       23     3547  7562934
datapath           port     rx-pkts  rx-bytes tx-pkts  tx-bytes bandwidth
----------------- -------- -------- -------- -------- -------- --------
0000000000000004         1    96244 145682141       23     3547  7575551
0000000000000004         2    14983 22652987       23     3547  7549813
0000000000000004         3        9      738   111204 168331581 15124860
datapath           port     rx-pkts  rx-bytes tx-pkts  tx-bytes bandwidth
----------------- -------- -------- -------- -------- -------- --------
0000000000000002         1    96243 145680627       23     3547  7577065
0000000000000002         2       23     3547    96243 145680627  7577065
datapath           port     rx-pkts  rx-bytes tx-pkts  tx-bytes bandwidth
----------------- -------- -------- -------- -------- -------- --------
0000000000000003         1    14982 22651473       23     3547  7549308
0000000000000003         2       23     3547    14982 22651473  7549308
```

Lab 4-5: Dynamic Routing

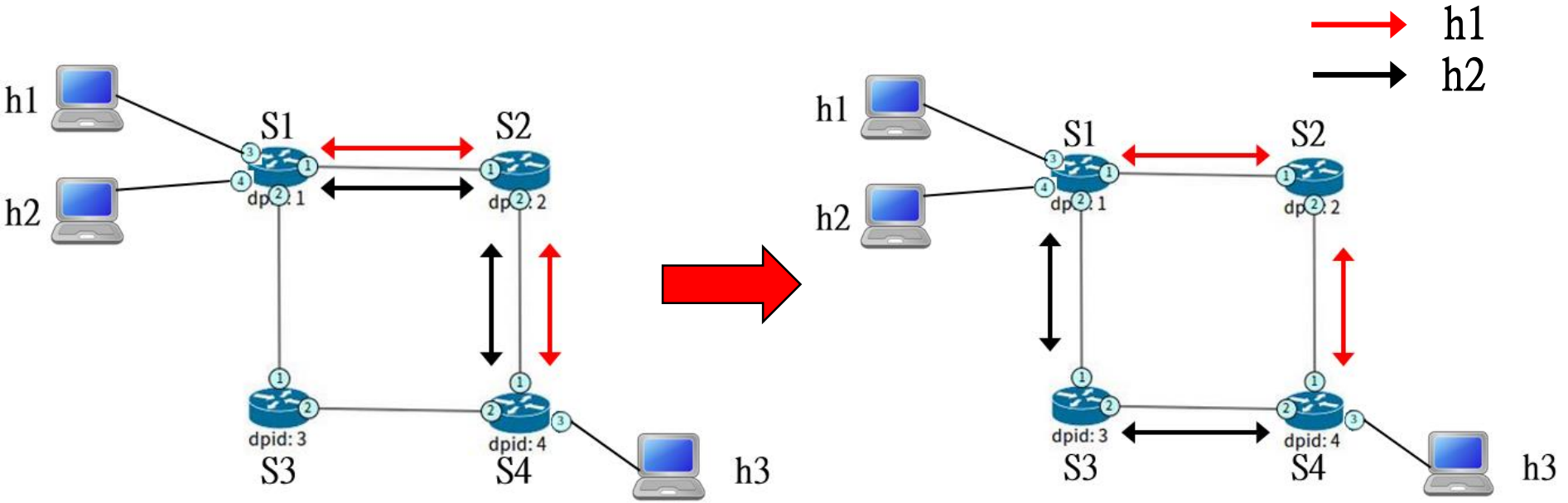
- Step 17: 查看S3裡的規則
  - 指令: ovs-ofctl –O OpenFlow13 dump-flows s3
  - 可以發現S3的路由表裡新增關於h2的規則。

```
 cookie=0x0, duration=698.172s, table=0, n_packets=3485326, n_bytes=5276783564,
idle_age=0, priority=5,dl_src=00:00:00:00:00:02 actions=output:2
 cookie=0x0, duration=698.172s, table=0, n_packets=0, n_bytes=0, idle_age=698, p
riority=5,dl_dst=00:00:00:00:00:02 actions=output:1
 cookie=0x0, duration=792.919s, table=0, n_packets=50, n_bytes=8132, idle_age=28
0, priority=0 actions=CONTROLLER:65535
```

  - 同樣的察看S1與S4裡的路由表裡，可以看到新增的h2規則。

Lab 4-5: Dynamic Routing

- 藉由以上實驗，我們完成了一個功能完整的控制器，從取得物理拓樸到動態路由機制，讓兩條流量分流達到負載平衡之目的(由7 Mbps->11 Mbps->7 Mbps)。



Thank you