Open VSwitch

2022-04-20

王定山

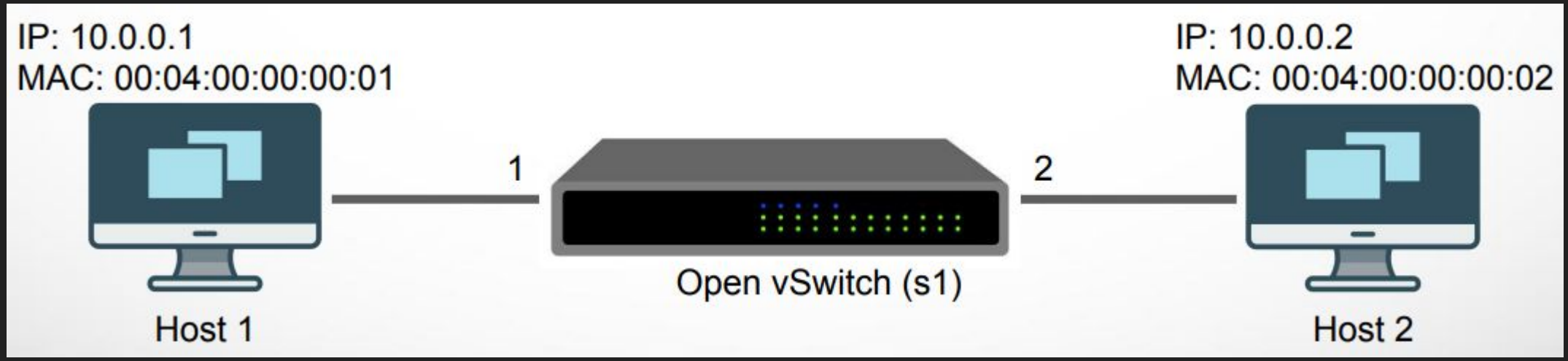
PPT: https://shorturl.at/psuTV

Meter

- A switch element that can measure and control the rate of packets.
- The meter trigger a meter band if the packet rate or byte rate passing through the meter exceed a predefined threshold.
- If the meter band drops the packet, it is called a **Rate Limiter**.

Meter

- Add meter entry (set maximum rate with 30 kbps)
  sudo ovs-ofctl add-meter **-O OpenFlow13** s1 meter=1,kbps,band=type=drop,rate=30000
- Add a flow entries with binding meter
  sudo ovs-ofctl add-flow **-O OpenFlow13** s1 in_port=1,actions=**meter:1**,output:2
  sudo ovs-ofctl add-flow **-O OpenFlow13** s1 in_port=2,actions=**meter:1**,output:1



Meter

mountain@openflow: ~/MininetTopology 63x4

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['40.9 Mbits/sec', '41.4 Mbits/sec']
mininet>
```

mountain@openflow: ~/MininetTopology 63x10

```
mountain@openflow:~/MininetTopology$ bash ./answer-TestbedA.sh
OFPST_METER_CONFIG reply (OF1.3) (xid=0x2):
meter=1 kbps bands=
type=drop rate=30000
 cookie=0x0, duration=0.068s, table=0, n_packets=0, n_bytes=0,
in_port="s1-eth1" actions=meter:1,output:"s1-eth2"
 cookie=0x0, duration=0.040s, table=0, n_packets=0, n_bytes=0,
in_port="s1-eth2" actions=meter:1,output:"s1-eth1"
mountain@openflow:~/MininetTopology$
```

mountain@openflow: ~/MininetTopology 65x4

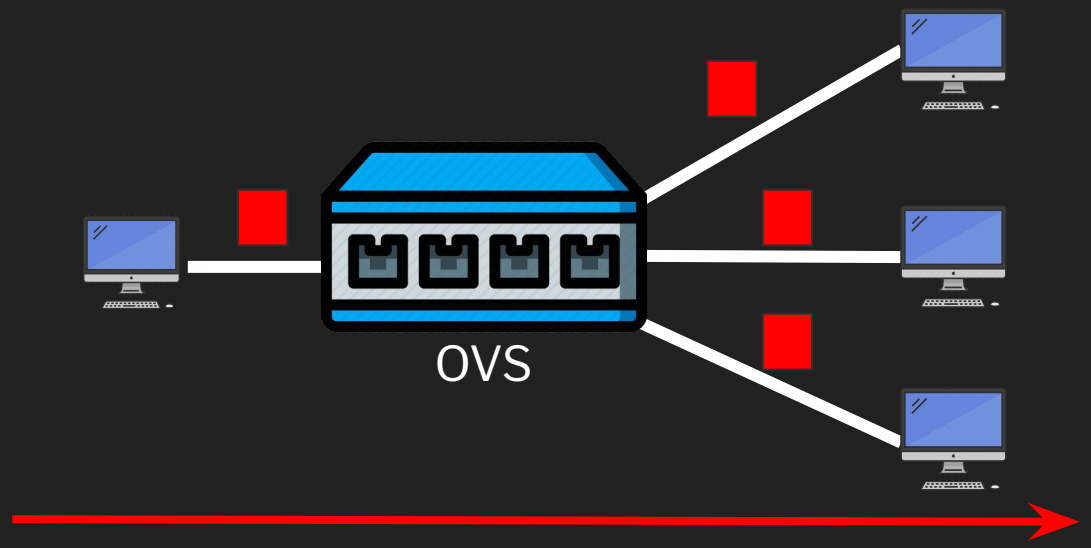
```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['95.7 Mbits/sec', '96.9 Mbits/sec']
mininet>
```

mountain@openflow: ~/MininetTopology 65x10

```
mountain@openflow:~/MininetTopology$ sudo ovs-ofctl dump-flows s1
 cookie=0x0, duration=13.602s, table=0, n_packets=0, n_bytes=0, i
n_port="s1-eth2" actions=output:"s1-eth1"
 cookie=0x0, duration=9.997s, table=0, n_packets=0, n_bytes=0, in
_port="s1-eth1" actions=output:"s1-eth2"
mountain@openflow:~/MininetTopology$
```

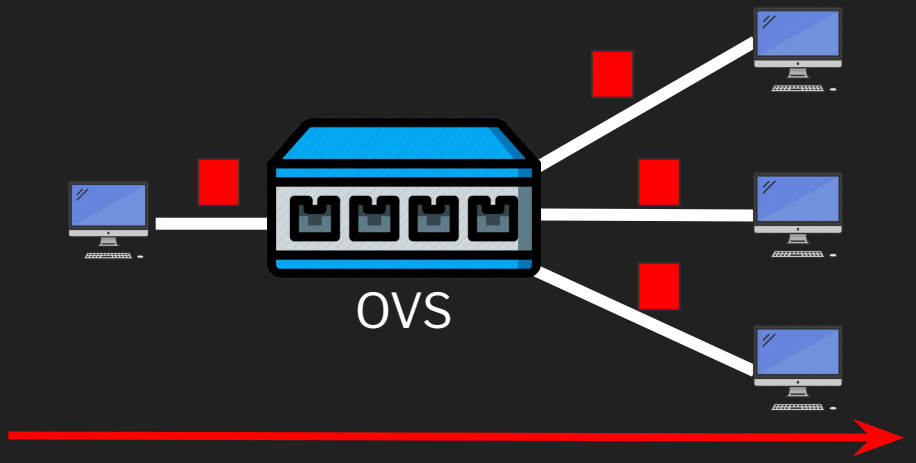
Groups

- Groups represent sets of actions for flooding, as well as more complex forwarding semantics (e.g. multipath, fast reroute, and link aggregation).
- As a general layer of indirection, groups also enable multiple flow entries to forward to a single identifier (e.g. IP forwarding to a common next hop).
- This abstraction allows common output actions across flow entries to be changed efficiently.



Groups

- Add group entry:
  sudo ovs-ofctl add-group s1
  group_id=1,type=all,bucket=mod_dl_src=00:00:00:11:11:11,mod_dl_dst=00:00:00:22:22:22,output:2,bucket=mod_dl_src=00:00:00:11:11:11,mod_dl_dst=00:00:00:22:22:22,output:3,bucket=mod_dl_src=00:00:00:11:11:11,mod_dl_dst=00:00:00:22:22:22,output:4
- Add flow entry with using group action:
  sudo ovs-ofctl add-flow s1 in_port=1,actions=group:1



Groups

```
mountain@openflow:~/MininetTopology$ sudo ovs-ofctl add-group s1 group_id=1,type=all,bucket=mod_dl_src=00:00:00:11:11:1
1,mod_dl_dst=00:00:00:22:22:22,output:2,bucket=mod_dl_src=00:00:00:11:11:11,mod_dl_dst=00:00:00:22:22:22,output:3,bucke
t=mod_dl_src=00:00:00:11:11:11,mod_dl_dst=00:00:00:22:22:22,output:4
mountain@openflow:~/MininetTopology$ sudo ovs-ofctl add-flow s1 in_port=1,actions=group:1
mountain@openflow:~/MininetTopology$ sudo ovs-ofctl dump-groups s1
NXST_GROUP_DESC reply (xid=0x2):
 group_id=1,type=all,bucket=bucket_id:0,actions=mod_dl_src:00:00:00:11:11:11,mod_dl_dst:00:00:00:22:22:22,output:"s1-et
h2",bucket=bucket_id:1,actions=mod_dl_src:00:00:00:11:11:11,mod_dl_dst:00:00:00:22:22:22,output:"s1-eth3",bucket=bucket
_id:2,actions=mod_dl_src:00:00:00:11:11:11,mod_dl_dst:00:00:00:22:22:22,output:"s1-eth4"
mountain@openflow:~/MininetTopology$ sudo ovs-ofctl dump-flows s1
 cookie=0x0, duration=21.017s, table=0, n_packets=1, n_bytes=70, in_port="s1-eth1" actions=group:1
```

```
mountain@openflow:~/MininetTopology$ sudo tcpdump -i s1-eth1 -e
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on s1-eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
15:08:06.056193 00:04:00:00:00:01 (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.
2 tell 10.0.0.1, length 28
15:08:07.076258 00:04:00:00:00:01 (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.
2 tell 10.0.0.1, length 28
```

Groups

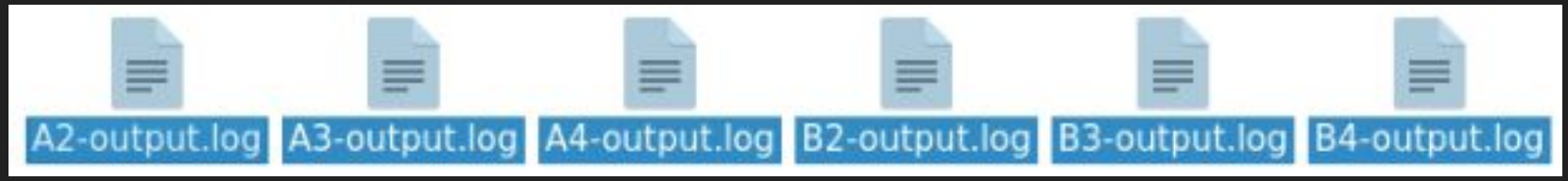
```
mountain@openflow:~/MininetTopology$ sudo tcpdump -i s1-eth2 -e
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on s1-eth2, link-type EN10MB (Ethernet), capture size 262144 bytes
15:07:58.884461 00:00:00:11:11:11 (oui Ethernet) > 00:00:00:22:22:22 (oui Ethernet), ethertype ARP (0x0806), length 42:
 Request who-has 10.0.0.2 tell 10.0.0.1, length 28
```

```
mountain@openflow:~/MininetTopology$ sudo tcpdump -i s1-eth3 -e
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on s1-eth3, link-type EN10MB (Ethernet), capture size 262144 bytes
15:07:49.668456 00:00:00:11:11:11 (oui Ethernet) > 00:00:00:22:22:22 (oui Ethernet), ethertype ARP (0x0806), length 42:
 Request who-has 10.0.0.2 tell 10.0.0.1, length 28
```

```
mountain@openflow:~/MininetTopology$ sudo tcpdump -i s1-eth4 -e
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on s1-eth4, link-type EN10MB (Ethernet), capture size 262144 bytes
15:08:31.652308 00:00:00:11:11:11 (oui Ethernet) > 00:00:00:22:22:22 (oui Ethernet), ethertype ARP (0x0806), length 42:
 Request who-has 10.0.0.2 tell 10.0.0.1, length 28
```

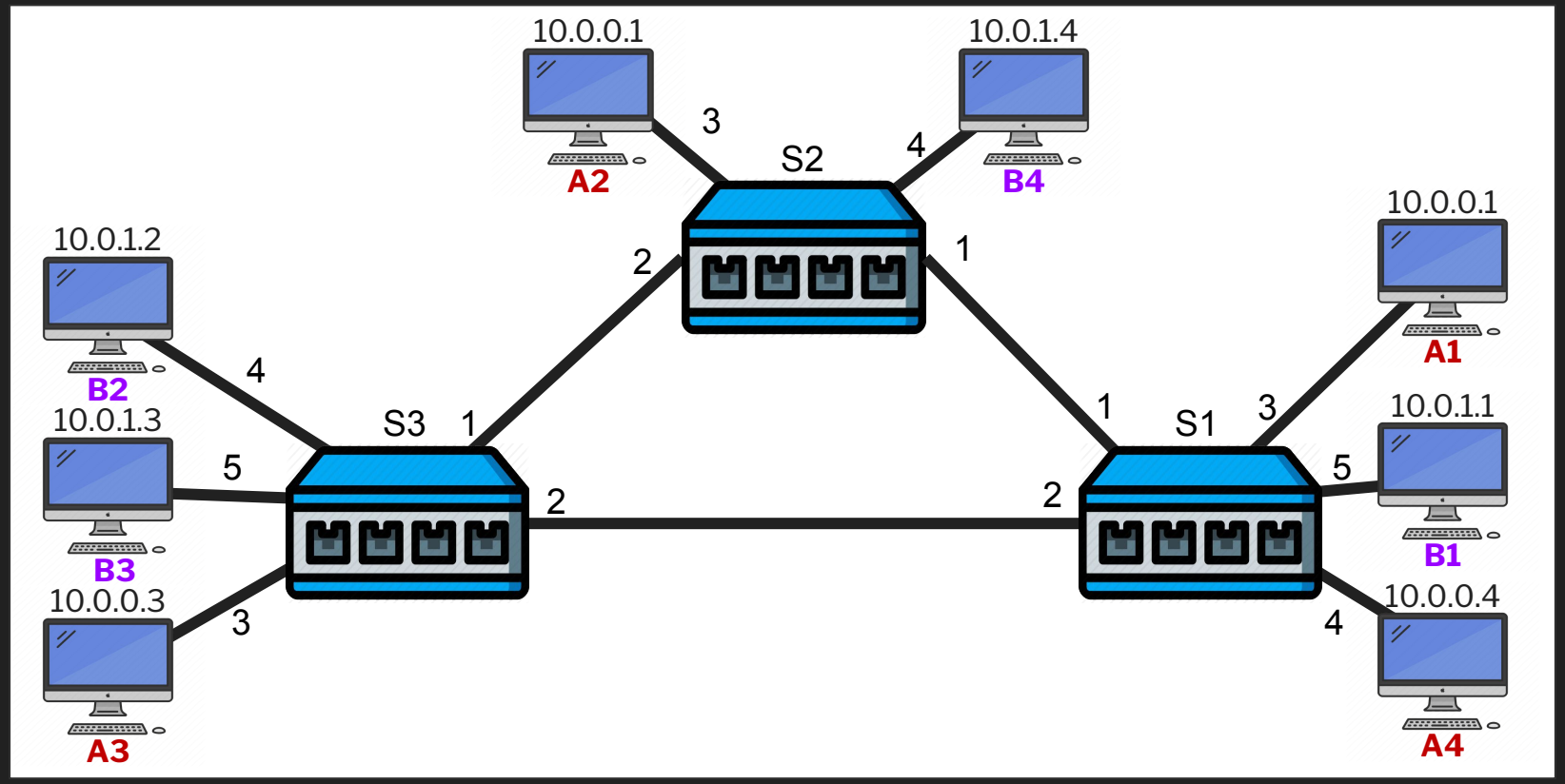
Experiment: IPv4 Multicasting

- Goal:
  - 編寫交換機S1,2,3路由規則，讓封包送到相同的Multicast group使用者。
- Multicast groups:
  - Group A: 10.0.0.0/24
  - Group B: 10.0.1.0/24
- 測試
  - 使用指令sudo python3 ./Lab2.py啟動Mininet測試環境
  - 使用xterm A1 B1 打開A1, B1之終端機
  - 於A1/B1終端機執TestScript_Lab1.py
    - A1: sudo python3 ./TestScript_Lab1.py 224.0.0.1
    - B1: sudo python3 ./TestScript_Lab1.py 224.0.0.2
  - 把記錄檔的內容擷圖表示即可。

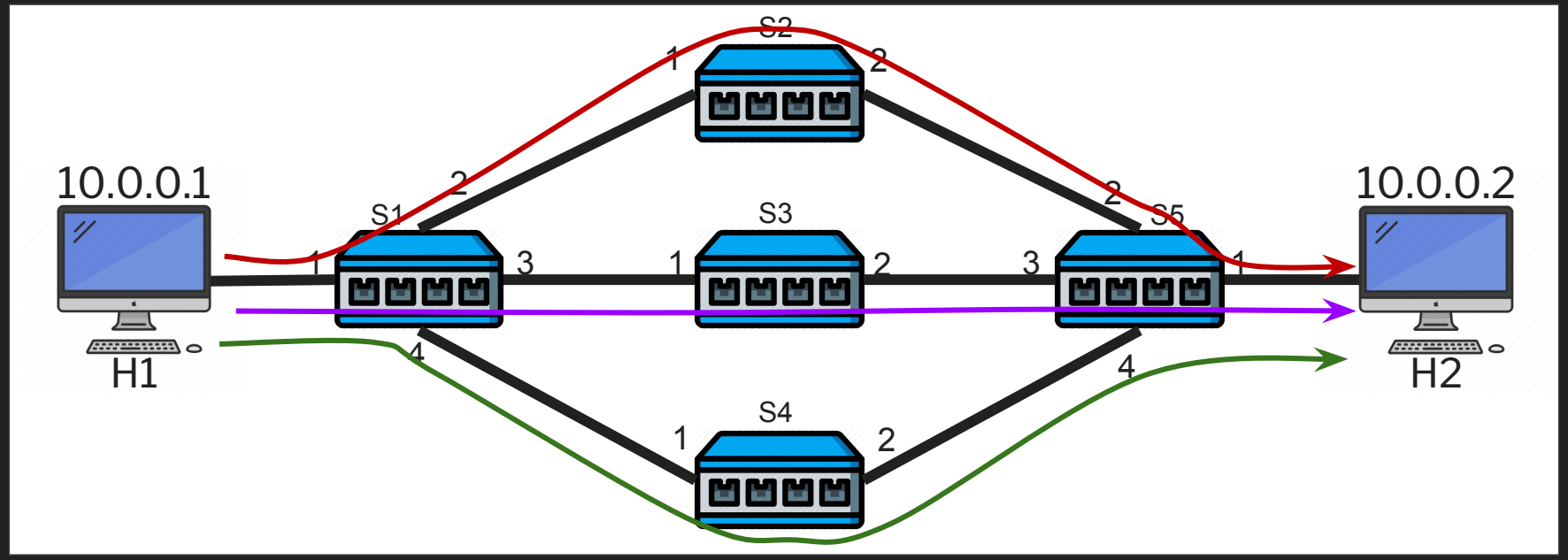


Experiment: IPv4 Multicasting

- Topology



Homework - IPv4 Routing with Meter Rate limiter



- 使用OpenVSwitch 指令對以下三條連線加入路由規則，並使用Meter 限定流量。
- 使用sudo ovs-ofctl dump-flows <bridge_id>把每一台交換機上的的規則印出並擷圖。
- 要求：使用IPv4 位址及UDP Port ID進行路由 (不需要寫ARP)
- 路線及使用的UDP Port
  - 紅色：UDP Port ID 50001，meter id = 1, drop, max bandwidth: 50Mbps
  - 藍色：UDP Port ID 50002，meter id = 2, drop, max bandwidth: 20Mbps
  - 綠色：UDP Port ID 50003，meter id = 3, drop, max bandwidth: 30Mbps

Homework - IPv4 Routing

1. 啟用環境: sudo python3 ./Homework.py
2. 編寫路由規則（Flow entries）
3. 於Mininet 使用xterm h1 打開Host 之終端機
4. 在h1的終端機上執行指令 *bash ./run_test.sh* 進行測試, 程式過程及結果會寫在log上。
5. 測試後把50001-s.log, 50002-s.log, 50003-s.log之檔案內容擷圖。

Hints:

編寫規則時要比對封包上ip_protocol 後才能比對UDP封包內容（PPT P.30）