

REPORT

I. GROUP Type

1. 經過 REST 上傳 group.json，group table 的 SELECT 會經過 hash，去選擇要執行哪個 buckets 的內容，換言之，雖然 flow 不會一次走 port2、一次走 port3，但經過幾次後，一定會走到另一個 bucket 的內容!

```
{
  "type": "SELECT",
  "appCookie": "0x1234abcd",
  "groupId": "1",
  "buckets": [
    {
      "treatment": {
        "instructions": [
          {
            "type": "OUTPUT",
            "port": "2"
          }
        ]
      },
      "weight": 1
    },
    {
      "treatment": {
        "instructions": [
          {
            "type": "OUTPUT",
            "port": "3"
          }
        ]
      },
      "weight": 1
    }
  ]
}
```

2. 經過 REST 一次上傳多個 flow rule，下圖為 switch 接收到來自 h1->h2 的 flow，會將 flow 送往 group

```
"flows": [
  {
    "priority": 42000,
    "timeout": 0,
    "isPermanent": true,
    "deviceId": "of:000000000000000001",
    "treatment": {
      "instructions": [
        {
          "type": "GROUP",
          "groupId": 1
        }
      ]
    },
    "selector": {
      "criteria": [
        {
          "type": "ETH_TYPE",
          "ethType": "0x0800"
        },
        {
          "type": "IN_PORT",
          "port": "1"
        },
        {
          "type": "IPV4_DST",
          "ip": "10.0.0.2/24"
        }
      ]
    }
  }
]
```

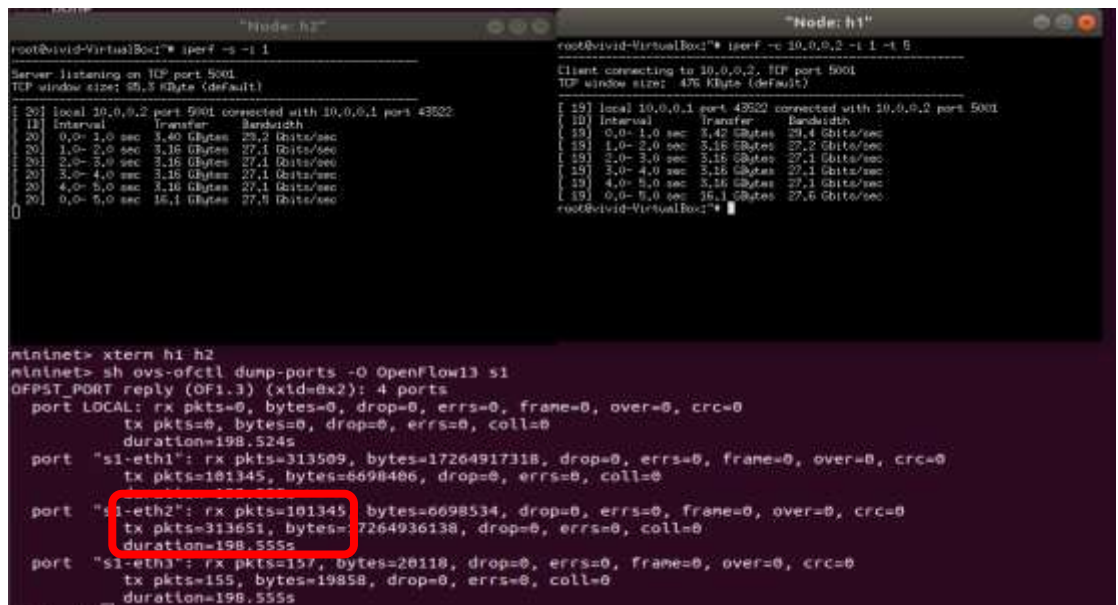
- 接下來就可以到 ONOS UI 看 FLOW 有沒有成功，並且在 Mininet 裡輸入"xterm h1 h2"，就可以利用 Iperf 打流量，並且輸入" sh ovs-ofctl dump-ports -O OpenFlow13 s1" 利用 switch1 觀察 h1->h2 的路徑!
- 還沒 iperf 前

```

OFFST_PORT reply (OF1.3) (xid=0x2): 4 ports
port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=0, bytes=0, drop=0, errs=0, coll=0
duration=5.079s
port "s1-eth1": rx pkts=7, bytes=646, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=21, bytes=2733, drop=0, errs=0, coll=0
duration=5.110s
port "s1-eth2": rx pkts=23, bytes=3113, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=20, bytes=2643, drop=0, errs=0, coll=0
duration=5.110s
port "s1-eth3": rx pkts=22, bytes=2903, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=20, bytes=2643, drop=0, errs=0, coll=0
duration=5.110s

```

- Iperf 後可以看到先走 s1->s2 的路徑

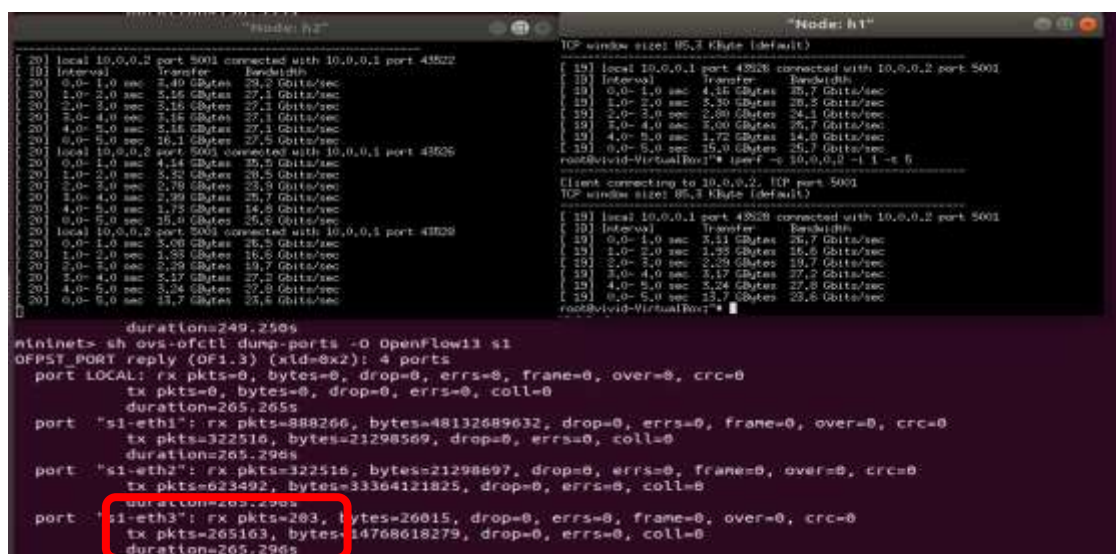


```

mininet> xterm h1 h2
mininet> sh ovs-ofctl dump-ports -O OpenFlow13 s1
OFFST_PORT reply (OF1.3) (xid=0x2): 4 ports
port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=0, bytes=0, drop=0, errs=0, coll=0
duration=198.524s
port "s1-eth1": rx pkts=313509, bytes=17264917318, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=101345, bytes=6698406, drop=0, errs=0, coll=0
duration=198.555s
port "s1-eth2": rx pkts=101345, bytes=6698534, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=313651, bytes=17264936138, drop=0, errs=0, coll=0
duration=198.555s
port "s1-eth3": rx pkts=157, bytes=20118, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=155, bytes=19858, drop=0, errs=0, coll=0
duration=198.555s

```

- 再經過 2 次 iperf 後，看到 flow 走 s1->s3 的路徑



```

mininet> sh ovs-ofctl dump-ports -O OpenFlow13 s1
OFFST_PORT reply (OF1.3) (xid=0x2): 4 ports
port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=0, bytes=0, drop=0, errs=0, coll=0
duration=265.265s
port "s1-eth1": rx pkts=888266, bytes=48132689632, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=322510, bytes=21298569, drop=0, errs=0, coll=0
duration=265.296s
port "s1-eth2": rx pkts=322510, bytes=21298597, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=623492, bytes=33364121825, drop=0, errs=0, coll=0
duration=265.296s
port "s1-eth3": rx pkts=203, bytes=26015, drop=0, errs=0, frame=0, over=0, crc=0
tx pkts=265163, bytes=14768618279, drop=0, errs=0, coll=0
duration=265.296s

```

II. Meter Type

1. 建立 `sudo mn --controller=remote --topo single,2` 的 topology
2. 輸入 `s1 ovs-vsctl set bridge "s1" datapath_type=netdev`，否則會有以下 error.

```
{"code":400,"message":"Must specify a cell id."}
```

3. 經過 REST 上傳 meter.json，將 TYPE 設定為 DROP，rate 設定 1000KB/s

```
{
  "deviceId": "of:0000000000000001",
  "unit": "KB_PER_SEC",
  "burst": true,
  "bands": [
    {
      "type": "DROP",
      "rate": 1000,
      "burstSize": 0,
      "prec": 0
    }
  ]
}
```

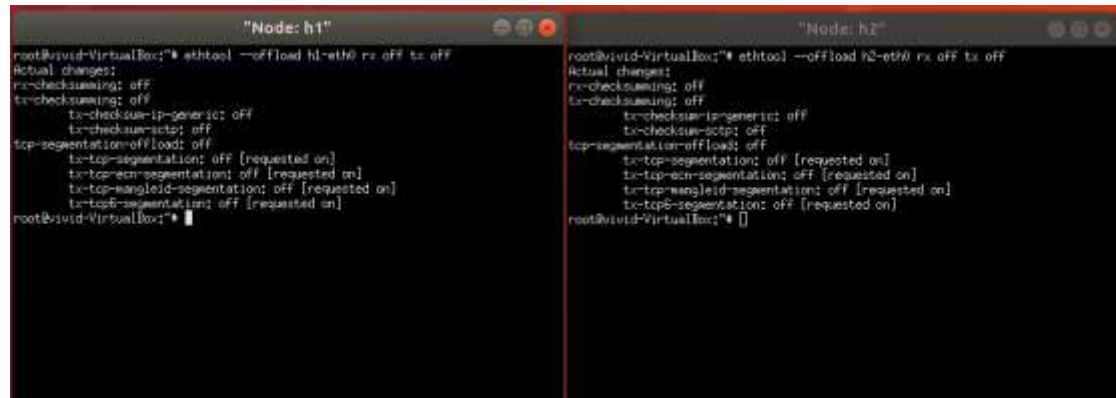
4. 經過 REST 上傳 flow_for_meter.json

```
"flows": [
  {
    "priority": 42000,
    "timeout": 0,
    "isPermanent": true,
    "deviceId": "of:0000000000000001",
    "treatment": {
      "instructions": [
        {
          "type": "METER",
          "meterId": 1
        },
        {
          "type": "OUTPUT",
          "port": "2"
        }
      ]
    },
    "selector": {
      "criteria": [
        {
          "type": "ETH_TYPE",
          "ethType": "0x0800"
        },
        {
          "type": "IN_PORT",
          "port": "1"
        },
        {
          "type": "IPV4_DST",
          "ip": "10.0.0.2/24"
        }
      ]
    }
  }
],
```

5. 設定完後同樣的開啟 h1 及 h2，並且設定 host 的虛擬網卡不要進行 rx 以及 tx 的 checksum，如此一來才能使 iperf 正常運行!

H1 輸入: `ethtool --offload h1-eth0 rx off tx off`

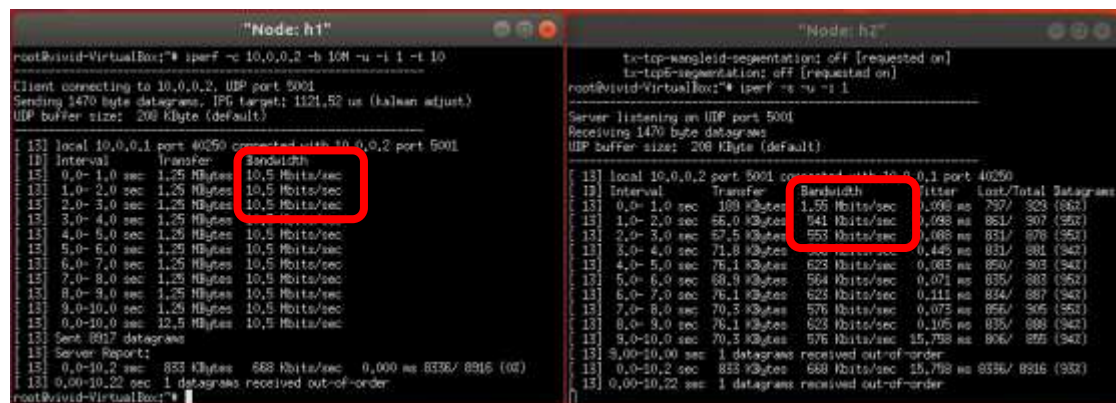
H2 輸入: `ethtool --offload h2-eth0 rx off tx off`



```
"Node: h1"
root@vivid-VirtualBox:~# ethtool --offload h1-eth0 rx off tx off
Actual changes:
rx-checksuming: off
tx-checksuming: off
tx-checksum-ip-generic: off
tx-checksum-sctp: off
tcp-segmentation-offload: off
tx-tcp-segmentation: off [requested on]
tx-tcp-ecn-segmentation: off [requested on]
tx-tcp-wangleid-segmentation: off [requested on]
tx-tcp6-segmentation: off [requested on]
root@vivid-VirtualBox:~#

"Node: h2"
root@vivid-VirtualBox:~# ethtool --offload h2-eth0 rx off tx off
Actual changes:
rx-checksuming: off
tx-checksuming: off
tx-checksum-ip-generic: off
tx-checksum-sctp: off
tcp-segmentation-offload: off
tx-tcp-segmentation: off [requested on]
tx-tcp-ecn-segmentation: off [requested on]
tx-tcp-wangleid-segmentation: off [requested on]
tx-tcp6-segmentation: off [requested on]
root@vivid-VirtualBox:~#
```

6. 利用 iperf 觀察流量變化，可以發現儘管 h1 送出 10M/s 的封包，h2 卻只收到的 bandwidth 低於 1M/s 的封包，有此可以見到 Meter 對 traffic speed 的限制!



```
"Node: h1"
root@vivid-VirtualBox:~# iperf -c 10.0.0.2 -b 10M -u -i 1 -t 10
Client connecting to 10.0.0.2, UDP port 5001
Sending 1470 byte datagrams, IPG target: 1121.52 us (kernal adjust)
UDP buffer size: 208 KByte (default)

[ 13] local 10.0.0.1 port 40250 connected with 10.0.0.2 port 5001
[ 13] Interval Transfer Bandwidth
[ 13] 0.0- 1.0 sec 109 KBytes 10.5 Mbits/sec
[ 13] 1.0- 2.0 sec 109 KBytes 10.5 Mbits/sec
[ 13] 2.0- 3.0 sec 109 KBytes 10.5 Mbits/sec
[ 13] 3.0- 4.0 sec 109 KBytes 10.5 Mbits/sec
[ 13] 4.0- 5.0 sec 109 KBytes 10.5 Mbits/sec
[ 13] 5.0- 6.0 sec 109 KBytes 10.5 Mbits/sec
[ 13] 6.0- 7.0 sec 109 KBytes 10.5 Mbits/sec
[ 13] 7.0- 8.0 sec 109 KBytes 10.5 Mbits/sec
[ 13] 8.0- 9.0 sec 109 KBytes 10.5 Mbits/sec
[ 13] 9.0-10.0 sec 109 KBytes 10.5 Mbits/sec
[ 13] 0.0-10.0 sec 12.5 KBytes 10.5 Mbits/sec
[ 13] Sent 8917 datagrams
[ 13] Server Report:
[ 13] 0.0-10.2 sec 833 KBytes 588 Mbits/sec 0.000 ms 8336/ 8916 (0%)
[ 13] 0.0-10.22 sec 1 datagrams received out-of-order
root@vivid-VirtualBox:~#

"Node: h2"
root@vivid-VirtualBox:~# iperf -s -u -i 1
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)

[ 13] local 10.0.0.2 port 5001 connected with 10.0.0.1 port 40250
[ 13] Interval Transfer Bandwidth
[ 13] 0.0- 1.0 sec 109 KBytes 1.55 Mbits/sec
[ 13] 1.0- 2.0 sec 55.0 KBytes 541 Kbits/sec
[ 13] 2.0- 3.0 sec 57.5 KBytes 563 Kbits/sec
[ 13] 3.0- 4.0 sec 71.8 KBytes 703 Kbits/sec
[ 13] 4.0- 5.0 sec 76.1 KBytes 623 Kbits/sec
[ 13] 5.0- 6.0 sec 68.9 KBytes 564 Kbits/sec
[ 13] 6.0- 7.0 sec 76.1 KBytes 623 Kbits/sec
[ 13] 7.0- 8.0 sec 70.3 KBytes 576 Kbits/sec
[ 13] 8.0- 9.0 sec 76.1 KBytes 623 Kbits/sec
[ 13] 9.0-10.0 sec 70.3 KBytes 576 Kbits/sec
[ 13] 0.0-10.0 sec 1 datagrams received out-of-order
[ 13] 0.0-10.2 sec 833 KBytes 588 Mbits/sec 15.798 ms 8336/ 8916 (93%)
[ 13] 0.0-10.22 sec 1 datagrams received out-of-order
root@vivid-VirtualBox:~#
```

III. Learn and solve:

1. 在看 ONOS dovs 裡的 GROUP 時，不需要有 weight，所以一直沒辦法成功上傳 group.json，之後到 onos 的 github 找才看到現在的版本是需要 weight!
2. 在設定 meter.json 的時候要將 band type 從 REMARK 改成 DROP 才能夠發揮限速的功能!因為兩者功能不同!

- **Optional: drop:** drop (discard) the packet. Can be used to define a rate limiter band.
- **Optional: dscp remark:** increase the drop precedence of the DSCP field in the IP header of the packet. Can be used to define a simple DiffServ policer.

3. 在使用 user space 的時候要關掉 checksum，原因是在 user space 的時候 rx 與 tx 的 checksum 會發生錯誤，而讓 host2 收不到封包!

參考資料: <https://mail.openvswitch.org/pipermail/ovs-discuss/2017-November/045764.html>