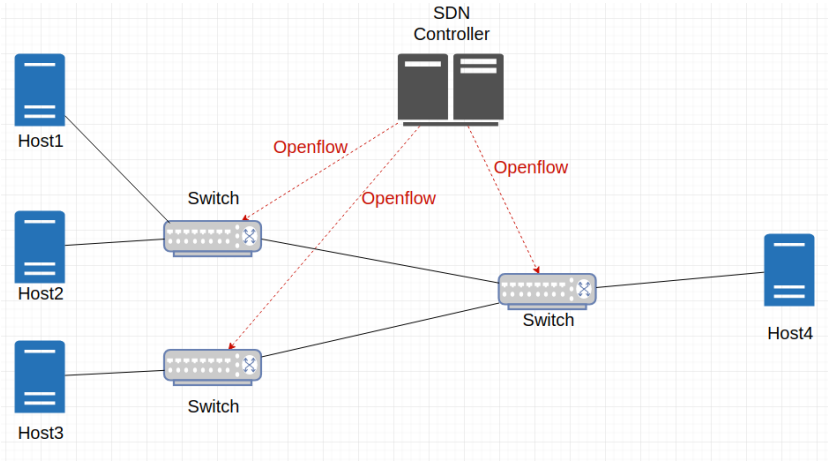
Network management Assignment3

李俊德，P76074478，蕭丞志，Q36074316
王昱翔，Q36074154，蘇致翰，P76074070

Introduction of SDN

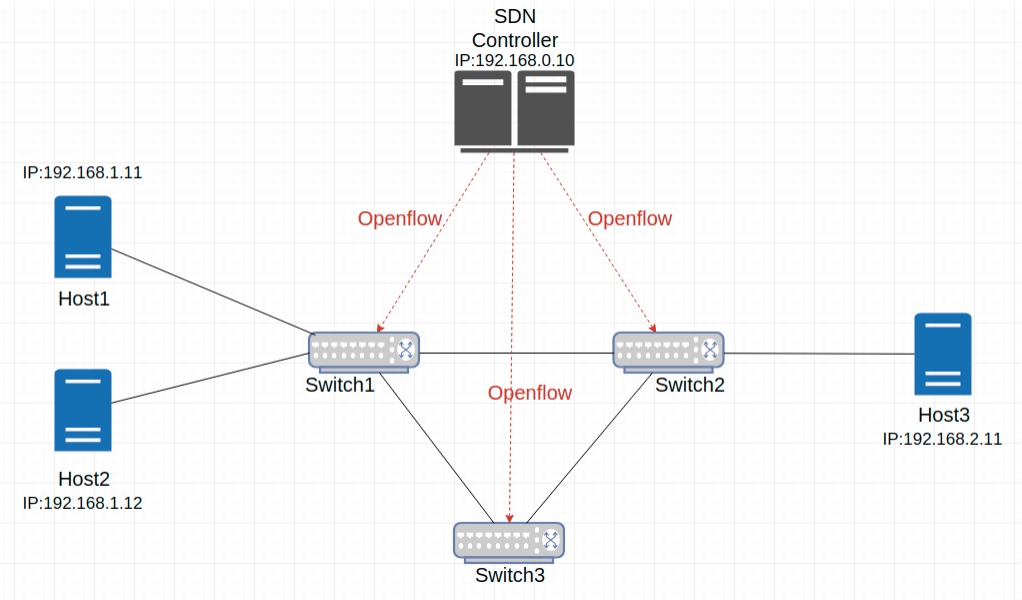
示意圖:



- 將 router 的 control plane 與 data plane 分離，這邊舉例利用 SDN controller 以軟體來控制網路行為，以 switch 來作為 data plane。
- SDN Controller 透過 Openflow Protocol 與 switch 溝通，訊息大致分為三種類型
  1. Controller-to-Switch message:
     由 controller 主動對 switch 發送，內容大多是對 switch 做設定，或是對 switch 做 request，要求 switch 回傳自己目前的狀態。
  2. Asynchronous message:
     由 switch 向 controller 傳輸，主要是要通知 switch 自身狀態的改變或是有需要 controller 幫忙處理的事情，一般會觸發 controller 的某個 event，由 controller 的 event handler 來處理這些事件。
  3. Symmetric message:
     這是在 controller 與 switch 為了建立溝通管道，雙方互相交換的訊息。

The network environment of our homework

Network architecture:



建立網路環境，主要是使用 mininet 建立虛擬網路環境，除了 controller 是使用本機。
以下操作都是在助教給的虛擬機環境：

1. 設定本機作為 SDN controller
   1. 先將虛擬機新增一張網卡
   2. 設定該網卡界面(enp0s8)
      1. $sudo vi /etc/network/interfaces
      2. 在檔案中加上
         ```
         auto enp0s8
         iface enp0s8 inet static
         address 192.168.0.10
         netmask 255.255.255.0
         ```
   3. $sudo ifdown enp0s8
   4. $sudo ifup enp0s8
2. 設定其他設備(三台主機及三台交換器)，交換器為 OpenvSwitch
   1. 撰寫 python 腳本使用 mininet 建立虛擬網路環境，預想中建立起來的網路拓樸應長跟設計架構一樣，腳本檔案附在 src 資料夾中。
   2. 執行 python 檔
      $sudo python network_setup.py
   3. 這時候因為還沒執行 controller，所以會顯示連接失敗是正常的。
   4. 萬一遇到 interface 已存在的錯誤可以使用以下指令清除先前建立的網路，然後再試一次。
      $sudo mn -c
   5. 檢查網路拓樸是否如預期的架構設計
      1. mininet> dump
      2. mininet> net
      3. mininet> sh ovs-ofctl -O OpenFlow13 show s1
      4. mininet> sh ovs-ofctl -O OpenFlow13 show s2
      5. mininet> sh ovs-ofctl -O OpenFlow13 show s3
   6. 從以上結果可以看出各個 switch 的每個 port 跟哪個設備相連，歸納如下：
      1. s1
         - port1: h1
         - port2: h2
         - port3: s2
         - port4: s3
      2. s2
         - port1: h3
         - port2: s1
         - port3: s3
      3. s3
         - port1: s1
         - port2: s2
   7. 確認網路狀態後，檢查 switch 狀態(以 s1 為例)
      1. 查看 switch 狀態
         mininet> sh ovs-vsctl show
         mininet> sh ovs-dpctl show
      2. 檢查協定及規則
         mininet> sh ovs-ofctl -O Openflow13 dump-flows s1
         目前還沒有任何規則被加入
   8. 以上都沒問題的話，環境架設差不多到這邊

Controller

以 python 的 ryu 套件實做，每個 switch 裡面有自己的規則。

1. Switch 1
   1. Table 0
      第一層實做 packet filter，我們過濾掉 TCP 層會到目標主機 port 22 的封包。
      如果封包會到目標主機 TCP 層 port 22，就送到 table3，否則送到 table1。
   2. Table 1
      - 如果目標主機的 ip 是 host1 就往 switch port1 送(先前已經知道 switch port1 是 host1)。
      - 如果目標主機的 ip 是 host2 就往 switch port2 送(先前已經知道 switch port2 是 host2)。
      - 如果目標主機的 ip 是 host3 就往 table2 送。
      - 剩餘的沒有規則會直接 drop。
   3. Table 2
      - Group table，讓來到這邊的封包平均送往 s2 及 s3。(以實驗結果來看是失敗的，目前全部都會送往 s2，還找不出原因)
   4. Table 3
      Drop the packet，來這邊的封包都是 table0 過濾掉的，所以不用給規則直接丟棄就可以了。
2. Switch 2
   1. Table 0
      1. 同 switch 1 過濾封包
   2. Table 1
      - 如果目標主機的 ip 是 host3 就往 switch port1 送(先前已經知道 switch port1 是 host3)。
      - 如果目標主機是 h1 或 h2 就往 table2 送。
      - 剩餘的沒有規則會直接 drop。
   3. Table 2
      - Group table，讓來到這邊的封包平均送往 s1 及 s3。(以實驗結果來看是失敗的，目前全部都會送往 s1，還找不出原因)
   4. Table 3
      Drop the packet，來這邊的都是被過濾掉的封包，直接丟棄。
3. Switch 3
   將從 s1 來的送到 s2，s2 來的送到 s1。

Demonstration of our homework

以下為從頭到尾的執行過程及結果截圖，請助教測試的時候將本機環境設定一張 IP: 192.168.0.10 的網卡(如前面提到的網路環境)，謝謝助教。

1. 建立網路環境
   ```
   $ sudo mn -c
   $ sudo python network_setup.py
   ```
2. 執行 controller
   ```
   $ ryu-manager HW3-ryu.py
   ```
3. 編輯測試腳本
   ```
   $ vi testfile.sh
   ```
   裡面有四筆測試資料分別模擬不同情境，以第一筆為例，模擬從 h1 產生一個封包經過 s1 到 h2。
   ```
   'ovs-appctl ofproto/trace s1 tcp,nw_src=192.168.1.11,nw_dst=192.168.1.12'
   ```
4. 執行腳本，可以在 mininet 的終端機執行也可以新開終端機
   1. 使用 mininet 執行腳本
      ```
      mininet> sh /bin/sh ./testfile.sh
      ```
   2. 新開一個 terminal
      ```
      $ sudo /bin/sh testfile.sh
      ```

5. 以下為各個測資的執行結果
   1. 模擬一個封包從 h1 經過 s1 到達 h2

```
#generate a packet from h1 to h2 (test general function)
ovs-appctl ofproto/trace s1 tcp,nw_src=192.168.1.11,nw_dst=192.168.1.12
```

```
nm@nm-VirtualBox:~/workspace/Network_Management_HW3$ sudo /bin/sh testfile.sh
Bridge: s1
Flow: tcp,in_port=ANY,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_src=192.168.1.11,nw_dst=192.168.1.12,nw_tos=0
,nw_ecn=0,nw_ttl=0,tp_src=0,tp_dst=0,tcp_flags=0

Rule: table=0 cookie=0 priority=0
OpenFlow actions=goto_table:1

        Resubmitted flow: tcp,in_port=ANY,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_src=192.168.1.11,nw_dst=1
92.168.1.12,nw_tos=0,nw_ecn=0,nw_ttl=0,tp_src=0,tp_dst=0,tcp_flags=0
        Resubmitted regs: reg0=0x0 reg1=0x0 reg2=0x0 reg3=0x0 reg4=0x0 reg5=0x0 reg6=0x0 reg7=0x0
        Resubmitted  odp: drop
        Resubmitted megaflow: recirc_id=0,tcp,in_port=ANY,nw_dst=192.168.1.12,nw_frag=no,tp_dst=0x0/0x2
        Rule: table=1 cookie=0 priority=2,tcp,nw_dst=192.168.1.12
        OpenFlow actions=output:2

Final flow: unchanged
Megaflow: recirc_id=0,tcp,in_port=ANY,nw_dst=192.168.1.12,nw_frag=no,tp_dst=0x0/0x2
Datapath actions: 1
nm@nm-VirtualBox:~/workspace/Network_Management_HW3$
```

可以看出最後封包成功送到 s1 的 port2(h2)

   2. 模擬一個封包從 h1 經過 s1 到達 h2，且在 TCP 層指定到 port 22

```
#generate a packet from h1 to h2 port 22 (test filter)
ovs-appctl ofproto/trace s1 tcp,nw_src=192.168.1.11,nw_dst=192.168.1.12,tcp_dst=22
```

```
nm@nm-VirtualBox:~/workspace/Network_Management_HW3$ sudo /bin/sh testfile.sh
Bridge: s1
Flow: tcp,in_port=ANY,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_src=192.168.1.11,nw_d
st=192.168.1.12,nw_tos=0,nw_ecn=0,nw_ttl=0,tp_src=0,tp_dst=22,tcp_flags=0

Rule: table=0 cookie=0 priority=1,tcp,tp_dst=22
OpenFlow actions=goto_table:3

        Resubmitted flow: tcp,in_port=ANY,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_s
rc=192.168.1.11,nw_dst=192.168.1.12,nw_tos=0,nw_ecn=0,nw_ttl=0,tp_src=0,tp_dst=22,tcp_flags=0
        Resubmitted regs: reg0=0x0 reg1=0x0 reg2=0x0 reg3=0x0 reg4=0x0 reg5=0x0 reg6=0x0 reg7=0x0
        Resubmitted  odp: drop
        Resubmitted megaflow: recirc_id=0,tcp,in_port=ANY,nw_frag=no,tp_dst=22
        Rule: table=254 cookie=0 priority=0,reg0=0x2
        OpenFlow actions=drop

Final flow: unchanged
Megaflow: recirc_id=0,tcp,in_port=ANY,nw_frag=no,tp_dst=22
Datapath actions: drop
nm@nm-VirtualBox:~/workspace/Network_Management_HW3$
```

可以看出最後封包被 drop。

   3. 模擬一個封包從 h1 經過 s1, s2 到達 h3

```
#generate a packet from h1 to h3 (test group table)
ovs-appctl ofproto/trace s1 tcp,nw_src=192.168.1.11,nw_dst=192.168.2.11
ovs-appctl ofproto/trace s2 tcp,nw_src=192.168.1.11,nw_dst=192.168.2.11
```

```
Bridge: s1
Flow: tcp,in_port=ANY,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_src=192.168.1.11,nw_d
st=192.168.2.11,nw_tos=0,nw_ecn=0,nw_ttl=0,tp_src=0,tp_dst=0,tcp_flags=0

Rule: table=0 cookie=0 priority=0
OpenFlow actions=goto_table:1

        Resubmitted flow: tcp,in_port=ANY,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_s
rc=192.168.1.11,nw_dst=192.168.2.11,nw_tos=0,nw_ecn=0,nw_ttl=0,tp_src=0,tp_dst=0,tcp_flags=0
        Resubmitted regs: reg0=0x0 reg1=0x0 reg2=0x0 reg3=0x0 reg4=0x0 reg5=0x0 reg6=0x0 reg7=0x0
        Resubmitted  odp: drop
        Resubmitted megaflow: recirc_id=0,tcp,in_port=ANY,nw_dst=192.168.2.11,nw_frag=no,tp_dst=0x0/0x2
        Rule: table=1 cookie=0 priority=2,tcp,nw_dst=192.168.2.11
        OpenFlow actions=goto_table:2

                Resubmitted flow: unchanged
                Resubmitted regs: reg0=0x0 reg1=0x0 reg2=0x0 reg3=0x0 reg4=0x0 reg5=0x0 reg6=0x0 reg7=0x0
                Resubmitted  odp: drop
                Resubmitted megaflow: recirc_id=0,tcp,in_port=ANY,nw_dst=192.168.2.11,nw_frag=no,tp_dst=0x0/0x2
                Rule: table=2 cookie=0 priority=0
                OpenFlow actions=group:0

Final flow: unchanged
Megaflow: recirc_id=0,tcp,in_port=ANY,vlan_tci=0x0000/0x1fff,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,n
w_src=192.168.1.11,nw_dst=192.168.2.11,nw_frag=no,tp_src=0,tp_dst=0
Datapath actions: 3
```

```
Bridge: s2
Flow: tcp,in_port=ANY,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_src=192.168.1.11,nw_d
st=192.168.2.11,nw_tos=0,nw_ecn=0,nw_ttl=0,tp_src=0,tp_dst=0,tcp_flags=0

Rule: table=0 cookie=0 priority=0
OpenFlow actions=goto_table:1

        Resubmitted flow: tcp,in_port=ANY,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_s
rc=192.168.1.11,nw_dst=192.168.2.11,nw_tos=0,nw_ecn=0,nw_ttl=0,tp_src=0,tp_dst=0,tcp_flags=0
        Resubmitted regs: reg0=0x0 reg1=0x0 reg2=0x0 reg3=0x0 reg4=0x0 reg5=0x0 reg6=0x0 reg7=0x0
        Resubmitted  odp: drop
        Resubmitted megaflow: recirc_id=0,tcp,in_port=ANY,nw_dst=192.168.2.11,nw_frag=no,tp_dst=0x0/0x2
        Rule: table=1 cookie=0 priority=1,tcp,nw_dst=192.168.2.11
        OpenFlow actions=output:1

Final flow: unchanged
Megaflow: recirc_id=0,tcp,in_port=ANY,nw_dst=192.168.2.11,nw_frag=no,tp_dst=0x0/0x2
Datapath actions: 11
```

可以看出封包成功送到 s1 的 group table 之後送到 port3(s2)，這邊嘗試很多次到 group table 都不會走 port4，可是在 controller python code 裡面他們兩條路徑的 weight 都是設定 50，這部份還沒找到原因。

接下來是到 s2，可以看出封包最後送到 s2 的 port1(h3)。

4. 模擬一個封包從 h3 經過 s2, s3 到達 h2

```
#generate a packet from h3 to h2 (test group table)
ovs-appctl ofproto/trace s2 tcp,nw_src=192.168.1.11,nw_dst=192.168.1.12
ovs-appctl ofproto/trace s3 tcp,nw_src=192.168.1.11,nw_dst=192.168.1.12
```

```
nm@nm-VirtualBox:~/workspace/Network_Management_HW3$ sudo /bin/sh testfile.sh
Bridge: s2
Flow: tcp,in_port=ANY,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_src=192.168.1.11,nw_d
st=192.168.1.12,nw_tos=0,nw_ecn=0,nw_ttl=0,tp_src=0,tp_dst=0,tcp_flags=0

Rule: table=0 cookie=0 priority=0
OpenFlow actions=goto_table:1

        Resubmitted flow: tcp,in_port=ANY,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_s
rc=192.168.1.11,nw_dst=192.168.1.12,nw_tos=0,nw_ecn=0,nw_ttl=0,tp_src=0,tp_dst=0,tcp_flags=0
        Resubmitted regs: reg0=0x0 reg1=0x0 reg2=0x0 reg3=0x0 reg4=0x0 reg5=0x0 reg6=0x0 reg7=0x0
        Resubmitted  odp: drop
        Resubmitted megaflow: recirc_id=0,tcp,in_port=ANY,nw_dst=192.168.1.12,nw_frag=no,tp_dst=0x0/0x2
        Rule: table=1 cookie=0 priority=2,tcp,nw_dst=192.168.1.12
        OpenFlow actions=goto_table:2

                Resubmitted flow: unchanged
                Resubmitted regs: reg0=0x0 reg1=0x0 reg2=0x0 reg3=0x0 reg4=0x0 reg5=0x0 reg6=0x0 reg7=0x0
                Resubmitted  odp: drop
                Resubmitted megaflow: recirc_id=0,tcp,in_port=ANY,nw_dst=192.168.1.12,nw_frag=no,tp_dst=0x0/0x2
                Rule: table=2 cookie=0 priority=0
                OpenFlow actions=group:0

Final flow: unchanged
Megaflow: recirc_id=0,tcp,in_port=ANY,vlan_tci=0x0000/0x1fff,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,n
w_src=192.168.1.11,nw_dst=192.168.1.12,nw_frag=no,tp_src=0,tp_dst=0
Datapath actions: 10
```

```
Bridge: s3
Flow: tcp,in_port=ANY,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_src=192.168.1.11,nw_d
st=192.168.1.12,nw_tos=0,nw_ecn=0,nw_ttl=0,tp_src=0,tp_dst=0,tcp_flags=0

Rule: table=0 cookie=0 priority=0
OpenFlow actions=FLOOD

Final flow: unchanged
Megaflow: recirc_id=0,ip,in_port=ANY,nw_frag=no
Datapath actions: 7,8,6
```

可以看出封包送到 s2 的 group table，但跟先前一樣，每次執行的結果都是一樣的，不會依照 weight 平分封包。

下面是直接模擬如果有要給 h2 的封包到達 s3 的話，s3 會把他用 flood 的方式送出去，其實這邊只要將 s2 來的送到 s1，s1 來的送到 s2 就可以了。

The contribution of each member

1. 李俊德
   1. Make the spec
   2. Network setup

    3. Integration test
    4. Fix bug
    5. Write the report
2. 蕭丞志
    1. table 0 of s1 and s2
    2. s3
    3. Unit test
    4. Fix bug
3. 王昱翔
    1. table 1 of s1 and s2
    2. Unit test
    3. Fix bug
4. 蘇致翰
    1. table 2 of s1 and s2
    2. Unit test
    3. Fix bug

Reference

- SDN 架構簡介
- ryu furewall 簡易實做
- Learning-SDN
- ryu group table
- 深入 OpenFlow 協定 看 Group Table 設定流程
- ryu packet generate
- connect ryu controller to switch
- mininet 介紹
- github:John-Lin/simple_loop.py
- 牛的大腦
- github:osrg/ryu
- ryu document
- 資訊人筆記
- openflow spec