教育部補助「5G 行動寬頻跨校教學聯盟計畫」

下世代 Network Slicing 模組設：開源軟體透過 Queue 設計實現保障優先度及流量管控

實驗單元：Mininet 搭配 Controller 使用 OpenFlow 模擬 SDN 網路環境

授課教師：李宗南

教材編撰：陳 陞

目錄

一、 課程目標

1. 課程目標 1：讓學生了解如何修改 switch 內部指令。
2. 課程目標 2:讓學生了解 Queue 的原理以及如何設計不同 Queue 的權重值以及優先度。
3. 課程目標 3：透過 Queue 來實現網路切片的流量保障以及優先度。

二、 實驗設備

1. 硬體:

電腦 : Ubuntu 作業系統 16.04

2. 軟體：

Mininet：2.3.0d4

gcc 編譯器

Python 2.072

vi/vim 文字編輯器

三、 技術介紹

1. Switch 架構介紹： 當 Data frame 進入到 Switch Interface 會先進行 classification 以及 making，也可同時先做 policing，但對於 router 來說，通常判斷 packet 的重要性，是透過利用 header 的 IPP 或者 DSCP 來判斷 packet 的重要性，但在 switch 的部份 Data frame header 中多一個選

項，就是參考 Cos(class of service)值，它跟 IPP 有點相似，是一個 3bit 長度的值，由 0-7 來判斷 Data frame 的重要性。

2. Cos 介紹： Class of Service(CoS)，值是 0-7，透過設定 Cos 能過更精確的設定重要的 data frame。

3. Queue 方法介紹：

Strictly Priority Queuing (SPQ)：這個方法很簡單，就是 Q7 送完換 Q6 送依序下去如果 Q7 一直沒送完則之後的 Queue 都不能送了

Weighted Fair Queuing (WFQ)：每個 Queue 都可以設定權重，Switch 會依據這些權重來分配最基本的保證頻寬，所以每個 Queue 都會有機會送出去資料。

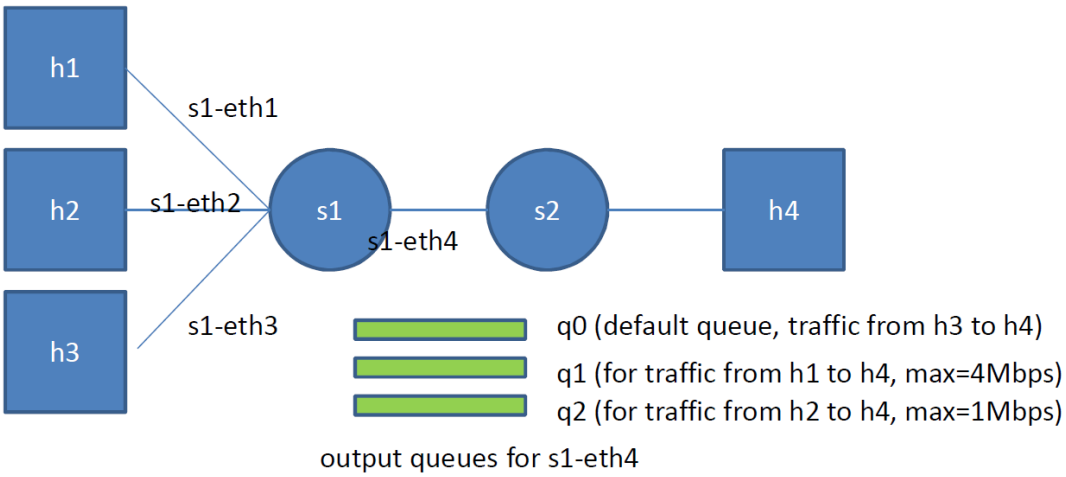
Weighted Round Robin Scheduling (WRR)：Round Robin Scheduling 的作法是指像排隊一樣，輪到了某一個 Queue 他就擁有全部的頻寬送完換下一個，自己就排到最後 WRR 也是用相同的作法，但是會根據 priority 跟 weight 去決定傳送的頻寬。

四、 實驗以及執行步驟教學

首先安裝 Mininet，在”在 Mininet 模擬環境下實現速率控制 “這個實驗已經撰寫過如何安裝，這邊跳過。

Step 1：參考 ”在 Mininet 模擬環境下實現速率控制 “撰寫出下圖拓譜的.py 檔

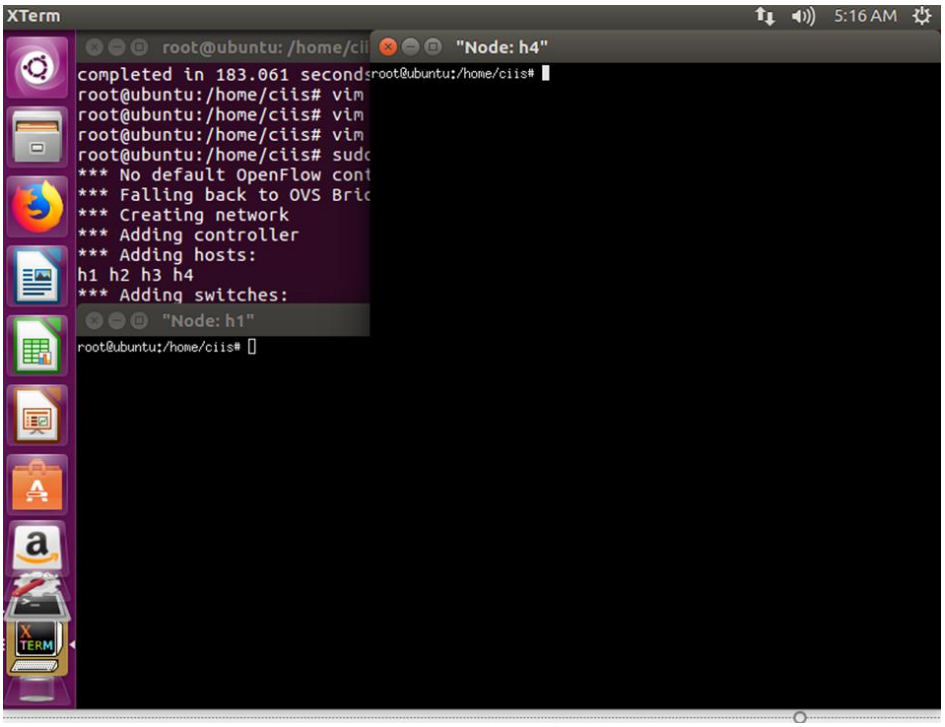
透過 Mininet 建立拓譜，拓譜如下圖：



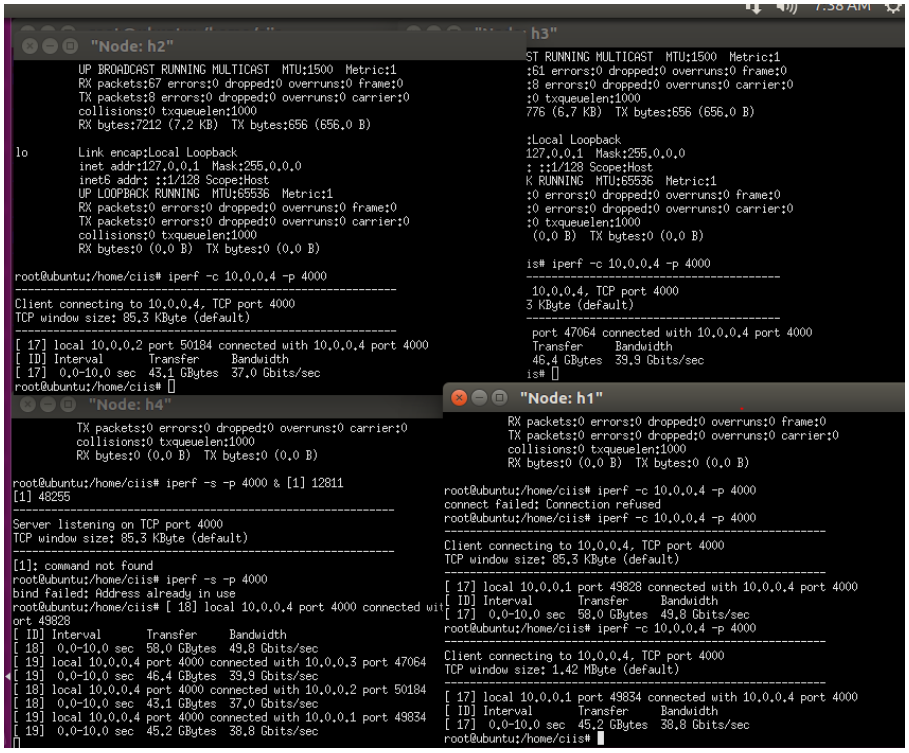
撰寫一個 controller.py 用來調整 switch queue 設定，並且監聽剛剛用 Mininet 建立的拓譜，並透過 Mininet 模擬一般使用情況製造流量，以驗證建立的 queue 是否能夠跟預期一樣效果。

```
root@ubuntu: /home/ciis
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.util import dpidToStr
log = core.getLogger()
s1_dpid=0
s2_dpid=0
def _handle_ConnectionUp (event):
global s1_dpid, s2_dpid
print "ConnectionUp: ",
dpidToStr(event.connection.dpid)
#remember the connection dpid for switch
for m in event.connection.features.ports:
if m.name == "s1-eth1":
s1_dpid = event.connection.dpid
print "s1_dpid=", s1_dpid
elif m.name == "s2-eth1":
s2_dpid = event.connection.dpid
print "s2_dpid=", s2_dpid
~
~
~
~
~
"controller.py" 18L, 509C
```

透過 Mininet xterm 功能針對各個節點，並且分別透過 iperf 指令從 h1、h2、h3 到 h4 做測試一開始的速率。如下圖：

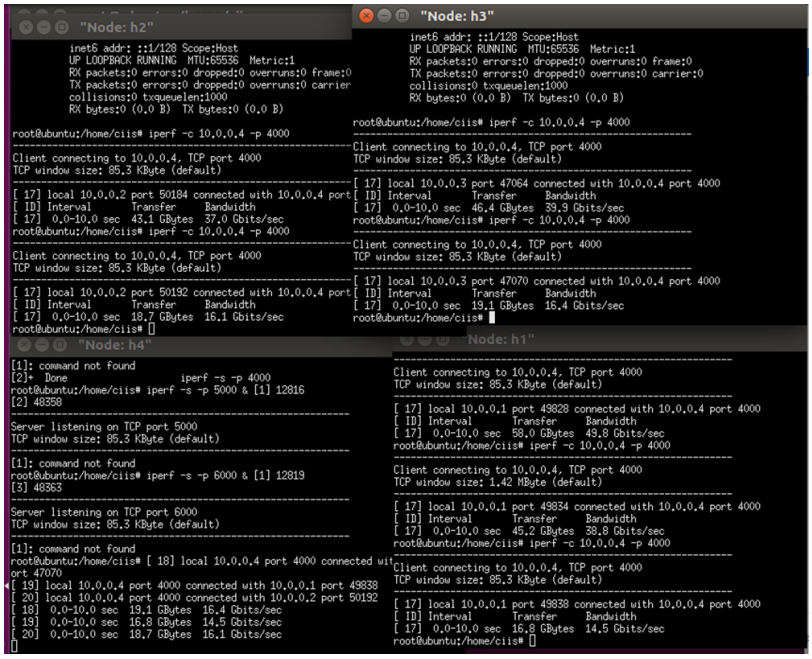


透過 iperf 分別進行測試，將 h4 當作 server 監聽 port4000，可以發現再分開測試中三個 host 到達 h4 的上傳以及下載速度是相距不多的。

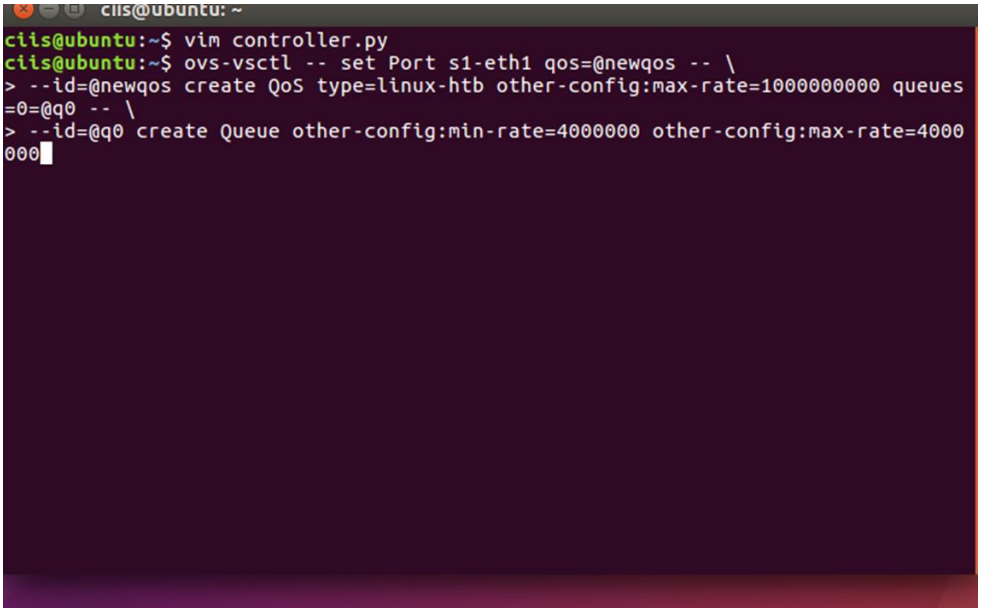


透過 iperf 分別進行測試，將 h4 當作 server 監聽 port4000，可以發現再同

時測試中三個 host 到達 h4 的上傳下載速度有什麼不同，並將原因解釋在你的回答中。

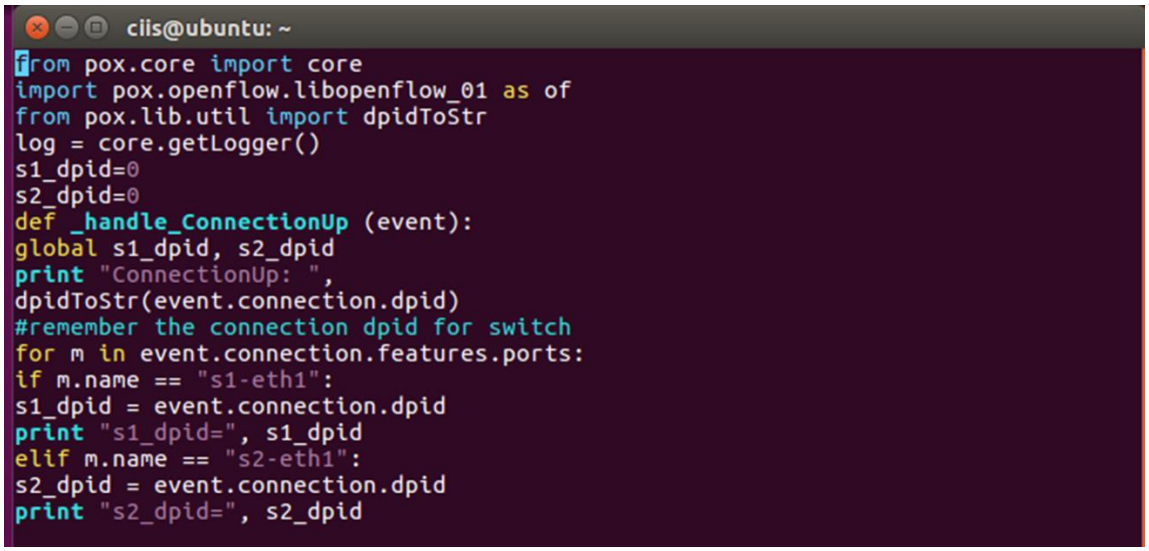


接下來加入 controller.py 來調整 queue 的優先度，然後再次進行測試，g.首先建立三個 queue 針對 s1-eth4，接下來調整 queue 設定將 h1、h2、h3 的流量分別導向 queue0、 queue1、 queue2，如下圖分別建立三條 queue。



接下來開啟另一個終端機開啟剛剛的 Mininet 模擬拓譜，另一個開啟 controller.py，再進行一次剛剛的實驗，並且將結果截圖以及試著分析解析為什麼

會有不同原因為何，controller.py 如下圖。



五、 實驗要求

任務 1：參考實驗步驟拓譜圖，撰寫拓譜.py 將檔案內容以截圖方式呈現。

任務 2：參考實驗步驟，透過 Mininet xterm 功能，分別對 h1、h2、h3 到 h4 同時做測試以及分開做測試，並且將實驗過程以及結果截圖方式呈現，並且在檔案中需附上對於這 2 個不同結果的解釋。

任務 3：參考實驗步驟，撰寫 controller.py，並且透過指令添加 3 個 queue，分別對 h1、h2、h3 到 h4 做測試，並且將實驗過程以及結果截圖方式呈現，並且在檔案中需附上對於結果的解釋。

六、 參考資料

[1]https://guiderworld.blogspot.com/2009/01/queue-method.html

[2]https://www.jannet.hk/zh-Hant/post/quality-of-service-qos-switch/

[3]https://www.southampton.ac.uk/~drn1e09/ofertie/openflow_qos_mininet.pdf