**電腦網路實驗實驗報告 < 軟體定義網路 >**

**姓名: 翁佳煌　　　學號: 409430030**

# 實驗名稱

Mininet & SDN Controller

# 實驗目的

**課堂實驗1和2:**

使用 OpenFlow 控制器後，當 h1 透過 ping 命令向 h2 發送 ICMP 封包時，封包是否會被傳送到控制器進行處理，並透過控制器將封包廣播到除了發送方之外的所有接口。透過觀察 xterm 中執行的 tcpdump，可以了解此處理過程的細節，並理解 OpenFlow 控制器與傳統集線器的運作差異。實驗1會看到相同的 ARP 和對應的ICMP封包，實驗2會看到三個未應答的 ARP 請求在 tcpdump xterms 中。

**課堂實驗3:**  
熟悉 OpenFlow 協定及其在網路交換機中的應用。在這個實驗中，使用~pox/pox/misc裡面的of\_tutorial.py和 Mininet 創建一個網路拓撲，透過控制器設定 OpenFlow 規則，控制網路交換機轉發封包。具體而言，實驗中將透過控制器向交換機發送控制消息，指示交換機根據特定的規則轉發封包，並觀察實驗結果中交換機行為的變化，以了解 OpenFlow 控制器與傳統交換機的不同之處。

# 實驗設備

**Linux作業系統之電腦。**

**Download & Install Mininet:**

$ cd

$ git clone https://github.com/mininet/mininet

$ ./mininet/util/install.sh -a

**安裝pox controller:**

$ git clone git://github.com/noxrepo/pox

# 實驗步驟

**課堂實驗1和2:**

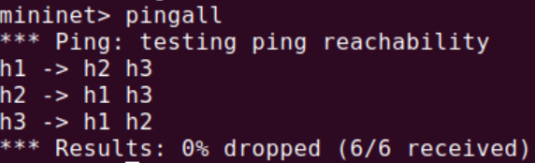
**步驟1.**

首先開啟第一個terminal，先cd到~/pox這個路徑裡，然後輸入./pox.py log.level --DEBUG misc.of\_tutorial。

**步驟2.**

開啟第二個Terminal，先輸入sudo mn -c 清除之前的紀錄，以防止未知錯誤發生，接著輸入sudo mn --topo single,3 --mac --switch ovsk --controller remote。

**步驟3.**

在第二個terminal輸入mininet> pingall確認是否成功連線。

**步驟4.**

接著在第二個terminal輸入create xterm for each host mininet> xterm h1 h2 h3，然後，

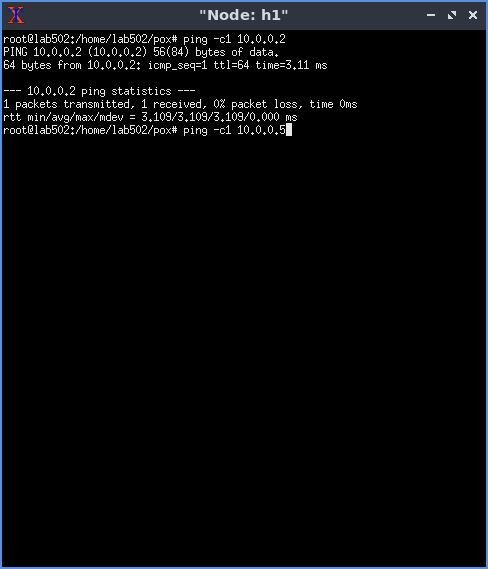
In xterm for h2 輸入 tcpdump -X -n -i h2-eth0

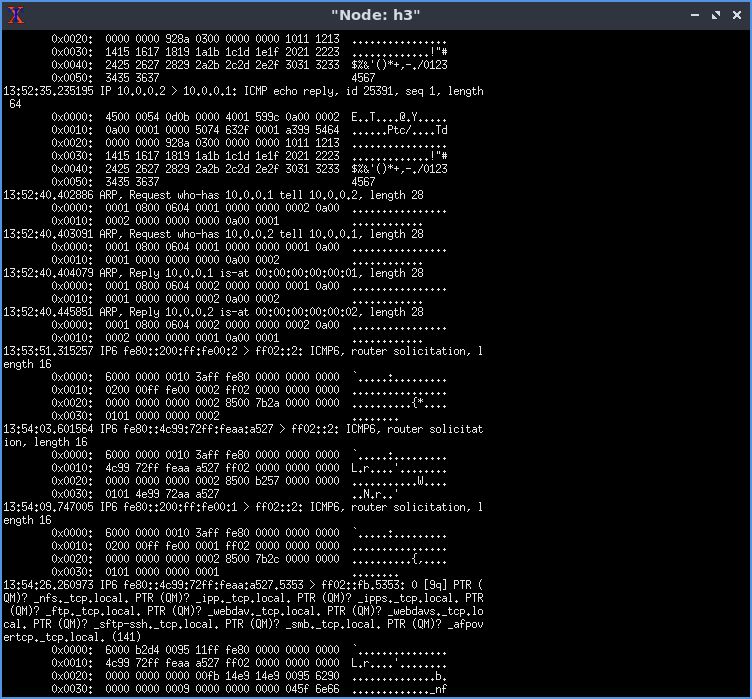
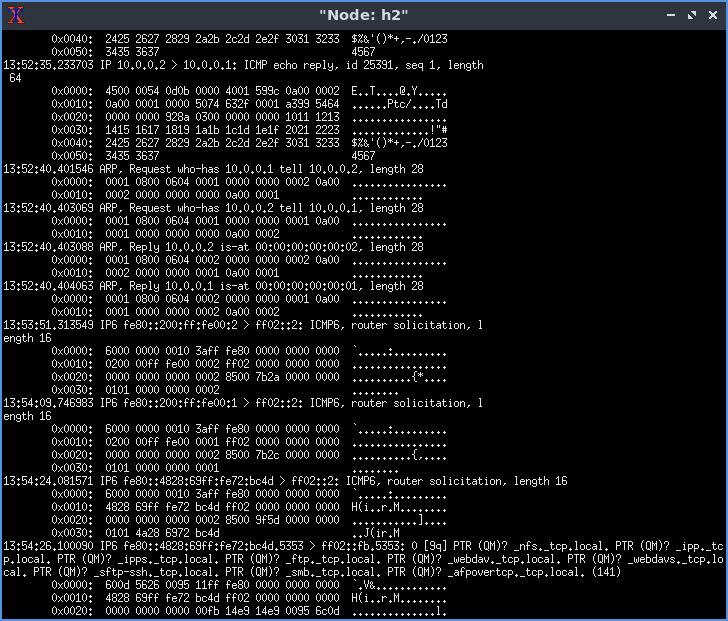
In xterm for h3 輸入 tcpdump -X -n -i h3-eth0

In xterm for h1 輸入 ping -c1 10.0.0.2 (實驗1和實驗2步驟相似只差在實驗2在此輸入ping -c1 10.0.0.5)

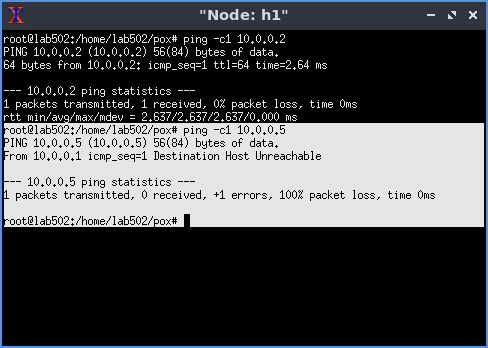
**步驟5.**

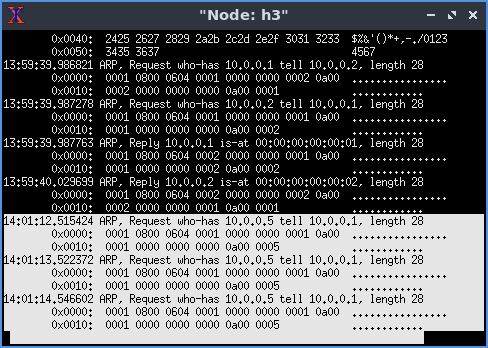
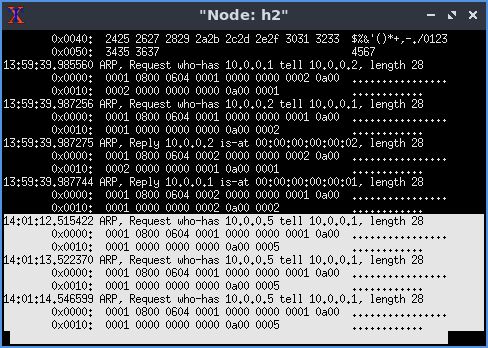
課堂實驗1觀察結果圖:





課堂實驗2觀察結果圖:





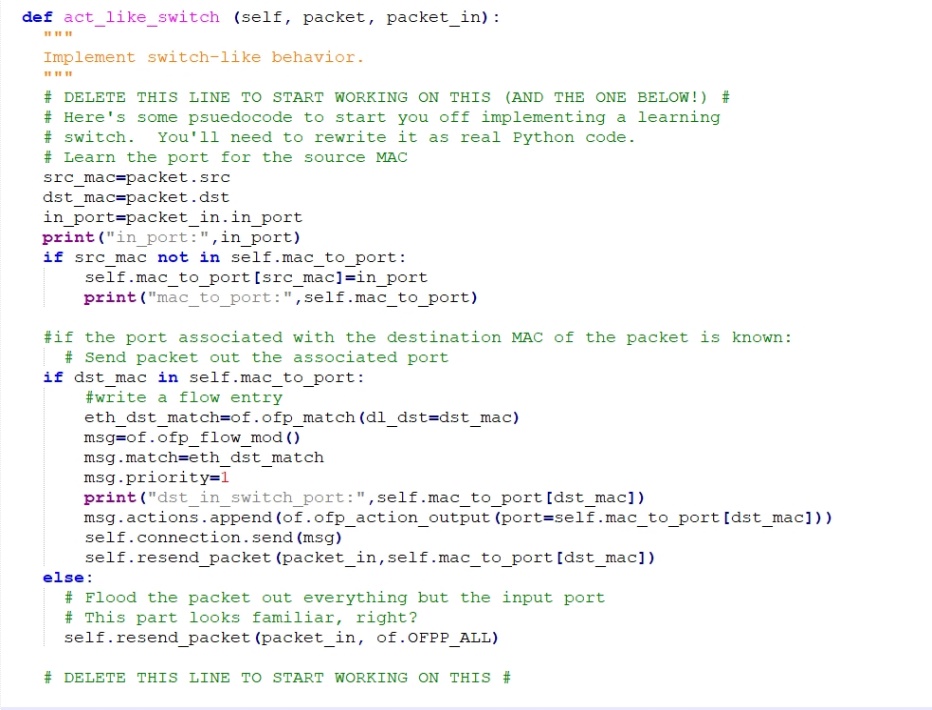
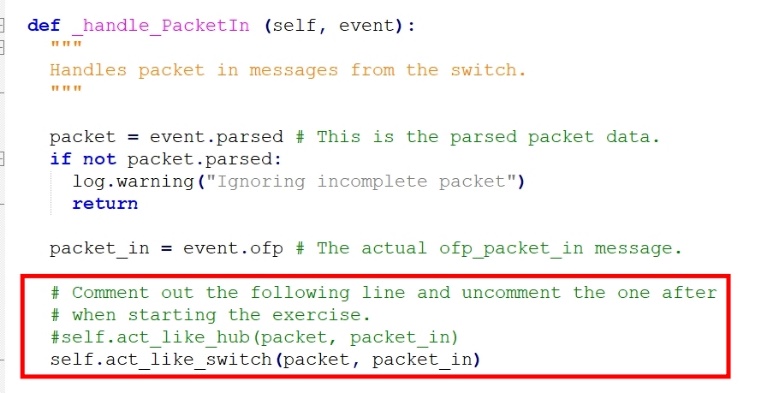
**課堂實驗3:**

**步驟1.**

開啟第一個Terminal，並cd ~/pox/pox/misc，然後輸入gedit of\_tutorial.py。

**步驟2.**

修改相關函式，如下圖1。



▲圖1

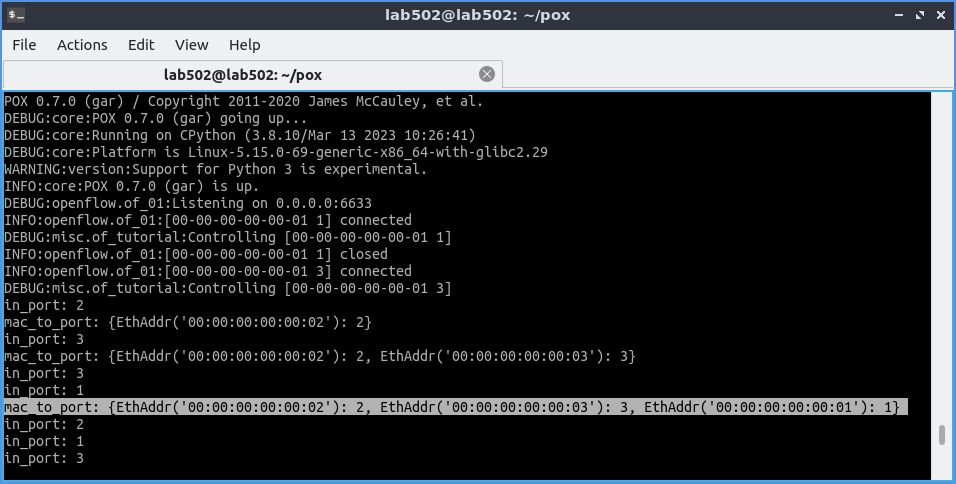
**步驟3.**

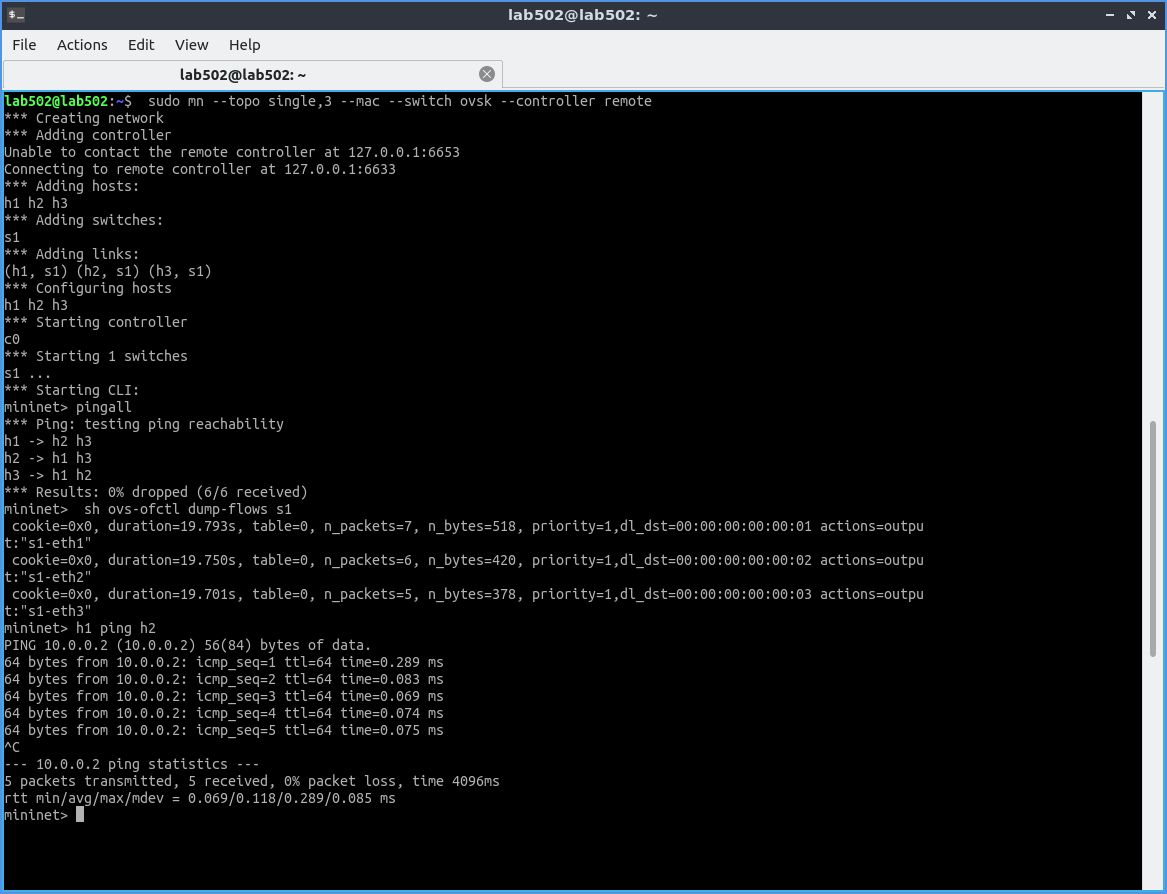
重啟POX 和Mininet ,接著輸入:  
mininet> h1 ping h2

mininet> sh ovs-ofctl dump-flows s1

**步驟4.**

觀察結果。





# 問題與討論

**課堂實驗1和2:**

第一個實驗中，觀察到ping數據包被傳送到控制器，然後被控制器洪泛到除了發送端口之外的所有接口。這種行為在一個簡單的網路中可能是可接受的，但在一個大型網路中，可能會導致控制器被大量流量淹沒，從而影響整個網路的性能。因此，這個實驗提醒我們，在實際網路中需要適當的OpenFlow規則，以最小化控制器的參與和最大化網路的自主性。

第二個實驗中觀察到的三個未回覆的ARP請求是因為在Mininet拓撲中只有三個主機（h1，h2和h3），因此當主機h1向10.0.0.5發出ARP請求時，因為該IP地址沒有被分配給任何主機，因此不會有任何主機回應這個ARP請求，從而導致三個ARP請求都未被回覆。

**課堂實驗3:**

在 Part A 中，當控制器向交換機發送控制消息時，交換機會根據該消息直接轉發封包，而不需要事先設置 OpenFlow 規則。

在 Part B 中，控制器除了向交換機發送控制消息外，還設置了一條 OpenFlow 規則，用來匹配封包的源 MAC 地址和目的 MAC 地址。當封包匹配到該規則時，交換機會根據規則中的指令轉發封包，而不需要再向控制器請求轉發指示。此外，在 mininet 命令行中可以使用 ovs-ofctl dump-flows s1 命令查看交換機的流表規則，以驗證規則是否已經配置成功。

# 心得與感想

這次實驗的重點在於控制器和OpenFlow交換機之間的通訊協議，以及如何使用控制器來控制網路流量。透過實驗，我了解到OpenFlow控制器使用OpenFlow協議來控制網路交換機，並且可以通過控制器來設置交換機上的流規則，以實現特定的網路流量控制。因為實驗環境已經被預先設置好了，所以我不需要自己配置環境。我只需要根據指示來操作，就可以完成整個實驗1和2，實驗3需要另外思考如何改py程式碼。此外，教材中也提供了詳細的指導和說明，這對我來說非常有幫助。

總體來說，這次的OpenFlow實驗讓我對軟體定義網路有了更深入的了解，也讓我學會了如何設置OpenFlow網路和控制網路流量。我相信這些知識和技能將對我的未來研究和工作非常有用。

# 參考文獻

<https://github.com/mininet/mininet>

<http://github.com/Hsun111/MininetTopology>

<https://www.vmware.com/tw/topics/glossary/content/software-defined-networking.html>

<https://ithelp.ithome.com.tw/articles/10235434>

<https://www.openedu.tw/course?id=1002>