§摘要:

單一路徑傳輸方法常有頻寬不足的問題產生，造成壅塞、封包傳送失敗等情形，此論文提出了一種多路徑傳輸方案，用不相交的路徑減少碰撞的發生，使得傳輸品質能夠提升。

§背景知識:

* Software-Defined Networking(SDN):

軟體定義網路(SDN)並不是一項具體的技術，而是一種網路設計概念、一種網路架構。在傳統網路中，封包的轉發行為是逐條控制的，每個網路設備中(例如交換器)有特定的處理能力和配置方式，這種控制是完全分散式的。

而SDN則把每個設備的**控制面從設備中脫離出來**，**放到一個統一的的外部伺服器**(即是這篇報告用到的**controller)**，這個控制器知道所有設備的資訊，且可以透過開放的API被上層的應用程式**通過程式設計控制**，就像這篇報告中我們透過撰寫ryu controller來控制網路拓樸中的交換器。  
如此可消除手動配置的過程，增加管理員對網路環境的視野，提高部署的效率。

* Openflow:

Openflow屬於OSI第二層中的一種網路通訊協定。一般認為openflow是第一個SDN的標準之一，它最初在SDN環境中定義了通訊協定，使SDN控制器能夠與物理和虛擬的交換機和路由器等網路設備的轉發平面直接進行互動，藉此改變網路封包所走的網路路徑。

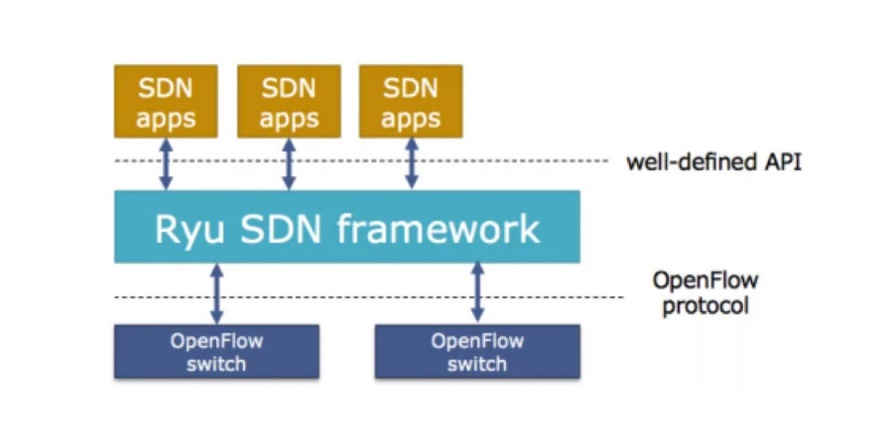
* Ryu Controller

SDN控制器五花八門，不同的控制器有不同的優點和特性。

Ryu因其架構簡單，部署方便，便於擴展得以廣泛應用，尤其適用於中小型網絡架構的流量調度和控制。Ryu是日本NTT公司主導開發的開源項目，其字面意思是日語中「Flow」的意思。

Ryu用python語言編寫，目標是提供一個擁有邏輯上集中控制能力的SDN作業系統，該系統具有設計完備的API接口，使網絡應用者能夠便捷的創建新的管理和控制應用。

▼Ryu總體框架

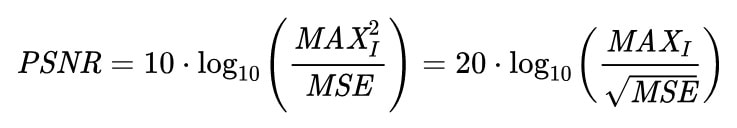


Ryu SDN framework主要提供控制能力，通過北向接口的Rest API為SDN apps提供服務，供SDN apps調度和控制流量和網絡。通過南向接口的Openflow等協議控制openflow交換機，完成流量交互。

* PSNR

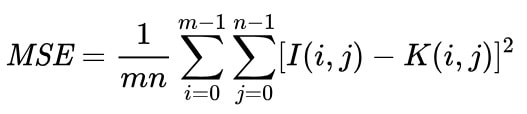
PSNR是峰值雜訊比(Peak signal-to-noise ratio)的縮寫。

PSNR 是一種用來表示訊號**最大可能功率**和影響它表示精度的**破壞性雜訊功率**的**比值**，而在影像裡面我們就可以用 PSNR 這種比較客觀(有一個量化數據) 的方法來**計算影像的失真**。在影像壓縮，有分為失真壓縮跟非失真壓縮。如果是失真壓縮的話，影像在經過壓縮、解壓縮之後，必定會有一些資訊遺失，我們想知道解壓縮出來的圖像跟原圖到底差多少，進而去評估這個壓縮算法好不好。PSNR定義如下:



PSNR的單位是用－分貝(dB) 來算，其中的 MAXI 為訊號的最大強度，例如在每個pixel 點用8-bit 表示的影像裡面就是 255。

MSE 就是統計學裡面講的均方誤差(mean-square error)，公式如下：



看到數學公式頭就開始痛了，以程式來表示其實更為直觀:

for( i = 0; i < m; i++){

　　for( j = 0; j < n; j++){

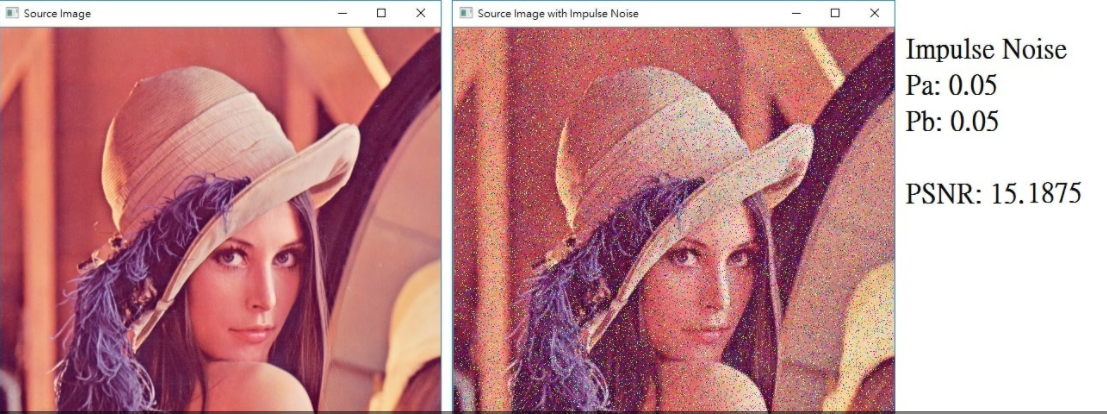
　　　 sum += pow( ( I[i][j] - K[i][j] ) , 2 );

　　}

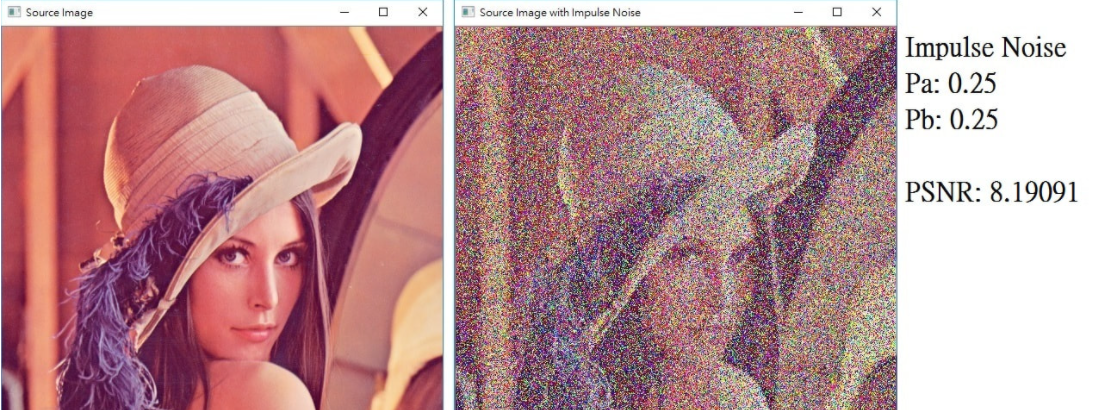
}

MSE = sum / (m\*n);

以例圖來比較一下不同雜訊濃度對其PSNR值各為多少:

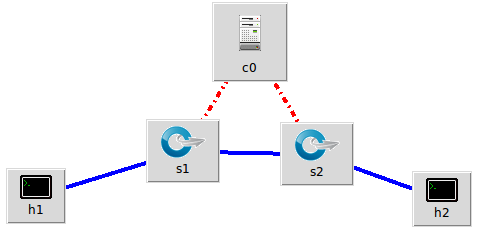
雜訊濃度10%，PSNR值約為15  


雜訊濃度50%，PSNR值約為8



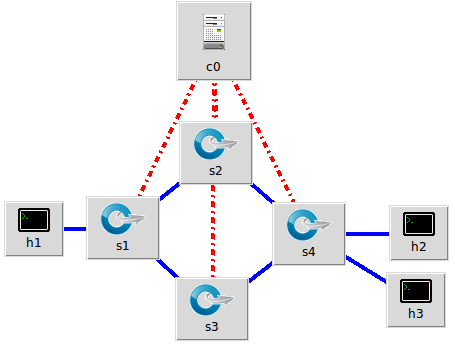
§研究方法與分析:作者創建了一個較為複雜的拓樸，從UDP、TCP傳輸的吞吐量，以及影像傳輸作為單路徑傳輸與多路徑傳輸的效能比較。我簡化了實驗環境中的拓樸，並以傳輸影像來比較單路徑傳輸與多路徑傳輸的差異(可以從肉眼直接觀察及用PSNR值來看出影像品質的好壞)。

單路徑傳輸的拓樸:



設有兩個終端節點h1、h2，並以兩個交換器s1、s2連接通道，一個預設的controller為s1、s2自動建立封包轉發的規則，封包將由h1傳送至h2。

多路徑傳輸拓樸:

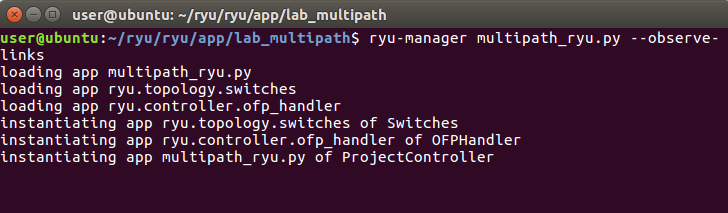


設有三個終端節點h1、h2、h3(h3未使用)，並以四個交換器s1、s2、s3、s4連接通道，一個ryu controller來控制交換器的轉發規則。根據我們對ryu controller編寫的規則，當資料流到達s1時，有70%將會發送給s2，30%發送給s3，接著再將資料送到h2。

**§實驗步驟(多路徑傳輸):**

▼啟動ryu controller，等待拓樸建立

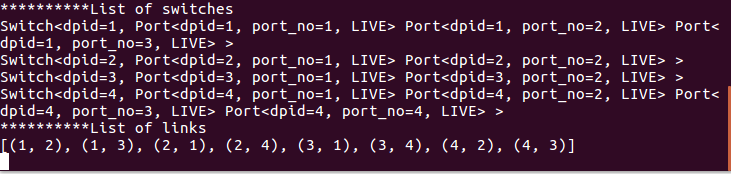
ryu-manager multipath\_ryu.py --observe-links



▼開啟另一個終端機，啟動網路拓樸

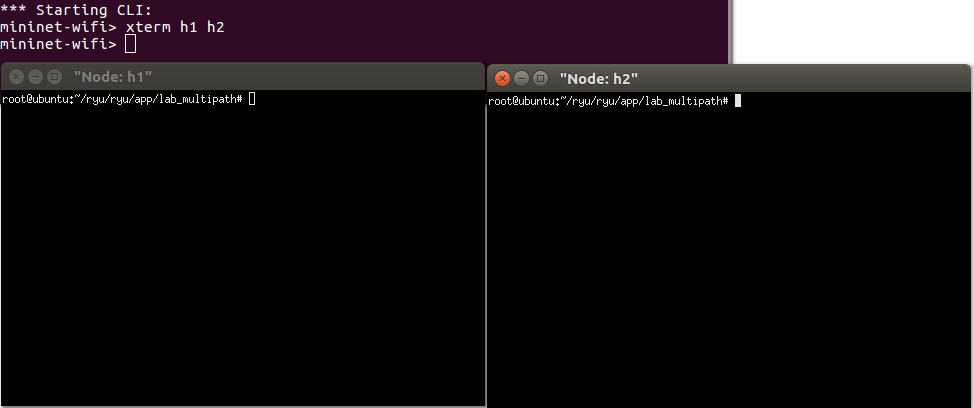
sudo mn --custom multipath\_topo.py --topo mytopo --controller=remote



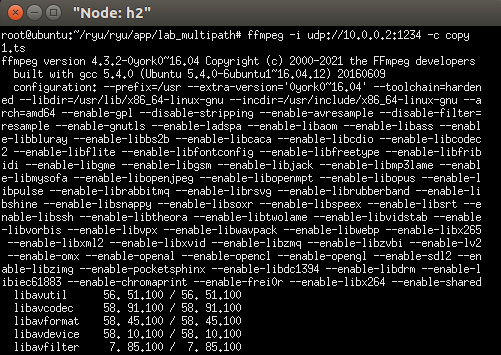
▼此時ryu controller的終端機會顯示交換器及連結訊息  


▼回到網路拓樸的終端機用xterm開啟兩個節點(h1、h2)

xterm h1 h2



▼h2作為接收端，傳輸到h2的影像以.ts儲存，這是一種比較適合傳輸的格式

ffmpeg -i udp://10.0.0.2:1234 -c copy 1.ts  


▼h1作為傳送端，將影片傳送到h2 (h2節點ip為10.0.0.2，port號1234)

ffmpeg -re -i 1.mp4 -c copy -f mpegts udp://10.0.0.2:1234



▼等h2收到後，再還原成.yuv檔，以便接下來用psnr來比較影像的品質

ffmpeg -i 1.ts 1.yuv



▼比較原影像(foreman\_cif.yuv)與節點h2接收到的影像(1.yuv)

若使用指令psnr則終端會列出每一個frame的品質，avgpsnr為psnr的平均值，後方參數代表解析度(resolution)為352\*288 ，YUV為4:2:0

./avgpsnr 352 288 420 foreman\_cif.yuv 1.yuv



▼單路徑傳輸沒有用到ryu controller，僅以預設的controller來控制交換器的封包轉發行為，因此建立好拓樸(singlepath\_topo.py)後進行與上述一樣步驟即可。

得到較低的平均psnr值



補充:

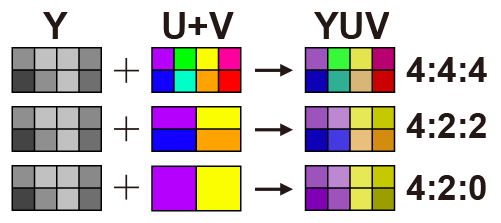
▼yuv檔可透過以下指令壓縮成.mp4檔案

ffmpeg -f rawvideo -s:v 352x288 -r 30 -i foreman\_cif.yuv -c:v libx264 -qp 30 -g 12 -bf 2 -f mpeg 1.mp4  
其中的qp值越高，則壓縮力道越大，即檔案越小。但影像品質也較差。

▼在虛擬機中我是使用ubuntu應用商店裡下載vlc media player來播放視訊。

但是在虛擬機中轉檔出來的mp4檔案我無法在windows播放。

▼註1:有關YUV



.yuv檔案是一種影像格式。YUV是一種顏色編碼方法。「Y」表示明亮度（Luminance、Luma），「U」和「V」則是色度、濃度（Chrominance、Chroma）。

虛擬機整合環境下載(包含mininet、ryu等): <http://csie.nqu.edu.tw/smallko/mininet-wifidockerp4.zip>

參考資料:

https://zh.wikipedia.org/wiki/%E5%B3%B0%E5%80%BC%E4%BF%A1%E5%99%AA%E6%AF%94

https://www.youtube.com/watch?v=tTS4UErEuAY&t=1444s

<https://media.xiph.org/video/derf/>

https://zh.wikipedia.org/wiki/OpenFlow

RYU BOOK

https://jason-chen-1992.weebly.com/home/-peak-single-to-noise-ratio

結論:

對於軟體定義網路的結構與實際運作方式更加了解，尤其是網路底層設備是如何以程式設計來轉發封包的讓我花了相當多的時間去學習，另一方面虛擬機的操作與環境建置也需要習慣，過程中許多錯誤讓我不知道究竟是設計上出錯了還是環境沒有設定好，所以只得簡化實驗來檢驗多路徑傳輸的優勢。當5G技術逐漸普及，SDN與NFV(網路功能虛擬化)又會被拿出來討論，這些東西也許將來會用上也說不定..。