**國立暨南國際大學電機工程學系通訊工程碩士班**

**碩士論文**

**異質網路中基於 OpenFlow 之頻寬聚集路由器實作**

**The Implementation and Analysis of OpenFlow-based Bandwidth Aggregation Routers in Heterogeneous Networks**

**指導教授：許孟烈 博士**

**吳坤熹 博士**

**研究生：賴意姍**

**中華民國104年X月XX日**

**國立暨南國際大學電機工程學系通訊工程碩士班**

**碩士論文**

**異質網路中基於 OpenFlow 之頻寬聚集路由器實作**

**The Implementation and Analysis of OpenFlow-based Bandwidth Aggregation Routers in Heterogeneous Networks**

**指導教授：許孟烈 博士**

**吳坤熹 博士**

**研究生：賴意姍**

**中華民國104年X月XX日**

# 學位考試委員會審定書

# 致謝

論文名稱：異質網路中基於 OpenFlow 之頻寬聚集路由器實作與分析

校院系：國立暨南國際大學科技學院電機工程學系通訊工程碩士班 頁數：

畢業時間：中華民國104年1月 學位別：碩士

研究生：賴意姍 指導教授：許孟烈博士、吳坤熹博士

# 摘要

傳統網路環境中，具有負載平衡功能的網路設備往往放置於廣域網路（Wide Area Network，WAN）和區域網路（Local Area Network，LAN）間，平均分配內部網路對外流量來存取外部服務。網路服務多樣化，網路管理者需要啟動任何一種路由協定演算法，必須對每台路由器（Router），逐一登入命令介面（Command Line Interface，CLI）設定。而 OpenFlow 為軟體定義網路（Software Defined Network，SDN）的網路通訊標準，提供彈性化、模組化的網路定義介面。本論文結合 OpenFlow 技術開發一套在動態路由器網路介面（Interface）上自動切換網路介面功能，並且協助網路管理者透過應用程式介面（Application Programmatic Interfaces，API）來管理。

關鍵詞： 負載平衡、OpenFlow、路由器、軟體定義網路、異質網路

Title of Thesis: The Implementation and Analysis of OpenFlow-based Bandwidth Aggregation Routers in Heterogeneous Networks

Name of Institute: Master Program of Communication Engineering, Department of Electrical Engineering of College of Science and Technology, National Chi Nan University Pages:

Graduation Time: Jan. 2015 Degree Conferred: Master

Student Name: Yi-Shan Lai Advisor Name: Dr. Meng-Lieh Sheu, Dr. Quincy Wu

# Abstract

**Key words**: load balance, OpenFlow, router, Software Defined Network, heterogeneous networks

**目錄**

[學位考試委員會審定書 I](#_Toc408261643)

[致謝 II](#_Toc408261644)

[摘要 III](#_Toc408261645)

[Abstract IV](#_Toc408261646)

[圖目錄 VII](#_Toc408261647)

[表目錄 IX](#_Toc408261648)

[第一章 緒論 1](#_Toc408261649)

[1.1. 研究背景 1](#_Toc408261650)

[1.2. 研究動機 1](#_Toc408261651)

[1.3. 論文架構 2](#_Toc408261652)

[第二章 背景知識及文獻探討 4](#_Toc408261653)

[2.1 外送流量備援容錯機制（Multihoming） 4](#_Toc408261654)

[2.1.1. 外送流量備援容錯機制的特性 5](#_Toc408261655)

[2.2 EIGRP（Enhanced Interior Gateway Routing Protocol） 8](#_Toc408261656)

[2.2.1. Equal Cost Multipath Routing 9](#_Toc408261657)

[2.2.2. Unequal Cost Path Load Balancing 10](#_Toc408261658)

[2.3 相關研究 12](#_Toc408261659)

[2.3.1. 固定指派（Fixed） 12](#_Toc408261660)

[2.3.2. 依比重輪流指派（Round-Robin by Weight） 13](#_Toc408261661)

[2.3.3. 依成本分配（Cost） 14](#_Toc408261662)

[2.3.4. 依連線數指派（By Connections） 15](#_Toc408261663)

[2.3.5. 依流量目前使用量（By Traffic） 16](#_Toc408261664)

[2.4 Software Defined Networking（SDN） 18](#_Toc408261665)

[2.5 OpenFlow 19](#_Toc408261666)

[第三章 系統實作 24](#_Toc408261667)

[3.1 Open vSwitch 安裝與設定 26](#_Toc408261668)

[**3.1.1.** **Open vSwitch安裝** 26](#_Toc408261669)

[**3.1.2.** **Open vSwitch 設定** 28](#_Toc408261670)

[3.2 Ryu（OpenFlow Controller）安裝與設定 31](#_Toc408261671)

[3.2.1. Ryu 安裝 31](#_Toc408261672)

[3.2.2. Ryu 設定 32](#_Toc408261673)

[3.3 SAPIDO 路由器設定 35](#_Toc408261674)

[3.3.1. 韌體更新 35](#_Toc408261675)

[3.3.2. 3G網路卡與 LAN 介面設定 36](#_Toc408261676)

[3.4 本實驗使用主要 OpenFlow 訊息 37](#_Toc408261677)

[3.5 動態負載平衡 38](#_Toc408261678)

[3.5.1. 系統架構 39](#_Toc408261679)

[3.5.2. 使用網路位址轉換機制之動態負載平衡 40](#_Toc408261680)

[3.5.3. 使用 ICMP Redirect 機制之固定位址 48](#_Toc408261681)

[第四章 效能分析與評估 56](#_Toc408261682)

[4.1 實驗環境與方法 56](#_Toc408261685)

[4.2 流量監控工具 56](#_Toc408261686)

[**4.2.1.** **MRTG 安裝** 57](#_Toc408261687)

[**4.2.2.** **MRTG 設定** 57](#_Toc408261688)

[4.3 數據量測與分析 59](#_Toc408261689)

[4.3.1. 鏈路使用流量實驗與結果 59](#_Toc408261690)

[4.3.2. 鏈路連線時間實驗與結果 60](#_Toc408261691)

[第五章 結論與未來展望 62](#_Toc408261692)

[參考文獻 63](#_Toc408261693)

# 圖目錄

[圖 1、使用 Multihoming 技術連接在同區域 4](#_Toc408261694)

[圖 2、使用 Multihoming 技術連接不同區域 5](#_Toc408261695)

[圖 3、冗餘備援（Redundant Link） 6](#_Toc408261696)

[圖 4、負載平衡（Load Balance） 6](#_Toc408261697)

[圖 5、策略路由（Policy Routing） 7](#_Toc408261698)

[圖 6、頻寬聚集（Bandwidth Aggregation） 7](#_Toc408261699)

[圖 7、距離向量 9](#_Toc408261700)

[圖 8、等價多路徑路由 (Equal Cost Multipath Routing) 10](#_Toc408261701)

[圖 9、不對等路徑度量負載平衡（Unequal Cost Path Load Balancing） 11](#_Toc408261702)

[圖 10、固定指派（Fixed） 13](#_Toc408261703)

[圖 11、比重輪流指派（Round-Robin by Weight） 14](#_Toc408261704)

[圖 12、成本分配（Cost） 15](#_Toc408261705)

[圖 13、連線數指派（By Connections） 16](#_Toc408261706)

[圖 14、流量目前使用量（By Traffic） 18](#_Toc408261707)

[圖 15、軟體定義網路主要概念 19](#_Toc408261708)

[圖 16、OpenFlow 網路架構 20](#_Toc408261709)

[圖 17、網路拓撲架構 25](#_Toc408261710)

[圖 18、Open vSwitch 成功安裝與設定的顯示資訊 R1（OpenFlow Switch） 30](#_Toc408261711)

[圖 19、Ryu（OpenFlow Controller）執行畫面 32](#_Toc408261712)

[圖 20、SAPIDO 韌體 36](#_Toc408261713)

[圖 21、R1（OpenFlow Switch） 36](#_Toc408261714)

[圖 22、R1（OpenFlow Switch）eth0 LAN 介面設定 37](#_Toc408261715)

[圖 23、R1（OpenFlow Switch） eth2 LAN 介面設定 37](#_Toc408261716)

[圖 24、系統架構 39](#_Toc408261717)

[圖 25、NAT-步驟1 41](#_Toc408261718)

[圖 26、NAT-步驟2 42](#_Toc408261719)

[圖 27、NAT-步驟3 43](#_Toc408261720)

[圖 28、NAT-步驟4 44](#_Toc408261721)

[圖 29、NAT-步驟5 45](#_Toc408261722)

[圖 30、NAT-步驟6 46](#_Toc408261723)

[圖 31、R1（OpenFlow Switch）load balance演算法 47](#_Toc408261724)

[圖 32、ICMP-步驟1 49](#_Toc408261725)

[圖 33、ICMP-步驟2 50](#_Toc408261726)

[圖 34、ICMP-步驟3 51](#_Toc408261727)

[圖 35、ICMP-步驟4 52](#_Toc408261728)

[圖 36、ICMP-步驟5 53](#_Toc408261729)

[圖 37、ICMP-步驟6 54](#_Toc408261730)

[圖 38、R1（OpenFlow Switch）ICMP Redirect演算法 55](#_Toc408261731)

[圖 39、實驗環境 56](#_Toc408261732)

[圖 40、使用 scp 指令 59](#_Toc408261733)

[圖 41、accumulated traffic 鏈路結果 60](#_Toc408261734)

[圖 42、使用SSH指令 60](#_Toc408261735)

[圖 43、SSH 連線時間 61](#_Toc408261736)

# 表目錄

[表 1、負載平衡設備價格 2](#_Toc408261737)

[表 2、EIGRP Topology 11](#_Toc408261738)

[表 3、常見的OpenFlow 訊息表 21](#_Toc408261739)

[表 4、部分OpenFlow 1.3可比對之封包欄位 22](#_Toc408261740)

[表 5、系統詳細規格表 25](#_Toc408261741)

[表 6、Open vSwitch 28](#_Toc408261742)

[表 7、Ryu（OpenFlow Controller）app 範例檔（simple\_switch\_13.py） 32](#_Toc408261743)

[表 8、此實作使用的OpenFlow 訊息 37](#_Toc408261744)

[表 9、NAT-步驟1 41](#_Toc408261745)

[表 10、NAT-步驟2 42](#_Toc408261746)

[表 11、NAT-步驟3 43](#_Toc408261747)

[表 12、NAT-步驟4 44](#_Toc408261748)

[表 13、NAT-步驟5 45](#_Toc408261749)

[表 14、NAT-步驟6 46](#_Toc408261750)

[表 15、利用 NAT 機制達成負載平衡之 Flow Table R1（OpenFlow Switch） 47](#_Toc408261751)

[表 16、ICMP-步驟1 49](#_Toc408261752)

[表 17、ICMP-步驟2 50](#_Toc408261753)

[表 18、ICMP-步驟3 51](#_Toc408261754)

[表 19、ICMP-步驟4 52](#_Toc408261755)

[表 20、ICMP-步驟5 53](#_Toc408261756)

[表 21、ICMP-步驟6 54](#_Toc408261757)

[表 22、利用 ICMP Redirect 機制之 Flow Table R1（OpenFlow Switch） 54](#_Toc408261758)

# 緒論

## 研究背景

隨著網路應用多樣化，在中小型企業網路環境中，網路頻寬需求量逐漸增加，已經不再使用單一 ADSL 鏈路，而是會使用多條鏈路 ADSL 來建構高頻寬網路給客戶端（Client）。在多條鏈路的情況下，一般可接多條鏈路的網路設備，如：一般在市面上常看到 D-Link Switch [1]，不一定會有鏈路負載平衡功能，而且這讓中小型企業並不能徹底使用到每條鏈路。

負載平衡設備是一種經常用來解決鏈路負載不平衡的方法。透過負載平衡功能，讓內部網路的對外流量，平均分配在各個鏈路，讓中小型企業完整的使用到各個鏈路。因為負載平衡的技術有很多種，所以要選擇一種適合的自家網路架構仍然是一項挑戰。

## 研究動機

現今的社會中無論到何處都會使用網路進行收送資料，非常需要一個網路設備找尋最佳化的路徑（Router）。

當路由器有多種選擇可到達目的地（Destination）網路路徑，並且這些路徑的度量值（metric value），如：跳數、頻寬相等，即所謂的等成本度量，稱為等價多路徑路由（Equal Cost Multipath Routing，ECMP）[2]。等價多路徑路由可設定使用靜態路由（static route）和動態路由（dynamic route）。而當發現這些路徑的度量值，如：跳數、頻寬不相等，如果路由器具備有對應功能的話，那麼仍然能夠透過多個網路介面（Interface）發送封包到相同的目的地，統一稱為不對等路徑度量負載平衡（Unequal Cost Path Load Balancing）[3]。

負載平衡設備目前有以下這些公司：Xtera [4]、路由器具有不對等路徑度量負載平衡設備的公司，只有思科推出的路由器產品才有這項功能。對於一個中小型企業來說，如果需要用此負載平衡設備就需要向公司購買，這是一個非常相當大的設備成本，並非每個中小型企業都能負荷，如所示。而網路管理者需要啟動任何一種路由協定演算法，必須對每台路由器（Router），逐一登入命令介面（Command Line Interface，CLI）設定。

表 1、負載平衡設備價格

|  |  |  |
| --- | --- | --- |
| 產品名稱 | 功能 | 價格 |
| Ascenlink 702 [5] | 固定（Fixed）、  輪流（Round Robin）、  下載流量（By Downstream Traffic）、  上載流量（By Upstream Traffic）、  最佳路徑（By Optimum Route）。 | 約10萬 |
| Cisco 2811 [6] | IOS 升級（12.3）後改為增強型內部閘道路由協定（Enhanced Interior Gateway Routing Protocol，EIGRP）、  無線服務（Wireless）、  安全防禦（Security）、  語音服務（Voice）、  視訊服務（Video）。 | 約10萬 |
| Cisco 2901 [7] | IOS 升級（12.3）後改為增強型內部閘道路由協定（Enhanced Interior Gateway Routing Protocol，EIGRP）、  應用服務（Application）、  WAN 連接（WAN Connectivity）、  安全防禦（Security）。 | 約10萬 |

本研究使用 OpenFlow 技術來提供一個具有動態負載平衡路由器設備功能來實驗是可行的，並且協助網路管理者透過應用程式介面（Application Programmatic Interfaces，API）來管理。

## 論文架構

本論文共分為五個章節。第一章主要是描述本論文的研究背景和動機；第二章將介紹相關 OpenFlow 架構與背景知識，主要包含的是外送流量備援容錯機制技術（Multihoming）、增強型內部閘道路由協定（Enhanced Interior Gateway Routing Protocol，EIGRP）技術；第三章為本論文重點，會依序說明系統的架構，各種功能的運作原理和溝通的格式，最後會展示實作的成果；第四章為效能量測；第五章本論文之總結與未來展望。

# 背景知識及文獻探討

## 外送流量備援容錯機制（Multihoming）

外送流量備援容錯機制（Multihoming）是網際網路連線的一種容錯機制，一般使用在客戶端（Client），以多條對外鏈路來傳輸資料流量。其中一條鏈路中斷時，自動切換到其他鏈路上，以提供較佳之可靠度。

外送流量備援容錯機制主要運作網路環境在於多個邊界路由器（Border Router，BR）同時具有多個 IP 位址連接到網際網路（Internet）。原理是由「兩個或多個鏈路連接在同個區域（area）」或「兩個或多個以上的鏈路連接到不同的區域（area）」，如、所示[8]。

兩個或多個鏈路連接在同個區域或不同區域，假設當邊界路由器機器故障 （BR-1），而還有另外一個邊界路由器（BR-2）取而代之，不會讓 A area 整個對外網路中斷。

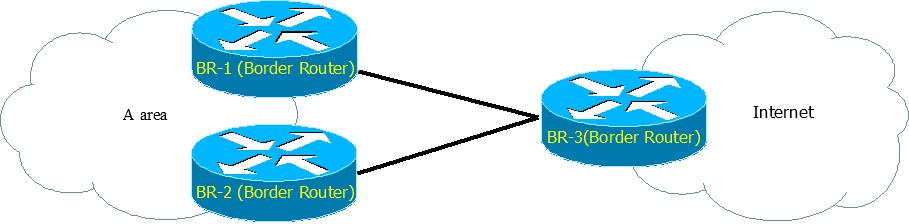


圖 1、使用 Multihoming 技術連接在同區域

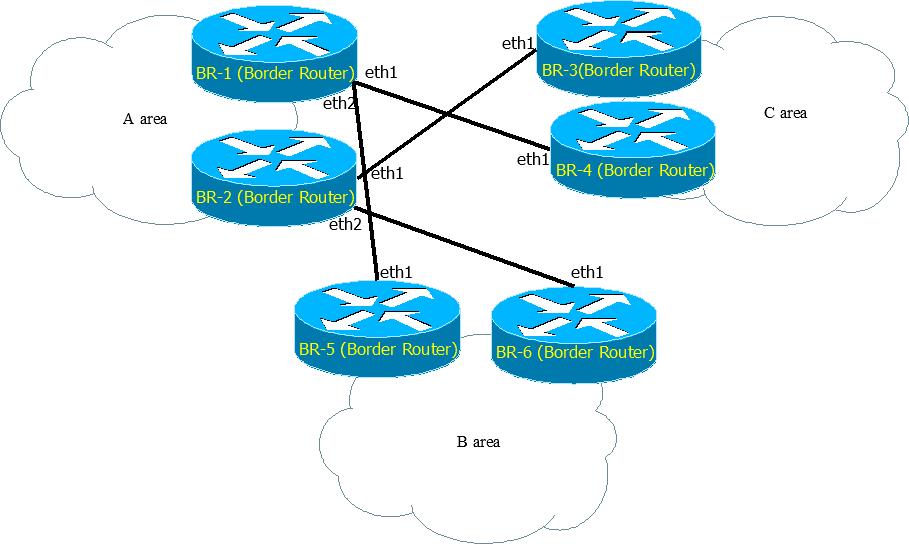


圖 2、使用 Multihoming 技術連接不同區域

### 外送流量備援容錯機制的特性

外送流量備援容錯機制的特性分為「冗餘備援（Redundant Link）」和「負載平衡（Load Balance）」和「策略路由（Policy Routing）」和「頻寬聚集（Bandwidth Aggregation）」 [8] [9]。

冗餘備援（Redundant Link）：當邊界路由器主要連線突然斷線時（eth1），可透過從其他網路介面建立連線，避免因網路斷線而造成損失。如所示，當 Client 1 對外存取資料時，往預設默認（default）邊界路由器傳送封包，由於鏈路 eth1 斷線時，經過邊界路由器偵測後，會改由鏈路 eth2 連線至網際網路上。

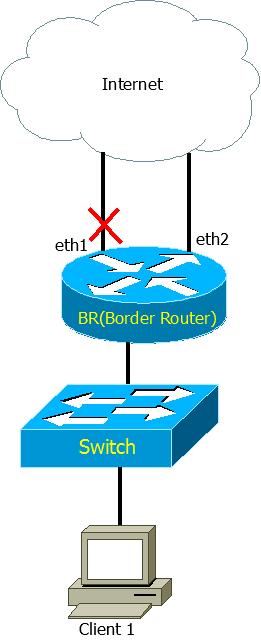


圖 3、冗餘備援（Redundant Link）

負載平衡（Load Balance）：先進行封包傳送，後偵測網路鏈路狀況，並且自動選擇符合頻寬需求的網路介面建立連線（Router-based multi-homing），避免單一網路傳輸過量負擔。如所示， Client 1 對外存取資料時，當邊界路由器偵測到原先鏈路 eth1 負載滿載時（紅色線），會選擇負載最少鏈路 eth2 連線（黑色線）至網際網路上。因此，這樣可提高網路可存取性與避免網路壅塞導致延遲，並且達到負載平衡目的。

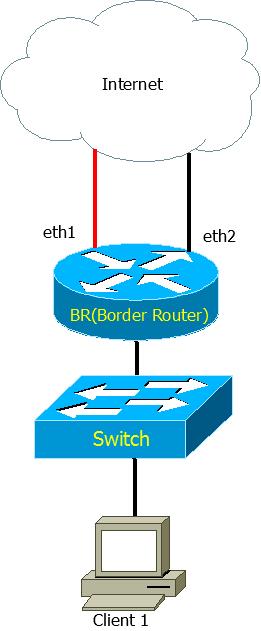


圖 4、負載平衡（Load Balance）

策略路由（Policy Routing）：網路應用服務類型有許多種類或者依照不同policy 檢查順序是由 route map 參數值來決定，參數低的優先權較高等，而會有不同的頻寬需求，因此可針對不同的鏈路性質而引導符合需求的網路連結 （network consolidated）。如所示，當 Client 1 存取資料時，往邊界路由器傳送封包，鏈路（eth2）具有較高的頻寬（HiNet），適用於超文字傳輸協定 （HyperText Transfer Protocol，HTTP）來存取國內或國外資料庫網站，然而鏈路 （eth1）具有較低的頻寬（TANet），適用於檔案傳輸協定（File Transfer Protocol，FTP）來存取學術界資料。

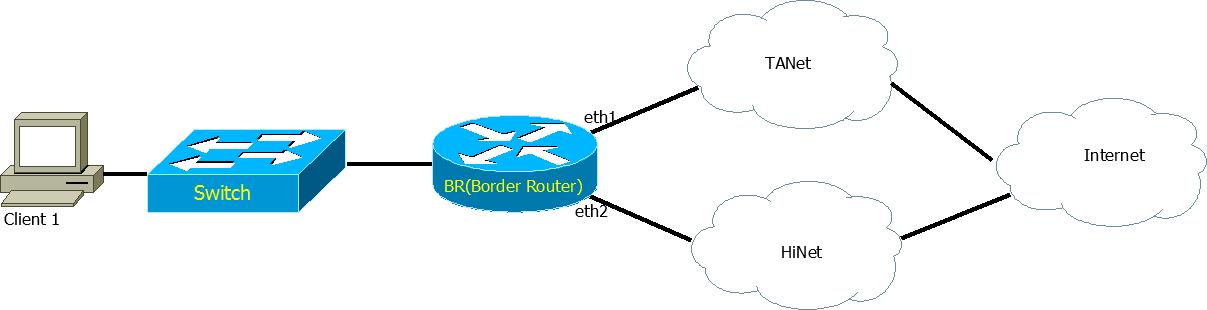


圖 5、策略路由（Policy Routing）

頻寬聚集（Bandwidth Aggregation）：將所有鏈路合併成一條較大的頻寬使用。將原本三條512Kbps 的實體鏈路聚集成一條高速的邏輯鏈路。如所示，當 Client 1 存取資料時，經過邊界路由器最大約可達 T1 速度（1.544Mbps）。

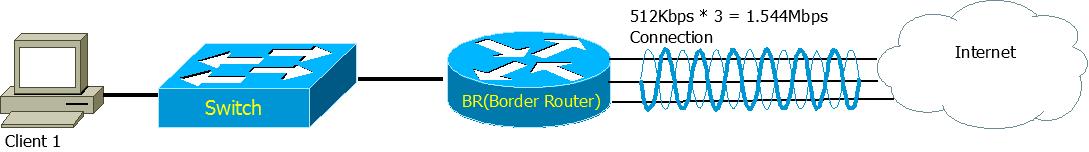


圖 6、頻寬聚集（Bandwidth Aggregation）

## EIGRP（Enhanced Interior Gateway Routing Protocol）

在1985年 [9] 由思科公司開發出內部閘道路由協定（Interior Gateway Routing Protocol，IGRP），其目的是發展出一個可以在同個自治系統 （Autonomous System，AS）區域中運作的路由協定，透過用戶配置，如延遲、頻寬、可靠性和負載等路由器進行的路由管理。內部閘道路由協定解決了路由資訊協定（Route Information Protocol，RIP）的某些限制，如使用跳數（Hop）量最大為15跳。而自從思科 IOS 升級（12.3）後，內部閘道路由協定不再支援，改為增強型內部閘道路由協定（Enhanced Interior Gateway Routing Protocol，EIGRP）。

在1992年 [3] 由思科開發出增強型內部閘道路由協定是一種距離向量路由協定（Distance Vector Routing Protocol）。顧名思義，距離向量意味著用距離和向量通告路由（advertise route）。距離表示為跳數（Hop），而向量則是下一跳 （Next Hop）路由器或出口網路介面。

使用距離向量路由協定的路由器並不了解到達目的地網路的整條路徑。只知道應該往哪一個方向或使用哪一個網路介面轉送封包、本身與目的地網路之間的距離。

例子：有三台 Router 分別為 Router A、Router B、Router C、，Router A 知道到達網路 192.168.2.0/24 的距離是 1 跳數，方向是從網路介面 eth1 到 Router B，如所示。



圖 7、距離向量

增強型內部閘道路由協定是思科內部閘道路由協定增強版。它們都是思科的專有通訊協定，只能在思科路由上執行。

### Equal Cost Multipath Routing

等價多路徑路由（Equal Cost Multipath Routing，ECMP）[2]，適用於在不同的路徑到達目的地的網路具有相同路由度量值（metric value），如所示。如果當有兩條鏈路或多條以上鏈路時，流量將在多個等價路徑中分配。流量在等價多路徑中的分配方式為流（flow），根據封包表頭中的來源端、目的地位址計算散列（Hash）找到往目的地的下一跳（Next Hop）。

以下所示為例子，基於流來做散列方法有三種方式 [2] ：1. Modulo-N 方式：將流做散列後，假設當流的散列值為6，對N條鏈路取餘數，N 為可到達目的地的鏈路數值，換句話說，假設 E 要通往 Internet 有三種不同鏈路 B、C、D，B 鏈路的值為0、C 鏈路的值為1、D 鏈路的值為2，當得知散列值（6/3）得餘數值為0，所以 E 該往 B 鏈路。使用此方法好處簡單，缺點則當有鏈路新增或減少時，需要改變鏈路的流，因每一個流都直接對應到一個鏈路上。2. hash-threshold 方式：能改善 Modulo-N 缺點，每條鏈路不再是一個值，而是一個範圍，判斷哪些流該如何走哪些鏈路，只需看流散列值落在哪些範圍內，換句話說，假設 E 要通往 Internet 有三種不同鏈路 B、C、D，B 鏈路的範圍值為[0,3]、C 鏈路的範圍值為[4,7]、D 鏈路的範圍值為[8,10]，當得知散列值6，所以 E 該往 C 鏈路。使用此方法好處當有路徑新增或減少時，只需要改變鏈路流數量。3. Highest random weight（HRW）方式：將流使用散列和加上鏈路的下一跳 IP 位址，對 N 條不同鏈路計算不同的散列值，且在這些值中選出最大的，假設 E 要通往 Internet 有三種不同鏈路 B、C、D，B 鏈路的下一跳為 IPB、C 鏈路的下一跳為 IPC、D 鏈路的下一跳為 IPD，因此我們需要分別算出不同的散列值，hash（flow, IPB）=HVB=2、hash（flow, IPC）=HVC=4、hash（flow, IPD）=HVD=6，所以 E 該往 D 鏈路。使用此方法好處當有鏈路新增或減少時，可減少流改變，缺點則如果散列值算出來，具有相同的流，例如：算出的散列值為6，剛好有兩條鏈路具有相同值，需要計算多次才更能決定鏈路。

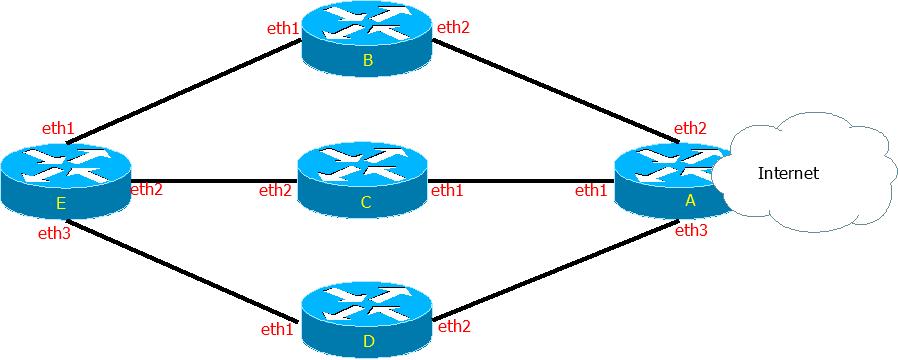


圖 8、等價多路徑路由 (Equal Cost Multipath Routing)

### Unequal Cost Path Load Balancing

不對等路徑度量負載平衡（Unequal Cost Path Load Balancing）[12]，適用於在不同的路徑到達目的地的網路具有不同路由度量值（metric value），如所示。如果當有兩條鏈路或多條以上鏈路時，流量將在多個不對等價路徑中平均分配。

以下所示為例子、所示，假設 A 要通往 Internet 有三種路徑方式可以進入 Internet：1. A（eth1）-B 路徑:30 metric。2. A（eth2）-B 路徑:20 metric。3. A（eth3）-B 路徑:145 metric，以下一跳（Next Hop）A IP 位址:192.168.1.2的路徑為例，括號中（435200/409600）前面代表可行距離（Feasible Distance，FD）和通告距離（Advertised Distance，AD），因此變異值（Variance）計算步驟如下：

1. 找出所有路徑中最低可行距離：在中可以得知為435200。
2. 檢查是否有任何路徑的通告距離高於最低的可行距離：如果有某一條路徑的通告距離值高於步驟1的最低可行距離值，則路徑無法參與負載平衡（load balance），直接刪除 A（eth3）。
3. 在剩下路徑中找出最高的可行距離：在中可以得知為2809856。
4. 將步驟3的最高可行距離值除以步驟1最低可行距離值，再以無條件進入法所得的整數即為變異值：因此將（2809856/435200），得到值6.456~，無條件進入法所得變異值為7。

將上述步驟得到的變異值設定在 EIGRP 的Routing Process中，此有兩條鏈路「A（eth1）、A（eth2）」是滿足可行條件（Feasibility Condition，FC）路由。

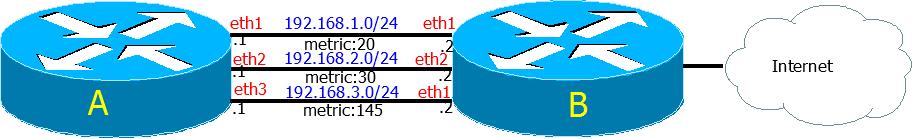


圖 9、不對等路徑度量負載平衡（Unequal Cost Path Load Balancing）

表 2、EIGRP Topology

|  |
| --- |
| A# show ip eigrp topology  IP-EIGRP Topology Table for AS (100) / ID (20.0.0.1)  Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,  r - Reply status  P 163.22.0.0/16, 1 successors, FD is 435200  via 192.168.1.2 (435200/409600), Ethernet 1  via 192.168.2.2 (2809856/409600), Ethernet 2  via 192.168.3.2 (2323456/2297856), Ethernet 3 |

## 相關研究

在這小節中，我們會探討關於利用多個網路介面達成負載平衡設備常見的對外連線分配演算法的相關研究。這些研究中，主要分為固定指派（Fixed）、依比重輪流指派（Round-Robin by Weight）、依連線數指派（By Connections）、依成本分配（Cost） 、依流量目前使用量（By Traffic），我們將在各小節中介紹之。

### 2.3.1. 固定指派（Fixed）

選擇固定的廣域網路鏈路，當有封包往對外連線產生時，強制走某一條選定的對外鏈路 [5]。

例子：有一台 load balancing 設備、兩台 Router 設備、一台 Switch 設備，load balancing 設定好使用 IP 固定模式為遇到來源端 IP 位址（Source IP Address）:172.16.1.1 封包，往 Router 1 傳送，其餘的來源端 IP 位址（Source IP Address)封包，往 Router 2 傳送，如所示。

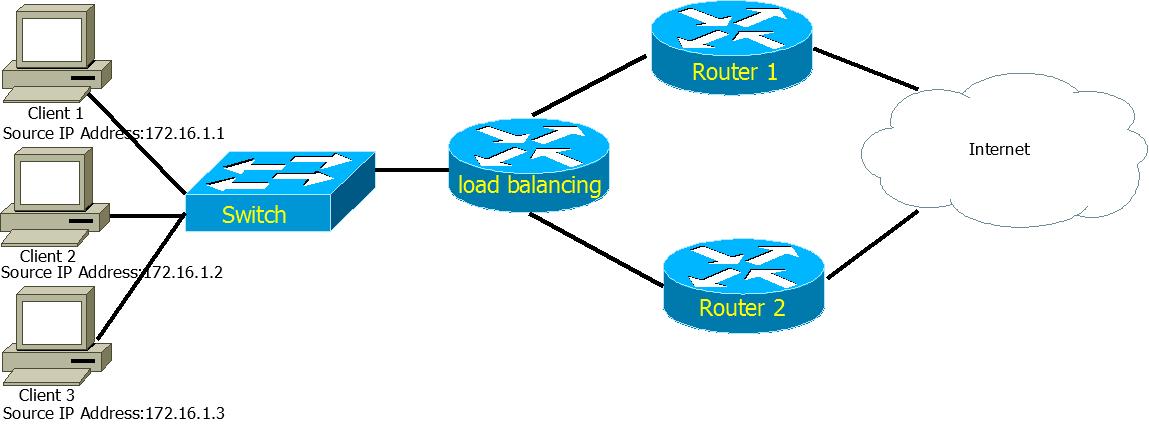


圖 10、固定指派（Fixed）

### 2.3.2. 依比重輪流指派（Round-Robin by Weight）

依照設定每一條鏈路的比重（Weight），當有封包往對外連線產生時，鏈路的比重輪流分配對外的路徑 [13]。

例子：有一台 load balancing 設備、兩台 Router 設備、一台 Switch 設備，load balancing 設定往 Router 1 的比重（weight）為7、往 Router 2 的比重 （weight）為5。當 Client 1要對外存取資料時，封包經過 load balancing ，將封包依照設定好的比重（weight）進行傳送至目的地（Destination），如所示。

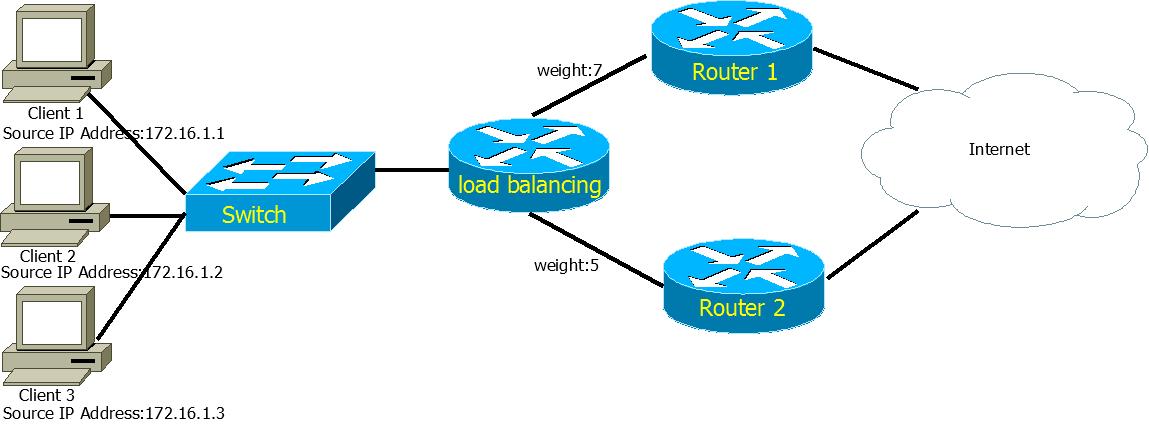


圖 11、比重輪流指派（Round-Robin by Weight）

### 2.3.3. 依成本分配（Cost）

依照負載平衡（Router）設備來動態監控每一條鏈路的成本，當客戶端 （Client）封包往對外連線產生時，將封包分配到最少的成本鏈路。

在 [14] 的研究中，作者提到在同個區域的自治系統中（Autonomous System，AS），有兩個邊界路由器（Border Router，BR）或兩個以上的網路介面（interface）時，連接到多個網際網路服務提供者（Internet Service Provider，ISP）至網際網路（Internet）中，如何使用負載平衡平均分配在各種邊界路由器鏈路，考慮到的問題是：使用默認路由（default route）的方式來達成負載平衡的話，並不是一個能真正達到最佳化負載平衡，於是作者提出了貪婪（Greedy）算法使用動態方式，貪婪算法是只看眼前的路徑誰的鏈路成本低就往哪一條路徑走。

例子：每台 Router 設備當接收到封包會動態判斷到達目的地鏈路誰的路徑成本最低，成本代表為鏈路值。 Client 1 對外存取資料時，封包經過 Router 1，偵測到達目的端路徑 eth1（藍色線）成本最低，進行通訊，如所示。

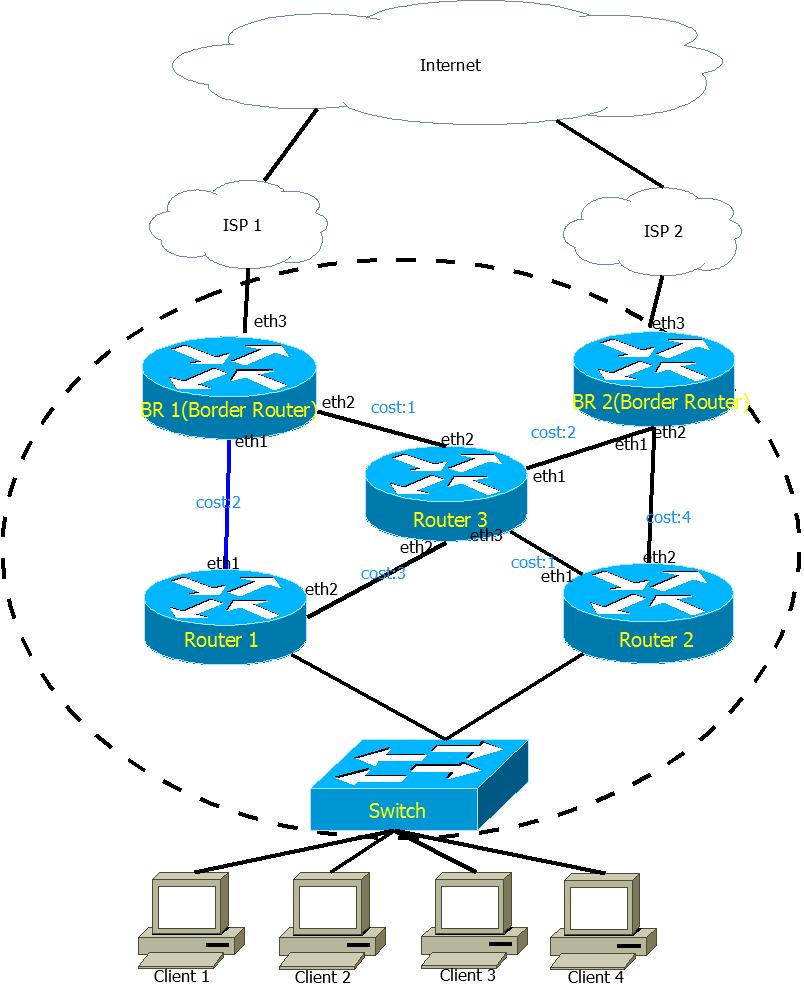


圖 12、成本分配（Cost）

### 2.3.4. 依連線數指派（By Connections）

依照負載平衡（Router）設備設定每一條鏈路的連線數比例，當客戶端（Client）封包對外連線產生時，將封包分配到預先設定好的比例進行連線。

在 [15] 的研究中，提到 [14] 的研究中，如果一直使用貪婪（Greedy）算法就有可能會一直在單一鏈路上傳輸過量負擔，於是作者 [15] 提出了使用貪婪加上連線數演算法。使用步驟如下：1. 先使用貪婪演算法方式選擇計算鏈路的方式。2. 得出結果後，判斷使用的鏈路是否過度重複使用鏈路，如果是就重新分配。3. 並且加上使用鏈路最少的連線者優先分配，最後算出的結果為新的鏈路最佳路徑。

例子：每台 Router 設備會先判斷自己到達目的地鏈路誰的路徑成本最低並且符合鏈路沒過度的使用。 Client 1 對外存取資料時，封包經過 Router 1，發現原本的路徑 eth1（紅色線）成本低並且過度的使用，而只好強制再重新分配路徑，改用 eth2（藍色線）繼續傳送封包，如所示。

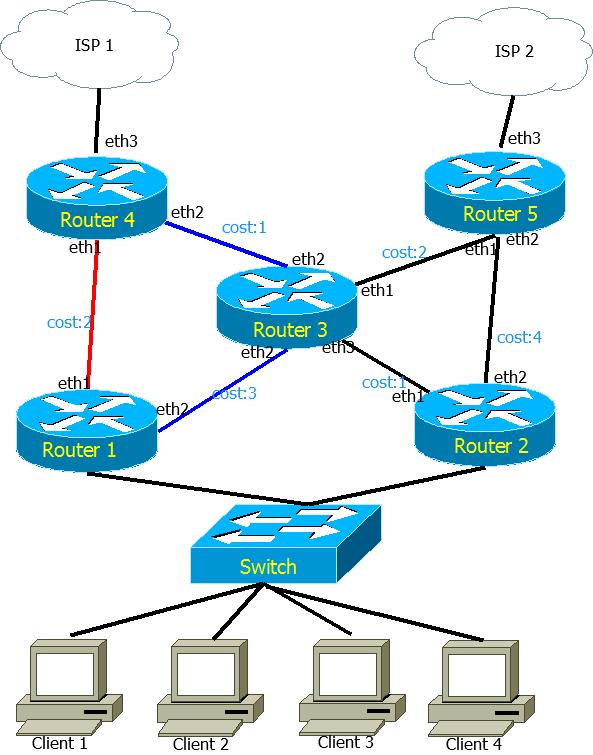


圖 13、連線數指派（By Connections）

### 2.3.5. 依流量目前使用量（By Traffic）

依照負載平衡（Switch）設備來動態監控每一條鏈路的流量使用狀況，當客戶端（Client）封包往對外連線產生時，將封包分配到流量最少的鏈路，以取得最佳的流量使用效率。

在 [16] 的研究中，作者介紹傳統 3\*3 Switch 架構中的方法來運用網路鏈路負載，級別1（level 1）為核心層（core layer）：以階層式設計，負責所有分佈層設備發送的流量，能夠快速轉送封包至目的端。級別2（level 2）為分佈層 （distribution layer）：負責存取層發送的封包，再將封包傳輸到核心層，最後發送到目的端。級別3（level 3）為存取層（access layer）：負責連接終端設備，如：電腦、印表機。主要目的是提供一種將設備連接到網路並控制允許網路上哪些設備進行通訊的方式。

例子：有2台 Switch（Switch 1、Switch 2）定義存取層負責連接終端設備；有2台 Switch（Switch 3、Switch 4）定義分佈層；有1台 Switch（Switch 5）定義核心層彙聚所有分佈層設備。當 Client 1 和 Client 6 之間進行網路視訊通話， Client 1要到 Client 6有3種路徑選擇，現在 Client 1 發送封包給上層 Switch 1（有兩條路徑），然後 Switch 1 設備會動態判斷目前鏈路流量誰為最忙碌（busy）、誰最為輕（lightest），必須把每條鏈路都算出來，再進行通訊，如圖 14所示。

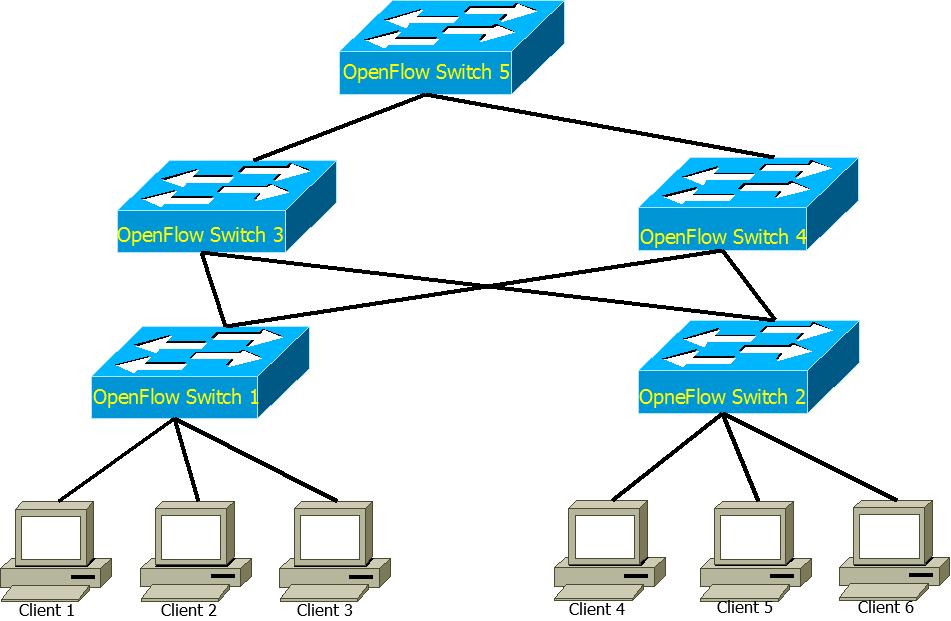


圖 14、流量目前使用量（By Traffic）

## Software Defined Networking（SDN）

傳統 Transmission Control Protocol / Internet Protocol（TCP/IP）通訊協定下，為了呈現各種網路協定，發送傳輸路徑（Data Path）由硬體路由器（Router）決定，所以每一層路由器都要花時間來演算封包路徑，在這樣情況下無法有效使用網路頻寬；而軟體定義網路（Software Defined Network，SDN）[17] 則是由控制層（control plane）的軟體來演算最佳封包傳送路徑，資料層（data plane）的設備完全聽命指令來傳送資料。SDN，是一種新型的網路概念架構，其目的為了使資料傳輸更加靈活、易於操作與管理，且達到降低成本、高相容性、能夠更好地應對應用程式發展和網路條件不斷變化的需求。

SDN 把網路設備的控制層（control plane）從資料層（data plane）中分離出來，以軟體方式實作，如所示。這個架構可以讓網路管理人員，在不更動硬體裝置的前提下，以中央控制方式，用程式重新規劃網路，為控制網路流量提供了新的方法，也提供了核心網路及應用創新的良好平台。

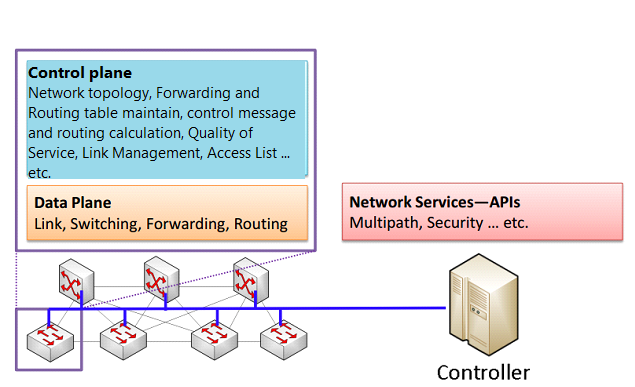


圖 15、軟體定義網路主要概念

## OpenFlow

OpenFlow [18] 是由美國史丹佛大學 Clean State 於2008 年所提出，此網路架構具體實現網路軟體定義 SDN 架構技術， OpenFlow 網路架構如所示，用於建立控制器與資料層之間的溝通主要包含三個部份：1. 決定網路封包流向的控制器（OpenFlow Controller）、2. 接受 OpenFlow Controller 控制並轉送封包的 OpenFlow 交換器（OpenFlow Switch）、3. 位於 OpenFlow Switch 中的 Flow Table 以及傳送控制訊號的 OpenFlow 通訊協定（OpenFlow protocol）。

OpenFlow Controller 與 OpenFlow Switch 硬體之間需先建立安全通道（Secure Channel）連接，雙方之間以 Secure Session Layer（SSL）加密，確保傳送之間的安全。網路管理者可藉由 OpenFlow protocol 所提供的開放原始碼與交換器硬體溝通，設定 OpenFlow Flow Table，由 OpenFlow Flow Table 定義封包的傳送路徑。

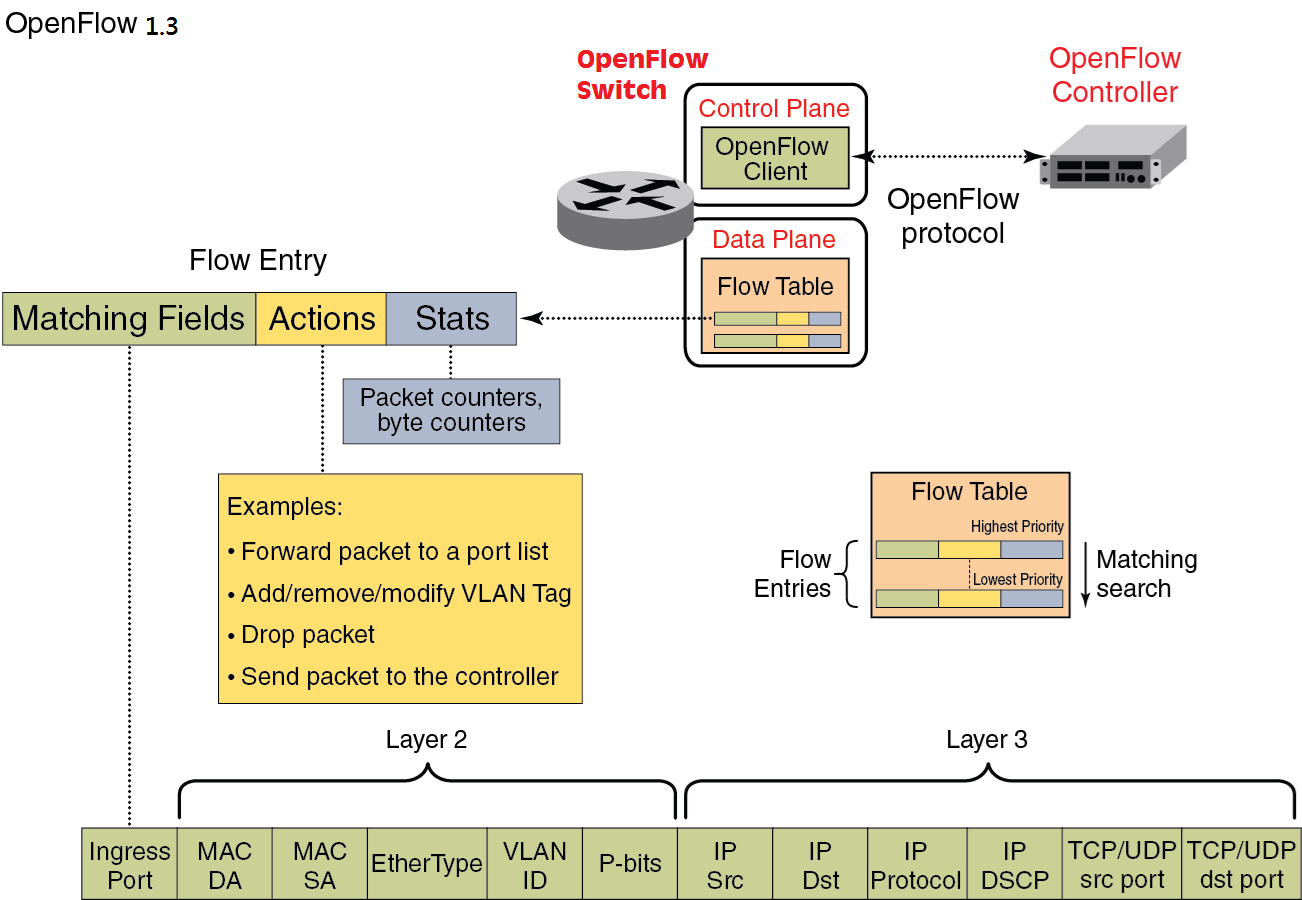


圖 16、OpenFlow 網路架構

OpenFlow Controller 可藉由 OpenFlow protocol 修改 OpenFlow Switch 內的Flow Table ，在Flow Table 中有三個欄位：Match、Action 以及Stats ；當OpenFlow Switch 收到一個封包以後， OpenFlow Switch 會將封包的標頭與Match 欄位進行比對，如果有Flow Entry 被比對到，則 OpenFlow Switch 根據Action 欄位的內容決定要如何處理封包並且更新Stats 欄位； OpenFlow Controller 也可以透過 OpenFlow protocol 對 OpenFlow Switch 詢問 Stats 欄位，以蒐集 OpenFlow網路狀況，藉由上述機制， OpenFlow Controller 能夠以動態、彈性的方式調整網路架構及網路流量。

OpenFlow 除了提供豐富的欄位比對外，也提供了標準化的控制介面，OpenFlow protocol 主要跑在TCP/IP 上，網路管理者能選擇是否需要透過 Secure Session Layer（SSL）加密，增加傳輸的安全性， OpenFlow protocol 主要功能就是傳遞 OpenFlow Controller 與 OpenFlow Switch 間的控制訊息，其控制訊息主要分成三種： Controller-to-Switch、Asynchronous 以及 Symmetric。

Controller-to-Switch 訊息通常是由 OpenFlow Controller 發起， OpenFlow Controller 可藉由此類訊息，修改 OpenFlow Switch 的設定或是 Flow Table ；Asynchronous 大多是由 OpenFlow Switch 先送出的，此類訊息通常會包含錯誤訊息、 Switch 狀態變更等訊息； Symmetric 則是讓 OpenFlow Controller 與 OpenFlow Switches 在固定時間內彼此交換訊息，以確定雙方的連線仍然有效，常見的 OpenFlow 控制訊息如。

表 3、常見的OpenFlow 訊息表

|  |  |
| --- | --- |
| **Controller-to-Switch** | |
| Features | OpenFlow Controller 一開始與 OpenFlow Switch 建立連線時，會交換如支援版本、Spanning Tree protocol（STP ）啟用、 Flow Table 內容等資訊。 |
| Modify-State | 此訊息主要功能就是讓 OpenFlow Controller 能夠新增、修改或刪除 OpenFlow Table 內的 Flow Entries。 |
| Packet-out | OpenFlow Controller 將封包藉由此訊息轉送至OpenFlow Switch ，並且讓 OpenFlow Switch 依照訊息內的 Action 欄位處理封包的去向。 |
| **Asynchronous** | |
| Packet-in | OpenFlow Switch 會將收到的封包包上 OpenFlow 的標頭，並轉送給 OpenFlow Controller 。此封包只有在 Action 欄位被設定成轉送封包給 OpenFlow Controller 或是封包比對 Flow Entry 時沒有成功比對到才會產生。 |
| **Symmetric** | |
| Hello | 當OpenFlow Switch 被設定要連線的OpenFlow Controller 資訊但還沒有與之建立連線時，隔一段時間就送出Hello 訊息給 OpenFlow Controller ，此訊息為建立連線的觸發訊息，OpenFlow Controller 收到此訊息後才會開始發出 Feature 訊息。 |
| Echo/Reply | 此訊息會讓 OpenFlow Controller 與OpenFlow Switch 在固定時間內彼此交換訊息，以確定雙方的連線仍然有效。 |

在 OpenFlow 1.0 [19] 版僅提供一個 Flow Table 供封包查詢及網際網路通訊協定第四版（Internet Protocol version 4，IPv4）封包的比對，到了 OpenFlow 1.1 [20] 開始支援多種 flow table 比對，而 OpenFlow 1.2 [21]加入了網際網路通訊協定第六版（Internet Protocol version 6，IPv6）封包查詢，到了 OpenFlow 1.3 [22] 已經支援高達41 種標頭欄位，常見的支援欄位如，在本系統架構中將會使用 OpenFlow 1.3 作為主要通訊，因為此版本與其他版本相比之下，在各家 OpenFlow Switch 中有較高的支援度。

表 4、部分OpenFlow 1.3可比對之封包欄位

| 可比對之封包欄位 | 描述 |
| --- | --- |
| in\_port | Port number of receiving port |
| in\_phy\_port | Physical port number of receiving port |
| metadata | Metadata used to pass information between tables |
| eth\_dst | Destination MAC address of Ethernet |
| eth\_src | Source MAC address of Ethernet |
| eth\_type | Frame type of Ethernet |
| vlan\_vid | VLAN ID |
| ip\_proto | Protocol type of IP |
| ipv4\_src | Source IP address of IPv4 |
| ipv4\_dst | Destination IP address of IPv4 |
| tcp\_src | Source port number of TCP |
| tcp\_dst | Destination port number of TCP |
| udp\_src | Source port number of UDP |
| udp\_dst | Destination port number of UDP |
| sctp\_src | Source port number of SCTP |
| sctp\_dst | Destination port number of SCTP |
| icmpv4\_type | Type of ICMP |
| icmpv4\_code | Code of ICMP |
| ipv6\_src | Source IP address of IPv6 |
| ipv6\_dst | Destination IP address of IPv6 |
| ipv6\_flabel | Flow label of IPv6 |
| icmpv6\_type | Type of ICMPv6 |
| icmpv6\_code | Code of ICMPv6 |
| ipv6\_nd\_target | Target address of IPv6 neighbor discovery |
| ipv6\_nd\_sll | Source link-layer address of IPv6 neighbor discovery |
| ipv6\_nd\_tll | Target link-layer address of IPv6 neighbor discovery |
| ipv6\_exthdr | Pseudo-field of extension header of IPv6 |

# 系統實作

本章將進行基於 OpenFlow 之頻寬聚集路由器實作的系統介紹，如所示。在本系統中， OpenFlow Controller 使用的是 Ryu（OpenFlow Controller） [23] ， OpenFlow Switch 選擇的是 Open vSwitch [24]， SAPIDO 則是負責對外連接 3G/Wi-Fi [25]， OpenFlow Controller、OpenFlow Switch 和 Client 1 之間的 Switch 為 HP 1410-24G Switch， Client 1 則當為一般客戶端。詳細的規格請參見。

本系統的最終目的會讓 Client 1 使用遠端檔案傳輸（secure copy，scp）指令方式傳送檔案給目的地（Destination），能藉由 OpenFlow 的技術，透過系統的網路位址轉換機制之動態負載平衡演算法，讓流量使用在不同鏈路上，最後封包還能到達目的地。

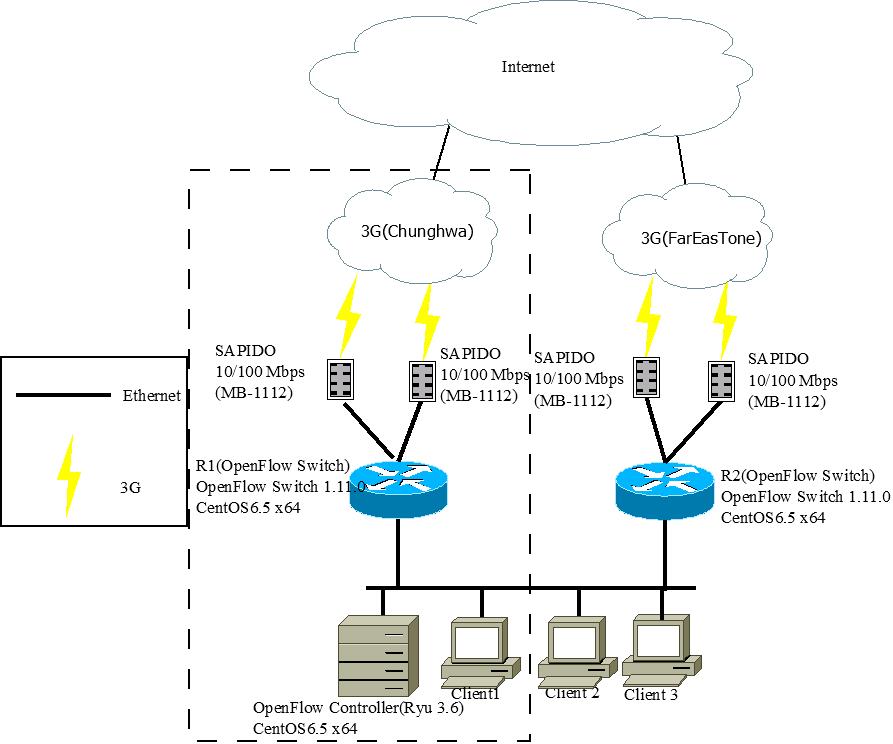


圖 17、網路拓撲架構

表 5、系統詳細規格表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 設備名稱  詳細  項目 | OpenFlow Controller | R1（OpenFlow Switch） | SAPIDO （Chunghwa） | Client 1 | Server | Switch |
| 廠牌型號 | Acer | Acer | SAPIDO | ASUS | Acer | HP |
| CPU | Intel（R）Pentium（R）2.00GHz | Intel（R）Celeron（R） 2.40GHz | RTL8196C BCJ71E3 | Intel（R） Pentium（R）3.40GHz | Intel（R）Pentium（R） 2.00GHz |  |
| RAM | 512MB | 2GB | 16MB | 1.5GB | 2GB | 512 KB |
| 作業系統/韌體版本 | CentOS 6.5 x64 | CentOS 6.5 x64 | 10/100 Mbps （MB-1112） Ver2.0.28 | CentOS 6.5 x64 | FreeBSD 9.0 -RELEASE |  |
| 需安裝軟體 | RYU 3.6 | OpenvSwitch 1.11 and MRTG |  |  |  |  |
| 網卡1 | Gigabit Ethernet | Gigabit Ethernet | 100Mbps Ethernet | Gigabit Ethernet | Gigabit Ethernet |  |
| 網卡2 |  | 100Mbps Ethernet |  |  |  |  |
| 網卡3 |  | 100Mbps Ethernet |  |  |  |  |

## Open vSwitch 安裝與設定

Open vSwitch [24] 為一實作OpenFlow Switch 的虛擬交換器，虛擬交換（Virtual Switch）顧名思義就是建立於虛擬平台上，透過軟體模擬出網路交換器的功能，其提供管理員能夠不依賴於實體網路和硬體，動態地重新配置虛擬環境中的網路架構。

此軟體由 Nicira, Inc. 所主導開發，是一套以 Apache 2.0 license 作為授權條款的開放原始碼軟體， 故只要利用此軟體結合一台有多張網卡的個人電腦，就可以得到一台便宜的 OpenFlow Switch 。此外，由於 OpenFlow Switch 支援諸如XEN [26]、Kernel-based Virtual Machine（KVM）[27]、VirtualBox [28] 等hypervisor ，所以利用此軟體建置一套 OpenFlow 的實驗平台是很方便的事。

* + 1. **Open vSwitch安裝**

在本實作架構下，會將所有 Open vSwitch 建置在 CentOS 6.5 x64 的個人電腦上，其安裝步驟流程如下：

1. 首先，先在機器上使用 wget 安裝 Open vSwitch ，指令如下：  
   # wget <http://openvswitch.org/releases/openvswitch-1.11.0.tar.gz>

# mkdir -p /root/rpmbuild/SOURCES

# cp openvswitch-1.11.0.tar.gz /root/rpmbuild/SOURCES

# cd /root/rpmbuild/SOURCES

# tar -zxvf openvswitch-1.11.0.tar.gz

# cd openvswitch-1.11.0

1. 修改 Open vSwitch kernel module

# vi rhel/openvswitch-kmod-rhel6.spec

搜尋Open vSwitch Linux kernel module.，下一行增加以下這三行

%files

/lib/modules/2.6.32-358.18.1.el6.x86\_64/extra/openvswitch/openvswitch.ko

/etc/depmod.d/openvswitch.conf

# vi rhel/openvswitch-kmod-rhel6.spec.in

搜尋Open vSwitch Linux kernel module.，下一行增加以下這三行

%files

/lib/modules/2.6.32-358.18.1.el6.x86\_64/extra/openvswitch/openvswitch.ko/etc/depmod.d/openvswitch.conf

1. Configure Open vSwitch

# ./boot.sh

# ./configure

# make dist

1. 創建 rpm 安裝包

# rpmbuild -bb rhel/openvswitch.spec

# rpmbuild -bb rhel/openvswitch-kmod-rhel6.spec

1. Install rpm OpenvSwitch 安裝包

# cd /root/rpmbuild/RPMS/x86\_64

# rpm -ivh openvswitch-1.11.0-1.x86\_64.rpm openvswitch-kmod-1.11.0-1.el6.x86\_64.rpm

# rpm -ivh kmod-openvswitch-1.11.0-1.el6.x86\_64.rpm

* + 1. **Open vSwitch 設定**

在本實作架構下，會將所有 Open vSwitch 建置在 CentOS 6.5 x64 的個人電腦上，其設定步驟流程如下：

1. 修改 Open vSwitch 網路卡設定（底線標記之內容須依據實際設備的設定修改）如下：

表 6、Open vSwitch

| **R1（OpenFlow Switch）** |
| --- |
| # vi /etc/sysconfig/network-scripts/ifcfg-eth0  DEVICE=eth0  BOOTPROTO=static  DEVICETYPE=ovs  TYPE=OVSPort  OVS\_BRIDGE=ovs0  HWADDR=44:87:FC:41:6C:0B  ONBOOT=yes  IPADDR=192.168.1.2  NETMASK=255.255.255.0  GATEWAY=192.168.1.254 |
| # vi /etc/sysconfig/network-scripts/ifcfg-eth1  DEVICE=eth1  BOOTPROTO=static  DEVICETYPE=ovs  TYPE=OVSPort  OVS\_BRIDGE=ovs0  HWADDR=00:05:5D:0B:44:31  ONBOOT=yes |
| # vi /etc/sysconfig/network-scripts/ifcfg-eth2  DEVICE=eth2  BOOTPROTO=static  DEVICETYPE=ovs  TYPE=OVSPort  OVS\_BRIDGE=ovs0  HWADDR=00:22:B0:03:D0:4B  ONBOOT=yes  IPADDR=192.168.2.2  NETMASK=255.255.255.0 |
| # vi /etc/sysconfig/network-scripts/ifcfg-ovs0  DEVICE=ovs0  BOOTPROTO=static  DEVICETYPE=ovs  TYPE=OVSBridge  ONBOOT=yes  IPADDR=172.16.1.254  NETMASK=255.255.255.0  HOTPLUG=no |

1. 啟動Open vSwitch 及重新啟動網路

# /etc/init.d/openvswitch restart

# /etc/init.d/network restart

1. Open vSwitch 安裝完成畫面，可藉由ovs-vsctl 印出虛擬Switch的資訊， 如圖 18所示。

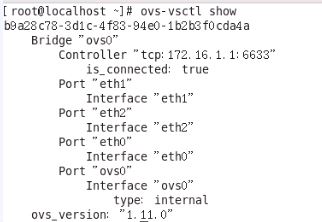


圖 18、Open vSwitch 成功安裝與設定的顯示資訊 R1（OpenFlow Switch）

1. Open vSwitch 與 OpenFlow Controller連接指令如下：

R1（OpenFlow Switch）

# ovs-vsctl set-controller ovs0 tcp:172.16.1.1:6633

1. 啟動 Open vSwitch 所支援的版本指令如下：

R1（OpenFlow Switch）

# ovs-vsctl set bridge ovs0 protocols=OpenFlow10,OpenFlow12,OpenFlow13

## Ryu（OpenFlow Controller）安裝與設定

Ryu（OpenFlow Controller）[23] 是一以 Python 語言實作的 OpenFlow Controller ，提供許多方便易用的 Application Programming Interface（API）供開發者使用，目前 Ryu（OpenFlow Controller）已經支援到 OpenFlow 最新版的 OpenFlow 1.4 [29] 是眾多 OpenFlow Controller 中更新速度最快的，故本論文選擇此 OpenFlow Controller 使用，使用的版本為 OpenFlow 1.3 [22] 。

### Ryu 安裝

在本實作架構下，會將所有 Ryu（OpenFlow Controller）建置在 CentOS 6.5 x64 的個人電腦上，其安裝步驟流程如下：

1. 在安裝 Ryu（OpenFlow Controller）前需要安裝一些相依性軟體，安裝指令如下：

#yum install python-eventlet python-routes python-webob python-paramiko python-devel python-pip libxml2-devel libxslt libxslt-devel git  
#wget <http://mirror-fpt-telecom.fpt.net/fedora/epel/6/i386/epel-release-6-8.noarch.rpm>  
#rpm -ivh epel-release-6-8.noarch.rpm

#wget http://python.org/ftp/python/2.7.6/Python-2.7.6.tar.xz

#tar xf Python-2.7.6.tar.xz

#cd Python-2.7.6

#./configure --prefix=/usr/local --enable-unicode=ucs4 --enable-shared LDFLAGS="-Wl,-rpath /usr/local/lib"

#make && make altinstall

#pip install msgpack-python oslo.config netaddr lxml ecdsa

1. 使用 pip 來下載安裝 Ryu（OpenFlow Controller），安裝指令如下：

# pip install ryu

1. 如果安裝成功可利用ryu-manager 執行python 腳本，作為Ryu （OpenFlow Controller）處理流量，執行畫面如圖 19所示。

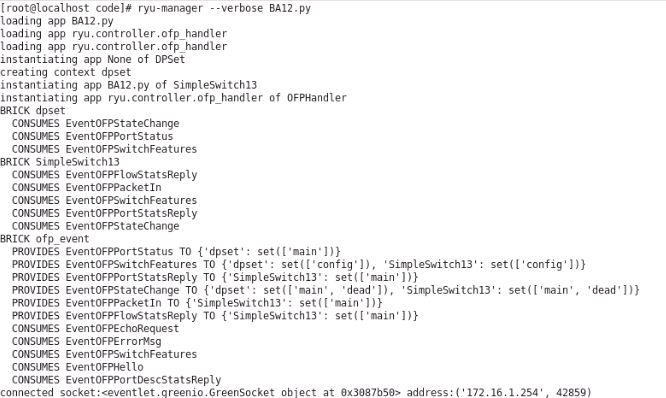


圖 19、Ryu（OpenFlow Controller）執行畫面

### Ryu 設定

在本實作架構下，會將所有 Ryu（OpenFlow Controller）建置在 CentOS 6.5 x64 的個人電腦上，其設定步驟流程如下：

1. 將所需要執行的程式檔案開啟，利用 ryu-manager 執行 python 腳本，如下為 Ryu（OpenFlow Controller）所提供的 app 範例檔：

#ryu-manager --verbose simple\_switch\_13.py

表 7、Ryu（OpenFlow Controller）app 範例檔（simple\_switch\_13.py）

|  |
| --- |
| from ryu.base import app\_manager  from ryu.controller import ofp\_event  from ryu.controller.handler import CONFIG\_DISPATCHER, MAIN\_DISPATCHER  from ryu.controller.handler import set\_ev\_cls  from ryu.ofproto import ofproto\_v1\_3  from ryu.lib.packet import packet  from ryu.lib.packet import ethernet  class SimpleSwitch13(app\_manager.RyuApp):  OFP\_VERSIONS = [ofproto\_v1\_3.OFP\_VERSION]  def \_\_init\_\_(self, \*args, \*\*kwargs):  super(SimpleSwitch13, self).\_\_init\_\_(\*args, \*\*kwargs)  self.mac\_to\_port = {}  @set\_ev\_cls(ofp\_event.EventOFPSwitchFeatures, CONFIG\_DISPATCHER)  def switch\_features\_handler(self, ev):  datapath = ev.msg.datapath  ofproto = datapath.ofproto  parser = datapath.ofproto\_parser  # install table-miss flow entry  #  # We specify NO BUFFER to max\_len of the output action due to  # OVS bug. At this moment, if we specify a lesser number, e.g.,  # 128, OVS will send Packet-In with invalid buffer\_id and  # truncated packet data. In that case, we cannot output packets  # correctly.  match = parser.OFPMatch()  actions = [parser.OFPActionOutput(ofproto.OFPP\_CONTROLLER,  ofproto.OFPCML\_NO\_BUFFER)]  self.add\_flow(datapath, 0, match, actions)  def add\_flow(self, datapath, priority, match, actions):  ofproto = datapath.ofproto  parser = datapath.ofproto\_parser  inst = [parser.OFPInstructionActions(ofproto.OFPIT\_APPLY\_ACTIONS,  actions)]  mod = parser.OFPFlowMod(datapath=datapath, priority=priority,  match=match, instructions=inst)  datapath.send\_msg(mod)  @set\_ev\_cls(ofp\_event.EventOFPPacketIn, MAIN\_DISPATCHER)  def \_packet\_in\_handler(self, ev):  msg = ev.msg  datapath = msg.datapath  ofproto = datapath.ofproto  parser = datapath.ofproto\_parser  in\_port = msg.match['in\_port']  pkt = packet.Packet(msg.data)  eth = pkt.get\_protocols(ethernet.ethernet)[0]  dst = eth.dst  src = eth.src  dpid = datapath.id  self.mac\_to\_port.setdefault(dpid, {})  self.logger.info("packet in %s %s %s %s", dpid, src, dst, in\_port)  # learn a mac address to avoid FLOOD next time.  self.mac\_to\_port[dpid][src] = in\_port  if dst in self.mac\_to\_port[dpid]:  out\_port = self.mac\_to\_port[dpid][dst]  else:  out\_port = ofproto.OFPP\_FLOOD  actions = [parser.OFPActionOutput(out\_port)]  # install a flow to avoid packet\_in next time  if out\_port != ofproto.OFPP\_FLOOD:  match = parser.OFPMatch(in\_port=in\_port, eth\_dst=dst)  self.add\_flow(datapath, 1, match, actions)  data = None  if msg.buffer\_id == ofproto.OFP\_NO\_BUFFER:  data = msg.data  out = parser.OFPPacketOut(datapath=datapath, buffer\_id=msg.buffer\_id,  in\_port=in\_port, actions=actions, data=data)  datapath.send\_msg(out) |

## SAPIDO 路由器設定

傻多（SAPIDO）[25] 是一以超文件標示語言（HyperText Markup Language，HTML）實作的網頁方式呈現，提供多台電腦共享（Share） Internet 連線的同時，還能運用防火牆、Wired Equivalent Privacy（WEP） [30]、Wi-Fi Protected Access （WPA）[31] …等安全防禦機制，確實保護網路環境的安全。對於一般家庭與商業環境應用而設計，強調高安全性、可靠性與操作簡單。

### 韌體更新

依據參考文獻 [32] 將 SAPIDO 韌體更新到最新版本，其中在更新步驟流程如下：

1. 將電腦 IP 位址設為192.168.1.2，網路遮罩為255.255.255.0。
2. 將電腦連接在 SAPIDO 的 LAN埠。
3. 打開瀏覽器在網址欄位那一欄輸入 IP 位址 192.168.1.1，預設帳號為：admin，密碼為：admin。
4. 查看 SAPIDO 型號。
5. 查看目前 SAPIDO　使用版本是否太舊。如果您的韌體版本為 ver1.0.0 先將把它更新到 ver1.0.22 版本的最新，再更新到 ver2.0.28 版本，如所示。



圖 20、SAPIDO 韌體

### 3G網路卡與 LAN 介面設定

依據參考文獻 [25] 選擇您所使用的 SAPIDO 型號版本使用手冊，設定步驟如下：

1. 3G廣域網路設定分為：R1（OpenFlow Switch），如所示。



圖 21、R1（OpenFlow Switch）

1. 兩個 R1（OpenFlow Switch）LAN 設定，如、所示。



圖 22、R1（OpenFlow Switch）eth0 LAN 介面設定



圖 23、R1（OpenFlow Switch） eth2 LAN 介面設定

## 本實驗使用主要 OpenFlow 訊息

本實作使用的訊息如。

表 8、此實作使用的OpenFlow 訊息

|  |  |
| --- | --- |
| **Controller-to-Switch** | |
| PortStatsRequest | OpenFlow Controller 將 port stats 詢問此訊息轉送至 OpenFlow Switch。 |
| FlowStatsRequest | OpenFlow Controller 將 flow stats 詢問此訊息轉送至 OpenFlow Switch。 |
| **Asynchronous** | |
| PortStatsReply | OpenFlow Switch 將 port statistics 訊息回應轉至 OpenFlow Controller。 |
| FlowStatsReply | OpenFlow Switch 將各自 flow statistics 訊息回應轉至 OpenFlow Controller。 |

## 動態負載平衡

傳統網路環境中，網路管理者需要啟動任何一種路由協定演算法，必須對每台路由器（Router），逐一登入命令介面（Command Line Interface，CLI）設定，一旦網路管理者不小心輸入了錯誤的指令或設置變得複雜化後鏈路流量就開始不聽使喚、不受控制和不能即時處理，就很容易造成整個網路癱瘓。在這些問題經常都發生在於中小企業，對網路而言，意味者需要一個更能自動化處理的技術來取代，因此，經由前面第二章的介紹，我們了解相關的背景知識，也看到許多關於動態負載平衡的研究與完成以上 OpenFlow Controller 與 OpenFlow Switch 建置後，本節將利用此 OpenFlow Controller 與 OpenFlow Switch 針對現有動態負載平衡架構提出一增加其更自動化的動態負載平衡架構。因此，接下來部份，本論文將提供三點為主要功能如下：

1. 使用 OpenFlow 技術可以讓網路管理者透過應用程式介面（Application Programmatic Interfaces，API）來管理，並且取代了輸入錯誤的指令或需要逐一登入命令介面設定。
2. 在任何時刻都是透過網路位址轉換機制之動態負載平衡來達成。
3. 除了，當遇到特定的 IP 位址：172.16.1.3 要對外存取資料時，將啟動使用 ICMP Redirect 機制之固定位址。

### 系統架構

本系統架構圖請參見。系統中主要元件有 Ryu（OpenFlow Controller）、R1（OpenFlow Switch）、Client 1、SAPIDO。

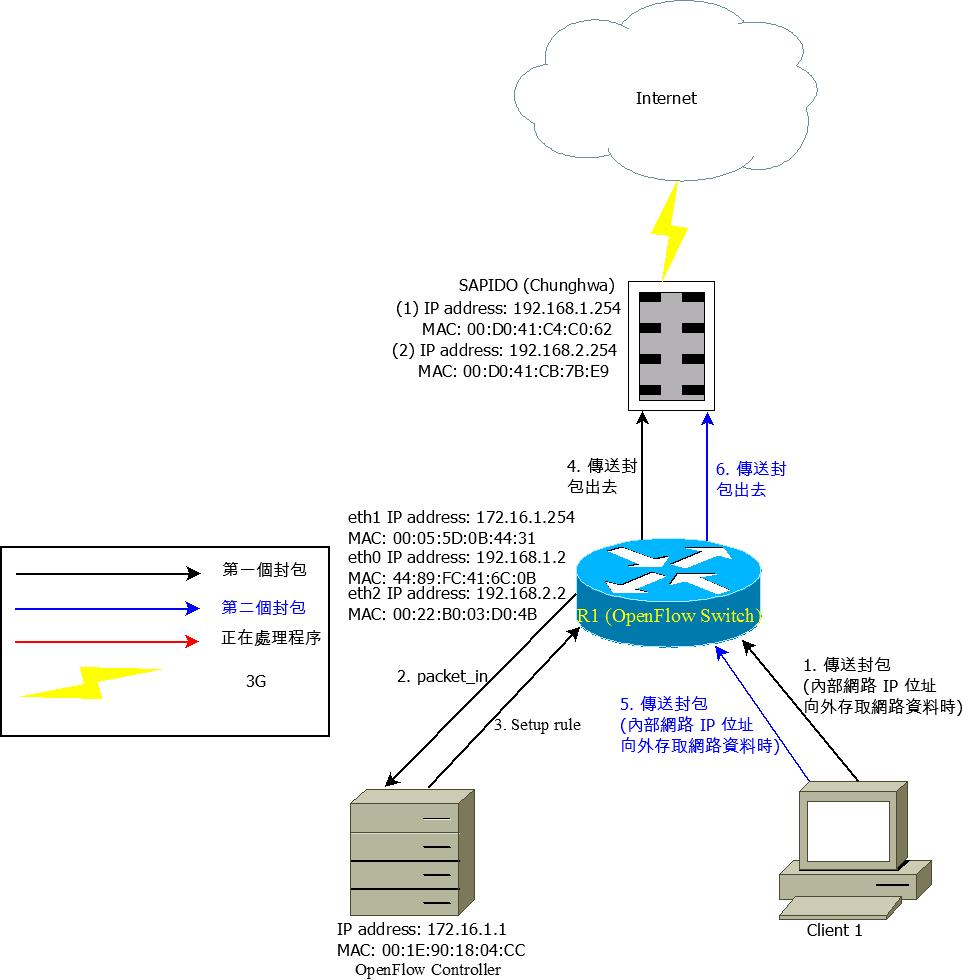


圖 24、系統架構

當 Client 1 需要向對外網路存取資料請求時， R1（OpenFlow Switch）會依照鏈路累積流量（accumulated traffic）最低來動態分配，選擇最適合的鏈路，將封包傳送到達目的地，且具備讓客戶端有最大的使用效益。

### 使用網路位址轉換機制之動態負載平衡

在1990年代開始網路位址轉換（Network Address Translation，NAT）[33] 被提出來，用來解決 IPv4 位址不足的方法，是將 IP 封包通過路由器或防火牆時，建立一個私有位址（Private IP）和公開位址（Public IP）的對應表格，並將重寫封包內部來源端 IP 位址或目的地 IP 位址的技術，因此，被普遍使用在多台機器上但只通過一個公開 IP 位址詢問網際網路的私有網路中。

預設使用場景：當有內部網路位址為 172.16.1.0/24 的網路底下，客戶端訪問學校或企業上載或下載資料時，客戶端封包使用傳輸控制協定（Transmission Control Protocol，TCP）封包給伺服器，封包的來源端 IP 位址為 172.16.1.2，目的地 IP 位址為 163.22.21.84，則使用動態負載平衡，請繼續參考，圖中步驟如下：

1. 一開始 OpenFlow Controller 設定 R1（OpenFlow Switch）的 Flow Table 如黑色字部分， Flow Table 中的 Match 欄位請參照。
2. Client 1 傳送第一個封包給伺服器，如所示、。

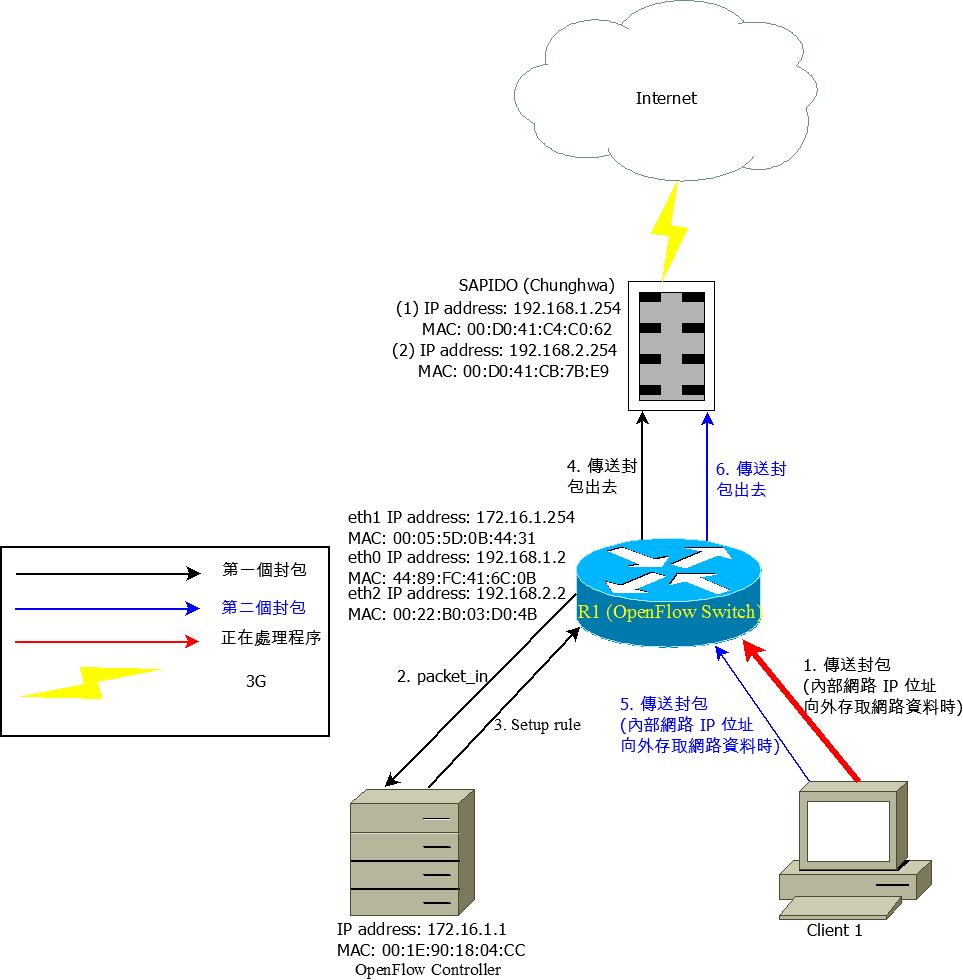


圖 25、NAT-步驟1

表 9、NAT-步驟1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步驟 | SRC IP Address | SRC MAC | DST IP Address | DST MAC | 描述 |
| 1 | 172.16.1.2 | 01:1A:92:02:34:95 | 163.22.21.84 | 00:05:5D:0B:44:31 | Client 1→R1（OpenFlow Switch） |

1. R1（OpenFlow Switch）收到封包，將封包與 Flow Table 進行比對，如沒有比對到，將封包利用 packet-in 訊息傳送給 OpenFlow Controller，如所示、。

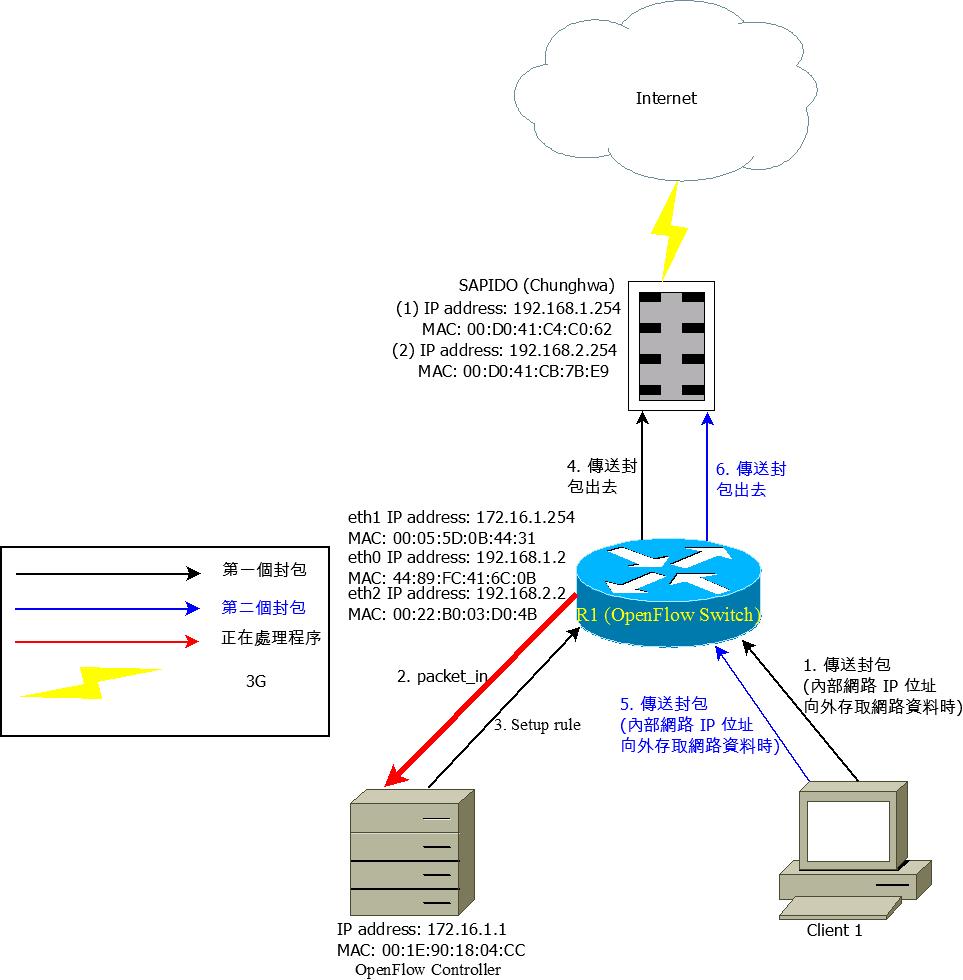


圖 26、NAT-步驟2

表 10、NAT-步驟2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步驟 | SRC IP Address | SRC MAC | DST IP Address | DST MAC | 描述 |
| 2 | 172.16.1.2 | 00:05:5D:0B:44:31 | 163.22.21.84 | 00:1E:90:18:04:CC | R1（OpenFlow Switch）→OpenFlow Controller |

1. OpenFlow Controller 收到 packet-in 訊息，拆開封包，分析封包的來源端 IP 位址、目地端 IP 位址與判斷鏈路累積使用流量，選擇一個最低流量鏈路，作為最佳路徑，寫入 R1（OpenFlow Switch） Flow Entry（Flow Entry 如藍色部分）中，如所示、、所示、。



圖 27、NAT-步驟3

表 11、NAT-步驟3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步驟 | SRC IP Address | SRC MAC | DST IP Address | DST MAC | 描述 |
| 3 | 172.16.1.2 | 00:1E:90:18:04:CC | 163.22.21.84 | 00:05:5D:0B:44:31 | OpenFlow Controller→R1（OpenFlow Switch） |

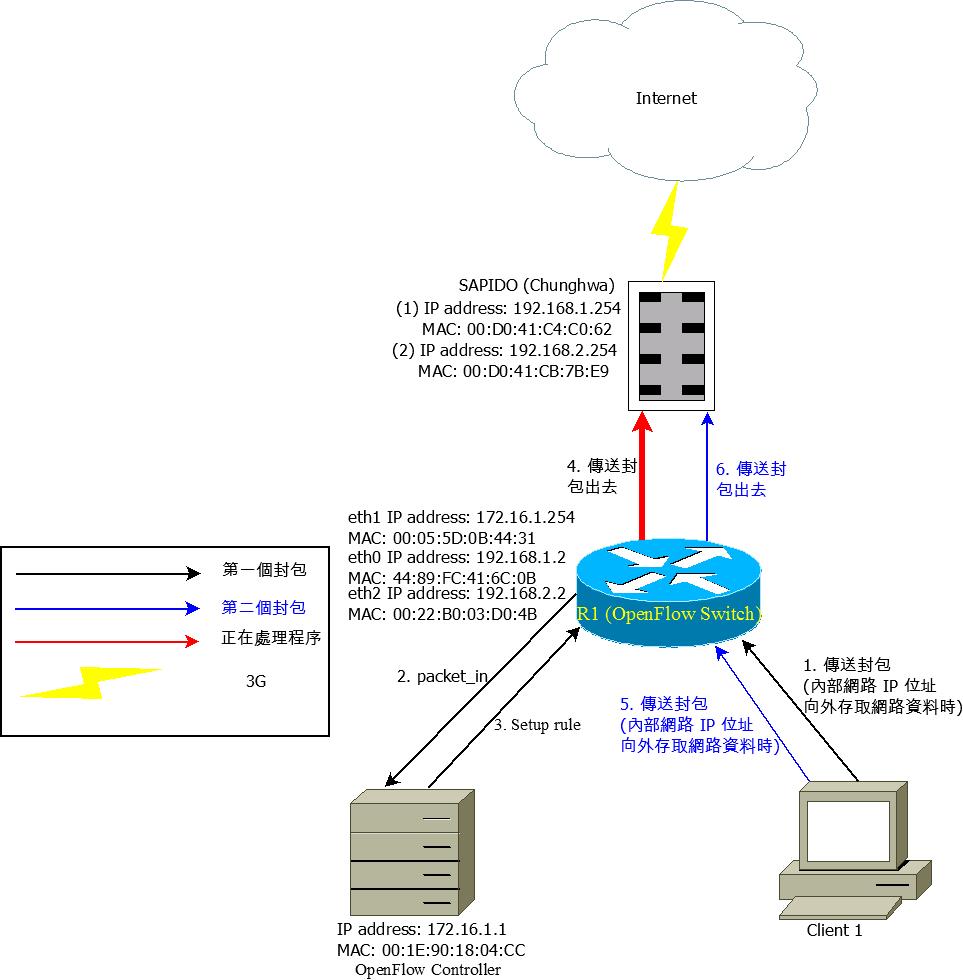


圖 28、NAT-步驟4

表 12、NAT-步驟4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步驟 | SRC IP Address | SRC MAC | DST IP Address | DST MAC | 描述 |
| 4 | 192.168.22 | 00:22:B0:03:D0:4B | 163.22.21.84 | 00:D0:41:CB:7B:E9 | R1（OpneFlow Switch）→SAPIDO（Chunghwa） |

1. OpenFlow Controller 將封包利用 packet-out 訊息傳送給 R1（OpenFlow Switch）。
2. Client 1 傳送第二個封包給伺服器，如所示、。

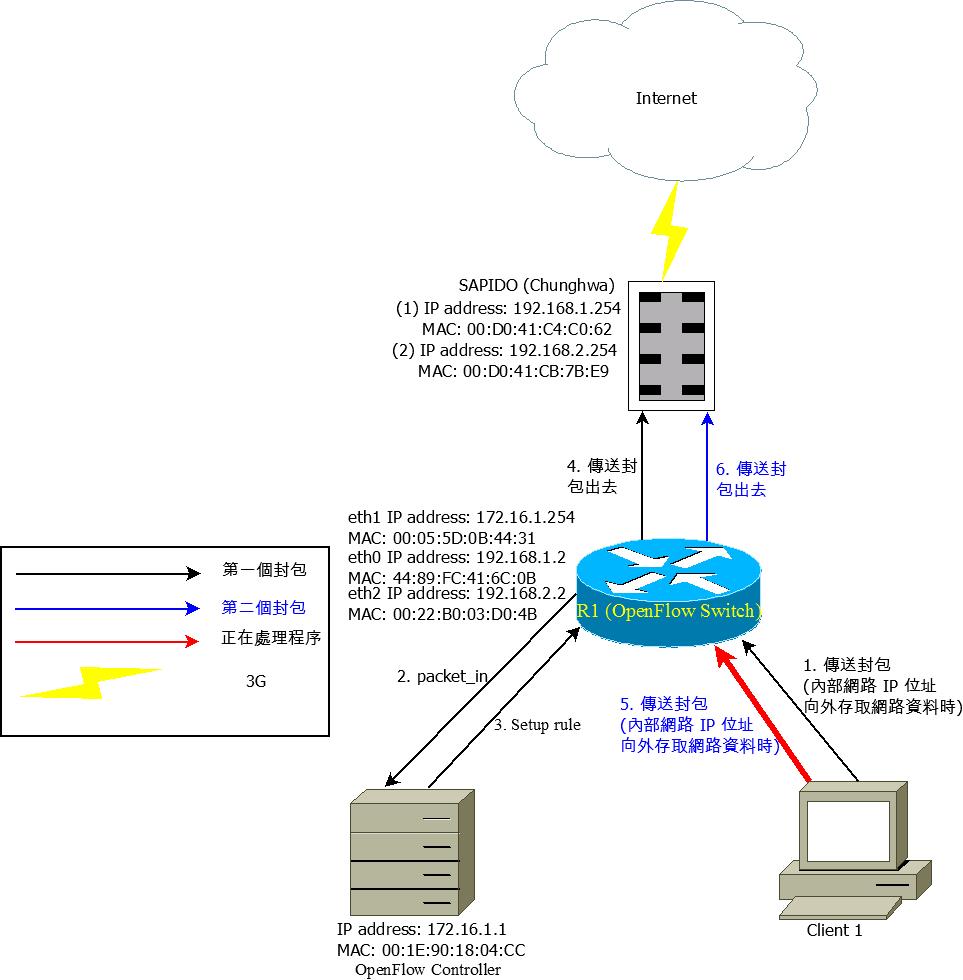


圖 29、NAT-步驟5

表 13、NAT-步驟5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步驟 | SRC IP Address | SRC MAC | DST IP Address | DST MAC | 描述 |
| 5 | 172.16.1.2 | 01:1A:92:02:34:95 | 163.22.21.84 | 00:05:5D:0B:44:31 | Client 1→R1（OpenFlow Switch） |

1. R1（OpenFlow Switch）收到封包，將封包與 Flow Table 進行比對，比對成功，依據 Flow Table 的 Action 設定修改封包的 eth\_src 欄位修改成00:22:B0:03:D0:4B、eth\_dst 欄位修改成 00:D0:41:CB:7B:E9、ipv4\_src 欄位修改成 192.168.2.2、 tcp\_src 的欄位修改成 43103、 Output 的欄位修改成 3，如所示、。

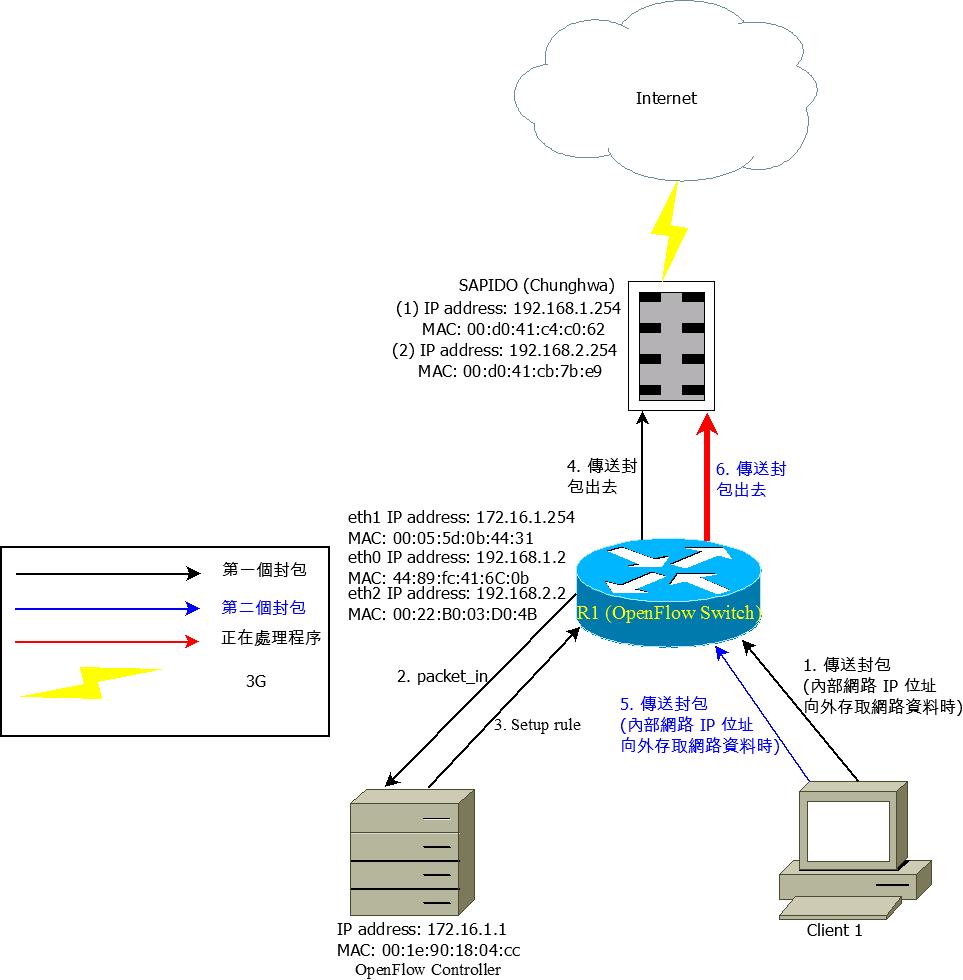


圖 30、NAT-步驟6

表 14、NAT-步驟6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步驟 | SRC IP Address | SRC MAC | DST IP Address | DST MAC | 描述 |
| 6 | 192.168.2.2 | 00:22:B0:03:D0:4B | 163.22.21.84 | 00:D0:41:CB:7B:E9 | R1（OpenFlow Switch）→SAPIDO（Chunghwa） |

在的步驟3間在 R1（OpenFlow Switch）上的演算法方式可以表示成如所示的流程圖，當 R1（OpenFlow Switch）接收到客戶端傳送來的封包後，將取得的封包進行判斷鏈路累積使用流量，選擇一個最低流量鏈路，作為最佳路徑。

本實驗方法針對傳輸控制協定（Transmission Control Protocol，TCP）或是用戶數據報協定（User Datagram Protocol，UDP）封包進行 IP 位址及埠號轉換，非 TCP 或UDP 的 IPv4 封包， OpenFlow Controller 會利用 packet-out message 將封包送至 R1（OpenFlow Switch）處理。

表 15、利用 NAT 機制達成負載平衡之 Flow Table R1（OpenFlow Switch）

|  |  |  |
| --- | --- | --- |
| **Table** | **Match** | **Action** |
| 0 | eth\_type=0x0800,  ipv4\_src=172.16.1.0/24 , ipv4\_dst=172.16.1.0/24 | Normal |
| 0 | eth\_type=0x0800,  ipv4\_src=172.16.1.1,  ip\_proto=6,  tcp\_src=44581,  tcp\_dst=3261 | Set eth\_src=00:22:B0:03:D0:4B,  Set eth\_dst=00:d0:41:cb:7b:e9,  Set ipv4\_src=192.168.2.2,  Set tcp\_src=43103,  Set Output:3 |

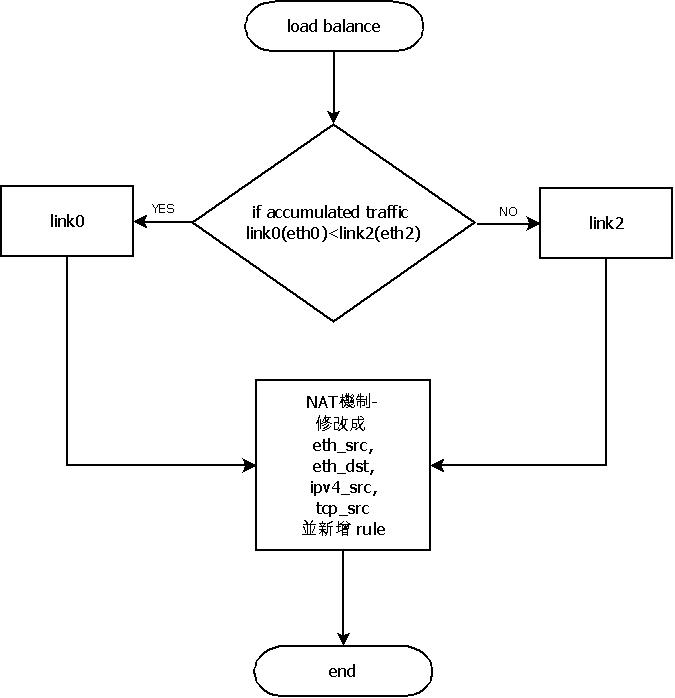


圖 31、R1（OpenFlow Switch）load balance演算法

使用網路位址轉換之動態負載平衡雖然成功將流量分配在各個鏈路上，但是由於此機制無法處理特殊用途的 IP 位址（172.16.1.3）做適當鏈路處理，如遇到 IP 位址為172.16.1.3將某一條鏈路專屬給它用。在3.5.3節提出使用固定位址處理方式。

### 使用 ICMP Redirect 機制之固定位址

網際網路控制訊息協定（Internet Control Message Protocol，ICMP）[34] ，是一個「錯誤偵測與回報機制」，主要目的讓我們能夠檢測網路的連線狀況，確保連線的準確性，其功能分別為：偵測遠端主機是否存在、建立及維護路由資料、重導資料傳送路徑、資料流量控制。

預設使用場景：當有內部網路位址為 172.16.1.0/24 的網路底下，客戶端訪問學校或企業上載或下載資料時，客戶端封包使用傳輸控制協定（Transmission Control Protocol，TCP）封包給伺服器，封包的來源端 IP 位址為 172.16.1.3，目的地 IP 位址為 163.22.21.84，則使用 ICMP Redirect 固定位址，請繼續參考，圖中步驟如下（黑色底線標記之與前一個使用網路位址轉換機制之動態負載平衡不一樣之處）：

1. 一開始 OpenFlow Controller 設定 R1（OpenFlow Switch）的 Flow Table 如黑色字部分， Flow Table 中的 Match 欄位請參照。
2. Client 1 傳送第一個封包給伺服器，如所示、。

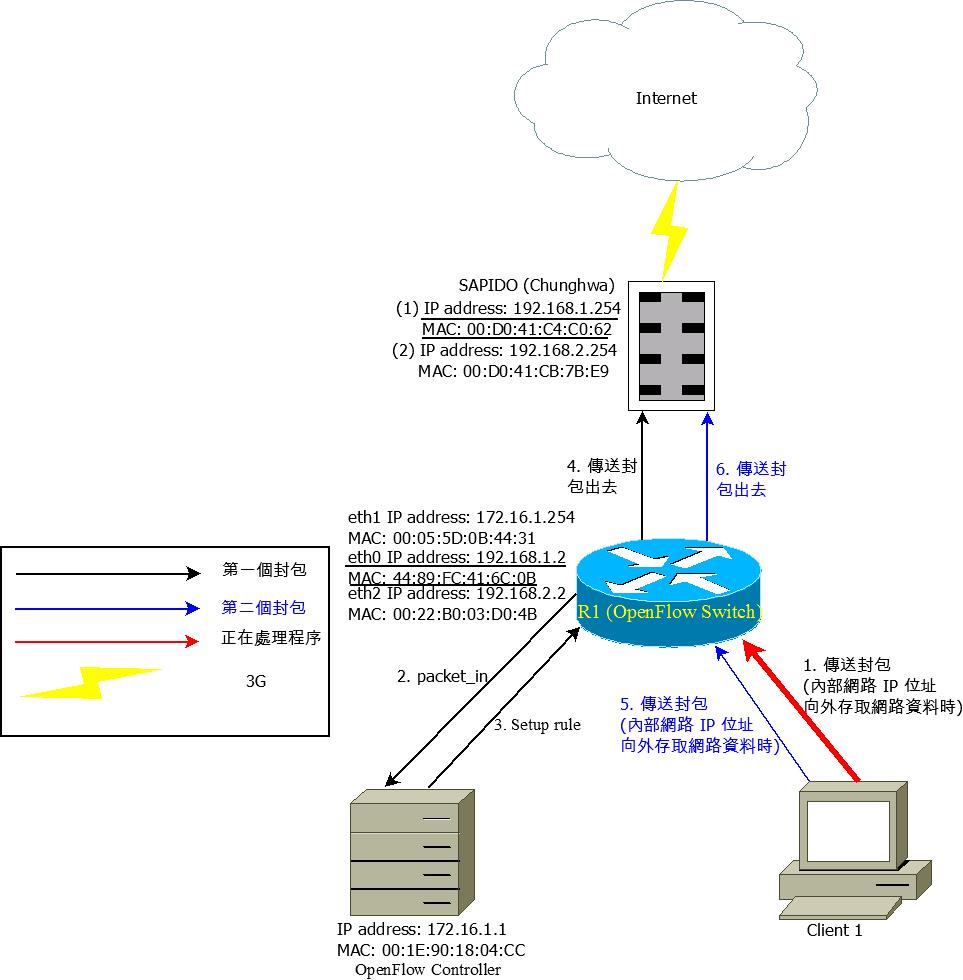


圖 32、ICMP-步驟1

表 16、ICMP-步驟1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步驟 | SRC IP Address | SRC MAC | DST IP Address | DST MAC | 描述 |
| 1 | 172.16.1.3 | 68:05:CA:16:CB:D4 | 163.22.21.84 | 00:05:5D:0B:44:31 | Client 1→R1（OpenFlow Switch） |

1. R1（OpenFlow Switch）收到封包，將封包與 Flow Table 進行比對，如沒有比對到，將封包利用 packet-in 訊息傳送給 OpenFlow Controller，如所示、。

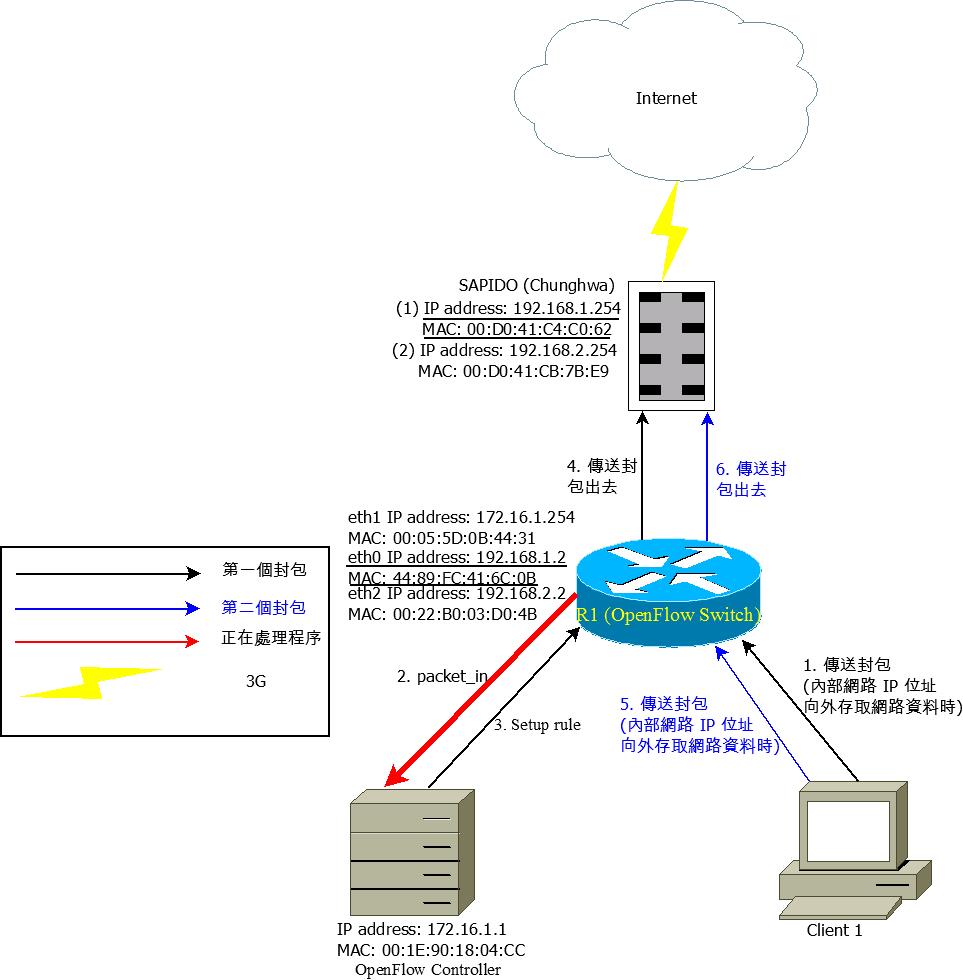


圖 33、ICMP-步驟2

表 17、ICMP-步驟2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步驟 | SRC IP Address | SRC MAC | DST IP Address | DST MAC | 描述 |
| 2 | 172.16.1.3 | 00:05:5D:0B:44:31 | 163.22.21.84 | 00:1E:90:18:04:CC | R1（OpenFlow Switch）→OpenFlow Controller |

1. OpenFlow Controller 收到 packet-in 訊息，拆開封包，分析封包的來源端 IP 位址、目地端 IP 位址與使用 ICMP Redirect 固定位址，寫入 R1 （OpenFlow Switch）Flow Entry（Flow Entry 如 藍色部分）中，如所示、、所示、。

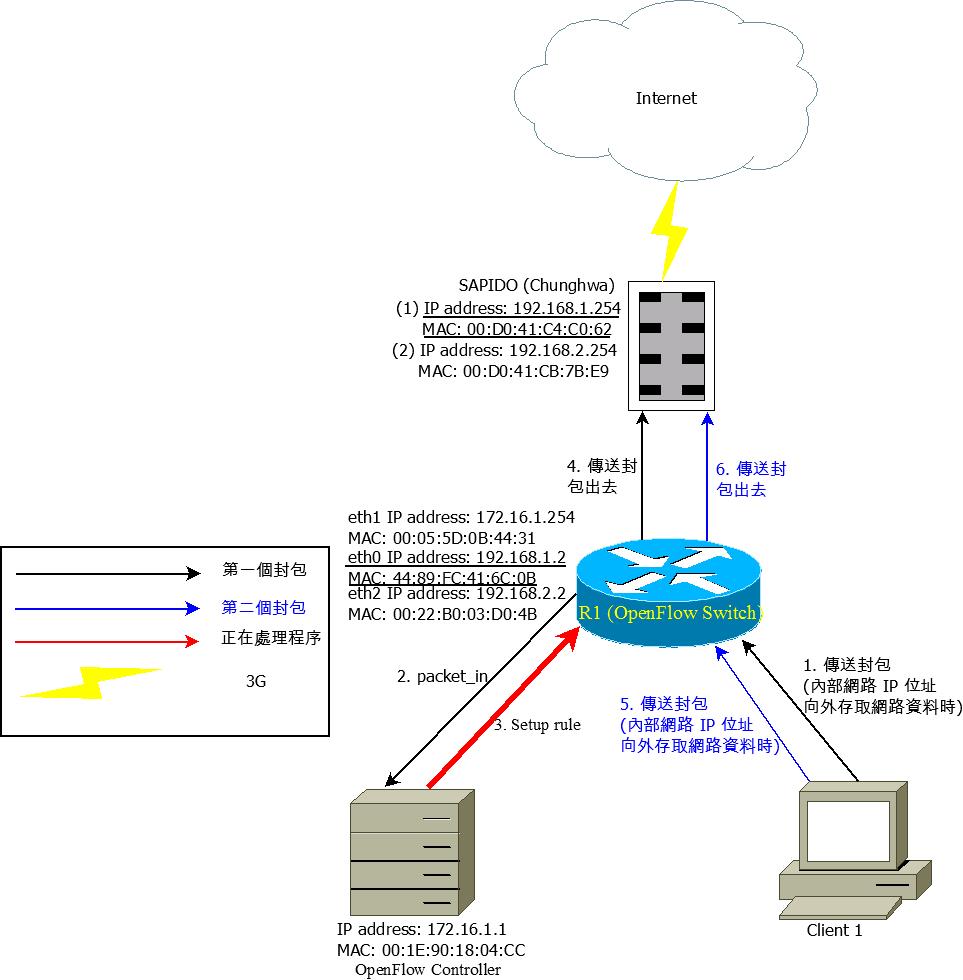


圖 34、ICMP-步驟3

表 18、ICMP-步驟3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步驟 | SRC IP Address | SRC MAC | DST IP Address | DST MAC | 描述 |
| 3 | 172.16.1.3 | 00:1E:90:18:04:CC | 163.22.21.84 | 00:05:5D:0B:44:31 | OpenFlow Controller→R1（OpenFlow Switch） |

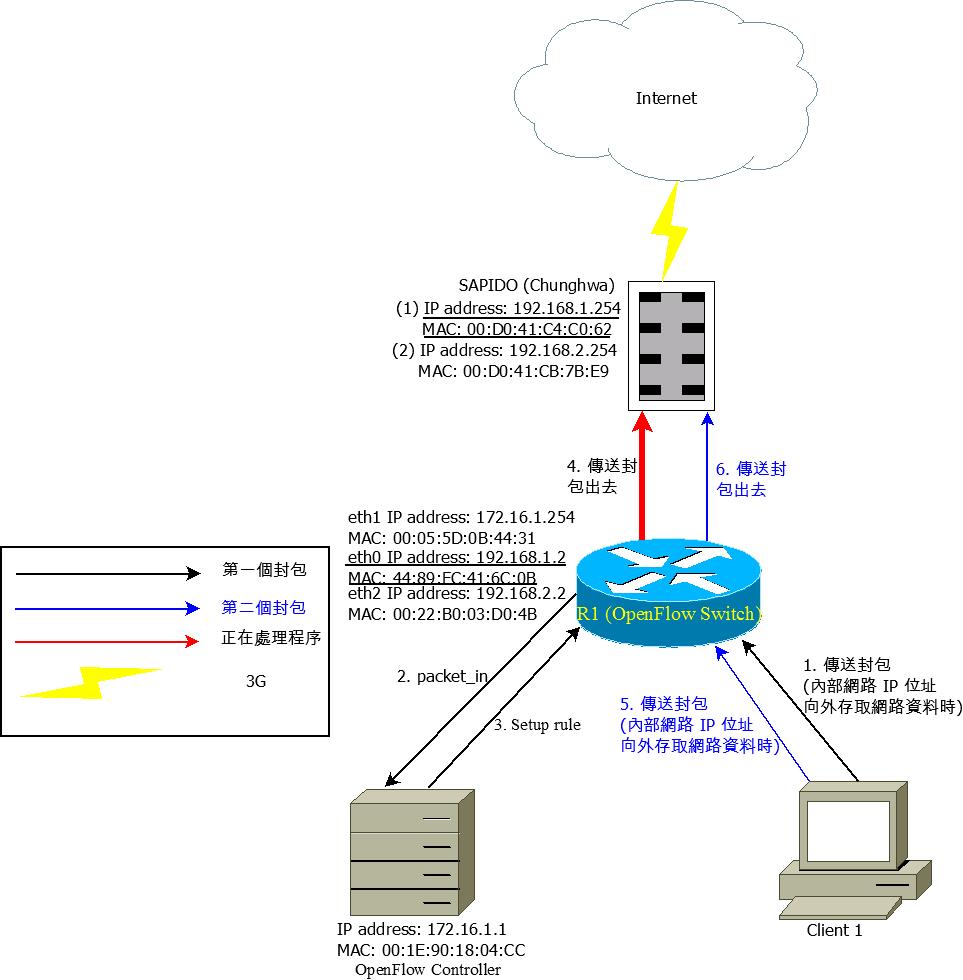


圖 35、ICMP-步驟4

表 19、ICMP-步驟4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步驟 | SRC IP Address | SRC MAC | DST IP Address | DST MAC | 描述 |
| 4 | 192.168.12 | 44:89:FC:41:6C:0B | 163.22.21.84 | 00:D0:41:C4:C0:62 | R1（OpenFlow Switch）→SAPIDO（Chunghwa） |

1. OpenFlow Controller 將封包利用 packet-out 訊息傳送給 R1（OpenFlow Switch）。
2. Client 1 傳送第二個封包給伺服器，如所示、。

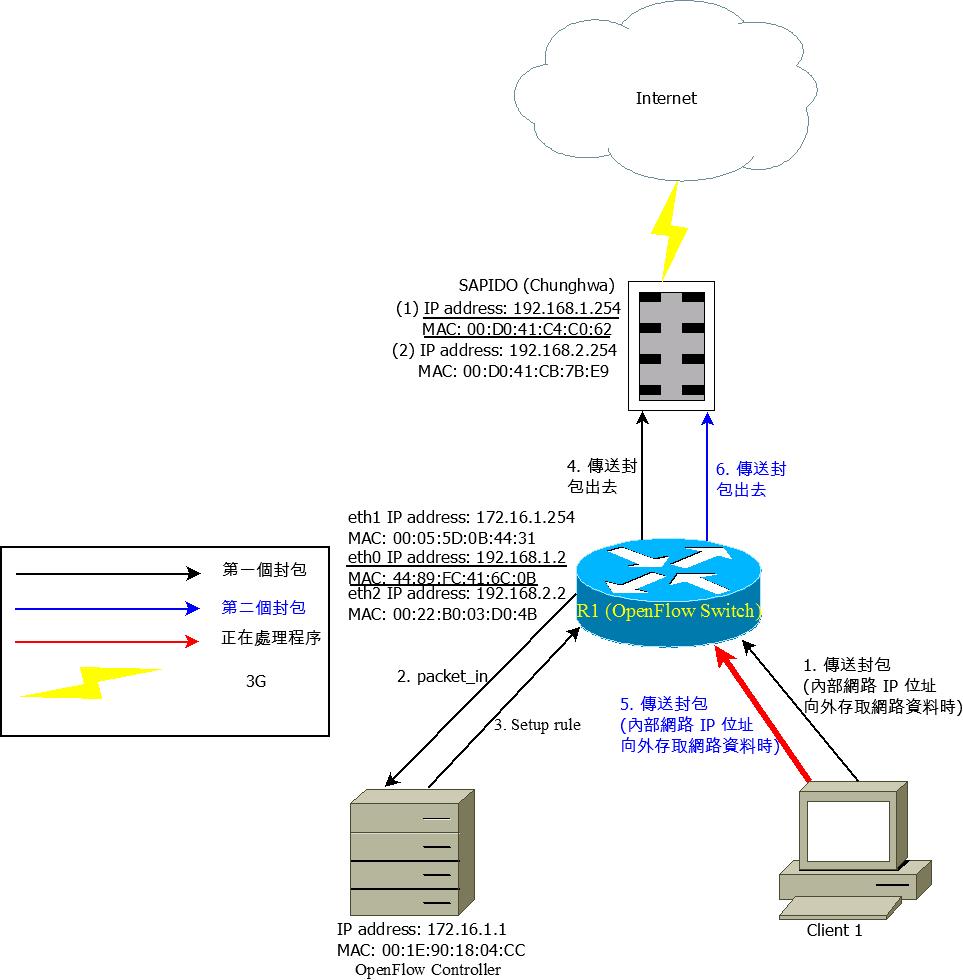


圖 36、ICMP-步驟5

表 20、ICMP-步驟5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步驟 | SRC IP Address | SRC MAC | DST IP Address | DST MAC | 描述 |
| 5 | 172.16.1.3 | 68:05:CA:16:CB:D4 | 163.22.21.84 | 00:05:5D:0B:44:31 | Client 1→R1（OpenFlow Switch） |

1. R1（OpenFlow Switch）收到封包，將封包與 Flow Table 進行比對，比對成功，依據 Flow Table 的 Action 設定修改封包的 eth\_src 欄位修改成 44:89:FC:41:6C:0B、eth\_dst 欄位修改成 00:D0:41:C4:C0:62、ipv4\_src 欄位修改成 192.168.1.2、 tcp\_src 的欄位修改成 36360、 Output 的欄位修改成 1，如所示、。

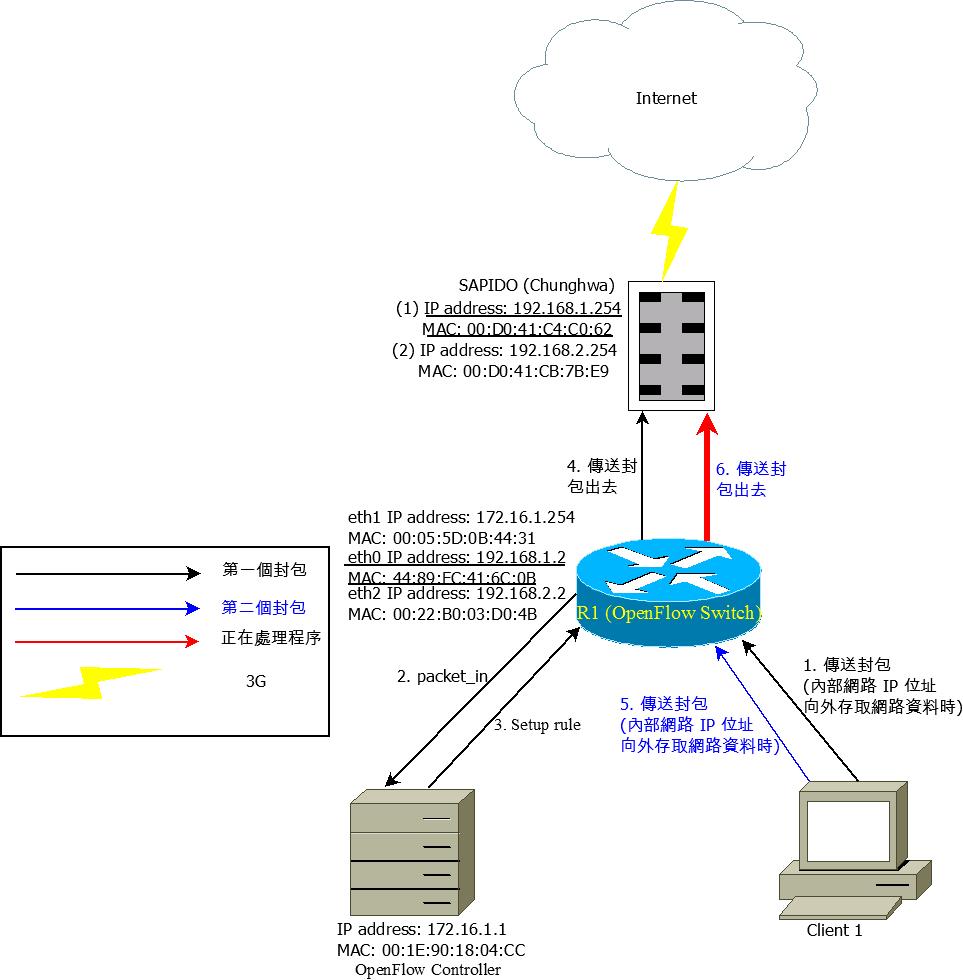


圖 37、ICMP-步驟6

表 21、ICMP-步驟6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步驟 | SRC IP Address | SRC MAC | DST IP Address | DST MAC | 描述 |
| 6 | 192.168.1.2 | 44:89:FC:41:6C:0B | 163.22.21.84 | 00:D0:41:C4:C0:62 | R1（OpenFlow Switch）→SAPIDO（Chunghwa） |

在的步驟3間在 R1（OpenFlow Switch）上的演算法方式可以表示成如所示的流程圖，當 R1（OpenFlow Switch）接收到客戶端傳送來的封包後，將取得的封包進行判斷 IPv4 位址是否屬於為 172.16.1.3，如果是就使用 ICMP Redirect 固定位址修改預設鏈路傳送封包；如果不是就使用預設鏈路傳送封包。

表 22、利用 ICMP Redirect 機制之 Flow Table R1（OpenFlow Switch）

|  |  |  |
| --- | --- | --- |
| **Table** | **Match** | **Action** |
| 0 | eth\_type=0x0800,  ipv4\_src=172.16.1.0/24 , ipv4\_dst=172.16.1.0/24 | Normal |
| 0 | eth\_type=0x0800,  ipv4\_src=172.16.1.3  ip\_proto=6  tcp\_src=49486 | Set eth\_src=44:89:FC:41:6C:0B,  Set eth\_dst=00:D0:41:C4:C0:62,  Set ipv4\_src=192.168.1.2,  Set tcp\_src=36360  Set Output:1 |

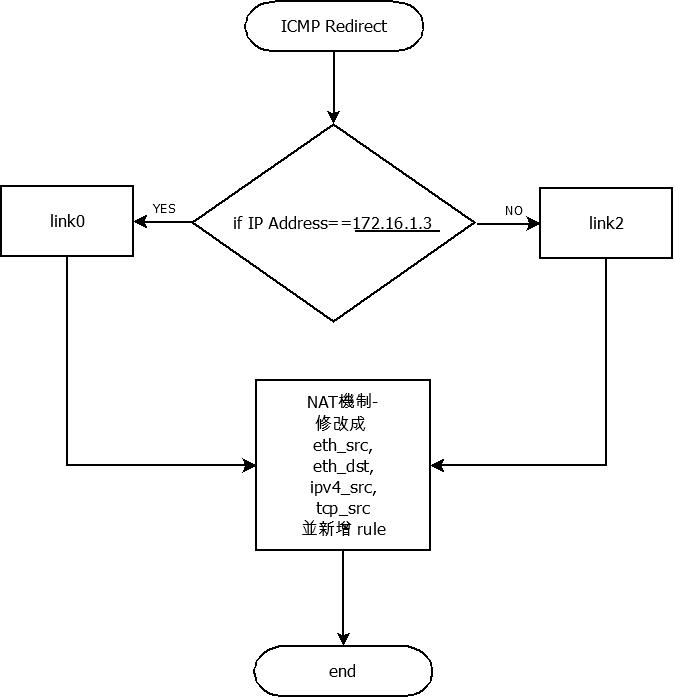


圖 38、R1（OpenFlow Switch）ICMP Redirect演算法

# 效能分析與評估



## 實驗環境與方法

本研究的效能分析主要測試情境為以下兩種情況：1. 每條鏈路使用狀況；2. 鏈路連線時間。如下所示，為測試情境實驗環境架構圖，請繼續參考。

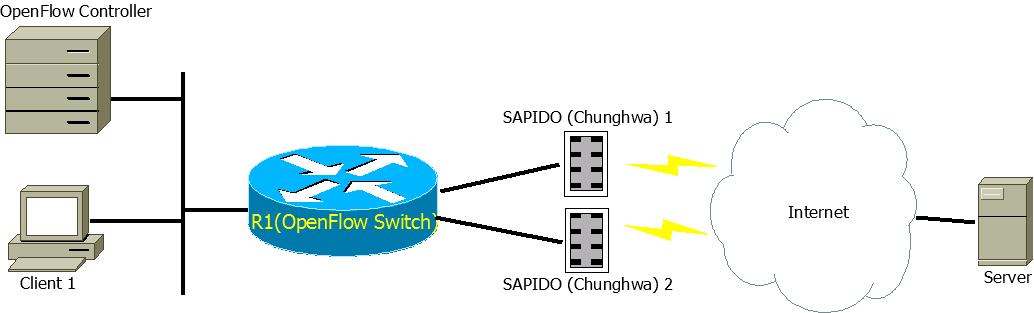


圖 39、實驗環境

## 流量監控工具

本研究的流量監控工具使用 MRTG。MRTG（Multi Router Traffic Grapher）[35] 是一套可用來繪出網路流量圖的軟體，由瑞士奧爾滕的 Tobias Oetilker 與 Dave Rand 所開發。一開始的發展是經由簡單網路管理協定（Simple Network Management Protocol，SNMP）[36] 所觀測並記錄路由器（Router）流量工具，設備取得資訊後繪出網路流量的統計圖，提供管理者方便且有彈性的方式來管理。

* + 1. **MRTG 安裝**

在本實作架構下，會將MRTG 流量監控軟體建置在 R1（OpenFlow Switch）的個人電腦上，其安裝步驟流程如下：

* + 1. MRTG 2.16.2 安裝基本套件

-#yum -y install perl gd libpng zlib httpd net-snmp\* mrtg

* + 1. **MRTG 設定**

在本實作架構下，會將 MRTG 流量監控軟體建置在 R1（OpenFlow Switch） 的個人電腦上，其設定步驟流程如下：

1. 設定 SNMP 與啟動 SNMP

-#vim /etc/snmp/snmpd.conf  
 增加底下兩行（在57行下）

view systemview included .1.3.6.1.2.1.2.2.1.10

view systemview included .1.3.6.1.2.1.2.2.1.16

-#service snmpd restart

1. 建立 MRTG 資料夾到 /var/www/html/mrtg

-#mkdir /var/www/html/mrtg

1. 設定 MRTG 所需要偵測的網路介面卡 （Interface）

-#vi /etc/mrtg/mrtg.cfg

增加底下總共14行（在最後面），每七行為一張網路介面卡

Target[eth0]: [2:public@127.0.0.1](mailto:3:public@127.0.0.1)

MaxBytes[eth0]: 1250000

Title[eth0]: Traffic Analysis

PageTop[eth0]: <H1>192.168.1.254 eth1</H1>

Refresh: 300

Interval: 5

Options[eth2]: growright

Target[eth2]: 4:public@127.0.0.1

MaxBytes[eth2]: 1250000

Title[eth0]: Traffic Analysis

PageTop[eth2]: <H1>192.168.2.254 eth1</H1>

Refresh: 300

Interval: 5

Options[eth2]: growright

1. 產生基本網頁

-執行三次下面指令，產生基本網頁

-# env LANG=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg

1. 產生 MRTG 首頁與重新啟動 httpd

#indexmaker /etc/mrtg/mrtg.cfg > /var/www/mrtg/index.html

# service httpd restart

1. 設定排程

-編輯 /etc/crontab

-# vim /etc/crontab

  \*/5 \* \* \* \* root env LANG=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg > /dev/null 2>&1

1. 啟動 MRTG 監控

-開啟瀏覽器

- localhost/mrtg/

## 數據量測與分析

### 鏈路使用流量實驗與結果

在這一小節，我們進行鏈路累積（accumulated）最低流量效能量測並分析結果。其實驗方式為 Client 1 使用遠端檔案傳輸（secure copy，scp）指令方式傳送檔案給 Server ，這個指令是一個用於在 Linux 下進行遠端複製檔案的命令，類似於複製（cp），不過 cp 只是在本機端進行拷貝不能跨 Server，且 scp 傳輸是加密的。命令格式方式為： scp [參數] [原路徑] [目標路徑]；運作方式為： Client 1 在 Terminal 下，打上這一行指令 scp –P 3621 openflow.pdf sandra@163.22.21.84:~/www/openflow.pdf ，-P為 port 指定資料傳輸用到的埠號，注意是大寫的P。資料傳輸檔案大小大約為 1.2MB ，如所示。

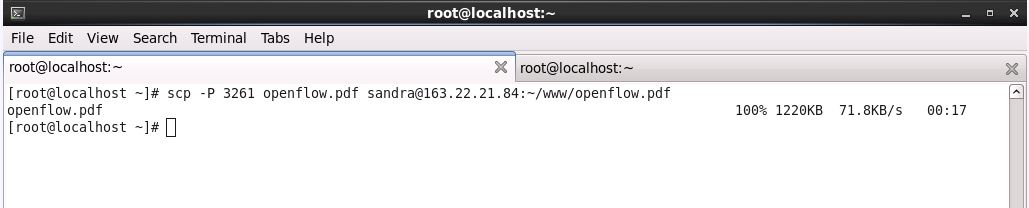


圖 40、使用 scp 指令

由 MRTG 流量工具將每一個網路介面的使用流量顯示出詳細流量資料，傳送10次數據結果，如得知，X軸單位為時間（time），每一個時間間隔為兩小時；Y軸單位為位元組（Bytes）。使用本系統網路位址轉換機制之動態負載平衡，得知結果每個鏈路平均的使用到。

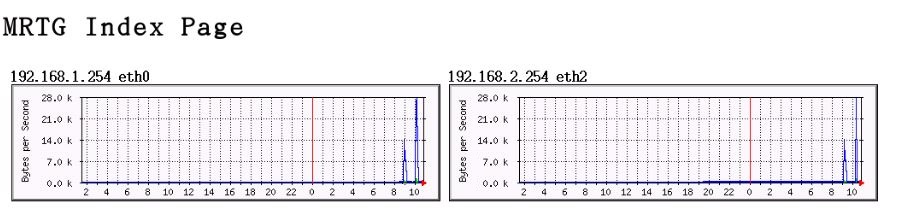


圖 41、accumulated traffic 鏈路結果

### 鏈路連線時間實驗與結果

這一小節，我們進行鏈路累積（accumulated）最低流量的鏈路連線時間效能量測並分析結果。其實驗方式為 Client 1 使用Secure Shell（SSH） 指令，是由 IETF 的網路工作小組（Network Working Group）所制定；SSH 為建立在應用層（Application Layer）和傳輸層（Transport Layer）基礎上的安全協定。運作方式為：Client 1 在 Terminal 下，打上這一行指令 time ssh sandra@163.22.21.84 –p 3261 date ，如所示。

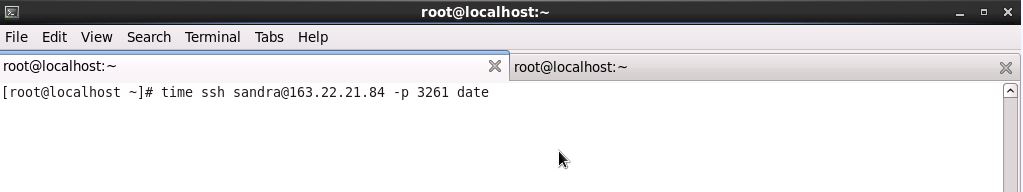


圖 42、使用SSH指令

由 SSH 安全機制將鏈路連線時間顯示出詳細資料，如得知，使用本系統網路位址轉換機制之動態負載平衡，測試出來的時間大約3s。

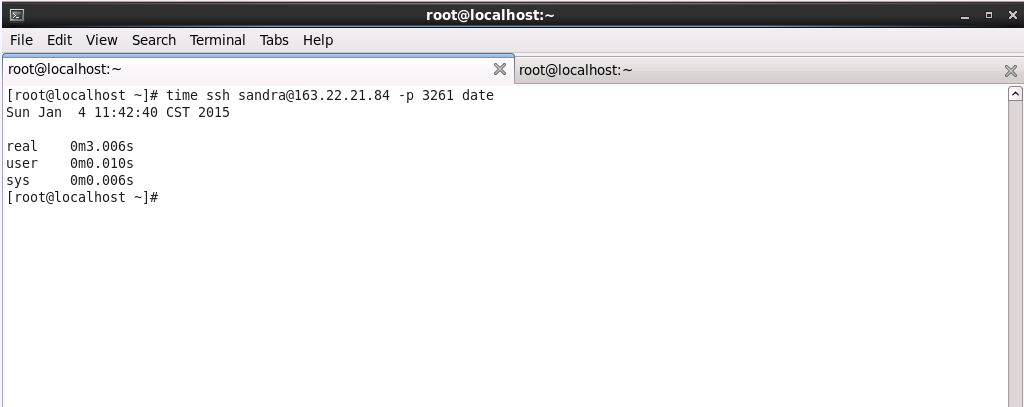


圖 43、SSH 連線時間

# 結論與未來展望

本研究以實作出完整的動態負載平衡系統環境，應用於中小企業都能適用的環境下。使用 Ryu （OpenFlow Controller）和 OpenFlow Switch 搭配，提供網路管理者可免於需要啟動任何一種路由協定演算法，必須對每台路由器（Router），逐一登入命令介面（Command Line Interface，CLI）設定。使用本研究的優點在於以下幾點：1. 低成本：不需購買專屬的硬體設備與免除網路管理者需要逐一對每台路由器手動設定（花費時間成本）。2. 可彈性化：傳統網路架構下，為了呈現各種網路協定，發送傳輸路徑（Data Path）由硬體路由器（Router）決定，所以每一層路由器都要花時間來演算封包路徑，在這樣情況下無法有效使用網路頻寬；而軟體定義網路（Software Defined Network，SDN）則是由控制層（control plane）的軟體來演算最佳封包傳送路徑，資料層（data plane）的設備完全聽命指令來傳送資料。SDN 把網路設備的控制層（control plane）從資料層（data plane）中分離出來，以軟體方式實作。這個架構可以讓網路管理人員，在不更動硬體裝置的前提下，以中央控制方式，用程式重新規劃網路，為控制網路流量提供了新的方法，也提供了核心網路及應用創新的良好平台。最後搭配 MRTG 流量工具顯示圖，協助網路管理者的運作效能分析。

未來可在 R1（OpenFlow Switch）與 R2（OpenFlow Switch）中間多增加一條鏈路，如果判斷 R1（OpenFlow Switch）對外鏈路超過一定流量時，可以將部分的封包導向 R2（OpenFlow Switch）的鏈路來做處理，繼續運作，提供更好的服務品質給客戶端。

# 參考文獻

1. D-Link,[ http://www.dlinktw.com.tw/]
2. D. Thaler, C. Hopps,”Multipath Issues in Unicast and Multicast Next-Hop Selection” RFC 2991, Nov. 2000
3. Cisco EIGRP,[ <http://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/16406-eigrp-toc.html>]
4. Xtera,[http://www.xtera.com/zh-Hant/Home]
5. 麟瑞科技,[http://www.ringline.com.tw/index.php/solution/commonsupply/lp5-101018.html]
6. Cisco 2800 Series Integrated Services Routers,[ http://www.cisco.com/c/en/us/products/routers/2800-series-integrated-services-routers-isr/index.html]
7. Cisco 2900 Series Integrated Services Routers,[ http://www.cisco.com/c/en/us/products/routers/2900-series-integrated-services-routers-isr/index.html]
8. 楊憲其，2002，「在 IPv6 環境下 Multihoming 設計之研究」，國立中正大學資訊工程研究所，碩士論文。
9. Internet Draft, J. Abley, B. Black, and V. Gill,” Goals for IPv6 Site-Multihoming Architectures”, June 13, 2003.
10. C. Hedrick,”Routing Information Protocol”, RFC 1058, June. 1988
11. C. Hopps,”Analysis of an Equal-Cost Multi-Path Algorithm” RFC 2992, Nov. 2000
12. Unequal Cost Path Load Balancing ,[ http://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/13677-19.html#topic1]
13. ZyXEL,[ http://download.zytpe.com.tw/zyxel/product/prod\_detail.php?no=0001154&PHPSESSID=kkngtbax]
14. A.S. Sairam, G. Barua, “Load Balancing Inbound Trafﬁc in Multihomed Stub Autonomous Systems,” Jan. 2009.
15. A.S. Sairam, G. Barua, “Distributed Route Control Schemes to Load Balance Incoming Trafﬁc in Multihomed Stub Networks,” Jan. 2010.
16. H. Long, Y. Shen, M. Guo, F. Tang, “LABERIO: Dynamic load-balanced routing in OpenFlow-enabled networks,” Mar. 2013.
17. Rubio-Loyola J, Galis A, Astorga A, Serrat J, Lefevre L, Fischer A, et al. Scalable service deployment on software-defined networks. IEEE Communications Magazine, 2011; 49(12):84-93.
18. N, McKeown, T. Anderson, H, Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, ”OpenFlow: Enabling Innovation in Campus Networks” ,Mar. 14, 2008
19. OpenFlow Switch Speciﬁcation Version 1.1.0, [https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.1.0.pdf]
20. OpenFlow Switch Speciﬁcation Version 1.1.0, [https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.1.0.pdf]
21. OpenFlow Switch Speciﬁcation Version 1.2.0, [https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.2.0.pdf ]
22. OpenFlow Switch Speciﬁcation Version 1.3.0, [https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf ]
23. Ryu, [ <http://osrg.github.io/ryu/> ]
24. Open vSwitch, [ http://openvswitch.org/ ]
25. SAPIDO 使用手冊, [<http://www.sapido.com.tw/CH/service01.htm>]
26. XEN, [http://www.xenproject.org/ ]
27. KVM, [http://www.linux-kvm.org/page/Main\_Page ]
28. VirtualBox, [https://www.virtualbox.org/ ]
29. OpenFlow Switch Speciﬁcation Version 1.4.0, [https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf ]
30. R. Shirey,”Internet Security Glossary, Version2”, RFC 4949, Aug. 2007
31. F. Bersani, H. Tschofenig,”The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method”, RFC 4764, Jan. 2007
32. SAPIDO 韌體, [<http://www.sapido.com.tw/CH/data/Download/mb1112_f.htm>]
33. P. Srisuresh, M. Holdrege,”IP Network Address Translator (NAT) Terminology and Considerations”, RFC 2633, Aug. 1999.
34. J. Postel,”Internet Control Message Protocol”, RFC 792, Sep. 1981
35. MRTG,[http://oss.oetiker.ch/mrtg/]
36. J. Case, M. Fedor, M. Schoffstall, J. Davin,”A Simple Network Management Protocol (SNMP),” RFC 1157, May. 1990