**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：中華民國一○四年X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

在傳統網路環境中，具有負載平衡功能的系統放置在對外骨幹地方，防制內部區域網路大量流量存取對外服務。網路服務越來越多樣化，然而網路管理者很難以滿足大家需求。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，本論文研究如何利用 OpenFlow 技術開發一套在動態路由器介面 (interface) 上自動切換介面功能，可以協助網路管理者掌握每各介面上都能有效的發揮最大的頻寬保持著兩倍以上頻寬，以提高網路頻寬效能並減少人為因素造成的設置錯誤。本論文所提出的方式，當網路管理者忙於開會或其他事情時，而不能即時處理，使用 OpenFlow 技術就不需要擔心，當設備一多也不需要擔心設備過多處理時間太長。本論文之優點在於利用下一代新的 OpenFlow 技術，可以減少網路管理者不能即時處理，提供一個動態切換介面功能。

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. 2014 Degree Conferred: Master

Student Name: Yi-Shan Lai Advisor Name: Dr. Meng-Lieh, 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](#_Toc405758563)

[致謝 II](#_Toc405758564)

[摘要 III](#_Toc405758565)

[Abstract IV](#_Toc405758566)

[圖目錄 VII](#_Toc405758567)

[表目錄 IX](#_Toc405758568)

[第一章 緒論 1](#_Toc405758569)

[1.1. 研究背景 1](#_Toc405758570)

[1.2. 研究動機 1](#_Toc405758571)

[1.3. 論文架構 2](#_Toc405758572)

[第二章 背景知識及文獻探討 3](#_Toc405758573)

[2.1 Software-Defined Networking (SDN) 3](#_Toc405758574)

[2.2 OpenFlow 4](#_Toc405758575)

[2.3 Multihoming 8](#_Toc405758576)

[2.3.1. Multihoming 的特性 9](#_Toc405758577)

[2.4 相關研究 12](#_Toc405758578)

[2.4.1. 固定指派 (Fixed) 12](#_Toc405758579)

[2.4.2. 依比重輪流指派 (Round-Robin by Weight) 12](#_Toc405758580)

[2.4.3. 依成本分配 (Cost) 13](#_Toc405758581)

[2.4.4. 依連線數指派 (By Connections) 13](#_Toc405758582)

[2.4.5. 依頻寬目前使用量 (By Traffic) 14](#_Toc405758583)

[第三章 系統實作 16](#_Toc405758584)

[3.1 Open vSwitch 安裝與設定 17](#_Toc405758585)

[3.2 Ryu (OpenFlow controller) 安裝與設定 21](#_Toc405758586)

[3.3 SAPIDO 路由器設定 23](#_Toc405758587)

[3.3.1. 韌體更新 23](#_Toc405758588)

[3.3.2. 3G網路卡與 LAN 介面設定 24](#_Toc405758589)

[3.4 OpenFlow Messages and Structures 26](#_Toc405758590)

[3.5 動態負載平衡 27](#_Toc405758591)

[3.5.1. 系統架構 27](#_Toc405758592)

[3.5.2. 使用網路位址轉換機制之動態負載平衡 28](#_Toc405758593)

[3.5.3. 使用 ICMP Redirect 機制之固定位址 33](#_Toc405758594)

[第四章 效能分析與評估 38](#_Toc405758595)

[4.1 實驗環境與方法 38](#_Toc405758598)

[4.2 流量監控工具 38](#_Toc405758599)

[4.3 數據量測與分析 40](#_Toc405758600)

[4.3.1. 封包吞吐量 (Throughput) 實驗與結果 40](#_Toc405758601)

[4.3.2. 封包遺失實驗與結果 40](#_Toc405758602)

[第五章 結論與未來展望 41](#_Toc405758603)

[參考文獻 42](#_Toc405758604)

# 圖目錄

[圖 2-1、軟體定義網路主要概念 3](#_Toc405195626)

[圖 2-2、OpenFlow 網路架構 4](#_Toc405195627)

[圖 2-3、使用 Multihoming 技術連接在同區域 8](#_Toc405195628)

[圖 2-4、使用 Multihoming 技術連接不同區域 9](#_Toc405195629)

[圖 2-5、冗餘備援( Redundant Link ) 10](#_Toc405195630)

[圖 2-6、負載平衡( Load Balance ) 10](#_Toc405195631)

[圖 2-7、策略路由 (Policy Routing) 11](#_Toc405195632)

[圖 2-8、頻寬聚集 (Bandwidth Aggregation)-單一專線 11](#_Toc405195633)

[圖 2-9、頻寬聚集 (Bandwidth Aggregation)-多條鏈路 12](#_Toc405195634)

[圖 2-10、Round-Robin by Weight 13](#_Toc405195635)

[圖 2-11、傳統 3\*3 網路架構[11] 15](#_Toc405195636)

[圖 3-12、網路拓撲架構 16](#_Toc405195637)

[圖 3-13、Open vSwitch 成功安裝與設定的顯示資訊(R1) 21](#_Toc405195638)

[圖 3-14、Ryu (OpenFlow controller)執行畫面 23](#_Toc405195639)

[圖 3-15、SAPIDO 韌體 24](#_Toc405195640)

[圖 3-16、R1 (Chunghwa) 25](#_Toc405195641)

[圖 3-17、R1(eth0) LAN 介面設定 25](#_Toc405195642)

[圖 3-18、R1(eth2) LAN 介面設定 25](#_Toc405195643)

[圖 3-19、系統架構 27](#_Toc405195644)

[圖 3-20、NAT-步驟1 28](#_Toc405195645)

[圖 3-21、NAT-步驟2 29](#_Toc405195646)

[圖 3-22、NAT-步驟3 29](#_Toc405195647)

[圖 3-23、NAT-步驟4 30](#_Toc405195648)

[圖 3-24、NAT-步驟5 30](#_Toc405195649)

[圖 3-25、NAT-步驟6 31](#_Toc405195650)

[圖 3-26、R1(OpenFlow switch) load balance演算法 32](#_Toc405195651)

[圖 3-27、ICMP-步驟1 33](#_Toc405195652)

[圖 3-28、ICMP-步驟2 33](#_Toc405195653)

[圖 3-29、ICMP-步驟3 34](#_Toc405195654)

[圖 3-30、ICMP-步驟4 34](#_Toc405195655)

[圖 3-31、ICMP-步驟5 35](#_Toc405195656)

[圖3-32、ICMP-步驟6 35](#_Toc405195657)

[圖 3-33、R1(OpenFlow switch) ICMP Redirect演算法 36](#_Toc405195658)

[圖 4-34、實驗環境 37](#_Toc405195659)

[圖 4-35、使用 scp 指令測試的結果 39](#_Toc405195660)

# 表目錄

[表格 2-1、常見的OpenFlow 訊息表 5](#_Toc405195661)

[表格 2-2、部分OpenFlow 1.3可比對之封包欄位 7](#_Toc405195662)

[表格 3-3、系統詳細規格表 17](#_Toc405195663)

[表格 3-4、Open vSwitch 19](#_Toc405195664)

[表格 3-5、此實作使用的OpenFlow 訊息 25](#_Toc405195665)

[表格 3-6、NAT-步驟1 28](#_Toc405195666)

[表格 3-7、NAT-步驟2 29](#_Toc405195667)

[表格 3-8、NAT-步驟3 29](#_Toc405195668)

[表格 3-9、NAT-步驟4 30](#_Toc405195669)

[表格 3-10、NAT-步驟5 30](#_Toc405195670)

[表格 3-11、NAT-步驟6 31](#_Toc405195671)

[表格 3-12、利用 NAT 機制達成負載平衡之 flow table (R1) 31](#_Toc405195672)

[表格 3-13、ICMP-步驟1 33](#_Toc405195673)

[表格 3-14、ICMP-步驟2 33](#_Toc405195674)

[表格 3-15、ICMP-步驟3 34](#_Toc405195675)

[表格 3-16、ICMP-步驟4 34](#_Toc405195676)

[表格 3-17、ICMP-步驟5 35](#_Toc405195677)

[表格3-18、ICMP-步驟6 35](#_Toc405195678)

[表格 3-19、利用 ICMP Redirect 機制之 flow table (R1) 36](#_Toc405195679)

[表格 4-20、實驗設備 37](#_Toc405195680)

[表格 4-21、使用 ping 測試的結果 39](#_Toc405195681)

# 緒論

## 研究背景

隨著網路應用多樣化，對網際網路服務提供者 (Internet Service Provider； ISP) [1] 而言，在校園或企業中，不像已往紙本文字而演進到即時訊息、 PPS 網路電視、風行網等多媒體播放軟體，而網路頻寬需求量逐漸增加也無法趕上使用者的需求。對於在這樣快速的網路成長情況下網路頻寬流量越來越大，負載平衡設備是一種經常用來解決鏈路負載過重的方法。然而，負載平衡的技術有很多種，要選擇一種適合的網路架構仍然是一項挑戰，並且負載平衡設備在市場上非常受到大家的喜愛。

傳統的網路設備因作業系統與應用程式各自獨立，網路管理者無法因應各自需求來調整、管理網路資料的傳遞路徑。

## 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)。在1980年代[28]由 CISCO 公司開發出內部網關路由協定 (Interior Gateway Routing Protocol，IGRP)，其目的是想做出一個可以在同一個自治系統 （Autonomous System，AS） 區域中運作的路由協定。內部網關路由協定對網路路徑的衡量標準為：網路頻寬 （Bandwidth） 、網路路徑上各介面的延遲 （Delay） 、網路路徑的穩定度 （Reliability） 、網路路徑的資料負載 （Loading） 、網路路徑的MTU （Maximum Transmission Unit） 值，來決定一個網路路徑的好壞。而自從 CISCO ISO 升級(12.3)後，內部網關路由協定不在支援，改用路由協定為增強型內部網關路由協定 (Enhanced Interior Gateway Routing Protocol ，EIGRP)。

當路由器有多種可到達目的地 (Destination) 網路路徑，並且這些路徑的度量值（跳數、頻寬等）相等時（即所謂的等成本度量），路由器將進行等價負載平衡；當如果發現這些路徑的度量值（跳數、頻寬等）不相等時，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，統一稱為非等價負載平衡。

負載平衡設備目前有以下這些公司：Xtera [2]、路由器具有非等價負載平衡[3]設備的公司，只有 CISCO 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，如果需要用此設備就需要向 CISCO 公司買，這是一個非常相當大的成本，並非每個中小型企業都能負荷的成本與必須網路管理者不停手動設定，為了避免這些額外的成本與減少網路管理者因素造成的設置錯誤等問題。

所以本研究使用 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，並且減少人為因素造成設置錯誤的成本與網路環境可彈性化下使用。

## 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 Multihoming 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

## 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，封包需要經過不停的拆裝和重組，在這樣情況下無法有效發揮網路頻寬。軟體定義網路 （Software Defined Network，SDN）[4]， 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 把網路設備的控制平面 （control plane） 從資料平面 （data plane） 中分離出來，以軟體方式實做，如圖 2-1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。

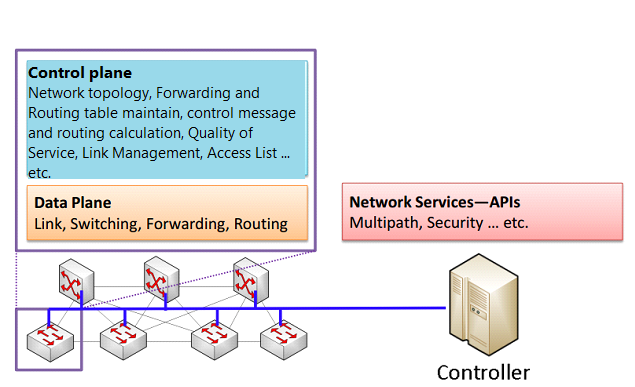


圖 2-1、軟體定義網路主要概念

## OpenFlow

OpenFlow [5] 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定一軟體 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 SSL 加密，確保傳送之間的安全。網路管理人員可藉由 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。

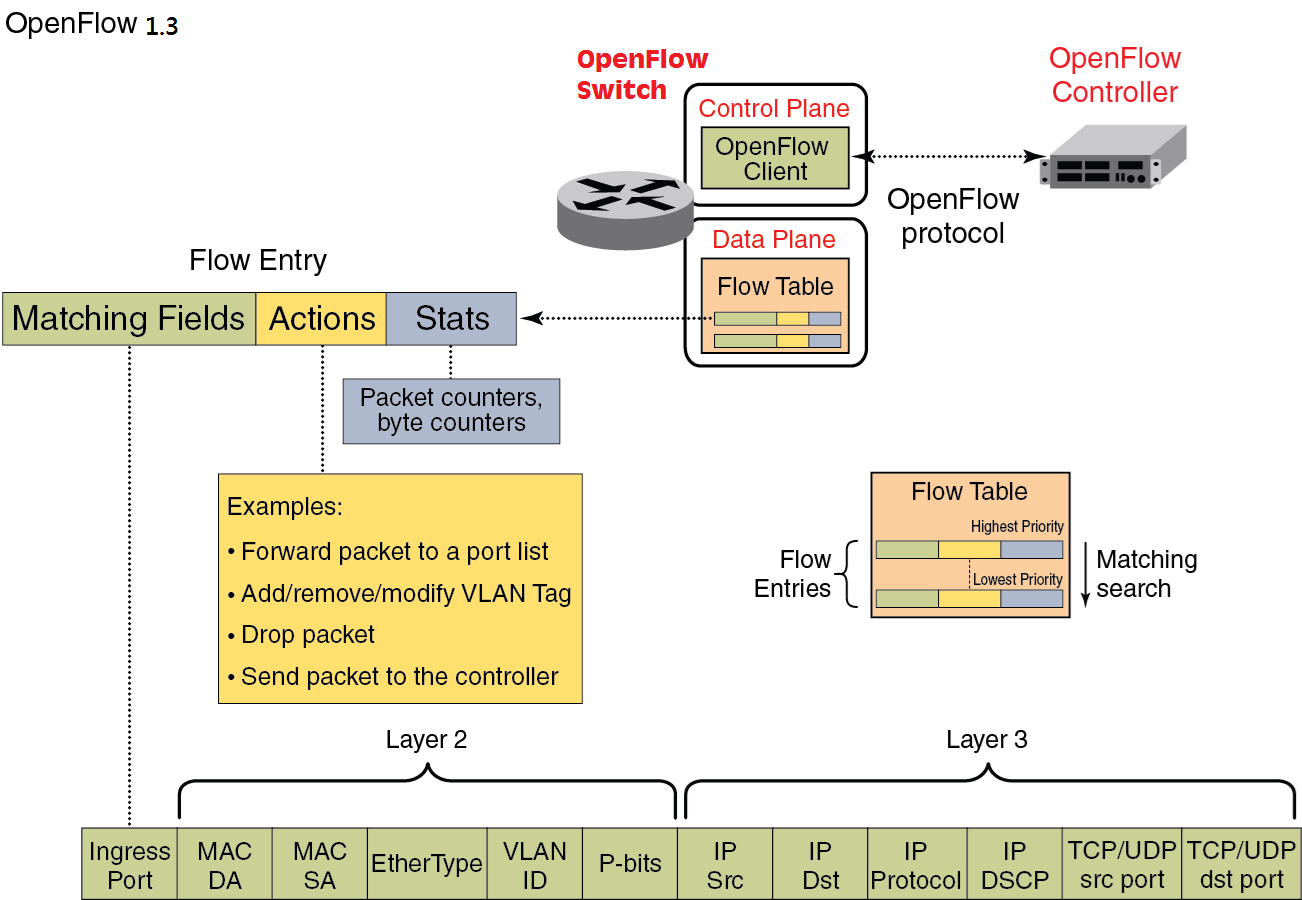


圖 2-2、OpenFlow 網路架構

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status ；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status 欄位； OpenFlow controller 也可以透過 OpenFlow protocol 對 OpenFlow switch 詢問 status 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。

表格 2-1、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 時沒有成供比對到才會產生。 |
| **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] 是提供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。

表格 2-2、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 |

## Multihoming

內送流量備援容錯機制 (Multihoming) 是一種可為中小企業建立穩定網路運作的機制[6][7]。

Multihoming 在中小企業網路運作機制是使用多條對外鏈路來傳輸資料流量，並且可以使用低廉與不需重新佈線，如ADSL、遠傳 ISP 、Cable Modem 等來取代一條專線，降低鏈路成本與改善對外網路效能、可靠度。

Multihoming 主要運作網路環境在於一個計算機主機 (computer host) 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一個區域 (area) 」和「兩個或多個以上的鏈路連接到不同的區域 (area)」，如圖 2-3、圖 2-4所示。

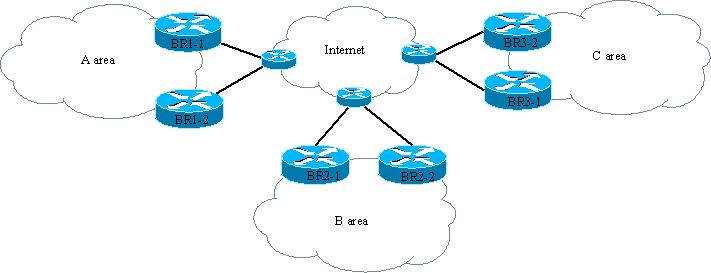


圖 2-3、使用 Multihoming 技術連接在同區域

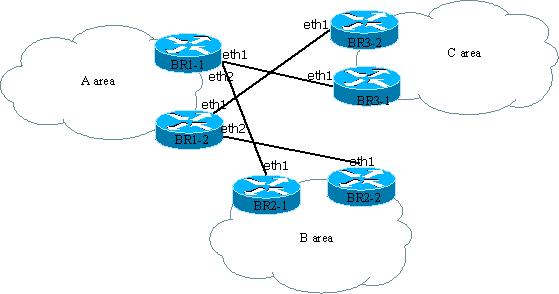


圖 2-4、使用 Multihoming 技術連接不同區域

### 2.3.1. Multihoming 的特性

Multihoming 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)」。

冗餘備援 (Redundant Link) ：當計算機主機主要連線突然斷線時 (eth1) ，還可透過其他計算機主機從其他網路介面建立連線，避免因網路癱瘓而造成損失。如圖 2-5所示，當 Client 1 對外存取資料時，往預設默認 (default) 閘道器 BR(Border Router) 主機傳送封包，由於鏈路 eth1 斷線時，經過 Multihoming 閘道器偵測後，會改由鏈路 eth2 連線至網際網路上。

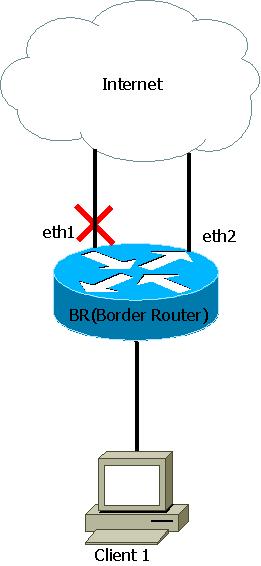


圖 2-5、冗餘備援( Redundant Link )

負載平衡 (Load Balance) ：可先行偵測網路鏈路狀況，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。如圖 2-6所示， Client 1 對外存取資料時，當 Multihoming 閘道器偵測到原先鏈路 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。

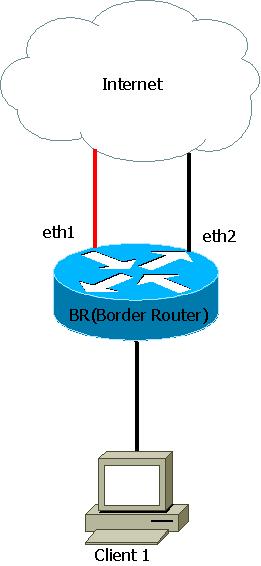


圖 2-6、負載平衡( Load Balance )

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 參數質來決定，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。如圖 2-7所示，當 Client 1 對外存取資料時，往預設默認 (default) 閘道器 BR (Border Router) 主機傳送封包，鏈路 (eth2) 具有較高的頻寬 (HiNet) ，適用於常用的 HTTP 來存取國內/國外資料庫網站，然而鏈路 (eth1) 具有較低的頻寬 (TANet) ，適用於 FTP 來存取學術界資料。

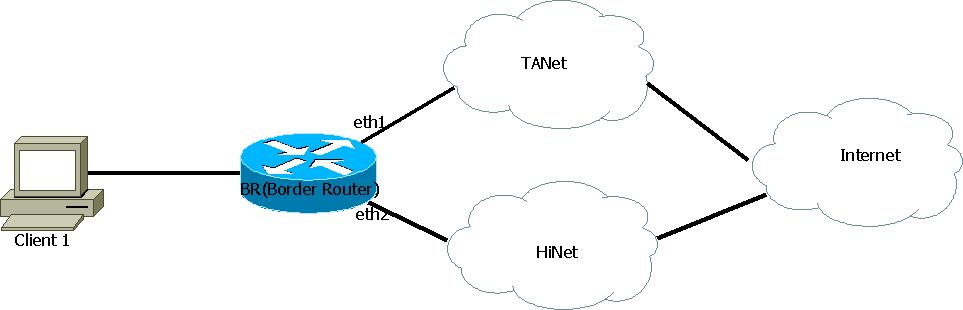


圖 2-7、策略路由 (Policy Routing)

頻寬聚集 (Bandwidth Aggregation) ：將所有鏈路合併成一條較大的頻寬使用。因此原本屬於單一專線就貴於同等頻寬的多鏈路較低的專線，為了符合經濟效益，且擁有多種連線的高可靠性，使用頻寬聚集的概念是一種很好的用法。如圖 2-8所示，為單一專線鏈路 T1 ，圖 2-9所示，為每條鏈路 512Kbps ，透過不同或相同的 ISP 連往 Internet ，具有高可靠性的連線模式。

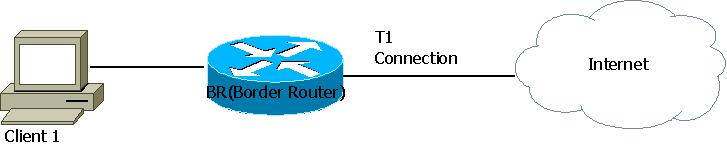


圖 2-8、頻寬聚集 (Bandwidth Aggregation)-單一專線

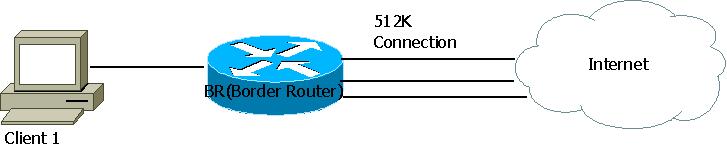


圖 2-9、頻寬聚集 (Bandwidth Aggregation)-多條鏈路

## 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) 、依頻寬目前使用量 (By Traffic)，我們將在各小節中介紹之。

### 2.4.1. 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。

### 2.4.2. 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鏈路上。假設有三台伺服器分別為 Server A、Server B、Server C，各別的比重為7、5、3，則在每一個服務的循環分別所分配到的工作量為 Server A：7/15、Server B：5/15、Server C：3/15。而封包分配到每台伺服器上順序為 AAABABABCABCABC，如所示。

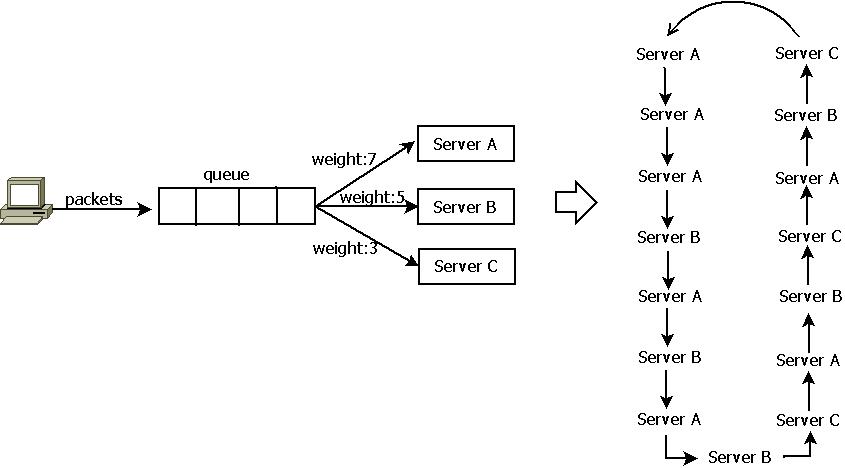


圖 2-10、Round-Robin by Weight

### 2.4.3. 依成本分配 (Cost)

在的研究中，作者提到在同一個區域的自治系統中 (Autonomous System；AS) ，有兩個邊界路由器或兩個以上的網路介面時，連接到多個 ISPs (multihomed) 至網際網路中，如何使用負載平衡平均分配在各種邊緣鏈路上，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成本低就往哪一條路徑走。

### 2.4.4. 依連線數指派 (By Connections)

在[9]的研究中，提到在[10]的研究中，如果一直使用以集中式 (centralized) 的貪婪 (Greedy) 算法基本上有三個邏輯步驟：1. 選擇方式計算鏈路的成本 (cost) ；2. 一但做出了選擇，就不再更改，且希望這樣的選擇可以達到最好的效果；3. 一般情況下，其結果大多是接近最佳解，因此，也很有可能會一直在單一路徑上傳輸過量負擔，於是作者提出了分散式 (Distributed) 算法。

使用分散式作法步驟如下：

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。

### 2.4.5. 依頻寬目前使用量 (By Traffic)

在[11]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) ：以階層式設計的核心層是互連網路的高速骨幹 (backbone) 。核心層是分佈層設備之間互聯的關鍵，因此核心層要保持高可用性和高備援性是非常重要的。核心層彙聚所有分佈層設備發送的流量，因此必須能夠快速轉送大量的資料；級別2 (level 2) 為分佈層 (distribution layer) ：分佈層先彙聚存取層交換器發送的資料，再將其傳輸到核心層，最後發送到最終的目的地。分佈層使用政策控制網路的通訊流並透過在存取層定義的虛擬 (VLAN) 之間執行路由功能來規劃廣播領域。利用 VLAN ，可將交換器上的流量分成不同的網段，置於互相獨立的子網內。為了確保可靠性，分佈層交換器通常是高性能、高可用性、具有高級備援功能的設備；級別3 (level 3) 為存取層 (access layer) ：存取層負責連接終端設備 (例如：電腦、印表機) 以提供對網路中其他部分的存取。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。

在3\*3網路架構每個 Switch 都緊緊相連著每台 Switch ，如所示。例子：h1 to h16 之間進行網路視訊與通話， h1 要到 h16 有多種路徑選擇，現在 h1 發送出封包給上層 S1 (有四條鏈路路徑) ，然後會判斷目前鏈路流量路徑誰為最忙碌，誰最為輕鬆，必須把每個鏈路都算出來，結果來選擇最輕路徑(lightest path)，進行通訊。

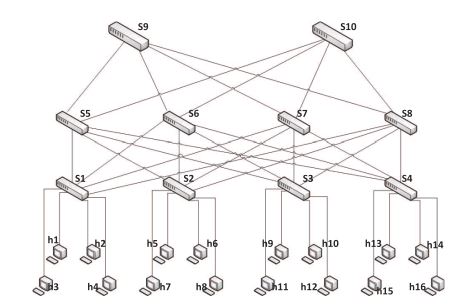


圖 2-11、傳統 3\*3 網路架構[11]

# 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) [16] ， OpenFlow switch 選擇的是 Open vSwitch [12]， SAPIDO 則是負責對外連接 3G/Wi-Fi [18]， OpenFlow controller、OpenFlow switch 和 Client 之間的 switch 為 HP 1410-24G Switch， Client1~Client4 則當為一般使用者。詳細的規格請參見。

本系統的最終目的會讓 Client1 使用上傳檔與下載檔分別讓流量使用在不同鏈路上，能藉由 OpenFlow 的技術，動態的方式將封包平均分配在不同鏈路上，最後封包還能到達目的地。

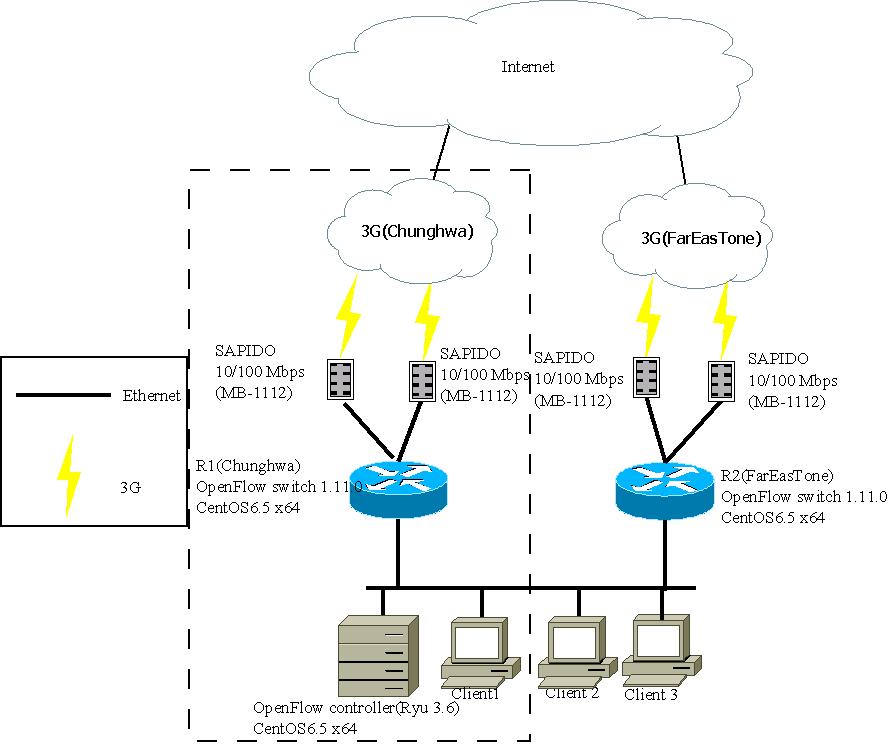


圖 3-12、網路拓撲架構

表格 3-3、系統詳細規格表

|  |  |  |  |
| --- | --- | --- | --- |
| 設備 | 硬體規格 | 作業系統/韌體版本 | 實作軟體 |
| OpenFlow controller | PC | CentOS 6.5 x64 | RYU 3.6 |
| OpenFlow switch | PC | CentOS 6.5 x64 | OpenvSwitch 1.11 |
| Client1~4 | PC | CentOS 6.5 x64 |  |
| R1(Chunghwa) | SAPIDO | 10/100 Mbps (MB-1112) Ver2.0.28 |  |
| Switch | HP 1410-24G |  |  |

## Open vSwitch 安裝與設定

Open vSwitch [10] 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 [13]、Kernel-based Virtual Machine（KVM ）[14]、VirtualBox [15] 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。

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，其安裝與設定步驟流程如下：

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. 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：

表格 3-4、Open vSwitch

|  |
| --- |
| R1(Chunghwa) |
| # 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 及重啟網路  
   # /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， 如所示。

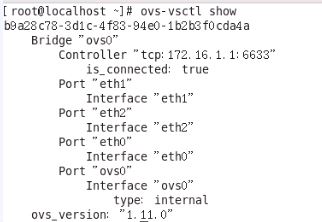


圖 3-13、Open vSwitch 成功安裝與設定的顯示資訊(R1)

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

(R1)

# 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)

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

## Ryu (OpenFlow controller) 安裝與設定

Ryu (OpenFlow controller)[16] 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 [23] 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] 。

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 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OpenFlow controller 處理流量，執行畫面如所示。

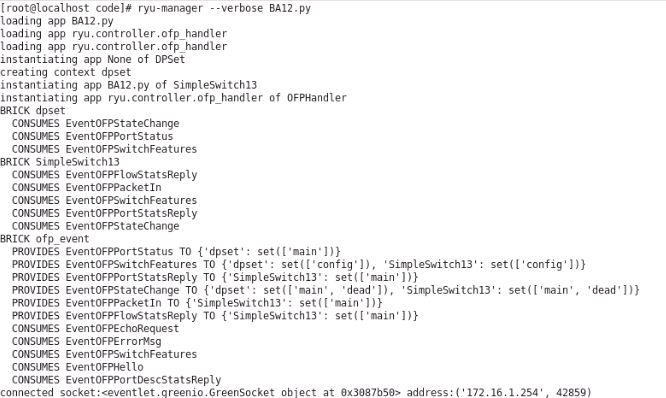


圖 3-14、Ryu (OpenFlow controller)執行畫面

## SAPIDO 路由器設定

傻多 (SAPIDO) 主要特點內建 AUTO WAN ，可自動偵測連線網路類型，3G/4G SIM 卡更是隨插即用，輕鬆完成網路分享。隨時隨地都能使用無線網路來瀏覽網頁、打Skype、聽音樂、玩遊戲。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[17]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，在更新到 ver2.0.28 版本，如所示。



圖 3-15、SAPIDO 韌體

### 3G網路卡與 LAN 介面設定

依據參考文獻[18] 選擇您所使用的 SAPIDO 型號版本使用手冊，其中在設定過程的步驟如下：

1. 3G廣域網路設定分為：R1 Chunghwa，如所示。



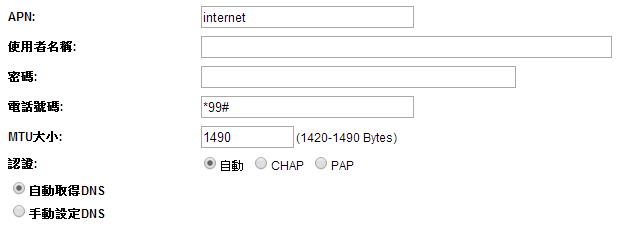


圖 3-16、R1 (Chunghwa)

1. 兩個 R1(Chunghwa) LAN 設定，如、所示。



圖 3-17、R1(eth0) LAN 介面設定



圖 3-18、R1(eth2) LAN 介面設定

## OpenFlow Messages and Structures

本實作使用的訊息如。

表格 3-5、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

在過去網路管理者需要不斷的對鏈路流量設置，一旦設置變得複雜化後鏈路流量就開始不聽使喚與不受控制或者是網路管理者不能即時處理，然而在這些問題經常都發生在於校園或者企業網路而言，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，本論文將提供兩點為主要功能為以下：

1. 使用 OpenFlow 技術可以讓網路管理者更加彈性，並且取代了網路管理者不能即時處理鏈路流量分配。
2. 針對特定的 IP 位址設定固定的鏈路。

### 系統架構

本系統架構圖請參見圖 3-19。系統中主要元件有 Ryu (OpenFlow controller)、OpenFlow switch、IPv4 客戶端、SAPIDO。

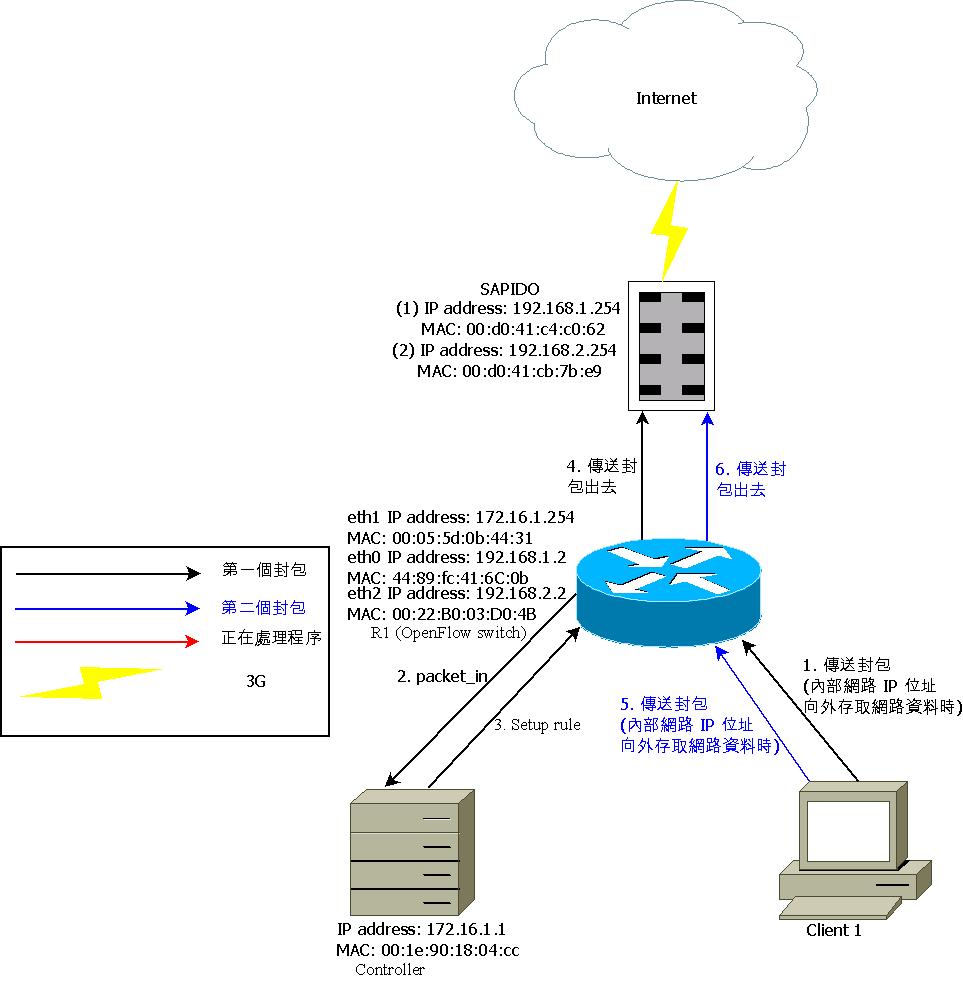


圖 3-19、系統架構

當客戶端需要向對外網路存取資料請求時， R1(OpenFlow switch) 會依照鏈路目前使用流量來動態分配，選擇最適合的路徑，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) 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 172.16.1.1，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 欄位請參照表格 3-3。
2. Client 1 傳送第一個封包給伺服器，如所示、。

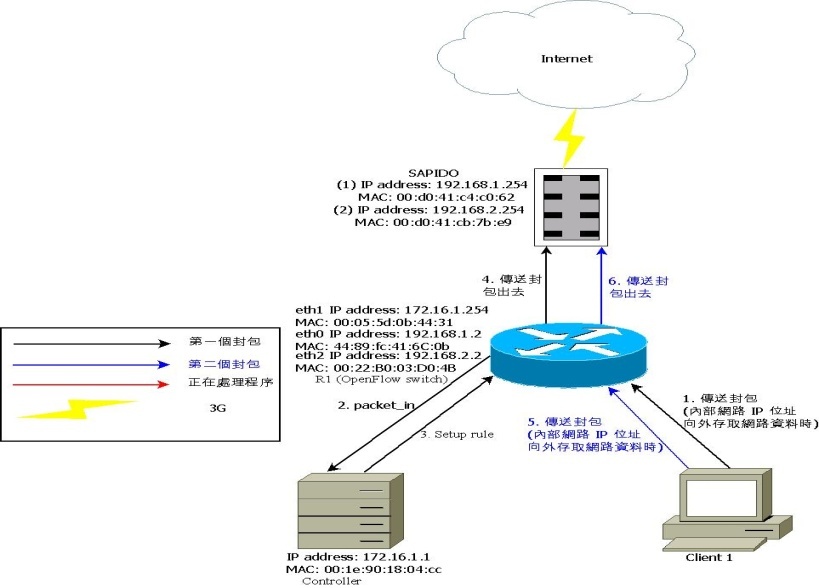


圖 3-20、NAT-步驟1

表格 3-6、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 |

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、。

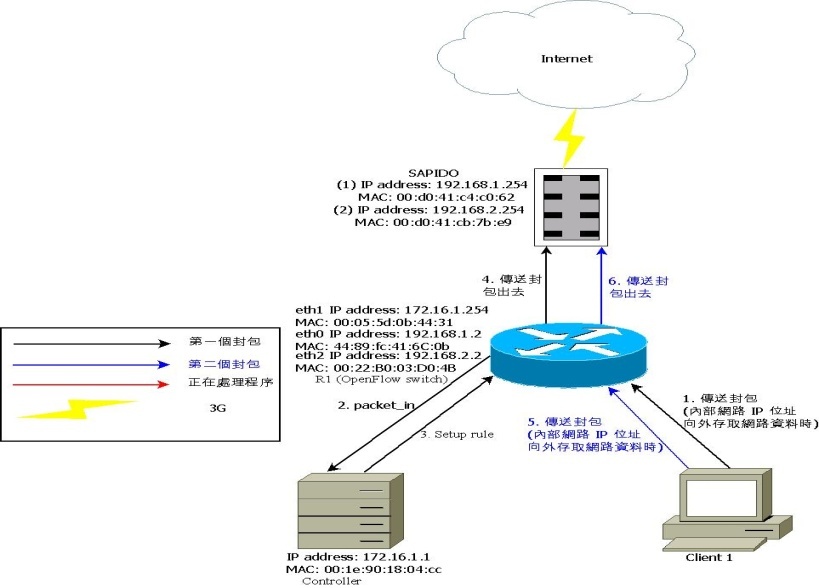


圖 3-21、NAT-步驟2

表格 3-7、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→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 位址與判斷目前鏈路使用流量，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、。

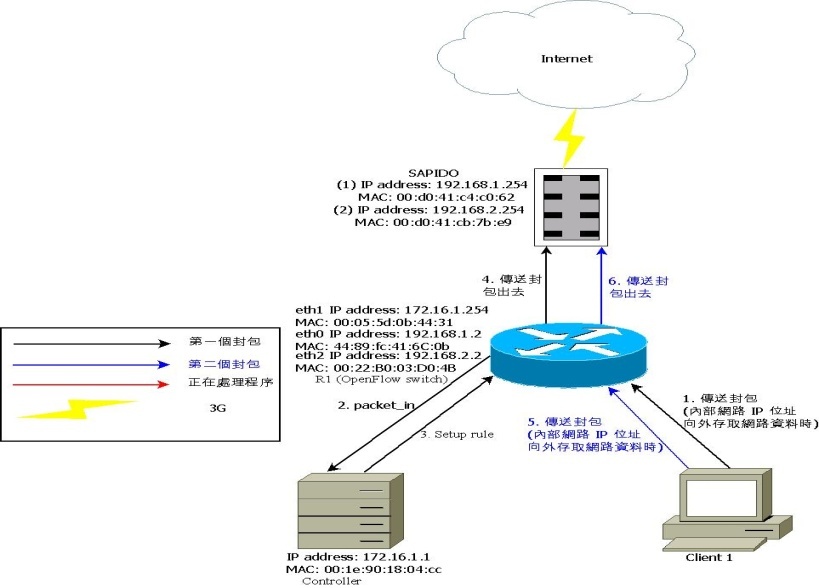


圖 3-22、NAT-步驟3

表格 3-8、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 | Controller→R1 |

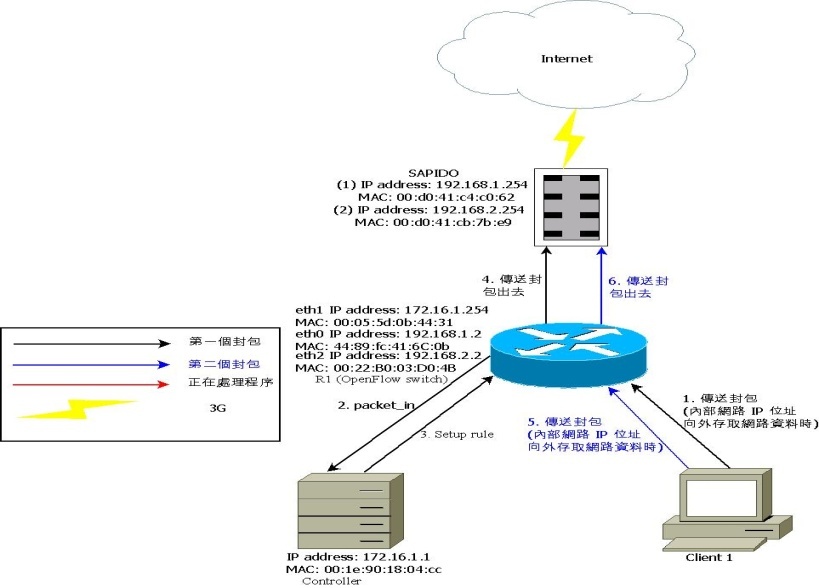


圖 3-23、NAT-步驟4

表格 3-9、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→SAPIDO |

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、。

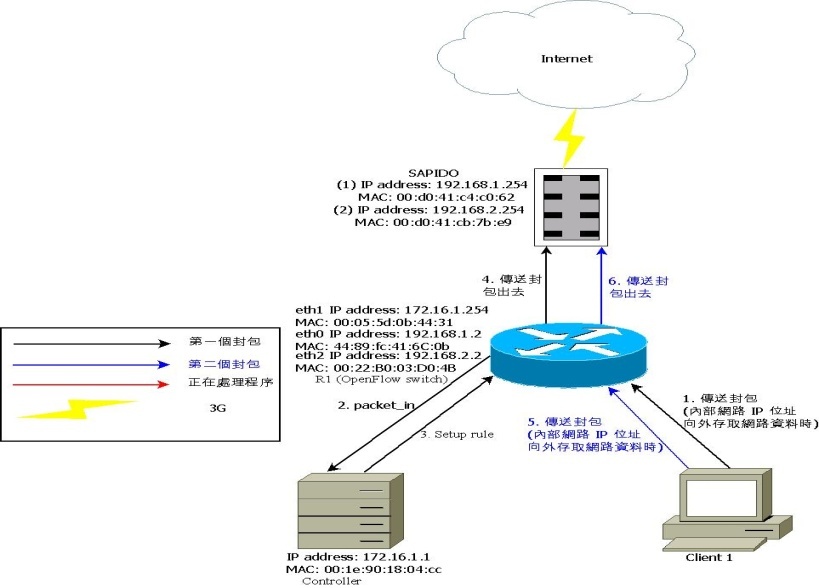


圖 3-24、NAT-步驟5

表格 3-10、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 |

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、。

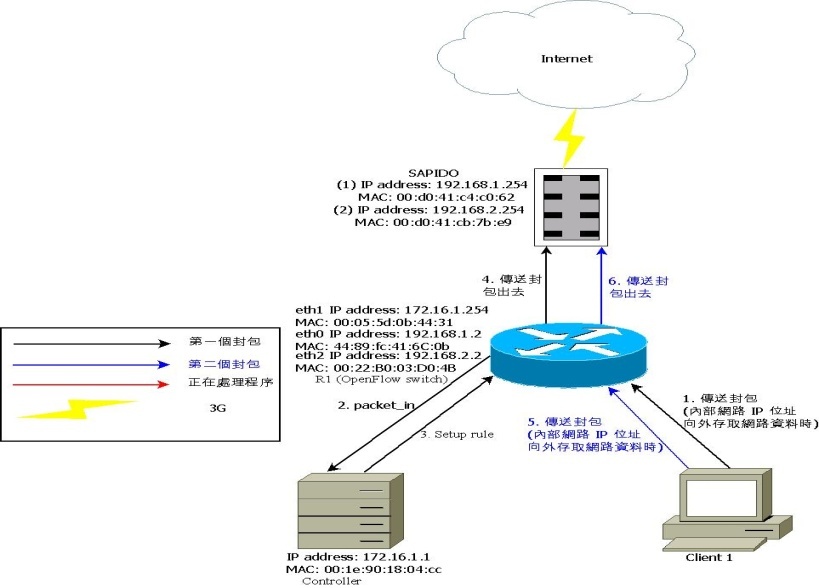


圖 3-25、NAT-步驟6

表格 3-11、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→SAPIDO |

在的步驟4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目前鏈路使用流量，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。

表格 3-12、利用 NAT 機制達成負載平衡之 flow table (R1)

|  |  |  |
| --- | --- | --- |
| **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 |

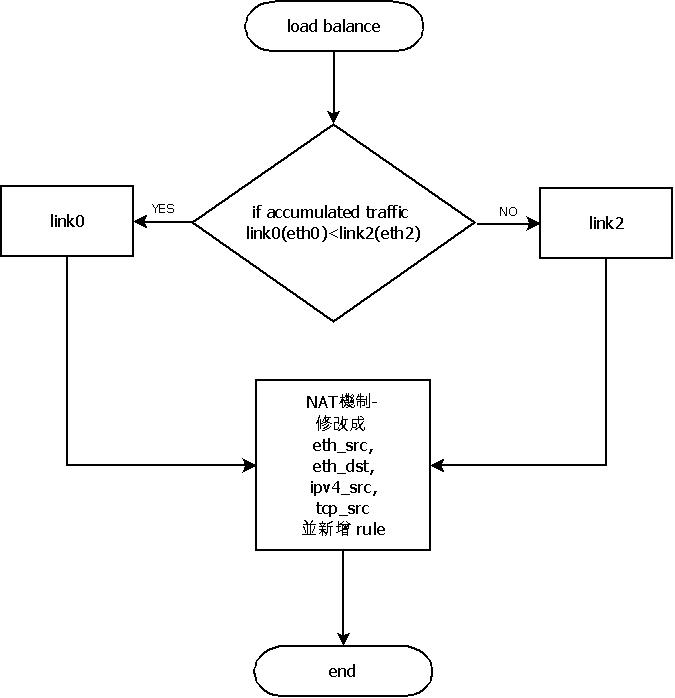


圖 3-26、R1(OpenFlow switch) load balance演算法

### 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) [25] ，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 172.16.1.1，目的地 IP 位址為 163.22.21.84，則使用 ICMP Redirect ，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、。

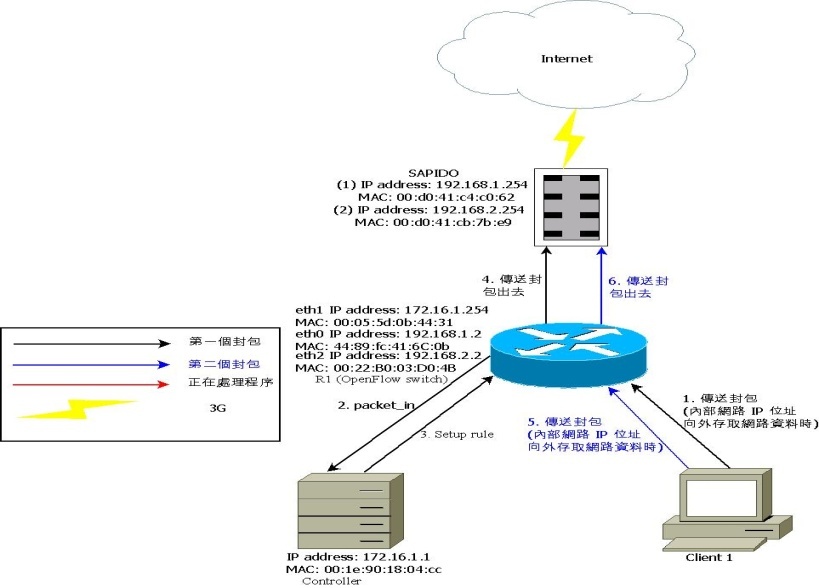


圖 3-27、ICMP-步驟1

表格 3-13、ICMP-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 |

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、。

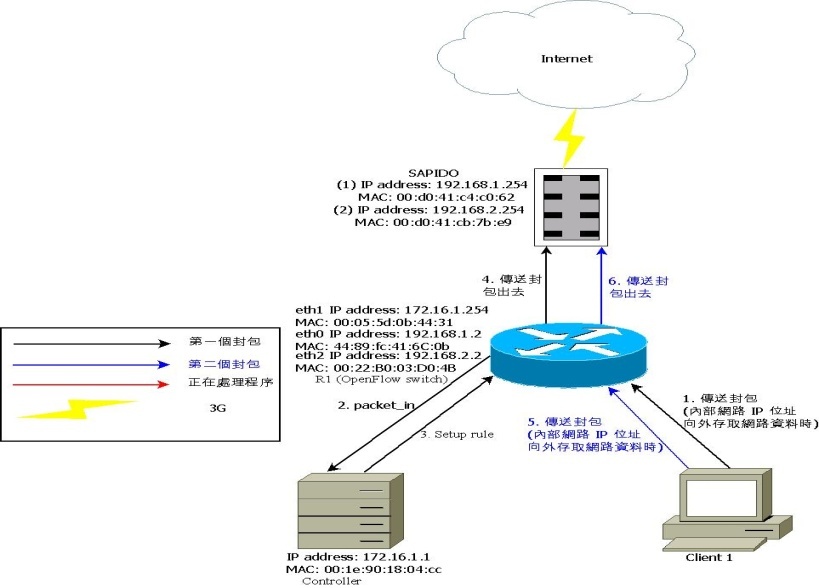


圖 3-28、ICMP-步驟2

表格 3-14、ICMP-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→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 ，寫入 R1(OpenFlow switch) flow entry (flow entry 如 藍色部分)中，如所示、、所示、。

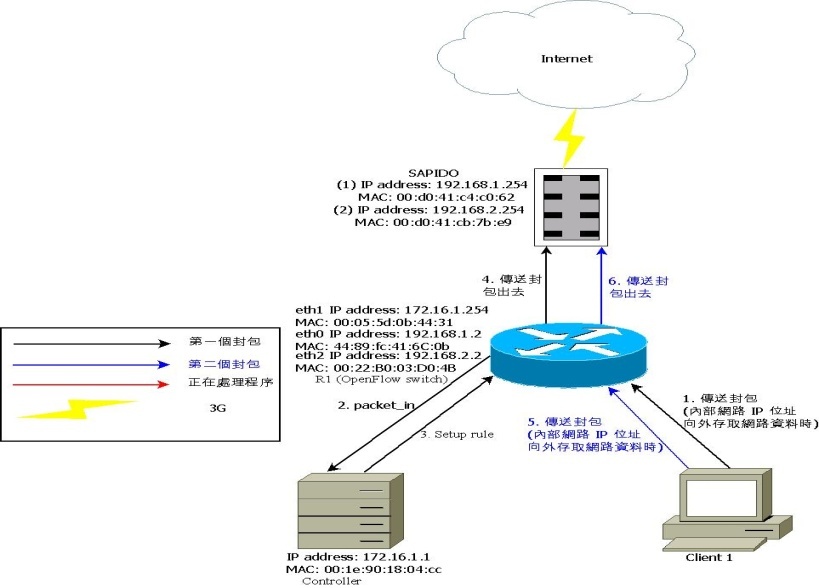


圖 3-29、ICMP-步驟3

表格 3-15、ICMP-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 | Controller→R1 |

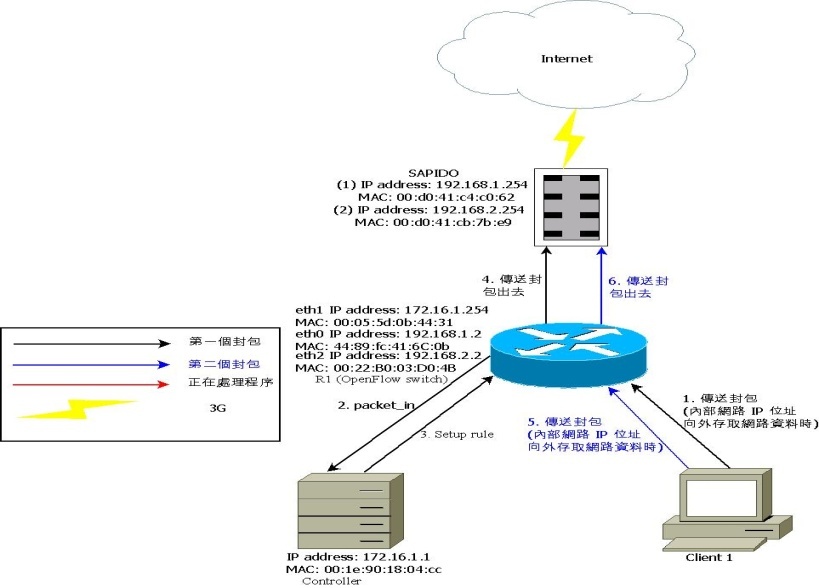


圖 3-30、ICMP-步驟4

表格 3-16、ICMP-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→SAPIDO |

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、。

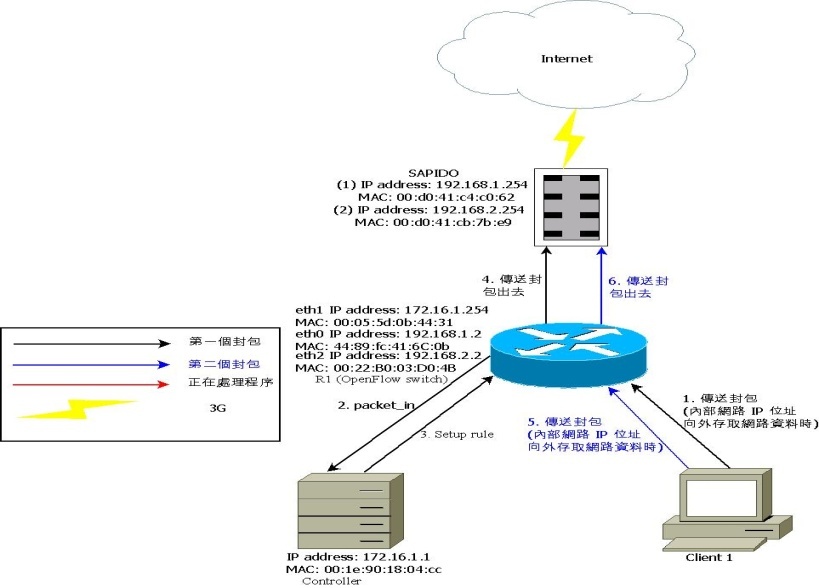


圖 3-31、ICMP-步驟5

表格 3-17、ICMP-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 |

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 192.168.1.2、 tcp\_src 的欄位修改成 42501、 Output 的欄位修改成 3，如所示、。

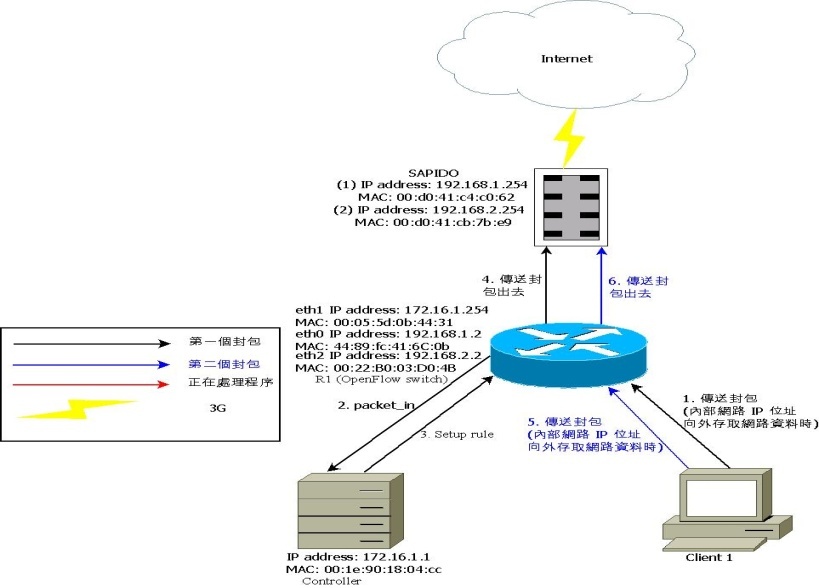


圖3-32、ICMP-步驟6

表格3-18、ICMP-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→SAPIDO |

在的步驟4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 172.16.1.1，如果是就使用 ICMP Redirect 修改預設路徑傳送封包；如果不是就使用預設路徑傳送封包。

表格 3-19、利用 ICMP Redirect 機制之 flow table (R1)

|  |  |  |
| --- | --- | --- |
| **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=49486 | 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=36360  Set Output:3 |

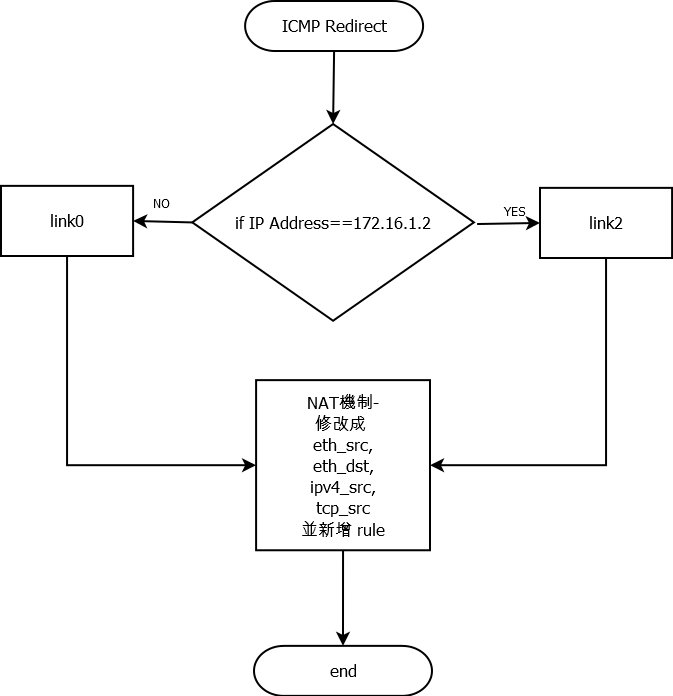


圖 3-33、R1(OpenFlow switch) ICMP Redirect演算法

# 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下：1. 每各鏈路使用狀況；2. 切換不同鏈路情況下封包遺失為多少。以下所示，為測試情境實驗環境架構圖。為實驗設備。

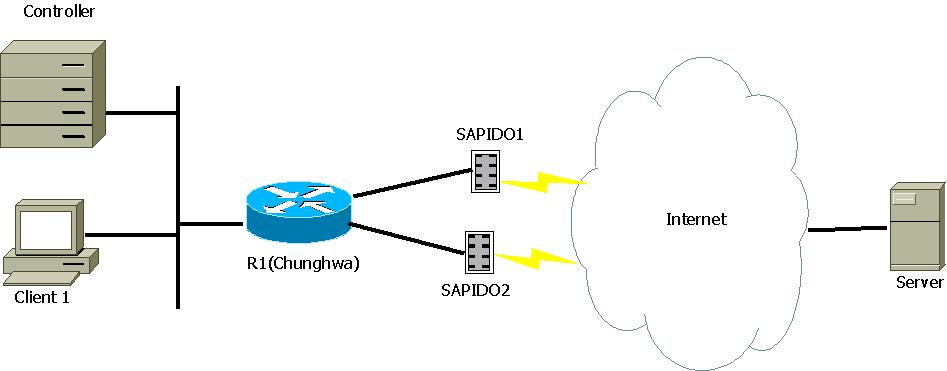


圖 4-34、實驗環境

表格 4-20、實驗設備

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 設備名稱詳細項目 | Controller | R1(Chunghwa) | SAPIDO | Client | Server |
| CPU | Intel(R) Pentium(R) 2.00GHz | Intel(R) Celeron(R) 2.40GHz |  | Intel(R) Pentium(R) 3.40GHz | Intel(R) Pentium(R) 2.00GHz |
| RAM | 512MB | 2GB |  | 1.5GB | 2GB |
| 網卡1 | Gigabit Ethernet | Gigabit Ethernet | 100Mbps Ethernet | Gigabit Ethernet | Gigabit Ethernet |
| 網卡2 |  | 100Mbps Ethernet |  |  |  |
| 網卡3 |  | 100Mbps Ethernet |  |  |  |

## 流量監控工具

本研究的流量監控工具使用 MRTG。MRTG(Multi Router Traffic Grapher)[26]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)[27]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 2.16.2 安裝基本套件

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

1. 設定與啟動 SNMP

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

-#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設定  
        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](mailto:3: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  
 [root@localhost ~]# env LANG=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg

1. 產生 MRTG 首頁與重啟 httpd

-執行  
 [root@localhost ~]# indexmaker /etc/mrtg/mrtg.cfg > /var/www/mrtg/index.html  
  [root@localhost ~]# service httpd restart

1. 設定排程

-編輯 /etc/crontab  
 [root@localhost ~]# 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

### 封包吞吐量 (Throughput) 實驗與結果

在這一小節，我們進行最低流量路徑的效能量測並分析結果。其實驗方式為 Client 使用 scp(Secure Copy) 指令方式傳送檔案給 Server ，這個指令是一個用於在 Linux 下進行遠端拷貝檔案的命令，類似於 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：終端設備在 Terminal 下，打上這一行指令 scp –P 22 XXX sandra@163.22.21.105:~/WWW/XXX ，-P為 port 指定資料傳輸用到的埠號，注意是大寫的P。封包資料大小為 XXKB ，所示，由 MRTG 流量工具將每個介面的使用流量顯示出詳細流量資料。

圖 4-35、使用 scp 指令測試的結果

由得知，X軸單位為時間；Y軸單位為Bytes，我們的方法結果

### 封包遺失實驗與結果

接下來這一小節，我們進行使用本實驗的方法之效能量測並分析結果。其實驗方式為 Client 使用 ping 指令，這個指令是一個電腦網路工具，能偵測雙方網路連線狀況。運作方式為：終端設備發送一個 ICMP echo request封包，等待接收端接收 ICMP echo reply的封包，如所示。

表格 4-21、使用 ping 測試的結果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Packet loss(%) | Time(Sec) | | | | |
| 1 | 5 | 9 | 13 | 17 |
|  |  |  |  |  |  |

# 結論與未來展望

# 參考文獻

1. ISP,[ <http://en.wikipedia.org/wiki/Internet_service_provider>]
2. Xtera,[http://www.xtera.com/zh-Hant/Home]
3. CISCO EIGRP,[ <http://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/16406-eigrp-toc.html>]
4. 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.
5. N, McKeown, T. Anderson, H, Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, ”OpenFlow: Enabling Innovation in Campus Networks” ,Mar. 14, 2008
6. Akella, B. Maggs, S. Seshan, A. Shaikh, and R.Sitaraman. A measurement-based analysis of multihoming. In Proc. of ACM SIGCOMM, August 2003.
7. Goldenberg, D.K., Qiuy, L., Xie, H., Yang, Y.R. and Zhang, Y. , “Optimizing Cost and Performance for Multihoming”, Proceedings of the 2004 ACM SIGCOMM Conference, pp. 79-92, USA, New York.
8. Internet Draft, J. Abley, B. Black, and V. Gill,” Goals for IPv6 Site-Multihoming Architectures”, June 13, 2003.
9. A.S. Sairam, G. Barua, “Load Balancing Inbound Trafﬁc in Multihomed Stub Autonomous Systems,” Jan. 2009.
10. A.S. Sairam, G. Barua, “Distributed Route Control Schemes to Load Balance Incoming Trafﬁc in Multihomed Stub Networks,” Jan. 2010.
11. H. Long, Y. Shen, M. Guo, F. Tang, “LABERIO: Dynamic load-balanced routing in OpenFlow-enabled networks,” Mar. 2013.
12. Open vSwitch, [ http://openvswitch.org/ ]
13. XEN, [http://www.xenproject.org/ ]
14. KVM, [http://www.linux-kvm.org/page/Main\_Page ]
15. VirtualBox, [https://www.virtualbox.org/ ]
16. Ryu, [ <http://osrg.github.io/ryu/> ]
17. SAPIDO 韌體, [<http://www.sapido.com.tw/CH/data/Download/mb1112_f.htm>]
18. SAPIDO 使用手冊, [<http://www.sapido.com.tw/CH/service01.htm>]
19. OpenFlow Switch Speciﬁcation Version 1.0.0, [https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.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. 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 ]
24. P. Srisuresh, M. Holdrege,”IP Network Address Translator (NAT) Terminology and Considerations”, RFC 2633, Aug. 1999.
25. J. Postel,”Internet Control Message Protocol”, RFC 792, Sep. 1981
26. MRTG,[http://oss.oetiker.ch/mrtg/]
27. J. Case, M. Fedor, M. Schoffstall, J. Davin,”A Simple Network Management Protocol (SNMP),” RFC 1157, May. 1990
28. C. Hedrick,”Routing Information Protocol”, RFC 1058, June. 1988.