軟體定義網路(Software-defined Network)之實作

林傳筆
朝陽科技大學
資訊與通訊系
cblin@cyut.edu.tw

盧皓威
朝陽科技大學
資訊與通訊系
s10230601@cyut.edu.tw

陳堃維
朝陽科技大學
資訊與通訊系
s10130606@cyut.edu.tw

王永祥
朝陽科技大學
資訊與通訊系
s10230603@cyut.edu.tw

摘要

網際網路與雲端運算的快速發展，提供大量且多樣的服務，為了處理大量服務的要求，需要建立一個資料中心 (Data Center) 來處理海量的資料，而資料中心是由一群的伺服器、交換器和路由器等網路設備所組成，這些設備與伺服器是經由 TCP/IP 網路協定來進行訊息的溝通。然而，隨著資料量增加而擴充伺服器與網路設備，當設備增加或改變時必須去更新或管理網路傳輸協定的設定，而軟體定義網路(Software Defined Networking，SDN)提出新的方式來解決上述內部網路的問題。

本研究提出建置一個 OpenFlow 協定的軟體定義網路平台，軟體定義網路有兩個架構：控制層與資料層。控制層是以電腦或伺服器來當 Controller，Controller 藉由 OpenFlow 協定來與 OpenFlow 交換器或虛擬交換器進行溝通，而資料層是交換器與交換器的溝通，因而提升網路的效能。

關鍵詞：軟體定義網路、OpenFlow 協定、虛擬交換器、內部網路、Controller

Abstract

To deal with a large amount of services, a Data Center is needed to build. The Data Center includes a group of servers, a number of switches, and some routers to handle those data. The network equipment uses the TCP/IP protocol to communicate each other. However, while the network equipment is increased for handling the big data, it is inconvenient and complicated to adjust or reset the protocol for an intranet. Therefore, Software-defined Network (SDN) is proposed to solve the above problems and makes the networks flexible.

The general SDN consists of two tiers: control plane tier and data plane tier. There is a big network controller used to communicate with the switches or virtual switches (vSwitch) by OpenFlow protocol in the control plane tier. The communications, in the data plane tier, are used between switches with switches or computers. Because of the two tiers, the efficiency of networks is enhanced.

Keywords: Software-defined Network (SDN), OpenFlow, Open Virtual Switch (Open vSwitch), Intranet, Controller

1. 前言

現今人們生活和網路息息相關，如：搜尋引擎、社群網路、雲端運算等。最新的網路應用服務，為了處理大量服務的要求，需要建立一個資料中心 (Data Center) 來處理海量的資料，而資料中心是由一群的伺服器、交換器和路由器等網路設備所組成。然而，隨著資料量增加時，需要擴充伺服器與網路設備，當設備擴充或改變時必須去更新管理網路傳輸協定的設定，因此利用網路技術來達到降低管理網路的複雜度及改善網路效能將是一個重要的議題。

因此軟體定義網路(Software Defined Networking，SDN)[1] 是被提出來，因為它能快速建置、可自行定義網路架構及傳輸協定等特色，以達到降低管理網路複雜度及提升網路頻寬的使用，而開放網路基金會(Open Networking Foundation，ONF)[2]正在積極的推動 SDN 網路架構、定義及管理 OpenFlow[3][4] 的標準協定，因此本研究將自行定義 SND 網路架構、OpenFlow 的標準協定及設置與管理，以解決實驗室與機房等內部網路所面臨到網路硬體設備汰換時的繞送設定與配置問題，並分析 SDN 網路架構與傳統網路架構的差異性。

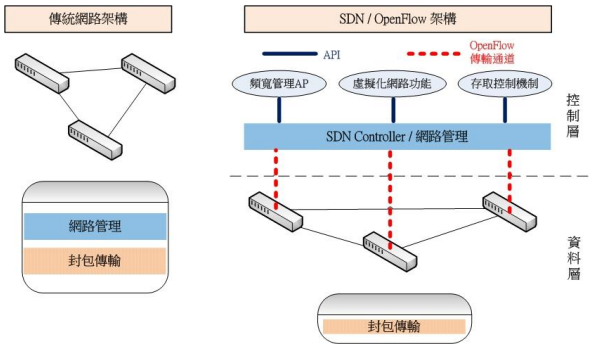
2. 相關技術

本節將針對軟體定義網路的 OpenFlow 協定、OpenFlow 交換器與 Controller 進行探討，另外也將探討使用於Open Virtual Switch 中的軟體。

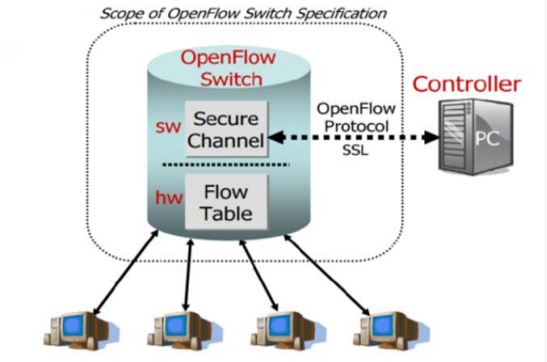
2.1 OpenFlow

SDN 軟體定義網路由美國史丹佛大學發

表，創新了一種網路架構以改善傳統 TCP/IP 的網路架構，來解決現有網路技術的瓶頸，如圖一所示。在***OpenFlow Enabling Innovation in Campus Networks***[4]中提到 OpenFlow 協定的運作方法，將傳統路由器或交換器的網路運作方式分離成控制(Control Plane)與資料平面(Data Plane)[1][5]，將原本舊有網路傳輸架構、封包處理路徑由 OpenFlow 交換器和 Controller 來完成，如圖二所示，控制平面使用單獨的 Controller 負責封包繞送的動作，而 OpenFlow 交換器透過安全通道(Secure Channel)與 Controller 進行溝通處理，Controller 再透過 OpenFlow Protocol 來控制 OpenFlow 交換器的 flow table 進行資料流轉發(forwarding)行為[6]。



圖一：SDN 架構與傳統網路架構的差異



圖二：OpenFlow 交換器結構[4]

因此 SDN 與傳統網路相較下，具備了下列優勢:

1. 網路管理和配置簡易化
2. 快速排除故障
3. 提升網路頻寬使用率
4. 更符合實際的測試、驗證、故障模擬
5. 網路效能和系統配置更具彈性
6. 有更多靈活性不受傳統網路設備所束縛

2.2 OpenFlow 交換器

OpenFlow 交換器[4]是在一般電腦上安裝 Linux 作業系統且使用 NetFPGA[7][8] PCI 介面卡，並結合 OpenFlow 協定能使研究人員測試自己設計的協定或政策，利用 flow tables 去新增或刪除 entry，來管理整個網路。

OpenFlow 交換器包含三個主要部分:

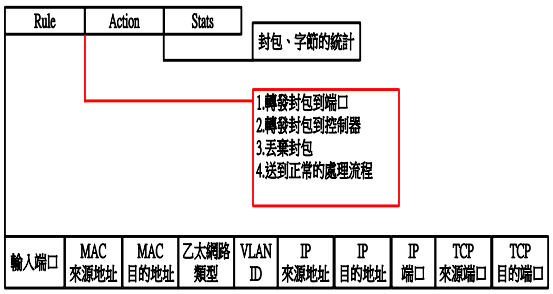
1. 路由表(Flow Table):用來定義網路封包傳輸路徑，每個 flow entry 相關動作告知交換器如何處理流量，如圖三所示。
2. 安全通道(Secure Channel)：連結 Switch 和遠端的 Controller，用來下達命令或是傳送封包。
3. OpenFlow 協定（OpenFlow Protocol）：作為傳輸溝通用，提供一個開放和標準的方式給 Controller 與 OpenFlow 交換器進行溝通。

In	VLAN	Ethernet			IP			TCP	
Port	ID	SA	DA	Type	SA	DA	Proto	Src	Dst

圖三：OpenFlow 路由表[4]

在 Flow Table 之中，每一個 entry 有三個區域：

1. 封包標頭(Packet Header)：定義了每一個 flow。
2. 動作(Action)：定義如何對每一個 flow 進行處理的動作，如圖四。
3. 統計(Statistics)：對於每一個 flow 去計算有多少 bytes 和 packets 符合，且當中的 time 是計算從最後一次有符合此 flow 的目前時間差距，可以幫助移除無用的 flow。



圖四：OpenFlow 路由表動作

OpenFlow 交換器的規範能更適合應用與多變的網路需求，高性能和降低營運成本與管理複雜度。

2.3 OpenFlow 控制器

OpenFlow 的 Controller 是 OpenFlow 的架構關鍵元件，在 Controller 中網路操作系統（network operating system，簡稱 NOS）實現控制邏輯功能。

Controller 如何應實現 OpenFlow 的協定並

沒有規定，各種不同的實現已被開發，每個都有其自己的行為和性能特性。如：NOX[9]、Maestro[10]、Beacon[11]、Floodlight[12]等。這些不同的 Controller，在不同情況下其特性也許比其它的 Controller 更適合。因此要選出適合的 Controller 必須分析其網路架構，其特性與差異如下列所示。

- NOX 控制器
  開發語言：Python/C++
  平台：Linux
  開發團隊：Nicira
  特色：最早實現的控制器，單執行緒操作，版本不斷更新，性能還在繼續完善。
- Maestro 控制器
  開發語言：Java
  平台：Win/Mac/Linux
  開發團隊：Rice
  特色：跨平台，易於開發和佈署，支持多執行緒操作，功能豐富，可以加入新的應用程序進行擴展。
- Beacon 控制器
  開發語言：Java
  平台：Win/Mac/Linux
  開發團隊：Stanford
  特色：跨平台，易於開發和佈署，採用模組化功能實現了基於事件和多執行緒的處理平台，可自定義擴展的介面框架。
- Floodlight 控制器
  開發語言：Java
  平台：Win/Mac/Linux
  開發團隊：Big switch
  特色：跨平台，基於 Beacon 開發，開放原始碼進行維護，遵循 Apache 開放原始碼規範，適合推廣。

2.4 OpenFlow 與現有技術的差異之處

OpenFlow 能大幅革新現有網路限制，即是由於在各方面都改善並突破現有技術的規範。簡單的將 OpenFlow 與現行技術的差異如下列所示，便於快速瞭解 OpenFlow 與現有技術的差異之處。

- 現行技術
  網路協定：box centric，以單一硬體為中心。
  網路管理：優化單一硬體效能，每個硬體擁有各自硬體效能的系統監控/管理/操作功能，並不互相配合及協調。
  測試環境：無法將原有網路系統分割，若欲進行網路測試和實驗，需另接網路以避免網路癱瘓。
  成本控制：點對點配對的複雜性也隨著全平台通訊而提升，因此需要性能更強的設施以應對網路通訊需求。
- OpenFlow 技術
  網路協定：fabric centric，將 WAN 以整體光纖為概念優化，而非視為多個硬體的集合。
  網路管理：將系統監控/管理/操作，從各別硬體切割出來，並集中控制網路傳輸，以整體網路傳輸做優化。且能根據必要的協定需求(protocol requirements) 選擇使用的硬體及軟體。
  測試環境：能以軟體劃分網路(software-defined networking) 並進行實驗，即使實驗線路故障仍不影響原有網路。
  成本控制：軟體升級不需再在硬體上進行，更節省人力成本，有效為企業降低單位傳遞成本。

2.5 Open Virtual Switch(Open vSwitch)軟體

Open vSwitch 是一個 Apache2.0 授權下的開放原始碼軟體，提供了多層虛擬交換器功能，目的是通過編程擴展，讓大規模的網路自動化且支持多種標準的管理介面和協定，好方便管理和配置虛擬機器的網路，支援多種虛擬化平台。Open vSwitch 透過 Kernel module 核心模組來處理封包的轉送，也提供命令列的管理介面。Open vSwitch 的特點[13]為:

1. 流量監測: 支援 Netflow 與 sflow。
2. 提升安全性與隔離性:VLAN，封包過濾
3. 負載均衡，支援 open flow 網路協定。
4. 提高 QoS 管理:為每一個虛擬機器端口進行流量管理。

3. 系統架構

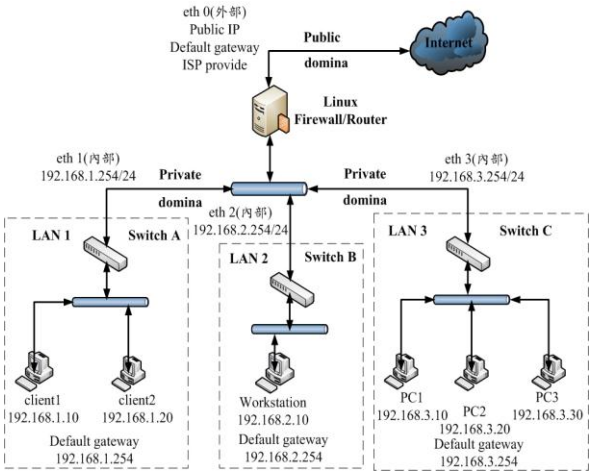
本研究為了要實現 OpenFlow 的想法及路由的觀念，建置實體區域網路架構以及虛擬機器佈署虛擬伺服器主機叢集，讓研究人員可自行定義 SND 網路架構、OpenFlow 的標準協定及設置與管理，並分析 SDN 虛擬叢集架構與傳統網路架構的差異性。

3.1 實體區域網路架構

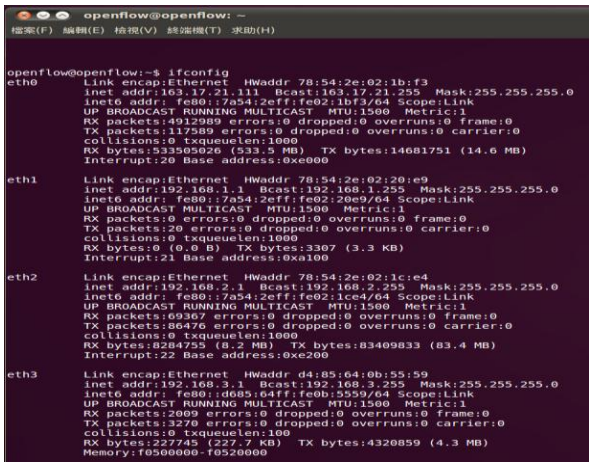
本研究所建置實體區域網路架構如圖五所

示，將一台擁有四張 PCI 網路介面卡的 Linux Firewall/Router 實體主機，模擬為 OpenFlow 中的 Controller 如圖六，在 Controller 底下連接著三台實體的 switch，每台 switch 底下則連接實體主機。Controller 的第一張網路卡(eth0)連接至外部網路專門處理內部與外部封包的傳遞；第二張網路卡(eth1) 連接 switch A 形成一個區域網路 LAN1，專門處理 LAN 1 內部要傳遞至外部的封包，為了安全性考量皆以虛擬 IP 方式展現，而 LAN2、LAN3 架構以此類推。

透過此方式，將需要一台實體機器作為 Controller 指揮 IP 與封包傳輸規則，在此將 Controller 當作 gateway 並利用 Linux route add 指令增加路由表的規則，然而在 Controller 中完成定義封包傳輸規則，以及開啟 ip forward 功能達到傳遞實體主機的封包後，每個子網域要進行封包傳遞時都會經過 Controller 所定義的規則來決定封包的傳輸路徑，透過以定義的封包傳輸規則且經由網路測試後確定每個子網路能夠進行封包傳遞。



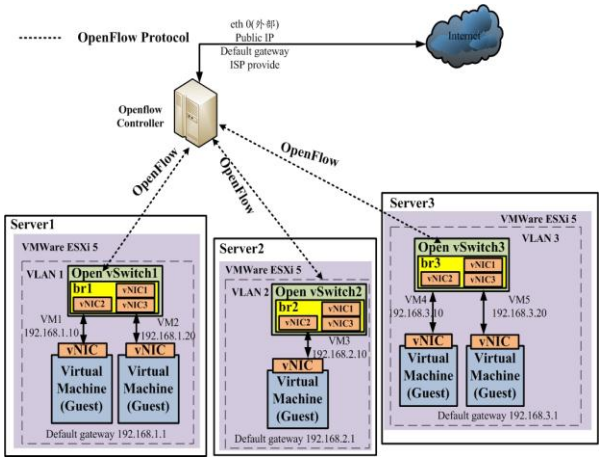
圖五：實體區域網路架構圖



圖六：模擬為 Controller 電腦的網路介面卡

3.2 虛擬化技術建置 SDN 網路架構

本研究由於目前無 Open Switch 和 NetFPGA 等硬體設備，因此利用 Open vSwitch 自由軟體來建置 SDN 網路環境架構如圖七，進行定義及配置 OpenFlow 協定與實務操作，首先建置一台實體主機來模擬為 Controller 且利用 OpenFlow protocol 與 Open vSwitch 進行溝通，再使用 VMware ESXi 虛擬化技術與三台伺服器(Server1、Server2 和 Server3)來建置虛擬的區域網路，且虛擬區域網路又分成三個子網域；分為 (VLAN1、VLAN2 和 VLAN3)，每個子網域內會將有一台虛擬機器來模擬 Open vSwitch。



圖七：利用虛擬化技術建置 SDN 網路架構

然而，Open vSwitch 運行原理則是本身核心實現了多個資料路徑(類似於橋接器)，每個資料路徑都有多個 vports(類似於橋接器內的端口)，每個資料路徑透過 flow table 來進行操作與設置。在本研究建置 SDN 環境中，每台 Open vSwitch 上有三張虛擬網路卡(vNIC)，以圖七的 Server1 來舉例，建立一個虛擬區域網路 VLAN1，而在 Open vSwitch1 中將 vNIC1(eth1) 設定為 Open vSwitch Bridge(此處命名為 br1)，將 vNIC2(eth2)和 vNIC3(eth3)設定為 vports 並加入至 br1 橋接器中如圖八，讓虛擬機器能夠透過 vports 進行資料傳輸。

圖八：Open vSwitch 的 Bridge 設定

4. 實驗和分析

因為目前無 Open Switch 和 NetFPGA 等硬體設備，本實驗就是藉由 Open vSwitch 的建置來探討與了解 OpenFlow 傳輸協定，來做為日後建置 SDN 網路能自行制訂一個屬於自己實驗室或系的內部網路協定。因此，本研究透過 OpenFlow protocol 測試了封包在內部網路的傳輸與流量來做測試。

當封包到達 Open vSwitch 時，OpenFlow 在處理封包會根據 flow table 內的標頭進行規則(rule)比對並尋找匹配的動作，如果找不到相對應的動作會將此封包的標頭(header)送至 Controller 進行尋找路徑的動作，再將此路由資訊利用 OpenFlow Protocol 與 Open vSwitch 儲存至 flow table；反之當到達的封包在 Open vSwitch 的 flow table 中找到相對應的動作，將直接使用正常處理的 pipeline 來轉傳封包。

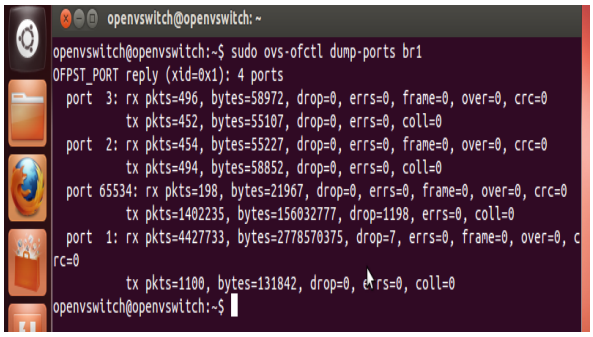
接著分析封包繞送的規則與流量，在 SDN 環境架構中將每個 VLAN 的 Open vSwitch Bridge 進行 flow 定義，在 Bridge br1 上有三個 ports 如圖九中的 1 (eth1)、2 (eth2) 、3 (eth3)，而在本研究將測試此 flow 由 br1 第二個 port 進來的封包全部丟棄、圖十所示。最後在本研究的 SDN 環境架構中，將設定讓所有 VLAN 之間的虛擬機器能夠互相傳遞封包，並在虛擬機器間模擬出封包的傳輸且將封包傳輸量紀錄起來如圖十一所示。

```
openvswitch@openvswitch:~$ sudo ovs-vsctl set bridge br1 datapa
openvswitch@openvswitch:~$ sudo ovs-ofctl show br1
OFPT_FEATURES_REPLY (xid=0x1): ver:0x1, dpid:0000000c29661c2b
n_tables:255, n_buffers:256
features: capabilities:0xc7, actions:0xfff
 1(eth1): addr:00:0c:29:66:1c:2b
     config:     0
     state:      0
     current:    10GB-FD COPPER
     advertised: COPPER
     supported:  1GB-FD 10GB-FD COPPER
 2(eth2): addr:00:0c:29:66:1c:35
     config:     0
     state:      0
     current:    10GB-FD COPPER
     advertised: COPPER
     supported:  1GB-FD 10GB-FD COPPER
 3(eth3): addr:00:0c:29:66:1c:3f
     config:     0
     state:      0
     current:    10GB-FD COPPER
     advertised: COPPER
     supported:  1GB-FD 10GB-FD COPPER
 LOCAL(br1): addr:00:0c:29:66:1c:2b
```

圖九、在 Bridge br1 中增加 flow 規則

```
openvswitch@openvswitch:~$ sudo ovs-ofctl add-flow br1 idle_timeout=120,in_port=
2,actions=drop
openvswitch@openvswitch:~$ sudo ovs-ofctl dump-flows br1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=3.928s, table=0, n_packets=0, n_bytes=0, idle_timeout=120,
in_port=2 actions=drop
 cookie=0x0, duration=181225.149s, table=0, n_packets=4436903, n_bytes=278070836
7, priority=0 actions=NORMAL
openvswitch@openvswitch:~$
```

圖十、在 Bridge br1 中第二個 port 進來的封包全部丟棄



圖十一：每個 port 封包傳輸量

5. 結論與未來展望

傳統網路資料傳送的傳輸路徑 (Data Path) 皆由路由器 (Router) 決定，所以造成路由器自行指定傳輸路徑，在舊有傳輸路徑失敗的狀況下，將會造成多餘的重複傳輸，導致系統資源的浪費；且當使用者希望更新傳輸方式時，將需由專業人員手動更新路由器內軟體，造成維運成本上升。然而 OpenFlow 將傳輸路徑的規劃交給控制器 (OpenFlow controller)，控制器能依使用者需求給出最佳化傳輸路徑，以節省成本。

未來可加入更多種服務，套用於不同的負載平衡機制，或者是使用不同種類的控制器，如 Floodlight、Maestro、Beacon controller 等，實現更豐富操作(例如：吞吐量限制、負載均衡、服務品質)來反映網路策略，以降低網路管理的複雜性。

參考文獻

[1] Software-Defined Networking: The New Norm for Network. ONF White Paper, https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[2] Open Networking Foundation, https://www.opennetworking.org/index.php

[3] Open Networking Foundation, ***OpenFlow switch specification***, version 1.4.0, Oct 14, 2013.

[4] McKeown, Nick, et al. "***OpenFlow: enabling innovation in campus networks.***" ACM SIGCOMM Computer Communication Review 38.2 (2008): 69-74

[5] Koponen T, Casado M, Gude N, Stribling J, Poutievski L, Zhu M, Ramanathan R, Iwata Y, Inoue H, Hama T, Shenker S. ***Onix: A distributed control platform for large-scale production networks***. In: Proc. of the 9th USENIX Conf. on Operating Systems

Designand Implementation (OSDI). Vancouver: USENIX Association, 2010. http://dl.acm.org/citation.cfm?id=1924968

[6] 陳俊良, "***SDN 之技術標準與發展現況***, " 網路通訊國家型科技計畫

[7] NetFPGA official site, http://netfpga.org/

[8] Greg Watson, Nick McKeown and Martin Casado, "***NetFPGA: A Tool for Network Research and Education***," Department of Electrical Engineering Stanford University

[9] Gude N, Koponen T, Pettit J, Pfaff B, Casado M, McKeown N, Shenker S. ***Nox: Towards an operating system for networks.*** ACM SIGCOMM Computer Communication Review, 2008,38(3):105−110. [doi: 10.1145/1384609.1384625]

[10] Cai Z, Dinu F, Zheng J, Cox AL, Eugene TS. ***The preliminary design and implementation of the maestro network control platform***. Technical Report, Department of Computer Science, Rice University, 2008. http://www.cs.rice.edu/~eugeneng/papers/Maestro-TR.pdf

[11] Beacon. 2012. http://www.beaconcontroller.net

[12] Floodlight. 2012. http://floodlight.openflowhub.org

[13] 廖俊傑, "***KVM 虛擬叢集的效能分析***," 2013 Conference on Information Technology and Applications in Outlying Islands, pp. 48-55.