以 SDN 連結 L3 網域及南北向介面開發

黃文源　　李慧蘭　　周大源　　劉德隆
財團法人國家實驗研究院國家高速網路與計算中心
{wunyuan, gracelee, 1203053, tlliu}@ narlabs.org.tw

摘要

本論文展示針對 SDN-IP 實作的網路介面系統。這套 SDN-IP 網路介面系統是以 SDN 中 ONOS 和其應用程式 SDN-IP 為核心。ONOS 與 SDN-IP 的組合能夠連結兩個不同的異質網域，使這兩個異質網域的用戶能夠透過 SDN 網路當中繼互相傳輸資料，然而 SDN-IP 於資訊的呈現上是不友善的，因此我們以 ONOS 的 Rest API 為基礎再搭配網頁伺服器與網頁語言實作一套 SDN-IP 網頁介面系統。從系統的介面上可以清晰的得知 SDN-IP 內儲存的與路由相關的資訊，有益於網路管理員維護網路。

關鍵詞：SDN-IP、ONOS、軟體定義網路、SDN

1. 前言

近十年網際網路的蓬勃發展，有許多的應用相繼出現，其中雲端運算與大數據傳輸是近年來資訊與網路科技的重大熱門。前者利用許多虛擬化的技術，將許多伺服器的計算能量重新規劃，讓原有的機器運算能量得以提昇。針對全球性雲端供應商而言，為提供更有效的企業級雲端服務，紛紛購置更大量的運算用伺服器，並考量成本低廉、管理便捷、政經情勢穩定等等因素，陸陸續續將更多資料中心 (Data center) 的進駐於世界各地。

然而，為了要能夠管理數量日漸擴增的雲端伺服器，若是雲端服務提供者必須透過現有網際網路通訊協定，勢必會有 IP 位址不足、管理不易等等問題。故新的網路型態與架構的研究已經是一個非常重要的研究課題。

軟體定義式網路(Software Defined Network；SDN)[1]是這些課題中的一個，且此技術也已經受到許多雲端服務提供者所採用，例如：Google、Amazon。SDN 與傳統網路的差別在於，其控制層(Control Plane)和資料層(Data Plane)是分開的，不像傳統網路將此兩層整合於設備之中，故使用者可以藉由 SDN 的可程式化開放架構的特性，於獨立的控制平台上新增與開發所需功能以滿足客製化網路需求、提高頻寬使用率和減少成本[2]等。

隨著雲端技術與 SDN 技術的發展，採用 SDN 環境的雲端技術其網域佈署也日漸廣泛，而利用 SDN 技術建置的網路勢必會面臨與傳統網路的介接，當兩個異質性網域介接時，一定會面臨封包在不同網路間路徑決策、異種網路路由資訊交換等問題。對於這些問題，有一個解決方法是使用 ONOS[3][4]來充當 SDN 的控制平台，ONOS 內有專門的應用程式可以解決這些問題。

ONOS 為 SDN 的其中一套控制平台軟體，類似的控制平台有 Floodlight[5]、OpenDaylight (ODL)[6]與 Ryu[7]。ONOS 包含著一些應用程式，例如：SDN-IP[8]、CORD(Central Office Re-architected as a datacenter)[9]、SONA(Simplified Overlay Network Architecture)[10]、Virtual Private LAN Service (VPLS)[11]等，其中 SDN-IP 就是前述所提的解決異質性網域介接路由問題的解決方案，此應用程式可接收其他網域送出來的 BGP[12]路由資訊，並告知 ONOS，故 ONOS 會利用這些資訊來決定如何將封包送到指定的地點，當要送往其他網域的封包進入 ONOS 管轄的網域時，SDN 交換器會根據 Flow Table 的紀錄來轉送至正確的出口，利用此方法 ONOS 管控的 SDN 網域便可串聯多個異質性網域。

由於 SDN-IP 是 ONOS 的應用程式，所以使用者要查詢與 SDN-IP 相關的資訊，例如：Bgp Routes，就得利用 ONOS 本身的 Command Line 來輸入指令獲取所需資訊，然而這方法較不友善且也較不便利的，故我們針對 SDN-IP 的資訊獲取方面開發一個 WEB UI 系統，如此一來使用者將可簡單的獲得 SDN-IP 的資訊，減少許多資訊收集的時間與人為的失誤。

在本篇論文的第二節將介紹研究背景及相關的技術，包含 SDN、ONOS 與 SDN-IP，第三節描述 WEB UI 資訊顯示系統的實作，第四節為所開發系統之實驗與測試結果。

2. 相關知識

利用 SDN 技術來改變網路架構和管理機制是新型態雲端資料中心之網路佈建的趨勢，異質性網域之間如何傳輸封包是一個必要解決的問題，了解解決此問題的技術也是必要的，故此節將描述 SDN、ONOS 與 SDN-IP 相關技術的知識。

2.1 SDN

SDN 網路最常聽到的就是 OpenFlow[13]，OpenFlow 是由 Stanford 大學的一項計畫提出新的網路設備溝通協定，此計畫就是將原本控制層自網

路設備中抽離至另外一部伺服器中，稱為控制器(Controller)這也就是 SDN 上所謂的控制層，而原本傳統網路設備的資料平台則是繼續保留在實體交換器之中，在 SDN 中被稱為資料層，在資料層中的交換器裡面會包含著 Flow Table，Flow Table 裡面的資料又稱為 Flow Entry，而這兩層間則是透過 TCP 或 SSL 連線利用 OpenFlow 協定來交換訊息。

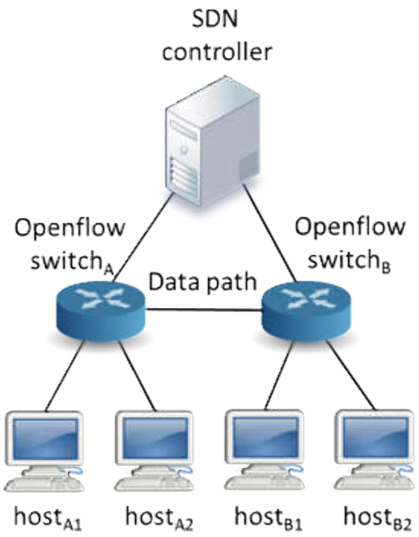

圖 1 Software Defined Networking 概念圖[14]

SDN 的概念如圖 1[14]所示。一部 SDN 的控制器(Controller)會控制兩部 SDN 交換器，而這兩部 SDN 交換器各自又有兩台用戶連接，switch$_A$ 與 switch$_B$ 間的資料路徑可以用實體線路或是 VPLS [15]達成。假設圖中的 host$_{A1}$ 要送資料給 host$_{B1}$ 時，則由 host$_{A1}$ 先向 switch$_A$ 發出第一個封包。而 switch$_A$ 收到後會先比對自己的 Flow Table，當沒有符合的 Flow entry，則會將此封包轉送至控制器，然後控制器會查看自己的拓樸，確認 host$_{B1}$ 的位置，依據最小路徑演算法，決定一條 host$_{A1}$→switch$_A$→switch$_B$→ host$_{B1}$ 的傳輸路線，緊接著便傳送設定命令至 switch$_A$ 和 switch$_B$ 來設定 Flow entry，設定完畢後將這個封包回傳回 switch$_A$，switch$_A$ 再依據 Flow entry 轉送所有的封包至下一個交換器，如此一來所有的封包將能順利送至 host$_{B1}$。

2.2 ONOS

隨著 SDN 的快速發展，電信商和雲端服務供應商也相繼採用 SDN 技術部署。ONOS(Open Network Operating System)是一開放原始碼 SDN 網路控制平台軟體，其主要的對象是服務供應商與建置 SDN 網路的各大企業與組織。ONOS 比傳統 SDN 控制器，例如：POX[16]、NOX[17]、Beacon[18]和 Floodlight 等，多了可擴充性(Scalability)，高可用性(High Availability)和高效能(High Performance)的特性。本身除了是電信專業等級的 SDN 控制平台外，也符合了當今資料中心虛擬化和雲端運算之基礎架構。

ONOS 是使用分散式叢集架構，在這架構下，若一台 ONOS 故障失效時，ONOS 會快速的轉移至其他它正常的系統，除此之外，每一個 ONOS 有著網路某部分的資料，故每個 ONOS 會有所屬的任務，因此在大規模的 SDN 網路環境下，此架構不止可以加強控制層的容錯率外也能加強網路工作的處理能力。

此外，ONOS 的北向介接介面有支援 Rest API[19]以及 Intent[20]處理機制，使用者可以開發應用程式透過這兩大介面來與 ONOS 交換訊息，例如：利用 Rest API 增加減少設備或是取得拓樸資訊，或是建立 Intent 應用程式請 ONOS 幫忙跟 SDN 交換器建立 Flow entry。因為此兩大介面的支援，使得開發變得更加簡單。在南向向介接介面上，可以透過 OpenFlow 和 NetConf[21]協定等管理 OpenFlow 設備、白牌交換機(White-Box Switches)和傳統網路設備。

2.3 SDN-IP

SDN-IP 是 ONOS 應用程式中的一個，他使用 Border Gateway Protocol(BGP)讓 ONOS 管控的 SDN 網路能夠與傳統網路順利介接與運作。從 BGP 的視角來看，ONOS 被視為一個 Autonomous System(AS)，其運作行為也如同其他傳統的 AS 一樣，另外 SDN-IP 也會運如同一個正規的 BGP Speaker，會接收 BGP 訊息，之後將這些訊息解析，然後透過 ONOS 的的 Intent 系統，請 ONOS 於 SDN 交換器上新增所需的 Flow entry，如此一來 SDN 交換器才懂得如何轉傳其他傳統網路傳送進來的封包。

ONOS 中 SDN-IP 使用了兩種型態的 Intent，分別是 Single-point to single-point intents 與 Multi-point to single-point intents。前者主要是讓 SDN-IP 與外部路由器建立單向性的連結，透過這些連結外部路由器的 DPID、連接埠號與 MAC 均會透過 SDN-IP 被 ONOS 得知，並被視為 SDN 網路的 attachment point，因此這些被 ONOS 當作 attachment point 的外部路由器，便能夠透過 ONOS 連在一起。Multi-point to single-point intents 的用途是將 ONOS 內的 host 與外部網路連線，這些 Intent 作法就是，根據 Ip prefix 找出一個出口的 attachment point，然後再將剩下幾個 attachment point 作為入口並與此出口配對，之後針對這些配對，計算出最佳的路徑，最後告知 ONOS，ONOS 就將 Flow entry 新增於路徑上的 SDN 交換器。

ONOS 透過 SDN-IP 接收外面 BGP 訊息、一對多與多對一的 Intent 便能夠成功的讓兩個傳統網路透過 SDN 網路互相連接，解決了 SDN 網路跟異質網路介接封包路由不相容的問題。

3. 設計與實作

利用 SDN-IP 能夠讓 SDN 網路與傳統網路介接時，不同架構間能順利的傳送封包，然而 SDN-IP 資訊查詢時的介面較不友善，故此章將描述我們替 SDN-IP 實作的 WEB UI。

3.1 問題定義

由於目前 SDN-IP 所提供之操作介面相當陽春，亦即使用者必須登入 ONOS，才能查詢 SDN-IP 相關資訊，並利用命令列的方式輸入命令。使用者一般情況下操作方式如下：

- Step 1: 啟動 ONOS 命令列 。
- Step 2: 輸入 SDN-IP 提供的與查詢 BGP 相關資訊查詢之指令。
- Step 3: 取得結果。
- Step 4:若要再查詢其他資訊，回到 Step2

故有以下缺點：

1. 使用者須記憶相當多的操作指令，時間久容易忘記，必須浪費寶貴時間查詢。
2. 以此方式操作，步驟相當繁瑣，一次只能操作一個命令。
3. 若不慎輸入錯誤指令，又必須重新輸入，浪費許多寶貴的時間。
4. 於結構較為複雜的網路時，資訊量多，就難以有效率地判讀。

我們擬實作 Web UI 系統，以網頁顯示的方式取代原本以命令列方式獲得資訊的方式。例如：原本命令列輸入 bgp-speakers、 bgp-neighbors、bgp-routes、routes 等命令才會出現的資訊，現在我們只需啟動 Web UI 系統就可以馬上獲知這些資訊，Web UI 操作的方式如下：

- Step1:打開瀏覽器。
- Step2:輸入網址。
- Step3:獲得 bgp 相關命令之資訊。

針對 SDN-IP 開發的 WEB UI，將有以下的優點：

1. 減少使用者輸入失誤的機會：利用此 Web UI 來操作，以選單、列表等等方式，提供更方便的介面給使用者，讓使用者無須擔心記不住指令，並且可以減少使用者人為資料輸入的失誤。
2. 降低使用者操作時的負擔：利用 Web UI 系統，所有 SDN-IP 相關資訊可以隨啟即用，不需再繁複地啟動 terminal 視窗、登入 ONOS 系統，輸入相關指令。如此可以大大的減輕使用者的負擔。
3. 資訊定期更新：利用 Web UI 系統，可以定時的向 ONOS 查詢資訊，因此網頁顯示的資料將會隨時更新，不需使用者再定期的利用命令列查詢資訊。
4. 資訊一目瞭然：透過本系統的整理，所有 SDN-IP 的資訊都會透過列表的方式，以使用者易判讀、易懂的模式呈現在 WEB UI 上，讓使用者一目瞭然，大大降低使用者獲得所需資訊的不便性。
5. 跨平台特性：由於採用 Web UI 系統，故使用者可以於有瀏覽器的一般電腦、手機或平板上獲得資訊。

3.2 SDN-IP Rest API 資訊處理實作

在系統架構設計上，我們需要分別建立 ONOS 與網頁伺服器平台，設計的架構圖如圖 2 所示。網頁伺服器和 ONOS 之間是利用 Http 的 Rest API 來交換訊息，整個系統運作流程如下：

- Step 1: 使用者開啟瀏覽器，並輸入系統的網址，此時瀏覽器會傳送 request 至網頁伺服器 。
- Step 2: 網頁伺服器收到 request，便根據 SDN-IP 提供的 Rest API 網址，來向 SDN-IP 要求其提供的訊息。
- Step 3: SDN-IP 透過 Rest API 介面收到網頁伺服器的 request 後，便向 ONOS 要求所需的資訊。
- Step 4: ONOS 把資訊傳給 SDN-IP，接著 SDN-IP 就會整理這些資訊。
- Step 5: Rest API 把這些資料回傳到網頁伺服器，
- Step 6: 網頁伺服器得到資料後再回傳至使用者並於瀏覽器上顯示。

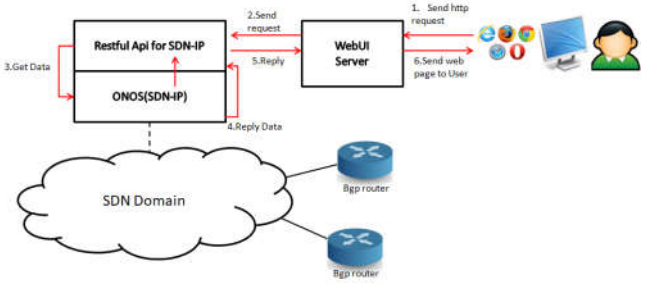

圖 2 系統架構圖

依據系統的設計，我們需要實作 SDN-IP 的 Rest API 與網頁伺服器的應用程式。ONOS 本身有支援 Rest API，因此我們可以利用 ONOS 系統提供的工具來新增一個屬於 SDN-IP 的 Rest API 介面。於已經安裝好 ONOS 的 Linux 系統中，可以參考 ONOS 的教學網頁來新增一個包含 Rest API 介面的應用程式骨架，新增方式請參考[22]。

有了骨架後，接著我們就在這上面新增要求 3 種資訊並處理的功能，這 3 種資訊分別對應以下 SDN-IP 的四個命令。

- bgp-neighbors：顯示本地端負責與 SDN-IP BGP Speaker 交換訊息的節點與和此節點連線的所有 SDN-IP BGP Speaker。
- bgp-routes：顯示收到 SDN-IP 整理後的路由表。
- bgp-speakers：顯示有哪些 SDN-IP 的 BGP

Speakers

我們針對 bgp-neighbors 這個命令的實作流程如圖 3。一開始 Rest API 框架收到 request 後，會交給我們實作在 Rest API 內的 TypeControl 函式，此函式會根據 request 的型態來指派給也是我們實作的 Bgp-Speakers、Bgp-Neighbors 與 Bgp-Routers 元件，這三個元件負責的工作如下：

- Bgp-Speakers GetterParser 元件:會跟 ONOS 的 NetworkCong Service 索取 bgpConfig 的資料，然後從中取出 speaker 欄位的所有資訊，例如：SpeakerName、connectPoint、vlan 與 peers 等等，取得後再根據 JSON 格式將這些資訊格式化並將處理後的資料字串回傳給 TypeControl。
- Bgp-Neighbors GetterParser 元件:此元件會與 BGP Service 索取 bgpSession 的資料，同樣的 bgpSession 內的資料，例如：remoteAddress、localAddres、remoteBgpId 與 localBgpId 等等，均會被取出來並將這些資料轉成 JSON 格式，轉換成功後就告知 TypeControl。
- Bgp-Routers GetterParser 元件：此元件與 Bgp-Neighbors GetterParse 一樣，都是跟 Bgp Service 要資訊，只是在這裡是要求 BgpRoutes4，也就是要求 IPv4 的 BGP 路由資訊，資訊內容有 Ip Prefix、nextHop、bgpIP 等資訊，一樣也是將這些轉成 JSON，然後交給 TypeControl 來回傳給使用者。

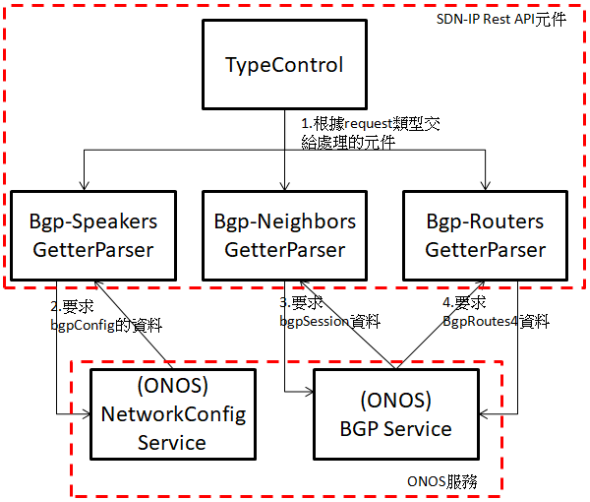

圖 3 SDN-IP Rest API 資訊處理的運作流程

3.3 網頁伺服器應用程式實作

在網頁伺服器應用程式這一部分，根據先前的設計，必須要滿足與 SDN-IP Rest API 交換訊息、資訊的解析與資訊的顯示這幾個功能。第一個功能我們利用 Servlet 實作，而最後兩個功能我們採用 JSP 實作。我們透過 Servlet 來執行後端資訊交換後，再將所得的資訊傳至 JSP 網頁來整理資訊並顯示，如此一來使用者就能夠看到 SDN-IP 相關命令的資訊。

網頁應用程式的執行流程如圖 4，當伺服器收到使用者的 Http Get 請求後便呼叫 Servlet 的 doGet，doGet 函式被呼叫後，會請 OnosCommunication Service 跟 SDN-IP 索要資訊，OnosCommunication Service 接收到請求後則是請 getBgpNeighbors、getBgpRoutes 與 getBgpSpeakers 這三個元件與 SDN-IP 聯繫來取得它們所要獲得的資訊，當 OnosCommunication Service 得到這三個元件的回復後，就將回復的資訊轉交給 JSP 網頁，JSP 得到的會是 JSON 格式的資料，因此 JSP 會解析 JSON 資料，並根據資料類型，以列表的方式顯示所得的資訊，最後使用者處就可以看到跟 bgp-routes、bgp-neighbors 與 bgp-speakers 三個命令相關的資料表單。

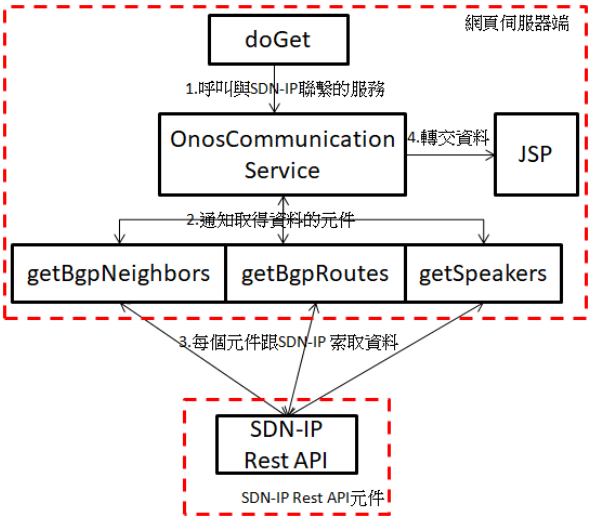

圖 4 網頁應用程式運作流程

4. SDN-IP 模擬環境建置與 WEB UI 測試

這一節首先我們會說明如何建置一個 SDN-IP 的模擬網路，接著會藉由這個模擬的網路來展示我們所開發的 WEB UI。要建立 SDN-IP 的模擬網路，最簡單的方式是至 SDN-IP 的 Tutorial 網頁下載官網設定好的虛擬機器來用，但是虛擬機器上的 ONOS 版本是老舊的版本，所以我們不利用這個方式建立，若是直接升級 ONOS 版本也將會遇到許多環境設定不相容的問題，因此我們採取全部軟體自己安裝與設定的方式，需要準備的軟體與工具分別是一台裝有 Ubuntu 的伺服器或虛擬機器、Mininet[23] 和 ONOS，在這裡我們利用 VirtualBOX[24]來建立一個 Ubuntu 16.04 作業系統的虛擬機器，並於這個虛擬機器中安裝最新版的 Mininet 與 1.85 版的 ONOS。

接著我們來說明如何建立 SDN-IP 的 SDN 模擬網路，圖 5[25]是我們要建立的網路拓樸，這個拓樸也是官網預設的實驗拓樸，這個圖是從 ONOS 拓樸圖截取的。r1~r4 是其他網域的 BGP 路由器，後面分別還接著 h1~h4 的 Client，IP 分別是 192.168.1.1、

92.168.2.1、192.168.3.1 與 192.168.4.1，而 bgp 的點是 SDN-IP 的 BGP Speaker。接著我們依照以下步驟來建立一個模擬網路。

1. 利用 apt-get 的方式安裝 quagga[26]。
2. 輸入"mkdir ~/Applications"建立 Applications 資料夾。
3. 下載 apache-karaf[27]，我們下載的版本是 3.0.5，將下載來的壓縮檔解壓至 Applications 資料夾下。
4. 進入 onos/tools/dev 資料夾，根據 karaf 下載的版本來修改 bash_profile 檔案中 KARAF_VERSION 與 KARAF_ROOT，圖 6 是我們修改的範例。
5. 若 ONOS 已依照官網說明下載設定好後，就利用 MAVEN 來編譯 ONOS，當編譯完成後，就利用 onos-karaf 指令啟動 ONOS。
6. 啟動 ONOS 後，再利用 ctrl+D 關閉 ONOS，之後可在~/Applications 底下發現 config 資料夾，請複製 onos/tools/tutorials/sdnip/configs 內的 gui.json 與 network-cfg.json，貼上到 ~/Applications/config。
7. 利用先前的方式啟動 ONOS，接著開新的 terminal 並進入 onos/tools/tutorials/sdnip/ 資料夾。
8. 在 sdnip 資料夾下輸入"mn --custom tutorial.py --topo sdnip --controller remote,127.0.0.1 –nolistenport"啟動 Mininet。

按照以上步驟便可以建立如圖 5 的實驗 SDN-IP 拓樸，可以嘗試在 Mininet 視窗下輸入指令，嘗試 h1 與 h2 之間是否能夠互 ping。

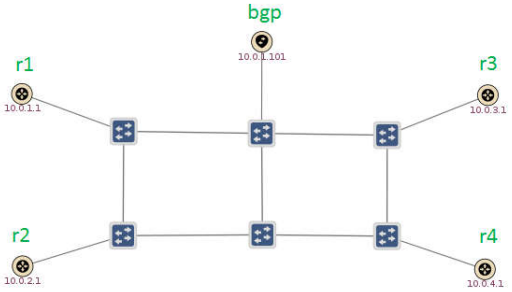

圖 5 實驗拓樸[25]

```
export KARAF_VERSION=${KARAF_VERSION:-3.0.5}
export KARAF_ROOT=${KARAF_ROOT:-~/Applications/apache-karaf-$KARAF_VERSION}
export KARAF_LOG=$KARAF_ROOT/data/log/karaf.log
```

圖 6 修改範例

接下來我們要將先前實作的程式套用在這個 SDN-IP 模擬網路上。首先可以參考 ONOS 操作指令將 SDN-IP Rest API 編譯並加入至正在執行的 ONOS 上並啟動。接下來我們安裝 TomCat 伺服器，並把網頁的應用程式打包成 war 檔，放至 TomCat 的 webapps 資料夾下，最後啟動 TomCat 伺服器。

我們開啟任一瀏覽器，輸入正確的網址後，便可以顯示出我們實作成果，其顯示結果如圖 7。圖中我們可以看到對應 bgp-routes、bgp-neighbors 與 bgp-speakers 命令的三個表格，每一表格均顯示這三個命令輸入後會得到的所有資訊。另外除了可以看到資訊外，使用者移動滑鼠至黑一列資料上，將會有 highlight 效果呈現。從網頁中使用者就能一覽無遺的看到命令所產生的結果，對於資料蒐集和比較上有很大的助益，再加上 highlight 的效果，讓使用者更能知道正在閱讀的地方，讓閱讀資料變得更容易。

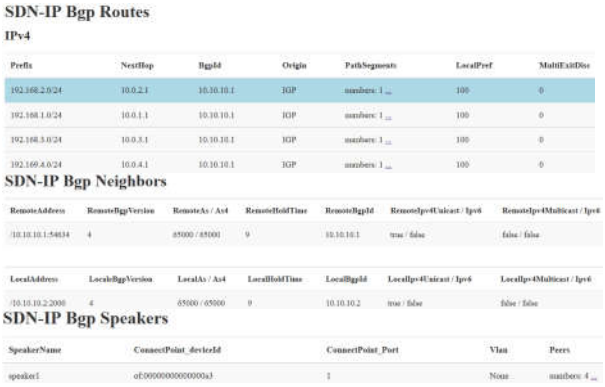

圖 7 網頁結果顯示

5. 結論

本論文針對 SDN-IP 實作一個專屬的 Rest API 介面，使用者可以利用此我們開發的這個介面本身提供的功能來獲取 SDN-IP 中與 BGP 和路由相關的資訊。除了這個介面外，本團隊也結合這個介面實作了一個 SDN-IP 的網頁介面系統，如此一來使用者就更簡單更輕易的能從這系統中獲取資訊。

未來我們將在現在的系統上持續的研究並且擴展，更進一步設計更多的功能，例如：網路拓樸、流量資訊與路由路徑的顯示，如此一來將有益於跨單位與跨國間傳統網路與 SDN 網路的串聯與管理。

參考文獻

[1] Software Defined Network, https://www.opennetworking.org/sdn-definition/

[2] S. Ortiz, "Software-defined networking: On the verge of a breakthrough?" Computer, vol. 46, no. 7, pp. 10–12, Jul. 2013.

[3] P. Berde, M. Gerola, J.Hart, et al. ONOS: Towards An Open, Distributed SDN OS. Proceedings of the 3rd Workshop on Hot topics in Software Defined Networking. ACM, 2014: 1-6.

[4] http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf

[5] Floodlight, http://www.projectfloodlight.org/

[6] OpenDaylight, https://www.opendaylight.org/

[7] Ryu, https://osrg.github.io/ryu/

[8] SDN-IP, https://wiki.onosproject.org/display/test/SDN-IP

[9] CORD, http://opencord.org/

[10] SONA, https://wiki.onosproject.org/display/test/SONA%3A+DC+Network+Virtualization

[11] Virtual Private LAN Service, https://wiki.onosproject.org/display/test/Virtual+Private+LAN+Service+-+VPLS

[12] Border Gateway Protocol (BGP), https://en.wikipedia.org/wiki/Border_Gateway_Protocol

[13] OpenFlow, http://archive.openflow.org/

[14] 周大源、楊哲男、古立其、劉德隆, " TWAREN SDN 虛擬專用連線之高速傳輸應用", TANet2016 論文集，花蓮，2016 年 10 月。

[15] M. Lasserre, V. Kompella, "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling," IETF RCF 4762, 2007.

[16] POX Controller, https://github.com/noxrepo/pox

[17] NOX Controller, https://github.com/noxrepo/nox

[18] Beacon, https://openflow.stanford.edu/display/Beacon/Home

[19] REST API, http://www.restapitutorial.com/

[20] Intent Framework, https://wiki.onosproject.org/display/test/Intent+Framework

[21] Network Configuration Protocol (NETCONF), https://tools.ietf.org/html/rfc6241

[22] ONOS tutorial, https://wiki.onosproject.org/display/test/Template+Application+Tutorial

[23] Mininet, http://mininet.org/

[24] Oracle VM VirtualBox, https://www.virtualbox.org/

[25] SDN-IP tutorial, https://wiki.onosproject.org/display/test/SDN-IP+Tutorial

[26] Quagga Routing Suite, http://www.nongnu.org/quagga/

[27] Apache Karaf, http://karaf.apache.org/