SDN 網域間可程式化路由交換機制的設計

黃文源　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, 1203053, tlliu}@ narlabs.org.tw

摘要

本論文在許多 SDN 網路構成的網域上設計一個可程式化的路由交換機制。在利用 SDN-IP 建構而成的大型 SDN 網域上，因為 SDN-IP 機制的原因，路由策略為挑選最短路徑，因此 SDN 網域內每對 IP 配對只會使用單一一條路徑，故當此路徑壅塞或是使用者利用多重路徑來試圖增加頻寬或減少負載時，往往 SDN 網域之間的路由將會是瓶頸，因此我們設計一個可程式化的路由交換機制來解決這個問題，並搭配去年我們所開發的介面使設計的機制能有友善的資訊介面。藉由上述的設計與組合將使網路更具靈活性、彈性以及減輕網路的負擔。

關鍵詞：SDN-IP、ONOS、SDN

1. 前言

近年來，由於網路急速發展，與網路相關的技術與概念也相繼被提出，例如：大數據傳輸與雲端計算，當然有許多應用就是以這些技術為基礎而發展出來的，像是人工智慧就需要運用到雲端計算。由於眾多應用的出現，使用者遽增，為了能夠順利提供服務，也為了能服務世界各國的人，各家應用服務的廠商開始於世界各地建置資料中心（Data center）。

然後增設資料中心時，將會面臨傳統網路方面的許多限制，例如：QoS、 IP 缺少或虛擬化等，為了解決這些問題，將會耗費許多的成本增購更多的硬體設備，因此為了成本的考量，新型態網路架構的採用和研究是不可避免的，軟體定義網路(Software Defined Network；SDN)[1]是最熱門的網路架構，且此架構也已經受到許多雲端服務提供者所採用，例如：Google 與 Amazon。

SDN 和傳統網路之間的區別在於控制層(Control Plane)和資料層(Data Plane)是分開的：資料層一樣放置於設備中，只負責資料的傳輸。控制層是獨立的，所有決策都由它決定。在傳統網路架構中，用戶受限於實體設備的廠商，有想要的功能就必須花費額外的成本請廠商於交換緝獲路由器上新增，而 SDN 則是可以自由的實作所需的更能，因此 SDN 可以降低許多成本。

圖 1 為 SDN 的架構圖，控制層由 SDN controller 控制，常見的控制平台有 ONOS(Open Network Operating System) [2][3] 、 Floodlight[4] 、 OpenDaylight (ODL)[5]與 Ryu[6]，資料層部分通常使用 OpenFlow switch，兩層之間的聯繫通常採用相容 SDN 的協定例如：OpenFlow，透過這個協協定，資料層就可以針對控制層的決策結果來設定其內部的 Flow Table，之後資料層處理封包就會依據 Flow Table 的紀錄來處理。藉由 SDN 的可程式化開放架構的特性，使用者就可以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[7]等，

當 SDN 網路建置完成後，接著就要考量與傳統網路介接的問題，當兩個異質性網路介接時，必須解決兩個網域間路由路徑與路由資訊交換等問題。這個問題於[8][9]中提出解決方法，我們結合 ONOS、SDN-IP[10]與 Quagga[11]來達成異質網路的介接，並設計與開發一套使用者介面讓使用者能夠知道 SDN 網路內的 BGP 資訊以及網路的拓樸。然而當資料中心頻寬不足需要多重路徑路由來增加頻寬或是減少網路負擔時，在文獻中的架構下，由於 SDN-IP 針對每個 IP 配對，只會使用最短路徑，因此多重路經將會於 SDN 網域內收斂，造成網路的瓶頸，故本文將設計一個機制來解決此問題。

本篇論文的第二節將介紹研究背景及相關的技術，包含 ONOS、SDN-IP 與 Quagga，第三節描述所設計的機制與介面，第四節為此機制的應用範例。

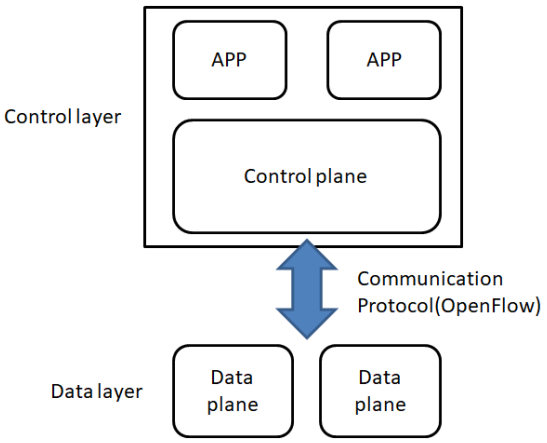

圖 1 SDN 網路架構

2. 相關知識

因為SDN結合SDN-IP有著可程式化且介接異種網域的能力，但 SDN-IP 本身的缺陷，因此本文設計一套路由交換機制，為了瞭解我們提出的機制，本章將會說明相關的背景知識。

2.1 ONOS

ONOS 是 SDN 網路上的一個控制平台，它是開放原始碼的，因此使用者可以自由的下載使用，此外它也支援資料中心虛擬化和雲端運算之基礎架構，因此服務供應商與建置 SDN 網路的組織是其服務的對象。ONOS 有著優秀的模組化、南北向介面的支援與眾多的應用程式，與 POX[12]、NOX[13]、Beacon[14]和 Floodlight 等最早的 SDN 控制平台而言，它有更加強大的擴充性(Scalability)、效能(High Performance)和可用性(High Availability)。

ONOS 因為屬於分散式叢集架構，因此當 ONOS 有故障時，ONOS 的將會遵照 SOP 快速的將所有的轉移資料至備援系統，另外網路的資料分散於每台 ONOS 之上，因此 ONOS 各自有不同的任務需處理，所以使用 ONOS 建置大型 SDN 網路有益於控制層的容錯率與網路工作的處理能力。

圖 2 為 ONOS 的內部架構，內部將分為七層，各層的功能如下所述：

- Apps：應用程式層，使用者自己開發的程式均屬於這一層。
- NB API：NorthBound API 層，主要是用來讓 App 層和核心層能交換資訊的 API，例如：Rest API[15]。
- Core：主要是負責 ONOS 運作的核心程式，例如：topology、path 與 Intent 等。
- SB API 與 Providers：這兩層的目的是 Providers 蒐集介面串流等資訊然後透過 Providers Service 告知 core 元件，將資訊紀錄於 core 之中。
- Protocols：負責讓實體設備與 ONOS 通訊，通常為 OpenFlow protocol。
- Network Elements：支援 SDN 的實體設備。

ONOS 的內部架構有南北向介面支援，北向介面除了使用上面提到的 Rest API 外，ONOS 尚有提供 Intent[16]的機制，利用此兩種方法便可與 ONOS 交換訊息。南向介面上有支援 P4[17]、OpenFlow、NETCONF[18]、TL1[19]、SNMP[20]與 RESTCONF[21]等協定，因此能管理 P4、OpenFlow 設備、白牌交換機(White-Box Switches)和傳統網路設備。

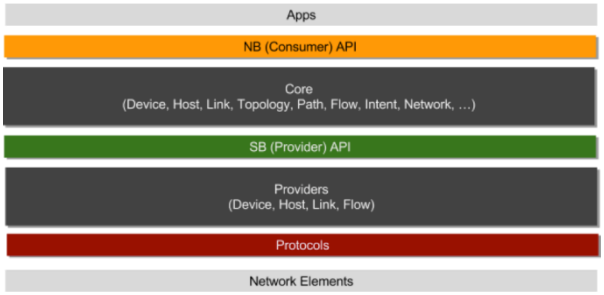

圖 2 ONOS 系統分層[22]

2.2 SDN-IP

SDN-IP 是 ONOS 應用程式中的一個，他使用 Border Gateway Protocol(BGP)[23]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。

圖 3 為 SDN-IP 的網路架構，SDN-IP 架構於 ONOS 之上，負責接收 BGP Speaker 的路由表，然後告知 ONOS 如何處理跟路由表相關的封包。SDN-IP、BGP Speaker 與 External BGP Router 之間路由交換是藉由 iBGP 和 eBGP 來達成的，BGP Speaker 和 External BGP Router 之間採用 eBGP，SDN-IP 與 BGP Speaker 則是利用 iBGP，值得注意一點的是，SDN-IP 並不會發送任何的 BGP 封包，他只負責接收而已。SDN-IP 透過一對多接收 BGP 封包，接著再利用多對一的 Intent 建立路徑的方式，使兩個不同 IP 網域的網路透過 SDN 的轉傳互相連接，解決了 SDN 網路跟異質網路介接無法互相傳輸封包的問題。

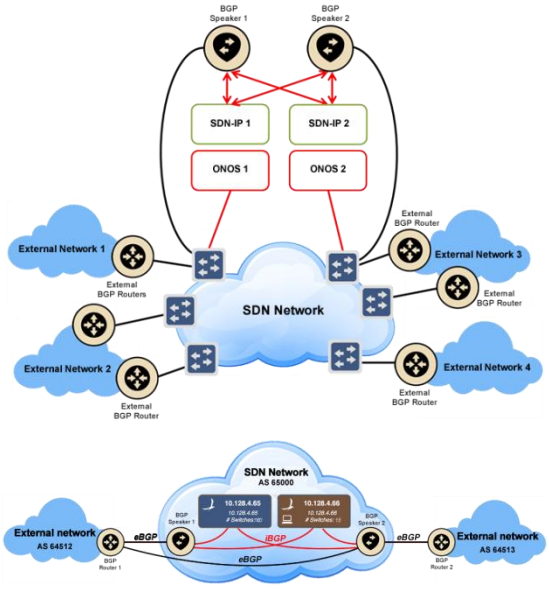

圖 3 SDN-IP network model[24]

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 網路跟異質網路之間路由交換的問題。

3. 可程式化路由交換機制規劃

在這章節中，我們將解釋我們規劃的機制，圖 4 為所規劃的機制範例，一開始 SDN 網域之間會利用 BGP 找出一條預設的最短路徑，但是此條最短路徑可能導致頻寬不足，或是 loading 太重，因此使用者可以透過使用者介面挑選路徑，後端伺服器會將使用者挑選的路徑資料，告知 ONOS，ONOS 便會增加 flow entries 到路徑上的 SDN 交換器，並賦予比較高的 priority，使後續的封包進入時能夠選擇客制化的路徑。

此機制主要由兩大部分所組成，分別是實體部分與使用者介面部分。實體部分，採用 SDN 網路與 Quagga 結合的方式來達到可程式化路由交換機制，而 SDN 網路又分為 SDN 交換器與 ONOS，其中 ONOS 會運行 SDN-IP 的應用程式，用來接收 eBGP 並設定相關的 flow entries，因此網路啟動一段時間後，就會替每對 IP 找出一條最短路徑的路由。

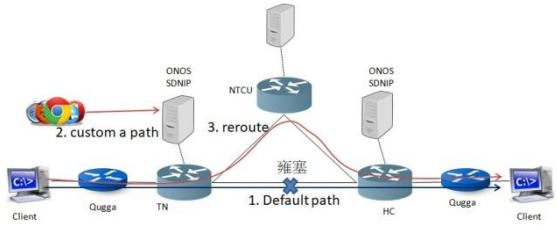

圖 4 拓樸範例

接下來為使用者介面部分，這部分我們預計於先前開發的 SDN-IP 使用者介面上新增路由選擇介面以及路由顯示介面，圖 5 為路由選擇介面的運作流程，使用者透過此介面設定 SDN 網路中要使用哪條路徑，當使用者決定好路徑後，使用者介面將會把設定資料告知後端伺服器，之後再轉成 JSON 格式的 Flow entry 並透過 RESTful API 將資料傳送至 ONOS 使其能新增相關的 Flow entries 於 SDN 交換器之上。路由顯示介面則是藉由修改 SDN-IP 使用者介面上的拓樸顯示介面來達成，在原本開發的拓樸顯示介面中就有路徑顯示功能，故我們將新增一個設定，當後端伺服器取得 flow entry 時，將會另外處理客制化的 flow entry，使拓樸顯示介面上能夠看到客制化的路徑。

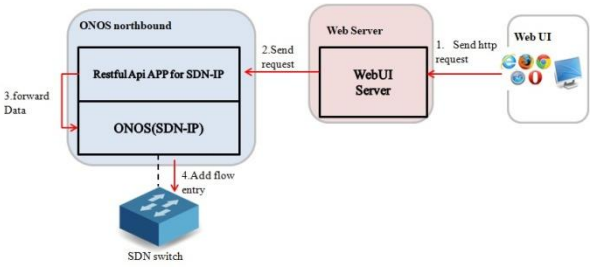

圖 5 使用者介面運作流程

4. 機制的應用範例

這前一章中解釋了我們規劃的路由交換機制，將提供一個此機制的應用情境。我們與成大楊竹星老師合作，在我們的機制上使用他們開發的多重路徑視訊串流，如果不應用我們的機制，雖然應用程式號稱多重路徑視訊串流，但是在 SDN 網域互相介接的情況下，仍然只會使用單一路徑，故結合我們規劃的機制變能夠讓視訊串流於此種環境下真正達到多重路徑的路由。圖 6 為預計於年底 SC 研討會上展示的應用，我們使用三台 SDN 交換器來構建一個 SDN 網域互相介接的拓樸，此三台交換器分別位於 NCTU、NCHC-TN 與 NCHC-HC，彼此間利用 TWAREN 線路並搭配 ONOS VPLS 來達成連線，另外我們也於配置 5 台 Quagga 來交換 BGP，2 台分別位於交換器與用戶端之前，用來和 ONOS 處的 Quagga 交換路由，其它 3 台位於 ONOS 處，負責接收 eBGP 並把路由告知 ONOS。

首先，我們必須先啟動 ONOS 上的 SDN-IP，讓 SDN 網域之間彼此交換路由以便建立欲由路徑，接著於 Video Streaming Server 端啟動串流伺服器，啟動命令為” python Server.py server_rtsp_port”，接著啟動 Video Streaming Client 內的串流用戶端，啟動命令格式為” python ClientLauncher.py server_ip server_rtsp_port client_rtp_port video_file flow_number”，我們 flow_number 先設定 2，表示將串流分成兩個 flow，然後觀察每台 ONOS Web UI 中的 flow entry 配對計數，發現不管分成幾個串流，於 SDN 網域之間，只會行走預設路徑。

接下來我們利用 RESTful API 將圖中的 flow entry 規則加入至 TN 的 ONOS，並等待 ONOS 設定 TN SDN 交換器的 flow entry，當 ONOS 設定完成

後，我們同樣依照前述的方式啟動視訊串流的伺服器端與用戶端，然後檢視每台 ONOS 上的 flow entry 配對計數，就可以觀察到視訊串流的 2 個 flow 使用不同的路由，真正的達到多重路徑視訊串流。

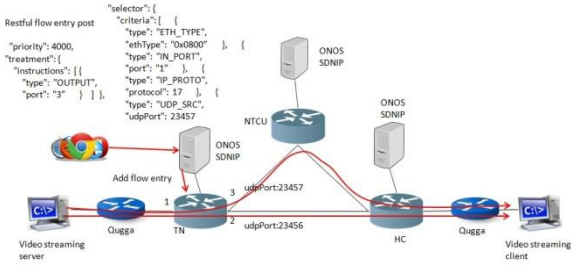

圖 6 SDN-IP network model

5. 結論

去年我們開發了 SDN-IP Web UI，可以透過 UI 來得知 SDN 網域介接時選擇的路徑以及相關 BGP 資訊，然而卻少了可程式化路由的功能，讓 SDN 網域之間路徑選擇少了許多的彈性。透過此論文，讓我們確認了可程式化路由的可行性，以及相關應用，如此一來將能夠大幅增加網路傳輸的效能與彈性。

未來方面，我們將持續的增加許多路由資訊顯示的相關功能，期待能夠提供便利更友善的 SDNIP 路由相關功能與介面

參考文獻

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

[2] 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.

[3] ONOS white paper, http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf

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

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

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

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

[8] 黃文源、李慧蘭、周大源、劉德隆,” 以 SDN 連結 L3 網域及南北向介面開發”, TANet2017 論文集，南投，2017 年 10 月。

[9] 黃文源、李慧蘭、周大源、劉德隆、Fei I Yeh、Jim Hao Chen、Joe J. Mambretti, ” SD-WAN 架構下 Layer 3 繞徑機制設計”, TANet2018 論文集，桃園，2018 年 10 月。

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

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

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

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

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

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

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

[17] P4 Language Consortium, https://p4.org/

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

[19] Transaction Language 1 (TL1) wiki, https://en.wikipedia.org/wiki/Transaction_Language_1

[20] Simple Network Management Protocol (SNMP), https://www.ietf.org/rfc/rfc1157.txt

[21] RESTCONF Protocol, https://tools.ietf.org/html/rfc8040

[22] ONOS System Components, https://wiki.onosproject.org/display/ONOS/System+Components

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

[24] SDN-IP User Guide, https://wiki.onosproject.org/display/ONOS/SDN-IP+User+Guide