軟體定義網路下(SDN)跨網域之端對端自動拓樸與流量檢視

黃文源　　胡仁維　　劉德隆
國家高速網路與計算中心
{ wunyuan, hujw, tlliu }@nchc.narl.org.tw

摘要

未來網路為現今熱門的研究課題之一，許多研究人員為了能解決傳統網路的問題，提出了一些新的網路架構，其中 SDN(Software-defined networking)較為知名。SDN 的最大好處是讓使用者能夠在支援 SDN 的 switch 上自由開發所需策略，因此不再受限於實體機器廠商提供的有限功能，將能夠節省許多資金。實作 SDN 的架構，比較知名的軟體為 OpenFlow ，現今有許多組織利用 OpenFlow 於它們的資料中心，例如:Google，因此對於 OpenFlow 網路的管理與維護將是很重要的一個工作，然而現今的 OpenFlow 網路不支援 Inter-domain，使用者從相關的 GUI 介面上無法得知其它網域的 host 與 OpenFlow switch 存在位置，執行 SLA 時，管理人員無法獲知正確的路由路徑，將造成管理與維運上的不便，故本篇文章，將利用先前實作的支援 Inter-domain 的網路拓樸與流量追蹤系統來研究與實作 host 與 host flow 的展示。

關鍵詞：未來網路，OpenFlow，NOX，ENVI，SDN。

1. 前言

在最近這幾年中，未來網路(Future Internet)[1][2][3][4][5]已經成為了研究網路方面的熱門主題之一了，眾多的專家學者們為了解決網路快速發展所衍生的問題，以及想發展一個能自由地設計所需協定或策略的新型態網路架構，紛紛熱情的針對未來網路進行相關研究。在住些研究當中，有專家學者提出了 SDN(Software-defined networking)[6][7][8][9]概念來解決現今網路上的問題。

在提出 SDN 之前，許多公司或住家在只有單一網路的情況下，會使用 switch 或 router 來讓眾多電腦能同時存取網路，然而傳統的 switch 與 router 其控制平台與資料平台是綁在一起，僅能使用廠商所提供的功能來制定策略，無法自由的開發所需的應用程式，再者現今網路規模越來越龐大，有許許多多的協定被使用，對於 switch 和 router 而言，它們就必須針對不同的協定來做正確的解封或是包裝封包，因而造成機器的負擔，使得網路效能不佳，而 SDN 的提出正好可以解決上述之問題。

SDN 跟傳統網路的差異是，控制平台與資料平台是分開的，資料平台的任務僅負責傳送資料，所有一切決策均在控制平台決定，而控制平台可能為裝著控制系統的電腦或伺服器，在此種架構下，對於網路管理人員而言，他們能夠根據自己的喜好在控制平台上開發程式，不用像傳統的 switch 或 router，想要額外的功能就得再花錢購置新的機器，如此一來也能節省金錢上的負擔。在架構 SDN 網路時，需要一些方法來讓控制平台能夠與資料平台聯繫，比較知名的是 OpenFlow，且現今 Google、Facebook 與 Amazon 在其資料中心上也套用 OpenFlow 來實作 SDN，有益於它們維護和管理所屬的資料中心。

由史丹佛大學開發的 OpenFlow，在資料平台中有著 Flow Table，當 Flow Table 中有對應的 Flow Entry，封包才能夠被 OpenFlow 所處理。在控制平台部分，使用者能依喜好安裝適合的控制軟體，例如：NOX[10] 、 POX[11] 、 Beacon[12] 與 Floodlight[13]，不管安裝哪套控制軟體，兩平台間均透過 OpenFlow 的協定來通訊。

OpenFlow[14][15][16]網路為中央式的網路控制，控制平台上會儲存著此 OpenFlow 網路的所有訊息，例如:flow 資訊、節點資訊與 link 資訊等，因此任何的決策和命令都是由控制平台來發布，並讓 OpenFlow switch 根據這些決策來處理封包。當不同組織有著各自的 OpenFlow 網路時，它們也會有著所屬的 flow 管理策略，當網路發生問題或是 flow 傳送不如預期時，依據 OpenFlow 的架構，這些組織們也僅能得知它們自己的 OpenFlow 網路狀況，無法確切知道問題發生在何處，在維修上將會耗費大量時間，若 OpenFlow 能支援跨網域所有節點與跨網域 Flow 顯示，對於任何組織來說，這樣能有益於網路管理與維護，而對於研究人員而言，也有益於快速與正確地開發所需之路由路徑與策略。

先前已完成 switch 自動拓樸與 switch 之間的流量檢視[17][18]，然而當 OpenFlow 網路需要執行 host QoS 與 SLA 時必須知道 host 的位置與 host 流量資訊，故我們擴充以前實作的系統，讓其能檢視 host 相關資訊。由於主控者平台原本的追蹤 host 機制的關係，在跨網域的系統上將會顯示錯誤的 host link 與 flow，故此擴充重點將在於如何在追蹤 host 機制的影響下，讓跨網域的系統能夠顯示正確的資訊。

本論文之架構與內容如下所述：第二章將介紹 OpenFlow 架構的跨網域的特點與方法，第三章詳細說明如何實作在跨網域的 OpenFlow 下正確顯示包含兩客戶端、link、flow 的完整拓樸，第四章說明如何架構跨網域的 OpenFlow 網路來展示實作之成果，最後為本論文結論與未來研發工作。

2. 背景知識

此章將分成兩小節簡單介紹 Inter-domain 與實作 Inter-domain 的相關文獻。第一節介紹 Inter-domain 以及它的重要性，第二節描述文獻中如何在 OpenFlow 網路上實作 Inter-domain 的技術。

2.1 Inter-domain introduction

在 OpenFlow 網路中，當控制平台控制著數台 OpenFlow switch，表示這些 OpenFlow switch 屬於同一個網域，反之若由不同的控制平台控制，則表示不同的網域，網域的多寡是依靠著控制平台的數量來決定的，而 Inter-domain Information 就是表示網域之間的資訊。

對於架設 OpenFlow 網路來合作與研究的任何組織而言，因為有著不同的控制平台，也沒有統一的網路 flow 策略，且管理人員也只能獲知自己所屬網域的 OpenFlow 網路狀況，無法得知 Inter-domain 的情況，這些因素將會在研究和管理上造成些許不便，然而現今 OpenFlow 已實作的軟體 NOX、Beacon 與 Floodlight，均尚無支援 Inter-domain 之技術，故需得於上述軟體上實作 Inter-domain。這四種軟體中，以 NOX 發展時間較早，且其所搭配的拓樸顯示的應用程式不僅能顯示節點分布情況與相關資訊外，也能在上面顯示各路由路徑，故本篇文章使用 NOX 來解說與實作 Inter-domain 技術。

ENVI[19]是與 NOX 搭配的拓樸顯示應用程式，它們的處理步驟如圖 1 所示。圖中 OF S/W_1 和 OF S/W_2 分別受 NOX_1 與 NOX_2 所控制，CL_A 和 CL_B 分別是 host，首先讓 CL_A 傳送 Flow 至 CL_B，接著 NOX_1 與 NOX_2 會設定拓樸資訊，此時 ENVI 傳送 Request 至 NOX_1，然後 NOX_1 傳送剛剛設定的拓樸資訊至 ENVI，最後 ENVI 處理所得到的資訊並以圖形化的方式顯示。從處理流程中得知，它們未存放與處理 Inter-domain Information，因此以圖 1 的例子而言，ENVI 尚無法觀察到 Inter-domain 的任何資訊，僅能觀察到 OF S/W_1，使用人員就沒辦法簡單地取得更多資訊。

若控制平台軟體能支援 Inter-domain 的技術，對各個組織而言，他們可以很方便的得知網路上所有的 flow 情況以及每個節點情況，若何處有異狀或是損毀，能夠馬上得知情報進而減少維修時間，而在開發路由或 flow 策略方面，也因為能夠觀察到整個網路的 flow 與拓樸，故能減少許多開發時間。

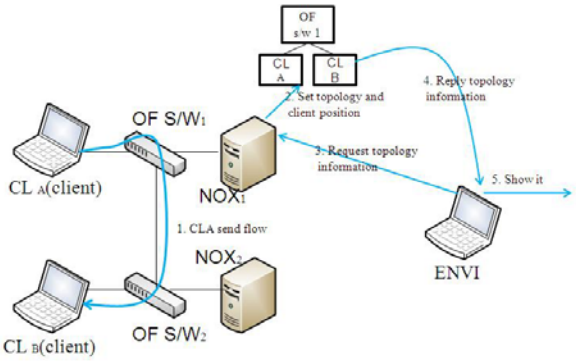


圖 1 NOX and ENVI processing step

2.2 Previous work

[17][18]均在 NOX 與 ENVI 上實作了 Inter-domain 技術，前者僅能夠顯示包含 Inter-domain 的 switch 拓樸分布，後者以前者為基礎，新增加 switch 之間的 Flow 與 Inter-domain flow 展示功能，使用者可以在 ENVI 上利用滑鼠點選任一 Flow 來觀看 Flow 方向、Flow 的起點與終點以及 host IP 資訊。

[17]此篇文章中 NOX 與 ENVI 流程如圖 2，紅色部分是新增與修改原本 NOX 與 ENVI 之處。圖 2 中的拓樸架構同圖 1，一開始讓 switch 間產生 Flow，此時 NOX_1 會接收到從 OF S/W_2 送過來的 LLDP，而這個 LLDP 也包含著 NOX_2 的 IP，故 NOX_1 設定拓樸時會將 OF S/W_2 加入拓樸之中並與 NOX_2 IP 配對，記錄至 switch ID 與 IP 配對表中，然後新增 OF S/W_1 與 OF S/W_2 間的 link，而 NOX_2 也會執行相同的做法來設定拓樸，接著讓 ENVI 先連接至 NOX_1 並取得拓樸資訊，這些資訊中有包含 NOX_2 的 IP，此 IP 將會被儲存至 IP 列表中，而其它資訊持續地被處理，然後於 ENVI GUI 上顯示處理完後的拓樸圖，當 NOX_1 資訊全部處理完畢後，將會根據 IP 列表中的 IP 連線至 NOX_2 上取得拓樸資訊，最後再將新取得的資訊跟原先 NOX_1 的資訊結合，並於 ENVI GUI 上顯示新增的資訊。

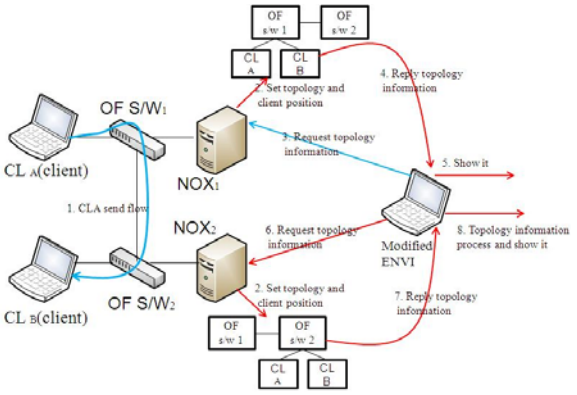


圖 2 Inter-domain implementation 1

圖 3 為[18]修改後的執行流程，與[17]的差異是，當 NOX 接收其他網域的 LLDP 時，不只會在拓樸新增節點外，也會標註接收此 LLDP 的 port 為 Inter-domain port。而原本 NOX 對於從 Inter-

domain port 進出的 Flow 不會有 Inter-domain flow 的紀錄，故 NOX 回應 flow request 時，於 Inter-domain flow 部分為空，ENVI 也就無法顯示 Inter-domain flow，因此[18]中，於回應 flow request 時，利用 Inter-domain port 來比對所有 flow，當確認 flow 由 Inter-domain port 進出後，即利用接收此 OF switch ID 與預定由哪一 port 進出等資訊再搭配 NOX 的 link 資料來找出 Inter-domain flow 的兩端點資訊，之後將這些資訊回應給 ENVI，如此一來 ENVI 就能顯示 Inter-domain flow。

利用上述文獻提供的方法，就能夠在 ENVI 上看到所有網域的 OpenFlow switch 節點，以及 switch 之間傳輸的 flow，然而卻無法得知 OpenFlow 網路上任何的 host，無法完全驗證網路的路由是否為正確，故 Inter-domain host 的顯示也需要被解決與實作。

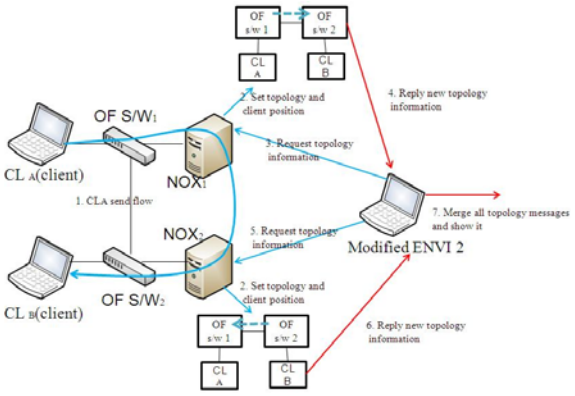


圖 3 Inter-domain implementation 2

3. INTER-DOMAIN HOST DISCOVERY

跨網域的環境中，使用前述支援跨網域顯示 switch 拓樸與 flow 的系統，能夠避免許多維修與管理的問題，然而原本的 OpenFlow 系統與前述的跨網域系統，尚無法正確顯示客戶端位置，對於 SLA(Service Level Agreement)將會造成困擾，故本節將針對此問題詳加說明並且也會提出相應的解決方法與實例。

3.1 問題定義

在跨網域的 NOX OpenFlow 網路中，雖然使用者可以透過 ENVI 來觀察封包傳送路徑，可是因為不支援跨網域的拓樸顯示，故無法得知其它網域的封包傳送路徑，例如下圖 4 的 OpenFlow 環境，有三台 OpenFlow switch OF_A、OF_B 與 OF_C 互相連接，且此三台 switch 分別受 NOX_1、NOX_2 與 NOX_3 控制，另外每台 switch 底下各接一台 host 分別是 1、2 與 3，若於此環境下，讓 1 送資料到 3，對於 NOX_1 而言，其會將 1 與 3 當成自己底下的 host，因此在 1 與 OF_A，3 與 OF_A 之間各自建立連結，使用者透過 ENVI 僅能得知如圖 5 的情形，卻無法得知封包是從 1->OFA->OFB->OFC->3 或是 1->OFA->OFC->3，如此一來對 SLA 而言將無任何幫助。

接著同樣使用圖 4 的拓樸環境，將原本的 NOX 用修改過可支援跨網域的的 NOX 來代替，同樣的讓 1 送資料至 3，假設 NOX_1 將資料由 OF_A 轉傳至 OF_C， NOX_3 再命令 OF_C 將資料送往 3，在此種的路徑下，NOX_1 與 NOX_3 均會得知 1 與 3 的存在，然後會如同前面敘述的，NOX1 會在 1 與 OF_A，3 與 OF_A 之間建立連結，而 NOX3 則是在 1 與 OF_C，3 與 OF_C 間建立連結，故會出現如圖 6 之情形。

雖然透過可支援 Inter-domain 的 NOX，可以得知 switch 間封包的傳輸路徑，可是在原本 NOX 決定客戶端機制的影響下，其形成的不正確 host 資訊將造成使用者的困擾。

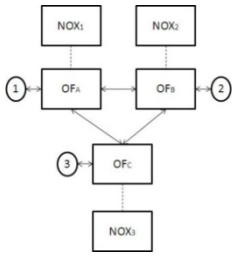


圖 4 SLA test topology

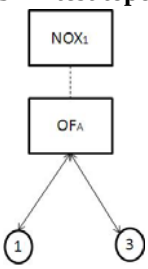


圖 5 SLA problem with default NOX

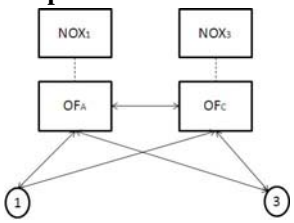


圖 6 SLA problem with inter-domain system

3.2 設計與實作

實作方面，主要分為 NOX 與 ENVI 兩大部分說明：

第一部分為 NOX 程式的修改，因為 NOX host 追蹤方法是根據 OpenFlow switch 從哪個 port 接收到 host 封包的資訊來設定位置，所以 Inter-domain 環境下，許多 host 雖然不在同一網域底下，可是每個網域的 NOX 卻會將所有 host 認定為自己所屬之 host，故為了要能取得正確的 host 位置，我們必須於 NOX 中辨識接收 host 封包的 port 的類型，如此才能順利的分析 host 是否存在於所屬之網域底下。

port 的類型分為 Inter-domain port 與 Intra-domain port，一般情況下 OpenFlow switch 的所有

port 均為 Intra-domain port，當某一 port 收到 Inter-domain 的 LLDP 封包時，NOX 將會從 LLDP 中取出節點相關資料於 Topology 中新增節點，同時也將接收的 OpenFlow switch ID 與接收此 LLDP 封包的 port 配對，然後放入用來確認是否為 Inter-domain port 的資料結構中。

接著修改 LAVI[20]中傳送 host 資訊的部分，首先須於回傳至 ENVI 的 Host Node JSON 中新增額外的辨識訊息，讓 ENVI 確認此為 host 而非 OpenFlow switch。然後 LAVI 中 host link 的部分，於 Host Link JSON 中的每條 link 資料內新增 InterDomain 欄位，用來讓 ENVI 判斷此 host link 是否為 Inter-domain link，若內容為 Yes 表示 Inter-domain host link，若為 No，表 Intra-domain host link。最後 Host Flow JSON 部分，我們於 Host Flow JSON 中新增些許欄位，讓 ENVI 能顯示更多的 Flow 資訊。

圖 7(a)(b)為修改成果的執行流程圖，流程圖中*表示為我們修改的部分，另外圖 7(b)處，由於 LAVI 中主要是修改 Host Link JSON 部分，故僅展示此部分的處理流程。圖 7(a)的流程中，當 NOX 收到 Inter-domain LLDP 後，於新增節點與 link 時也建立 Inter-domain 的 hash table，此 table 內容是 <Switch dp ID, port received LLDP >。圖 7(b)中，當 NOX 收到 ENVI 的請求時，會取得任一條 host link，然後使用 host link 中 switch 端的 switch id 至先前的 Inter-domain hash table 中查詢裡面紀錄的 Inter-domain port(port received LLDP)與 switch port 是否一樣，若一樣將會在 JSON message 新增的 interDomain 欄位上紀錄為 YES，不一樣則為 NO，最後再將此 Host Link JSON 回傳給 ENVI。

第二部分為 ENVI 程式的修改，因為我們有新增欄位於 JSON message 之中，故須修改 ENVI 中接收資訊的部分，讓所新增加的欄位資訊能正確地被讀取，這部分中最重要的是 host link 的處理，我們需在處理 host link 的部分新增一個辨識 Intra-domain host link 與 Inter-domain host link 的方法，當 ENVI 取得 Host Link JSON 中的資訊時，除了利用內部的資料建立 link 資料結構外，也利用資訊中的 interDomain 來判斷 host link 的類型，當為 Inter-domain host link 類型，表示它不直接與正連線的控制平台所控制的 switch 連接，故此處不會繼續進行 link 的處理，反之，Intra-domain host link 則會持續進行新增 link 的處理。

另外 Host Flow JSON 處理的部分，我們也需於此處增加一個判斷條件，當 ENVI 根據資訊建立新的 host flow 資料結構後，再利用其兩端點資料來判斷是否有此 link 的存在，若 host link 存在表示此為正常的 flow，然後繼續新增 flow，反之 host link 不存在則表示此為 Inter-domain host flow，故不應被儲存與顯示。

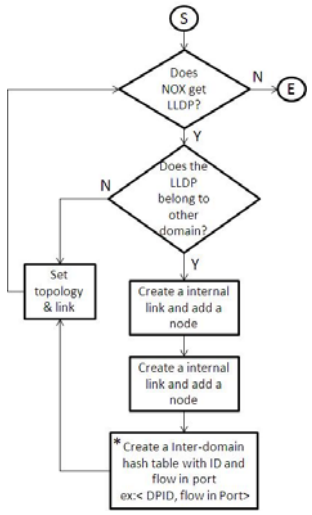


圖 7(a) Flow char of add an Inter-domain node

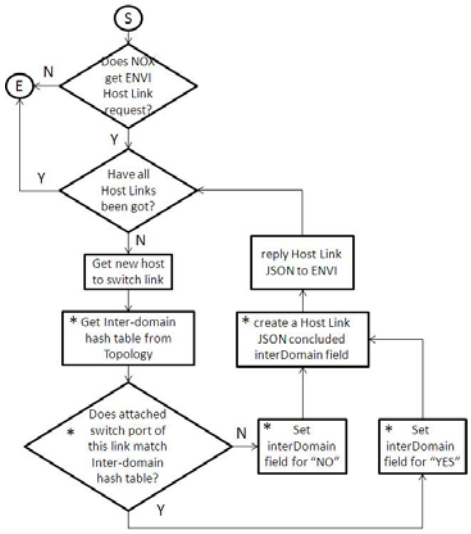


圖 7(b) Identify the Inter-domain host link within LAVI

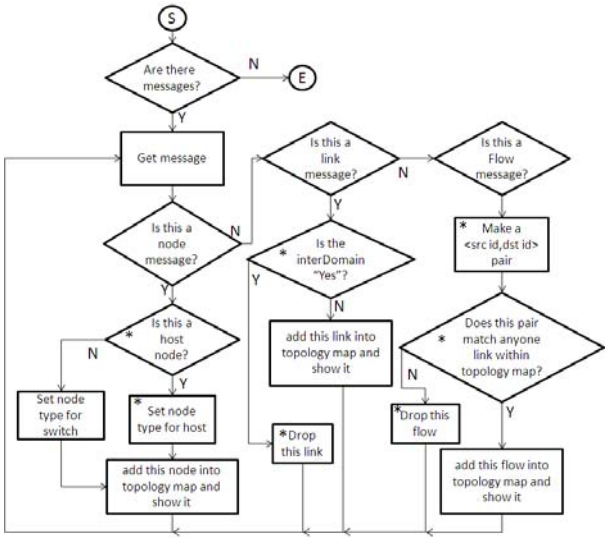


圖 8 The flow chart of ENVI handle messages

圖 8 為此階段修改後的執行流程圖，圖中*符號為修改的部分。ENVI 收到從 NOX 送來的 message 時，會先判斷 message 的型態，當為節點型態時，

會判斷此節點為 switch 或 host，然後將節點型態設成正確的類型，最後儲存入 topology map 之中並在 GUI 顯示。當 message 為 link 的資訊時，先判斷資訊中的 interDomain 是否為 Yes，Yes 則將資訊放入 topology map 並顯示，否則將此 link 丟棄。最後一種 message 類型為 flow 型態，一開始從 flow 資訊中取出來源端與目的端的 ID 做配對，然後至 topology map 中尋找對應的 link，如找到對應的 link，此 flow 資訊將被顯示並且儲存於 topology map 之中，反之則丟棄。

4. 實驗

接著利用 Oracle VM VirtualBox 建立如圖 9 的拓樸來進行測試。首先於 VirtualBox 中先建立三台 OS 為 Ubuntu 10.04 64bit 版本的 VM，並且安裝修改後的 NOX，此三台分別對應圖中的 NOX_1、NOX_2 與 NOX_3，而 OpenFlow switch 部分，則是建立四台 Ubuntu 10.04 32bit 版本的 VM，並在這四台 VM 上安裝 OpenFlow switch 軟體，此四台分別對應圖中的 OF_1、OF_2、OF_3 與 OF_4，最後 host 部分，建立四台任意 OS 且名為 CL_1、CL_2、CL_3 與 CL_4 的 VM，並且各自分配 10.0.0.1~10.0.0.4 的 IP 給它們。VM 間連線設定方面，NOX_1 管控 OF_1 與 OF_2，而 CL_1 與 CL_2 分別連接至此兩台 OpenFlow switch，NOX_2 管控 OF_3，CL_3 與 OF_3 連線，NOX_3 則是管控 OF_4，CL_4 與此 switch 連接

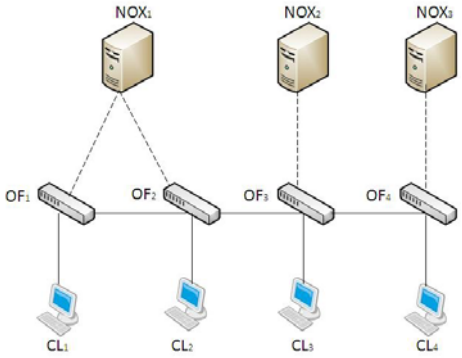
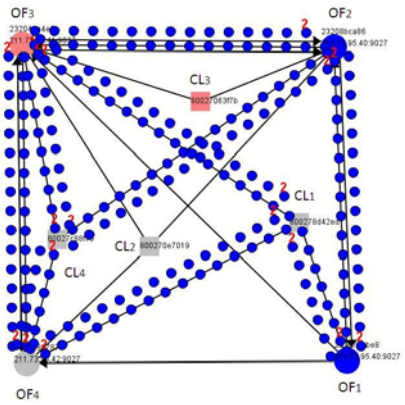


圖 9 Testbed topology

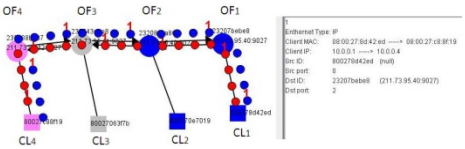
在測試方面，我們檢驗兩種不同的 flow 傳送路徑，第一種是由 CL_1 傳送 flow 至 CL_4， 第二種則是由 CL_2 傳送 flow 至 CL_3。首先先將文獻中的 Inter-domain ENVI Flow Viewer 系統顯示 host 與 host flow 功能開啟，然後執行第一種測試，測試結果如圖 10(a)，從圖中可以看到受 NOX 決定 host 位置的方法影響，對於 CL_1 和 CL_4，CL_1-OF_3、CL_1-OF_4、CL_4-OF_2 與 CL_4-OF_3 等 Inter-domain host link 以及 Inter-domain host flow 均被顯示出來，才會造成圖中網狀的拓樸。圖 10(b)為實作成果執行第一種測試的結果，可以從圖中觀察到 Inter-domain host link 以及 Inter-domain host flow 消失了，在 GUI 上的拓樸分布與實體上連接的線路分布一樣，flow 的路徑與資訊也為正確的結果。

在執行第二種測試方面，同樣的先利用 Inter-domain ENVI Flow Viewer 測試，圖 11(a)為測試結果，CL_2-OF_3 與 CL_3-OF_2 為 Inter-domain host link，而 link 之上的 flow 也為 Inter-domain host flow，與第一種測試的結果一樣，會顯示 Inter-domain host link 與 Inter-domain host flow，因此形成網狀的拓樸。圖 11(b)為實作的測試結果，測試結果與第一種測試相同，均能顯示正確的拓樸以及 flow 的情況。

另外我們也增加了兩種功能，第一種是可以藉由 host 顏色來分辨所屬的網域，例如：CL_1、OF_1 與 OF_2，它們屬於同一網域，故顯示的顏色是相同的。第二種是 host flow 資訊的顯示，藉由點選 host 與 switch 之間的 flow，可以從 GUI 右邊的 flow 視窗中看到此 flow 的內容，例如：兩端點 IP、MAC、port 與 flow type 等資訊，而這些新增的功能，有助於使用者管理與研究跨網域的 OpenFlow 網路。

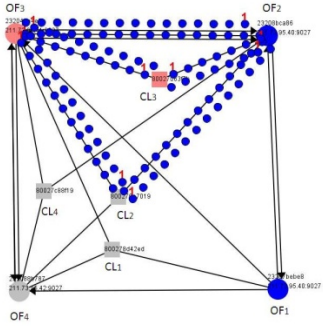


(a) Inter-domain ENVI Flow Viewer

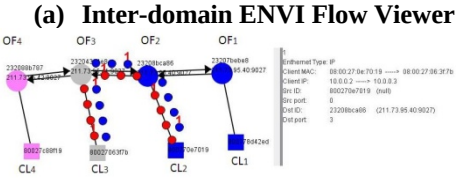


(b)The Implementation result of case 1

圖 10(a)(b) CL_2 forward flow to CL_3



(a) Inter-domain ENVI Flow Viewer



(b) The Implementation result of case 2

圖 11(a)(b) CL_1 florward flow to CL_4

5. 結論

在本篇中我們介紹如何解決 NOX 與 ENVI 顯示 host 與 host flow 的問題，藉由修改 NOX 的 LAVI 內的 host 相關元件，讓 NOX 能夠分辨出 host link 是否為 Inter-domain host link，並且藉著修改 ENVI 軟體，使其能從 NOX 上接收與處理新增的 Host JSON 資訊，並且在 GUI 上正確顯示 host 與 host flow。未來我們將研究並且擴展此系統，除了能進行未來網路管理之功能外，也將套用於其它控制平台軟體之上，如此將有益於跨單位及大型跨國未來網路平台之建置與維運。

參考文獻

[1] D. Cameron, "Internet2: The Future of the Internet and Next-Generation Initiatives, " Computer Technology Research Corp., February 1999.

[2] G. Fairhurst, B. Collini-Nocker, and L. Caviglione, "FIRST: Future Internet — a role for satellite technology," Satellite and Space Communications, 2008. IWSSC 2008. IEEE International Workshop on, pp. 160-164, Oct.2008.

[3] D. Y. Kim, L. Mathy, M. Campanella, R. Summerhill, J. Williams, S. Shimojo, Y. Kitamura, and H. Otsuki, "Future Internet: Challenges in Virtualization and Federation," Telecommunications, 2009. AICT '09. Fifth Advanced International Conference on, pp.1-8, May.2009.

[4] J. Lee, S. Kang, Y. Lee, and J. Lee, " A Study on the Future Internet Requirement and Strategy in Korea," Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on, Vol. 1, pp. 627-629, Feb.2008.

[5] P. Stuckmann, R. Zimmermann, "European research on future Internet design," IEEE Wireless Communications, Vol. 16, Issue 5, pp. 14-22, Oct.2009.

[6] SDN white paper, https://www.opennetworking.org/sdn-resources/sdn-library/whitepapers

[7] A.C. Risdianto, E. Mulyana, "Implementation and analysis of control and forwarding plane for SDN," Telecommunication Systems, Services, and Applications (TSSA), 2012 7th International Conference on, pp. 227-231, Oct. 2012.

[8] K. Bakshi, "Considerations for Software Defined Networking (SDN): Approaches and use cases," Aerospace Conference, 2013 IEEE, pp. 1-9, Mar. 2013.

[9] Myung-Ki Shin, Ki-Hyuk Nam, and Hyoung-Jun Kim, "Software-defined networking (SDN): A reference architecture and open APIs," ICT Convergence (ICTC), 2012 International Conference on, pp. 360-361, Oct. 2012

[10] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: towards an operating system for networks," ACM SIGCOMM Computer Communication Review, Vol.38, Issue 3, pp. 105-110, July2008.

[11] POX, http://www.noxrepo.org/pox/about-pox/

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

[13] Floodlight, http://floodlight.openflowhub.org/

[14] OpenFlow, http://www.openflow.org/

[15] OpenFlow white paper, http://www.openflow.org/documents/openflow-wp-latest.pdf

[16] OpenFlow Switch Specification, http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf

[17] Wun-Yuan Huang, Jen-Wei Hu, Shu-Cheng Lin, Te-Lung Liu, Pang-Wei Tsai, Chu-Sing Yang, Fei I Yeh, Jim Hao Chen, and Mambretti, J.J., " Design and Implementation of an Automatic Network Topology Discovery System for the Future Internet Across Different Domains," Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on, pp. 903-908, Mar. 2012.

[18] Wun-Yuan Huang, Jen-Wei Hu, Ta-Yuan Chou, Te-Lung Liu, "Design and Implementation of Real-Time Flow Viewer across Different Domains. " Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on, pp. 619-624, Mar. 2013.

[19] ENVI, http://www.openflow.org/wk/index.php/Extending_ENVI

[20] LAVI, http://www.openflow.org/wk/index.php/LAVI