利用隧道技術實現虛擬路由器

游千冊　　王奕元　　侯廷昭
中正大學電機/通訊工程研究所及電信研究中心
m9425@cn.ee.ccu.edu.tw　dadai.cm91@gmail.com　tch@ee.ccu.edu.tw

摘要

路由器架構的演進，從傳統上將所有功能統一運作在一個作業系統上，漸漸轉變成分散式的路由器架構，依功能分為控制平面(Control Plane)與資料平面(Data Plane)。控制平面主要負責路由協定的運作，而資料平面主要負責處理大量的資料封包轉送。分散式的路由器架構雖然在效能與彈性化上比傳統路由器好，但卻多出了控制與資料平面溝通的問題。此外，路由器的架構也慢慢朝向虛擬路由器(Virtual Routers)發展。本文主要提出使用現有隧道技術實現分散式虛擬路由器。設計並實作基於 Quagga 的控制平面架構，並且於 PC 環境下實現資料平面的功能，最後經由實際的 OSPF 路由環境來驗證虛擬路由器的功能。

關鍵詞：控制平面、資料平面、隧道技術、虛擬路由器

1. 前言

近年來，為了因應網路封包的大量成長，路由器的發展已經從一般傳統的中央處理架構到專門用來處理封包的分散式架構。在路由器中，依功能可分為負責路由協定運作的控制平面與負責資料封包轉送的資料平面，藉此可以使得資料平面能夠將軟硬體資源用來快速處理大量的封包，無須分心訊令與路由協定，讓網路於高速的情況下亦能得到理想的效能。但是若將控制平面與資料平面分開運作，或是使用一個控制平面來控制多個資料平面，勢必會造成一些原本沒有的問題。另一方面，路由器的架構也朝著虛擬路由器邁進，虛擬路由器主要是將一台實體路由器邏輯劃分成多個虛擬路由器，每個虛擬路由器有獨立的路由表、路由協定運作與參予路由的網路介面，可以將原本一個控制平面只管理一個資料平面，轉變成一個控制平面管理多個資料平面，充分利用控制平面的各種資源。

本篇論文主要目的為設計並實作出分散式的 OSPF 虛擬路由器，控制平面採用 Quagga，資料平面則以 PC 作為開發平台。第二節以及第三節主要說明如何建置分散式的 OSPF 路由器，說明使用隧道技術作為控制平面與資料平面的溝通設計以及在開發過程中控制平面與資料平面的各項議題。第四節則延伸第三節分散式 OSPF 路由器的設計，擴展控制平面 Quagga 的舊有功能，使其支援虛擬路由器的運作。第五節則實際建置 OSPFv2 與 OSPFv3 路由環境，驗證分散式虛擬路由器之各項功能。

2. 背景介紹

在此節，主要說明使用隧道技術來實現虛擬路由器相關的背景知識，首先針對目前廣泛使用的隧道技術做介紹，接著介紹 Quagga 的系統架構。

2.1 Tunnel 技術簡介

Tunnel，中文譯為隧道，在網路中顧名思義是用來連接兩個原本無法溝通的區域，包括性質不相同(IPv4 與 IPv6 的溝通)、私人網域(Private Network)等，或是為了網路安全及移動性(Mobility)而建立的隧道。目前隧道技術的應用即是為了滿足不同目的而發展出來，以下將對各種隧道技術做介紹：

- IP encapsulation within IP (IP-in-IP)：它是由 IETF Mobile IP 組織所發展出來，目的是為了讓 Mobile Hosts (MH) 能夠在 Mobile IP 的架構下與 Home

Agents (HA)保持連線。在 Linux 作業系統中細分 IPIP 與 Simple Internet Transition (SIT)。在 Linux IP-in-IP 隧道技術中，外層使用的隧道協定為 IPv4 協定，因此隧道標頭也就是 IPv4 的標頭，內層(第三層)協定可以是 IPv4 (即為 IPIP)或是 IPv6 (即為 SIT)。

- Generic Routing Encapsulation Protocol (GRE)：一開始是 Cisco 所提出的隧道協定，由於 IP-in-IP 這種隧道技術只侷限於外層和內層協定均為 IP，因此發展出 GRE 這種通用的隧道協定。RFC 1701 定義了廣泛的 GRE 隧道協定，其內層和外層的協定可以為任何的網路協定。RFC 1702 則是定義了狹義的 GRE，目標鎖定在普遍使用的 IPv4 網路中，外層協定為 IPv4，內層則可以是任何協定。
- IP Security Protocol (IPSec)：Internet 發展至今，在設計上並沒有把安全性考慮在內，IETF 組織定義了一套將多種安全(Security)協定加入 TCP/IP 網路的方法，安全協定包括了 Authentication Header (AH)、Encapsulating Security Payload (ESP)、Security Associations (SAs)、金鑰管理與安全機制演算法。
- Layer 2 Tunnel Protocol (L2TP)：我們把 Microsoft 所發展出來的 Point to Point Tunnel Protocol (PPTP)與 Cisco 發展出來的 Layer 2 Forwarding (L2F)歸類為 L2TP 的一種，應用於 Internet-based 遠端存取服務。

2.2 Quagga in Control Plane

Quagga 為一套提供許多路由軟體套件的集合，支援以 TCP/IP 為基礎的路由軟體，目前支援多數常見的路由協定。除了傳統的 IPv4 的路由協定，Quagga 亦支援 IPv6 的路由協定。

傳統的路由軟體是以單一程式去提供所有路由協定的功能。Quagga 則採取不同的方法，它集合不同的路由協定，共同去建構核心路由表。在 Quagga 的軟體架構下，可擁有不只一個路由軟體，Zebra 角色則為核心路由的管理者。OSPF6d 為處理 OSPFv3 協定的常駐程式，依此類推。Zebra 常駐程式則負責寫入核心的路由表，以及在不同路由協定之間交換路由資訊。Quagga 系統架構可以輕易加入新的路由常駐程式，且不會影響其他的路由協定。Quagga 軟體架構中，將不同的路由軟件模組化、具備延展性特性、可維護性，此外亦提供可攜性的系統架構。

像 Quagga 這樣一個多程式架構，每一個路由常駐程式擁有自己的設定檔與終端介面 (Terminial Interface)，網路管理者可透過該路由協定的終端介面與設定檔設定相對應的路由協定，但這對於管理者來說不是很方便。為了解決這一問題，Quagga 更近一步提供一個整合的使用者介面 shell 稱為 Vtysh。網路管理者可透過 Vtysh 切換至任何一個路由常駐程式，針對相對應的路由協定作設定。

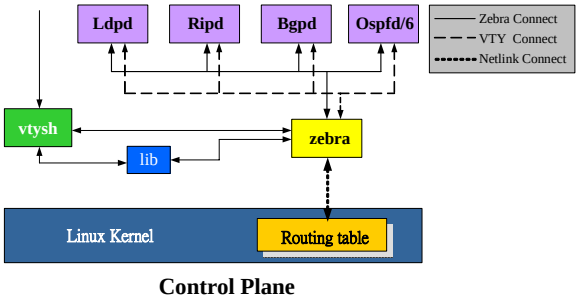


圖 1 Quagga 系統架構圖

圖1為Quagga所提出的系統架構。首先，zebra connect的連線為zebra透過Unix domain socket 與不同路由常駐程式作溝通的方式。Netlink connect 的連線為Zebra常駐程式透過Netlink socket 將路由資訊寫入核心路由表中。最後VTY connect的連線為Vtysh透過UNIX domain socket 對不同的路由常駐程式下達設定的指令方式。

2.3 OSPF 路由協定

OSPF 是由 IETF 的 IGP 工作小組為 IP 網路開發的路由協議。OSPF 創建的原因是到了八零年代中期，RIP 不能服務於大

型網絡的缺陷愈來愈明顯。OSPF 有兩個主要的特性。首先該協議是開放的，公佈的 OSPFv2 規範是 RFC2328 ，應用於 IPv4 環境，OSPFv3 規範是 RFC2740 ，應用於 IPv6 環境。另外一個特性是 OSPF 使用 Dijkstra 演算法來計算路由。

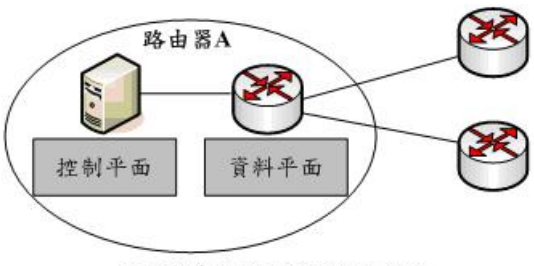
OSPF是個鏈接狀態路由協議，在同一層的區域內與其它所有路由器交換鏈接狀態公告(Link State Advertisement)訊息。OSPF的LSA中包含連接的介面、使用的metric及其它的訊息。OSPF路由器收集鏈接狀態訊息，並使用SPF演算法來計算到各節點的最短路徑。

3. 運用隧道於控制平面和資料平面溝通的考量

本節主要介紹在分散式路由器的架構下，如何利用隧道技術作為控制平面和資料平面溝通的橋樑，以及說明控制平面和資料平面必須有哪些功能。接續說明控制平面和資料平面在設計上的考量，並闡述控制平面如何把路由更新資訊傳遞給資料平面。

3.1 架構設計

傳統的路由器採用控制平面與資料平面皆在同一台機器上，因此共用一些硬體資源。所謂的控制平面與資料平面分離，如圖 2 所示，主要是把一台路由器依照功能區隔開來，控制平面負責路由協定的運作、路由表的建立、介面資訊的維護等，而資料平面專責封包的轉發，將控制平面與資料平面分隔開來使用不同的硬體，可以增加效率以及封包處理速度。



控制與資料平面分離的路由器

圖 2 控制與資料平面分離的路由器

然而，控制平面與資料平面是分別的兩台機器，當然可以用乙太網路來連接，互相傳送封包當然也沒問題，其實所謂的溝通，是指以下兩件事：

- 控制平面能模擬資料平面的網路介面
- 資料平面能正確地轉送路由封包

為了達到上述的兩件事情，我們提出使用隧道技術來完成。如圖 3 所示，OSPF 路由器 A 是由控制平面與資料平面組成，並分別由資料平面上的 eth0 與 eth1 介面連結 OSPF 鄰居路由器 B 與 C。因此對路由器 A 而言，參與 OSPF 路由的網路介面是資料平面上的 eth0 與 eth1。控制平面與資料平面透過乙太網路連結，而我們在控制平面與資料平面之間建立兩條 IP-in-IP 的隧道，建立兩條隧道後，會在控制平面增加兩個隧道介面分別為 tun1 與 tun2，此時我們可以用這兩個隧道介面分別來模擬資料平面上的 eth0 與 eth1 介面，例如 IP 位址，這樣的作法可以滿足上述控制平面與資料平面溝通要點的第一項。

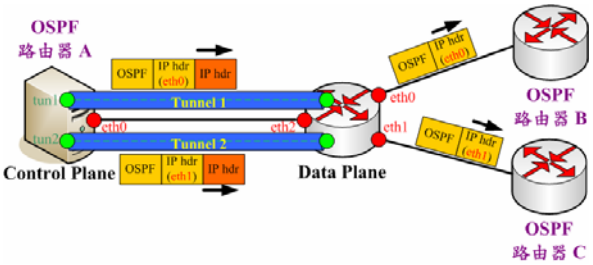


圖 3 利用隧道作為控制與資料平面的溝通橋樑

控制平面與資料平面溝通要點的第二項，關於路由封包的轉發功能，這點我們也必須使用隧道來達成。當我們使用隧道介面來模擬資料平面的真實介面時，會把 IP 及所屬的網路區段設定成一樣，如果不使用隧道來進行控制平面與資料平面的溝通，會使得兩台機器之間有 IP 衝突的問題，因此必須透過隧道增加一層隧道標頭來隱藏原本的 IP 位址，才能將封包傳遞給資料平面。如圖 3，OSPF 路由封包由控制

平面透過隧道介面經由真實介面傳出，封裝形式最內層為 OSPF 資料，接著為內層 IP 標頭，其中來源地位址為資料平面上的 eth0 或 eth1 位址，最後，封裝上外層 IP 標頭經由真實介面傳送出去，外層 IP 標頭的來源地與目的地位址分別為 Tunnel 1 與 Tunnel 2 的本地端和遠端位址。當資料平面經由真實介面(圖 3，資料平面的 eth2 介面)收到由控制平面送來的封包後，會解析外層 IP 標頭，來區分是 Tunnel 1 還是 Tunnel 2 的封包，解封裝外層 IP 標頭後將其傳送給資料平面上的隧道介面，在資料平面上我們可手動設定由 Tunnel 1 送來的路由封包從 eth0 介面送出，Tunnel 2 送來的封包則由 eth1 送出，因此對於 OSPF 鄰居路由器 B 和 C 來說，會以為收到的路由封包是由資料平面所送出的封包，但其實卻是由控制平面交給資料平面轉遞的封包。

以上敘述了利用隧道技術來建立控制平面和資料平面之間的溝通，接下來要針對控制平面和資料平面必須具備的功能做介紹。

- 控制平面 (Control Plane)
  - 必須與資料平面建立數條 IP-in-IP 的隧道。
  - 利用隧道介面來模擬資料平面的真實路由網路介面。
  - 路由常駐程式負責路由協定的運作，並利用隧道介面來計算正確的路由。
  - 負責資料平面的路由表維護。
- 資料平面 (Data Plane)
  - 必須與控制平面建立數條 IP-in-IP 的隧道，作為反向路由封包的傳遞路徑。
  - 轉送 OSPF 路由封包。
  - 轉送一般資料封包。
  - 接收控制平面傳來的路由更新資訊，並將其寫入核心路由表中。

利用圖 3 的架構，雖然能夠增加路由器處理資料封包的效率，但在實作上，控制平面以及資料平面各自都會有必須克服的議題。以下先敘述控制平面上所遇到的議題，再來討論資料平面上的議題。

3.2 控制平面的議題

3.2.1 控制平面與鄰居路由器對 OSFP 環境認知的差異

如圖 3 所示，OSPF 路由器 B 與 C 所管理的路由介面處於廣播環境中，因此路由器 B 與 C 會試圖與鄰居路由器 A 交換資訊來選舉 DR 和 BDR。但因為我們使用隧道介面來模擬資料平面對外的真實網路介面，而隧道介面屬於點對點的裝置，因此在控制平面上的路由常駐程式(OSPFd 或 OSPF6d)會認為這是屬於點對點的網路環境而不做 DR 與 BDR 的選舉。因此我們必須修改路由常駐程式，當路由常駐程式從核心讀取介面資訊後，我們必須修改隧道介面的參數，使其成為一般的乙太網路裝置型態，這樣的修改是欺騙 OSPF 路由常駐程式，使其認為隧道介面是一般的乙太網路介面而能正常運作。經過修改隧道介面的 flags 與 hw_type 參數之後，我們就能解決控制平面與外部鄰居路由器對於 OSPF 環境認知差異的問題。

3.2.2 MTU 不匹配的差異

對於資料平面外部的路由器而言，參與路由的介面為乙太網路裝置，MTU 值為 1500 位元組，控制平面必須使用隧道介面計算路由，而隧道介面因為多了一層隧道標頭的封裝，因此能傳輸的資料量大小會較乙太網路裝置小，就會產生 MTU 不匹配的差異，但是這樣的差異並不會影響路由封包的傳遞。真正的問題在於 OSPF 路由常駐程式會利用資料庫描述(DD)封包來交換介面 MTU 的資訊，並且檢查 MTU 值是否相同，因此 MTU 不匹配的情形會導致程式無法正常運作。由於這樣的差異是在我們預期之內，因此可以忽略程式中對於 MTU 的檢查，使程式能順利運作。

3.2.3 OSPFv2 External Route 的議題

在 OSPF 協定中，路由器會與 DR 交換 LSA (Link State Advertisement)，LSA 又可分成許多型態，其中有一項 LSA 稱為 AS-External LSA。我們發現，當路由器 A 與外部路由器 B 與 C 交換完 LSA 後，路由器 B 與 C 仍然會知道控制平面和資料平面之間區段的存在。此原因在於控制平面的 OSPFd 在初始化的過程，會透過 Zebra 向核心讀取核心路由表，而核心路由表中，亦存在此用來做控制平面與資料平面溝通的區段。由於此區段並沒有加入 OSPFd 的路由設定檔中，因此 OSPFd 會利用 AS-External LSA 來散佈控制平面和資料平面之間的網路區段給外部路由器 B 與 C 知道。這個問題最根本的解決方法，就是不讓 OSPFd 知道真實介面的存在。

3.3 資料平面的議題

3.3.1 轉送群播(Multicast)位址的封包

OSPF 屬於群播(Multicast)路由協定，參予 OSPF 運作的路由器，都必須加入一個群播位址來與其它 OSPF 路由器通訊。因此資料平面需要轉送目的地為群播位址的封包，例如 Hello 封包。資料平面會從隧道 1、隧道 2、eth0 與 eth1 介面收到同樣目的地為群播位址的封包，因此資料平面無法經由查找路由表而得知封包適當的導出介面。我們將使用一套軟體名為 Iptables，用來設定轉送的規則，使資料平面能正確轉送目的地為群播位址的封包。

3.3.2 IPv6 環境中轉送 Link-Local 位址的封包

在 IPv6 環境中，定址的方式分成三種，而 OSPFv3 使用 Link-Local 位址與區段上的路由器溝通，目的地為 Link-Local 位址的封包本來就不應該被路由器轉送，因此資料平面無法將外部路由器送來的路由封包轉送給控制平面處理，也無法將控制平面傳來的路由封包轉送給外部路由器。除了轉送 Link-Local 位址的封包有問題外，在 OSPFv3 環境中，當然也會面臨轉送群播位址的困難。同樣的，我們使用 Iptables 軟體，使資料平面不用查找路由表而可以依照我們設定的規則進行轉送封包的動作。

3.3.3 Iptables/Ip6tables (Patch ROUTE Target)

爲了解決資料平面必須能轉送群播位址的路由封包以及在 IPv6 環境中轉送 Link-Local 位址的封包，因此我們的解決方式是使用 Iptables/Ip6tables 軟體(Patch ROUTE Target)，針對所有流經資料平面的封包，在路由判斷前(PREROUTING)設定攔截規則與導出介面，符合規則的路由封包就能從適當的介面導出。Iptables 原本的功能是在 IPv4 的環境中建立防火牆，根據事先設定好的防火牆規則去處理每一個進來的封包，做出相對應的動作。隨著 IPv6 的發展，適用於 IPv6 環境中的 Ip6tables 也隨之出現，Ip6tables 基本的運作原理與 Iptables 類似。

3.3.4 TTL 的議題

資料平面在轉送路由封包的時候不應該遞減 TTL 值，由於我們使用 Iptables/Ip6tables 轉送封包，因此封包在資料平面做路由判斷前就從設定的介面導出，而不會經過核心的 TCP/IP 協定堆疊(Protocol Stack)，因此遞減 TTL 的工作是由 Iptables 負責。藉由修改 Iptables 程式碼，忽略 ip_decrease_ttl 函式的執行，即可讓資料平面對於透過 Iptables 轉送的封包，不會遞減 TTL 值，但對於一般經過 Linux TCP/IP Protocol Stack 轉送的資料封包仍會遞減 TTL 值。

3.4 路由更新資訊的傳遞

在控制平面和資料平面分離的路由器架構下，控制平面必須負責 OSPF 協定的運作以及負責資料平面路由表的維護。控制平面的路由常駐程式(OSPFd/OSPF6d)利用隧道介面計算好路由後，會透過 Zebra

更新控制平面的核心路由表。一筆路由的更新資訊，從 OSPFd/OSPF6d→Zebra→Linux 核心的傳遞過程中，我們必須尋找適當的切入點，將路由更新資訊傳遞給資料平面。如圖 4，從 OSPFd/OSPF6d 切入取得路由更新資訊，利用 TCP socket 將路由更新資訊傳遞給 Zebra 時，複製一份傳遞給資料平面，這種方式，取得的路由更新資訊將只有 OSPFd 與 OSPF6d 所送出的路由更新。

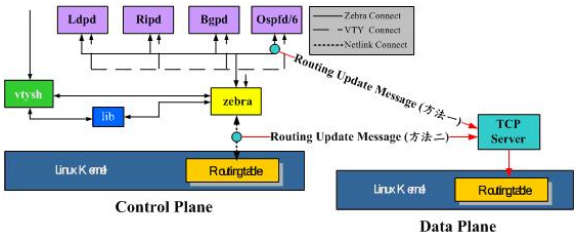


圖 4 路由更新資料的取得

控制平面取得路由更新訊息後，必須將訊息傳遞給資料平面，因此必須預先在資料平面與控制平面之間建立用來傳遞路由更新的 TCP socket 連線，資料平面為接收資料的 Server 端，而控制平面為傳送資料的 Client 端。控制平面的 TCP Client 必須將路由更新資訊(包括 prefix、prefix 長度、導出介面、next hop 等)傳遞給資料平面，資料平面的 Server 接收訊息後，必須分析訊息並修改導出介面後，利用 system call 使用 ip route 的指令將訊息寫入資料平面的路由表中。修改導出介面的原因在於，控制平面是透過隧道介面來計算路由，因此路由更新訊息中的導出介面(OIF)也將是隧道介面，但對於資料平面而言，對外路由介面是屬於真實介面(非隧道介面)，因此資料平面收到訊息後，必須將訊息裡的導出介面(隧道介面)對應到資料平面上的一個真實介面，修改此筆路由訊息的導出介面後，才能新增此筆路由更新。

4. 虛擬路由器

這一節主要介紹虛擬路由器的設計與實作的方法，主要延伸第三節所闡述的內容，基於控制平面與資料平面分離的路由器架構下，如何在控制平面運行多個路由常駐程式(OSPFd/OSPF6d)，來管理不同的資料平面。首先我們會介紹控制與資料平面分離的虛擬路由器架構，接續說明在此架構下，控制平面的設計考量與實作上的議題。最後，闡述如何使用 GRE 隧道來增加效能。

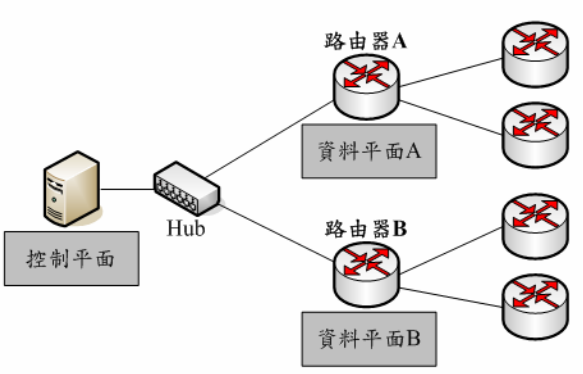


圖 5 虛擬路由器架構

圖 5 為虛擬路由器架構示意圖。在第三節中我們闡述了功能分離的路由器架構下之各項議題與實作方式，基於此架構下，我們繼續延伸控制平面的功能來完成虛擬路由器。在虛擬路由器的架構下，一個控制平面可以管理多個資料平面，控制平面與資料平面之間的連結，可以如圖 5 使用集線器或是橋接器等網路裝置;若使用跳線來連接控制平面與資料平面，其好處是可以做到資料的完全分離，但控制平面就必須有多張網路卡來連接資料平面，造成額外的負擔。控制平面與資料平面之間的溝通，仍然可以使用隧道技術。如圖 6 所示。

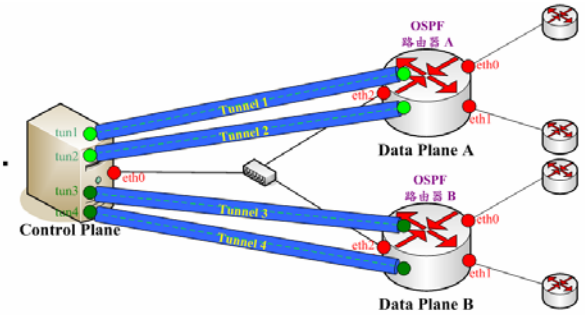


圖 6 利用隧道技術實現虛擬路由器

利用隧道技術實現控制平面與資料平面的溝通機制與第三節談論的一樣，在控制平面與資料平面 A 之間建立兩條隧道，於控制平面產生兩個隧道介面(tun1 與 tun2)分別模擬資料平面 A 的 eth0 與 eth1 介面。在控制平面與資料平面 B 之間建立兩條隧道，於控制平面產生兩個隧道介面(tun3 與 tun4)分別模擬資料平面 B 的 eth0 與 eth1 介面。因為控制平面必須負責兩台資料平面的路由維護工作，因此必須運作兩個 OSPF 路由協定常駐程式分別管理兩個資料平面。對於控制平面、資料平面必須達到的功能都與第三節所描述的一樣。

虛擬路由器的實現，必須在一台機器上邏輯分割成多台虛擬路由器，其中包括路由表的獨立，介面的獨立以及協定運作的獨立。在圖 6 的環境中，資料平面 A 與 B 各自擁有轉送資料的路由表，而控制平面也利用不同的隧道介面來模擬資料平面 A 與 B 的真實路由介面，最終的關鍵就是在控制平面上同時運作兩套 OSPF 路由協定常駐程式且不互相影響。控制平面我們仍然採用 Quagga 路由軟體來實現，軟體架構如圖 7 所示。在各種考量之下，我們決定使用架構二來實現虛擬路由器。以下針對第二種架構來介紹。

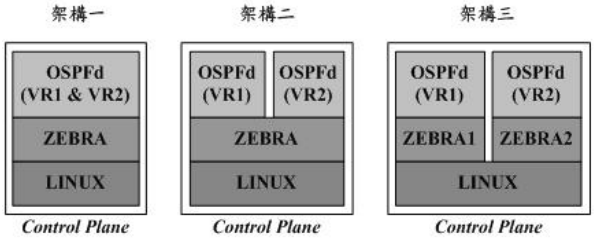


圖 7 控制平面的軟體架構

在架構二的軟體架構下，Linux 核心只會有一份路由表，這份路由表只會提供 Zebra 在初始化建立 RIB 時使用。在 Quagga 中只會運行一套 Zebra 常駐程式，且 Zebra 只會維護一份 RIB。但會針對兩個資料平面而同時運行兩套 OSPF 路由常駐程式(VR1 與 VR2)，因此每一路由常駐程式有自己必須管理的介面、鏈結資料庫以及鄰居狀態的維護。由於只會運作一套 Zebra，因此每一路由常駐程式在初始化的過程中，必須過濾從 Zebra 送來的資訊。例如圖 6 的環境下，控制平面運作兩套 OSPF 路由協定常駐程式(VR1 與 VR2)，分別管理資料平面 A 與 B，對於 VR1 的路由常駐程式而言只需要知道 tun1 與 tun2 介面以及關於這兩個介面的路由資訊，因此必須在 VR1 初始化時做過濾的動作。架構二雖然效能沒有架構一來得好，但開發的難易度相較於架構一容易許多，且對系統的負擔也較架構三少，在考量功能面與開發難易度之下，我們最終選擇架構二作為虛擬路由器的控制平面軟體架構。

5. GRE 隧道技術的使用

控制平面必須與各個資料平面建立隧道，建立隧道的數目則依照每個資料平面的路由介面數目而定。例如圖 6 環境中，控制平面必須建立四條 IP-in-IP 的隧道分別與兩個資料平面溝通，當資料平面的路由介面增加，或者控制平面管理更多的資料平面時，隧道的數目將大大增加，而使用 IP-in-IP 隧道技術，隧道數目的增加將浪費控制平面與資料平面真實溝通介面的 IP 位址。圖 6，為了隧道的建立就必須配置兩個 IP 位址給控制平面的 eth0，同樣的也必須配置兩個 IP 位址給資料平面的 eth2 介面，來建立反向的隧道。一條點對點的隧道(雙向)會佔用兩個屬於溝通區段的 IP 位址，因此減少隧道的建立可以改善效能並減少 IP 位址的浪費。在 2.1 節中，簡單介紹過 GRE 隧道技術以及封裝方式，我們可以使用 GRE 隧道技術來減少控制平面與資料平面之間的隧道數目，以下將說明如何使用 GRE 隧道技術於虛擬路由器架構中。

GRE 隧道標頭的欄位如圖 8 所示，其中必要的欄位只有前 4 個位元組，其後的欄位的為選擇性的功能，前 4 個位元(C、R、K、S)分別對應到 Checksum、Routing、Key、Sequence Number 欄位，用來表示是否有啟動該功能，Offset 欄位伴隨著 Routing 欄位的出現，第五個位元(s)表示

Routing 欄位是否為 Strict Source Route，Recur 和 Flags 欄位預設值為 0，其後的 Version 欄位預設值也為 0，Protocol Type 欄位用來表示內層(Payload)使用的協定為何。我們使用 GRE 隧道技術來減少控制平面與資料平面之間的隧道數目，是利用 GRE 標頭中的 Key 欄位作為多工與解多工的功能，關於 GRE 的其它功能我們就不在此多做解釋。

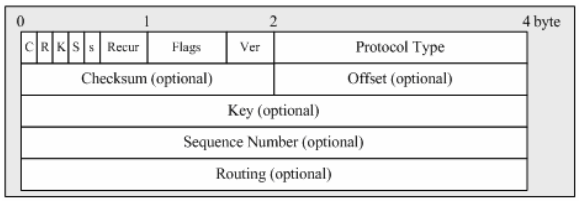


圖 8 GRE 標頭

在 Linux 系統中，建置 GRE 隧道，除了指定隧道的遠端位址與本地位址外，還可附加 key、ikey 與 okey 參數來設定 Key 值，其中 okey 為 outgoing key，用來設定 GRE 標頭的 Key 欄位值；ikey 為 incoming key，此值並不會隨著封包傳遞出去，而是設定從 GRE 隧道送來的 GRE 封包，Key 欄位必須與所設定的 ikey 值一樣，否則丟棄；如果只使用另一個參數 key (不設定 ikey 與 okey)，則會使 ikey 與 okey 皆為所設定的 key 值，在隧道另一端的鄰居，也必須設定相同的 key 值才可通訊。將此多工與解多工的功能應用於控制平面與資料平面分離的虛擬路由器架構下，將可減少隧道數目，範例如圖 9 所示。

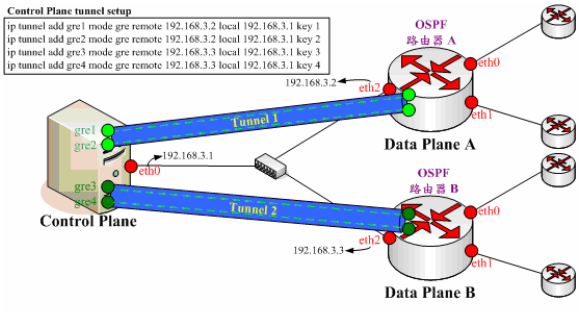


圖 9 GRE 隧道的使用

我們只需要配置一個 IP 位址於控制平面、資料平面 A 與資料平面 B 的真實溝通介面即可建立四個隧道介面，控制平面分別與資料平面 A、資料平面 B 建立一條隧道。在 Tunnel 1 隧道中，雖然 gre1 與 gre2 都有相同的遠端位址、本地位址，但可以利用 key 值的不同而區隔開。範例中，建立 gre1 使用的是 key 參數，代表 ikey 和 okey 都為 1，相同地，資料平面 A 在建立反向隧道與控制平面溝通時，key 值也必須設為 1。另外，管理人員也可將 ikey 與 okey 設為不同數值。控制平面在建立 gre3 與 gre4 隧道和資料平面 B 溝通時，key 值也不一定要為 3 與 4，甚至可以與 gre1、gre2 設定的 key 值一樣，因為 gre3 與 gre4 的隧道遠端位置與 gre1、gre2 不一樣，所以 key 並不會有衝突。簡單地說，key 的影響範圍只會在某一 GRE 隧道中而已。使用 GRE 隧道(2 條)，的確可以減少原本使用 IP-in-IP 隧道(4 條)的隧道數目，藉由 key 值做到多工與解多工的功能。

6. 結果驗證

圖 10 是網路環境的拓樸。控制平面管理兩台資料平面，路由器 A 和路由器 B。路由器 A 連結路由器 C 和路由器 D；路由器 B 則連結路由器 E。

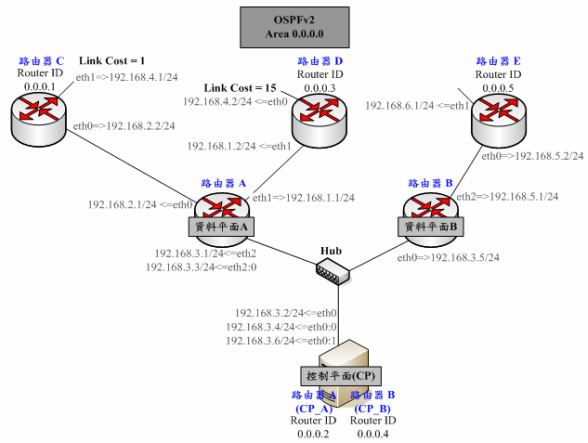


圖 10 OSPF 實驗環境

控制平面與資料平面之間採用 IP-IP 隧道溝通，由於路由器 A 與路由器 B 之間並沒有連線，因此可以將此路由環境分成

兩個群組(路由器 A、C 與 D 一組；路由器 B 與 E 一組)，控制平面與資料平面的外部路由器 C、D、E 皆使用 Quagga 軟體來運作路由協定，當所有 OSPF 路由器完成訊息交換後，可以預期路由器 A 的控制平面(CP_A)將新增一筆 192.168.4.0/24 的路由且下一轉運點為路由器 C，因為路由器 C 通往 192.168.4.0/24 網路區段的 Cost (1)較路由器 D 通往該區段的 Cost (15)來得小，因此選擇最佳路徑為通過路由器 C 到達該區段。而路由器 B 的控制平面(CP_B)將會新增一筆到達 192.168.6.0/24 網路區段的路由。

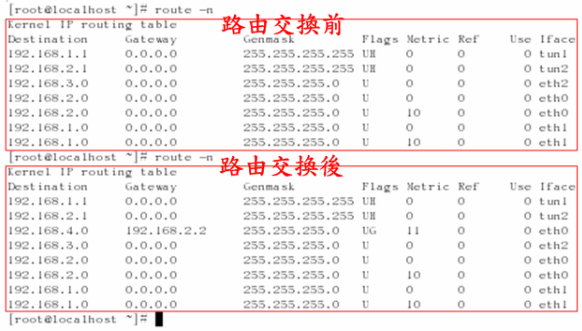


圖 11 路由器 A 之路由表(資料平面)

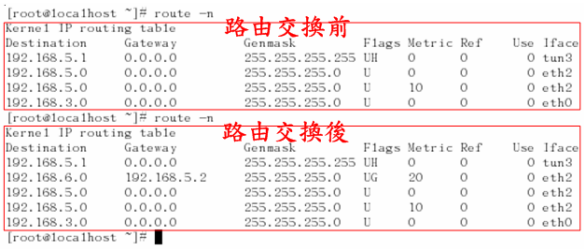


圖 12 路由器 B 之路由表(資料平面)

路由器 A 的路由表(資料平面 A)，如圖 11 所示，路由交換後新增一筆通往 192.168.4.0/24 網段的路由且導出介面為 eth0，代表下一轉運點為路由器 C，最佳路徑符合預期結果。

路由器 B 的路由表(資料平面 B)，如圖 12 所示，路由交換後新增一筆通往 192.168.6.0/24 網段的路由且導出介面為 eth2，代表下一轉運點為路由器 E，符合預期結果。

GRE 隧道的驗證也使用圖 10 的網路環境，因為使用 GRE 隧道取代 IPIP 隧道，因此資料平面 A、資料平面 B 與控制平面溝通的真實介面都只需要配置一個 IP 位址，達到節省 IP 位址的優點。控制平面的 gre1 與 gre2 隧道介面雖然有同樣的遠端與本地位址，但可藉由 ikey 與 okey 來區分不同的介面。控制平面上路由器 A 與路由器 B 的 OSPFd 以及 Zebra 設定檔與之前說明 OSPFv2 虛擬路由器測試時極為相似，只要把改變介面名稱即可(例如:tun1 改為 gre1)。

圖 13 為路由器 C 送出的 Hello 封包，經由資料平面 A 透過 GRE 隧道送至控制平面。可以發現在 GRE Header 中的 Key 值為我們在資料平面 A 上設定 gre2 的 okey 值且為控制平面設定 gre2 的 ikey 值，因此控制平面可以接收。

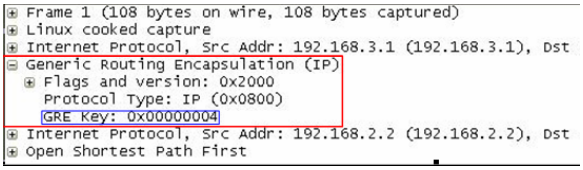


圖 13 GRE 封包觀測

資料封包轉送測試中，一開始路由器 C 會發出 ICMP Ping 的資料封包，目的地為 192.168.7.1(路由器 D)，在路由訊息還未完全交換前，資料平面 A 的路由表並沒有 192.168.7.0/24 區段的資訊，因此無法轉送 ICMP 資料封包，當控制平面傳遞路由更新至資料平面 A 後，資料平面 A 更新其路由表，並順利轉送 ICMP 封包。

7. 結論

我們提出利用隧道技術實作控制平面與資料平面溝通的方式，成功的使用現有的 IP-in-IP 隧道技術作為控制平面與資料平面溝通的橋樑。以此為基礎，我們接著設計並實作基於 Quagga 的虛擬路由器控制平面架構，並且於 PC 環境下實現虛擬路由器資料平面的功能。控制平面方面，延伸 OSPFd 與 OSPF6d 的功能，使其支援多個 OSPF 路由常駐程式的運作，並且可

以使用隧道介面模擬資料平面上的真實路由介面。資料平面方面，在虛擬路由器架構的設計上我們考量到彈性化，因此不論是 PC 或是網路處理器等，只要能達到三個功能(支援隧道技術、轉送路由封包與作為路由更新的 TCP server)即可成為資料平面，對於虛擬路由器的控制平面將不需要做任何變動。

參考文獻

[1] C. Perkins, "IP Encapsulation within IP", RFC 2003, October 1996.

[2] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)", RFC2409, November 1998.

[3] Jacco de Leeuw, "Setting up an L2TP/IPsec VPN between Windows/Mac and Linux", www.jacco2.dds.nl/.

[4] J. Aweya, On the design of IP routers Part 1: Router architectures. Journal of Systems Architecture, 46(2000) 483~511, 2000.

[5] J. Moy, "OSPF Version 2", RFC 2328, April 1998.

[6] K. Ishiguro, et al., "A routing software package for TCP/IP networks (Quagga 0.99.4)", www.quagga.net, July 2006.

[7] L. Yang, et al., "Forwarding and Control Element Separation (ForCES) Framework", RFC 3746, April 2004.

[8] R. Coltun, et al., "OSPF for IPv6", RFC 2740, December 1999.

[9] S. Hanks, et al., "Generic Routing Encapsulation (GRE）", RFC 1701, October 1994.

[10] S. Hanks, et al., "Generic Routing Encapsulation over IPv4 networks", RFC 1702, October 1994.

[11] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol", RFC2401, November 1998.

[12] S. Kent and R.Atkinson, "IP Authentication Header", RFC2402, November 1998.

[13] S. Kent and R.Atkinson, "IP Encapsulating Security Payload (ESP)", RFC2406, November 1998.

[14] W. Townsley, et al, "Layer Two Tunneling Protocol (L2TP)", RFC2661, August 1999.