# 前言

大多數歐萊禮讀者的第一個問題，便是作者為什麼選擇那個封面動物？針對這本書，讀者可能會問：「為什麼選鴨子？」。好吧，其實作者最先選的是裝飾了彩虹絲帶閃閃發光的獨角獸。

這樣的回答肯定會引得大家發笑（相信你也笑了幾聲）。獨角獸的形象也反應了許多經驗豐富的網路專業人士對 SDN（Software Defined Networks，軟體定義網路）的共同感覺。雖然作者認為這種感覺在某些方面是合理的，但這隻獨角獸的確有血有肉，而不僅僅是個傳說。

那麼言歸正傳，對於開頭的問題，作者有個更好的答案：人們在水面上見不到鴨子的全部動作。鴨子[1]游動時大多數的動作都在水下完成，而這些動作是無法輕易觀察到的。

1. 真正的答案是本書的其中一位作者十分喜歡鴨子，他在家庭農場裡也飼養了一些番鴨。

強健的腳掌在水面下滑動著，推動鴨子前進。這與軟體定義網路在很多方面有著相似之處。

SDN 的表象可能導致觀察者輕率地產生如下看法。首先，定義 SDN 是什麼或可能是什麼，是很多機構為了搶救他們的商業計畫或振興他們的標準化組織而作出的瘋狂嘗試。其次，SDN 僅僅是一種品牌的重塑，為已有的產品附上它們所沒有的「魔力」。很多廠商宣稱他們四五年前開發的產品是 SDN 的源頭，所以從那以後他們所做的也都是 SDN。

順著這種說法，重新給所有東西貼上 SDN 標籤的行為，以及近三、四年來大量 SDN 創業公司的宣傳鼓噪，都進一步強化了人們的這種負面看法。

如果觀察者們因為抱有網路信仰與政治的成見而對 SDN 不屑一顧，那麼 SDN 看上去就只是一個飄忽不定的想法。

看懂 Gartner 公司的技術成熟度曲線[2]，就可以理解新技術發展的遠景，並且可以看到該曲線所預測的新技術發展所處的階段。

正是當前這場 SDN 運動與上述成熟度曲線表現出的相同特徵，促使我們為這頭光環籠罩的獨角獸極力遊說，只為說明一點：我們看到了 SDN 的與眾不同之處。

兩年多以來，筆者參與許多相關的客戶會議、論壇和產業協會以及標準化組織的會議，討論各種 SDN 相關話題，還跟很多 SDN 創業公司、支持者及早期應用者一起工作。這些經歷使作者相信看似平靜的水面下發生著一些值得關注的事情。那就是推動 SDN 朝設定目標前進的大量努力，而這個目標就是為網路和使用網路的應用提供最佳的營運效率和靈活性。

有確鑿的證據顯示，SDN 最終還是引起了一場討論，這場討論涉及網路的可程式設計性、控制模型、面向網路的應用介面現代化和上述事物事實上的開放性。

有鑑於此，雖然資料中心內虛擬化所引起的可管理網路終端浪潮是現在 SDN 的主要推動力，但 SDN 並沒有被限制在諸如資料中心這種單一的網路領域。SDN 並沒有被局限於單一的客戶類型（如研究 / 教育），或者單一的應用（如資料中心服務編排），甚或單一的協定 / 架構（如 OpenFlow）。SDN 也沒有被單一的架構模型所限制（如集中式的控制器以及一組被操控的交換機的模型）。希望讀者可以從本書中看出這一點。

2. 請參考 *http://www.gartner.com/technology/research/methodologies/hype-cycle.jsp*

第 1 章

# 簡介

事實上，直到幾年前「**儲存／運算／網路**」等資源的管理仍刻意分開，甚至用來管理這些資源的系統也是分開的。監控管理這些系統和資源進行互動的應用，都被 IT 部門以非常複雜的存取原則、系統…等安全機制所管控著。只有等到資料中心需要導入較為廉價的運算能力、儲存空間和網路之後，此時才不得不將這些網路元件放在一起考慮。也正因為這種改變，才讓管理和操作這些資源的應用比以前更加緊密的整合在一起。

「**資料中心**（*Data Center*）」，一開始被設計用來分隔實體上的傳統運算單元（例如：PC 伺服器），以及所使用的儲存及網路。因此，資料中心的運算資源往往用於特定服務，例如：郵件伺服器、資料庫伺服器…等，以便為使用者提供各式各樣的服務。在以前，企業或組織當中通常運作數千台（或更多）桌上型電腦，並且由專用的伺服器群來提供相關服務。隨著時間的演變，眾多的實體伺服器紛紛遷移至資料中心當中，主要目的是為了管理上的便利性，其次則是硬體資源更容易分享給使用者。

但是，大約在 10 年前發生一個有趣的變化。一家叫做 VMware 的公司發明了一項有趣的技術，在一個熱門的 Linux 發行版本當中，同時運作一個或多個客體作業系統（例如：Windows）。VMware 的作法是建立應用程式，以便整合虛擬環境（例如，虛擬網路卡、BIOS、音效卡和視訊裝置…等），然後在多台 VM 虛擬主機之間分配資源，並且將 VM 虛擬主機進行隔離。這種監控程序被稱為 VM 虛擬主機「**管理程序**（*Hypervisor*）」。

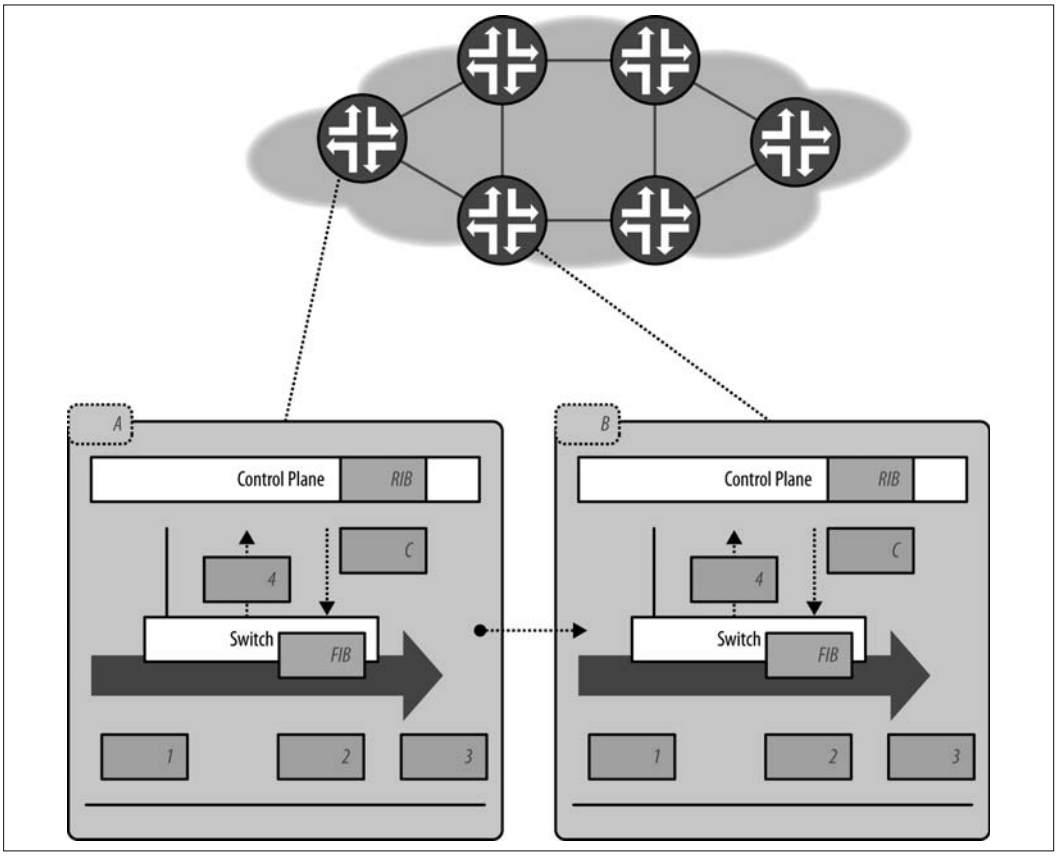


圖 2-2：一個典型網路的控制平面和資料平面

在實際環境中，剛才所討論的網際網路控制平面是介於 L2 和 L3 控制平面的某種組合。因此，L2/L3 網路組成控制平面的協定出現和進化也不奇怪。事實上，網際網路的發展，就是由於在這 L2 網路方面協定的進化，以及硬體廠商掌握高擴充性和高可用性的技術來達成協定的進化。

L2 控制平面，專注於硬體或實體層位址如 IEEE MAC 位址。建構 L2 控制平面，則是為了使用網路層位址如 IP 位址。在 L2 網路環境當中，將會圍繞在 MAC 位址的學習、保證無迴路拓撲的機制（例如，大多數讀者所熟悉的「擴充樹協定（*Spanning Tree Protocol*，*STP*）」），以及大量的 BUM（廣播、單點傳播、多點傳播）洪水流量，都會造成可擴展性方面的挑戰以及相關的功能局限。為了解決這些問題及其他問題，已經針對標準的 L2 控制協定進行多次的更新。最值得注意的部份是，這些協定中包括了 IEEE 的 SPB/802.1aq 以及 IETF 的 TRILL。

在稍微低一些的層級當中，某些類型的網路有一些輔助控制流程，提供建構更強大的控制平面所需要的資訊。這些流程所提供的服務，包括連結可用性或品質資訊的驗證 / 通知、鄰居探索和位址解析。

這些服務當中，有些會非常嚴格的要求效能（為了縮短事件檢測時間），所以不管控制平面採用的是什麼原則，它們幾乎不約而同的，在本地端資料平面設備中達成（例如，運作、管理、維護 OAM）。在圖 2-3 當中，說明由多種路由通訊協定，以及 RIB 到 FIB 所組成的控制平面核心來描繪這一點。請注意，這裡並非要硬性規定控制 / 資料平面所處的位置，只是說資料平面應該放置於卡板（圖 2-3 當中的 LC 框線內）當中，而控制平面則應位於路由處理器內（圖中 RP 框線內）。

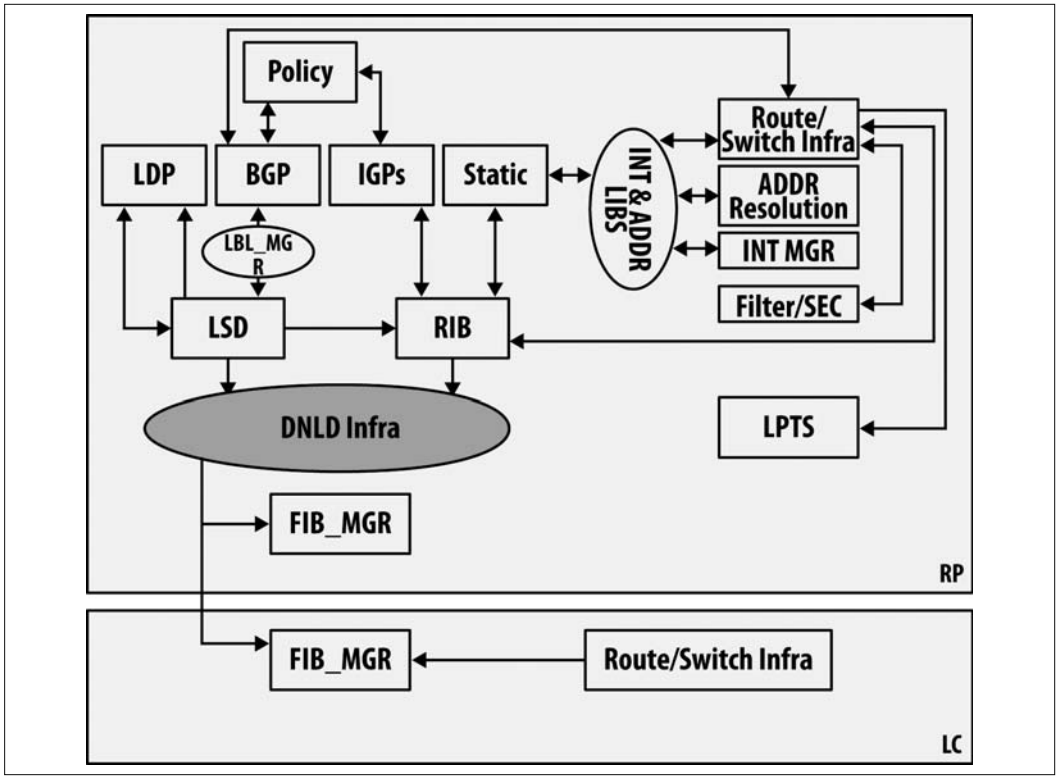


圖 2-3：典型網路設備的控制平面和資料平面

## 資料平面（Data Plane）

資料平面透過一系列連結層級的操作，來處理（纜線、光纖或無線媒介）的封包，這些操作通常包括封包的收集和基本的完整性檢查[3]。資料平面查詢動作，由控制平面所設計的 FIB 資料表（在某些機制中，可能有多個 FIB 資料表）來處理一個格式正常的封包，有時也稱為資料處理的快速路徑，因為它除了利用已經設計好的 FIB 資料表，以快速識別封包的目的地位址之外並不需要額外的操作。在此流程中唯一的例外是，當封包無法比對這些規則時，例如，檢測到一個未知的目的地位址時，這些封包就會被轉送到路由處理器中，如此一來控制平面就能進一步利用 RIB 來處理它們。重要的是，FIB 資料表可能存在於相關的轉送引擎當中，方式有純軟體的轉送、搭配硬體加速的軟體轉送（例如，Intel 或 ARM 的 GPU/CPU 產品）、採用晶片的轉送（NPU，例如，Broadcom、Intel、Marvell 在乙太網路交換機市場推出的產品）、採用 FPGA 的轉送以及採用專用積體電路的轉送（ASIC，例如，Juniper Trio），或其他組合方案[4]，最後取決於網路環境中相關元件的設計。

這裡所說明的「**軟體路徑**（*Software Path*）」，是以目前專用的網路元件（例如，路由器或交換機）所採用的 CPU 轉送方式為例，它以處理器進行密集式的查詢動作（在核心或使用者層級執行，是由廠商的設計所決定並受限於硬體作業系統的架構和特徵），以換取處理器記憶體看似無限的轉送資料表項目儲存。在目前新一代的運算環境中，所採用的 VM 虛擬主機管理程式（Hypervior-Based），其軟體式交換機或橋接器有許多硬體轉送模型上的優化（但也伴隨一些相關限制）。

從過去演變的歷史來看，硬體資料表查詢技術已經可以達到相當高的封包轉送效能，因此在網路元件設計中佔有相當重要的地位，尤其是在需要高頻寬的網路環境當中。然而，受到雲端運算成長和創新應用模式的推動，最近在處理器輸入輸出方面的增強，使得採用專用積體電路轉送晶片設計的中低階設備，互相競爭的情況變得相當激烈。

3. 除了檢查大小、對齊、封裝是否符合規則要求及校驗與驗證之外，還會進行其他的檢查作業。特別是當封包的「類型（Type）」被正確識別之後，可能進行另外的「虛假性（Bogon）」檢查機制，以了解對於該類型是否有不恰當之處。

4. 在硬體平臺設計中，會有一個資料表在硬體中專用於「快速路徑」的查詢，當查詢失敗或需要更多資訊時，這個查詢隨後在一個軟體維護表上進行重試——此時便是在「緩慢（Slow）」的路徑中查詢，硬體平台會有「溢位（Overflow）」的問題相當常見（通常因為資料表的數量，或是欄位數量受到硬體資源的限制）。

硬體轉送設計方面的差異牽涉多種因素，包括（卡板和機架）空間、預算、效能以及吞吐量[5]的目標需求。這些會導致維持特定的目標封包長度（或混合的封包長度），以及轉送（介面中接近信號或理論上最大的吞吐量）所需要的記憶體類型（速度、頻寬、大小和位置）的差異，還有執行預算（封包數量、序列或類型）的差異。最後，仍有轉送特性支援和轉送規模（例如，轉送資料表項目的數量、路由資料表的數量）等設計上的差異。

資料平面從尋找結果而進行的典型操作，就是**轉送**（在特殊情況下，如多點傳播中典型的操作是複製）、**丟棄**、**重新標記**、**計數及佇列**，某些動作可以組合或串連起來。但是，在某些情況下轉送操作會返回設備內的本地端連接埠當中，這資料表流量被重新導向到該設備內部運作的行程，例如，OSPF 或 BGP[6]。這些封包會採用上傳路徑以離開硬體轉送路徑，並使用內部通道轉送到路由處理器，此路徑通常是一條相對較低吞吐量的路徑，因為它並非設計用來做為高吞吐量轉送封包的用途。但是，有的設計可以簡單的在內部交換結構裡增加一條額外通道，以達成在設備內部接近線速轉送的目的。

除了封包轉送的操作之外，資料平面還包括一些服務及特色，通常稱為轉送特性（例如，存取控制清單和服務品質 / 原則）。在一些系統當中，這些特性使它們自己擁有獨立的資料表項目，有的系統則是當作轉送資料表的擴充機制（透過增加資料表項目的寬度）來執行。此外，不同的設計可以達成不同的特性和轉送操作順序（如圖 2-4 所示），有一些排序作業可能會讓某些操作互相排斥。

使用這些特色功能，使用者可以在一定程度上調整或進行轉送資料表的查詢，舉例如下：

- **存取控制清單**（*ACL*）可以針對某個特定的流程，進行比對之後執行一個丟棄的操作（請注意，在 ACL 中，一個較為廣泛的參數集合可以被用到轉送作業中）。由於 ACL 也可以有合法的轉送資料表項目，所以封包不會被丟棄。
- **服務品質**（*QoS*）原則最後把一個流程對應到出口的一個佇列，或重新標記它的 TOS/COS 欄位內容，以針對網路原則的服務進行規格化。就像 ACL，不管是否已經存在到這個目的地的轉送資料表項目，它都可以將封包標記為丟棄（或進行管控）。

5. 在 ASIC 設計中有許多特別的連鎖因素，除了消耗功率、散熱和尺寸的考慮之外，最後還需要從工藝及模具尺寸，到邏輯佈線、時鐘及振盪頻率（這可能帶來最多耗損），以及在資料表共用等方面的效益 / 成本中找到最後的平衡點。

6. 有許多這種範例，包括前面所述的 OAM、BFD、RSTP 和 LACP。

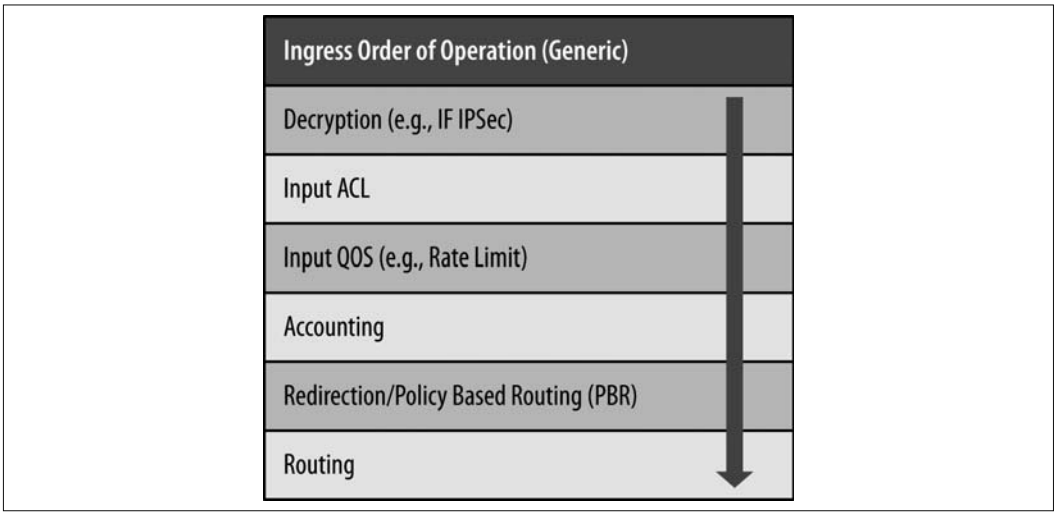


圖 2-4：傳統路由器 / 交換機中，封包進入路徑的特色功能應用典型範例

這些轉送特性，將會在第 7 章內容中詳細說明。簡單來說，這些特性是作為服務的資料平面和控制平面所存在，當本書討論工作階段管理、代理程式以及封包標頭的大規模轉換…等服務時，這些定義就有明顯的差異，作為轉送操作的組成部分，資料平面需要針對某種程度的封包進行標頭內容的重寫。

## 在控制 / 資料平面之間傳遞資訊

目前大型多插槽 / 多卡板（即機架式）的分散式轉送系統內部功能，與 SDN 邏輯上集中但實體上分散式的管控機制相仿，特別是路由資料表的發佈以及在硬體方面會令人感興趣，研究典型的分散式交換內部工作情況，就可以了解與控制平面相仿的一些功能和行為。例如，在系統當中，控制平面位於獨立的處理器 / 卡板之上，而資料平面在其他獨立的卡板當中，為了系統的故障復原與容錯機制，圍繞在這些元件之間的通訊需要有某些操作。如果控制平面從主機中移走或放置於更遠的位置（例如，邏輯上或最嚴格的集中式控制平面），那麼所有的通訊機制是否都還需要，將是很值得探討的問題。

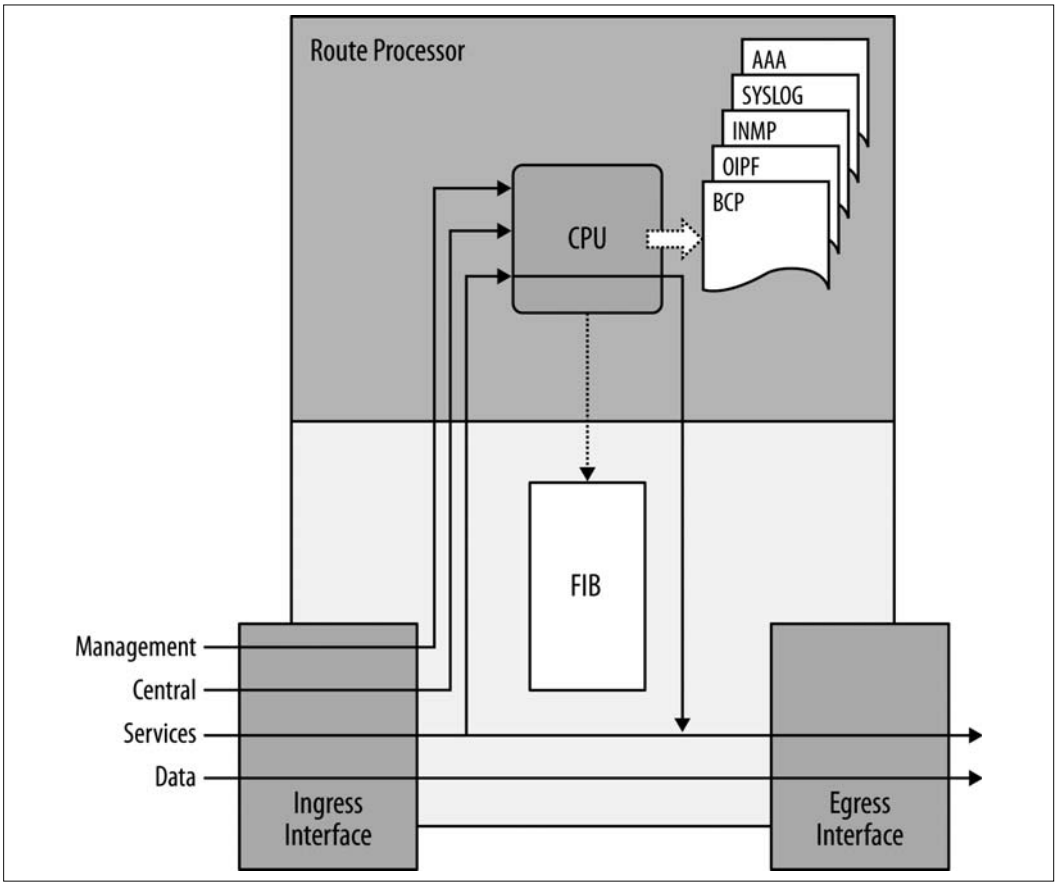


圖 2-6：控制平面及資料平面的整合範例

除此之外，協定依照此結構來設計就可以優化和提高效能。舉例來說，多協定標籤交換（MPLS）採用 IP 協定來承載並控制這些流量，這樣便可以理想化在一般通用 CPU 上運作專用路由處理引擎。而 MPLS 固定採用標籤的交換功能，最適合用於每片卡板上需要簡捷且高效能的封包處理器來達成。

在 SDN 技術被大家討論之前，控制 / 資料平面分離的範圍大約一公尺內的距離（在同一台設備機殼內，或在彼此相連的多台設備內）。在前面各節中描述的控制平面和資料平面，雖然是分散式卻又緊密整合在一起（相對緊密的放置在一起），並採用硬體和軟體組合的方式進行搭建和管理。除了這些元件和非常多的系統外部觀察者所隱藏的內部結構外，這些網路元件經常在相同的硬體產品線之上進行開發，以採用服務、管理、控制和資料平面之間的平衡設計為著眼點，並在吞吐量（以及複雜性）上有所變化。這種多

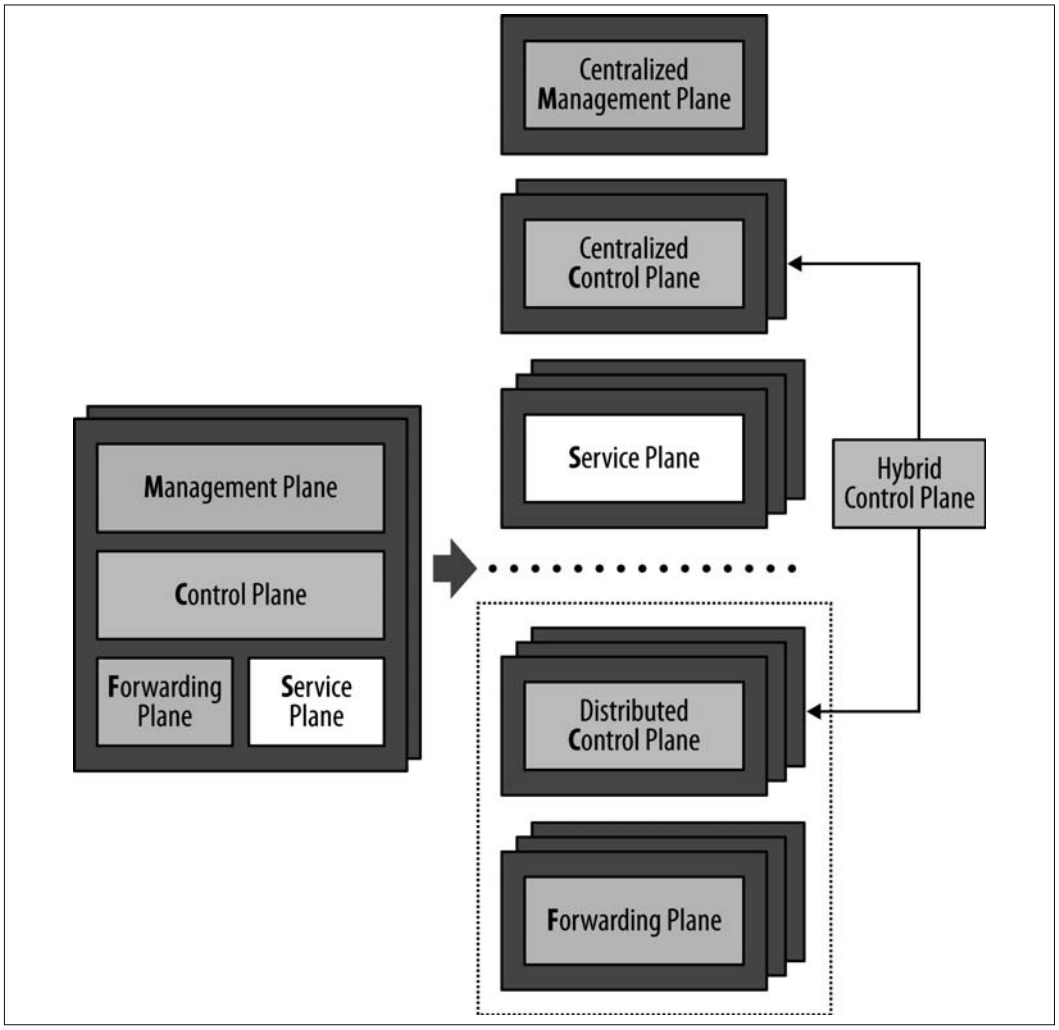


圖 2-7：將管理、控制、服務和轉送平面進行分離，並使它們都能獨立擴充

## 成本開銷

雖然這是一個吸引你目光的字眼，但是和其他推動控制平面與資料平面分離的因素相比，關於成本開銷的內容相對較少。成本開銷，是由資產部分（投資成本 CAPEX）和營運部分（營運成本 OPEX）所組成。成本開銷由這些因素來決定：規模（投資成本的推動力）、複雜性和穩定性（營運成本的推動力）。首先，從投資成本的角度出發，對於眾多客戶來說（特別是服務供應商或營運資料中心的大型企業），和他們的網路元件處

這個資訊透過「遞迴（*Recursion*）」機制，使得在控制 / 資料平面裡更容易存取和擴充。遞迴允許網路控制平面使用不同協定的不同屬性來發佈資訊，透過一系列共用欄位將這些協定的資訊連結起來。在圖 2-11 當中，這項優化的範例是大量 BGP NLRI 資訊（網路層可到達性資訊），可以用 IGP 資料集合裡的單個 IP 目的地位址來表示（如果服務供應商使用控制平面協定特性，允許多條最佳路徑到達目的地，或有多條等價路徑到達宣告的實體，則用一個位址集區來進行表示）。當控制平面為資料平面建立資料表項目（FIB）時，BGP 資訊就被關聯到無迴路的 IGP 資訊中。

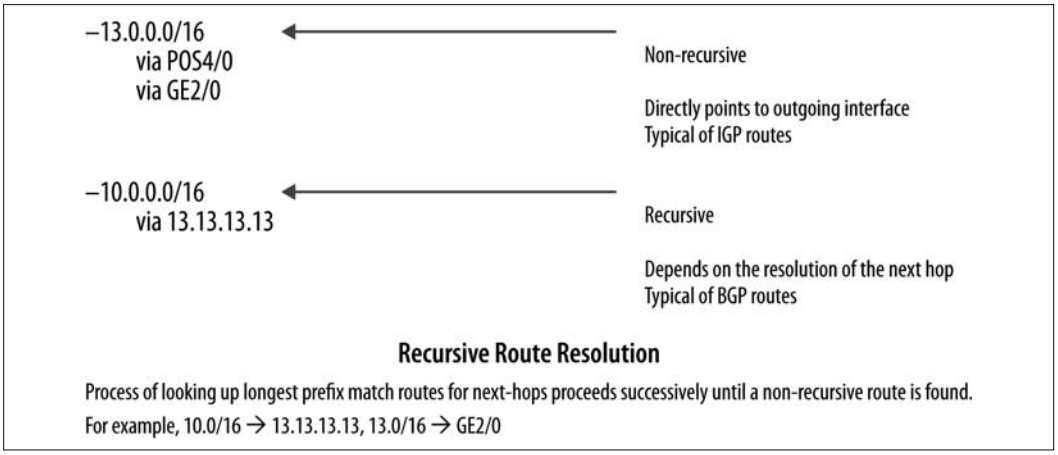


圖 2-11：路由遞迴示意圖

在資料平面中，遞迴機制最後透過 IGP 的「下一個跳躍點（*Next Hop*）」，將 BGP 前置碼解析為對 L2 轉送資料表項目中鄰接關係的一個指標，最後用於目的地 MAC 資料表項目裡，表示下一個跳躍點的路由器 / 交換機介面。

在外部邊界內，透過多種原則工具的使用，服務供應商具有額外控制狀態的彙總和宣告的管理權。在 IP 模型中，允許為資料平面轉送所選的控制平面資料集上有一些額外、局部化的控制，標準化或服務供應商獨有的操作 [14]，是允許原本原則來控制學習到的網路可到達性資訊優先順序。

例如，這些工具能夠透過一條靜態路由資料表項目，重新發佈到一個動態路由通訊協定裡，或在重新宣告之前修改一條前置碼的屬性內容，以間接為某條特定路由狀態影響鄰居的優先順序 [15]。如果在 RIB 中不存在該筆路由，或靜態路由具有較高的管理優先順

14. 在 RFC 1104 文件當中，提供原則式路由模型資訊。

15. 可以說，這種處理對於 BGP 環境特別有用。

提供特定類型的邏輯堆疊網路，由多台伺服器（或類似控制器這樣的功能）來支撐。這些伺服器的功能中雖然沒有提供溝通介面，但提供最基本的高可用特性。

## 路由伺服器

「**路由伺服器**（*Route Server*）」已經演變為，ISP 用來處理 BGP「**對等**（*Peer*）」的運作模式，以及處理外部 BGP 對等原則的一種手段。另一種稍微不同的機制是「**路由反射**（*Route Reflector*）」，目前已經用於處理內部對等目標而標準化了。

在圖 2-16 當中，路由伺服器是一個採用外部邊界閘道協定（eBGP）的控制點，通常位於所有獨立「**自治系統**（*Autonomous System*）」各方所共用的網段，用來與每一個參與方接收控制狀態更新（網路層可到達性資訊 NLRI），將各種過濾和各種原則套用到這些更新上，然後根據最後結果，運算出最佳路徑（這可能與在中間原則步驟算出的 BGP 最佳路徑不同），並為每個參與方建立一個 RIB（把運算結果返回給參與者[28]）。路由伺服器對於 AS Path 這樣的 BGP 屬性是透明的，每個參與方只需要一個 BGP 工作階段進行資訊交換即可。

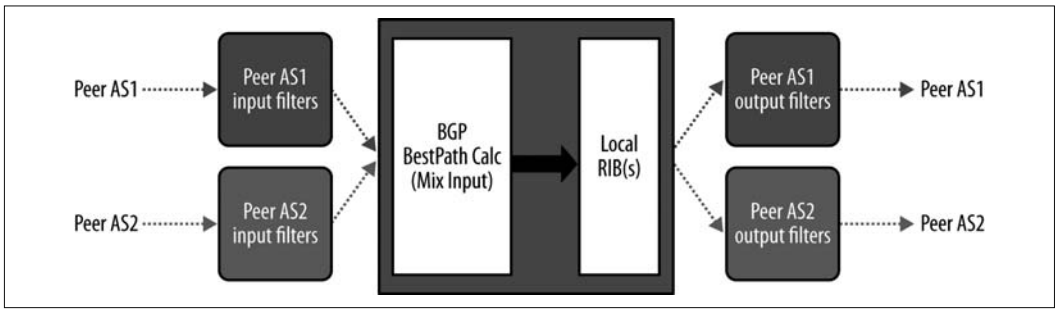


**圖 2-16：路由伺服器運作架構示意圖**

路由伺服器可以連接到路由註冊資訊[29]。註冊是路由物件（ASN 自治系統號碼、原則、前置碼、認證資訊）的分散式儲存，它提供一個工具集，可以使獨立的對等節點（邊界路由器）和路由伺服器達成自動佈建（如圖 2-17 所示）。

28. 最著名的實作，包括 Quagga（GNU Zebra 的一個分支）和 BIRD。

29. RIPE IRR 就是這種路由註冊的範例。