第 3 章

OpenFlow

簡介

在第 2 章當中，已經針對網路控制平面和資料平面進行說明。本章，將會專注於不斷發展中的 OpenFlow 提案及協定。OpenFlow 協定，被許多人視為重新開始的全新設計理論和 SDN 技術的發起者。本章還將討論，就一般情況而言，SDN 控制器如何達成網路環境中控制平面的角色，並且有可能重新塑造一個服務供應商網路環境的新面貌。

OpenFlow 協定，最初為史丹佛大學網路研究的一部分，起初的重點是在校園網路中建立用於研究和實驗性的測試用協定。在此之前，通常必須要重頭開始建立自己的實驗平臺，由這個核心想法開始演變出的一種觀點，認為 OpenFlow 可以取代商用交換機和路由器裡 L2 和 L3 的網路功能。這種機制稱之為「**重新開始的設計**（*Clean Slate Proposition*）」。

在 2011 年，由一些服務提供者成立「**開放網路基金會**（*Open Networking Foundation*，*ONF*）[1]」的非營利性組織，以商業化、標準化和促進營運網路中使用 OpenFlow 協定。ONF 是一種新型態的標準開發組織，它成立並推動 OpenFlow 協定發展和其他 SDN 技術相關的市場。此外，該組織每年都會舉辦開放網路高峰會會議。

1. 目前 ONF 組織已經有 90 多個成員，包括學術單位、政府機構、企業、服務供應商、軟體公司和設備廠商。

從宏觀的角度來看，大家的注意力能夠集中在軟體定義網路技術要歸功於 ONF 組織。OpenFlow 模型的關鍵元件（如圖 3-1 所示），已經成為 SDN 技術常見定義的一部分，如下所示：

- 控制平面和資料平面的分離（對於 ONF 來說，控制平面是邏輯上集中式的控制器系統）。
- 在控制器和網路元件代理程式之間使用的一個標準協定，用於即時狀態（在 OpenFlow 環境中，指的是轉送狀態）。
- 透過現代化可擴充的 API 機制，以集中化的檢視圖來提供網路可程式設計性。

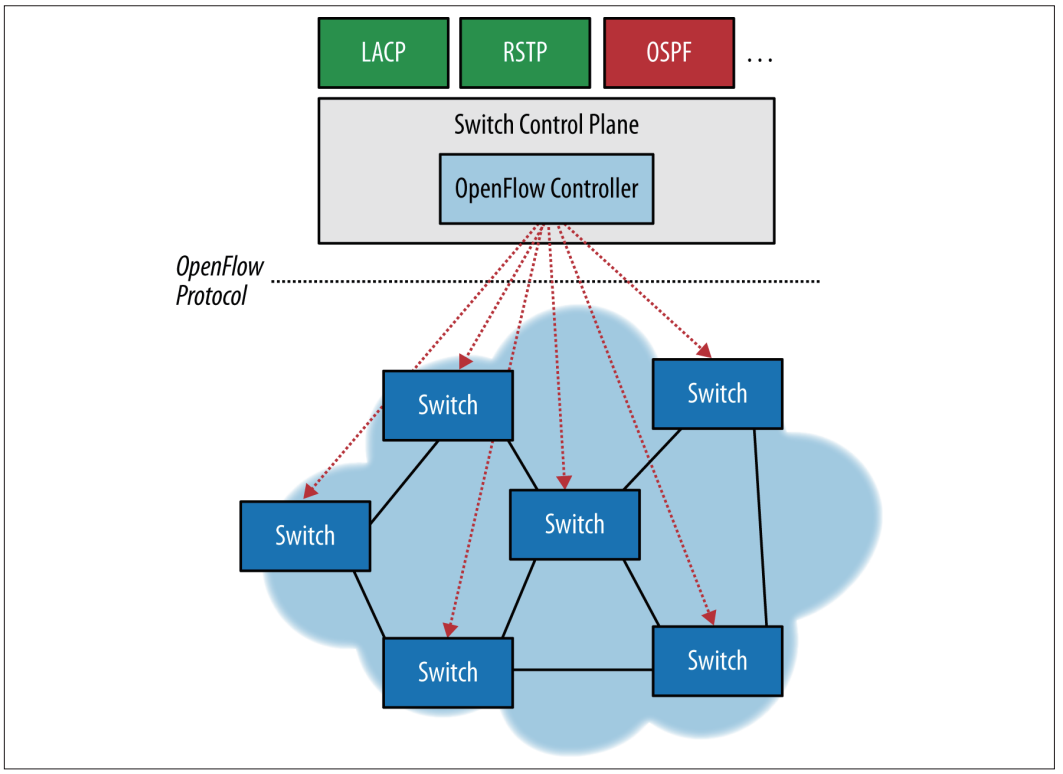

圖 3-1：OpenFlow 運作架構（有些控制平面的應用程式會在控制器之上，以模擬傳統的控制平面應用行為）

OpenFlow 是一個協定和 API 溝通介面，它本身並不是一個產品或功能。換句話說，如果沒有應用程式（可能不止一個），提供指令說明哪些資料流程流向到哪個網路元件（出於應用程式自己的目的）的話，控制器什麼事情都不能做。

OpenFlow 協定，目前被分為兩個部分：

- **連接協定**（Wire Protocol，目前版本為 1.3.x）：用於建立控制工作階段，它定義流程的修改方法，以進行互動和針對統計資料進行收集的訊息結構，同時定義交換機（連接埠和流程資料表）的基本結構。在 1.1 版本時，新增支援多個資料表項目（Multiple Table）、針對動作執行進行暫存以及中繼資料傳遞功能，這些功能最後在交換機內建立邏輯的管理機制用來處理流程。
- **配置與管理協定** *of-config*（目前版本為 1.1）：它採用 NETCONF（使用 Yang 資料模型）為特定控制器分配實體交換機連接埠，並定義高可用性（主要 / 備用機制）和控制器連接失敗時的行為。雖然 OpenFlow 協定，可以配置 OpenFlow 的命令 / 控制等基本動作，但它還不能啟動或維護網路元件（尚未達到 FCAPS 層級的管理機制，其中 FCAPS 為故障、配置、計費、效能、安全等五個項目的縮寫）。

由於 OpenFlow 連接協定 1.0 版之後的複雜性。在 2012 年時，ONF 將「**測試活動**（*Plugfest*）」這種互通性和一致性測試，變成更正式的測試（外包給印第安那大學）。

雖然 ONF 組織已經討論多時並建立 OpenFlow 協定，但截至本書寫作時仍尚未真正發生（雖然已經有很多開放原始碼的控制器出現）。

OpenFlow 協定，並未直接提供網路切割的能力（這是一個有吸引力的功能，將網路元件切割成單獨控制的連接埠群組，或將網路功能切割成單獨的管理區域）。然而，FlowVisor[2]（擔任多個控制器和網路元件之間的透明代理程式）等工具，以及特定廠商（能為不同控制器工作階段，建立多個虛擬交換機的代理程式），已經能夠提供這樣的功能。

2. FlowVisor 導入一些中間延遲機制，因為它必須處理交換機和控制器之間的封包。

OpenFlow 連接協定

那麼，OpenFlow 協定有哪些新內容？

首先，它為各廠商協定配置的死板設計，以及非標準化的語法導入暫時置換狀態的運作概念[3]（流程資料表項目，並不儲存在網路元件當中的永久記憶體內）。暫停狀態可以有效避開過去網路自動化當中，配置提交模型執行速度過於緩慢的問題。

對於大多數網管人員來說，這種配置的最後結果是建立轉送狀態（儘管是分散式的狀態，並且是在分散式控制環境中學習到的）。事實上，對於許多人來說，配置正確性的測試就是驗證轉送狀態（路由資料表、轉送資料表或橋接資料表）。當然，與傳統的網路元件上分散式管理機制相比，如果總是想事先在轉送資料表裡放入某些轉送規則的話[4]，這種方式至少是將維護狀態的管理負擔轉移到控制器上。

其次，在 OpenFlow 流程項目裡，整個資料封包的標頭（至少 L2 和 L3 的欄位）可用於比對和修改操作（如圖 3-2 所示）。很多欄位的比對可以進行遮罩[5]，已經發展出不同的 OpenFlow 版本[6]，在圖 3-2 當中已經說明「L2 + L3 + ACL」的轉送功能（下一個跳躍點抽象的快速收斂）可以是比較複雜的。各資料表中所支援的組合，使得轉送功能可以支援非常廣泛的突發事件組合。

3. 在程式設計控制流程中，建立瞬間狀態的能力可能是 OpenFlow 的暫時優勢，因為已經有一些現代化的程式設計方法（例如，NETCONF），已經增加這些功能提案了。

4. 這種配置方式並非是獨特的，在過去移動網路中 PCRF/PCEF/PCC 系統（與 Diameter 互相溝通），就已經為每個成員這麼做了。標準組織一直在進行系統各元件之間的交換資訊，以及和廠商互相操作的明確定義和標準化處理工作。毫無疑問，移動原則系統可能演變成 SDN 系統並具備 SDN 特點。這種 SDN 系統與 OpenFlow 之間的主要區別，可能是靈活性（客觀來說，稱為簡潔性更合適）。

5. 所支援的匹配類型（連續型或偏移型），是另一種依賴於平臺的能力。

6. 可惜的是，在 ofp_match 中加入一個 TLV 結構（匹配欄位被進行重組），因此與舊有的 1.2 版本和以前的版本無法相容。事實上，由於修改的數量和種類 1.2 版本被認為無法實現（儘管 2012 年終於發佈開放原始碼代理程式）。以 HELLO 握手機制，針對某些修改以達成版本的探索功能，但是與交換機版本不相容，因此無法與控制器互相連接。

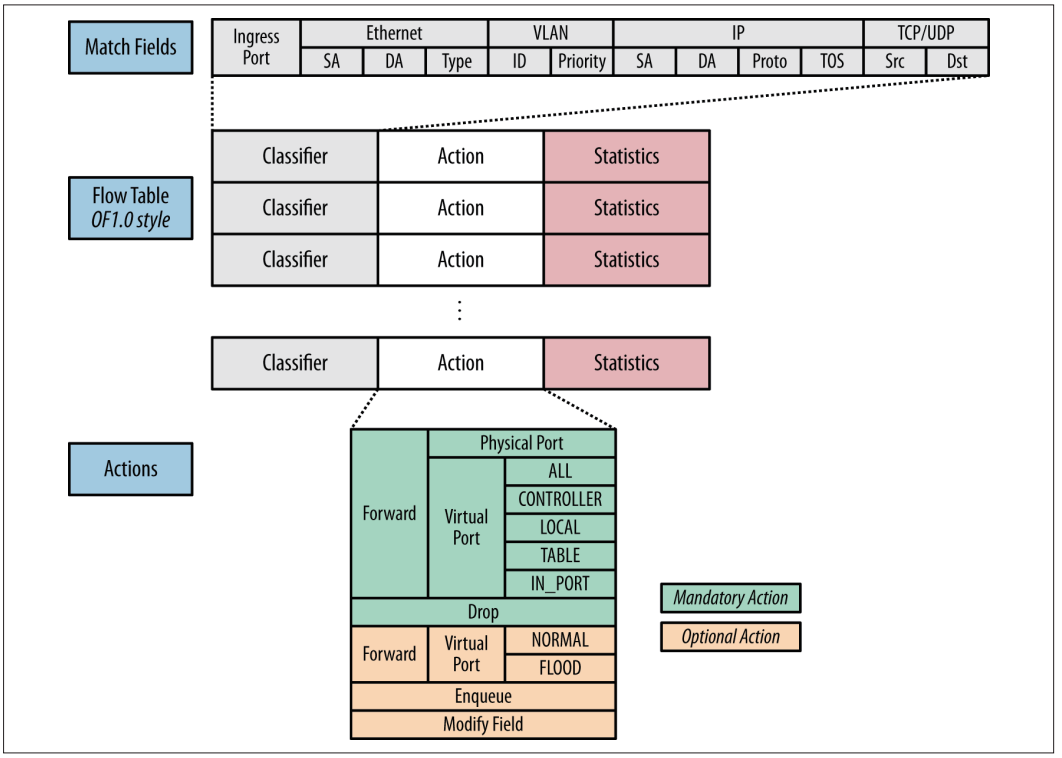

圖 3-2：OpenFlow（連接協定）1.0 版本原始協定內容

OpenFlow 與分散式 IP/MPLS 模型相比，與我們所能控制的部份有很明顯的差異（OpenFlow 有 11-tuple 的比對空間）。下面列出相關可能性：

- 在 OpenFlow 中比對指令的遮罩能力，網路環境能夠模擬採用 IP 目的地位址的轉送行為。
- 在 L2 和 L3 網路上，都可以透過資料表現出來源端 / 目的地的路由行為。
- OpenFlow 的封包比對能力目前還沒有其他標準能做到，使得在替代原則式路由或分散式控制環境中的比對 / 轉送機制方面非常強大。

最後就是 OpenFlow 修改動作。起初的概念是交換機（透過在交換機之外運作的應用程式），但也可以當成像服務裝置一樣，執行「網路位址轉譯（*Network Address Translation*，*NAT*）」或防火牆等服務。採用硬體的轉送系統能否做到這一點，高度依賴

於廠商的機制（支援的指令、順序和保持作業的數量）[7]。隨著標籤管理機制被加入到連接協定 1.3 版本後，被 OpenFlow 協定所控制的網路元件，能夠很容易的模擬 MPLS LSR 等平臺（或其他傳統的分散式平臺功能）。

OpenFlow 協定，透過 EXPERIMENTER 延伸模組（可以是公共的或私有的），來達成訊息控制、流程比對欄位、計量資料表操作、統計資料延伸模組，以及廠商特定的延伸模組（可以是公共的或私有的）。

資料表項目可以有優先順序（如果資料表項目有重疊），並且有時間到期機制（在某些情況下，有效避免清理操作並對控制器遺失等情況下，對於流程項目有終止效果）。

OpenFlow 支援 PHYSICAL、LOGICAL 和 RESERVED 的連接埠類型（分別對應實體、邏輯和預先保留的連接埠類型）。這些連接埠被當成入口、出口或雙向結構。

在 RESERVED 連接埠中，IN_PORT 和 ANY 的功能已經不言而喻。

TABLE 被要求建立多資料表管線（OpenFlow 最多支援 255 個無類型的資料表，透過任意的 GoTo 機制在管線中進行排序）。

其餘的 RESERVED 連接埠，則用來達成重要（有意思）的行為[8]。

LOCAL

僅用於出口連接埠，該邏輯連接埠允許 OpenFlow 應用程式，存取網路元件的主機作業系統連接埠（進而能存取執行程序）。

NORMAL

僅用於出口連接埠，該邏輯連接埠允許交換機，能夠像傳統乙太網路交換機（洪水流量 / 學習行為）那樣運作。根據該協定功能規範中，該連接埠僅在「混合（*Hybrid*）」[9] 交換機中受到支援。

7. 後面的應用案例探討中，在控制器上建立這樣的應用程式，或是在虛擬服務路徑中針對此功能進行虛擬化。

8. CONTROLLER 是下面幾個預留連接埠中，唯一必需的（其他為可選擇的）。其他必須預留連接埠為 ANY、IN_PORT、ALL 和 TABLE。這裡列出的組合，是它們在混合模式當中的相互作用。

9. 這裡「混合（Hybrid）」的原始定義是，一台交換機可以當成 OpenFlow 交換機，也可以（在 OpenFlow 區域裡面的連接埠）當成單純的 Layer2 交換機。

FLOOD

僅用於出口連接埠，該邏輯連接埠使用網路元件的複製引擎，將封包發送到所有標準（非預先保留）連接埠。FLOOD 與 ALL（另外一個預先保留連接埠）不同，ALL 連接埠包含入口連接埠。FLOOD 利用網路元件的封包複製引擎。

CONTROLLER

允許轉送封包的流程規則（透過控制通道），從資料路徑轉送封包到控制器（或相反方向的轉送）。因此，能夠順利啟動 PACKET_IN 和 PACKET_OUT 的行為。

OpenFlow 的轉送模型提供兩種模式：主動模式（預先提供）和被動回應模式（資料平面驅動）。在主動模式下，控制程式會先把需求放置到轉送資料表項目當中。如果收到的資料流量，不能與現有的流程項目互相進行比對作業，則我們有兩種（全域）選擇：放棄這個流程，或使用 PACKET_IN 選項來進行決定，以建立這些封包的流程項目（無論是正面、轉送或負面，還是其他處置），也就是被動回應模式。

控制通道，最初為一個對稱式的 TCP 工作階段（可能用 TLS 來保障其安全性）。該通道用於配置和管理（存放流程資料表、收集事件以及統計），並提供交換機與控制器 / 應用程式之間，能夠發送和介面封包的路徑。

統計資料支援流程（Flow）、彙總（Aggregate）、資料表（Table）、連接埠（Port）、佇列（Queue）和廠商特定的計數器（Counter）。

在 OpenFlow 協定 1.3 版本當中，允許採用多種輔助連接（TCP、UDP、TLS 或 DTLS）機制，以便處理任何 OpenFlow 訊息類型或子類型。但是，UDP 和 DTLS 通道無法保障封包的順序，因此規範裡有行為準則，以便確保對封包的特定操作是對稱的（避免在控制器上的順序問題）[10] 。

OpenFlow 支援 BARRIER 訊息，以便建立一個調度機制（建立單元或流程控制），用於與後續訊息有依賴關係的場景（提供的範例是，PACKET_OUT 操作首先需要一個流程項目，以比對封包和進行轉送）。

10. David Ward 在 JIRA 系統中（ONF 討論或制定標準中，用於提交提案的系統），提出關於這些協定變化相當完整的解釋，特別是針對從連線階段的通道變為 UDP 的問題。

複寫

OpenFlow 協定，提供幾種封包複寫機制。

ANY 和 FLOOD 預先保留的虛擬連接埠，主要用於模擬 / 支援現有協定的行為（例如，LLDP 用來為控制器收集拓撲，並採用 FLOOD 作為出口連接埠）。

群組資料表允許把連接埠組合成一個出口連接埠集合，以支援多點傳播、多重路徑、間接轉送和快速故障切換。每個群組資料表，實際上就是一個動作列資料表（資料表動作之一，便是輸出到一個出口連接埠）。雖然，一共有四個群組資料表類型，但只有下面兩個是必需的：

All

用於多點傳播，列表中的所有動作都必須被執行[11]。

Indirect

用來模擬 IP 轉送中下一個跳躍點的行為，以進行更快速的路由收斂行為。

套用執行動作之後，便能建立一個輸出 / 連接埠動作的資料表，以便進行連續的複製動作（套用動作在 OpenFlow1.0 版本中，為單向操作）。

轉送抽象工作群組（FAWG）

OpenFlow 交換機模型（如圖 3-3 所示），採用軟體交換機（在規模和封包操控特性上，有突出的靈活性），或一些假設的硬體轉送實體（例如，像 TCAM 那樣有容量、寬度、深度的多重記憶體）上得到很好的支援。但由於不是所有的設備都符合上述條件，所以在 OpenFlow 的原始架構、多資料表及其他 OpenFlow 強大功能所支援的封包管控機制，設備的支援度方面有非常多的變化。

11. 在規範當中，ALL 群組類型可以用於多重路徑，但這並不是 IP 轉送的多重路徑，資料封包會被複製到兩條路徑。這種行為更加符合 Live/Live Video 類型的要求，或其他多重路徑的要求。

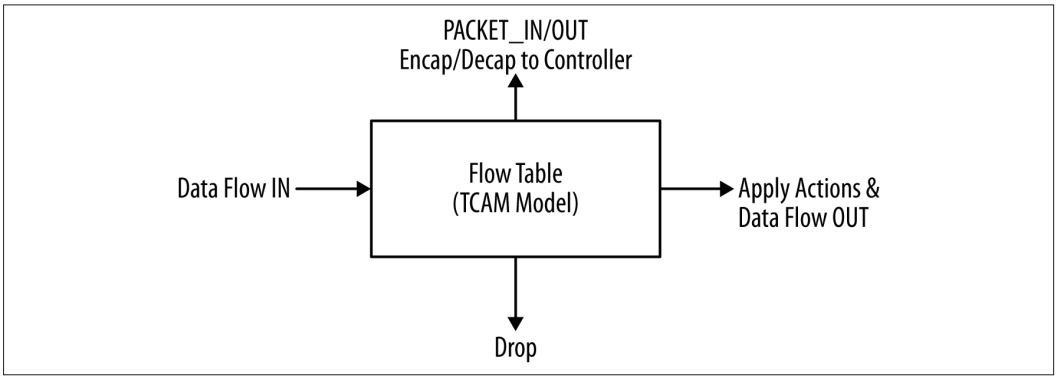

圖 3-3：OpenFlow 1.0 版本中，轉送模型運作示意圖（一個非常簡單的共用資料表模型）

一般來說，OpenFlow 1.1 版本以及後續相關版本，具有潛在組合的複雜性（如圖 3-4 所示），在採用 ASIC 晶片的轉送硬體設備中無法有效運作。因此，OpenFlow 應該選擇何種抽象層級已經成為一個問題，因為這關系到對於應用程式的適用性。

雖然這是一個普遍的觀點，但是在 Martin Casado（OpenFlow 之父）的訪談當中，關於 OpenFlow 抽象層級的爭論中更普遍被引用[12]。

在訪談內容中，Martin 舉例說明 OpenFlow 在流量工程中的應用，說明採用現有的 ASIC 晶片來解決 OpenFlow 的局限性（一般觀點），然後就 OpenFlow 對於網路虛擬化的適用性，給了一個特別的評論：「我認為 *OpenFlow* 協定，對於網路虛擬化來說太過於低階」[13]。

12. 詳細資訊請參考 http://searchnetworking.techtarget.com/news/2240174517/Why-Nicira-abandoned-OpenFlow-hardwarecontrol。

13. OpenFlow 協定其複雜的運算式，由 David Meyer 提供。

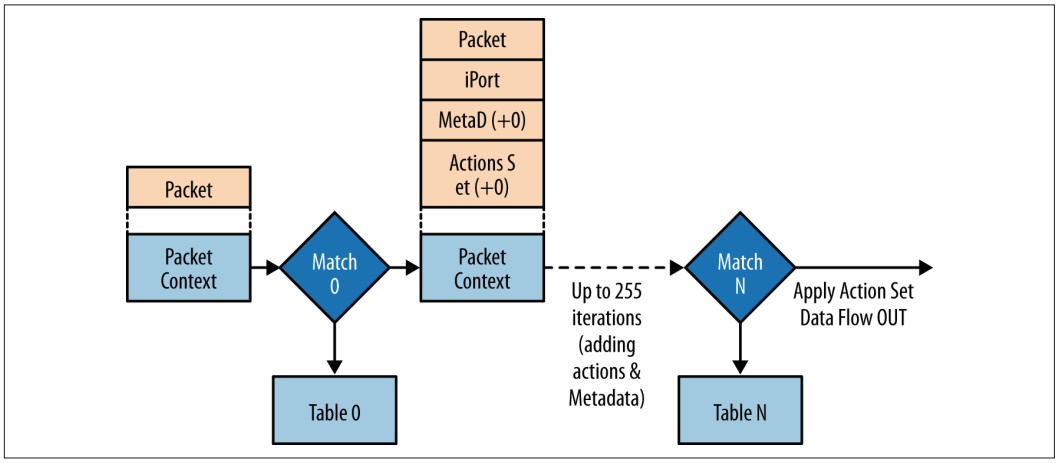

圖 3-4：OpenFlow 1.1 版本以及後續版本，當中的管線模型（非常複雜，其組合複雜度為 O(n!×a(2^l)) 路徑，其中 n= 資料表的數量，a= 動作的數量，l= 比對欄位的寬度）

該協定早期版本中僅具備有限的檢測功能，在 1.3 版本中重構為支援一些原始資料表能力的描述功能（為每項比對欄位增加比對類型，例如，精確比對、萬用字元比對和最長前置碼比對）。

以下是從參考文獻當中，所引述現有抽象層級的不足之處[14]：

- 資訊遺失。
- 資訊洩漏。
- 控制平面到資料平面抽象能力薄弱。
- 組合狀態爆增。
- 資料平面驅動的控制事件。
- 間接基礎架構匱乏。
- 高敏感度的時間週期訊息。
- 多個控制引擎。
- 可擴展能力薄弱。
- 缺少初始架構。

14. 轉送平面模型（FPMOD），以及資料表歸類（Table Type）該何去何從？此文章出自於 David Meyer/Curt Beckmann，文件編號 ONF TAG-CoC 07/17/12。

「轉送抽象工作群組（Forwarding Abstraction WorkGroup，FAWG）」，正在透過「資料表歸類模式（Table Type Pattern，TTP）[15]」，嘗試以第一代協商式交換機模型 FAWG，開發出一個建立、標識（唯一的標識）和共用 TTP 的過程。此外，還開發出協商演算法（採用 Yang 模型建構）和訊息，以建立控制器與交換機之間協定的 TTP（可能會增加到 of-config 版本 1.4 當中）。

TTP 模型是預先定義的交換機行為模型（例如，HVPLS_over_TE 轉送和 L2+L3+ACL），由特定資料表描述（比對 / 遮罩和操作）和資料表互相連結（資料表的邏輯管道）來表示。這些描述可能隨著元素，在服務流程中的角色不同而有所不同（例如，對於 HVPLS 轉送行為，該元素是開頭、中點還是出口）。

早期的模型貢獻建議，已經在 OpenFlow 的 1.3.x 版本中，透過進一步擴充以達成 TTP 機制。

如果 FAWG 成功了，那麼控制器上的應用程式至少可以從行為描述角度，知道網路元件所具備的能力。

以下是一個需要 TTP（或 FPMOD）[16] 的簡單範例。

如果硬體資料表之間，包含相似的資料及資料表索引不同性較低時，就可以共用資料表項目（例如，包含 MAC 位址轉送和 MAC 位址學習，這兩個檢視圖的邏輯資料表）。這個資料表可以用很多不同的方式來達成，例如，作為一個單一的硬體資料表。達成 MAC 學習 / 橋接的 OpenFlow 控制器，必須準備兩個不同的資料表：一個用於 MAC 位址學習，另一個用於 MAC 橋接（這是 OpenFlow 資料表的限制）。目前，還沒有辦法把這兩個不同的檢視圖綁在一起。在圖 3-5 的範例當中，可能會出現計時的情境，將兩個獨立的 OpenFlow 資料表，實體流程資料表修改後進行同步（也就是說，不能在 MAC 位址學習之前進行轉送）。

15. 一個更複雜的解決方案「轉發平面模型（Forwarding Plane Models，FPMOD）」，已經被 OpenFlow 討論群組提出，但是因 FAWG 正在開發更簡單的資料表類型描述模型（由技術顧問群組 TAG 所建議）而擱置了。這個解決方案模型較少，主要是原始架構的集合在交換機硬體抽象層編碼時進行對應，而非採用控制器的原始架構（當然，是採用行為的協商模型，但不一定是靜態、預先定義、限於管線描述的模型）。

16. 這個範例，同時也來自於本章開頭所提到的 Meyer/Beckman 參考文獻。

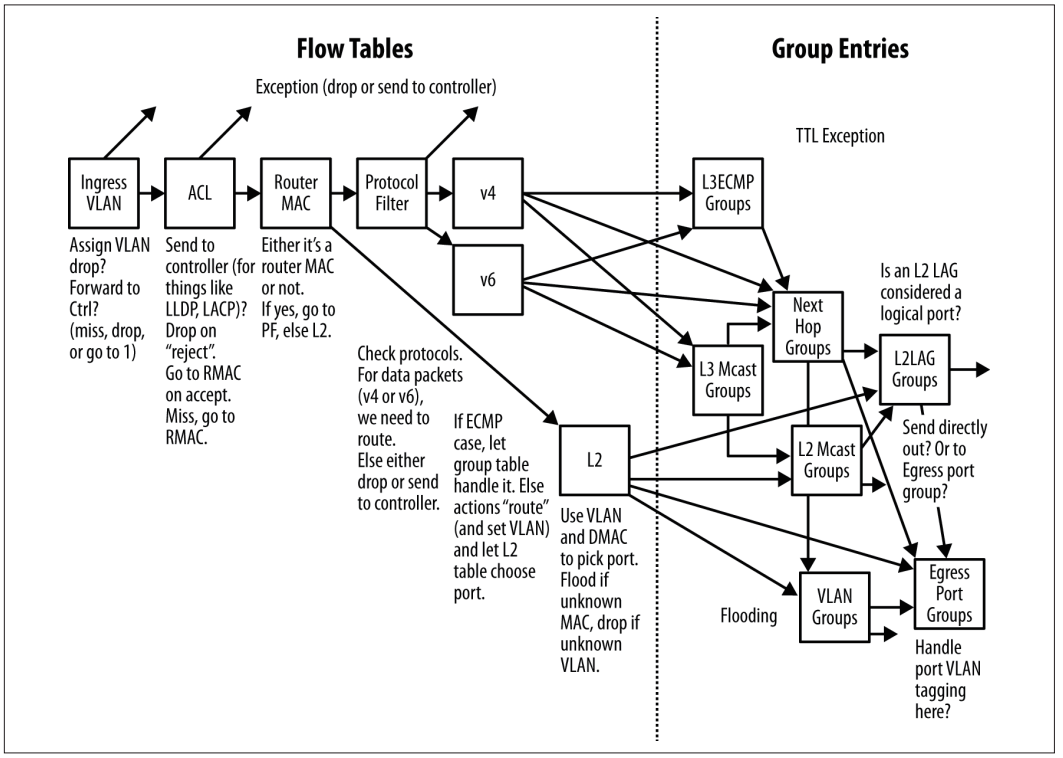

圖 3-5：TTP 模型複雜度的範例：L2+L3+ACL/PBR（原則式路由）的 TTP（資料來源：Brocade 公司的 D. Meyer 和 C. Beckmann）

如圖 3-5 所示的範例內容中，IPv4 和 IPv6 資料表指向群組資料表，以便模擬傳統 FIB 對下一個跳躍點的抽象層級（為了更快速的進行收斂動作）。

配置和擴充性

of-config 協定，最初設計來配置網路元件上 OpenFlow 的相關資訊（of-config 1.0 版本）。該協定的運作架構，圍繞在 XML 模式、Yang 資料模型和 NETCONF 協定。

對擴展 of-config 的建議，來自於 Config-Mgmt 工作群組（配置和管理工作群組）或其他工作群組（例如，轉送抽象工作群組或傳輸工作群組[17]）。

17. 光纖交換機大多數組態配置是靜態或永久性的，所以需要做的擴充可能更適用於 of-config。

從 of-config 1.1 版本開始，該標準將本身與 FlowVisor（或類似的外部切割代理程式）進行分離，以達成在實體交換機中多台虛擬交換機的抽象層級。這使得工作模型變為實體交換機後，可以有多個內部的邏輯交換機（如圖 3-6 所示）。

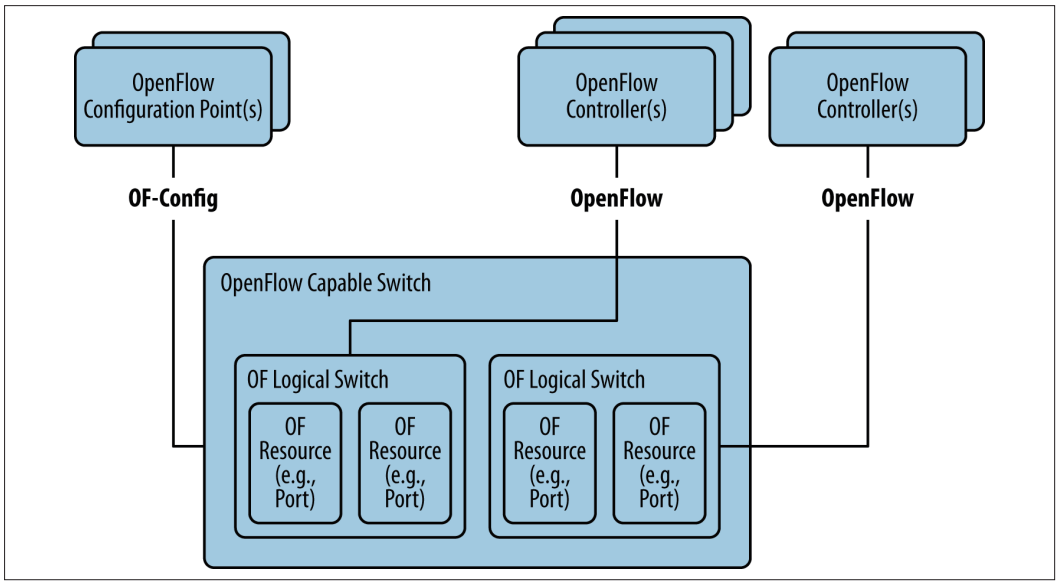

圖 3-6：配置協定與連接協定之間的關係（資料來源：OF-Config 版本 1.1）

使用 of-config 1.1 版本，除了控制器、憑證、連接埠、佇列和交換機能力，我們可以配置一些邏輯隧道類型（IP-In-GRE、NVGRE、VX-LAN）。這個延伸功能，需要交換機支援以便建立邏輯連接埠。

建議進一步擴展 of-config 功能，以及 of-config 協定 1.2 版本中（如圖 3-7 所示），擴充支援更多原生的傳統交換機功能（例如，配置本機 OAM 協定執行程序的能力，已經被提議作為一種延伸功能）。

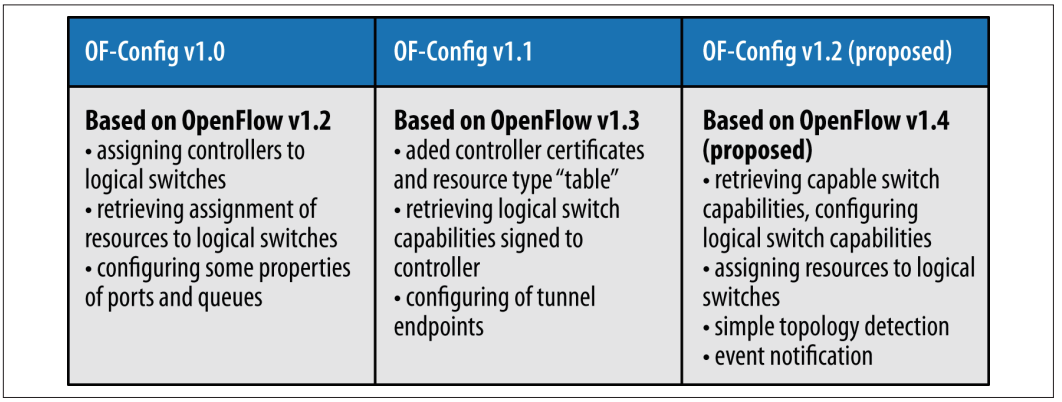

圖 3-7：OF-Config 版本的發展歷程

透過擴充 of-config 支援的本機群組元件，ONF 組織開始討論混合運作的議題，也可能造成一些與標準相關的疑惑[18]。

運作架構工作群組的工作，是負責研究連接協定與配置協定的合併機制。運作架構工作群組章程裡，並沒有輸出任何協定或規範的要求，因此這個合併將來可能由其他工作群組來完成。

對於 NETCONF 的使用，也可以延伸到請求支援的情境（交換機發起的連接），但採用 BEEP（這是 of-config 裡對 NETCONF 連接類型的要求）這個協定，可能需要協定規範的修改或與 IETF 進行協調。

「擴充性（*Extensibility*）」工作群組，負責審查對連接協定提出的功能擴充，並將新功能加入到 OpenFlow 協定當中（欲瞭解 OpenFlow 協定的發展歷程，請參考圖 3-8）。

18. of-config 使用 NETCONF/Yang，所以工作群組正在為這些實體（隧道、OAM）建立自己的 Yang 資料模型。從一個標準組織的角度來看，這可能不是一個很好的功能推進模式。

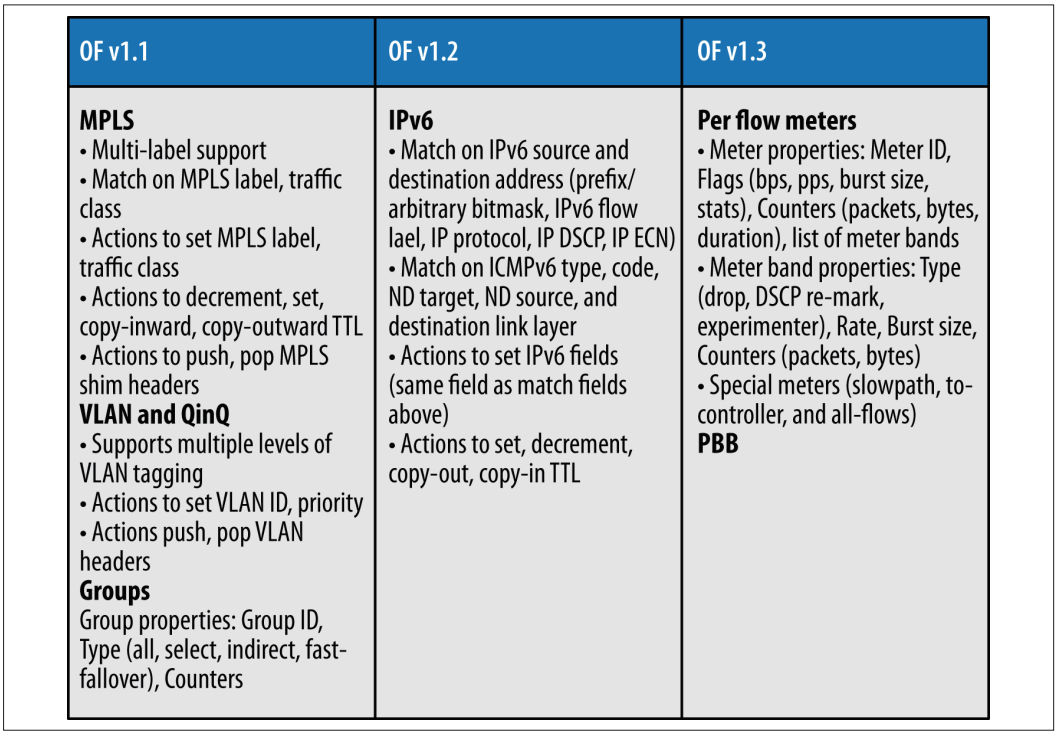

圖 3-8：OpenFlow 協定從 1.1~1.3 版本的特色功能發展歷程

2012 年 4 月，當 OpenFlow 連接協定 1.3 版本發佈時，ONF 組織決定放慢擴充版本的發佈腳步，直到該版本有更高的使用率，並允許臨時的錯誤修正版本（例如，1.3.1 版本修正 1.3 版本中的一些錯誤）[19] 。

19. ONF 組織要求新的擴充或配置管理建議，都需要針對協定的擴充部分進行 POC 概念性驗證（有點像其他標準組織所要求的那樣，標準需要有可自動化運作的程式支援）。

OpenFlow 連接協定 1.4 版本的主要擴充功能，來自於新成立的傳輸討論群組[20]，重點是 OpenFlow 協定與光傳輸網管系統之間的溝通介面，以建立一個標準的廠商傳輸網路控制（例如，佈建）的環境[21]。

有關 OpenFlow 協定 1.4 版本當中的增強功能，將會在本書的未來版本中詳細說明。

傳輸和 OpenFlow 協定的第一次整合，是將光傳輸網路抽象成 OpenFlow 協定可以了解的交換機模型，透過抽象之後的檢視圖來建立一個虛擬的堆疊環境。

目前傳輸討論群組擬定的傳輸方案運作架構，是將光傳輸資訊模型（例如，OTN 光傳輸的網路元件、乙太網路的網路元件和 MPLS-TP 網路元件），與網路的資訊模型（MTOSI、MTNM）組合起來，在加上 OpenFlow 驅動的控制平面進行控制及管理[22]。

即使在直接管控的情況下，各種混合控制平面情境的問題依然存在。也就是針對同一個傳輸網路進行控制行為時，傳統的 EMS/NMS 協定是否可以對 OpenFlow 驅動的控制進行整合。圖 3-9 當中將會說明這種情況。

20. 擴充連接協定來支援光交換技術的建議（編號 EXT-154）。這個擴充功能中，負責處理簡單的波長和連接埠的進一步定義。

21. 事實上，GMPLS 應該提供這種標準化作法。但是，GMPL 中定義、解釋以及實作都非常不一致，導致無法保證在不同廠商設備之間的互通性。

22. 解決方案如果不允許直接控制網路設備，就是利用一個類似 FlowVisor 的功能，並為每個光傳輸設備導入用戶端控制器的概念（適合用於傳輸環境中，普通的業務應用程式）。

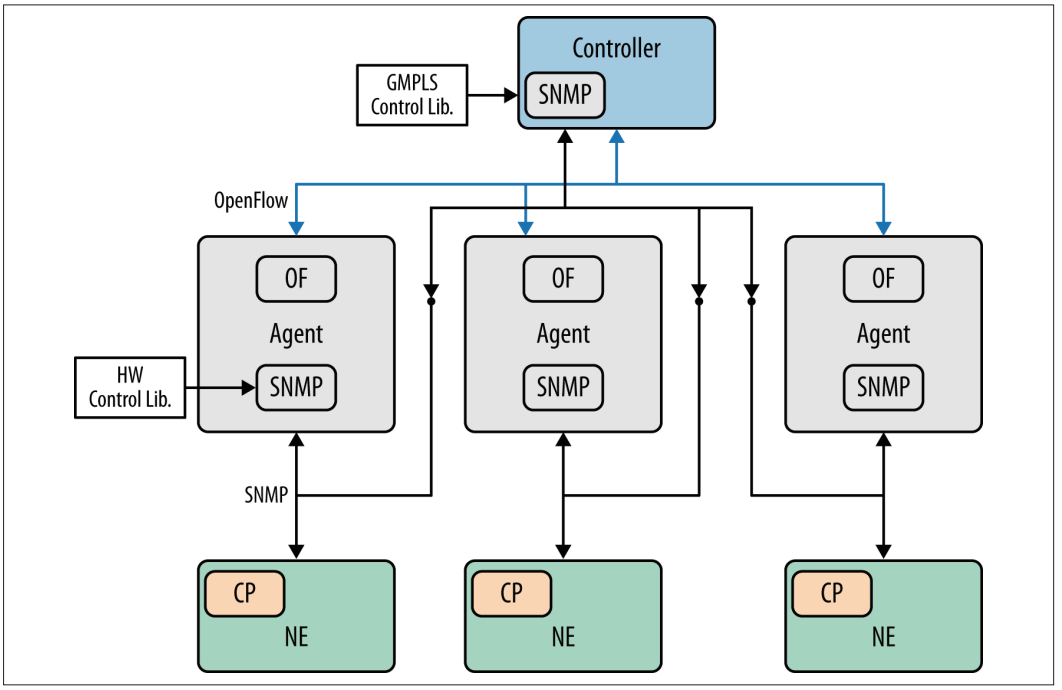

圖 3-9：由 OpenFlow 傳輸網路所控制的混合環境

運作架構

雖然 OpenFlow 協定，已經提供一個標準化的「南向（*Southbound*）」（控制器與網路元件代理程式之間）協定，對資料流程進行管控，但無論是「北向（*Northbound*）」（應用程式介面）API 或「東向（*East*）/ 西向（*West*）」的 API 都還沒有一個確定的統一標準。

大多數可用控制器的「東向 / 西向」狀態分佈，是採用資料庫的分散式模型以允許單一廠商的多控制器組成聯盟，但是並不允許不同廠商控制器之間進行互動操作的狀態交換。

運作架構工作群組，正在嘗試解決這個問題至少是間接的解決，以便為 SDN 技術定義一個通用的運作架構。ONF 組織有段時間將 SDN 的定義和 OpenFlow 定義整合在一起。如果沒有這些標準化的介面，就會出現這樣的問題，也就是 ONF 所定義的 SDN 是否具備開放性。

大多數的 OpenFlow 控制器（如圖 3-10 所示），提供一套基本的應用服務：路徑運算、拓撲結構（透過 LLDP 確定並限制了 L2 拓撲）和佈建。為了支援 of-config，它們需要支援 NETCONF 驅動程式。

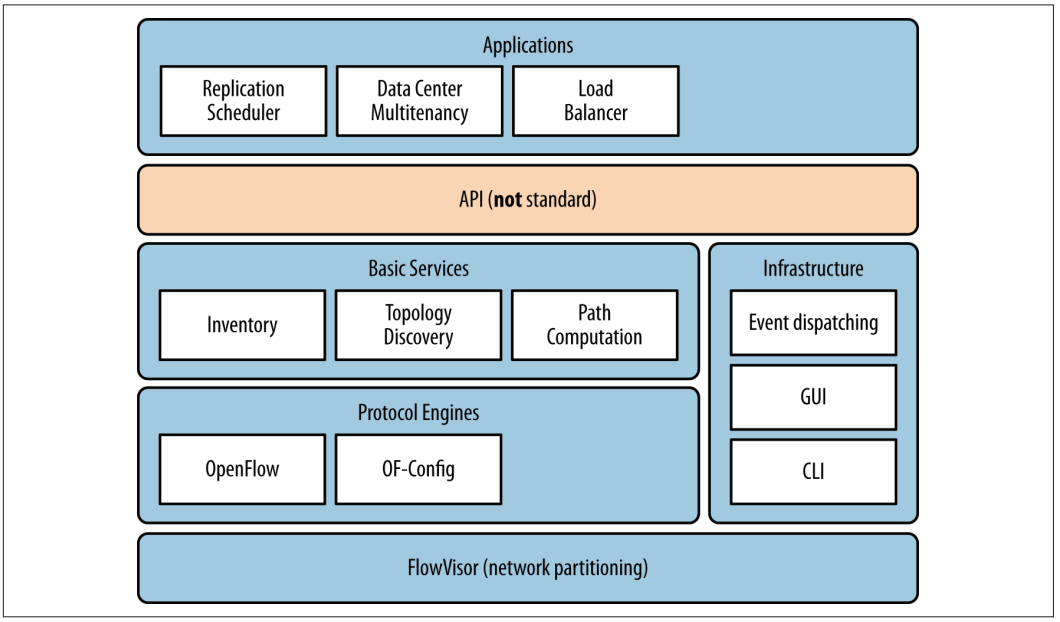

圖 3-10：OpenFlow 控制器元件（FlowVisor 和應用程式是互相獨立的個體）

關於 SDN 運作架構和 OpenFlow 協定一直存在的問題是，OpenFlow 控制器（和 OpenFlow 工作的網路層）提供應用服務的類型，對所有可能的 SDN 應用程式來說是否足夠[23]。

圍繞 OpenFlow 協定模型的主題（例如，故障排除、OpenFlow 語法的原則資料表，以及控制器和網路元件之間驗證層的需要），相關的研究正由許多學者和研究單位進行中，特別是開放式網路研究中心（Open Network Research Center，ONRC）。

混合方案

ONF 組織成立「混合型工作群組（Hybrid Working Group）」。該工作群組建議的運作架構是「夜航（*Ships In the Night*，*SIN*）」模型，另一個是整合的混合模型。ONF 委員會目前僅接受 SIN 模型的推薦。

整合後的混合模式，有一系列圍繞在混合網路安全和疏忽所造成的問題。

23. OpenFlow 協定是否等同於 SDN 技術，一直以來都是一個備受爭議的問題。

假設一個控制的分界點是在網路元件上（在 OpenFlow 控制平面和本地端控制平面之間），安全問題圍繞著如何利用預先保留的連接埠（尤其是 CONTROLLER、NORMAL、FLOOD 和 LOCAL），以允許存取混合網路或本地端網路（傳統網路），其本地端執行程序（應用程式在控制器上或 OpenFlow 連接埠上，可能假冒 IGP 對方端點或其他協定工作階段，來執行插入或匯出的狀態，進而造成本地端網路環境的安全隱憂）。

在一個意外連接建立的混合網路情況下，安全邊界便會進行擴充。這發生在：外部 / 非迴路連結的一端被連接到 OpenFlow 區域，而另一端連接到一個本地原生區域當中。

新成立的安全工作群組可以解決混合安全的隱憂，在撰寫本書時雖然還沒有獲得足夠的資訊來進行討論。但在本書的未來版本當中，將會針對這部份進行更詳細的說明及討論。

夜航模型

夜航模型（SIN Model），是假設一個連接埠（實體或邏輯）只能用於 OpenFlow 或本地端，但不能同時用於兩種環境（如圖 3-11 所示）。夜航模型的重點在於：

- 限制 OpenFlow 行程所分配的資源，使其無法影響本地端網路環境的操作行為（反之亦然，限制本地端網路所分配的資源，使其影響 OpenFlow 網路的操作）。提出的建議包括在本地端主機（混合模式交換機），其作業系統當中（或透過虛擬化）採用現代化執行程序層級的隔離機制。
- 避免 OpenFlow 網路，與本地端網路的控制平面之間同步狀態的需求，或不同步事件通知。
- 針對使用 LOCAL、NORMAL 和 FLOOD…等，保留連接埠的流量並採取嚴格的規則（有明確的注意事項）。

SIN 擴充 ONF 之前對混合方式的定義（反映在 NORMAL 這種預先保留連接埠的定義中）。

SIN 模型允許連接埠透過邏輯連接埠或 VLAN 來隔離，並推薦在這樣的環境下使用 MSTP 作為產生樹達成（此步驟對於某些特定類型的整合式混合是必要的）。

最後，SIN 指出預先保留連接埠之間的互動還是含糊不清的，鬆散的連接埠委託模型是 SIN 混合方式的潛在改進領域。

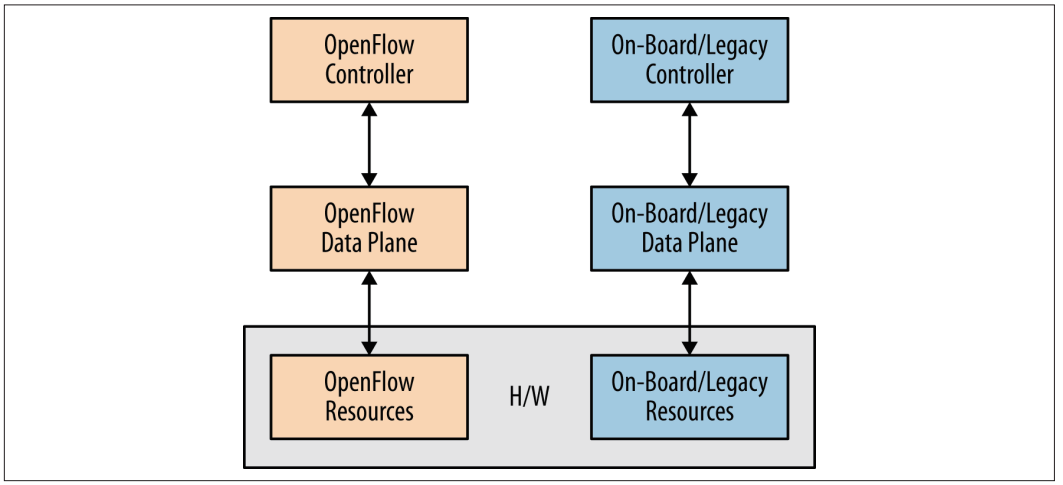

圖 3-11： SIN 運作架構（資料來源：ONF 混合式 SIN 工作群組）

雙功能交換機

事實上，混合工作群組的整合式運作架構白皮書被 ONF 組織拒絕了，ONF 董事會後來建議成立遷移工作群組，以協助 OpenFlow 的使用者進行 OpenFlow 網路架構的部署，而無須經過混合使用的過渡期。然而，對於整合混合式運作架構的需求仍然存在，而新成立的遷移工作群組可能會解決混合設備和混合網路的問題[24]。

一個整合現有 / 已部署的模型，將針對 OpenFlow 區域與本地區域在控制層級（例如，RouteFlow）上進行整合。與整合式混合架構不同的是，這種方式將會特地建立一個混合式網路環境（如圖 3-12 所示）。

這種作法背後的概念，是在虛擬主機上運作路由通訊協定，並將主機上 VM 虛擬主機管理程式的虛擬交換機虛擬連接埠，與 OpenFlow 交換機相關的實體連接埠進行繫結。透過這些連接埠，虛擬路由器與本地網路的 IGP 和 BGP 在適當的實體邊界交換機上，透過開啟流程資料表中相對應的流程形成鄰接關係。虛擬路由器透過適當的邊界點，發佈給 OpenFlow 區域的前置碼（從本地網路來看，它們就像是相鄰對等所學習到的資訊）。此外，虛擬路由器在 OpenFlow 區域內建立流程規則（透過內部邏輯和原則），將流量導引到從鄰居那裡所學習到的目的地，透過流程規則最後再指向邊界交換機適合的連接埠。

24. 對於希望採用混合網路環境的人來說，混合式網路設計方案其運作結構類似一個資料中心堆疊模型，在 ONS 的 2013 年會議上被提出。

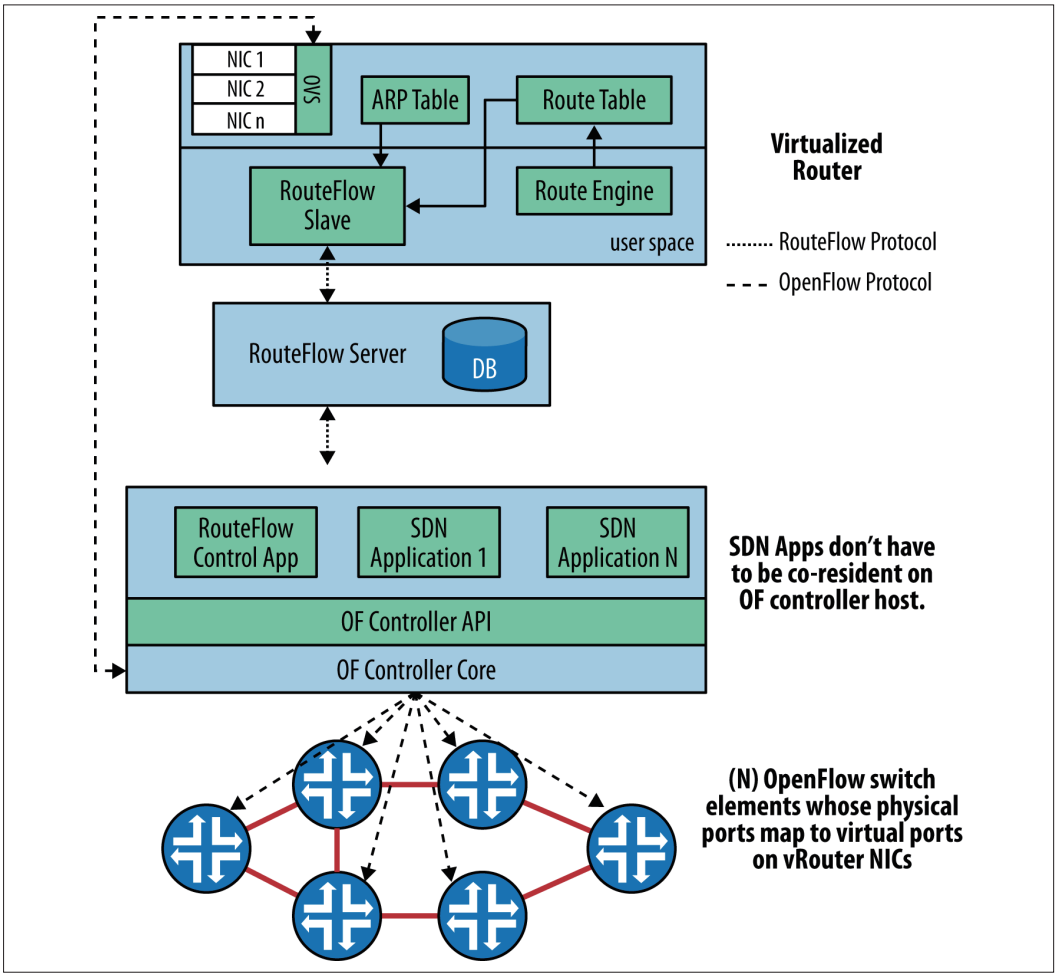

圖 3-12：RouteFlow 的運作架構（資料來源：http://cpqd.github.io/RouteFlow/）

這種混合設計的缺點是，流量管理和封包 I/O 連續在一個共同的 TCP 工作階段上發生，這使得該設計回到傳統的分散式控制平面要解決的問題，也就是阻塞、控制封包 I/O、延遲、佇列管理以及硬體程式設計速度。其中一些問題可以透過其他控制通道（在 OpenFlow 1.3 版本中提出）得到改善，這種想法將會在 OpenFlow 連接協定中發展並日趨成熟。

目前，能用於整合式混合連接（在同一設備上的 OpenFlow 協定和本地端協定）的工具是：資料表及溝通介面。

採用資料表的方案，可以使用 OpenFlow 的 GoToTable 語法來設計，在本地端資料表裡進行二次尋找任務。目前，OpenFlow 並不瞭解除了它本身資料表之外的其他資料表。解決方案是允許在工作階段初始化時，所發現的本地端資料表。但是，該解決方案有下列問題：

- 對於本地區域中 VRF（虛擬路由轉送）資料表的名稱空間而言，OpenFlow 的資料表空間太窄。
- 本地端可以建立大量的動態資料表，特別是在服務供應商邊界或資料中心閘道設備上，該資料表需要更新到控制器（重新開機工作階段可能是繁重的，而動態探索機制又需要更多的標準化作業程序）。
- 本地區域在某些設備上，可以有 64 個以上的資料表。
- 儘管 GoToTable 方案簡潔（包含所有透明度的假設），但是它似乎是一種複雜且影響重大的路由機制。

目前有一些非官方的溝通介面解決方案，以達成區域之間的雙向流程。最常見的是在 OpenFlow 交換機區域中，插入一個新的 L3 元件。此 L3 元件可以被 NOMRAL 連接埠行為、DHCP 和 ARP 的組合所影響，如此一來終端主機就可以在 OpenFlow 區域內發現一個轉送閘道設備。此方案雖然可以運作，但距離完整解決方案還很遙遠。在 OpenFlow 部份，NORMAL 邏輯連接埠僅用於出口，所以在相反方向上控制流程是不可能的。此外，一些系統管理人員 / 操作人員基於安全性考量，並不喜歡使用 NORMAL 連接埠。

建立規則，以直接交叉連接 L3 新元件與 OpenFlow 控制的連接埠，進而支援入口和出口的規則，這是可以做到的。如果進一步對溝通介面的定義進行延伸，那麼就能夠針對 L3 轉送器或本地端連接埠打上 Tag 標籤（這裡不討論標籤的語法）。例如，在 Juniper 的 Junos 作業系統中，有一種稱為邏輯隧道的結構（如圖 3-13 所示）。這種結構可以一端在 OpenFlow 區域，另一端在本地端的任何路由區域。對於我們來說，這提供一個可延伸、透明的混合解決方案，但我們唯一可以進行的操作只有標籤而已（因為，這是 Juniper 的獨家技術）。

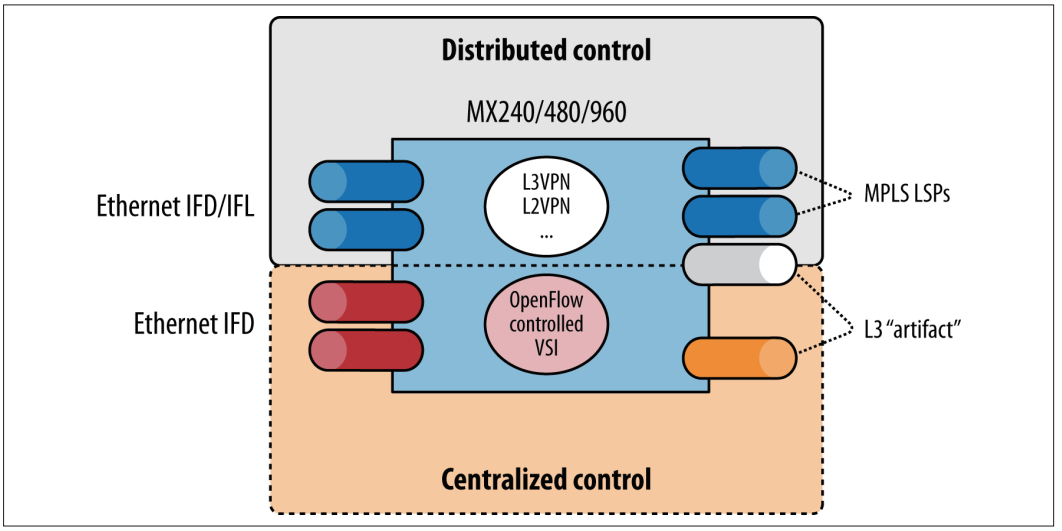

圖 3-13：Juniper 的混合式環境整合解決方案（資料來源：Juniper）

採用溝通介面的解決方案，需要執行以下操作：

- 至少一個連接埠的延伸描述，將本地端的新元件進行標記（區域之間的存取點）。其他額外的屬性可能指出區域的內容（例如，IP/MPLS），以及它們的路由執行個體。這些屬性可以與廠商代理程式在 port-status 訊息或 Features_reply 訊息中，進一步的做為連接埠資訊來互相交換。（這些是對 OpenFlow 標準所建議的延伸屬性）。
- 廠商代理程式，應該達成雙向流程所需的所有 MAC 功能（自動將新元件的 MAC 位址，分配給這個點的前置碼或在本地資料表中指出該元件）。
- 廠商代理程式應該支援 OpenFlow-ARP 的相關功能，以便 OpenFlow 區域內的設備可以發現新元件的 MAC 位址 。
- 原生連接埠可以建立一個內部的迴路連接埠（首選方案），或形成為一個外部的迴路（對稱解決方案優於非對稱解決方案）。
- 如果某些應用程式透過 LLDP 拓撲探索，排除本地端元件 / 連接埠是可行的（此為預設的操作行為）。

混合式的整合支援虛擬介面（例如，共用一個連結到 VLAN Tag）。任何共用連結（支援 VLAN 中繼的一個連接埠）的外部 / 本地功能，都應該在跨越兩個區域的整個流量中

生效（區域平行作業，但不要交叉連接）[25]。此外，本地端介面功能可以在元件（連接各區域）裡應用，但是並沒有假設它們必須被支援。此行為與廠商有關，並且支援、後果（意外行為）功能的順序都需要廠商，以便向他們的客戶提供明確定義。

總結

目前，大家認為是由 OpenFlow（和標準組織 ONF）開啟 SDN 技術的討論，並提供現代化 SDN 技術的第一印象：一個管控機制的集中點，一個說明拓撲的北向 API、路徑運算、為控制器上層的應用提供服務，以及標準化用於一家或多家廠商基礎架構中，對於轉送狀態進行的南向協定。

可惜的是，OpenFlow 運作架構並沒有提供一個標準化的北向 API，也沒有提供一個標準化的東西向分佈狀態協定，以達成兩個應用程式的可攜性和控制器廠商之間的互通性。標準化工作可以透過 ONF 新成立的運作架構工作群組，甚至是新的開放原始碼組織 OpenDaylight 專案向前推進。

OpenFlow 為那些利用全套 OpenFlow 原始架構的平臺，提供大量的流程 / 流量管控機制。ONF 已經成立一個工作群組，來協助解決當廠商使用 OpenFlow 原始架構建構網路應用模型時，廠商硬體實作能力的描述與探索問題。

儘管對 OpenFlow 所達成的抽象水準，以及最後的 API 資料表是否為一個完整的 SDN API 等問題還有疑問，但是相關應用方面仍很有意思，而且持續在混合式操作方面的努力不懈，可能會更容易的整合在傳統分散式網路，或者是在 OpenFlow 區域與本地區域之間的邊界上，比對和衡量網路流量的能力。

25. 使用者端已經請求，透過一種方法在實體連接埠上使用 QoS 的能力，以防止某個區域（本地區域或 OpenFlow 區域）的 VLAN 在共用鏈路上，消耗大量且過多的頻寬。