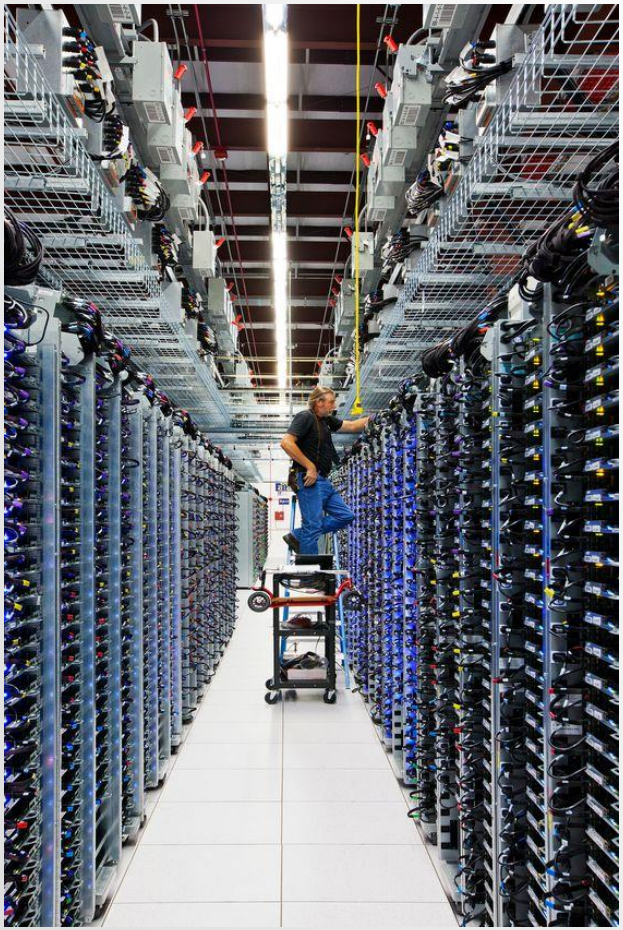
軟體定義網路(SDN)簡介與發展

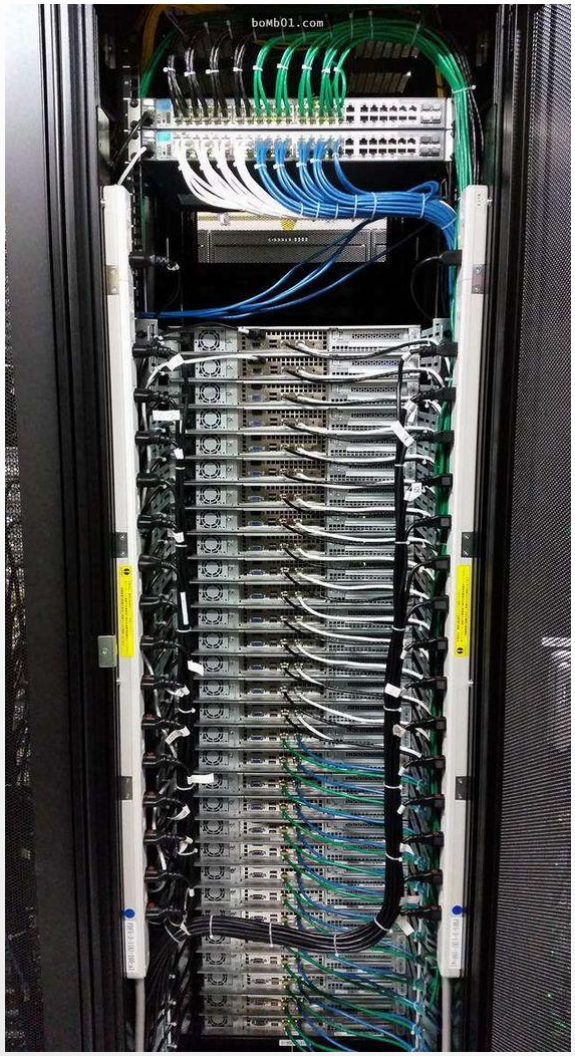
數位活氧科技 高銘聰

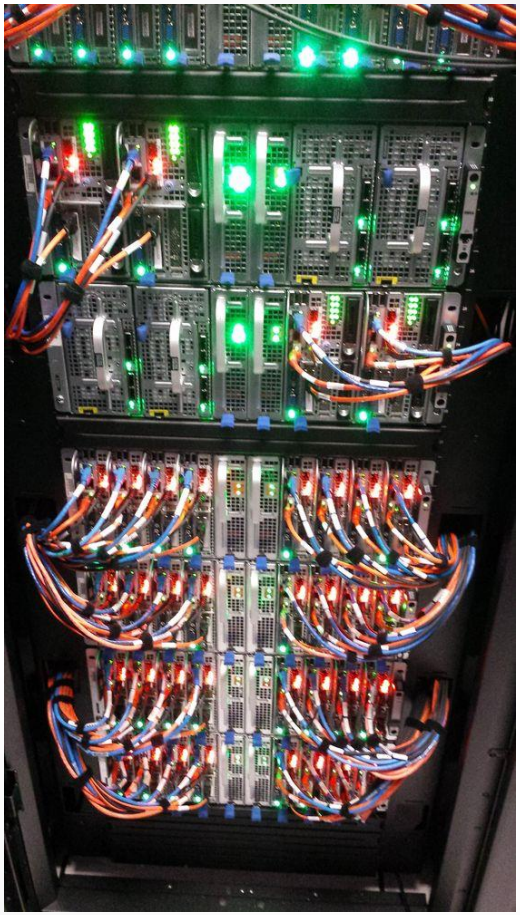
OUTLINE

- 傳統網路 vs SDN
- 什麼是 SDN?
- SDN的運作方式
- 發展、使用 SDN?

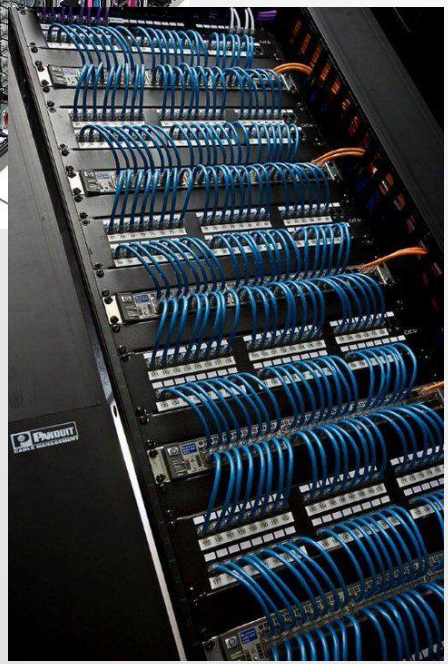
傳統網路 DATA CENTER

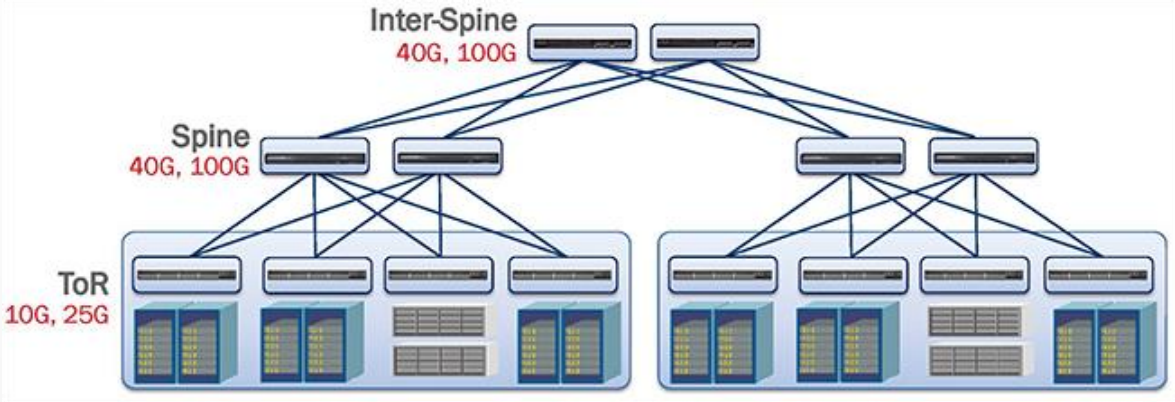


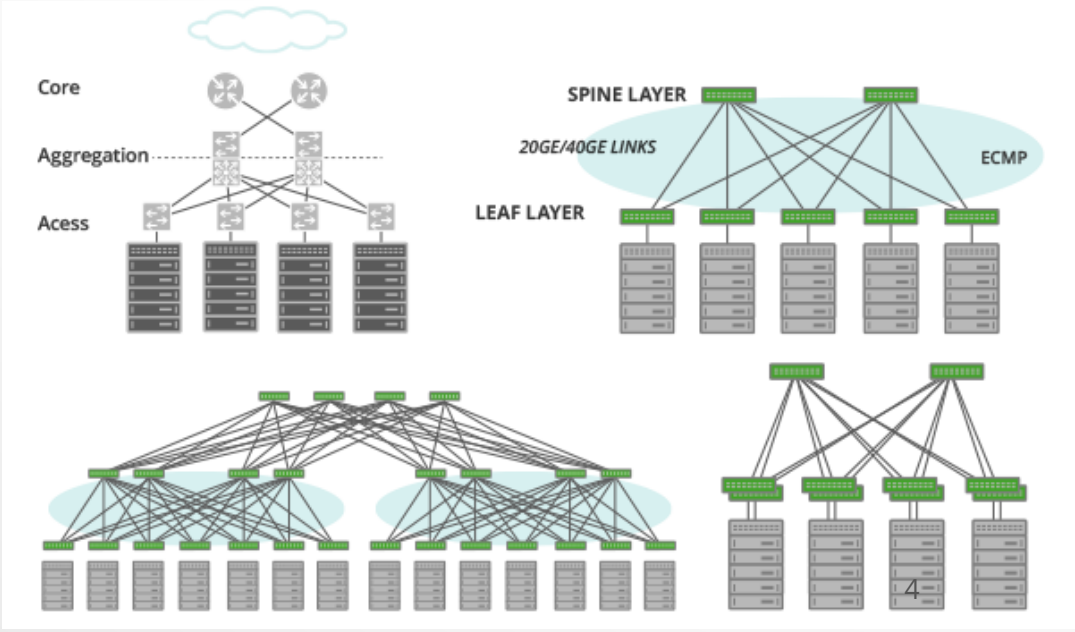




傳統網路 DATA CENTER 拓樸







NETWORK DEVICES

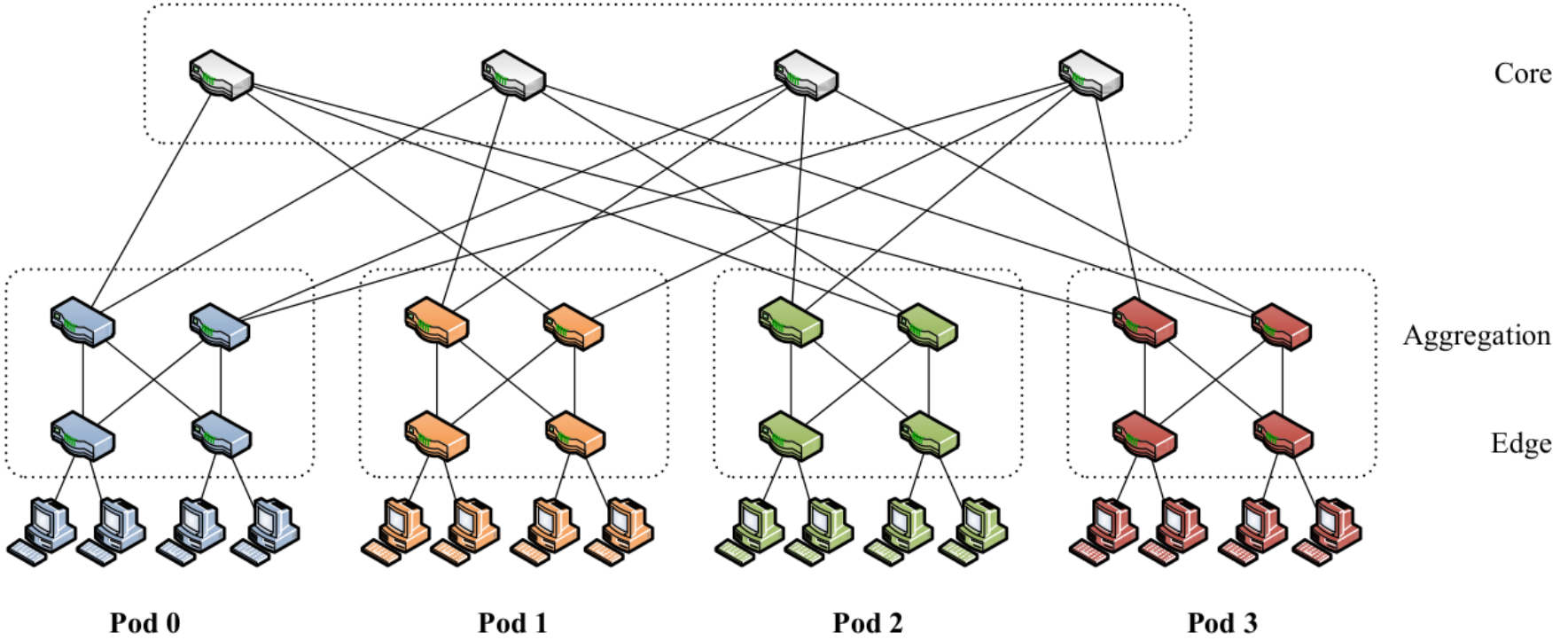
CISCO Core Router
Cisco 7604

F5 Load Balancer

CISCO Switches

NetScreen Firewall

傳統網路 FAT TREE TOPOLOGY



現今的網路架構越來越不敷使用

- 現今的網路架構是建立於擴展樹協定（Spanning Tree Protocol，STP）上的三層式架構，透過各種傳輸協定來傳送封包，然而，隨著雲端應用服務及巨量資料需求日益增加，網際網路的路由表越來越複雜，讓目前的網路架構產生了許多問題，越來越不敷使用。

- 為了要實現各種網路協定，交換器或是路由器必須不斷的拆分及重組封包，導致傳輸效率不佳，無法有效發揮網路頻寬；

- 網路管理人員需要客製調整各種網路設定時，必須針對每臺交換器或路由器，逐一登入命令執行介面（command-line interface，CLI）設定，相當麻煩，也不易快速變動網路架構來因應企業建置新系統的需求。而且透過人工逐一設定的方式也有很高的風險，一旦網路管理人員輸入了錯誤的指令，很容易造成網路服務癱瘓。

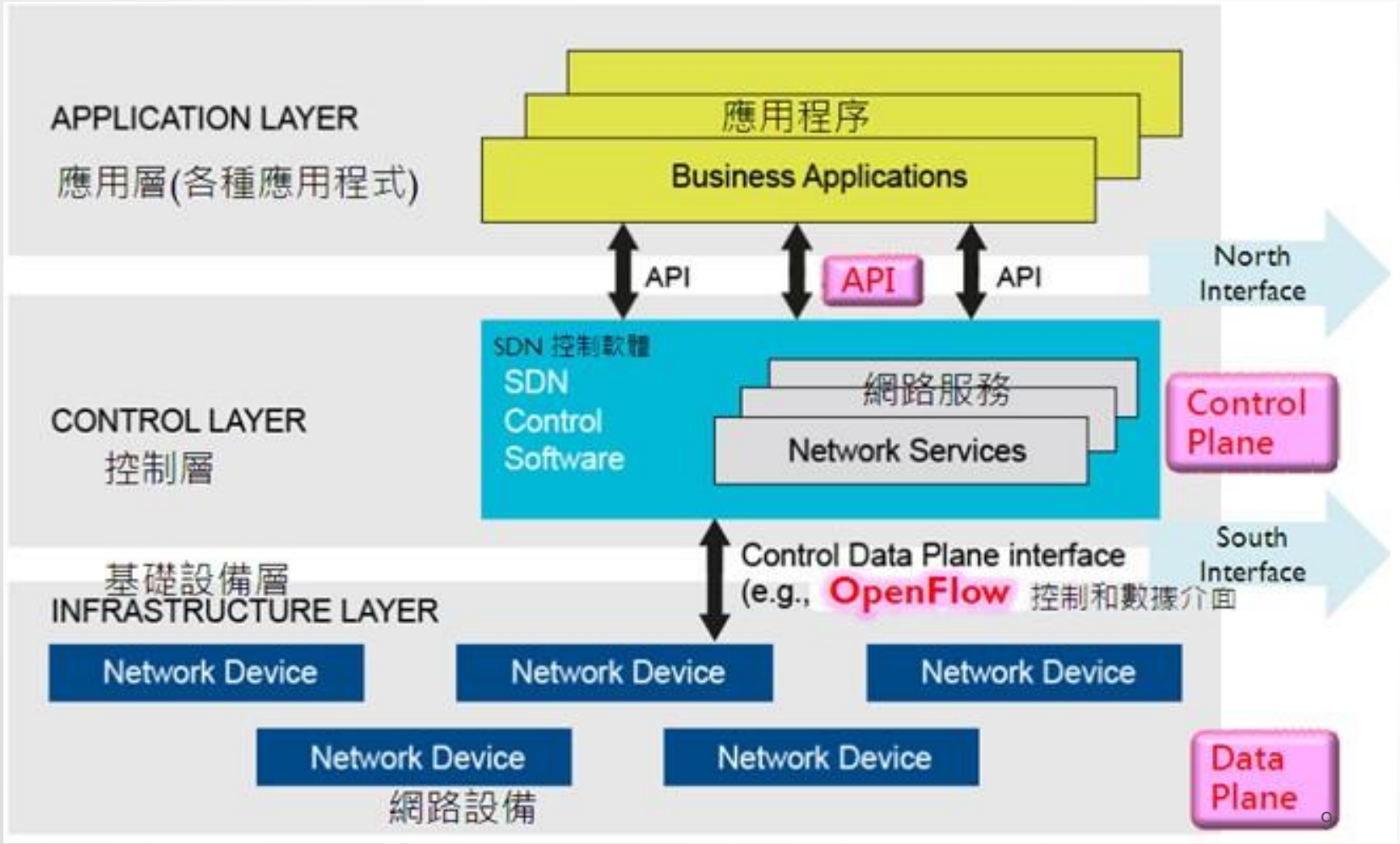
- 此外，網通廠商的設備雖然能通過共通的協定進行傳輸，但是各有各的網路管理技術或是網路作業系統軟體，網管軟體彼此之間難以相容，一旦企業購買某一廠牌的設備，未來更新設備時就必須遷就於該廠牌的網管功能，無法選用其他廠牌的設備，造成被網通廠商挾持的情形。

軟體定義網路(SDN)

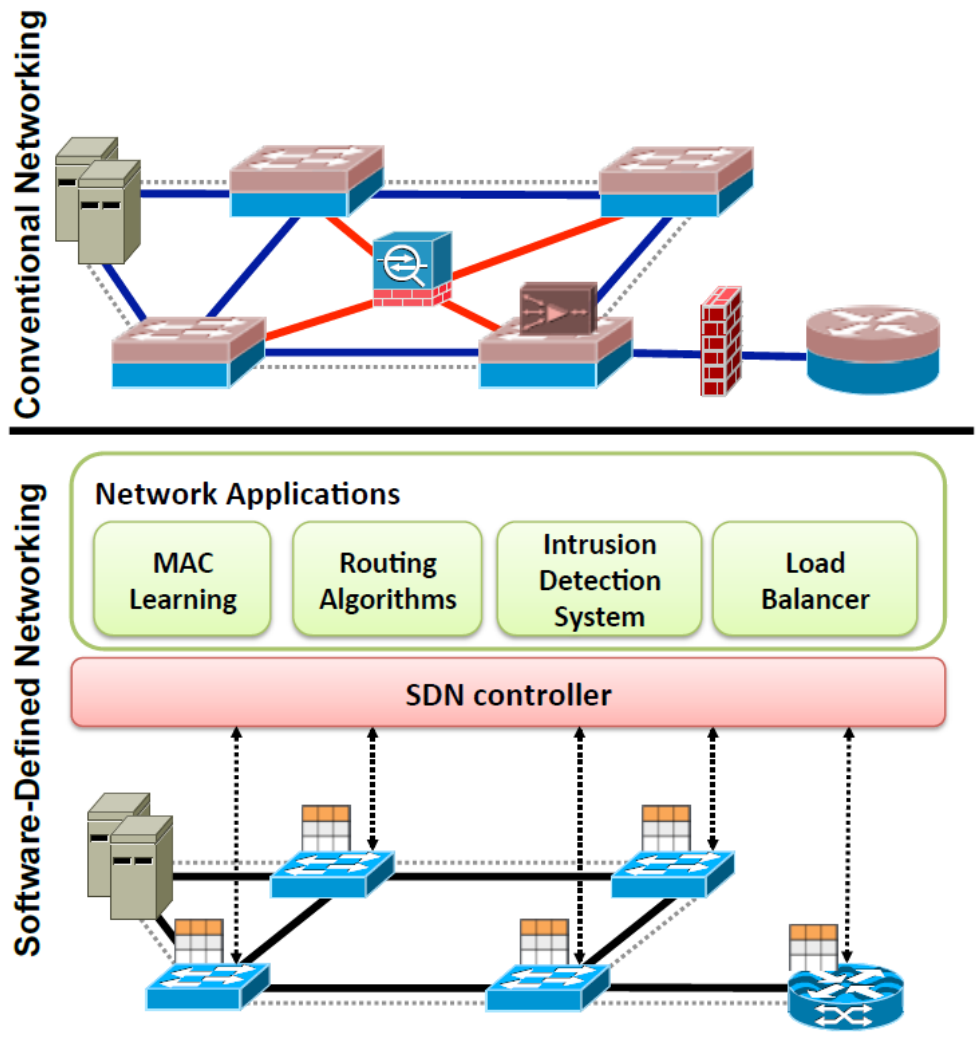
- 軟體定義網路（英語：Software-defined networking，縮寫為SDN），一種新的網路架構。利用OpenFlow協定，把路由器的控制平面（control plane）從資料平面（data plane）中分離出來，以軟體方式實作。這個架構可以讓網路管理員，在不更動硬體裝置的前提下，以中央控制方式，用程式重新規劃網路，為控制網路流量提供了新的方法，也提供了核心網路及應用創新的良好平台。

- 真正讓SDN對企業具有重要意義的主要有三大因素：自動化、快速應用部署，以及簡單的網絡管理。

SDN 架構

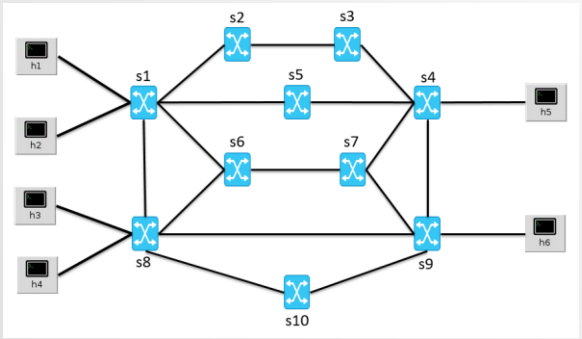

APPLICATION LAYER
應用層(各種應用程式)
應用程序
Business Applications
API
API
API
North Interface
SDN 控制軟體
SDN Control Software
網路服務
Network Services
Control Plane
CONTROL LAYER
控制層
Control Data Plane interface
(e.g., OpenFlow 控制和數據介面
South Interface
基礎設備層
INFRASTRUCTURE LAYER
Network Device
Network Device
Network Device
Network Device
Network Device
網路設備
Data Plane

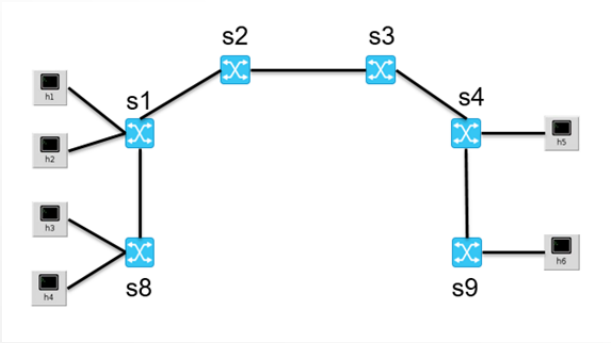
傳統網路 VS SDN


Conventional Networking
Software-Defined Networking
Network Applications
MAC Learning
Routing Algorithms
Intrusion Detection System
Load Balancer
SDN controller

傳統網路 VS SDN

- Ethernet network use Spanning Tree Protocol(STP) to forward frame.
  - IEEE 802.1D
  - avoid loop and ARP storm
  - analysis and decide which port can transmit
  - Single path routing
- STP





傳統網路 VS SDN

- Load balancing
  - Achieve higher bandwidth utilization
  - Balancing the traffic load
  - Static load balancing
  - Dynamic load balancing

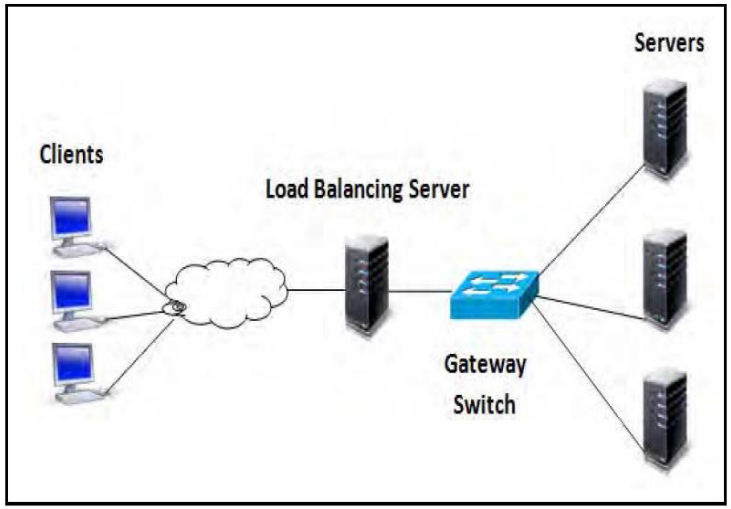


Figure 1. Ttraditional load balancing model

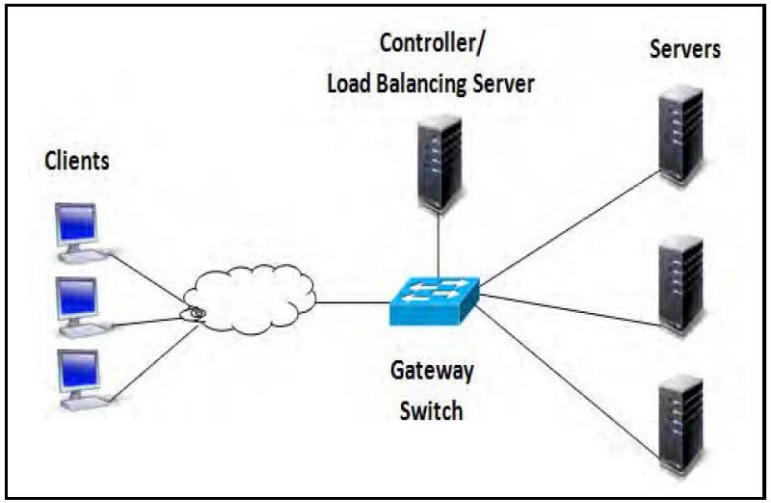


Figure 2. SDN based load balancing model

SDN為新一代網路概念與架構

- 將全網的Control Plane與Data Plane完全獨立。
- 透過Controller軟體來集中管理全網資料流量行為。
- Controller軟體提供了可程式化介面(API)可與其他上層設備(如VM)，做更進一步整合。
- 利用可程式化介面(API)可以用使用者發展出多樣的附加服務在Controller上，如Firewall、IDP，
- DPI(Deep Packet Inspection) ，LB(Load Balance)，Schedule ...等，可做統一佈署，提供更多元化服務項目給企業使用。

SDN 架構

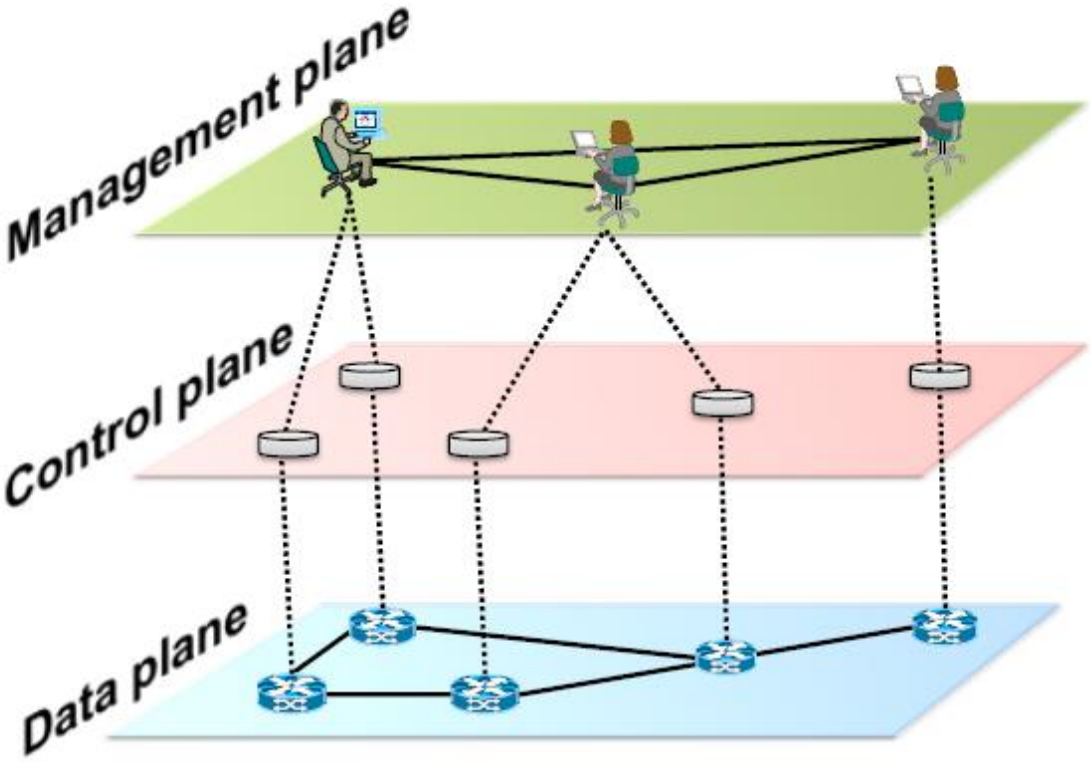
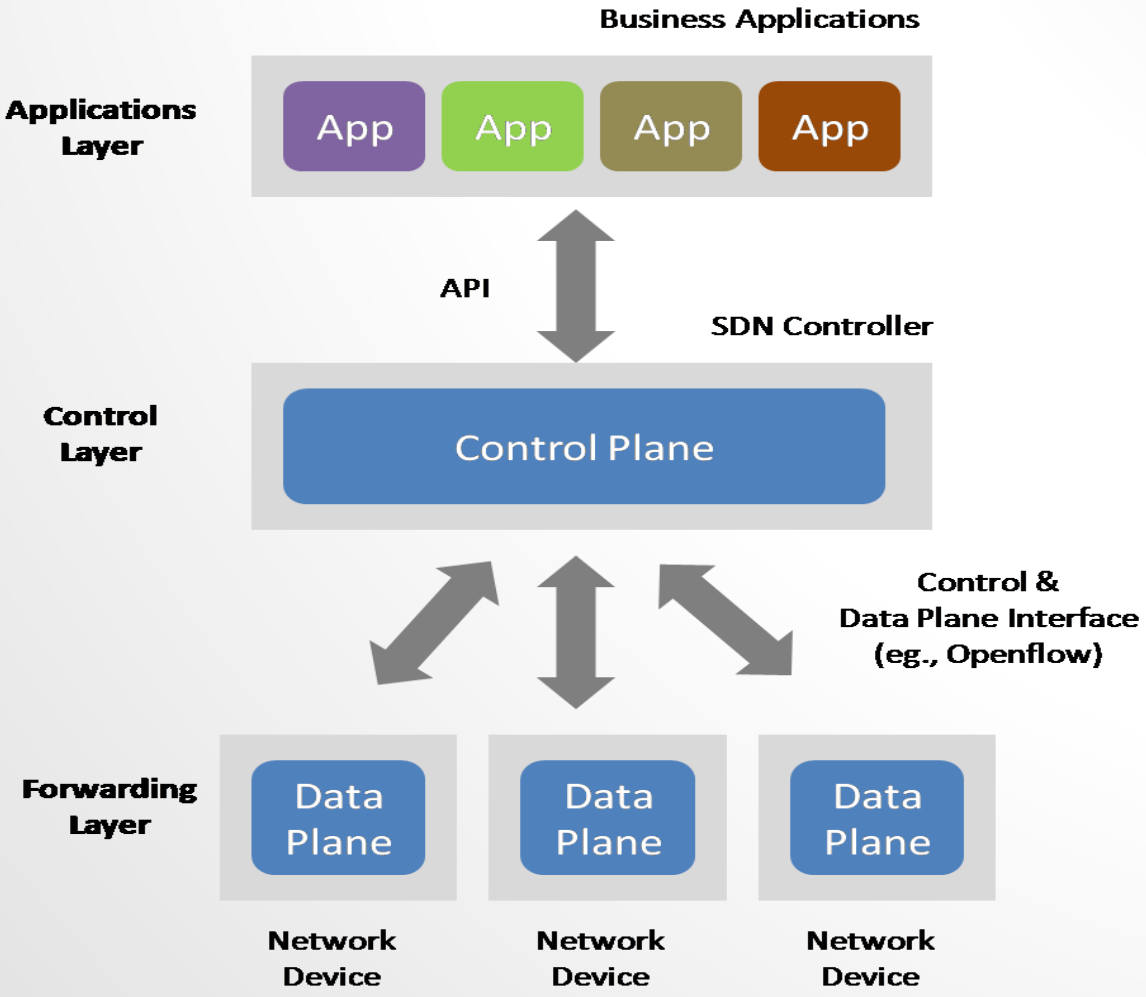
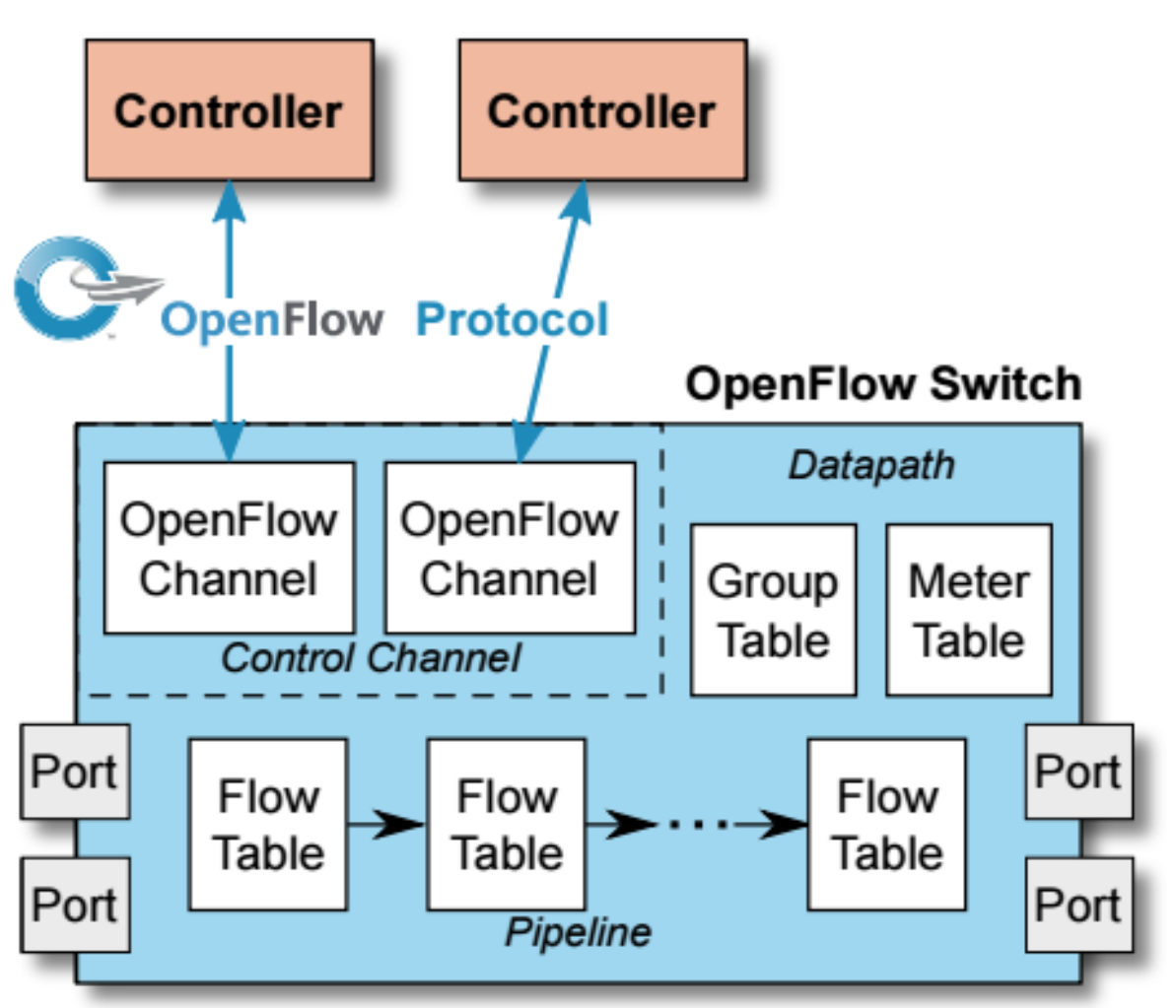


Fig. 3. Layered view of networking functionality.

SDN 架構


Business Applications
Applications Layer
App
App
App
App
API
SDN Controller
Control Layer
Control Plane
Control &
Data Plane Interface
(eg., Openflow)
Forwarding Layer
Data Plane
Data Plane
Data Plane
Network Device
Network Device
Network Device

OPENFLOW 協定架構圖



OPENFLOW 協定架構圖

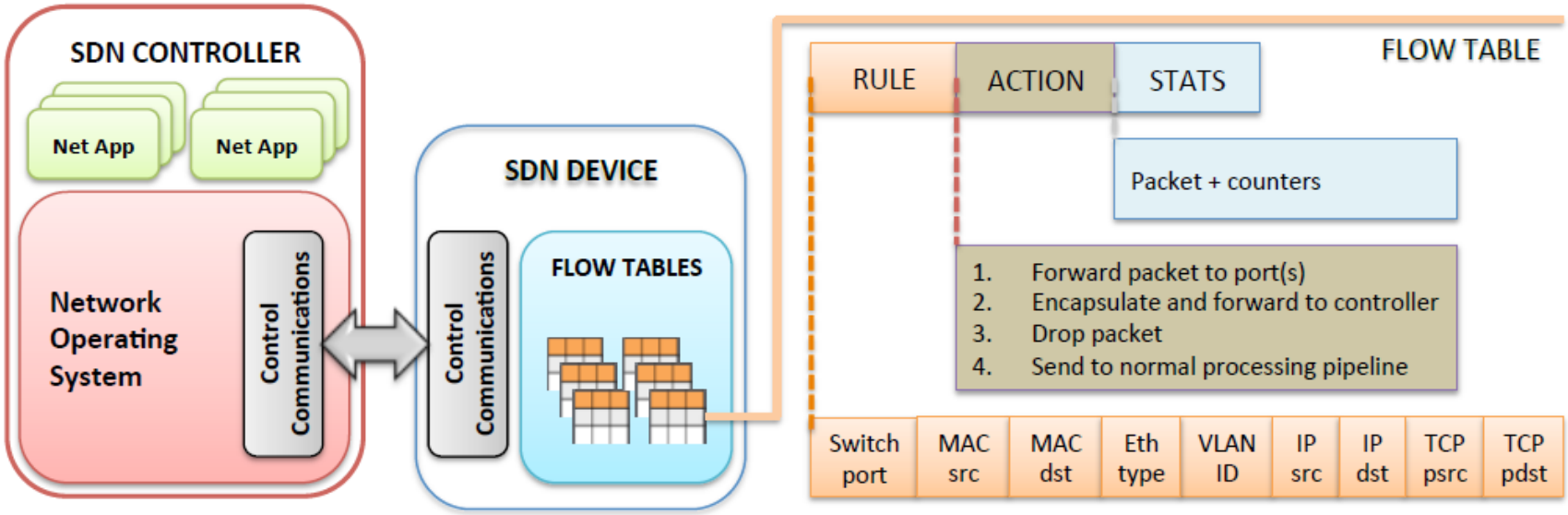
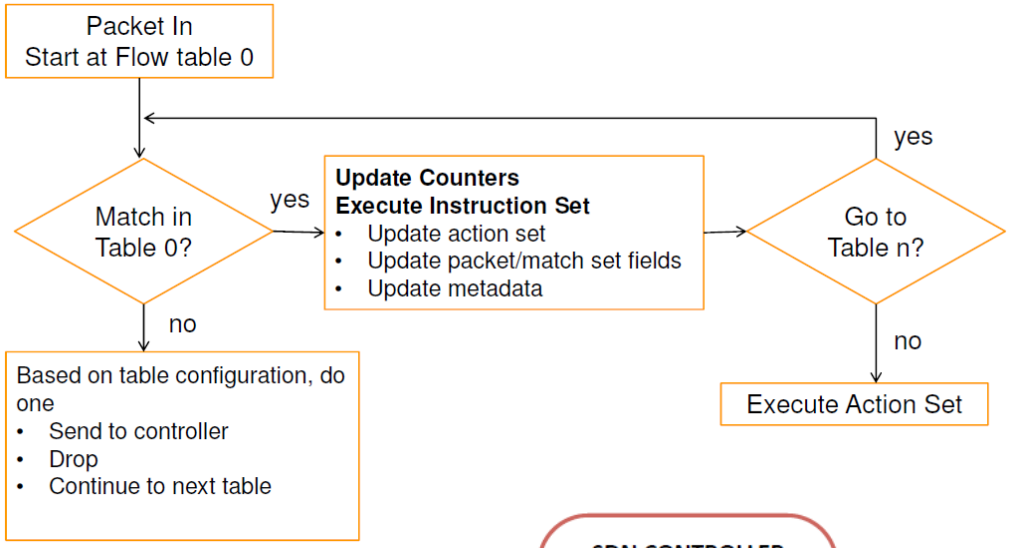
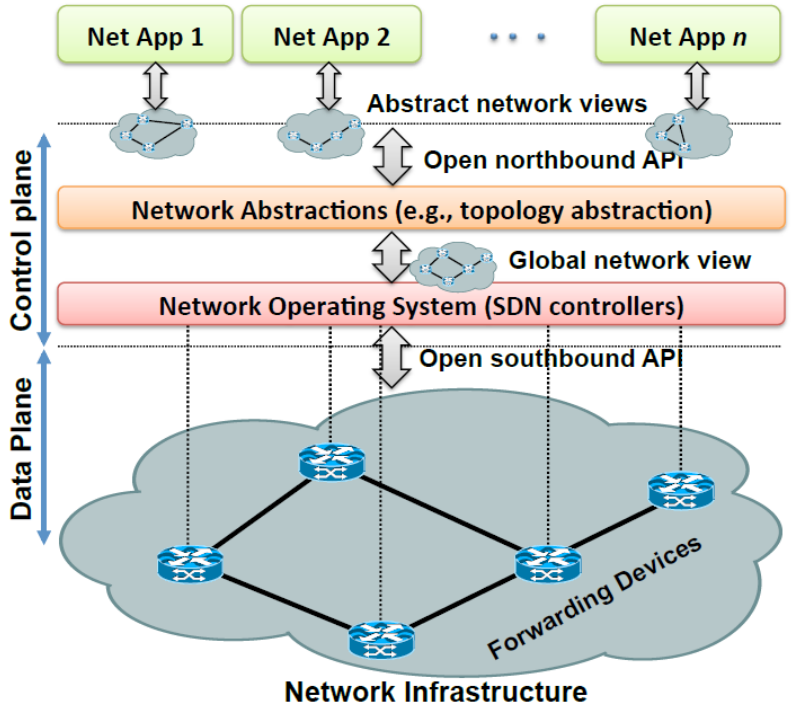


Fig. 7. OpenFlow-enabled SDN devices

PACKET-IN 處理流程圖





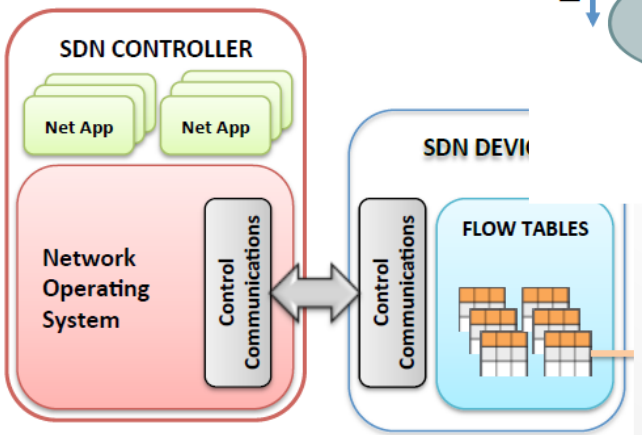
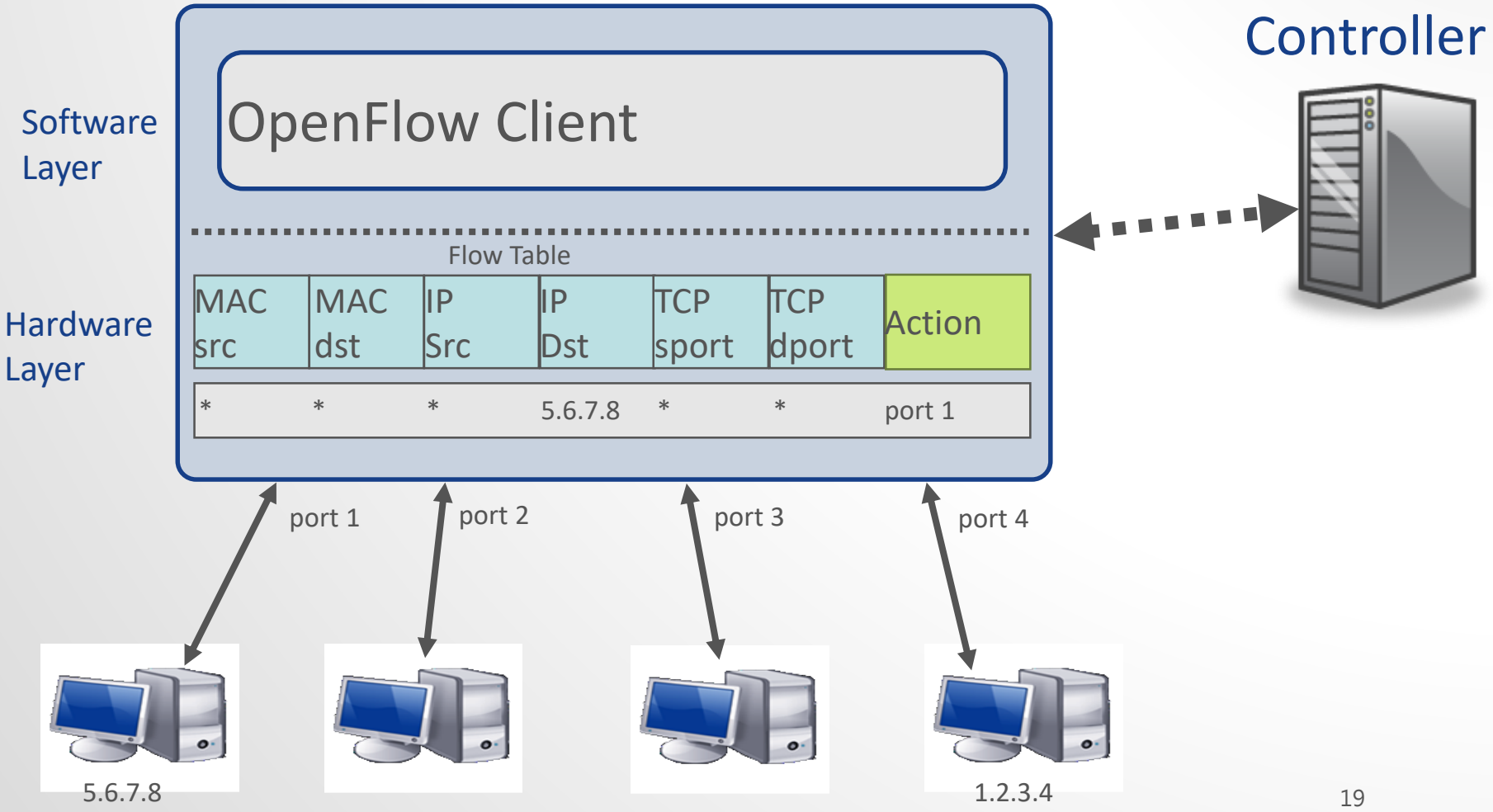


Fig. 7. OpenFlow-enabled SDN devices

FLOW TABLE EXAMPLE 1



FLOW TABLE EXAMPLE 2

Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

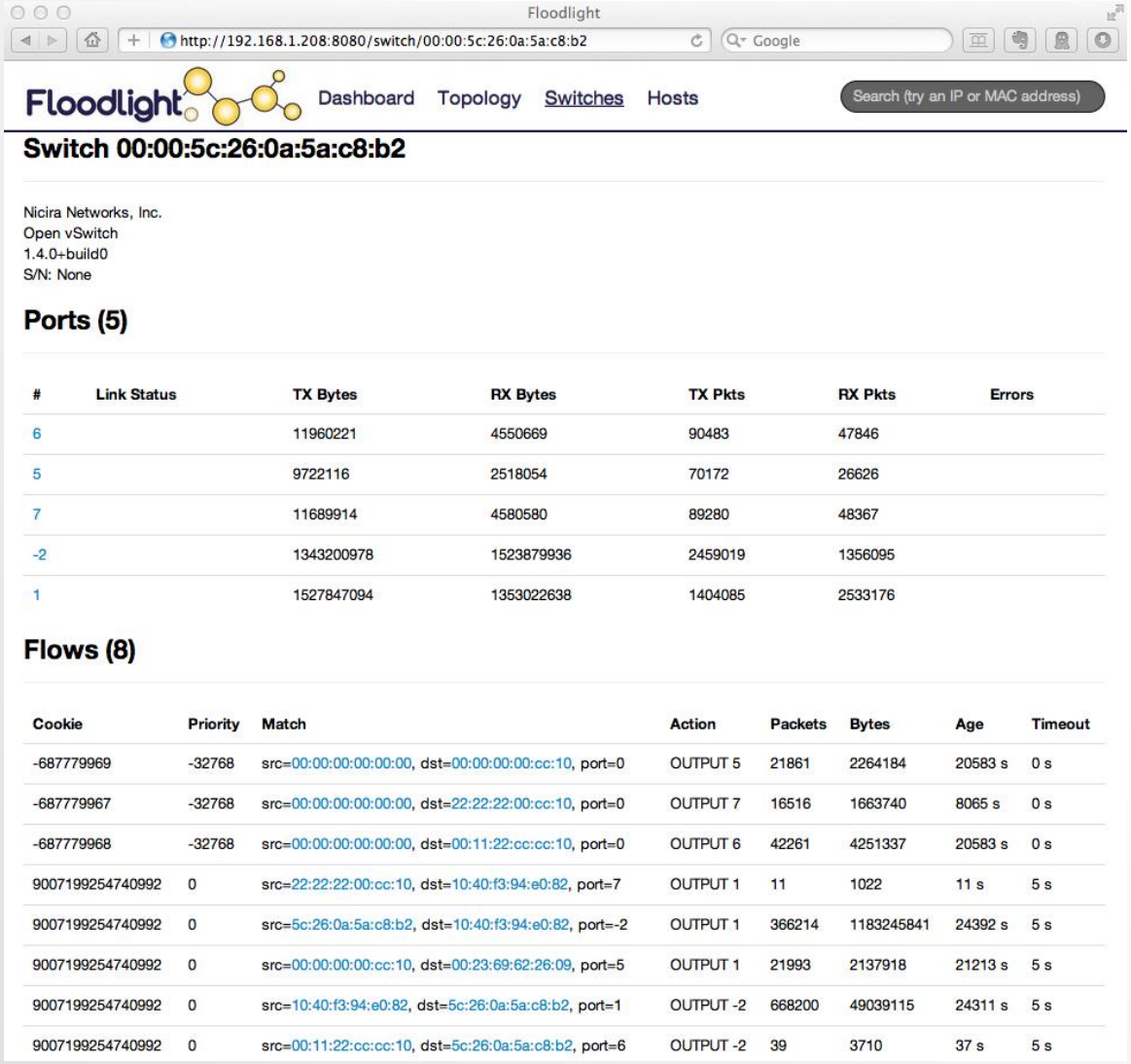
Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

FLOW EXAMPLE 1

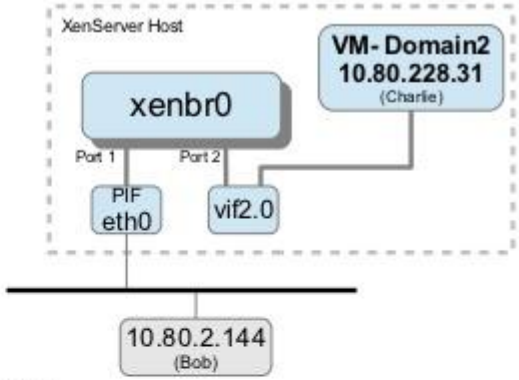


FLOW EXAMPLE 2

Flow Table Entries

- VM Charlie sends a ping to Bob.

```
# ovs-dpctl show xenbr0
system@xapi0:
        lookups: hit:103033 missed:77944 lost:0
        flows: 30
        port 0: xenbr0 (internal)
        port 1: eth0
        port 2: vif2.0
```



- Then dump flows:

```
# ovs-dpctl dump-flows xenbr0 | grep "10.80.2.144"
```

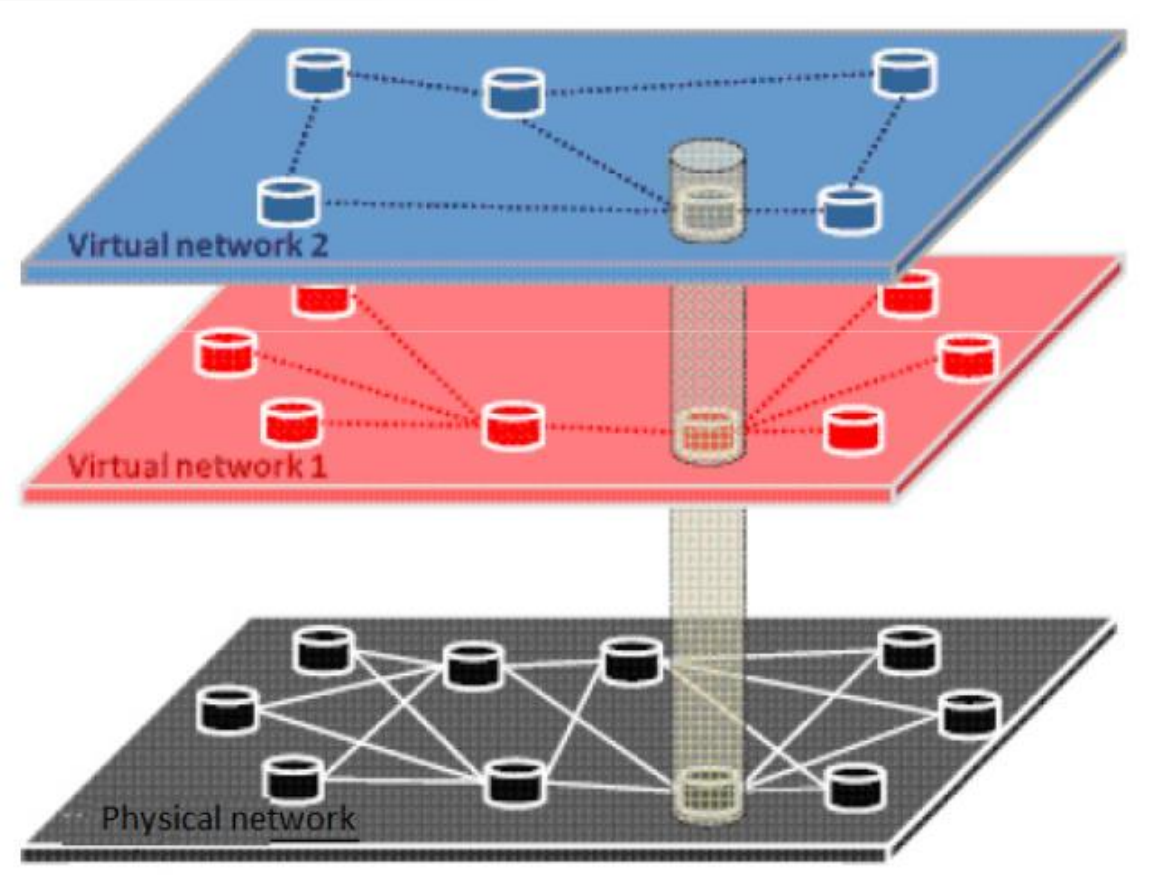
Flow 1:
```
in_port(2),eth(src=72:41:36:a2:8c:d9,dst=00:21:1b:f3:63:45),eth_type(0x0800),i
pv4(src=10.80.228.31,dst=10.80.2.144,proto=1,tos=0,ttl=64,frag=no),icmp(type=8
,code=0), packets:5013, bytes:491274, used:0.760s, actions:1
```

Flow 2:
```
in_port(1),eth(src=00:21:1b:f3:63:45,dst=72:41:36:a2:8c:d9),eth_type(0x0800),i
pv4(src=10.80.2.144,dst=10.80.228.31,proto=1,tos=0,ttl=62,frag=no),icmp(type=0
,code=0), packets:5013, bytes:491274, used:0.760s, actions:2
```

- L2-L4 Exact Match
- Total Number of packet matches of this type
- Total Number of bytes for this flow match
- Time flow was last updated
- Actions. In this case switch packet to Port 2.

NETWORK SLICE

- OpenFlow將實際網路(物理層)切割成許多虛擬的網路拓樸提供給不同的用戶/服務。



OPENFLOW 協定架構圖

TABLE III

DIFFERENT MATCH FIELDS, STATISTICS AND CAPABILITIES HAVE BEEN ADDED ON EACH OPENFLOW PROTOCOL REVISION. THE NUMBER OF REQUIRED (REQ) AND OPTIONAL (OPT) CAPABILITIES HAS GROWN CONSIDERABLY.

OpenFlow Version	Match fields	Statistics	# Matches		# Instructions		# Actions		# Ports	
			Req	Opt	Req	Opt	Req	Opt	Req	Opt
v 1.0	Ingress Port	Per table statistics	18	2	1	0	2	11	6	2
	Ethernet: src, dst, type, VLAN	Per flow statistics								
	IPv4: src, dst, proto, ToS	Per port statistics								
	TCP/UDP: src port, dst port	Per queue statistics								
v 1.1	Metadata, SCTP, VLAN tagging	Group statistics	23	2	0	0	3	28	5	3
	MPLS: label, traffic class	Action bucket statistics								
v 1.2	OpenFlow Extensible Match (OXM)		14	18	2	3	2	49	5	3
	IPv6: src, dst, flow label, ICMPv6									
v 1.3	PBB, IPv6 Extension Headers	Per-flow meter	14	26	2	4	2	56	5	3
		Per-flow meter band								
v 1.4	—	—	**14**	**27**	**2**	**4**	**2**	**57**	**5**	**3**
		Optical port properties								

SDN 控制器分類

Name	Architecture	Northbound API	Consistency	Faults	License	Prog. language	Version
Beacon [186]	centralized multi-threaded	ad-hoc API	no	no	GPLv2	Java	v1.0
DISCO [185]	distributed	REST	—	yes	—	Java	v1.1
Fleet [199]	distributed	ad-hoc	no	no	—	—	v1.0
Floodlight [189]	centralized multi-threaded	RESTful API	no	no	Apache	Java	v1.1
HP VAN SDN [184]	distributed	RESTful API	weak	yes	—	Java	v1.0
HyperFlow [195]	distributed	—	weak	yes	—	C++	v1.0
Kandoo [228]	hierarchically distributed	—	no	no	—	C, C++, Python	v1.0
Onix [7]	distributed	NVP NBAPI	weak, strong	yes	commercial	Python, C	v1.0
Maestro [188]	centralized multi-threaded	ad-hoc API	no	no	LGPLv2.1	Java	v1.0
Meridian [192]	centralized multi-threaded	extensible API layer	no	no	—	Java	v1.0
MobileFlow [222]	—	SDMN API	—	—	—	—	v1.2
MuL [229]	centralized multi-threaded	multi-level interface	no	no	GPLv2	C	v1.0
NOX [26]	centralized	ad-hoc API	no	no	GPLv3	C++	v1.0
NOX-MT [187]	centralized multi-threaded	ad-hoc API	no	no	GPLv3	C++	v1.0
NVP Controller [112]	distributed	—	—	—	commercial	—	—
OpenContrail [183]	—	REST API	no	no	Apache 2.0	Python, C++, Java	v1.0
OpenDaylight [13]	distributed	REST, RESTCONF	weak	no	EPL v1.0	Java	v1.{0,3}
ONOS [117]	distributed	RESTful API	weak, strong	yes	—	Java	v1.0
PANE [197]	distributed	PANE API	yes	—	—	—	—
POX [230]	centralized	ad-hoc API	no	no	GPLv3	Python	v1.0
ProgrammableFlow [231]	centralized	—	—	—	—	C	v1.3
Rosemary [194]	centralized	ad-hoc	—	—	—	—	v1.0
Ryu NOS [191]	centralized multi-threaded	ad-hoc API	no	no	Apache 2.0	Python	v1.{0,2,3}
SMaRtLight [198]	distributed	RESTful API	no	no	Apache	Java	v1.0
SNAC [232]	centralized	ad-hoc API	no	no	GPL	C++	v1.0
Trema [190]	centralized multi-threaded	ad-hoc API	no	no	GPLv2	C, Ruby	v1.0
Unified Controller [171]	—	REST API	—	—	commercial	—	v1.0
yanc [196]	distributed	file system	—	—	—	—	—

SDN 方案優勢

- SDN為這個本來就複雜的世界帶來了更高的自動化。減少企業因為人為導致的錯誤配置而容易受到攻擊。
- 通過SDN，客戶只需要選擇自己想要在雲中運行的應用和必要的資源。控制平面利用計算、存儲和網絡資源的最佳配置而直觀地部署服務。
- 是否能夠迅速部署並擴展應用，可以成就一個企業!另一方面，也可能毀掉一個企業。如果員工不必手動配置交付應用所需的計算、存儲和網絡資源，那麼企業就能夠更快上線並運行新服務。除了讓員工易於訪問，SDN還可增強公司的競爭優勢，因為它能夠快速響應不斷變化的業務，並縮短新產品進入市場的時間。
- 最後，SDN將大大改變網絡基礎架構的配置和管理方式。通過把控制功能和網絡其它部分分離開來，SDN讓IT團隊能夠以鳥瞰業務的方式來管理網絡環境。也就是說，各個業務不會再孤立地運行。

SDN的效益

- 可開發應用程式使得網路資料流量更靈活，頻寬的使用更充分。
- 等同流量工程，並知道即時流量狀態。
- 依頻寬使用狀況動態更改流量路徑，提高網路使用率。
- 可加入排程作靈活運用。
- 降低維運人力/設備成本。
- 統一控管，易於操作管理。
- 提昇障礙排除速度。
- 集中管理，單一查測。
- 不限設備廠牌，統一運作模式。
- 同一標準，跨越廠牌限制。
- 彈性多變的加值開發空間。
- 可整合未來的FW、IDP(入侵偵測與防禦)、DPI、VM、LB、Schedule..等，提供多樣化服務。

SDN架構將全由軟體發號施令 1

- SDN網路架構就是為了要解決傳統網路的這些問題，SDN的特色是修改了傳統網路架構的控制模式，將網路分為控制層（Control Plane）與資料層（Data Plane），將網路的管理權限交由控制層的控制器（Controller）軟體負責，採用集中控管的方式。

- 控制器軟體就像是人類的大腦，統一下達指令給網路設備，網路設備則專責於封包的傳遞，就像是人類的四肢負責執行各項動作。這樣的概念讓網管人員能更靈活也更彈性地配置網路資源，日後網管人員只需在控制器上下達指令就可以進行自動化的設定，無須逐一登入網路設備進行各別的設定，節省人力成本也降低了人為部署發生疏失的可能性。

SDN架構將全由軟體發號施令 2

- 而OpenFlow技術則是一項通訊協定，用於控制層和資料層間建立傳輸通道，就像是人類的神經一樣，負責大腦與四肢的溝通，OpenFlow協定目前也是實現SDN架構最主流的技術。
- OpenFlow技術將封包傳送的路徑看成是一條「Flow」，就好像是專屬的傳輸路徑，網管人員可依據企業政策或是服務層級協議（Service Level Agreement，SLA）在控制器軟體上設定各項網管功能以及預先建立邏輯網路，來決定封包傳輸方式，例如經過哪些交換器，需要多少的網路頻寬，再將傳輸路徑設定成OpenFlow路由表（Flow Table）。

SDN架構將全由軟體發號施令 3

- 接著在控制層和資料層之間利用SSL加密技術建立起安全的傳輸通道，控制器會將設定好的OpenFlow路由表透過傳輸通道傳送給資料層的網路設備來進行封包派送。因為傳輸路徑已預先設定完成，交換器不需要透過不斷學習來尋找封包傳送的路徑，可大幅提升傳輸效率，降低延遲（Latency）的時間。
- 企業日後僅需透過廠商提供的OpenFlow韌體進行更新，即成為支援OpenFlow技術網路設備，就可以透過支援OpenFlow技術的控制器軟體來管理。也就是說，不論企業採購哪一家廠商支援OpenFlow技術的網路設備，都將交由控制器統一管理，被單一網通廠商綁定的問題就可以迎刃而解了。

SDN的誤解 1

與任何新技術一樣，SDN只要存在，必然就有唱反調的人。對任何企業來說，都希望在部署SDN解決方案之前，先了解那些最大誤解背後的真相。

- SDN並不適合小型數據中心
  - 提到SDN，人們往往認為它只適合大型數據中心——也就是提供公有、私有和混合雲服務的數據中心。儘管這些較大的提供商自然屬於早期採用者，事實上，SDN對於各種級別的數據中心都大有裨益。不僅僅是因為它能讓配置、管理和監測任務變得簡單，它也可以大大降低IT部門的負擔，這是擁有精幹團隊的小公司的完美選擇。
- SDN意味著許多IT崗位將消失
  - 與傳統網絡環境相比，支持SDN的環境需要更少的手動工作就能維持正常運轉。這種說法是正確的，但這並不意味著傳統網絡管理職位將消失。隨著企業過渡到SDN模式，網絡技能不斷演進，這樣對網絡技能的需求也只增不減。事實上，新IP時代所需的技能類型將不斷變化。企業和IT專業人士應意識到這一點，並相應地量身定製自己的培訓和發展計劃。

SDN的誤解 2

- 如果伺服器已經虛擬化，就不需要SDN了
  - 這不是真的。通過用更靈活的虛擬化網絡基礎架構來替換傳統硬體，把伺服器虛擬化的原理延伸到網絡中，這樣做將帶來更多同樣重要的好處。SDN也可發揮更大作用，尤其是它能夠讓網絡延伸到伺服器中，提供更高效的管理並且能夠將伺服器之間的流量可視化。
- 要想實施SDN，整個數據中心網絡都必須替換
  - 「推倒重來」並不是成功實施SDN的必要條件，更科學的方法是從傳統網絡基礎架構逐步遷移到SDN! 其實，實施SDN非常簡單：把SDN設備作為網絡組件的默認選擇，成為現有硬體更新計劃的一部分;或者在新項目或擴張需要添加新設備的時候部署SDN。

誰在發展、使用 SDN?

- Google、Facebook、Yahoo、微軟等多家指標型的大企業投入了SDN架構與OpenFlow技術的發展，這個新世代的網路架構改變了傳統網路控制的模式，對諸多網通業者帶來不小的衝擊。
- Google運用了OpenFlow傳輸協定來打造內部資料中心的SDN 架構，新的網路架構讓Google原本只有30～40%的網路頻寬使用率，一口氣提升了3倍，高達95%的使用率，換句話說，Google就算使用相同的網路設備和線路，可以承載的網路流量也變成了3倍。因此，Google還計畫要將承載使用者流量的骨幹網路，也轉換成這個全新的SDN網路架構。

thank you!