SDN 交換器與傳統交換器的架構與效能比較

廖俊傑 林盈達
國立交通大學資訊工程系
Email: jy333221.cs02g@nctu.edu.tw; ydlin@cs.nctu.edu.tw
September 30, 2013

摘要

在網路的世界中，設備的管理一向是重要的議題。而現今 SDN 網路將控制層從網路設備的資料層分離，集中由外部控制器進行管理，網路設備只進行封包傳遞的動作，大幅簡化繁多設備管理的問題。本篇報告進行傳統交換器與 SDN 交換器的效能測試與比較，在傳輸速率方面，傳統交換器能達到頻寬的 9 成，而 SDN 交換器在 TCP 傳輸只有 25%，UDP 是幾乎無法傳送的狀態，而 SDN 交換器整體延遲時間約為傳統交換器的 13.23 倍。由於 SDN 交換器 CPU 時脈不高，使得封包處理速度慢，以及封包被轉送到控制器，增加多餘的延遲時間，加上封包發送速率過快，又或者封包過大需要分割，導致大量的封包被重送或遺失，這四種因素造成 SDN 交換器效能不及傳統交換器。

關鍵字：SDN、交換器、OpenFlow、架構、效能

1. 簡介

在現今企業的網路中，雲端運算已成為不可或缺的一部分，將分散的伺服器集中管理，使實際上位於不同處的資源看似於放置在同一個地方。如此一來，不但擁有大量的資源外，更可以提升系統容錯的能力，也就是資源虛擬化。研究人員發現虛擬化所帶來的好處後，也想將負責連接與傳送資源的網路給虛擬化，使其網路的效能及管理方面能夠進一步的提升。但在實作後才發現，網路與其它資源不同，就算將設備集中，也只是達到集中化的效果，在管理方面還是得對每一部設備進行各別的設定，並無法進一步達到虛擬化的效果。為了解決網路虛擬化的問題，Nick McKeown 帶領的研究團隊提出將網路的控制層獨立出來集中管理，分散於各處的網路設備就僅是負責資料傳遞的服務，也就是軟體定義網路 (Software-Defined Networking, SDN)[1] 的概念。如此一來，路徑的運算、環境的配置以及設備的維護都可以集中處理，達到網路控制層虛擬化的目的。本篇文章會針對傳統交換器與 SDN 交換器的效能進行五個項目的測試：(1) 傳輸速率、(2) UDP 封包遺失率、(3) OpenFlow 控制封包產生率、(4) 資料封包重新導向率以及 (5) 延遲時間，來了解傳統交換器與 SDN 交換器兩者之間的差異。

2. SDN 介紹

SDN 是將網路設備中的控制層(control plane) 從資料層(data plane, forwarding plane) 分離，由外部控制器(controller) 集中管理控制層，僅剩下資料層的網路設備就只需要負責處理封包傳遞的部分。在控制器中以軟體的形式執行所定義的網路行為，依照軟體所定義的行為不同，底層的網路設備就會具有不同的行為，例如在控制器中定義收到封包後一律進行廣播的動作，這時底層的網路設備就是一個集線器(hub)；又或者是定義收到封包後進行封包標頭(header) 的解析並傳送給指定的埠(port)，此時底層的網路設備就會變成是一個交換器(switch)。透過這種方式，SDN 可以快速且不需更換實體的網路設備，便能夠部署各式各樣不同的網路環境，更重要的是，集中且程式化的管理控制層，能夠省下大量的部署時間與進行維護人力資源，真正達到虛擬化的目的。

2.1 SDN 架構

在 SDN 發展的過程當中，有許多研究人員提出各式各樣實作 SDN 架構的方法，其中就以 OpenFlow[2] 最為有名，以下將以 OpenFlow 為例介紹 SDN 網路的架構，並針對交換器做為網路設備進行討論及比較。

OpenFlow 是由 Nick McKeown 所帶領的研究團隊提出，目的是提供一個開放的通訊協定(protocol) 使得程式可以控制不同交換器上的流程表(flow table)，達到 SDN 的效果。圖 1 為 OpenFlow 的架構圖，主要有三個部分：

- 流程表：在交換器中寫入封包的流向，封包進入交換器後依照流程表所定義的流向來傳送封包。
- 控制器：在流程表中沒定義其流向的封包會進入控制器，由控制器決定封包接下來的流向。
- OpenFlow 通訊協定：透過 SSL 加密通道，讓交換器以及控制器進行溝通。

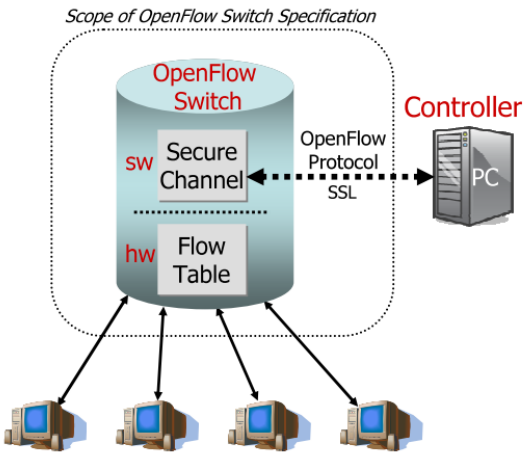

圖 1：OpenFlow 架構圖[2]

2.2 與傳統交換器架構之比較

在現今的網路環境中，交換器本身具有許多的功能，本篇文章中我們稱之為傳統交換器，在此與 SDN 交換器的架構進行比較，如圖 2 所示。圖 2 的左半邊為傳統網路的架構，右半邊為 SDN 網路的架構。從傳統網路架構中，可以看出交換器集所有功能於一身，從最底層的封包傳遞，中層的網路管理，到最上層的應用程式都由一台交換器一手包辦，其優點是可以由硬體的設計來達成交換器的行為，提升交換器的處理速度；但缺點是相同的功能出現在多個交換器上，需要改變交換器的行為時，必須對逐一對每台交換器進行設定，更糟的情況是要變更的行為已經寫入硬體時，必須直接更換整台交換器來完成設定，此時將浪費許多的成本。

而 SDN 網路架構與傳統網路不同，底層只負責封包傳遞，位於傳統網路中層及上層的功能被移轉到控制器上，由控制器中的軟體進行控制，最大的優點是可以快速且不需更換交換器，部署各式各樣不同的網路環境，並省下大量的部署時間與進行維護人力資源，達到虛擬化的效果。

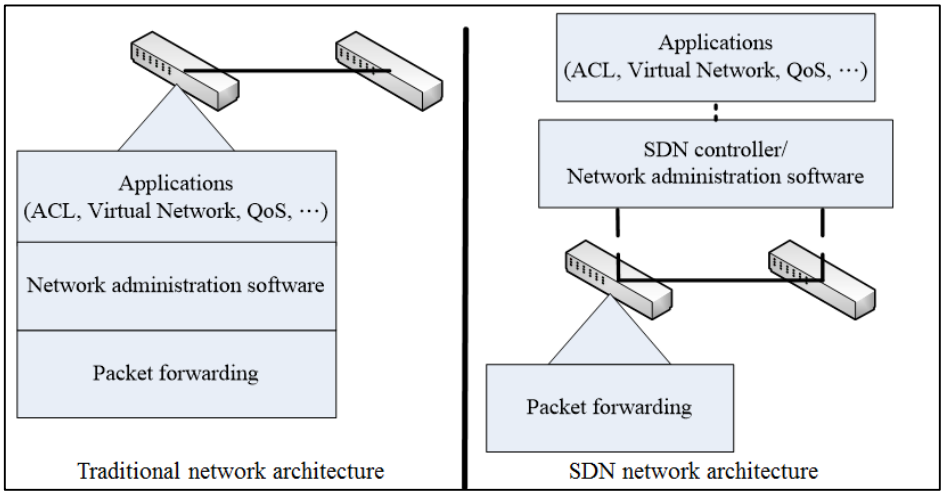

圖 2：傳統與 SDN 架構的功能位置圖[7]

傳統交換器為了應付各種網路的問題，本身必須具有一些功能，其中我們列出自我學習(self-learning)、生成樹(spanning tree protocol)以及虛擬區域網路(VLAN) 這三項較為重要的功能來與 SDN 網路架構來進行比較，如表 1 所示。從表 1 中看出，傳統交換器具有自我學習來決定封包該往哪個埠進行傳送，也具有生成樹來避免網路迴圈，至於虛擬區域網路則需看廠商是否有將此功能加入交換器中。而在 SDN 網路的架構中，SDN 交換器是僅負責比對每個封包的虛擬區域網路標籤並轉送到控制器處理，達到網路切割的效果，其它則由控制器來控制這些功能。當控制器開啟學習交換器(learning switch) 時，SDN 網路就具有自我學習的功能；當控制器開啟生成樹的模組時，SDN 網路就具有生成樹的功能。透過這個比較可以發現，一些網路功能的控制已經逐漸交由上層集中的控制器進行控制，再次看到 SDN 所帶來的好處。

表 1：SDN 與傳統架構功能比較表

function	traditional switch	SDN switch	SDN controller
self-learning	✓		✓
spanning tree protocol	✓		✓
VLAN	optional	✓	

3. OpenFlow 交換器介紹

OpenFlow 交換器是一個可程式化的交換器，主要在交換器上運行 ofprotocol 與 ofdatapath[2] 兩個程式以及記錄一張流程表。ofprotocol 的功能是記錄封包進出交換器的訊息，ofdatapath 的功能是處理進入交換器的封包，流程表的功能則是記錄規則及處理動作。當封包從外部網路進入到交換器時，會先由 ofprotocol 得知該封包是由哪個埠進入交換器中，並記錄在封包內，然後交由 ofdatapath 處理。ofdatapath 又分為三個步驟，如圖 3 所示。

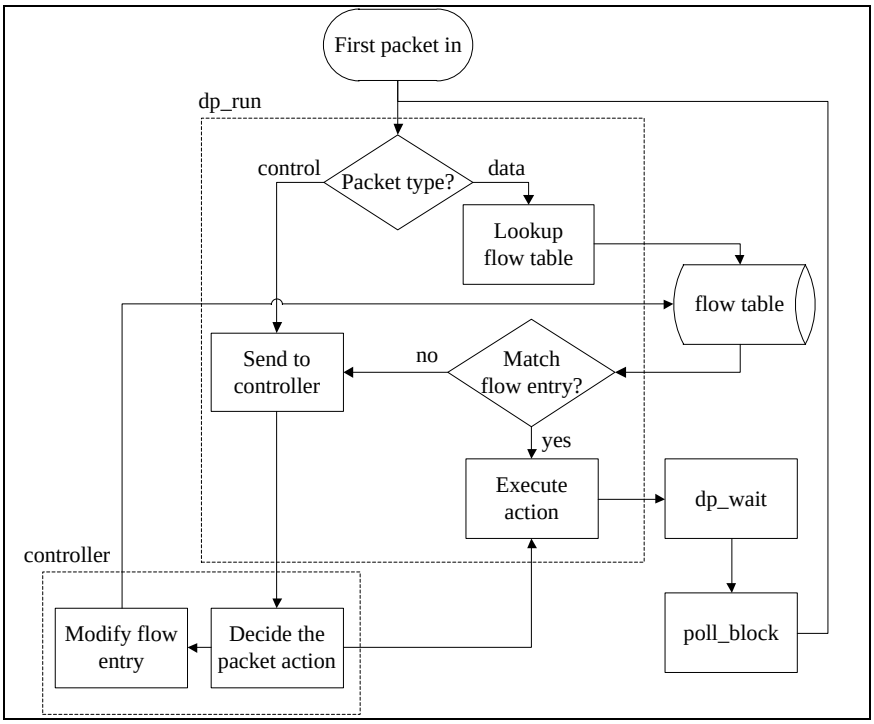

圖 3：ofdatapath 封包處理流程圖

dp_run 是真正處理封包的部分，會先判斷封包的類別，若是屬於控制封包，則直接送到控制器，由控制器處理完後回到交換器執行動作；若是屬於資料封包，則會先比對流程表，有符合規則的就執行，沒有的話就送往控制器決定。封包處理完後會到下個階段 dp_wait，dp_wait 會記錄下個被輪詢的封包，以便下一次的封包處理。最後由 poll_block 暫停整個迴圈，當有事件被觸發或是經過設定的等待時間後才會繼續整個迴圈。

由上述流程可以得知，OpenFlow 交換器主要是傳遞封包，僅能判斷控制與資料封包的類別以及執行流程表上的定義的幾個簡單的動作，其餘的部分會因為流程表上未定義而送往控制器去決定，達到控制層與資料層分離的效果。

4. 實驗架構與方法

本次實驗是利用三台電腦以及一台實體交換器，在 100 Mbps 的網路速度下，進行網路效能的測試，架構如圖 4 所示。其中三台電腦的作業系統都是 GNU Linux/Debian Wheezy，linux_kernel 版本為 3.2.0-4，交換器使用 TP-LINK TL-WR1043ND，在上面搭載 OpenWRT[3] 的系統，並使用 POX[4] 做為 SDN 網路中的控制器。這台交換器是以軟體實作流程表，因此在本實驗當中交換器的 CPU 常會出現高負載的情況。

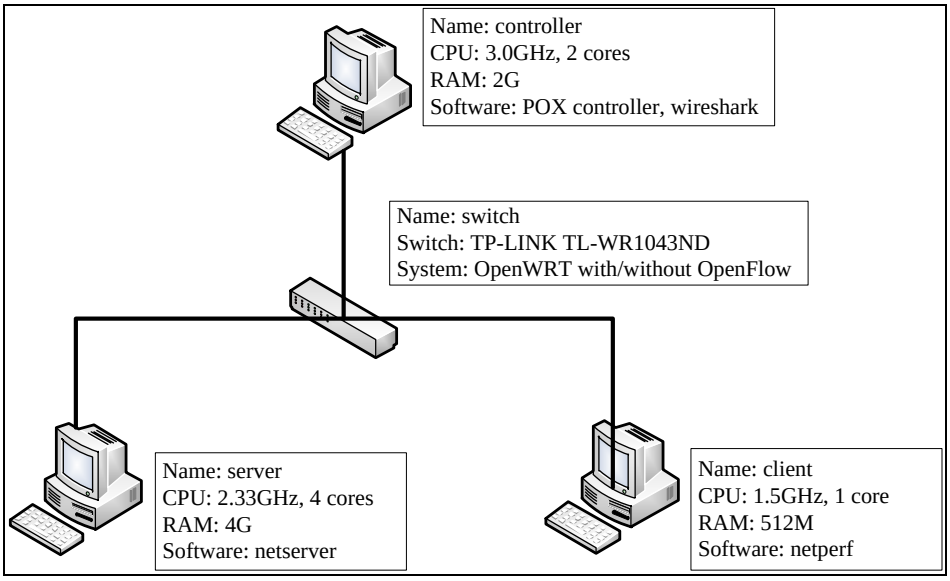

圖 4：實驗環境架構圖

本次實驗將測試五個項目：(1) 傳輸速率、(2) UDP 封包遺失率、(3) OpenFlow 控制封包產生率、(4) 資料封包重新導向率以及(5) 延遲時間。主要的測試是利用 netperf[5] 這個程式進行，首先使用 netperf 預設參數在傳統交換器與 SDN 交換器下分別進行 TCP 與 UDP 大量資料的傳送來測試傳輸速率；接著透過調整封包大小以及傳送速率測試 UDP 封包遺失率；然後再透過 wireshark[6] 分析，分別探討 OpenFlow 控制封包的產生率以及資料封包的重新導向率；最後分析傳統交換器與 SDN 交換器發送封包從 client 端到 server 端之間各段的延遲時間。

5. 實驗結果

5.1 傳輸速率

表 2 是利用 netperf 預設參數，進行傳統交換器與 SDN 交換器的 TCP 與 UDP 傳輸速率的比較。傳統交換器不論是在 TCP 或是 UDP 的傳輸上都有 9 成以上的

速度；反觀 SDN 交換器，受限於交換器 CPU 速度的影響，加上控制器與交換器溝通導致交換器 CPU 一直處於高負載，使得 TCP 傳輸速率只有 25.70Mbps，而 UDP 甚至因為大量的封包遺失，傳輸速率只有 0.10Mbps。

表 2：傳統交換器與 SDN 交換器的 TCP 與 UDP 傳輸速率

device	TCP_STREAM	UDP_STREAM		
	throughput (Mbit/s)	throughput (Mbit/s)		packet loss rate
		send	receive	
traditional	92.04	92.03	92.03	0%
SDN	25.70	96.26	0.10	99.89%

5.2 UDP 封包遺失率

表 3 是不同封包大小以及傳送速率，測試對 UDP 封包遺失率的影響。其中在 1000 bytes 下最多送 9000 個封包不遺失；在 2000 bytes 則最多送 500 個封包不遺失。由此實驗看出，在 5.1 節實驗結果不佳是因為 TCP 封包重送或是 UDP 封包遺失造成傳輸速率低落。

表 3：不同訊息大小以及不同封包發送速率對於封包遺失率的影響

message size (bytes)	packet # (in 5 secs)	packet loss rate	recv. throughput (Mbps)
1000	9000 (max)	0%	14.40
	15500	19.0%	20.09
	25500	49.26%	21.11
2000	500 (max)	0%	1.60
	3500	23.08%	8.60
	5000	47.44%	8.41

5.3 OpenFlow 控制封包產生率

表 4 調整不同訊息大小，由 wireshark 觀察 OpenFlow 控制封包產生率。根據分析結果得知，訊息大小小於 MTU(1480 bytes)時，控制封包產生率低；反之則高。在 netperf 產生的 5000 個資料封包中，訊息大小為 1000 bytes 時會多產生 0.22%的控制封包；而在訊息大小為 2000 bytes 時則會多產生 45.48%的控制封包。

表 4：不同訊息大小下 OpenFlow 控制封包產生率

message size (bytes)	details	packet #	ratio
1000	packet out to switch	7	0.14%
	flow modification	4	0.08%
2000	packet out to switch	33	0.66%
	flow modification	2241	44.82%

5.4 資料封包重新導向率

表 5 調整不同訊息大小，由 wireshark 觀察資料封包重新導向到控制器的比率。分析結果與 5.3 節類似，訊息大小小於 MTU 時，資料被重新導向的比率較低；反之則高。在 netperf 產生的 5000 個資料封包中，訊息大小為 1000 bytes 時會有 0.08%的封包被重新導向；而在訊息大小為 2000 bytes 時則會有 52%的封包被重新導向。

表 5：不同訊息大小的資料封包重新導向率

message size (bytes)	details	packet #	ratio
1000	tunnel generated	2	0.04%
	data packet in to controller	2	0.04%
2000	tunnel generated	2	0.04%
	data packet in to controller	2598	51.96%

5.5 延遲時間

表 6 為 1 秒內送出 500 個封包，傳統交換器的延遲時間與 SDN 交換器是否符合流程表規則的延遲時間，在 SDN 交換器測試的 500 個封包當中，前 4 個是不符合流程表規則的封包，會送往控制器處理，每個封包約需 9.9 ms；其它 496 個封包有符合流程表的規則，會直接執行流程表內的動作將封包傳送到指令的位置，每個封包約需 1.52 ms。由表 6 可以看出，SDN 交換器不符合流程表規則的封包延遲時間約為傳統交換器的 82.5 倍，符合流程表規則的封包延遲時間約為傳統交換器的 12.67 倍。

表 6：傳統交換器與 SDN 交換器的延遲時間

device	time (ms)
traditional	0.12
SDN flow table mismatch	9.9
SDN flow table match	1.52

圖 5 是從 wireshark 數據中分析封包處理的七個階段以及每段的延遲時間。其中封包在不同設備之間傳輸的延遲時間取決於實際的網路環境，本實驗中為傳輸延遲為 0.57 ms，是利用交換器與控制器之間溝通的封包計算出來；控制器的計算時間是利用控制器收到來自交換器內不符合流程表規則的資料封包到產生流量修改封包(flow modification packet) 之間的時間差所推測出來，為 2.23 ms；而流量表的修改時間是從產生出流量修改封包到交換器回覆控制器之間的時間差減去傳輸時間而得到，為 5.01 ms；最後用第一個封包處理完成的時間必須小

於第一個符合流程表規則的封包，藉此推算出查詢流程表的時間為 0.38 ms。

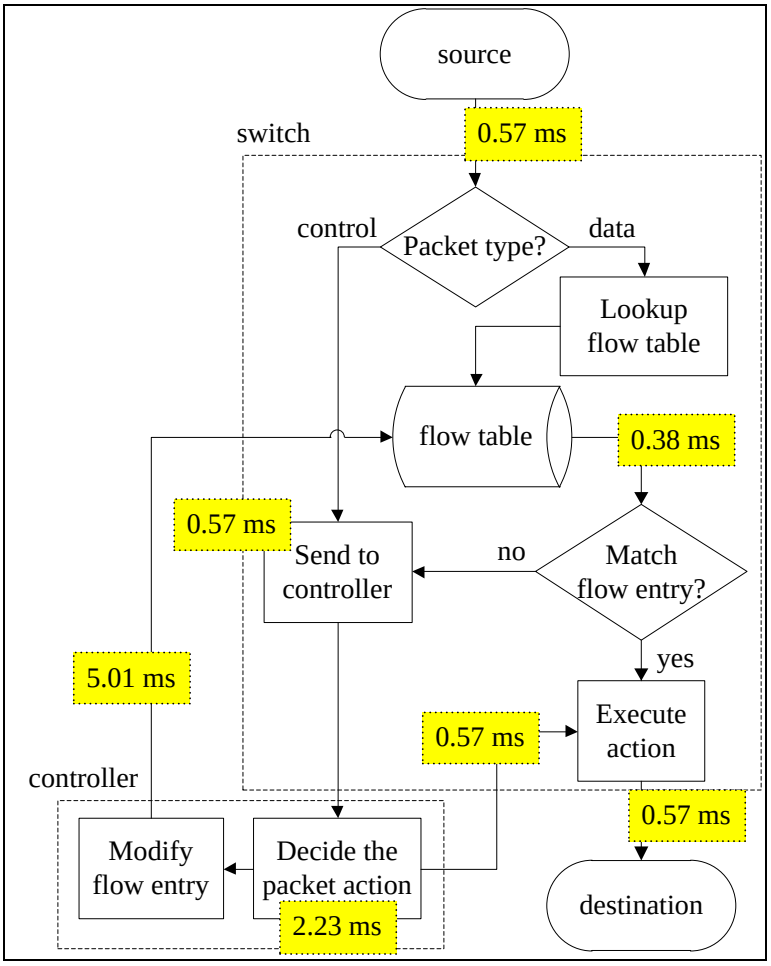

圖 5：SDN 交換器封包傳輸時各段的延遲時間

6. 結論

本次實驗是傳統與 SDN 網路的架構，進行網路效能的測試。在傳輸速率方面，傳統交換器能使用到頻寬的 9 成，而 SDN 交換器在 TCP 傳輸只能用到 25% 的頻寬，UDP 是呈現無法使用的狀態。而在延遲時間方面，SDN 交換器整體延遲時間約為傳統交換器的 13.23 倍。單就 SDN 交換器本身的影響，在相同交換器以及控制器的條件下，由於交換器 CPU 時脈不高，加上這台交換器是以軟體實作流程表，使得封包處理速度慢，又因為封包被轉送到控制器，增加多餘的延遲時間，再加上封包發送速率過快，或者封包過大需要分割，導致大量的封包被重送或遺失。綜合以上幾點，SDN 交換器效能是低於傳統交換器。

參考文獻

[1] K. Greene, "Software-defined networking," Technology review - the 10 emerging technologies of 2009, March 2009.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks,” SIGCOMM Comput. Commun. Rev., 38(2):69–74, 2008.

[3] OpenWrt, `https://openwrt.org/`

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

[5] Netperf, `http://www.netperf.org/netperf/`

[6] Wireshark, `http://www.wireshark.org/`

[7] 張景皓, “新一代網路架構 SDN 顛覆傳統網路的控制模式,” iThome 電腦報 no. 583, pp38, Nov. 22, 2012.