# SDN 兩大控制器之比較: ONOS vs OpenDayLight

王浩丞 林盈達 國立交通大學資訊工程系

Email: <u>qee93.cs01@g2.nctu.edu.tw</u>, <u>ydlin@cs.nctu.edu.tw</u> September 23,2016

# 摘要

SDN(Software-Defined Networking)網路架構是指將控制面(control plane)和資料面(data plane)分離,並將控制面統一集中由外部控制器管理。本篇選擇近年最受矚目的兩大控制器 ONOS 以及 Opendaylight 進行分析並比較優劣,其中針對 ONOS 以及 Opendaylight 所提出較為新穎的分散式叢集(distributed clustering)功能進行進實驗,測試兩者效能上與高可用性上的差異。實驗結果顯示 Opendaylight 在效能表現上優於 ONOS,但在穩定性與高可用性上則是 ONOS 較好。兩者在架構設計上各有其理念且互有優缺點,然而在使用的過程中 ONOS 有著較好的使用者體驗。

關鍵字:SDN、ONOS、Opendaylight、cluster

# 1. 簡介

在本文中,我們首先會介紹 SDN 的基本概念以及 SDN 控制器的基礎架構,接著更進一步比較 ONOS 以及 Openday light 以及他們的 cluster 功能的差異。在第三節我們針對 ONOS 以及 Openday light 相對於其他 SDN 控制器特有的 cluster 功能做性能與高可用性的測試。接著我們將比較 ONOS 與 Openday light 的架構、Northbound API 以及 Southbound API,基於 cluster 功能比較彼此的差異以及性能比較,最後根據實際操作的經驗以及上述比較的結果提出結論。

#### SDN

SDN(Software-Defined Networking)[1]為近年新型的網路型態,其核心概念是把控制面與資料面分離,以簡化網路設備處理封包的流程並將並將控制的部分集中交由外部的控制器去處理。如此一來,控制器將能取得整個網路的拓樸,並且實現網路的可程式化。

## SDN 基礎架構

如圖 1 顯示, SDN 架構[2]大略可以分為應用層、控制層以及基礎架構層三

層,應用和控制層之間透過 Northbound API 交換訊息,控制層和基礎架構層之間透過 Southbound API 交換訊息。其中控制層、Southbound API、Northbound API 構成一個 SDN 控制器。

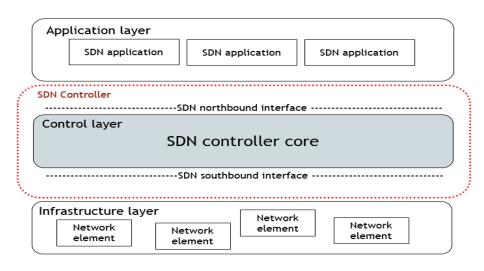


圖 1:SDN 控制器基本架構

應用層主要由許多像是 Forwarding、Firewall 和 Security 等等的 SDN Apps 所組成,透過 Northbound API 取得整個 SDN 網路的相關資訊去做相對應的處理。控制層內由許許多多模組組成一個可以應付各種需求的控制器。基礎架構層則是由各種網路設備所組成,透過 Southbound API 將設備內各種資訊送到控制器內做整合,也接受控制器經由 Southbound API 傳來的控制訊息對 flow table 等設備內的設定作調整。

#### 2. SDN 控制器

SDN 控制器是一支可以執行在伺服器端的程式,是整個 SDN 網路的大腦中樞,所有網路上的封包該怎麼處理以及該送往哪裡全都由 SDN 控制器來決定。自 SDN 提出以來至今已有十數款控制器誕生,有名的像是 Ryu、NOX/POX、Beacon、Opendaylight、ONOS 等等,其中 Opendaylight 與 ONOS 是近年來較新且開源的控制器,因此本篇針對這兩個控制器做分析。

根據表 1 所顯示,ONOS 與 Openday l ight 在和新的設計理念上兩款控制器是十分相似的,兩者都是使用 Java 語言編寫,使控制器能夠在所有能支援 Java 的硬體與作業系統上運作。且兩款控制器都是使用各式各樣的模組來組成控制器的核心,藉由負責不同面向的模組共同合作來維持控制器的運作,並同樣採用 OSGi框架來達成能動態加入或移除控制器內模組的功能。

#### Northbound API

ONOS 與 Openday light 在 Northbound API 都支援使用 REST API 和控制器溝通,不過 ONOS 另外提供了一個叫做 intent framework 的 API 讓管理者與開發者使用。兩者在 REST API 的說明文件編寫讓仍尚未完成,所以目前若要使用 REST

API 需要經過很多的嘗試才能成功對控制器做設定。ONOS intent framework的目的是將對控制器做設定的動作更加抽象化。舉例來說,若是想建立一條從主機A 到主機B 的路線,使用 intent framework 的話就只需要告訴控制器我想建立從主機A 到主機B 的路線,由控制器內部根據現有的網路拓樸計算出最佳的路徑並自動將設定部署到整個網路,不用再由自己對每一個交換機做設定。ONOS 的intent framework內部提供了GUI和 CLI 兩種介面,GUI 介面結合了ONOS 控制器內建的網頁 GUI,在網頁 GUI 檢視全網路拓樸的頁面中就能輕鬆新增簡易的intents,然而由 GUI 介面提供的功能目前還只能新增,並不能修改或刪除intents。另一方面,只要進入 ONOS 控制器的主控台就能新增、編輯或刪除intents,雖然操作上較 GUI 介面複雜,不過和 REST API 比較還是較容易去做設置。

## Southboud API

ONOS 與 Openday light 在 Southbound API 的差異主要是在理念上的不同,ONOS 強調完全的抽象化,而 Openday light 則以能盡量支援現有的各種協定為主。ONOS 為了將控制面以及資料面完整的分離,在 Southbound API 上做了完整的抽象化,讓管理者以及開發者不需為了不同品牌或型號的網路設備對控制器做不同的設定。Openday light 則主張盡量支援現行的各種協定,好讓現有的網路設備能夠直接通用在 SDN 的網路上,因此除了兩者都有支援的協定外,Openday light 還能支援 LISP、BGP、OPFLEX 等較多種協定。然而這樣的缺點是管理者或開發者可能需要注意不同的網路設備可能無法通用同一套設定。

	ONOS	Opendaylight	
Core	OSGi+Java、模組化	OSGi+Java、模組化	
Northbound API	Intent framework, REST API	REST API	
Southbound API	完全抽象化	支援較多協定	
Clustering	控制器成為交換器的 master 或	控制器彼此間形成 master-slave	
	slave		

表 1: ONOS 與 Opendaylight 之比較

# 叢集架構

ONOS 自推出以來就以分散式核心作為主打特色,用許多 ONOS 控制器組成 ONOS cluster,目的是透過增加控制器數量來取得更好的效能表現以及可靠性,已成為電信服務供應商可以選擇的一款電信級 SDN 控制器。ONOS cluster為了提供更簡易的使用環境,將整個 cluster 抽象化為一個邏輯核心,如此一來管理者或 SDN apps 就不用再擔心要處理多個控制器核心可能會有的問題,對其中一個控制器更改設定就能讓其他在同 cluster 內的控制器同步相同的設定。圖 2 為 ONOS 的 cluster 基本組成架構,ONOS 的 cluster 模式要求由三個以上的控制器來組成,以確保 cluster 內資訊同步的正確性。ONOS cluster 運行模式不同於

一般 cluster 內彼此之間是 master-slave 的模式,而是 cluster 內每一個控制器都相對於底下網路設備成為 master 或 slave 控制器,當網路設備與自己的 master 控制器失去連線時就會從多個 slave 控制器中挑選一個做為新的 master 控制器。

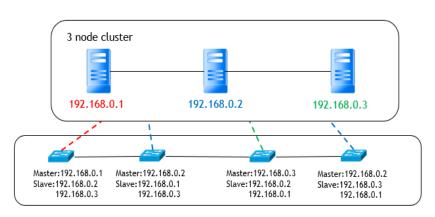


圖 2: ONOS cluster

Opendaylight cluster 自 2015 年第三版 Lithium 推出時被加入進Opendaylight 的功能裡,與ONOS 相同透過 cluster 功能增加控制器的數量來獲得更好的效能表現以及可靠系。然而在 cluster 的架構上和ONOS 有布一樣的架構。如圖 3 所示,Opendaylight 的 cluster 也必須以至少三台控制器組成才能確保資訊同步的正確性,Opendaylight cluster 內部採用 master-slave 模式,cluster 內的控制器會自動挑選一台做為 master,其他控制器則為 slave。底層的網路設備主要和 master 做資訊上的傳遞,以此來獲得全網路的完整拓撲,當master 失去功能時就會自動從 slave 間選出一個成為新的 master,而 slave 在此的功用除了提供確保服務部會中段的高可用性外,slave 也會幫忙分攤 master的計算量以達到負載平衡的功能。

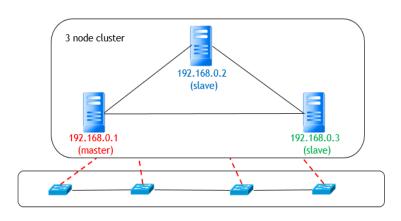


圖 3: Opendaylight cluster

# 3. 叢集模式實驗

ONOS 與 Opendaylight 是目前眾多 SDN 控制器裡少數提供 Clustering 功能的控制器,其功能是為了提供能夠達到電信級的控制器運算能力以及 High

Availability 能力,並且抽象化為單一個邏輯核心以利 SDN apps 或網路管理者對 SDN 網路去做控制。因此我們設計一個實驗以測試 ONOS 以及 Openday light 在 Cluster 模式下彼此的差異。

# 測試環境

圖 4 為我們這次實驗的測試環境,我們建立一個以 3 台控制器組成的 cluster,並使用 mininet[6]來模擬一個由 9 個交換器以及 72 個主機所組成的網路環境。

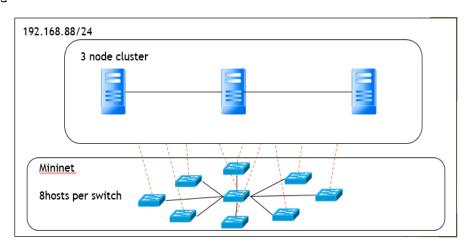


圖 4: test environment

ONOS 控制器採用 Hummingbird 1.7.0 版本,Opendaylight 控制器採用 Beryllium SR2 版本。所有的控制器都只啟用新安裝時最原始的內建模組,另外 再加上 reactive forwarding 和設定 cluster 必須使用的模組。

每一個控制器以及 mininet 都分別安裝在 Vitrualbox 模擬的 2CPU 2G RAM 虛擬機,所有的虛擬機都跑在一台 intel i3-2100 12G RAM 的主機上。

# 測試流程

- a. 對於兩種 SDN 控制器我們採用相同的測試步驟。首先啟動三台控制器,等到 cluster 順利執行後,設定成 drop-all-flows 模式,如此一來所有網路傳送的封包都會透過 Packet In 傳送到控制器做處理,以測試控制器的效能。
- b. 接著啟動 mininet 的執行腳本來模擬網路環境,然後等待控制器成功抓取 到所有的交換機,若是失敗的話則重頭開始。
- c. 使用 mininet 內建的 ping 以及 iperf 工具測試 TCP 和 UDP 的頻寬以及延遲,這個步驟重複十次取平均值,如此一來就獲得控制器全力處理時的效能。
- d. 在 drop-all-flows 模式下,隨機挑選主機 A 和主機 B 互相 ping 對方主機 , 並在 ping 工具執行的過程中關閉負責管理 A 的控制器,以測試高可用性的穩定性。
  - e. 重新啟動被關閉的控制器,觀察控制器加入 cluster 是否順利。
  - f. 關閉 drop-all-flows 模式,並重複步驟 c.,以取得在一般狀況控制器與

網路的效能。

# 實驗結果

所有的測試結果製作成表格比較,表 1 為啟動 drop-all-flows 模式時針對控制器所測試的效能,表 2 為關閉 drop-all-flows 模式時觀察一般狀況整體網路時的工作效能。

UDP Bandwidth TCP Bandwidth Latency

ONOS 2.49Mbits/sec 2.92Mbits/sec 17.98ms

Opendaylight 67.56Mbits/sec 86.39Mbits/sec 0.37ms

表 2:控制層效能

表 3:資料層效能

	UDP Bandwidth	TCP Bandwidth	Latency
ONOS	719Mbits/sec	14.18Gbits/sec	0.38ms
Opendaylight	739Mbits/sec	16.65Gbits/sec	0.06ms

由表 2 可以看出,單從控制器效能方面來看 Opendaylight 明顯比 ONOS 還要好上許多,經過觀察他們所使用的系統資源後我們猜想是因為 ONOS 需要較多的系統資源,然而因為測試環境的不足導致效能有足足 40 倍左右的差距。若是從表 3 來看模擬一般網路使用狀況的結果,除了 ONOS 的延遲略高之外兩者是擁有差距不大的效能。

然而,實際操作時常常遇到 Opendaylight 無法正確捕捉網路拓樸,導致 mininet 裡的主機間無法互通,約重新實驗五次才能正常捕捉到網路拓樸並繼續實驗。且在測試高可用性時,Opendaylight 有時會有無法成功替換 master-slave 的狀況,而 ONOS 則很少發生以上所述的問題。

#### 4. 結論

我們分別以網路管理者以及開發者的角度並根據本篇報告歸納出以下結論: 以網路管理者的角度:

- 1. 依 cluster 的功能測試結果, Openday light 的效能在擁有較低系統資源 時的效能表現較 ONOS 好上約 40 倍。
- 2. 在 cluster 設定以及管理上, ONOS 可以只設定一次控制器就能將設定快速部署到所有其他的控制器上, 而 Openday light 則必須對每一台重複做同樣的設定, 在控制器數量一多的時候 ONOS 會較為方便。
- 3. Opendaylight 在 cluster 模式的穩定性以及高可用性較差, ONOS 能較好的維持網路的連線。
- 4. ONOS 的網頁 GUI 在功能上較 Openday light 多且較方便管理網路。 以開發者的角度:
  - 5. ONOS 的 Northbound API 提供了 intent framework 讓開發 app 更為簡易 且便利,而兩者所提供的 REST API 的使用手冊因還未完全編寫完成所以

- 在使用上較為困難。
- 6. ONOS 與 Opendaylight 在控制器所採用的語言、平台以及架構上十分相近,然而在 Southbound API 上,Opendaylight 以支援更多的協定為目標,而 ONOS 則堅持將 Southbound API 完整的抽象化,讓開發者不需顧慮不同設備間的差異。
- 7. cluster模式的差異上,ONOS是讓每台控制器都負責一部份的網路設備, 而 Opendaylight 則是選出一台控制器做為 master 對外服務所有的網路 設備。

# 参考資料

- [1] Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K., & Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. IEEE Communications Surveys & Tutorials, 16(3), 1617-1634.
- [2] "Understanding the SDN Architecture"
- https://www.sdxcentral.com/sdn/definitions/inside-sdn-architecture/
- [3] Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., ... & Parulkar, G. (2014, August). ONOS: towards an open, distributed SDN OS. In Proceedings of the third workshop on Hot topics in software defined networking (pp. 1-6). ACM.
- [4] Bondkovskii, A., Keeney, J., van der Meer, S., & Weber, S. (2016, April). Qualitative comparison of open-source SDN controllers. In Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP (pp. 889-894). IEEE.
- [5] Medved, J., Varga, R., Tkacik, A., & Gray, K. (2014, June). Opendaylight: Towards a model-driven sdn controller architecture. In Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014.
- [6] "Mininet: An Instant Virtual Network on your Laptop (or other PC)Mininet"

http://mininet.org/