Low-Complexity Distributed SDN Controller Environments on Switch Migration Testbed

Mohd Nazri Bin Mohd Warip¹, Ng Yen Phing¹, Anuar Mat Safar¹, R. Badlishah Ahmad¹, Naimah Yaakob¹, Amiza Amir¹ and Zunaidi Ibrahim²

¹School of Computer and Communication Engineering, Universiti Malaysia Perlis, Pauh Putra Campus, 02600 Arau, Perlis, Malaysia
²Faculty of Technology, University of Sunderland, St Peter's Campus, Sunderland, SR6 0DD, United Kingdom

Abstract. Control Path Management is an important component in the Distributed SDN Controller, and plays a key role in Software Defined Networking (SDN). Testbed simulations are done with an implementation of several controllers are deployed to improve the scalability and reliability of the control plane framework by employed network clustering into several subdomains with separate controllers. This paper investigates the deployment of distributed SDN controller but logically centralized systems for decouples control plane using Distributed Traffic Engineering mechanism. The results obtained show that the proposed mechanism can achieve controller load balancing with better performance.

1. Introduction
The key distinction the traditional network and Software Defined Networking (SDN) is that decouples control plane from data plane [1][5]. It has been noted that to designing Software Defined Network (SDN) [1][5] are rooted in the network architecture. Low complexity distributed SDN controllers designed to improve the scalability, network flexibility management, rapid innovation and agility. In addition, the role of control plane is to enable directly programmable as well as offering centralizes network management. This paper investigates the deployment of distributed SDN controller but logically centralized systems for decoupling control plane using Load-Balanced Distributed Traffic Engineering. Figure 1 provides the typical SDN architecture and taxonomy of topology discovery.
Table 1. Comparison Between SDN and Legacy Network [3]

|                        | SDN                                                                 | Traditional Network                                      |
|------------------------|---------------------------------------------------------------------|---------------------------------------------------------|
| **Features**           | Decoupled control plane from data plane to provide a programmability environment | High complexity of network control and protocol problem |
| **Configuration**      | Automated configuration with centralized validation                 | Manual configuration with error prone                    |
| **Performance**        | Dynamic global control with cross layer information                 | Relative static configuration and limited information    |
| **Innovation**         | Can develop for new ideas by programming, easily to build virtual testbed for test environment | Difficult to implement new ideas on hardware and limited testing environment |

2. Simulation Model
We establish SDN simulation model setting in Mininet [4]. It is capabilities to emulate a large SDN network environment in a single machine. Mininet quickly creating a virtual network that running actual kernel, switch and software application code over a desktop. Custom network topology can run using the MiniEdit, mininet GUI editor.

3. SDN Controller
The SDN Controller Platforms in software-defined network (SDN) perform as the centralize network based on Ryu platform. It is able to work with the common application interface using OpenFlow and open virtual switch database (OVSDB) protocols. A distributed decision mechanism for controller load balancing based on switch migration in SDN is proposed by [6]. Type of SDN controller is shown in Table 2.

Table 2. Type of SDN Controller and Description [5]

| Controller | Description                                                                 |
|------------|-----------------------------------------------------------------------------|
| POX        | Communicating with SDN switches using OPenFlow or OVSDB protocols. Used as basic SDN controller. Complex controller can create through the POX components. |
| Ryu        | Ryu controller was an open source product using SDN technology to increase the agility of the network and supported by NTT. Providing software components with well-defined application program interfaces (APIs). Easing creation of new network management and control applications. |
| Floodlight | Offering the ability for software adaption and development. Present of representational state transfer application program interface (REST APIs). Easing the programming interface and offering aids for developers in building products. Able to deploy in variety working environment. |
| ONOS       | Offered simplified programmatic interfaces that flexible to create and deploy new network services. Ability to create new network applications without altering the data plane programs. Platform extensible and distributed controller. Grant abilities to developers to expend the network dynamically. |
| OpenDaylight | Supporting multiple SDN platforms including OpenFlow, OVSDB, NETCONF and BGP. Enable SDN controller to work with forwarding plane and network changes. |
4. Design of SDN Topology for Load Balancing

The design of SDN topology for load balancing is shown in Figure 3 for low complexity distributed SDN controller environments on switch migration testbed. Environment topology setting encompass of 9 hosts and 21 switch were implemented in Mininet using CLI command as below:

```
$ sudo mn --topo=custom, 4.py --mac --switch=ovsk --controller=remote
```

Each switch been renamed to show different between switch.

![Figure 3. Design of SDN Topology for Load Balancing](image1.png)

5. Results and Discussions

Figure 4 provides the results that generated based on packets captured on topology network shown in Figure 3. The graph contains a lot of packets transfer delay. This might due the topology designed connecting all the switches with the controllers. The transmission took longer time might due to the noise that occur at the controller’s sites.

![Figure 4. IO Graph for h10 ping with s7](image2.png) ![Figure 5. Host Ping Test When Both Controllers Inactive](image3.png)

By using topology developed, inactive controller test being conducted to observed that the host communication. Figure 5 about the test case conducted. The connection failed for every device if both controllers being inactive. Resulting 100% packet dropped. This due to absent of centralized management. To solve the problem at least one controller needed to be activated. Second test for the design is by enable only one port of controller being activate.

![Figure 6. Port for c0 Being Activate](image4.png) ![Figure 7. Host Ping Test After One Controller Activate](image5.png)
In Case 2 regarding only one controller being activated. Since the design (refer Figure 3) connecting all the devices together, it enables the controller manages the network even though only one controller being activated. This enable the devices to send packets without problem with only one controller management. In summary, event one controller fail to operate as long the backup controller present the network still can be managing.

5.1 UDP and TCP Test
The UDP and TCP results is shown in this section.

**Figure 8.** UDP Wireshark IO Graph

**Figure 9.** UDP Graph by Jperf

**Figure 10.** TCP Wireshark IO Graph

**Figure 11.** TCP Throughput Graph

**Figure 12.** TCP Window Scaling Graph

**Figure 13.** TCP Graph by Jperf
6. Conclusions
This study has examined the impact of the SDN controller architecture designed for increasing the network agility and performance. The separating of the network brains (controller) with the body (devices hosted on the network) also ease the network maintenance. These findings have significant implications for the understanding of how the SDN technology actually better to be deploy in bigger network enterprise due to the ability to expand and centralized management. The SDN controller functionalities can be tested over the virtual environment. The main objectives of SDN controller was to enable the fail-over system in Load balancing environment.

7. References
[1] B. Beach and B. Beach, “AWS Architecture Overview,” Pro Powershell Amaz. Web Serv., no. 1, pp. 1–6, 2014.
[2] D. Hasan and M. Othman, “Efficient Topology Discovery in Software Defined Networks: Revisited,” Procedia Comput. Sci., vol. 116, pp. 539–547, 2017.
[3] A. Ahurra, K. M. Kamurali, D. O. Briain, and D. Okello, “Software Defined Networking (SDN),” no. May, 2017.
[4] K. Kaur, J. Singh, and N. S. Ghumman, “Mininet as Software Defined Networking Testing Platform,” Int. Conf. Commun. Comput. Syst., no. December, pp. 3–6, 2014.
[5] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, “SDN controllers: A comparative study,” Proc. 18th Mediterr. Electrotech. Conf. Intell. Effic. Technol. Serv. Citizen, MELECON 2016, no. April, 2016.
[6] T. Hu, P. Yi, J. Zhang, and J. Lan, “A distributed decision mechanism for controller load balancing based on switch migration in SDN,” China Commun., vol. 15, no. 10, pp. 129–142, 2018.