Global Journals LaTeX JournalKaleidoscope™
Artificial Intelligence formulated this projection for compatibility purposes from the original article published at Global Journals. However, this technology is currently in beta. *Therefore, kindly ignore odd layouts, missed formulae, text, tables, or figures.*

Performance Analysis of Different Openflow based Controller Over Software Defined Networking

SM Shamim[1]

[1] Mawlana Bhashani Science and Technology University

Received: 12 December 2017 Accepted: 1 January 2018 Published: 15 January 2018

Abstract

Software Defined Networking (SDN) is a new networking paradigm where control plane is separated from data plane. Over the past several years, SDN has emerged as a compelling paradigm for developing and deploying new network capabilities and services. OpenFlow is the most commonly deployed Software Defined Networking architecture. Multiple networking switches can be controlled by a single centralized controlled OpenFlow controller. Different Python and Java based OpenFlow controller are available for Software Defined Networking. This paper implements Ryu, POX and Pyretic OpenFlow based Python controller in tree networking topology over Software Defined Networking. The result of this paper shows that these Python based OpenFlow controller performs well over SDN. All the implementation of different controller has been done using Mininet Emulator. The result of this paper also shows Pyretic controller has an excellent performance over Software Defined Networking compare to POX and Ryu Controller.

Index terms— software-defined networking; openflow; mininet emulator; openflow controller; network topology.

1 Introduction

oftware-defined Networking [1,2,3] (SDN) has emerged as a new paradigm of networking that enables network operators, vendors, and even third parties to innovate and create new capabilities at a faster pace. This SDN paradigm shows potential for all domains of users. SDN played an important role in increasing the capabilities of traditional networking system [4]. Software-Defined Networking (SDN) has recently gained an unprecedented attention from industry and research communities. SDN provides us a simplified network management by enabling network automation, fostering innovation through programmability. Different types of controller in SDN technology are being used for observing the performance of networking system.

SDN provides some great features that allow the network providers and administrators to act as fast as possible to access, interchange and update any system easily [5]. It consists of decoupling the control and data planes of a network. It relies on the fact that the simplest function of a switch is to forward packets according to a set of rules. However, the rules followed by the switch to forward packets are managed by a software-based controller. One motivation of SDN is to perform network tasks that could not be done without additional software for each of the switching elements [6]. It allows abstracting the underlying infrastructure and program and open flow of data into the network by separating the control plane and the data plane. It has been gaining a great popularity both in the research communication & industry. Most network operators and owners are actively exploring SDN. For example, Google has switched over to Open Flow and SDN for its interdatacenter network [7]. Different types of controller in SDN technology are being used for observing the performance of networking system. This paper analyzes performance of different OpenFlow based controllers in Software Defined Networking.

The structure of this paper is as follows. Section II, the research methodology is described. In Section III, proposed model Test Bed Setup is illustrated. Section IV, evaluates the results after the experiment. Section V; conclude the paper with future work.

2 II.

3 Research Methodology

A literature review is performed to find out details about the routing algorithm over Software Defined Networking. After studying required software tools and hardware equipment are selected for implementing the different controller. Then software tools have been selected for the experiment. After then a preliminary experiment setup is designed which include the hardware setup and software configurations. Various software tools have been performed among OFNet [8], Maxinet [9], EstiNet [10], NS-3 [11], OMNET++ [12] and Mininet [13,14].

4 Test Bed Setup

All the simulation has been done over Software Defined Networking using Mininet Emulator. In order to simulate tree networking topology has been used which shows on Figure **??**1. Mininet creates virtual hosts by using a process-based virtualization method and the network namespace mechanism, which is a feature supported since Linux version 2.2.26, to separate network interfaces, routing tables, and ARP tables of different virtual hosts. Performance Analysis of Different Openflow based Controller Over Software Defined Networking Designed tree networking consists of seven OpenFlow switch and eight hosts where two hosts is connected each of the switch. Host h1, h2 is connected to switch S1 and host h3, h4 is connected to switch S2. In addition, host h4, h5 is connected to switch S3 and host h5, h6 connected with switch S4.

IV.

5 Result and Analysis

Different Python based OpenFlow controller has been implemented separately over Software Defined Networking. Firstly, Ryu controller has been implemented in designed tree network topology. In the designed network topology Ping executed from host h1 to host h5 and host h4 to host h8. Figure **??**

6 C

Round Trip Time (RTT) is the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgment of that signal to be received. This time delay therefore consists of the propagation times between the two points of a signal. Smallest Round Trip Time is always expected for analyzing networks performance. While Ping from host h1 to host h5 corresponding minimum, maximum and average Round Trip Time for each controller has been shown in table-1. From the table-2, OpenFlow POX controller has largest average RTT 0.185ms and largest minimum RTT 0.145ms while Ping from host h4 to host h8. Among the three OpenFlow based controller, Pyretic controller has smallest minimum RTT 0.108ms, maximum RTT 0.179ms and average RTT 0.140ms. From the network performance analysis graph Figure **??** and Figure 5, Pyretic controller has better performance over Software Defined Networking compare Ryu and POX controller.

V.

7 Conclusions

For the Next Generation Networks (NGN) and future internet technologies, Software Defined Networking using OpenFlow protocol will be the most deployed networking architecture. OpenFlow protocols provide standards for routing and delivery of packets on a switch. OpenFlow Controller uses the OpenFlow protocol to connect and configure the network devices in order to determine the best path for application traffic. In this paper, several OpenFlow based controller has been implemented separately over Software Defined Networking. All the evaluation has been done using Mininet Emulator. The result of this paper shows Pyretic controller shows better performance over Software Defined Networking compare to Ryu and POX controller. Future works involves performance analysis of different OpenFlow based controller over Software Defined Wireless Networks (SDWN).

8 Global Journal of Computer Science and Technology

Volume XVIII Issue I Version I [1] [2] [3]

[1] © 2018 Global JournalsPerformance Analysis of Different Openflow based Controller Over Software Defined Networking

[2] © 2018 Global Journals

[3] ( ) C © 2018 Global Journals Performance Analysis of Different Openflow based Controller Over Software Defined Networking

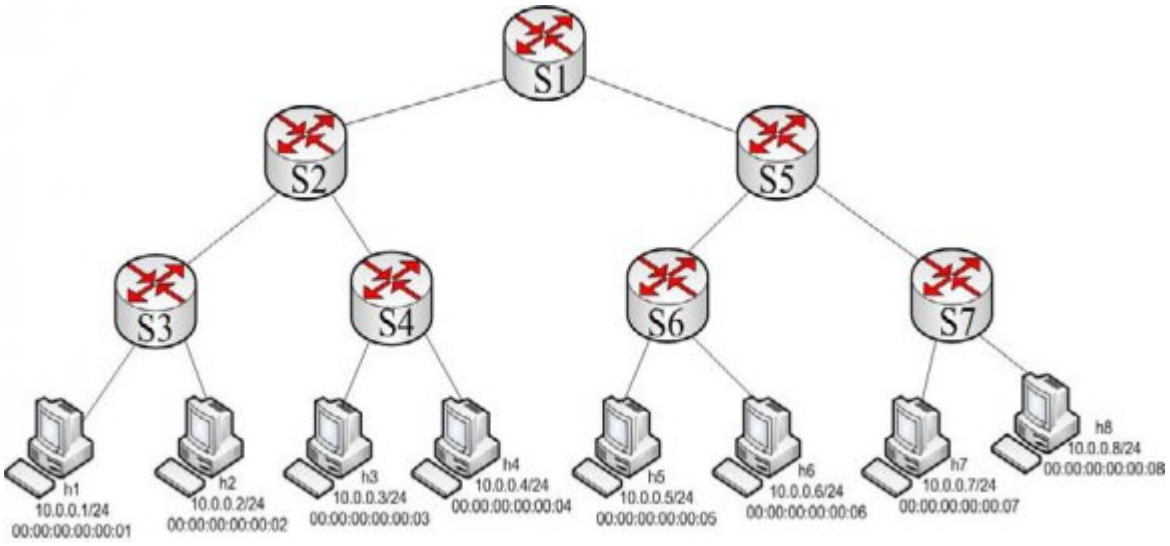


Figure 1:

12

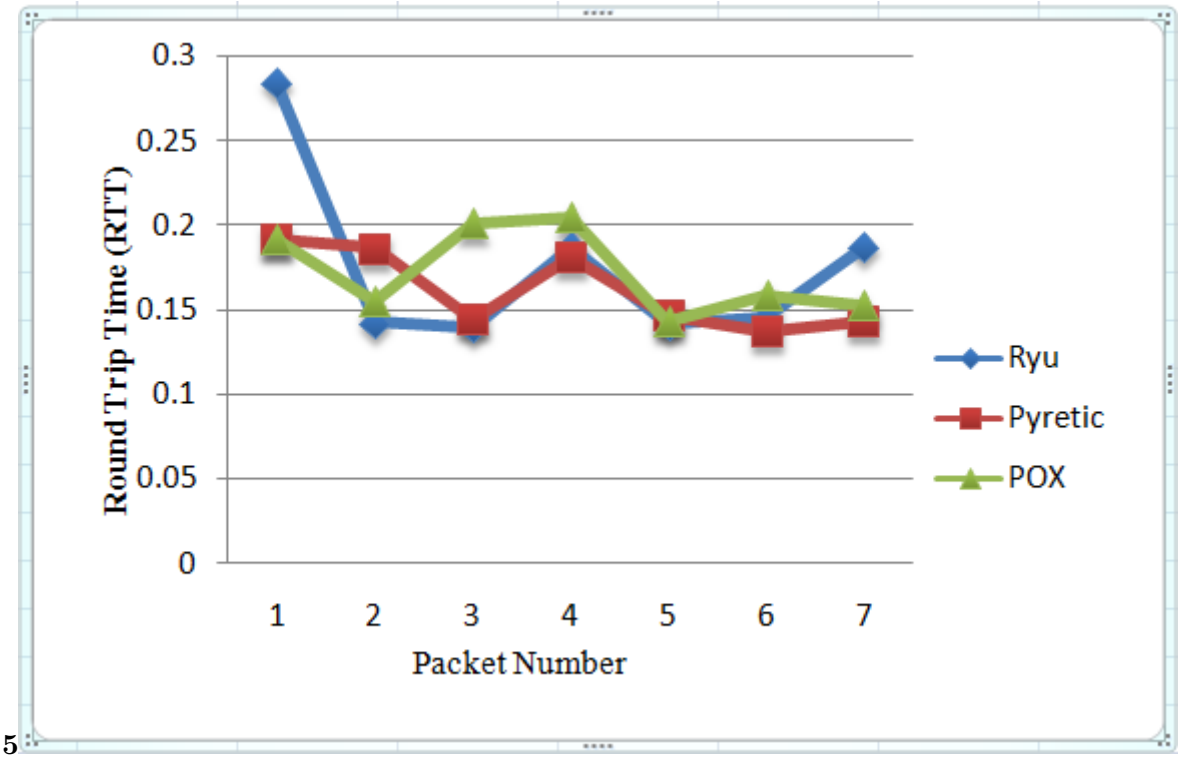
```
"Node: h1"
root@sdnhubvm:~[03:06]$ ping -c7 10.0.0.5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=0.284 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.142 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.139 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=0.187 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=0.141 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=0.145 ms
64 bytes from 10.0.0.5: icmp_seq=7 ttl=64 time=0.187 ms

--- 10.0.0.5 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6013ms
rtt min/avg/max/mdev = 0.139/0.175/0.284/0.048 ms
root@sdnhubvm:~[03:07]$
```

Figure 2: Fig. 1 :Fig. 2 :

34

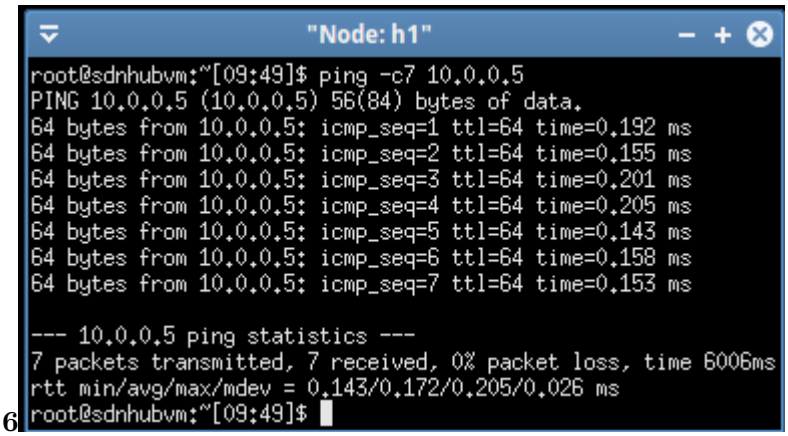
```
"Node: h4"
root@sdnhubvm:~[03:08]$ ping -c7 10.0.0.8
PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data.
64 bytes from 10.0.0.8: icmp_seq=1 ttl=64 time=0.215 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=0.172 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=0.230 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=64 time=0.187 ms
64 bytes from 10.0.0.8: icmp_seq=5 ttl=64 time=0.145 ms
64 bytes from 10.0.0.8: icmp_seq=6 ttl=64 time=0.144 ms
64 bytes from 10.0.0.8: icmp_seq=7 ttl=64 time=0.146 ms

--- 10.0.0.8 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6001ms
rtt min/avg/max/mdev = 0.144/0.177/0.230/0.032 ms
root@sdnhubvm:~[03:08]$
```

Figure 3: Fig. 3 :Fig. 4 :



5

Figure 4: Fig. 5 :



6

Figure 5: Fig. 6 :

1

From table-1, Pyretic controller has smallest minimum RTT 0.137ms, maximum RTT 0.191ms and average RTT 0.161ms. Ryu controller has largest maximum RTT 0.284ms and largest average RTT 0.175ms.

Name of Controller	Minimum RTT (ms)	Maximum RTT (ms)	Average RTT (ms)
Ryu	0.139	0.284	0.175
POX	0.143	0.205	0.172
Pyretic	0.137	0.191	0.161

Figure 6: Table 1

2

Name of Controller	Minimum RTT (ms)	Maximum RTT (ms)	Average RTT (ms)
Ryu	0.144	0.230	0.177
POX	0.145	0.227	0.185
Pyretic	0.108	0.179	0.140

Figure 7: Table 2

[Simulator] , Omnet++ Discrete Event Simulator . https://omnetpp.org/ *Online Available* p. .

[Jarraya et al. ()] 'A survey and a layered taxonomy of software-defined networking'. Y Jarraya , T Madi , M Debbabi . *IEEE Communications Surveys & Tutorials* 2014. 16 (4) p. .

[Keti and Askar ()] 'Emulation of Software Defined Networks Using Mininet in Different Simulation Environments'. F Keti , S Askar . *6th International Conference on in Intelligent Systems, Modeling and Simulation (ISMS)*, 2015. 2015. IEEE. p. .

[Simulator] *EstiNet 9*, | Simulator . Accessed: 2016-09-30. http://www.estinet.com/ns/?page_id=21140

[Going With the Flow: Google's Secret Switch to the Next Wave of Networking] *Going With the Flow: Google's Secret Switch to the Next Wave of Networking*, http://www.wired.com/wiredenterprise/2012/04/going-with-the-flow-googledesseccA p. .

[MaxiNet: Distributed Network Emulation] *MaxiNet: Distributed Network Emulation*, https://maxinet.github.io/ p. .

[Mininet: An Instant Virtual Network on your Laptop (or other PC) -Mininet] *Mininet: An Instant Virtual Network on your Laptop (or other PC) -Mininet*, http://mininet.org/ Online Available. p. .

[OFNet-Quick User Guide Online Available] 'OFNet-Quick User Guide'. http://sdninsights.org/ *Online Available* p. .

[Sonkoly et al. (2012)] 'OpenFlow virtualization framework with advanced capabilities'. B Sonkoly , A Gulyás , F Németh , J Czentye , K Kurucz , B Novák , G Vaszkun . *European Workshop on Software Defined Networking (EWSDN)*, 2012. October. IEEE. p. .

[Mckeown et al. ()] 'OpenFlow: enabling innovation in campus networks'. N Mckeown , T Anderson , H Balakrishnan , G Parulkar , L Peterson , J Rexford , J Turner . *ACM SIGCOMIM Computer Communication Review* 2008. 38 (2) p. .

[Bholebawa et al. ()] 'Performance analysis of proposed openflow-based network architecture using Mininet'. I Z Bholebawa , R K Jha , U D Dalal . *Wireless Personal Communications* 2016. 86 (2) p. .

[Casado et al. ()] 'Rethinking enterprise network control'. M Casado , M J Freedman , J Pettit , J Luo , N Gude , N Mckeown , S Shenker . *IEEE/ACM Transactions on Networking (ToN)* 2009. 17 (4) p. .

[Kreutz et al. ()] 'Software-defined networking: A comprehensive survey'. D Kreutz , F M Ramos , P E Verissimo , C E Rothenberg , S Azodolmolky , S Uhlig . *Proceedings of the IEEE*, (the IEEE) 2015. 103 p. .