Scientific Journal of Informatics
Vol. 3, No. 2, November 2016
p-ISSN 2407-7658 http://journal.unnes.ac.id/nju/index.php/sji e-ISSN 2460-0040

Performance Test of Openflow Agent on Openflow Software-Based Mikrotik RB750 Switch

Rikie Kartadie[1]

[1] Education of Information and Technology, STKIP PGRI Tulungagung
Email: 1rikie.kartadie@stkippgritulungagung.ac.id

Abstract

A network is usually developed by several devices such as router, switch etc. Every device forwards data package manipulation with complicated protocol planted in its hardware. An operator is responsible for running configuration either to manage rules or application applied in the network. Human error may occur when device configuration run manually by operator. Some famous vendors, one of them is MikroTik, has also been implementing this OpenFlow on its operation. It provides the implementation of SDN/OpenFlow architecture with affordable cost. The second phase research result showed that switch OF software-based MikroTik resulted higher latency value than both mininet and switch OF software-based OpenWRT. The average gap value of switch OF software-based MikroTik is 2012 kbps lower than the value of switch OF software-based OpenWRT. The average gap value of throughput bandwidth protocol UDP switch OF software-based MikroTik is 3.6176 kBps lower than switch OF software-based OpenWRT and it is 8.68 kBps lower than mininet. The average gap throughput jitter protokol UDP of switch OF software-based MiktoTik is 0.0103ms lower than switch OF software-based OpenWRT and 0.0093ms lower than mininet.

Keywords: mininet, mikrotik, openflow, Software-defined Network

1. INTRODUCTION

Network is developed by some devices such as router, switches, etc which function is forwarding data package manipulation in complicated protocol planted inside every single part. An operator is responsible for running configuration either to manage rules or application applied in the network. Human error may occur when device configuration run manually by operator. Software-Defined Network (SDN) was created to overcome such problem. SDN/OpenFlow is controller architecture with special application which can be implemented for specific need in a network. It can lead data directly and dynamically based on data type, data traffic and available data track.

OpenFlow was a new protocol designed and implemented by Stanford University in 2008. It can control *data plane* switch, which physically has been separated with *control plane* using controller software in a server. *Control Plane* communicates with *data plane* through OpenFlow protocol. The types of *Software-Defined Networking* (SDN) allow researchers, administrators, and operators to control their network using specific software and to provide switch with *Application Programming Interface* (API) on *forwarding* table from different vendor [1] .

The implementation of SDN/OpenFlow architecture requires high cost while the use of mininet emulator provides good simulation on research scale; however this implementation still requires hardware [2].

Some famous vendors, one of them is MikroTik, has also been implementing this OpenFlow on its operation. It has added *OpenFlow agent* to OS version 6.17 on the OS Router and provides the implementation of SDN/OpenFlow architecture with affordable cost. The implementation of OpenFlow agent on MikroTik OS Router is worth to have performance testing compared to the test of *switch SDN/OpenFlow software-based* OpenWRT [3]which was conducted previously by the writer.

2. METHOD

2.1. Research Method

Chunk Y. EE., (2012) stated that OpenFlow can be implemented on NetFTPGA to be a firewall and monitored using wireshark. The basic difference between this research and others is the devices used. This research uses switch OpenWrt which is directly connected to controller, so if it works, the implementation on network infrastructure would not take too much cost because there would not be significant change on infrastructure hardware used [4].

Applement, M. dan De Boer, M., analyzed the performance of openflow hardware such as NetFPGA card, Pica8 OpenFlow on a Pronto switch and the Open vSwitch. Trial tests were conducted on some variables; QoS, Port Mirroring, failovel speed and performace overhead. The significant difference in this research is the use of hardware type. It uses software-based openflow switch with platform openwrt hardware [1]

Based on researcher's thesis entitled "The Prototype of Software-Defined Network Infrastructure with OpenFLow Protocol and UBuntu as Controller" [2], software-based openflow switch has similar performance with openflow switch hardware-based. The thesis research analyzed the switch performance which was implemented on huge scale infrastructure. The openflow switch software-based was implemented on topology in huge SDN infrastructure.

The research result showed that switch OpenFlow with OpenWRT based has good performance. It showed low (not high) average gap value on every testing while the jitter value showed same result [3].

Tanutama stated that MikroTik is an independent Linux-based operating system installed on router computer. Mikrotik was designed as user friendly OS which is good to manage computer network administration such as designing and developing a computer network system both in simple and complicated scale. Mikrotik was developed in 1995 to serve Internet Service Provider (ISP) companies which provide wireless technology to their clients. Nowadays, MikroTik provides services of wireless ISP for internet access in many countries in the world, including in Indonesia. Mikrotik on Personal Computer (PC) hardware is famous as an Operating System with good control quality, stable and flexible for various data package and routing. Mikrotik as computer based router

gives advantages to ISP companies in running both simple and complicated applications. Mikrotik can also be used to manage access capacity such as bandwidth, firewall, wireless access point (WiFi), backhaul link, hotspot system, Virtual Private Network server, etc., besides used for routing [5].

Software Defined Network Infrastructure

Sofware-Defined Network (SDN) concept was firstly introduced in 2007 by Martin from Stanford University though his article entitled "*Ethane: Taking Control of the Enterprise*" [6]. It was stated that ethane is a new architecture for a company which allows manager to define a network extent and policy then to run it directly. Ethane is an extreme simplification of ethernet switch with centered controller which manages routing the routing entry and flow system.

Software-Defined Networking (SDN) or split architecture is a concept allows network operator to flexibly manage router and switch using software installed on external server [7][8]. Open Network Foundation defines SDN as a new network architecture where network control is separated from *forwarding and it is directly programmed.* [9][10].

OpenFlow

OpenFlow is first standard communication interface which defines SDN controller with forwarding layers on SDN architecture [9]. OpenFlow is a simple concept which centralizes network complication on controller software that allows an administrator to easily manage it by only controlling the software. McKeown, N. [7]introduced this OpenFlow concept with idea to make network manageable/controllable.

OpenFlow was described by Mateo M.P. [11]as en Ethernet protocol. It also has field which is described on figure 1 as follows:

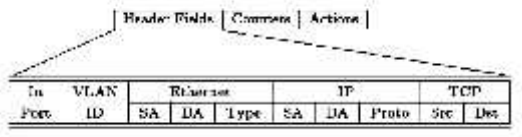

Figure 1. Open Flow Table fields [11]

OpenFlow Switch

There are two kinds of OpenFlow Switch type, the first is *hardware-base switch*, has been sold commercially by some vendors. This switch type modified its *hardware* with TCAM. Ternary CAM allows to match three "Xes" atau "do not care" for one or more bit in saved dataword, so it gives more flexibility in searching process. For example, ternary CAM may have save data word from "10XX0" which will be matched with four key searching word "10000", "10010", "10100", or "10110". The flexibility of searching needs bigger source than biner CAM so it will need additional internal memory to have coding of three possibilities (not two)

of biner CAM. It can be implemented by adding mask (bit "care" or "do not care" to every single memory cell [12], and needs special OS to implement *Flow-Table* and *OpenFlow* protocol. The second type of switch is *software-base switch* with UNIX/Linux system to implement whole functions of *OpenFlow* switch [11].

OpenvSwitch

OpenvSwitch is an application that presents a virtual multilayer switch under Apache 2.0 license. OpenvSwitch was designed to support network architecture change using automatic network technology that works on additional program [13]

Controller

McKeon. N, et.al, stated that an openflow controller is responsible to add or to remove the content of openflow table flow in its device [7]. There are 2 types of controller:

1) Static, a static controller can be a device that can statically add or remove flow from *flow table*s.
2) Dynamic, a dynamic controller dynamically manipulates flow content so it can support and work on some configurations.

Vishnoi, A. and A. Kumbhare, 2013, stated that a controller responsibilities are as follow [14]:

1) Providing mechanism of connection and interaction with u*nderlying platform* (in this case, applied on openflow switch)
2) Interpreting message delivered by openflow switch.
3) Providing all instructions on every single specification (both major and additional specification) to be programmed into the switch.
4) Delivering mechanism to switch for collecting both general and specific/detailed information to have correct respond interpretation.

Latency

Latency is time delay or time interval between simulation and respond. It also can be said that latency is time delay between cause and effect from some physically changes in analyzed system. It means that in this case, latency of the switch is the amount of respond provided by switch per second.

Throughput

Throughput in network can be defined as achievement level of success in delivering message through communication channel. I can also be defined as amount of data which has been successfully delivered to whole terminal in network.

J. Padhye V. Firoiu D. Towsley and J. Kurose in Platonov A.P stated that TCP throughput can be counted using below formula (1).

$$B(av) = \frac{W}{D} \quad (1)$$

This formula can be used to calculate bandwidth size which is available between two connected network point, where W is the transmitted data package size and D is delay of the package. The value of data package is influenced by the value of *latency* [15] .

There are two main steps applied in this research: (1) simulation using mininet emulator as comparing test (considered as representative of dedicated switch openflow), and (2) performance testing of MikroTik RB750-based Openflow switch, and compare the result with previous study using OpenWRT-based Openflow switch. [3]. Topology was designed before before conducting the simulation on mininet emulator.

2.2. Topology Design

Topology used in this research is linear topology that involves 2 *software-based* switches with RB750 MikroTik base which are connected to a controller, as described in figure 2. It is a topology used in mininet simulation and will also be used as topology on prototype.

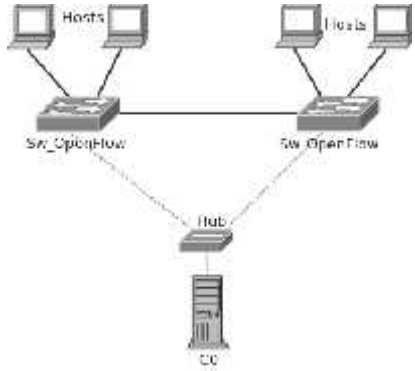

Figure 2. Test Tipology

C0 is controller 0 (zero) which is centralized controller that controlling 2 switches.

Hosts is host which is directly connected to switch

Sw_OpenFlow is tested OpenFlow switches, the switches are connected each other and also directly connected to controlled by a hub.

interdevice connection on OpenFlow network

connection on controller network (connection from controller to switch)

Testing

The designed topology is tested on mininet emulator. It should be done to have comparing data of latency value and throughput with data on tested software-based switch OF.

The topology which will be tested was showed on the below script and saved as ujibanding.py file.

```
from mininet.topo import Topo

class MyTopo( Topo ):

    "Topologi prototipe."
   def __init__( self ):

        "Membuat topologi untuk uji pembanding."
       # Initialize topology
        Topo.__init__( self )

       # Add hosts and switches
        left0Host = self.addHost( 'h1', ip='192.168.10.1/24' )
        right0Host = self.addHost( 'h2', ip='192.168.10.2/24' )
        left1Host = self.addHost( 'h3', ip='192.168.10.3/24' )
        right1Host = self.addHost( 'h4', ip='192.168.10.4/24' )
        leftSwitch = self.addSwitch( 's1' )
        rightSwitch = self.addSwitch( 's2' )

       # Add links
        self.addLink( left0Host, leftSwitch )
        self.addLink( left1Host, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, right0Host )
        self.addLink( rightSwitch, right1Host )

topos = { 'topoujibanding': ( lambda: MyTopo() ) }
```

The script is run on mininet emulator with controller put outside virtual box commanded by mininet. Script is run with below command:

```
mininet@mininet#sudo mn --custom ujibanding.py --topo
topoujibanding --controller=remote, ip=192.168.1.76, port=6633 --mac
--link tc,bw=100
```

After simulating the mininet with available variable, the next step should be done is creating network prototype using *OpenWrt switch*. The switch used in this step is TP-Link WR1043nd with *hardware* 1.11 version. *Firmware used in this* switch belongs to pantou. (the tutorial can be retrieved at http://archive.openflow.org/wk/index.php/Pantou_:_OpenFlow_1.0_for_OpenWRT)

The next step is creating OpenFlow switch *software-base prototype using* MikroTik RB750. There are 2 steps in adding Openflow Agent into MikroTik Rb750 routerboard, First step is to change the OS with OpenWRT and do same steps conducted on previous steps using TPLink WR1043nd switch, Second step is adding package provided by MikroTik (MikroTik itself does not give any license

for public to use this package)In this research, the second step was used to add OpenFlow agent into MikroTik RB750 OS router since the researcher considered that there would not be any significant difference with his previous study if he applied the first step/way.

3. RESEARCH RESULT AND DISCUSSION

Based on obtained data, both data from comparing test of OpenWRT and MikroTik software-based, the researcher then analyzed the data obtained from latency and throughput value.

Latency test value

Data of average latency value are showed on the table 1 and figure 3.

Table 1. Average Latency Value on Each Test

	64byte	**128 byte**	**256 byte**	**512 byte**	**1024 Byte**	**2048 Byte**	**4096 byte**	**8192 Byte**
mininet	2.288	3.999	4.855	5.191	4.341	4.726	4.477	9.149
openWRT	1.622	4.545	4.613	3.585	4.499	6.480	5.890	10.421
MikroTik	3.006	5.155	5.380	5.435	6.017	6.692	7.086	11.226

It is showed that switch OF software-based MikroTik resulted higher latency value than both mininet and switch OF software-based OpenWRT. The latency value of switch openflow software-based MikroTik are more stable than two others, which can be seen in figure 3:

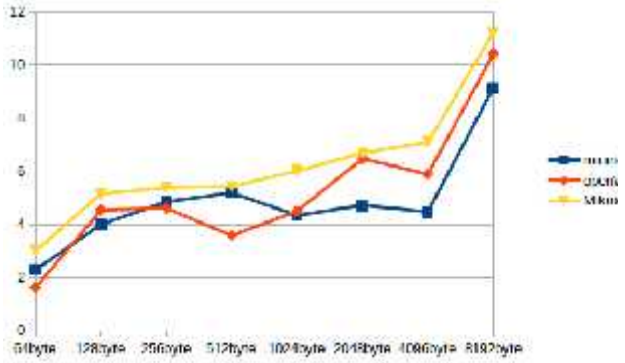

Figure 3. Latency Value Chart

2) TCP Throughput Testing

The TCP throughput testing was obtained from bandwidth size from mininet, switch non OF and switch OF software-based, as showed on table 2 and figure 4. The data used is Rx data (*Receive*).

Table 2. TCP Trhougput Rx Value

	128 kbps	**256 kbps**	**512 kbps**	**1024 kbps**
mininet	93216	93400	93392	96771
OpenWR	95012	92131	92001	92661
MikroTik	90119	89989	90649	93000
Average Gap				
Mikrotik vs mininet			**Mikrotik vs OpenWRT**	
-3255.5			-2012	

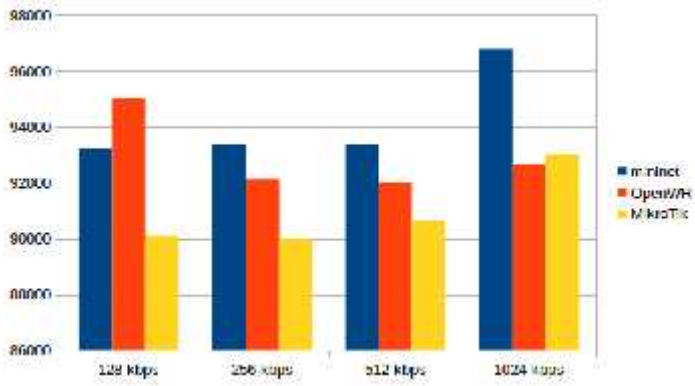

Figure 4. Grafik TCP Throughput Rx

The chart in figure 4 shows that TCP protocol throughput of switch OF software-based MikroTik value are still lower than the value of other comparing switches. It does not mean that the switch OF software-based MikroTik does not have good throughput value on TCP protocol. The average gap value of switch OF software-based MikroTik is 2012 kbps lower than the value of switch OF software-based OpenWRT. It means that the performance of switch OF software-based MikroTik is close to the erformance of switch OF software-based OpenWRT.

3) UDP throughput value test

UDP Protocol throughput value test results 2 types of value; bandwidth size and jitter value. Table 3 shows the difference between bandwidth which is resulted by UDP protocol switch. Figure 5 shows that the UDP protocol throughput value of

switch OF software-based MikroTik is lower than the value of switch OF software-based OpenWRT, and this value is still lower than comparing test (mininet).

Table 3. UDP Throughtput Bandwitdth Size

	128 kbps	**256 kbps**	**512 kbps**	**1024 kbps**
mininet	15.6	31.3	62.5	125
OpenWR	15.1	28.3	56.9	114.01
MikroTik	12.44	26.65	58.1	102.65
Gap Rata-rata				
Mikrotik vs mininet		**Mikrotik vs OpenWRT**		
-8.68		-3.6175		

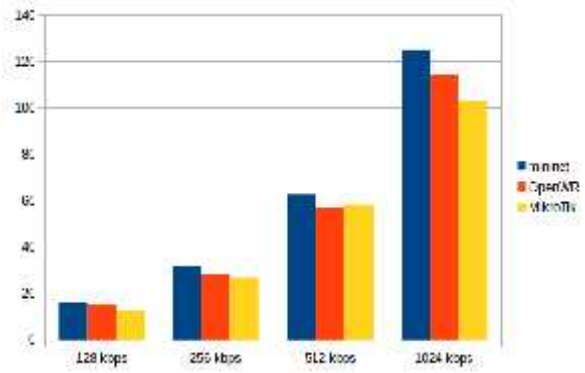

Figure 5. The Result of UDP Throughput

The average gap value of throughput bandwidth protocol UDP switch OF software-based MikroTik is 3.6176 kBps lower than switch OF software-based OpenWRT and it is 8.68 kBps lower than mininet. This value shows that switch OF software-based MikroTik is able to deliver UDP data closely to switch OF software-based OpenWRT.

The jitter value of three switches can be seen on table 4 and figure 6:

Table 4. Jitter Value

	128 kbps	**256 kbps**	**512 kbps**	**1024 kbps**
mininet	0.027	0.026	0.021	0.027

OpenWR	0.025	0.023	0.018	0.031
MikroTik	0.034	0.032	0.034	0.0382
Average Gap				
Mikrotik vs mininet		**Mikrotik vs OpenWRT**		
0.0093		0.0103		

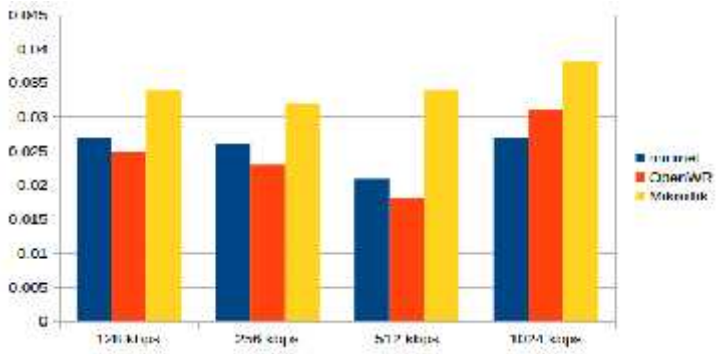

Figure 6. Jitter Value

The Jitter value shows that switch OF software-based MiroTik provides god value although it is higher than the two comparing switches. The average gap throughput jitter protokol UDP of switch OF software based MiktoTik is 0.0103ms lower than switch OF software-based OpenWRT and 0.0093ms lower than mininet. The jitter value on throughput protokol UDP test results same value as the comparing test value.

4. CONCLUSION

Based on data test, it can be concluded that Performance of switch OF software-based MikroTik is good indicated by the low average gap value, even the jitter test value showed same level. It is showed that switch OF software-based MikroTik resulted higher latency value than both mininet and switch OF software-based OpenWRT. The latency value of switch openflow software-based MikroTik are more stable than two others. The average gap value of switch OF software-based MikroTik is 2012 kbps lower than the value of switch OF software-based OpenWRT. It means that the performance of switch OF software-based MikroTik is close to the performance of switch OF software-based OpenWRT. The average gap value of throughput bandwidth protocol UDP switch OF software-based MikroTik is 3.6176 kBps lower than switch OF software-based OpenWRT and it is 8.68 kBps lower than mininet. This value shows that switch OF software-based MikroTik is able to deliver UDP data closely to switch OF software-based

OpenWRT. The Jitter value shows that switch OF software-based MiroTik provides god value although it is higher than the two comparing switches. The average gap throughput jitter protokol UDP of switch OF software-based MiktoTik is 0.0103ms lower than switch OF software-based OpenWRT and 0.0093ms lower than mininet. Based on test results, switch OF software-based MikroTik can be used to replace dedicated openflow switch in implementing SDN both in middle scale (company) and simpler scale (college) although it still needs better improvement to have optimal result.

5. Refference

[1] Appelman, M. and M. De Boer, "Performance Analysis of OpenFlow Hardware," p. 28, 2012.

[2] Kartadie, R., "Software-Defined Network Infrastructure Prototype With OpenFlow Protocol Using Ubuntu As Controller," STMIK AMIKOM Yogyakarta, 2014.

[3] Kartadie, R. and B. Satya, "Uji Performa Implementasi Software-Based OpenFlow Switch Berbasis OpenWRT Pada Infrastruktur Software-Defined Network," *DASI*, vol. 16, no. 3, p. 87, 2015.

[4] Yik, E. C., "Implementation of an Open Flow Switch on Netfpga," Universiti Teknologi Malaysia, 2012.

[5] Tanutama, L., *Jaringan Komputer*, 1st ed. Yogyakarta: Elex Media Komputindo, 1996.

[6] Casado, M. et al., "Ethane: taking control of the enterprise," *Sigcomm '07*, pp. 1–12, 2007.

[7] Mckeown, N. et al., "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[8] Shirazipour, M. et al., "Realizing packet-optical integration with SDN and OpenFlow 1.1 extensions," *2012 IEEE Int. Conf. Commun.*, pp. 6633–6637, 2012.

[9] Noname, "Software-Defined Networking: The New Norm for Networks [white paper]," *ONF White Pap.*, pp. 1–12, 2012.

[10] Paper, W., "Infrastructure SDN with Cariden Technologies," pp. 1–14, 2012.

[11] Mateo, M. P., "OpenFlow Switching Performance," Politecnico Di Torino, 2009.

[12] Hucaby, D., *CCNP BSCI Official Exam Certification Guide*, 1st ed., vol. 205, no. 4594. Indiana: Cisco Press, 2007.

[13] Hariyani, Y. S. et al., "ROUTING IMPLEMENTATION BASED-ON SOFTWARE DEFINED NETWORK USING RYU CONTROLLER AND OPENVSWITCH," *J. Teknol.*, vol. 8, no. 2015, pp. 89–93, 2015.

[14] Vishnoi, A. and A. Kumbhare, "Open Flow 1.3.1 Support: Controller View," 2013. [Online]. Available: https://wiki.opendaylight.org/images/d/dc/Openflow1.3_Support_for_Opendaylight.pdf.

[15] Platonov, A. et al., "Estimation of available bandwidth and measurement infrastructure for Russian segment of Internet," *Arxiv Prepr. arXiv0803.1723*, pp. 1–8, 2008