EAI EndorsedTransactions
on Scalable Information Systems
Research Article EAI.EU

Software-Defined Network Testbed Using ZodiacFX a Hardware Switch for OpenFlow

Fahad Nazir[1]*, Qazi Humayun[2], R Badlishah Ahmad[3], Shamsul Jamel Elias[4]

[1,2]University Malaysia Perlis (UniMAP), Perlis Malaysia, [1]E-mail: fahadnazirbhatti@gmail.com
[3] Universiti Sultan Zainal Abidin (UniSZA), Malaysia
[4] Universiti Teknologi MARA (UiTM) Kedah, Malaysia

Abstract

Software-defined network (SDN) is a new programmable networking designed to perform tasks easier by enabling network administrators to add network control via a centralized control platform and open interfaces. Common network use procedures such as traffic shifts, troubleshooting and various types of device configuration. Thus devices are needed to reconfigure with the network, in order to create a reaction with events. In this paper, we have developed an SDN testbed using Zodiac FX a hardware switch for OpenFlow experiment. This research is utilized Zodiac FX a hardware switch to test the usage and discuss about the SDN controllers. Ryu controller is configured for testbed development. The main contribution of this paper in threefold as follows: first it provides the configuration of Ryu controller, second, it provides the configuration of Zodiac FX switch and lastly it develops a simple SDN testbed for OpenFlow.

Keywords: Software-Defined Network (SDN), Ryu Controller, OpenFlow.

Received on 16 August 2017, accepted on 13 September 2017,published on 25 September 2017

doi: 10.4108/eai.25-9-2017.153150

1. Introduction

The recent advent of Software-Defined Network (SDN) has boosted the network infrastructure in different emerging fields. The SDN is a promising solution for future networks due to its open-nature network architecture. SDN is not an innovative offer thus SDN is a redesign of previous proposals through an examination of earlier models. A brief explanation can be found [in 1, 2] which primarily deals with programmable networks and control data plane division tasks.

The main objective of SDN is to decouple the control plane from the data plane of the OpenFlow switch. The aim of this is to provide centralized control of the entire network. The architecture of SDN is divided into three main layers including data layer, control layer, and application layer. The communication between data layer and control layer is performed by an OpenFlow protocol.

*Corresponding author. Email:fahadnazirbhatti@gmail.com

The control layer contains a main entity known as a controller that can be used to act as the heart for the entire SDN infrastructure. The controller is responsible for controlling the network devices in the data layer while injecting different applications from the application layer. The controller can communicate with another controller connected through east and west bound interfaces. However, the OpenFlow protocol uses southbound interface that is further used to manage communication between switches and the controller. OpenFlow is used by different researchers to perform their experiments as SDN is in its initial stage of implementation.

The SDN idea is based on separating control from the network forwarding features such as switches and routers [2]. The basic principle of the SDN is allied with OpenFlow protocol. OpenFlow is a protocol which has been designed by academia at Stanford University in 2008 [3]. OpenFlow has proposed at the initial stage in [4], which it is executable available for researchers to perform more experiments [5]. Open Networking Foundation (ONF), an open source

OpenFlow architecture has two building-blocks: First OpenFlow Controller and multiple OpenFlow Switches.

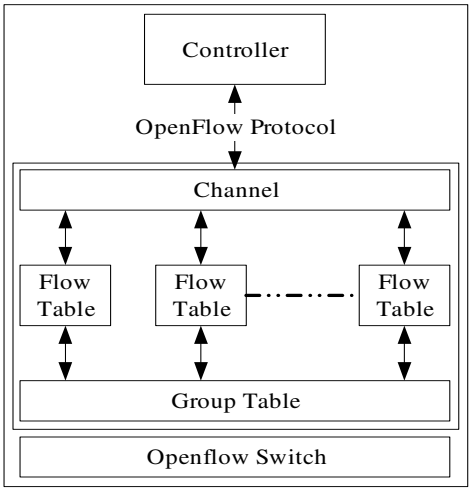

Figure 2. Overview of OpenFlow Design

It is enabled between the controller and switches that maintain a table on every switch. The table contains the "match and action" entry that is essential to examine at the traffic flow. Furthermore, in the case of failure of any matching entry, the controller can be determined by the action upon that failure match entry to make a suitable decision. OpenFlow packet processing deals with various packet processing towards the Ethernet or IP. While it is implemented in its design, therefore OpenFlow packet processing can be done by hardware design.

Figure 3 shows the whole process of packet forwarding in OpenFlow. In general, the process of reaching packets tested by compared to a flow table. In the broad view, the table is contained "match and action" sequence having n terms. The unique match options, such as: *in_port, dl_src, dl_dst, dl_vlan, dl_type, nw_proto, nw_src, nw_dst, tp_src, tp_dst. dl_*are referred to the data link layer, *nw_* is referred to the network layer, *tp_* is referred to the transport layer.

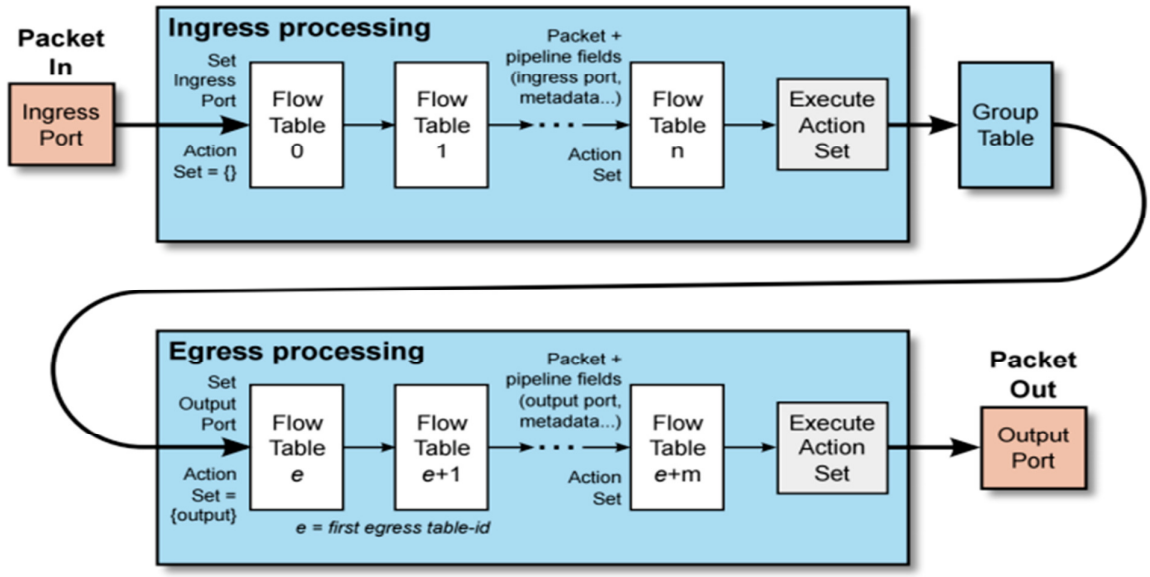

Figure 3. OpenFlow Packet Processing

3. Ryu SDN Framework

Ryu SDN framework is available as an open source with its documentation, it is a tool-set for SDN development, includes an OpenFlow controller, among other tools. This is a component-based framework for SDN. It offers software components with well-defined API that makes it easy to create network management and control applications.

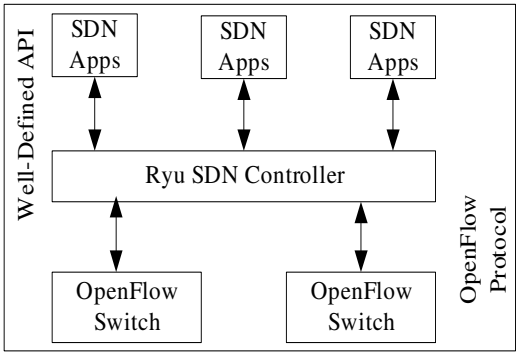

Figure 4. Ryu SDN Controller

Figure 4: shows the overview of Ryu controller. Ryu supports several protocols for dealing with the network devices, such as OpenFlow, Netconf, OF-config, etc. Ryu is a fully developed in Python and codes are available under the Apache 2.0 license [19, 20]. To configure Ryu controller is available via command line is following.

#sudo apt-get install python
#python -v (check python version)
#sudo apt-get install python-pip
#sudo pip install ryu (including open virtual switch)

Ryu has provided well-organized documentation and defined API for creating the different SDN scenario applications. Figure 5 shows the Ryu controller connection with Zodiac FX for SDN scenario. It is fully supported all types of networks applications for example firewall, Intrusion Detection Systems, Intrusion Privation Systems, fault tolerance and load balancing can be designed under Ryu controller. The main advantage of this controller is to support all OpenFlow versions. At the initial stage, ryu-manager appname.py command is executed the Ryu controller.

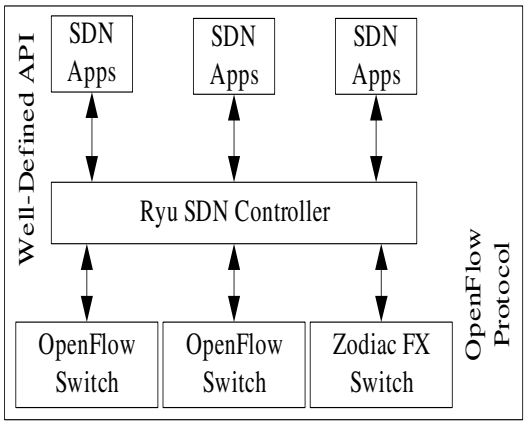

Figure 5. Ryu controller connection with Zodiac FX

4. SDN Testbed Setup

Zodiac FX is designed four ports network implement board that is developed for researchers and developers for modeling and design an SDN services and applications. Figure 6 shows the Zodiac FX switch. It is considered the smallest switch: flexible enough to fit on desktop users. The size of the switch is 10 x 8 cm.

It is a low-cost board that can be performed various experiments. At the first step, it was developed to let inexpensive execution to OpenFlow hardware [21].

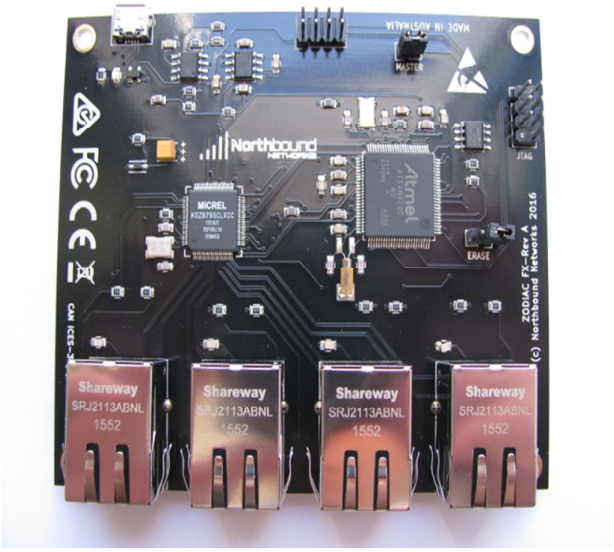

Figure 6. Zodiac FX a hardware Switch

It has provided firmware code that makes user flexible to generate various own versions. It provides various type of device such as the applications may have Router, Bridge, Load Balancer, Web server, VPN concentrator, TOR client. Zodiac FX command line is available via the USB port. To configure Zodiac FX using gtkterm and process of configuration is described as following, Figure 7 shows the port for Zodiac Fx.

#sudo apt-get install gtkterm (Port configrator)
*#ls /dev/tty**
#Sudo gtkterm

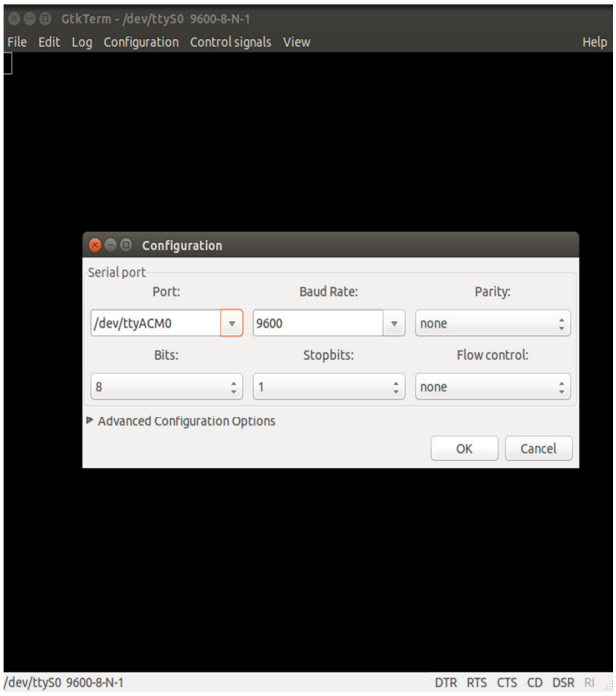

Figure 7. Connecting Zodiac FX port

Gtkterm can be configured the zodiac FX which provides the port to configure. It shows different ports are connected in system while /dev/ttyACMO is representing to Zodiac FX. Once chose the port, it can be configured Zodiac FX. Figure 8 shows the configuration. As mentioned in Zodiac FX manual port four is the OpenFlow control plane port and while other ports are the OpenFlow forwarding plane ports.

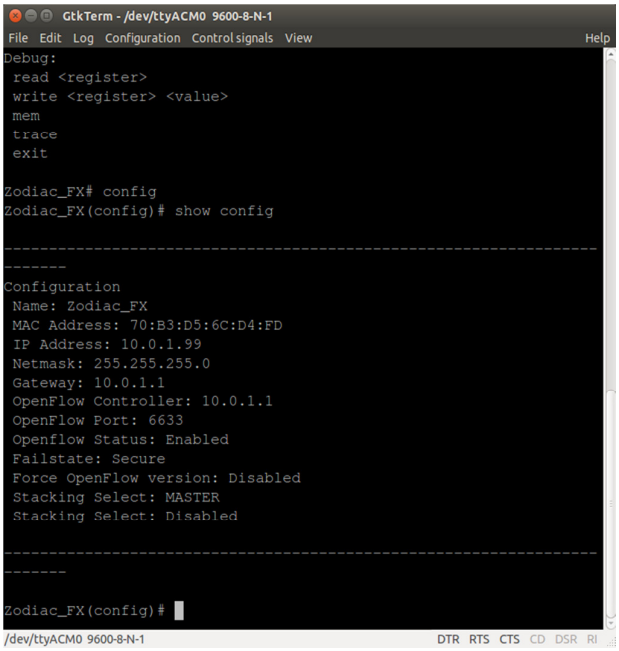

Figure 8. Zodiac FX configuration

To design a simple SDN testbed, we have used Ryu controller open source available codes, and Zodiac FX a hardware switch. Zodiac FX is a four-port hardware, one port is connected directly to Ryu controller while other ports are connected to hosts, figure 9 shows the connectivity of Ryu controller and Zodiac FX.

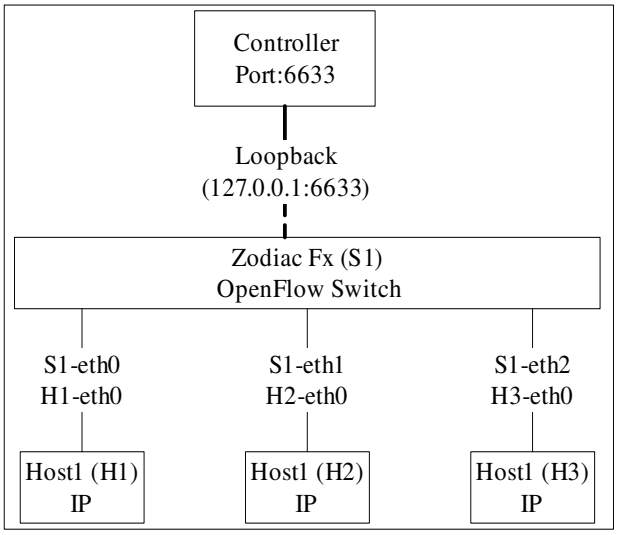

Figure 9. FX connection with Ryu controller

3. Conclusion

In this paper, we developed SDN Testbed and surveyed the design of OpenFlow and SDN Ryu Controller. It is observed a real programmable network by setting up a Ryu SDN controller and Zodiac FX for the SDN scenario. We presented all possible configurations that show how to build a real SDN scenario via a hardware switch Zodiac FX. Furthermore, we have provided a detail of OpenFolow and Ryu SDN controller that can provide a better observation of the setup and design at the implementation level. Via presented configuration, SDN Testbed can be further analyzed through the performance. All possible configurations are provided in this paper based on SDN standard OpenFlow protocol and Ryu SDN open source controller. Hence, it can say that SDN is more flexible than traditional network and main advantage of SDN is a cost efficient and programmable. Zodiac FX can be used to perform more experiments and examine the performance via Rya OpenFlow controller and other SDN controllers. It is observed that the major limitation of SDN controllers that if any active controller fails it can rapidly break down the entire network.

References

[1] H. Farhady, H. Lee, A. Nakao. Software-Defined Networking: A survey. In Elsevier, Computer Networks, Volume 81, 22 April 2015, Pages 79–95.
[2] Sezer S, Scott-Hayward S, Chouhan PK and Fraser B, Lake D, Finnegan J, and Viljoen N, Miller M, Rao N. Are we ready for SDN? Implementation challenges for software-defined networks. Communications Magazine, IEEE 2013. p. 36-43.
[3] Stanford, OpenFlow Available: http://cleanslate.stanford.edu/
[4] N. McKeown et al., “OpenFlow: Enabling innovation in campus networks,” SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Apr. 2008.
[5] R. Sherwood et al., “Can the production network be the testbed?” in Proc. 9th USENIX Conf. OSDI, 2010, pp. 1–6.
[6] “OpenFlow switch specification,” version 1.4.0 (Wire Protocol 0x05), Oct. 2013.
[7] Z. Bozakov and V. Sander, “OpenFlow: A perspective for building versatile networks,” in Network-Embedded Management and Applications, A. Clemm and R. Wolter, Eds. New York, NY, USA: Springer-Verlag, 2013, pp. 217–245.
[8] A. Lara, A. Kolasani, and B. Ramamurthy, “Network innovation using OpenFlow: A survey,” IEEE Communication Surveys Tutorials, vol. 16, no. 1, pp. 493–512, 1st Quart. 2014.
[9] OpenFlow. [Online]. Available: http://www.openflow.org/
[10] N. McKeown et al., “OpenFlow: Enabling innovation in campus networks,” ACM SIGCOMM Comput. Commun. Review, vol. 38, no. 2, pp. 69–74, Apr. 2008.
[11] NOX. Available: http://www.noxrepo.org/nox/about-nox/
[12] POX.Available: http://www.noxrepo.org/pox/about-pox/
[13] Trema: Full-Stack OpenFlow Framework in Ruby and C. Available: https://github.com/trema/
[14] Floodlight: A Java-Based OpenFlow Controller. Available: http://floodlight.openflowhub.org/
[15] A. Orebaugh, G. Ramirez, J. Burke, and L. Pesce, Wireshark & Ethereal Network Protocol Analyzer Toolkit (Jay Beale’s

Open Source Security). Sebastopol, CA, USA: Syngress Publishing, 2006.
[16] J. Pelkey, OpenFlow Software Implementation Distribution,Available: http://code.nsnam.org/jpelkey3/openflow
[17] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in Proc. ACM SIGCOMM Workshop Hot Topics Netw., New York, NY, USA, 2010, pp. 19:1–19:6.
[18] OpenVSwitch.Available: http://openvswitch.org/development/openflow-1-x-plan
[19] Ryu SDN framework web page. http://osrg.github.io/ryu/
[20] Ryu project team. Ryu SDN Framework - English Edition. RYU project team, 2014.
[21] Zodiac FX a hardware switch manual [Online], Available: http://forums.northboundnetworks.com/