Science Publishing Corporation

International Journal of Engineering & Technology

www.sciencepubco.com/index.php/IJET

ISSN 2227-524X

Dr. Nicolae Crisan, Technical University of Cluj-Napoca, Romania

Mr. Tabara Octavian Adrian, University Politehnica of Bucharest, Romania

Prof.Dr.Md. Osman Goni, Khulna University of Engineering and Technology, Bangladesh

Dr Fabio Mottola, University of Naples Federico II Department of Electrical Engineering and Information technology, Italy

Adebowale Shadare, Prairie View A&M University, United States

Dr Roozbeh Abedini Nassab, Mechanical Engineering and Materials Science Department, Duke University, United States

Dr Sunil Patil, ANSYS Inc., United States

Prof Ahmad Mujahid Ahmad Zaidi, Faculty of Engineering, National Defense University of Malaysia, Malaysia

Prof. M. Dev Anand, Professor and Deputy Director Academic Affairs, India

Prof Jawad K. Ali, Microwave Research Group, Department of Electrical Engineering, University of Technology, Iraq

Dr. Eng. Liliana Marilena MATICA, University of Oradea, Romania

Ivan Protsenko, Sumy State University, Ukraine

Dr. Muhammad Anisuzzaman Talukder, University of Maryland, Baltimore County, Baltimore, MD 21250, USA, Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000, Bangladesh

Prof. Lorand Szabo, Technical University of Cluj-Napoca, Romania

Mr. Spehro Pefhany, Trexon Inc., Canada

Jean-Bernard Fullenwarth, Alcatel-Lucent / Accenture, France

Prof.Dr. Chiu Huang-Jen, National Taiwan University of Science and Technology

Prof Valentina Emilia Balas Aurel Vlaicu University of Arad, Romania, Romania

Khaled Bataineh, Jordan University of Science and Technology, Jordan
Angelo Algieri, University of Calabria, Italy

Yi-Chan Chung, The Department and the Graduate Institute of Business Administration, Yuanpei University, Taiwan

Dr. Raja Rizwan Hussain,
King Saud University, Saudi Arabia

International Journal of Engineering & Technology, 8 (1.9) (2019) 17-22

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Link Redundancy for High Availability Network based on OpenFlow Software Defined Network

Indrarini Dyah Irawati[1*], Sugondo Hadiyoso[1], Yuli Sun Hariyani [1]

1School of Applied Science, Telkom University, Bandung, Indonesia 40257, Indonesia
**Corresponding author E-mail: indrarini@telkomuniversity.ac.id*

Abstract

Nowadays, Internet traffic is growing rapidly, as a result needed a realible network connectivity. The problem arise when the network is damage, i.e., link failure, server failure. It is important to create high availability network. This paper proposed link redundancy for high availability network based on OpenFlow Software Defined Network (SDN). OpenFlow supports port grouping for handling fast-failure while link broken. In this paper, we use cascade topology that consists 2-layer with 5 switches and emulate it using tools mininet and Ryu controller. The results for all scenario show that fast-failure method can detect link failure and recover without terminate the connection.

Keywords: *Software Define Network, High Availibility Network, Link Failure, OpenFlow*

1. Introduction

The growth of Internet network users being rapidly lead to a surge of network traffic. It will cause a decrease in network performance that will be felt on the user.

Therefore, the required settings dynamic network resources to improve network performance and lower latency. In this study designed a high availability network (HA network) based on OpenFlow that is dynamic and flexible according to the needs.

In the previous research, the simulation of Spanning Tree Protocol (STP) based on Software Defined Network using mininet emulator have been done (Irawati et al., 2015). The result shows that the network can handle a link failure using STP because it provides backup links between switches. STP also stops for flooding and avoids broadcast storm on the network. However, it takes about 60s to change the status of port when a link failure occurs until the backup link works well. Therefore, in this study, we design link redudancy for HA network based on openflow fast-failover group on Software Defined Network.

2. Research Methodology

2.1. OpenFlow Group Table

There are three tables that are defined by the OpenFlow in the logical architecture OpenFlow Switch, the Flow Table, Group Table and Meter Table. Open flow group table is part of the open flow consisting of group entries. Group table is an additional method of forwarding scheme. Group tables were implemented in 1.1 OpenFlow is used to do more complex process/algorithm on packets that can't be defined within a flow itself (Open Networking Foundation. 2013.).

In OpenFlow, groups are action for flooding and forwarding semantics which more complex. The examples of group actions are multipath, fast reroute, and link aggregation. Groups enable to carry multiple functions entries on a single identifier (ie. IP forwarding to the common next hop). Hence, it make more efficient actions. (Open Networking Foundation. 2013.).

The groups table contains lists of actions capability, and each action list is depends on the OpenFlow bucket. The list can put into the packets entering. The appropriate behavior depends on the kind of group. There are several kind of groups that apply OpenFlow additional parameters that will be explained in detail on each type of OpenFlow groups (Izard, 2016).

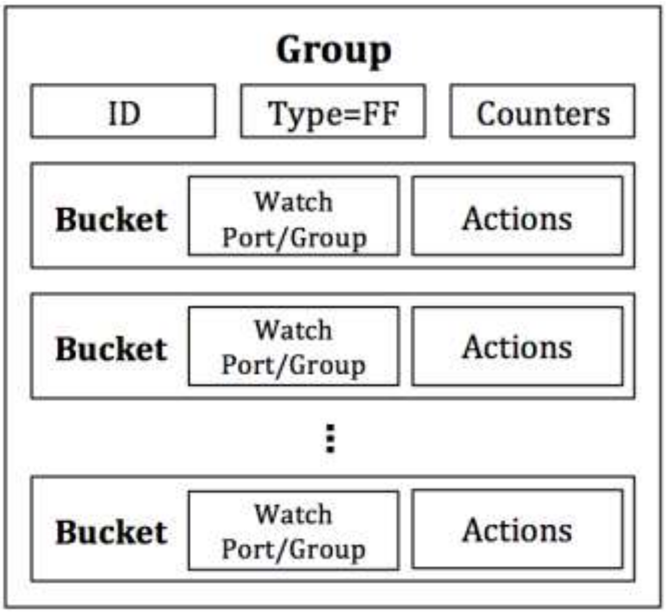


Fig 1: The main components of group entries in group table (Izard. 2016).

Each group entry (see Figure 1) is identified by its group identifier and contains:

• ID the unique code (32 bit length)

• Type to define group semantics

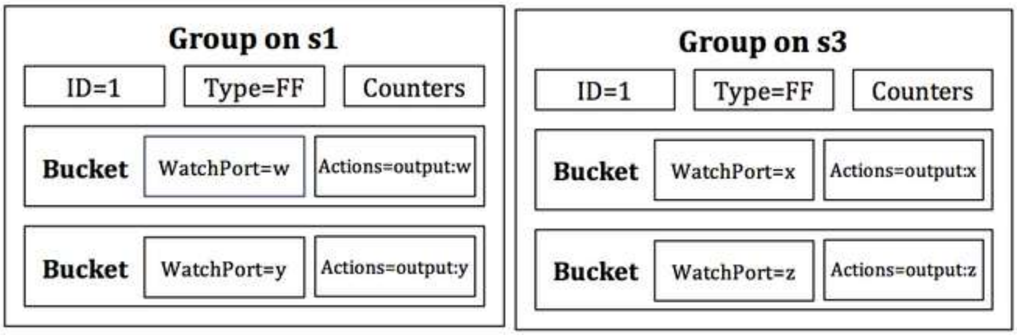
• Counters status updated as packets are worked on group

• Buckets an ordered list of action buckets, where each action bucket hold a group of instructions to process and associated parameters. (Open Networking Foundation. 2013.)

2.2. The Fast-Failover Group

The Fast-Failover (FF) group is a set of buckets, where the buckets have a special parameter (watch port and / or watch group). In Figure 2, it represents of the group that will be installed in s1 and s3. The watch port monitor the status of the group, whether in up or down position. The down status indicates that the bucket can't work. Then, if the group status is up, the bucket can work. There is only one bucket can work at a time.

In fact, there is no time guarantee of transition to replace a new bucket when a failure happened. The transition time is depend on search time to find a group that is up and on the switch implementation. However, the motivation behind using a Fast-Failover group is that it is almost guaranteed to be faster than consulting the control plane to handle the port down event and inserting a new flow or set of flows. With Fast-Failover groups, link failure detection and recovery takes place entirely in the data plane. (Izard, 2016).



(a) FF group on s1 (b) FF group on s2

Fig 2: Fast-Failover (FF) group. Note the correlation between the watch port and output port within each bucket. (Izard, 2016).

3. Scenario

We use mininet emulator version 2.3.0d1 that installed on Ubuntu 16.04 based on x64 bit Operating System and run the cascade network topology using Open vSwitch and OpenFlow version 1.3.

3.1. Network Design

In Figure 3 shows network design. The network is designed using cascade topology that consists 2-layer as a backbone. The first layer has 2 switches (S2, S4) and the second layer has 3 switches (S1, S3, S5). There are 2 hosts as a client (H1) and a server (H2), and 3 types of link. The main link is the primary link that used to transmit data packets between switches. The backup link is a link redundancy that works when the primary link is damaged. Whereas the control link is a connecting link between the switch to the controller. The controller uses Ryu software that support openflow v1.3.

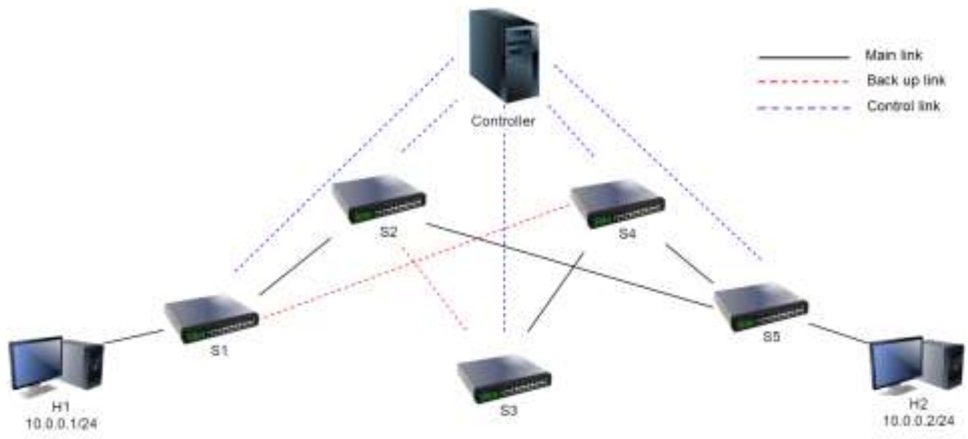


Fig 3: Network Design

3.2. Flow Table Mechanism

Ryu controller distributes flow table entry and group table entry to all switches according to the scenario, hence H1 and H2 can communicate each other, avoid broadcast storm and support link failure detection.

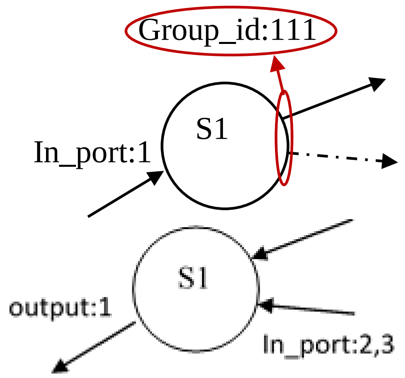


Fig 4: Flow table

Figure 4 show S1 flow table. When packet in received from port 1, then switch will forward the packet to group id. Group table schema can be seen in figure 2. The group using bucket to identify watch_port and output according destined port. And, if the packet received from port 2 or 3, the packet will forward via port 1. Flow table entry for each switch can be seen in Table 1. While FF group table for each switch can be seen di Table 2.

Table 1: flow table entry each switch

Switch	Input	Output
S1	Port:1	Group:1111
	Port:2,3	Port:1
S2	Port:1	Group:2211
	Port:2,3	Port:1
S3	Port:1	Port:2
	Port:2	Port:1
S4	Port:1,2	Port:3
	Port:3	Group:4422
S5	Port:2,3	Port:1
	Port:1	Group:5522

Table 2: Fast-Failover group table each switch

Switch	Group_id	Bucket	
		Watch_port	Output
S1	1111	Port:2	Port:2
		Port:3	Port:3
S2	2211	Port:2	Port:2
		Port:3	Port:3
S4	4422	Port:1	Port:1
		Port:2	Port:2
S5	5522	Port:2	Port:2

		Port:3	Port:3

3.3. Performance Evaluation

We use 2 scenarios for evaluating the performance of link redudancy (see in Figure 5). The first scenario is breaking the link between S1 and S2. The second scenario is breaking the link between S2 and S5. The connectivity test is performed by sending Packet Internet Grouper (PING) from H1 to H2. We use wireshark to measure recovery time and throughput while the link broken.

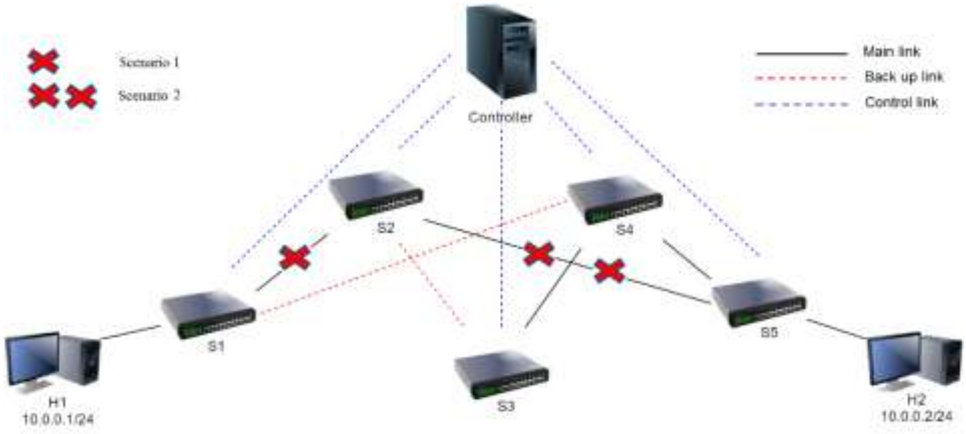


Fig 5: Link down scenarios

3.4. Link down evaluation

Link down evaluation can be done using Packet Internet Grouper (PING) to check the connectivity between H1 and H2. While PING is running, the link will be disconnected. The observation were made on the network. In figure 6, shown the respon from controller when S1 to S2 down.

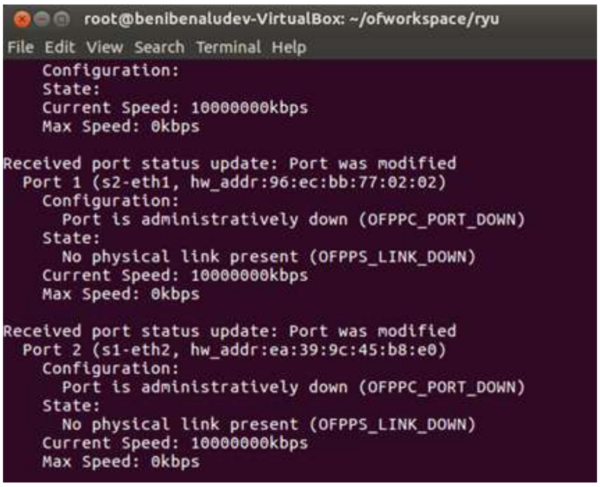


Fig 6 Controller response for link S1 to S2 down

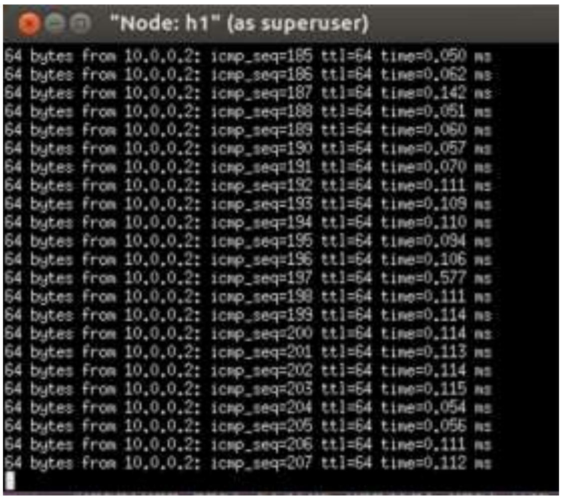


Fig 7 PING from H1 to H2

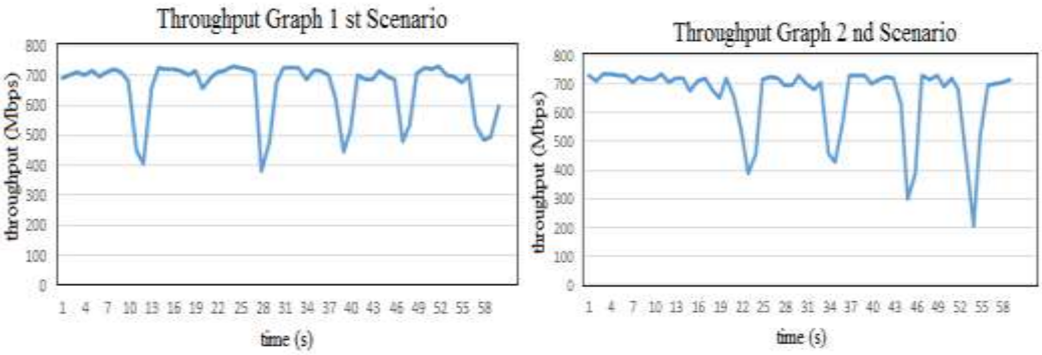
We simulate link disconnection at icmp_seq=196. In Figure 7 shows that the end-to-end communication between H1 and H2 still connected. In table 3 shown the impact of link down on the network.

Table 3: link down evaluation in HA network

Skenario	Source	Destination	Link					Status
			S1	S2	S3	S4	S5	
1	H1	H2	X	X	O	O	O	Connected
	H2	H1	X	X	O	O	O	
2	H1	H2	O	X	O	O	X	Connected
	H2	H1	O	X	O	O	X	

3.5. Throughput evaluation

We use Iperf for measuring the throughput between H1 as a client and H2 as a server using UDP packet with 1GB load in 60 second. In Figure 8, shown when the interupption of the link, the throughput decrease for some times, but the connection still persist. The result of average throughput measurement are 685.96 Mbit/sec in scenario 1 (Figure (a)) and 612.88 Mbps/sec in scenario 2 (Figure (b)).



(a)1st Scenario (b) 2nd Scenario

Fig 8 Throughput graph

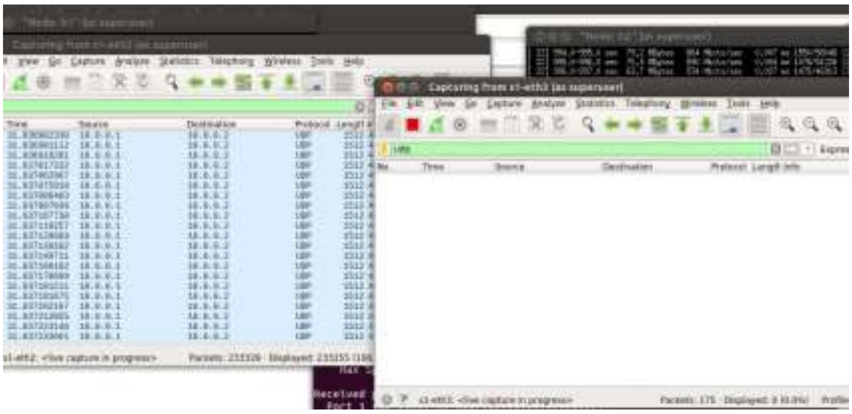


Fig 9: S1-S2 normal

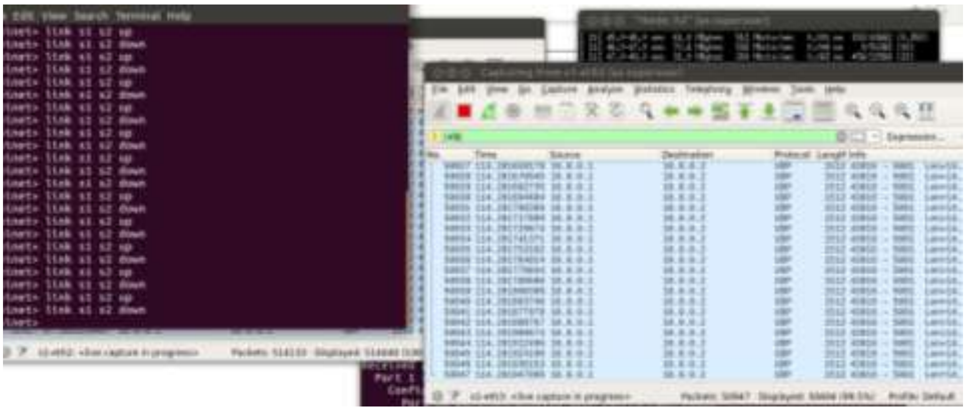


Fig 10: S1-S2 down

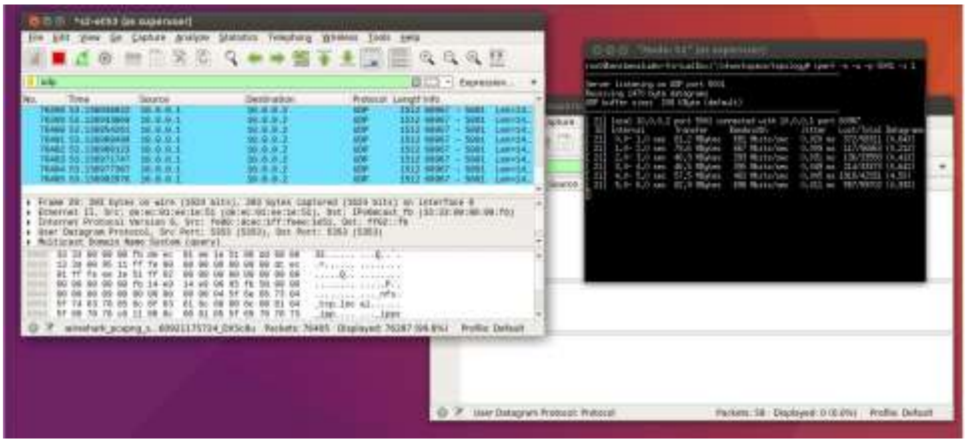


Fig 11: S2-S5 normal

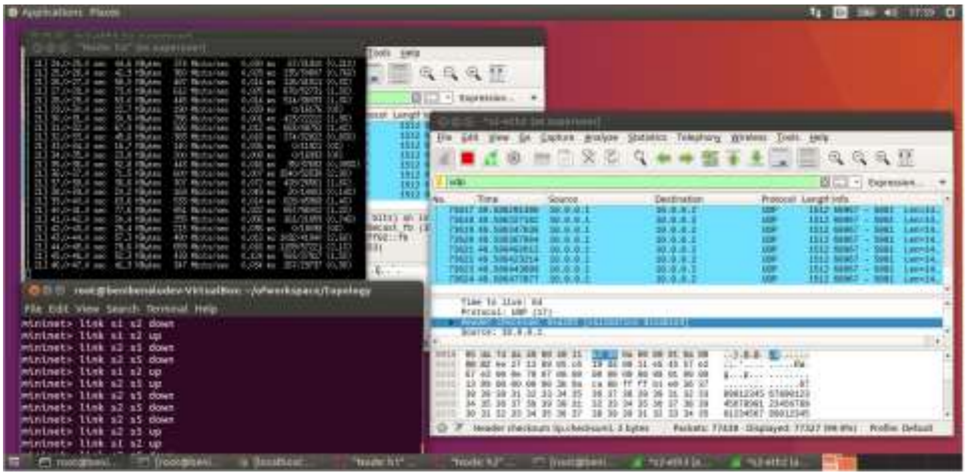


Fig 12: S2-S5 down

4. Discussion and Conclusion

We conclude our research that the OpenFlow Fast-Failover group table can handle link failure so that the High Availability Network can be realized. Fast-Failover method can detect link failure and recover the link without the occurrence of termination. When link is broken, network performance is still maintained, that the average throughput measurement is about 685.96 Mbit/sec in scenario 1 and 612.88 Mbps/sec in scenario 2.

Acknowledgement

This work was supported and funded by KemenRistekDIKTI under Hibah Bersaing research grant.

References

[1] Braun, Wolfgang & Michael Menth. (2014). Softward-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choice, Germany: Future Internet 2014, 6

[2] Hariyani. Yuli Sun., Indrarini Dyah Irawati, Danu Dwi S. & Mohammad Nuruzzamanirridha. (2015). Routing Implementation Based-on Software Defined Network using Ryu Controller and OpenvSwitch. Jurnal Teknologi 78:5, 295-298

[3] Izard, Ryan. (2016). How to Work with Fast-Failover OpenFlow Groups. [Online].Available: https://floodlight.atlassian.net/

[4] Irawati, Indrarini Dyah & M. Nuruzamaniridha. (2015). "Spanning Tree Protocol Simulation Based on Software Defined Network Using Mininet Emulator" ICSIIT, 395-403.

[5] Li, Cheng. Multipath and QoS Application on RYU [Online]. Available: http://www.muzixing.com/pages/2014/11/07/multipath-and-qos-application-on-ryu.html

[6] Adrichem, Niels L. M. van, Benjamin J. van Asten & Fernando A. Kuipers (2014). Fast Recovery in Software-Defined Networks. IEEE Computer Society, 61-66

[7] Open Network Foundation (2012). OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04). ONF TS-006 [Online]. Available: https://www.opennetworking.org

[8] Open Networking Foundation. (2013). OpenFlow Switch Specification Version 1.4.0. [Online]. Available: http://www.opennetworking.org

[9] Patel, Gunjan, Athreya, Adithi S., Erukulla, Swetha. (2013). OpenFlow Based Dynamic Load Balanced Switching. COEN233, Project Report.

[10] Ryu Project Team. Ryu SDN Framework. [Online]. Available: http://osrg.github.io/ryu-book/en/Ryubook.pdf

[11] Catro, Flavio. (2014). Shortest Path forwarding with Openflow on RYU. [Online]. Available: https://sdn-lab.com/2014/12/25/shortest-path-forwarding-with-openflow-on-ryu/

[12] Scott, Leo. (2015). Subscribing to Port Events. [Online]. Available: http://ofdpa.com/2015/06/10/subscribing-to-port-events/

[13] Yamahata, Isaku. (2013) Ryu: SDN framework and Python experience, Japan: Pycon APAC 2013, September 14