

Link Aggregation Control Protocol on Software Defined Network

I. D. Irawati¹, Y. Sun Hariyani², S. Hadiyoso³

Telkom Applied Science School, Telkom University, Indonesia

Article Info

Article history:

Received Mar 29, 2017

Revised May 31, 2017

Accepted Aug 11, 2017

Keywords:

Link Aggregation,
Redundancy,
Ryu controller,
SDN,
Traffic

ABSTRACT

A physical connection of computer network must be made reliably. Breaking connection will cause communication between nodes (for example routers, switches, hosts) can be disconnected. One of the solutions is implementation of link aggregation (LA). LA integrates several of physical ports together to make a single logical communication link. Accordingly, there is load sharing traffic among the member port of the group, high-throughput increasing via a single link, and redundancy providing for broken links. We present the implementation of link aggregation using Ryu controller on Software Defined Network (SDN) topology. The results show that the implementation of SDN with OpenvSwitch and Ryu controller can successfully run link aggregation function to solve the problem of link failure.

Copyright © 2017 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

I. D. Irawati,
Telkom Applied Science School,
Telkom University,
Telekomunikasi Rd. Terusan Buah Batu,
Bandung 40257 Indonesia.
Email: indrarini@telkomuniversity.ac.id

1. INTRODUCTION

Switch or router over the network [4]. OpenFlow is developed by Open Networking Foundation (ONF) Today, the performance of computer networks is a great concern by the customers. Network performance is influenced by the reliability of the physical network topology. When the physical connection between two nodes is failed, then the packet can not be sent through the link. Otherwise, if a physical connection is a good condition with limited bandwidth, it can cause congestion since several nodes using the link for delivery. This problem can be solved by Link Aggregation (LA). LA is a method that aggregates multiple network connections acts as a single logical interface using software. LA provides redundancy for link failure, for example: if one or more physical interface on a logical aggregate loss, the other physical connection still up and operate. Moreover, LA increases throughput passing through a single link which can reduce bottleneck by allowing packets to be traversed over multiple interfaces. LA is defined in IEEE 802.3ad that allows to group Ethernet interface at the physical layer to form a single link interface, namely Link Aggregation Group (LAG). Link Aggregation Control Protocol (LACP) is a mechanism for changing the port to preserve LAG bundles. LACP provides a dynamic configuration that means the other end can handle link aggregation. In addition, it handles failover automatically when a link failed [1].

Software Defined Network (SDN) is a new architecture for a computer network that separates network control and forwarding functions, which allowing the controller can connect directly programmable. SDN's characteristics are manageable, adaptable, scalable, appropriate for high bandwidth, and dynamic features of recent applications compared to traditional network architecture [2]. Hence, users can assign the logical network topology using software, regardless of underlying network structure [3]. In general, SDN

associates with OpenFlow protocol as a communication protocol that remote communication to the forwarding plane of a network as an organization that promotes networking through SDN and standardizing the OpenFlow protocol. The OpenFlow architecture consists of OpenFlow controller and OpenFlow Switch. The function of OpenFlow controller manages OpenFlow switch.

Studies on link redundancy on SDN had been done [5]-[7]. For example, M. Steinbacher et al. presented LACP implementation using Floodlight OpenFlow controller [5] and P. Skoldstrom et al. proposed virtual aggregation forwarding model on backbone topology [7]. While I.D. Irawati et al. designed link redundancy for high availability network based on OpenFlow fast-failover for cascade SDN topology.

Other related research in terms of maintaining quality and network performance presented in [8]-[9]. Yahya et.al present Extended Dijkstra's with (edges and nodes weights) parameters to find the shortest server [8]. This algorithm is used for SDN applications. The use of a genetic algorithm load balancer for service recovery on the intercloud network is done by Jena et al. [9].

In this paper, we present the implementation of IEEE 802.3ad dynamic LACP configuration on SDN topology. We apply the LACP between a server and a switch using OpenFlow, i.e., Ryu controller. Ryu manages LACP function to perform failover functionality.

2. BASIC THEORY

2.1. SDN Architecture

Software Defined Network (SDN) is an approach to apply open protocols, such as OpenFlow to implement software control by the edge of the network to access the entire network devices. The architecture of SDN is as shown in Figure 1. It composes by three layers, such as infrastructure layer, control layer, and application layer. An Infrastructure layer represents network device as a switch. Control layer manages both Southern interface and northern interface. The Southern interface is used to control the state of the network. While northern interface provides program applications to configure, supervise, and optimize the service applications [10].

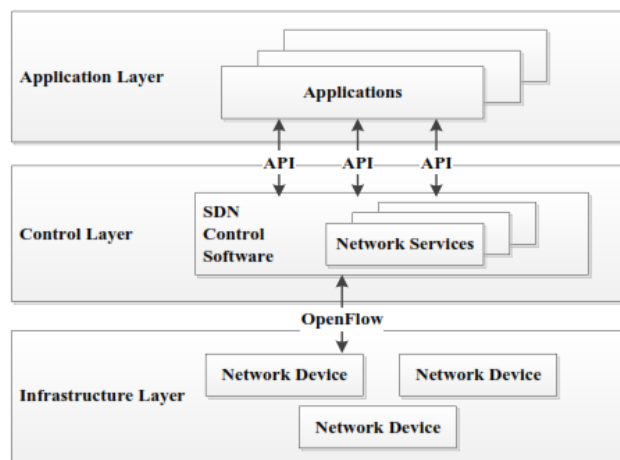


Figure 1. SDN architecture [8]

2.2. Link Aggregation Control Protocol (LACP)

Link aggregation is a technique noted in IEEE 802.1ax-2008 by David Law for local and metropolitan area network link aggregation standard [11]. And an automatic configuration feature that known as link aggregation control protocol (LACP) is determined in IEEE 802.3ad [1]. LA combines several physical interfaces to be as a logical link that useful for increasing bandwidth between network devices and handling fault tolerance by link redundancy. LA functions are shown in Figure 2 [11].

There are several features of LACP. The maximum quantity of integrated port appropriates with the channels of the device, usually 1 to 8 channels. LACP packets use multicast MAC address 01:80:c2:00:00:02 for sending packets. LACP provides a load-balance mode that identifies the member of links which serve load balancing. LACP mode has two states, i.e., active and passive. Active mode enables LACP unconditionally, while passive mode allows LACP only when an LACP device is detected [12].

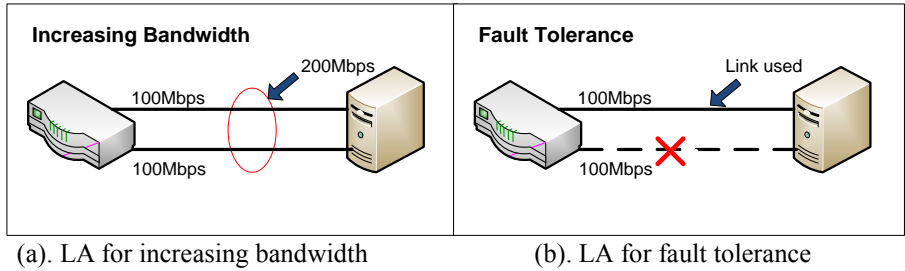


Figure 2. Link aggregation function, (a). Increasing bandwidth, (b). Fault tolerance [12]

3. DESIGN AND IMPLEMENTATION

We design and implement LACP according to the Ryu controller using Python programming language. The topology consists of Controller (C0), Switch (S1), Host 1 (H1) as a server, and Host 2 (H2) as a client that is shown in Figure 3. Host1 configuration is intended to create a server that supports link aggregation capabilities. It is configured by using modprobe bonding mode four that indicates the dynamic LA is performed using LACP. There is two physical link connection between S1 and server H1 via eth0 and eth1 that run LACP function.

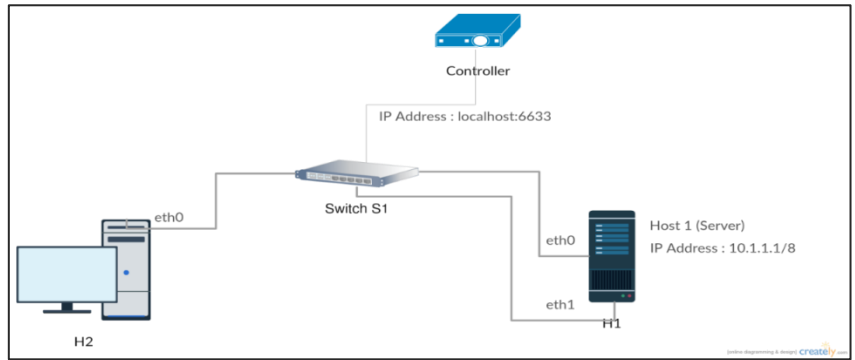


Figure 3. Implementation topology

4. RESULTS AND ANALYSIS

The configuration of link aggregation on server H1 is shown in Figure 4. The LACP setup is successful. LACP rate is slow with 30-second intervals.

```

root@benih:~/ha-network/single-node# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer2 (0)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

802.3ad info
LACP rate: slow
Min links: 0
Aggregator selection policy (ad_select): stable
Active Aggregator Info:
  Aggregator ID: 1
  Number of ports: 2
  Actor Key: 33
  Partner Key: 33
  Partner Mac Address: 22:90:38:53:61:44

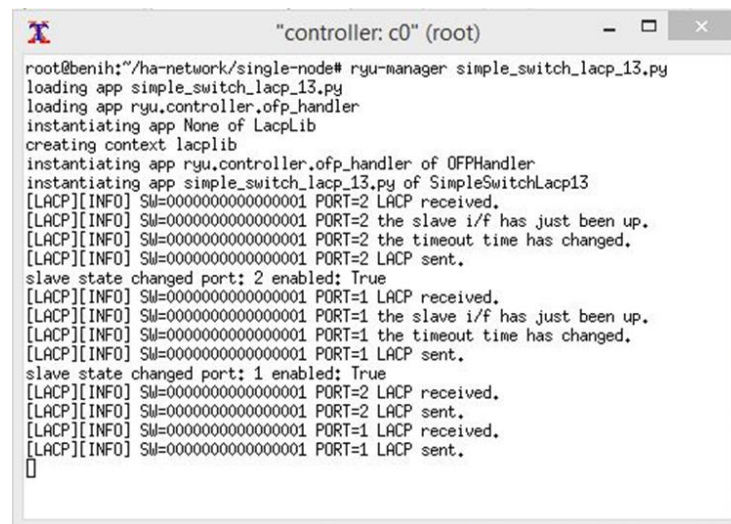
Slave Interface: h1-eth0
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:00:00:00:11
Aggregator ID: 1
Slave queue ID: 0

Slave Interface: h1-eth1
MII Status: up
Speed: 10000 Mbps
Duplex: Full
Link Failure Count: 0
Permanent HW addr: 00:00:00:00:21
Aggregator ID: 1
Slave queue ID: 0
    
```

Figure 4. LA configuration on server H1

H1-eth0 and H2-eth1 act as link aggregation with up status and each link have 10000 Mbps rate.

Ryu controller runs LACP function to perform fault tolerance functionality. SDN Apps operates simple_switch_lacp_13 that running on OpenFlow 1.3. When server H1 transmits LACP data unit every 30 seconds, the Switch S1 takes in the LACP data unit from server H1. We can see the process that occurs on the controller C0 as shown in Figure 5. LACP received status indicates an LACP data unit was received by the port and LACP sent status means an LACP data unit was sent by the port. Whereas the slave i/f has just working state represents a change in state from disable to enable. The state of the LACP data unit time has changed denotes that the time for communication monitoring was changed. In this event, the standard state is converted into long timeout periode (0 to 90 seconds).



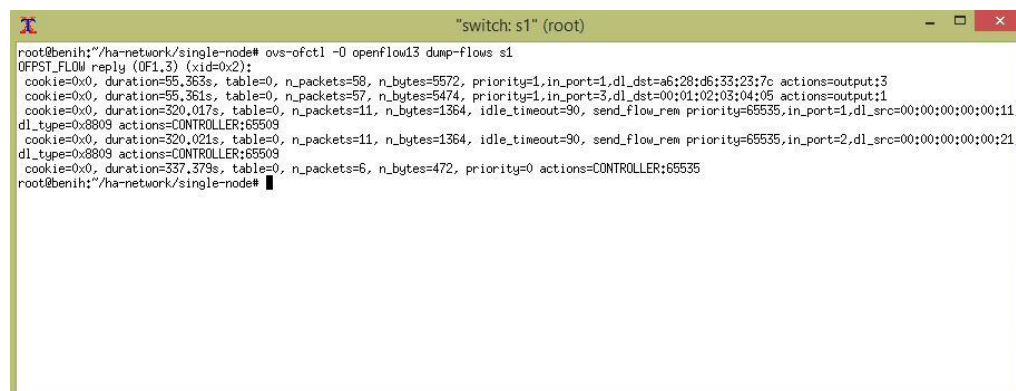
```

root@benih:~/ha-network/single-node# ryu-manager simple_switch_lacp_13.py
loading app simple_switch_lacp_13.py
loading app ryu.controller.ofp_handler
instantiating app None of LacpLib
creating context lacplib
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app simple_switch_lacp_13.py of SimpleSwitchLacp13
[LACP][INFO] SW=0000000000000001 PORT=2 LACP received.
[LACP][INFO] SW=0000000000000001 PORT=2 the slave i/f has just been up.
[LACP][INFO] SW=0000000000000001 PORT=2 the timeout time has changed.
[LACP][INFO] SW=0000000000000001 PORT=2 LACP sent.
slave state changed port: 2 enabled: True
[LACP][INFO] SW=0000000000000001 PORT=1 LACP received.
[LACP][INFO] SW=0000000000000001 PORT=1 the slave i/f has just been up.
[LACP][INFO] SW=0000000000000001 PORT=1 the timeout time has changed.
[LACP][INFO] SW=0000000000000001 PORT=1 LACP sent.
slave state changed port: 1 enabled: True
[LACP][INFO] SW=0000000000000001 PORT=2 LACP received.
[LACP][INFO] SW=0000000000000001 PORT=2 LACP sent.
[LACP][INFO] SW=0000000000000001 PORT=1 LACP received.
[LACP][INFO] SW=0000000000000001 PORT=1 LACP sent.

```

Figure 5. Ryu controller runs LACP

The information of flow entry on S1 is shown in Figure 6. LACP data unit from server H1 is received by the Switch S1. Switch S1 responds by sending LACP data unit. The Switch S1 replays by Packet-In message in case LACP data unit with ethernet type 0x8809 is transmitted from H1-eth0 using the s1-eth1 input port and the MAC address 00:00:00:00:00:11. Moreover, the Switch S1 replays by the Packet-In message in case LACP data unit with ethernet type 0x8809 is transmitted from H1-eth1 using the s1-eth2 input port and the MAC address 00:00:00:00:00:21. The switch ports have been able to perform the LACP functions.



```

root@benih:~/ha-network/single-node# ovs-ofctl -O openflow13 dump-flows s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
cookie=0x0, duration=55.363s, table=0, n_packets=58, n_bytes=5572, priority=1,in_port=1,dl_dst=a6:28:d6:33:23:7c actions=output:13
cookie=0x0, duration=55.361s, table=0, n_packets=57, n_bytes=5474, priority=1,in_port=3,dl_dst=00:01:02:03:04:05 actions=output:1
cookie=0x0, duration=320.017s, table=0, n_packets=11, n_bytes=1364, idle_timeout=90, send_flow_rev priority=65535,in_port=1,dl_src=00:00:00:00:11,
dl_type=0x8809 actions=CONTROLLER:65509
cookie=0x0, duration=320.021s, table=0, n_packets=11, n_bytes=1364, idle_timeout=90, send_flow_rev priority=65535,in_port=2,dl_src=00:00:00:00:21,
dl_type=0x8809 actions=CONTROLLER:65509
cookie=0x0, duration=337.379s, table=0, n_packets=6, n_bytes=472, priority=0 actions=CONTROLLER:65535
root@benih:~/ha-network/single-node#

```

Figure 6. Switch LACP

In Figure 7, we test the LACP fault tolerance by breaking the link of H1-eth0. We run the command `ip link set h1-eth0 nomaster`. The state of controller C0 is shown in Figure 8. The state of LACP exchange timeout has occurred denotes that there is no communication controlling until elapsed time. In addition, the status of slave state changed port: 1 enable: False represents that there is a change in handling of data transmission from port 2 to port 1. Then, all listed that contained previous MAC addresses and flow entries will be deleted automatically. If the new communication will be recreated, the new flow entries and new MAC addresses are registered again.

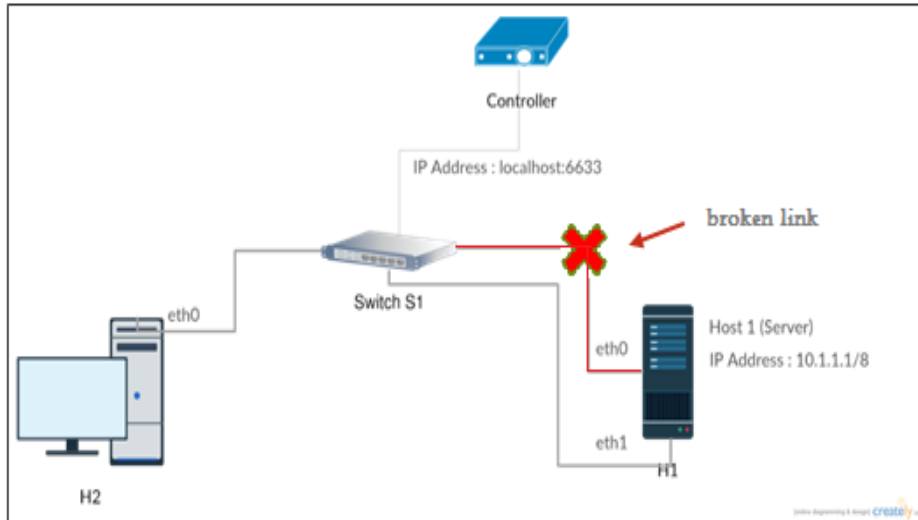


Figure 7. Link broken scenario

```

"controller: c0" (root)
[LACP][INFO] SW=0000000000000001 PORT=2 LACP sent.
packet in 1 7a:8d:71:20:a6:48 ff:ff:ff:ff:ff:ff 3
[LACP][INFO] SW=0000000000000001 PORT=2 LACP received.
[LACP][INFO] SW=0000000000000001 PORT=2 LACP sent.
[LACP][INFO] SW=0000000000000001 PORT=2 LACP received.
[LACP][INFO] SW=0000000000000001 PORT=2 LACP sent.
packet in 1 7a:8d:71:20:a6:48 ff:ff:ff:ff:ff:ff 3
[LACP][INFO] SW=0000000000000001 PORT=1 LACP exchange timeout has occurred.
slave state changed port: 1 enabled: False
packet in 1 7a:8d:71:20:a6:48 00:01:02:03:04:05 3
packet in 1 00:01:02:03:04:05 7a:8d:71:20:a6:48 2
packet in 1 7a:8d:71:20:a6:48 00:01:02:03:04:05 3
[LACP][INFO] SW=0000000000000001 PORT=2 LACP received.
[LACP][INFO] SW=0000000000000001 PORT=2 LACP sent.
packet in 1 00:01:02:03:04:05 33:33:00:00:00:16 1
packet in 1 00:01:02:03:04:05 33:33:ff:03:04:05 1
packet in 1 00:01:02:03:04:05 33:33:00:00:00:01 2
[LACP][INFO] SW=0000000000000001 PORT=1 LACP received.
[LACP][INFO] SW=0000000000000001 PORT=1 the slave i/f has just been up.
[LACP][INFO] SW=0000000000000001 PORT=1 the timeout time has changed.
[LACP][INFO] SW=0000000000000001 PORT=1 LACP sent.
    
```

The port state changed into false

Figure 8. Ryu controller is shown link break

In Figure 9 is shown the ping state of H1 to S1 connection. Because of broken link on H1-eth0 to S1, ping can not be sent from H1 to S1. The time required for recovery is about 90 seconds. The throughput result is shown in figure 10. The observation is done in 600 seconds. When the event of link failure, the throughput will decrease until 0 Mbps over 90 seconds. The average of throughput in normal condition is about 7413.33 Mbps.

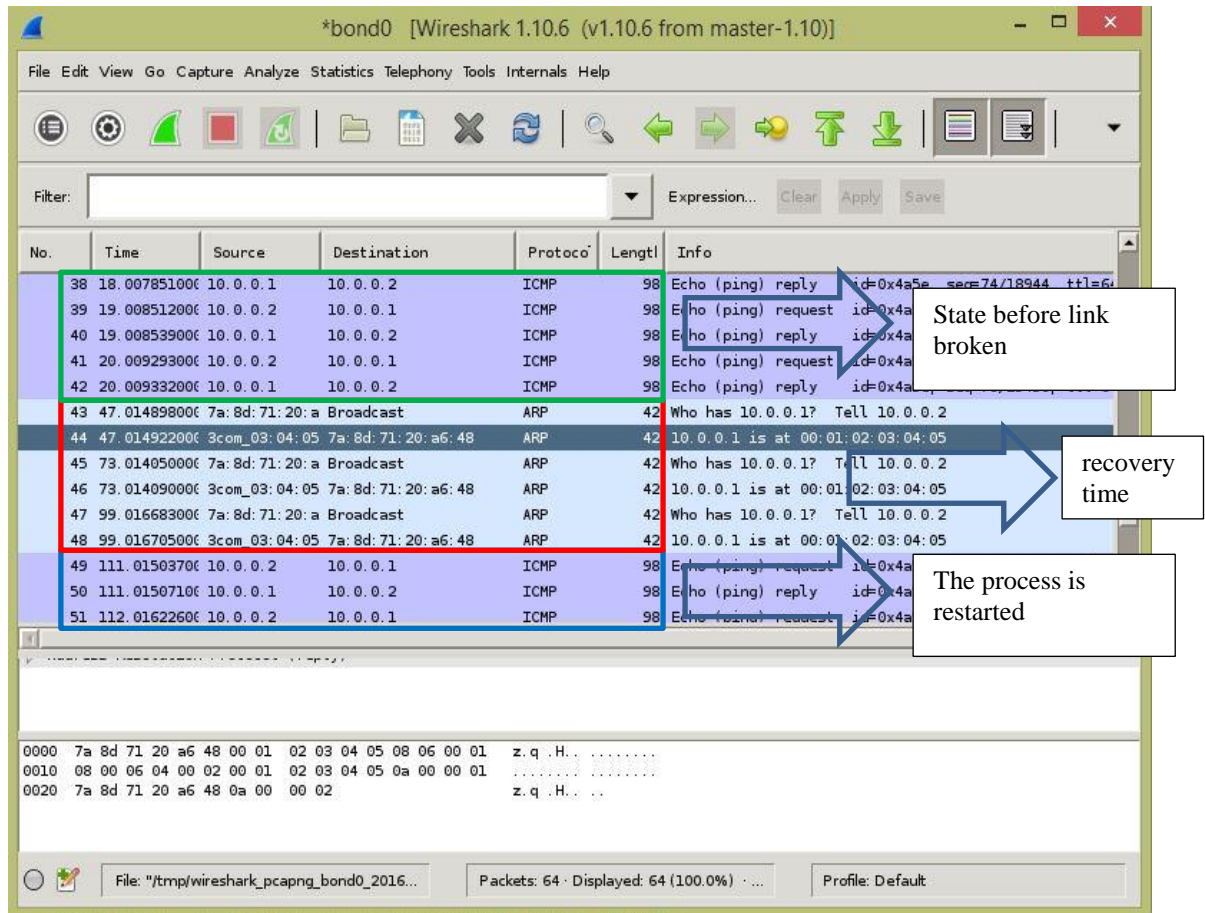


Figure 9. Ping state observation using Wireshark

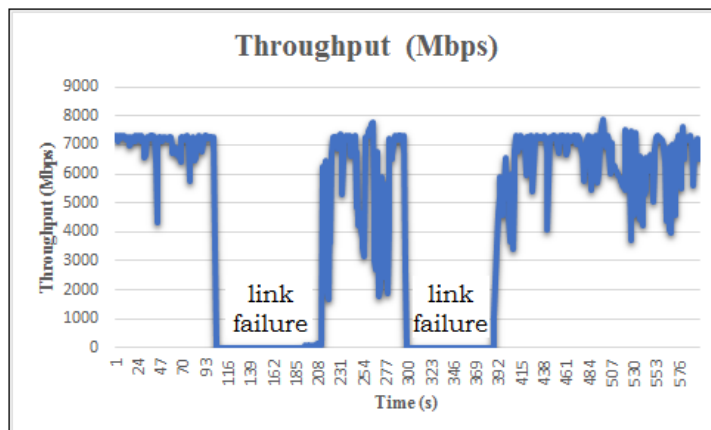


Figure 10. Throughput observation in link failure period

5. CONCLUSION

We implement link aggregation successfully using Ryu controller on SDN topology. LACP provides fault tolerance in the communication monitoring between nodes in the network. The recovery time is about 90 seconds. This procedure can apply for all connection entire the network. For improving the performance, we can set the input parameter of LACP according to the network requirements.

REFERENCES

- [1] *IEEE Standard for Ethernet Link Aggregation: IEEE802.3ad*, available: <http://www.ieee802.org/3/ad/>, accessed at: June 13, 2016, 2016-01-10.
- [2] L. Junyi, L. Lin, Z. Yongxin, "Direct Radio Frequency Sampling System on Software Defined Radio," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 12, no. 11, pp. 7824-7831, 2014.
- [3] K. Greene. *TR10: Software-Defined Networking*. MIT Technology Review. 2009; March–April.
- [4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner. *OpenFlow: Enabling Innovation in Campus Networks*. ACM SIGCOMM Computer Communication Review. 2008; 38(2): 69-74.
- [5] M. Steinbacher and M. Bredel. *LACP Meets OpenFlow – Seamless Link Aggregation to OpenFlow Networks*. Available: <https://tnc15.terena.org/getfile/1867>. Accessed at: January 13, 2016.
- [6] I.D. Irawati, S. Hadiyoso and Y. S. Hariyani. Link Redundancy for High Availability Network based on Software Defined Network. *Pertanika Journal of Science & Technology*. (under review)
- [7] P. Skoldstrom and B. C. Sanchez. *Virtual Aggregation using SDN*. IEEE Proceeding: European Workshop on Software Defined Networks. 2013.
- [8] W. Yahya, A. Basuki and J.R. Jiang. The Extended Dijkstra's-based Load Balancing for OpenFlow Network. *International Journal of Electrical and Computer Engineering (IJECE)*. 2015; 5(2): 289-296.
- [9] T. Jena and J. R. Mohanty, Disaster Recovery Services in Intercloud Using Genetic Algorithm Load Balancer. *International Journal of Electrical and Computer Engineering (IJECE)*. 2016; 6(4): 1828-1838.
- [10] Y. Zhou, L. Ruan, L. Xiao and R. Liu. *A Method for Load Balancing based on Software-Defined Network*. *Advanced Science and Technology Letters*. 2014; 45: 43-48.
- [11] D. Law, *IEEE Standard for Local and Metropolitan Area Network-Link Aggregation*. Available: <http://standards.ieee.org/findstds/standard/802.1AX-2008.html>. Accessed at: February 10, 2016.
- [12] R. P. Team. *RYU SDN Framework*. Available: <https://osrg.github.io/ryu-book/en/Ryubook.pdf>. Accessed at: January 4, 2016.