

SPANNING TREE PROTOCOL SIMULATION BASED ON SOFTWARE DEFINED NETWORK USING MININET EMULATOR

Indrarini Dyah Irawati¹, Mohammad Nuruzzamanirridha¹

¹Telkom Applied Science School, Telkom University, Bandung, Indonesia
indrarini@telkomuniversity.ac.id, mohammadnuruz@gmail.com

Abstract. Software Defined Networking (SDN) is a new networking paradigm [1] that separates the control and forwarding planes in a network. This migration of control, formerly tightly bound in individual network devices, into accessible computing devices enables the underlying infrastructure to be abstracted for applications and network services, which can treat the network as a logical or virtual entity. There are three methods for implementing SDN based network architecture, which uses Mininet emulator, Net-FPGA, and OpenFlow-based S/W switch [2]. In this research, simulation of Spanning Tree Protocol (STP) based SDN using mininet emulator. SDN is performed by OpenvSwitch (OVS) as a forwarding function, Ryu as OpenFlow Controller, and Mininet that installed on Raspberry-Pi. Then see the effect of the use of STP at Ryu Controller on network performance. The results of this research show that the simulation of SDN with OVS and Ryu controller can successfully runs STP function and STP on Ryu controller prevents broadcast storm.

Keywords: Spanning Tree Protocol, Software Defined Network, OpenFlow, OpenvSwitch, Ryu Controller, Raspberry-Pi

1 Introduction

Nowadays, the development of computer network technology is growing rapidly. The growth will affect the increase of data traffic in the network. The traditional network architecture that integrates the forward plane and control plane into the same device cannot support these requirements. To solve this problem, the new network architecture method called Software Defined Network proposed. SDN is a new networking paradigm [1] that separates the control and forwarding planes in a network. The network can be dynamically managed depending on networking policies [3].

Mininet is a standard Linux computer network emulator that can create virtual topologies such as virtual hosts, switches, controllers, and link and support the OpenFlow protocol that can be used for computer networks based SDN simulation. Mininet is suitable for the purposes of research, development, learning, prototyping, testing, debugging, and some other functions just by using PC or laptop without any additional devices.

In this study, used mininet emulator to simulate a network topology SDN. SDN network consists of the forwarding plane in the form of OVS and control plane in form of Ryu Controller. Ryu Controller functions is as STP.

2 Theory

2.1 OpenFlow

OpenFlow protocol provides standards for routing and delivery of packets on a switch. In the conventional network architecture, switch only works through packet forwarding to the appropriate port without being able to distinguish the type of protocol data transmitted. OpenFlow can perform the function of flow forwarding based network layer and manage centrally packet flow from layer 2 to layer 7 (flow granularity), so that the flow of packets in the network can be set up and configured independently. This can be done by making the algorithm and its forwarding rules in the controller which distributed to the switches on the network. OpenFlow architecture is shown in figure 1 that consists of OpenFlow Controller and OpenFlow Switch. OpenFlow Controller functions are controlled the path and managed OpenFlow switch. The example of OpenFlow controller are NOX, POX, Floodlight, OpenDaylight and Ryu. OpenFlow switch as forwarding plane can implement by using hardware OpenFlow switch, Ethernet card that support FPGA, Mininet OpenFlow Switch Emulator and OpenFlow *S/W based* (example : OpenvSwitch that installed on Linux OS).

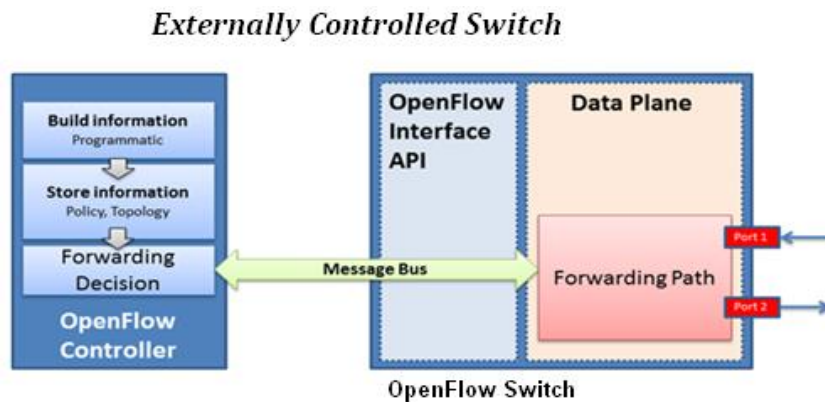


Fig. 1. OpenFlow architecture (Source: bradhedlund.com) [4]

2.2 Ryu Controller [5]

Ryu is a component-based software defined networking framework that shown in figure 2. Ryu provides software components with well defined Application Programming Interface (API) that make it easy for developers to create new network management and control applications. Ryu supports various protocols for managing network devices, such as OpenFlow, Netconf, OF-config, etc. About OpenFlow, Ryu supports fully 1.0, 1.2, 1.3, 1.4. All of the code is freely available under the Apache 2.0 license.

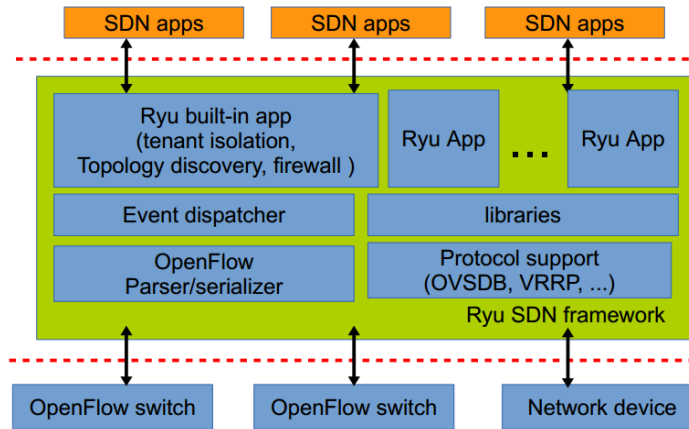


Fig. 2. Ryu SDN framework [6]

2.3 Spanning Tree Protocol (STP)

Spanning Tree Protocol is layer 2 protocol that defined by IEEE 802.1D standard. Spanning-Tree Protocol is a link management protocol that provides path redundancy while preventing undesirable loops in the network [7,8]. STP provides backup links between bridges and switches. The need for STP is based due to a broadcast storm. The broadcast storm caused a broadcast frame circling endlessly in the network due to the destination address in an unknown network. STP ensures only one path exists between anytwo stations.

3 Simulation Design

3.1 Network Design

In this research, using Raspberry Pi model B as a device that can be used as controller. Raspberry pi model B has specification with 512 MB of RAM, two USB port and 100 Mbps Ethernet port [9]. Design of network topology on this simulation is using mininet that installed on Raspberry Pi. The simulation is remote by secure socket shell (SSH) protocol on personal computer (PC).

The network simulation consists of 3 pieces of switches are connected in mesh topology. Each switch is connected with a host. Network design is shown in figure 3.

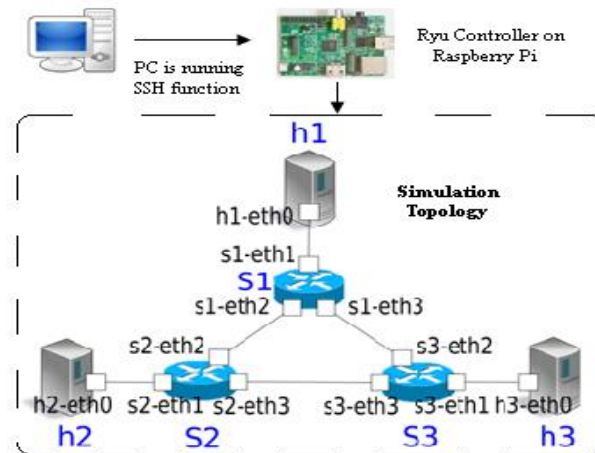


Fig. 3. Network topology

3.2 SDN Controller

The SDN controller in this simulation uses Ryu controller software that installed in the raspberry pi with the Debian Wheezy version 3.12 Linux operating system. Ryu controller acts handle Spanning Tree Protocol (STP) on the network. Ryu SDN framework is programed with python programming language.

4 Result

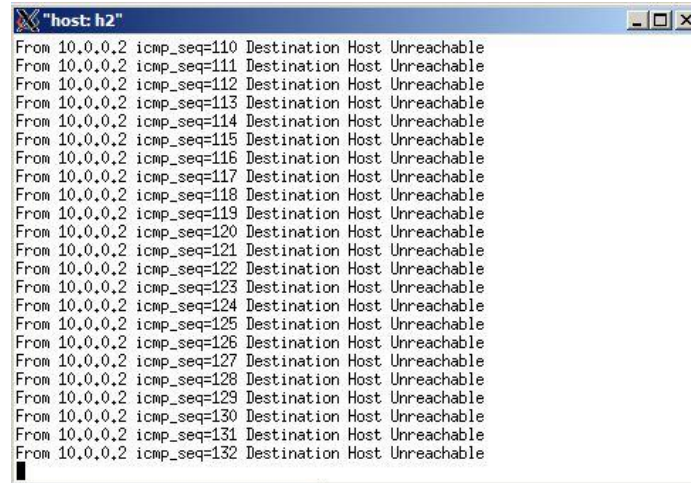
There are two scenarios to evaluate the performance of Ryu controller. The first scenario is without using STP configuration on Ryu Controller and the second scenario is using the STP configuration.

To check the connection on the network is performed Packet Internet Grouper (PING) test from host 2 with IP Address 10.0.0.2/8 to host 1 with IP Address IP 10.0.0.2/8. PING operates by sending Internet Control Message Protocol (ICMP) echo request packets to the target host and waiting for an ICMP response. The ping message is used to reach the end-host.

4.1 SDN without STP Configuration

Based on the first scenario test, SDN without using the STP configuration, then the results are as follows :

- Broadcast storm occurs on the network that caused the ICMP packet is sent from the host with IP address 10.0.0.2 2 to host 1 with the IP address 10.0.0.1 is not sent (the status of host unreachable) as shown on figure 4. The figure is captured from host 2. The ICMP packet is cycling on the network.



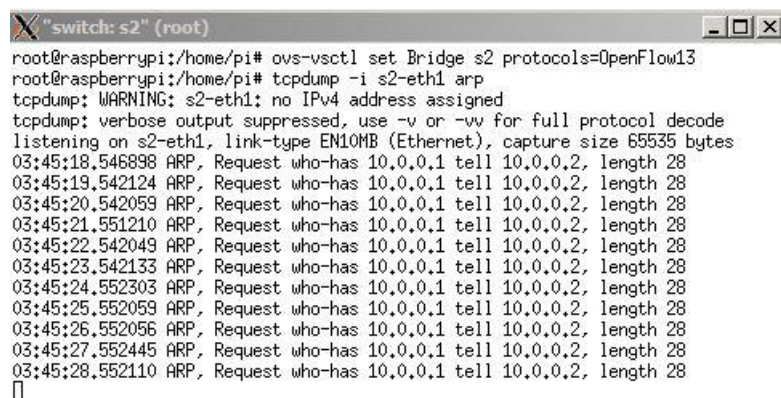
```

"host: h2"
From 10.0.0.2 icmp_seq=110 Destination Host Unreachable
From 10.0.0.2 icmp_seq=111 Destination Host Unreachable
From 10.0.0.2 icmp_seq=112 Destination Host Unreachable
From 10.0.0.2 icmp_seq=113 Destination Host Unreachable
From 10.0.0.2 icmp_seq=114 Destination Host Unreachable
From 10.0.0.2 icmp_seq=115 Destination Host Unreachable
From 10.0.0.2 icmp_seq=116 Destination Host Unreachable
From 10.0.0.2 icmp_seq=117 Destination Host Unreachable
From 10.0.0.2 icmp_seq=118 Destination Host Unreachable
From 10.0.0.2 icmp_seq=119 Destination Host Unreachable
From 10.0.0.2 icmp_seq=120 Destination Host Unreachable
From 10.0.0.2 icmp_seq=121 Destination Host Unreachable
From 10.0.0.2 icmp_seq=122 Destination Host Unreachable
From 10.0.0.2 icmp_seq=123 Destination Host Unreachable
From 10.0.0.2 icmp_seq=124 Destination Host Unreachable
From 10.0.0.2 icmp_seq=125 Destination Host Unreachable
From 10.0.0.2 icmp_seq=126 Destination Host Unreachable
From 10.0.0.2 icmp_seq=127 Destination Host Unreachable
From 10.0.0.2 icmp_seq=128 Destination Host Unreachable
From 10.0.0.2 icmp_seq=129 Destination Host Unreachable
From 10.0.0.2 icmp_seq=130 Destination Host Unreachable
From 10.0.0.2 icmp_seq=131 Destination Host Unreachable
From 10.0.0.2 icmp_seq=132 Destination Host Unreachable

```

Fig. 4. Ping message from host 2 to host 1

- Results tcpdump on Switch 2 - eth1, there is only Address Resolution Protocol (ARP) Request seen repeatedly without ARP Reply as shown on figure 5. The send ARP function sends ARP request to obtain the physical address that corresponds to the specified destination IP Address (10.0.0.1).
- The network configuration is consisting of three switch (Switch 1, Switch2, Switch3), and three network segment (Switch 1, Switch2, Switch3). While Host 2 (IP address 10.0.0.2) transmits data to Host 1 (IP Address 10.0.0.1) then the traffic is broadcast from Switch 2 over to Segment 1; and it fails, then Switch 2 also broadcast traffic over segment 3. These switches would further create a flood. Hence, all switches keep sending and resending the traffic, eventually resulting in a flood loop or broadcast loop. Finally the network melts down, causing failure in all network links, which is referred to as a broadcast storm.



```

switch: s2" (root)
root@raspberrypi:/home/pi# ovs-vsctl set Bridge s2 protocols=OpenFlow13
root@raspberrypi:/home/pi# tcpdump -i s2-eth1 arp
tcpdump: WARNING: s2-eth1: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on s2-eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
03:45:18.546898 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
03:45:19.542124 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
03:45:20.542059 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
03:45:21.551210 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
03:45:22.542049 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
03:45:23.542133 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
03:45:24.552303 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
03:45:25.552059 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
03:45:26.552056 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
03:45:27.552445 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
03:45:28.552110 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28

```

Fig. 5. tcpdump on S2-eth1

- Capture using wireshark as shown figure 6, only ARP Request seen repeatedly with information 'Who has 10.0.0.1 ? Tell 10.0.0.2, without an ARP Reply. That means the switch cannot find the destination address on the network.

Destination	Protocol	Length	Info
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2

Fig. 6. Capture result from Wireshark

4.2 SDN with STP Configuration

- STP can be run by selecting the switch port state. When STP is enabling, the switch port starts from BLOCK state, and later changes to LISTEN state and LEARN state. BLOCK state is condition that the port does not participate in frame forwarding and discards frame received from the attached network segment. During the LISTEN state the port discards frames received from the attached network segment and it also discards frames switched from another port for forwarding. In the LEARN state , the port begin to process frame and starts update MAC Address table. FORWARD state is normal state that frame passed through the port. The port state can seen on the SDN controller as shown on figure 7 .

```

"controller: c0" (root)
[STP][INFO] dpid=0000000000000003: [port=3] DESIGNATED_PORT / BLOCK
[STP][INFO] dpid=0000000000000003: Non root bridge.
[STP][INFO] dpid=0000000000000003: [port=1] DESIGNATED_PORT / LISTEN
[STP][INFO] dpid=0000000000000003: [port=2] NON_DESIGNATED_PORT / LISTEN
[STP][INFO] dpid=0000000000000003: [port=3] ROOT_PORT / LISTEN
[STP][INFO] dpid=0000000000000001: [port=1] DESIGNATED_PORT / LEARN
[STP][INFO] dpid=0000000000000001: [port=2] DESIGNATED_PORT / LEARN
[STP][INFO] dpid=0000000000000001: [port=3] DESIGNATED_PORT / LEARN
[STP][INFO] dpid=0000000000000002: [port=1] DESIGNATED_PORT / LEARN
[STP][INFO] dpid=0000000000000002: [port=2] ROOT_PORT / LEARN
[STP][INFO] dpid=0000000000000002: [port=3] DESIGNATED_PORT / LEARN
[STP][INFO] dpid=0000000000000003: [port=1] DESIGNATED_PORT / LEARN
[STP][INFO] dpid=0000000000000003: [port=2] NON_DESIGNATED_PORT / LEARN
[STP][INFO] dpid=0000000000000003: [port=3] ROOT_PORT / LEARN
[STP][INFO] dpid=0000000000000001: [port=1] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000001: [port=2] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000001: [port=3] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000002: [port=1] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000002: [port=2] ROOT_PORT / FORWARD
[STP][INFO] dpid=0000000000000002: [port=3] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000003: [port=1] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000003: [port=2] NON_DESIGNATED_PORT / BLOCK
[STP][INFO] dpid=0000000000000003: [port=3] ROOT_PORT / FORWARD
    
```

Fig.7. Running STP on Ryu Controller

- The results of the ping message from host 2 IP address 10.0.0.2/8 to host 1 with the IP address 10.0.0.1/8 shown on figure 8. There is ICMP reply from host 2. The network still connected.

```

"host: h2"
root@raspberrypi:/home/pi# ping -c4 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_req=1 ttl=64 time=366 ms
64 bytes from 10.0.0.1: icmp_req=2 ttl=64 time=1.58 ms
64 bytes from 10.0.0.1: icmp_req=3 ttl=64 time=0.290 ms
64 bytes from 10.0.0.1: icmp_req=4 ttl=64 time=0.280 ms

--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.280/92.156/366.474/158.378 ms
root@raspberrypi:/home/pi#

```

Fig. 8. Ping message from host 2 to host 1

- ARP Request and ARP Reply process is happened on Ryu Controller as shown on Figure 9. Ryu controller can handle the connection on the network and runs STP function. Capture from wireshark is shown on figure 10.

```

"controller: c0" (root)
[STP][INFO] dpid=0000000000000001: [port=3] DESIGNATED_PORT / LEARN
[STP][INFO] dpid=0000000000000002: [port=1] DESIGNATED_PORT / LEARN
[STP][INFO] dpid=0000000000000002: [port=2] ROOT_PORT / LEARN
[STP][INFO] dpid=0000000000000002: [port=3] DESIGNATED_PORT / LEARN
[STP][INFO] dpid=0000000000000003: [port=1] DESIGNATED_PORT / LEARN
[STP][INFO] dpid=0000000000000003: [port=2] NON_DESIGNATED_PORT / LEARN
[STP][INFO] dpid=0000000000000003: [port=3] ROOT_PORT / LEARN
[STP][INFO] dpid=0000000000000001: [port=1] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000001: [port=2] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000001: [port=3] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000002: [port=1] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000002: [port=2] ROOT_PORT / FORWARD
[STP][INFO] dpid=0000000000000002: [port=3] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000003: [port=1] DESIGNATED_PORT / FORWARD
[STP][INFO] dpid=0000000000000003: [port=2] NON_DESIGNATED_PORT / BLOCK
[STP][INFO] dpid=0000000000000003: [port=3] ROOT_PORT / FORWARD
packet in 2 c2:73:9a:af:9d:f9 ff:ff:ff:ff:ff:ff 1
packet in 1 c2:73:9a:af:9d:f9 ff:ff:ff:ff:ff:ff 2
packet in 1 6a:de:a2:0c:9d:47 c2:73:9a:af:9d:f9 1
packet in 3 c2:73:9a:af:9d:f9 ff:ff:ff:ff:ff:ff 3
packet in 2 6a:de:a2:0c:9d:47 c2:73:9a:af:9d:f9 2
packet in 2 c2:73:9a:af:9d:f9 6a:de:a2:0c:9d:47 1
packet in 1 c2:73:9a:af:9d:f9 6a:de:a2:0c:9d:47 2

```

Fig. 9. ARP Request and ARP Reply on Ryu Controller

stpcapture [Wireshark 1.8.2]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

	Destination	Protocol	Length	Info
:75	LLDP_Multicast	STP	52	Conf. Root = 32768/0/96:9d:97:2b:9f:38 Cost =
	10.0.0.1	ICMP	98	Echo (ping) request id=0x6547, seq=2/512, ttl=
	10.0.0.2	ICMP	98	Echo (ping) reply id=0x6547, seq=2/512, ttl=
	10.0.0.1	ICMP	98	Echo (ping) request id=0x6547, seq=3/768, ttl=
	10.0.0.2	ICMP	98	Echo (ping) reply id=0x6547, seq=3/768, ttl=
:75	LLDP_Multicast	STP	52	Conf. Root = 32768/0/96:9d:97:2b:9f:38 Cost =
	10.0.0.1	ICMP	98	Echo (ping) request id=0x6547, seq=4/1024, ttl=
	10.0.0.2	ICMP	98	Echo (ping) reply id=0x6547, seq=4/1024, ttl=
:75	LLDP_Multicast	STP	52	Conf. Root = 32768/0/96:9d:97:2b:9f:38 Cost =
l:47	c2:73:9a:af:9d:f9	ARP	42	Who has 10.0.0.2? Tell 10.0.0.1
l:f9	6a:de:a2:0c:9d:47	ARP	42	10.0.0.2 is at c2:73:9a:af:9d:f9
:75	LLDP_Multicast	STP	52	Conf. Root = 32768/0/96:9d:97:2b:9f:38 Cost =
:75	LLDP_Multicast	STP	52	Conf. Root = 32768/0/96:9d:97:2b:9f:38 Cost =

Fig. 10. Capture result from Wireshark

5 Conclusion

In this paper, the simulation of SDN with OpenFlow Switch and Ryu controller can successfully runs STP function. Ryu controller can perform STP function. STP stops flooding and prevents broadcast storm on the network. For future works, we will deploy the proposed SDN functionalities as STP controller using OpenFlow-based S/W switch to compare the result from mininet emulator.

Reference

What is OpenFlow?, <http://archive.openflow.org/wp/learnmore/>

Kim, Hyunmin., Jaebeom Kim., Young-Bae Ko.: Developing a Cost-Effective OpenFlow Testbed for Small-Scale Software Defined Networking: ICACT2014, pp. 758 - 761 Korea (2014)

S. Sezer, S. Scott-hayward, P.K Fraser, D. Lake, J. Finnegan, N. Vijoen, M. Miller, and N. Rao., Are we ready for SDN? Implementation challenges for software-defined networks: Communications Magazine, IEEE vol. 51(7), 2013.

On data center scale, OpenFlow, and SDN, <http://bradhedlund.com/2011/04/21/data-center-scale-openflow-sdn/>

What is Ryu?, <http://osrg.github.io/ryu/>

Yamahata, Isaku.: Ryu: SDN framework and Python experience: Pycon APAC 2013, Japan , September 2014

802.1D MAC Bridges, <http://www.ieee802.org/1/pages/802.1D-2003.html>

Spanning Tree Protocol STP Tutorial, <http://www.9tut.com/spanning-tree-protocol-stp-tutorial>

Raspberry Pi Model B, <http://www.raspberrypi.org/products/model-b/>