



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

THESIS NO. : 069/MSI/607

Broadcast Storm Prevention in Software Defined Network

by

Kabin Shrestha

A THESIS

**SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND
COMPUTER ENGINEERING IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN
INFORMATION AND COMMUNICATION ENGINEERING**

**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
LALITPUR, NEPAL**

OCTOBER, 2016

Broadcast Storm Prevention in Software Defined Network

by

Kabin Shrestha

069/MSI/607

Thesis Supervisor

Dr. Surendra Shrestha

A thesis submitted in partial fulfillment of the requirements for the Degree of Master
of Science in Information and Communication Engineering

Department of Electronics and Computer Engineering

Institute

of Engineering, Pulchowk Campus

Tribhuvan University

Lalitpur, Nepal

October, 2016

COPYRIGHT©

The author has agreed that the library, Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk Campus, may make this thesis freely available for inspection. Moreover the author has agreed that the permission for extensive copying of this thesis work for scholarly purpose may be granted by the professor(s), who supervised the thesis work recorded herein or, in their absence, by the Head of the Department, wherein this thesis was done. It is understood that the recognition will be given to the author of this thesis and to the Department of Electronics and Computer Engineering, Pulchowk Campus in any use of the material of this thesis. Copying of publication or other use of this thesis for financial gain without approval of the Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk Campus and author's written permission is prohibited.

Request for permission to copy or to make any use of the material in this thesis in whole or part should be addressed to:

Head

Department of Electronics and Computer Engineering

Pulchowk Campus, Institute of Engineering

Pulchowk, Lalitpur

Nepal


TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

APPROVAL PAGE

The undersigned certify that they have read and recommended to the Department of Electronics and Computer Engineering for acceptance, a thesis entitled “**Broadcast Storm Prevention in Software Defined Network**”, submitted by **Kabin Shrestha** in partial fulfillment of the requirement for the award of the degree of “**Master of Science in Information and Communication Engineering**”.

Defense Date: 25th October, 2016

Examination Committee

Professor Dr. Shashidhar Ram Joshi
Chairperson



Professor Dr. Dinesh Kumar Sharma
Member

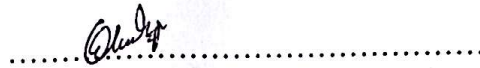


Reader Dr. Surendra Shrestha
M.Sc. Coordinator, Supervisor



Department of Electronics and Computer Engineering
Pulchowk Campus, Institute of Engineering, Tribhuvan University

Er. Om Bikram Thapa
External Examiner
Chief Technical Officer
Vianet Communications Pvt. Ltd.
Jawalakhel, Lalitpur





TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS



DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

Ananda Niketan, Pulchowk, Lalitpur, P.O. Box 1175, Kathmandu, Nepal.

Tel : 5534070, 5521260 extn. 315, Fax : 977-1-5553946, E-mail - doece@ioe.edu.np

Our Ref :

DEPARTMENTAL ACCEPTANCE

The thesis entitled "Broadcast Storm Prevention in Software Defined Network", submitted by Kabin Shrestha, in partial fulfillment of the requirement for the award of the degree of "Master of Science in Information and Communication Engineering" has been accepted as a bonafide record of work independently carried out by him in the department.

Dr. Dibakar Raj Pant
Head of the Department
Department of Electronics and Computer Engineering
Pulchowk Campus, Institute of Engineering
Tribhuvan University, Nepal

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to the Department of Electronics and Computer Engineering, Institute of Engineering for accepting my thesis entitled “Broadcast Storm Prevention in Software Defined Network”. I would like to extend my sincere thanks for providing me with all the essential co-operation, valuable suggestions for choosing the thesis topic.

I am grateful to my supervisor Reader Dr. Surendra Shrestha for providing useful information and guidance regarding this thesis. I am also grateful to Professor Dr. Subarna Shakya and Dr. Dibakar Raj Pant for their valuable recommendations during the thesis defense. I am also indebted to Er. Krishna Prasad Bhandari, Deputy Manager, Nepal Telecom and Er. Om Bikram Thapa, Chief Technical Officer, Vianet Communications Pvt. Ltd. for providing some valuable suggestions.

Last but not the least I would like to express my heartfelt thanks to respected teachers, my families and friends who have helped and supported me directly and indirectly during the thesis.

ABSTRACT

Software-Defined Networking (SDN) is a new principle in the networking paradigm, which makes a communication network programmable. In SDN, control and management are centralized and decoupled from data plane, thus making the network programmable. In SDN, for a single change in network, the network configurations are changed only at central or some specific controller(s) rather than touching individual network devices. A Local Area Network (LAN) is prone to Layer-2 Broadcast Storm and an early safety measure must be taken to ensure the broadcast storm does not take down the whole network. A looped network topology in a LAN is basically what is needed for the broadcast storm to strike. So in order to prevent the undesired creation of loop network, Spanning Tree Protocol (STP) has been already in use in traditional networking infrastructures. This thesis presents the application of STP in SDN as a loop prevention mechanism. The network simulation of the looped topology is performed in mininet (Linux based emulator) and with Ryu (SDN controller). A SDN controller application specifically for the loop prevention is developed and tested.

Keywords: Software Defined Networking, Loop Avoidance, Rapid Spanning Tree Protocol, Layer-2 broadcast storm.

Table of Contents

COPYRIGHT©.....	III
APPROVAL PAGE	IV
DEPARTMENTAL ACCEPTANCE.....	V
ACKNOWLEDGEMENT	VI
ABSTRACT.....	VII
TABLE OF CONTENTS.....	1
LIST OF TABLES	3
LIST OF FIGURES	4
LIST OF ABBREVIATIONS.....	5
CHAPTER ONE: INTRODUCTION.....	6
1.1. RELATED THEORY	8
1.1.1. Broadcast Storm	8
1.1.2. Spanning Tree Protocol.....	10
1.1.3. RSTP Overview.....	10
1.1.4. RSTP Port States.....	11
1.1.5. RSTP Port Roles	12
1.1.6. Spanning Tree Packets	13
1.1.7. BPDU format	13
1.1.8. Choosing a Root Bridge.....	15
1.1.9. Choosing the Least-Cost Path and port role selection.....	17
1.1.10. Blocking Loop Paths.....	19
1.2. PROBLEM STATEMENT	19
1.3. OBJECTIVE.....	19
CHAPTER TWO: LITERATURE REVIEW	20
2.1. SOFTWARE DEFINED NETWORKING.....	20
2.2. LIMITATIONS IN TRADITIONAL NETWORKING CONCEPT.....	22
2.3. SDN ARCHITECTURE	23

2.4. OPENFLOW	24
2.5. RYU CONTROLLER	25
CHAPTER THREE: METHODOLOGY	26
CHAPTER FOUR: SIMULATION PROCESS	29
CHAPTER FIVE: RESULTS AND DISCUSSIONS.....	31
5.1. SDN WITHOUT RSTP CONTROLLER	32
5.2. SDN WITH RSTP CONTROLLER	35
5.3. ANALYSIS OF THE CONTROLLER.....	40
CHAPTER SIX: LIMITATIONS AND RECOMMENDATIONS.....	42
CHAPTER SEVEN: CONCLUSION	43
REFERENCES	44

List of Tables

Table 1.1	RST BPDU parameters and format	14
Table 1.2	RST BPDU parameters and their description.....	15
Table 1.3	Port Path Cost values.....	17
Table 3.1	Simulation Tools.....	28
Table 4.1	Network details.....	29
Table 5.1	Packets to and from Controller.....	40
Table 5.2	Variation in packets to and from controller with switches....	40

List of Figures

Figure 1.1	Ring topology of three switches.....	9
Figure 1.2	Alternate Port going to forward state when Root Port goes down.....	13
Figure 1.3	Root Bridge Selection.....	16
Figure 1.4	Different port roles assigned by RSTP.....	17
Figure 2.1	Traditional network compared to SDN network.....	22
Figure 2.2	Software Defined Networking Architecture.....	24
Figure 3.1	Network Topology.....	27
Figure 3.2	Spanning Tree with port roles.....	27
Figure 4.1	Simulated Network in mininet with interface numbers.....	30
Figure 5.1	Running Switched network in mininet with empty flow-table shown.....	31
Figure 5.2	Ping request from host h1 to h2 under no RSTP controller running.....	32
Figure 5.3	Packets flow in the looped network.....	32
Figure 5.4	ARP message Broadcast storm as observed in Wireshark...	34
Figure 5.5	Running Ryu controller.....	35
Figure 5.6	RST BPDU packet captured by Wireshark.....	36
Figure 5.7	Final Port States in switches.....	37
Figure 5.8	Ping reply from host h2 under RSTP controller running.....	38
Figure 5.9	ARP request and reply.....	38
Figure 5.10	Traffic flow in interface 1 of switch s1 after RSTP controller running.....	39
Figure 5.11	Controller Packets vs. No of switches.....	41

List of Abbreviations

Abbreviations	Full Form
API	Application Programming Interface
ARP	Address Resolution Protocol
BID	Bridge Identification
BPDU	Bridge Protocol Data Unit
CLI	Command Line Interface
DB	Designated Bridge
DP	Designated Port
LAN	Local Area Network
MAC	Media Access Control
QoS	Quality of Service
RP	Root Port
RST BPDU	Rapid Spanning Tree BPDU
RSTP	Rapid Spanning Tree Protocol
SDN	Software Defined Networking
STP	Spanning Tree Protocol
TC	Topology Change
TCA	Topology Change Acknowledgement
TCP	Transmission Control Protocol
TTL	Time To Live

CHAPTER ONE: INTRODUCTION

A Local Area Network (LAN) is an interconnected network of IP devices under the same IP network. The fundamental method of communication between these devices is guided by the Layer-2 protocols and specifications of TCP/IP networking protocol stack. One of the major remarkable feature of this layer is that all the devices under this layer are in the same Broadcast Domain. So a broadcast message that is destined to Broadcast MAC address ff:ff:ff:ff:ff:ff is received and processed by every device in the same Broadcast Domain. In practical LAN, redundant links are created to avoid complete network failure in an event of failure in one link. But in such scenarios, Broadcast message can go round the loop until the network capacity is saturated. This is referred to as Broadcast Storm. It is the function of a network switch to prevent layer-2 switching loops and broadcast storms. The IEEE 802.1D-2004 bridging standard provides a rapid spanning tree protocol to avoid this problem by automatically suppressing layer 2 switching loops [1].

In every LAN network, a switch functions on the basic of two functions – learning and forwarding. Learning is the process of obtaining the MAC address of connected devices. When a frame reaches into the port of a switch, the switch reads the MAC address of the source device from Ethernet frame and compares it to its MAC address table. If the switch cannot find a corresponding entry in MAC address table, the switch will add the address to the table with the port number via the Ethernet frame arrived. If the MAC address is already available in the MAC address table, the switch compares the incoming port with the port already available in the MAC table. If the port numbers are different, the switch updates the MAC address table new port number. Forwarding is the process of passing network traffic a switch receives from one port of the switch to another port on the switch. When a Layer 2 Ethernet frame reaches a port on the switch, it reads the source MAC address of the Ethernet frame as a part of learning function, and it also reads the destination MAC address also as a part of forwarding function. The destination MAC address is important to determine the port number which the destination device is connected. If the destination MAC address is found on

the MAC address table, the switch forwards the Ethernet frame via the corresponding port of the MAC address. If the destination MAC address is not found on the MAC address table, the switch generates an Address Resolution Protocol (ARP) based Ethernet frame to find the destined host and forwards the frame through all its ports except the source port. This a Broadcast message as it is destined for all the hosts in the LAN, but only the destined host with the specific MAC address reply back as prescribed by ARP protocol. The destined host then reply backs to ARP frame and thus the switch now knows the destination port (learning) to send the Ethernet frame for this destined host (forwarding).

Every networking device initially is unaware of the other devices in the network. So ARP message which is a Broadcast traffic is required initially for the communication to establish. But in case there is a network loop due to redundant links, the same ARP message can go round the loop saturating the bandwidth of the network as well as loading the processor of the switches. To allow switches to automatically create a loop-free set of paths, even in a complex network with multiple paths connecting multiple switches Rapid Spanning Tree Protocol defined by IEEE 802.1D is implemented in the LAN [1]. It provides the ability to dynamically create a tree topology in a network by blocking any packet forwarding on certain ports, and ensures that a set of switches can automatically configure themselves to produce loop-free paths. The IEEE 802.1D standard describes the operation of spanning tree, and every switch that claims compliance with the 802.1D standard must include spanning tree capability. Based on the tree topology created by Spanning Tree Protocol, real data traffic flows through the network.

In traditional switch, the Spanning Tree Protocol runs individually on each switching device in the network on the control plane in the device. The control plane is one of the three logical planes in networking infrastructure of a switch and is responsible for device to device signaling traffic flow to determine actual forwarding policies of data in data plane. In the new concept, Software Defined Networking (SDN) device will have the control plane removed from network hardware and implemented it in software

instead, which enables programmatic access and, as a result, makes network administration much more flexible [2].

Moving the control plane to software allows dynamic access and administration. A network administrator can shape traffic from a centralized control console without having to touch individual switches. The administrator can change any network switch's rules when necessary - prioritizing, de-prioritizing or even blocking specific types of packets with a very granular level of control.

This thesis presents the application of Rapid Spanning Tree Protocol in Software Defined Networking as a loop prevention mechanism. The network simulation of the looped topology has been carried out in mininet (Linux based emulator) and with Ryu (SDN controller) based on OpenFlow v1.3 protocol of SDN.

1.1. Related Theory

1.1.1. Broadcast Storm

Ethernet network traffic is classified as unicast, multicast, or broadcast traffic. Unicast traffic refers to a message being sent directly from one Ethernet device to another Ethernet device on the network. Multicast traffic refers to when one Ethernet device sends a message to a specific group of Ethernet devices. Broadcast traffic refers to one Ethernet device sending a message to all other devices on the network. All three types of Ethernet traffic are common to any Ethernet network and are essential to the proper operation of the Ethernet network. However a problem can arise when excessive broadcast traffic results in a broadcast storm [3].

Broadcast traffic by itself is a normal part of Ethernet network communications. One example of Broadcast traffic is the Address Resolution Protocol (ARP) message. Ethernet devices use ARP messages to resolve their IP addresses on the network.

Broadcast storms happen when a network is saturated with a large volume of broadcast traffic. They can occur for a short duration or for an extended period of time. Broadcast storms consume precious resources from every Ethernet device on the network. This is

because every device needs to queue the broadcasts to be processed. Ethernet devices will also have to queue broadcast protocol messages that they do not support. In an Ethernet network, broadcast storms need to be reduced in order to minimize unnecessary CPU usage and to keep the network operating properly.

One of the major reason of occurrence of a broadcast storm is switching loop in the Ethernet wiring topology (i.e. two or more paths exist between end stations). For instance, when switches are interconnected for redundancy in a ring topology as shown in Figure 1.1 a broadcast originating from a device connected to any switch, can cause the circulation of broadcasts around the network and can saturate the network consuming all available bandwidth. The loop creates broadcast storms as broadcasts are forwarded by switches out every port, the switch or switches will repeatedly rebroadcast the broadcast messages flooding the network. Since the Layer 2 header does not support a time to live (TTL) value, if a frame is sent into a looped topology, it can loop forever.

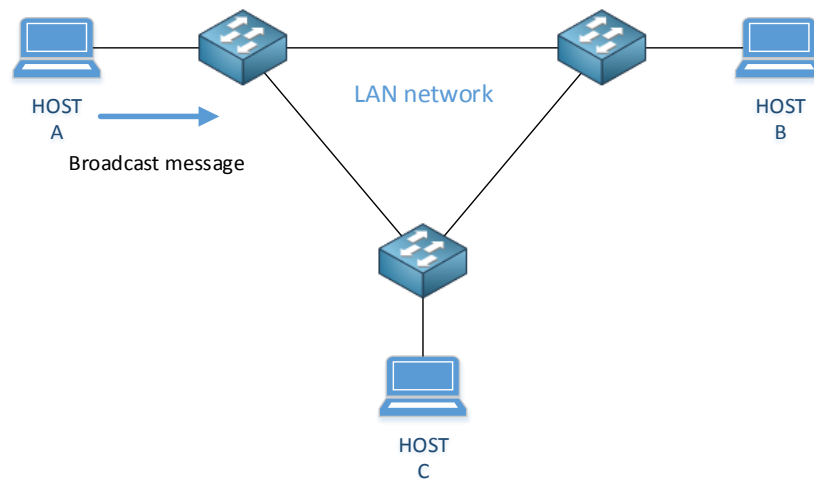


Figure 1.1: Ring topology of three switches

To prevent such loop condition and to retain redundancy in the network, dynamic loop avoidance mechanism guided by Spanning Tree Protocol is generally implemented that changes the loop topology to a linear tree structure spanning the whole networking devices.

1.1.2. Spanning Tree Protocol

The purpose of the spanning tree protocol (STP) is to allow switches to automatically create a loop-free set of paths, even in a complex network with multiple paths connecting multiple switches. Switches have many ports for interconnection to other devices. The ports can be electrical port (RJ 45 interface, serial interface, etc.) or optical port (Optical Module interface). When a number of switches are interconnected with redundant links creating loops, STP provides the ability to dynamically create a tree topology in a network by blocking any packet forwarding on certain ports, and ensures that a set of Ethernet switches can automatically configure themselves to produce loop-free paths. The IEEE 802.1D standard describes the operation of spanning tree, and every switch that claims compliance with the 802.1D standard must include spanning tree capability. IEEE Std 802.1D-2004 has incorporated Rapid Reconfiguration, which specified the Rapid Spanning Tree Algorithm and Protocol (RSTP). So, STP has now been superseded by the Rapid Spanning Tree Protocol (RSTP) specified in Clause 17 of IEEE Std 802.1D-2004 [1].

1.1.3. RSTP Overview

The Rapid Spanning Tree Protocol (RSTP) configures the Port State of each Bridge Port in the Bridge Local Area Network. A switch is referred to as Bridge in this protocol. RSTP ensures that temporary loops in the active topology do not occur if the network has to reconfigure in response to the failure, removal, or addition of a network component, and that erroneous station location information is removed from the Filtering Database after reconfiguration.

Each of the Bridges in the network transmits Configuration Messages. Each Configuration Message contains spanning tree priority vector information that identifies one Bridge as the Root Bridge of the network, and allows each Bridge to compute its own lowest path cost to that Root Bridge, information that will in turn be transmitted in Configuration Messages. A Port Role of Root Port is assigned to the one Port on each Bridge that provides that lowest cost path to the Root Bridge, and a Port Role of Designated Port to the one Port attached to each LAN that provides the lowest

cost path from that LAN to the Root Bridge. Port roles of Alternate Port and Backup Port are assigned to Bridge Ports that can provide connectivity if other network components fail.

RSTP provides rapid recovery of connectivity to minimize frame loss. A new Root Port, and Designated Ports attached to point-to-point LANs, can transition to Forwarding without waiting for protocol timers to expire. A Root Port can transition to Forwarding without transmitting or receiving messages from other Bridges, while a Designated Port attached to a point-to-point LAN can transition when it receives an explicit role agreement transmitted by the other Bridge attached to that LAN.

A Bridge Port attached to a LAN that has no other Bridges attached to it may be administratively configured as an Edge Port. RSTP monitors the LAN to ensure that no other Bridges are connected, and may be configured to automatically detect an Edge Port. Each Edge Port transitions directly to the Forwarding Port State, since there is no possibility of it participating in a loop.

1.1.4. RSTP Port States

Each Bridge Port has an operational Port State that governs whether or not it forwards MAC frames and whether or not it learns from their source addresses. RSTP has been reworked to classify 3 port states.

- Discarding
- Learning
- Forwarding

Discarding and Forwarding are the stable states, while Learning is the only transitory state.

Discarding State – This State basically means that the port is not forwarding frames nor is it receiving frames. Any frames that are transmitted to this port the MAC is not learned. A port in a discarding state will still continue to process BPDUs. This is also the default state for when a port first turns up.

Learning State – In this State, a port is not forwarding frames however the frames it is receiving it is learning the MAC addresses of those frames as they enter the interface.

Forwarding State – In this State, ports that are in this state are actively transmitting and receiving any and all frames passed through its interface.

The forwarding and learning performed by each Bridge Port is dynamically managed to prevent temporary loops and reduce excessive traffic in the network while minimizing denial of service following any change in the physical topology of the network.

1.1.5. RSTP Port Roles

Each port of a bridge in the network is assigned a specific role based on its position with respect to the Root Bridge. The ports can be either towards the Root Bridge or away from the Root Bridge. RSTP includes four port roles:

- Root Port
- Designated Port
- Alternate Port
- Backup port

Root port is a forwarding port that is the closest to the root bridge in terms of path cost. Hence it receives BPDU which is superior to all other BPDUs received on other remaining ports of the bridge.

Designated Port is the port that is forwarding the least cost superior BPDU onto other LAN segments.

Alternate Port is a best alternate path to the root bridge. It is considered as an alternate replacement for the switch's Root Port. The alternative port moves to the forwarding state if there is a failure on the designated port for the segment.

Backup Port is a port that is considered an alternate replacement for the switch's Designated port into a shared LAN segment. The backup port applies only when a

single switch has two links to the same segment (collision domain). To have two links to the same collision domain, the switch must be attached to a hub.

In the event a Root Port goes down, these Alternate ports can immediately takeover converging the network in less than one second as shown in Figure 1.2.

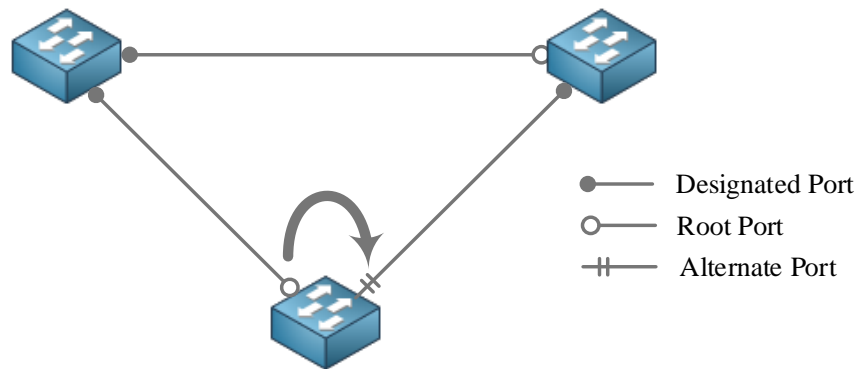


Figure 1.2: Alternate Port going to forward state when Root Port goes down

1.1.6. Spanning Tree Packets

Operation of the spanning tree algorithm is based on configuration messages sent by each switch in packets called Bridge Protocol Data Units, or BPDUs. Each BPDU packet is sent to a destination multicast address (Bridge Group Address 01:80:c2:00:00:00) that has been assigned to spanning tree operation. All IEEE 802.1D switches join the BPDU multicast group and listen to frames sent to this address, so that every switch can send and receive spanning tree configuration messages.

1.1.7. BPDU format

All BPDUs shall contain an integral number of octets. The octets in a BPDU are numbered starting from 1 and increasing in the order they are put into a Data Link Service Data Unit (DLSDU). The bits in an octet are numbered from 1 to 8, where 1 is the low-order bit.

Rapid Spanning Tree BPDUs (RST BPDUs)

The format of the RST BPDUs is shown in Table 1.1. The Protocol Identifier is encoded in Octets 1 and 2. It takes the value 0000 0000 0000 0000.

- a) The Protocol Version Identifier is encoded in Octet 3. It takes the value 0000 0010.
- b) The BPDU Type is encoded in Octet 4. This field takes the value 0000 0010. This denotes a Rapid Spanning Tree BPDU.
- c) The Topology Change flag is encoded in Bit 1 of Octet 5.
- d) The Proposal flag is encoded in Bit 2 of Octet 5.
- e) The Port Role is encoded in Bits 3 and 4 of Octet 5.
- f) The Learning flag is encoded in Bit 5 of Octet 5.
- g) The Forwarding flag is encoded in Bit 6 of Octet 5.
- h) The Agreement flag is encoded in Bit 7 of Octet 5.
- i) Topology Change Acknowledgment flag is encoded in Bit 8 of Octet 5 as zero.
- j) The Root Identifier is encoded in Octets 6 through 13.
- k) The Root Path Cost is encoded in Octets 14 through 17.
- l) The Bridge Identifier is encoded in Octets 18 through 25.
- m) The Port Identifier is encoded in Octets 26 and 27.
- n) The Message Age timer value is encoded in Octets 28 and 29.
- o) The Max Age timer value is encoded in Octets 30 and 31.
- p) The Hello Time timer value is encoded in Octets 32 and 33.
- q) The Forward Delay timer value is encoded in Octets 34 and 35.
- r) The Version 1 Length value is encoded in Octet 36. It takes the value 0000 0000, which indicates that there is no Version 1 protocol information present.

Table 1.1: RST BPDU parameters and format

Components	Octet	Value
Protocol Identifier	1-2	0000 0000 0000 0000
Protocol Version Identifier	3	0000 0010
BPDU Type	4	0000 0010
Flags	5	
Root Identifier	6-13	
Root Path Cost	14-17	
Bridge Identifier	18-25	
Port Identifier	26-27	
Message Age	28-29	
Max Age	30-31	
Hello Time	32-33	
Forward Delay	34-35	
Version 1 Length	36	0000 0000

Table 1.2: RST BPDU parameters and their description

Message Field	Description
Protocol Identifier	Contains the value zero
Flag	1 st bit for topology change flag 2 nd bit for proposal flag 3 rd and 4 th bit for port role (00:Unknown,01:Root,10:Designated:11:Alternate/Backup) 5 th bit for learning state 6 th bit for forwarding state 7 th bit for Agreement flag 8 th bit for Topology Change acknowledgement
Root ID	Identifies the root bridge by listing its priority and ID
Root Path Cost	Contains cost of the path from bridge sending BPDU to root bridge
Bridge ID	Identifies priority and ID of the bridge sending BPDU
Port ID	Identifies port from which BPDU was sent
Message Age	Specifies amount of time elapsed since root sent BPDU on which current configuration is based
Maximum Age	Indicates when the current configuration message should be detected. Recommended value = 20 sec
Hello Time	Provides time period between root bridge configuration messages. Recommended value = 2 sec
Forward Delay	Provides length of time that bridges should wait before transitioning to a new state after topology change. Recommended value = 15 sec

1.1.8. Choosing a Root Bridge

The process of creating a spanning tree begins by using the information in the RST BPDU messages to automatically elect a root bridge. The election is based on a bridge ID (BID) which, in turn, is based on the combination of a configurable bridge priority value (32,768 or 0x8000 by default) and the unique Ethernet MAC address assigned on each bridge for use by the spanning tree process, called the system MAC. Bridges send RST BPDUs to one another, and the bridge with the lowest BID is automatically elected to be the root bridge.

RSTP Bridges send information to each other, in Configuration Messages as shown in Table 1.1, to select a Root Bridge and the shortest path to it from each LAN and each of the other Bridges. The information sent for this purpose is known as a spanning tree priority vector.

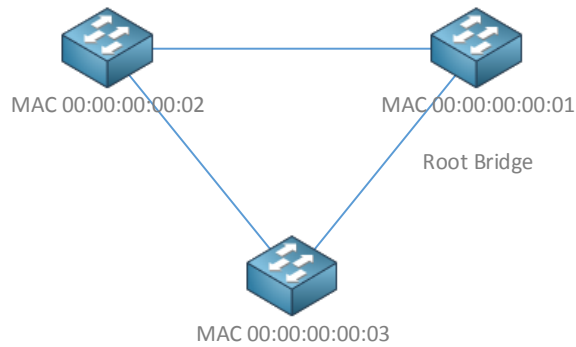


Figure 1.3: Root Bridge Selection

Spanning tree priority vectors provide the basis for a concise specification of RSTP's computation of the active topology. Each priority vector comprises:

- a) Root Bridge Identifier, the Bridge Identifier of the Bridge believed to be the Root by the transmitter
- b) Root Path Cost, to that Root Bridge from the transmitting Bridge
- c) Bridge Identifier, of the transmitting Bridge
- d) Port Identifier, of the Port through which the message was transmitted
- e) Port Identifier, of the Port through which the message was received (where relevant)

For all components, a lesser numerical value is better, and earlier components in the above list are more significant. The selection of Root Bridge and the shortest path to it from each LAN is computed from the received priority vector in the Configuration Message in a sequence:

- a) Root Bridge is the one with least Root Bridge Identifier
- b) If Root Bridge Identifier is same, choose Root Bridge from Root Path Cost
- c) If Root Path Cost is equal, choose Root Bridge from Bridge Identifier of transmitting Bridge
- d) If Bridge Identifier is same, choose Root Bridge from Port Identifier of transmitting Bridge
- e) If Port Identifier is same, choose Root Bridge from Port Identifier of the receiving Bridge

1.1.9. Choosing the Least-Cost Path and port role selection

Once a root bridge is chosen, each non-root bridge uses that information to determine which of its ports has the least-cost path to the root bridge, then assigns that port to be the root port (RP) as shown in Figure 1.4. Every Bridge has a Root Path Cost associated with it. For the Root Bridge this is zero. For all other Bridges, it is the sum of the Port Path Costs on the least cost path to the Root Bridge. Each Port's Path Cost may be managed, Table 1.3 recommends default values for Ports attached to LANs of various speeds. All other bridges determine which of their ports connected to other links has the least-cost path to the root bridge. The bridge with the least-cost path is assigned the role of designated bridge (DB), and the ports on the DB are assigned as designated ports (DP).

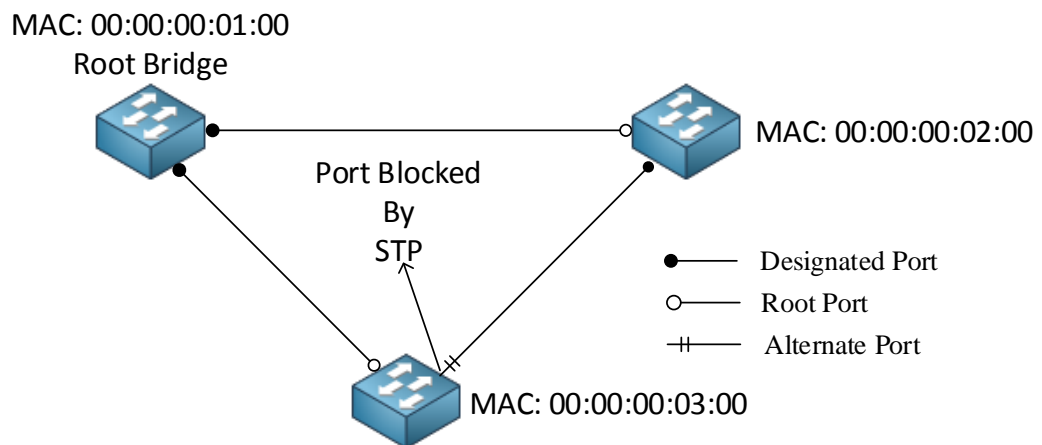


Figure 1.4: Different port roles assigned by RSTP

As BPDU packets travel through the system, they accumulate information about the number of ports they travel through and the speed of each port. The total cost of a given path through multiple switches is the sum of the costs of all the ports on that path.

Table 1.3: Port Path Cost values

Link Speed	Recommended Value	Recommended Range
<=100 Kb/s	200 000 000	20 000 000–200 000 000
1 Mb/s	20 000 000	2 000 000–200 000 000
10 Mb/s	2 000 000	200 000–20 000 000
100 Mb/s	200 000	20 000–2 000 000

1 Gb/s	20 000	2 000–200 000
10 Gb/s	2 000	200–20 000
100 Gb/s	200	20–2 000
1 Tb/s	20	2–200
10 Tb/s	2	1–20

(Source: IEEE Std 802.1D-2004, Mac Bridges)

The Port on each Bridge with the lowest Root Path Cost is assigned the role of Root Port for that Bridge (the Root Bridge does not have a Root Port). If a Bridge has two or more ports with the same Root Path Cost, then the port with the best Port Identifier is selected as the Root Port. Part of the Port Identifier is fixed and is different for each Port on a Bridge, and part is a manageable priority component. The relative priority of Ports is determined by the numerical comparison of the unique identifiers, with the lower numerical value indicating the better identifier.

Each LAN in the Bridged Local Area Network also has an associated Root Path Cost. This is the Root Path Cost of the lowest cost Bridge with a Bridge Port connected to that LAN. This Bridge is selected as the Designated Bridge for that LAN. If there are two or more Bridges with the same Root Path Cost, then the Bridge with the best priority (least numerical value) is selected as the Designated Bridge. The Bridge Port on the Designated Bridge that is connected to the LAN is assigned the role of Designated Port for that LAN. If the Designated Bridge has two or more ports connected to the LAN, then the Bridge Port with the best priority Port Identifier (least numerical value) is selected as the Designated Port.

In a Bridged Local Area Network whose physical topology is stable, i.e. RSTP has communicated consistent information throughout the network, every LAN has one and only one Designated Port, and every Bridge with the exception of the Root Bridge has a single Root Port connected to a LAN. Since each Bridge provides connectivity between its Root Port and its Designated Ports, the resulting active topology connects all LANs (is “spanning”) and will be loop free (is a “tree”).

Any operational Bridge Port that is not a Root or Designated Port is a Backup Port if that Bridge is the Designated Bridge for the attached LAN, and an Alternate Port if the Bridge has already chosen a better Root port. An Alternate Port offers an alternate path

in the direction of the Root Bridge to that provided by the Bridge's own Root Port, whereas a Backup Port acts as a backup for the path provided by a Designated Port in the direction of the leaves of the Spanning Tree. Backup Ports exist only where there are two or more connections from a given Bridge to a given LAN; hence, they (and the Designated Ports that they back up) can only exist where two ports are connected together in loopback by a point-to-point link, or where the Bridge has two or more connections to a shared media LAN.

1.1.10. Blocking Loop Paths

Once the spanning tree process has determined the port roles, then the combination of root ports and designated ports provides the spanning tree algorithm with the information it needs to identify the best paths and block all other paths as shown in Figure 1.4. Packet forwarding on any port that is not a root port or a designated port is disabled by blocking the forwarding of packets on that port. While blocked ports do not forward packets, they continue to receive BPDUs.

1.2. Problem Statement

The broadcast storm caused by a broadcast frame circling endlessly in layer-2 network due to loop network topology will also reside in layer-2 Software Defined Network.

1.3. Objective

The main objectives of this thesis are

- To realize Local Area Network in Software Defined Network architecture.
- To prevent broadcast storm in Local Area Network using spanning tree protocol defined by IEEE 802.1D-2004.
- To analyze the performance of spanning tree protocol in Software Defined Network.

CHAPTER TWO: LITERATURE REVIEW

2.1. Software Defined Networking

In networking devices, there exist three planes: data plane, control plane and management plane. Data plane refers to the hardware part where forwarding takes place, and control plane refers to the software part where all network logics and intelligence takes place. Typically in networking devices, control plane consist of firmware developed and maintained by vendors only. Management plane is typically a part of control plane and is used for network monitoring and controlling purposes.

In a traditional network shown in Figure 2.1, the data plane and control plane are embedded inside the network device [2]. The network administrator provides the configurations for the data flows, paths, routing & forwarding logic etc. These controls or instructions are pushed to the data plane where the network data traffic is handled. In this model, after the controls are defined, the only way to modify or adjust the data flow is through reconfiguration of the device. And such modifications have to be done over 1000s of devices (though certain degree of scripting or automation may be achieved). This tight coupling between the data plane and control plane is too restrictive to network operators who have to respond to traffic changes.

The problem arises when the data flow changes and new paths have to be defined or when new devices have to be provisioned or new protocols or application policies have to be applied. The only way to achieve this is by re-configuring the device and re-writing the data flow rules by network administrators who are familiar with device specific instructions. These device configurations can be done only by the network operators. With the increase of network usage and data flow growth, network operators and users are keen to see a scaling and easily adopting network.

Software Defined Networking (SDN) is a new concept of network resource virtualization. “In SDN architecture the control and data plane are decoupled, network intelligence and state are logically separated, and the underlying network infrastructure

is abstracted from the application”, as defined by The Open Networking Foundation (ONF) [2].

SDN focuses on key areas, which are:

1. Separation of data plane from control plane.
2. Centralization of control plane.
3. Standardized interfaces between the device and controller.
4. Programmability of control plane by external applications.

1. Separation of data plane from control plane.

Data plane is the hardware substrate or the infrastructure that is responsible to move the data from one point to another. To transport these data/messages, the data plane uses routing and forwarding rules; these rules (also called controls or configurations) are managed by the control plane. By separating the control plane from the data plane, SDN provides the flexibility to view the entire data plane infrastructure as a virtual resource that can be configured and controlled by an upper layer control plane. The SDN architecture is shown in Figure 2.1 highlighting the separation of network devices from control system. In a SDN architecture, the network appears as one logical device to the applications.

2. Centralization of control plane.

The control layer provides a global view of all the network wide resources, representing all the network devices as one virtual logical network. Centralizing the control plane allows it to inspect the state of the data layer and make adjustments dynamically to respond to new demands and changing conditions.

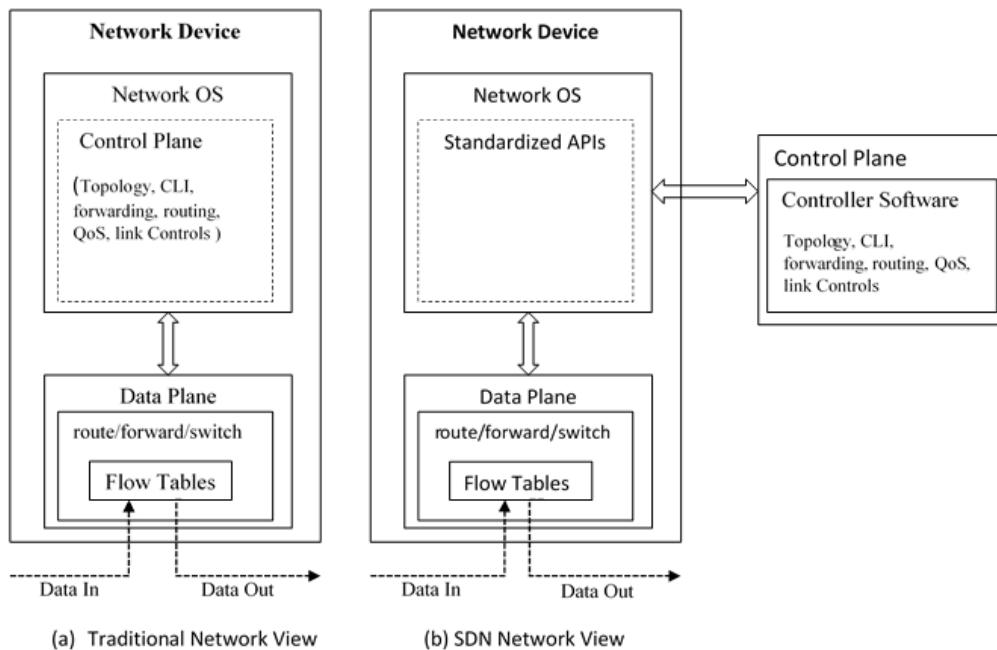
3. Standardized interfaces between the device and controller.

In Figure 2.2, the SDN architecture is shown with data flowing between application, control and data layers generally referred as North-South messages. Southbound messages from the controller communicate with lower level hardware infrastructure and northbound messages communicate with business applications. To facilitate easy adoption and make the SDN architecture vendor independent, these

communications (APIs) should ideally follow well defined standards. Applications can use the northbound messages to extract information about the network.

4. Programmability of control plane by external applications.

With the control plane centralized and standard interfaces defined between the different layers, it is easy to implement system wide policies, provision the entire network for different demand conditions, split it into multiple virtual networks, and dynamically adopt for changing business demands. The programmable control plane allows the different components of the network communicate seamlessly, and gives a network flexible adoption control. SDN controller software runs on a separate hardware providing a centralized access to the entire network.



Source: Open Networking Foundation

Figure 2.1: Traditional network compared to SDN network

2.2. Limitations in Traditional Networking Concept

The changing traffic patterns, rise of cloud services, and growing demand of bandwidth has led service operators to look for innovative solutions, since traditional networking

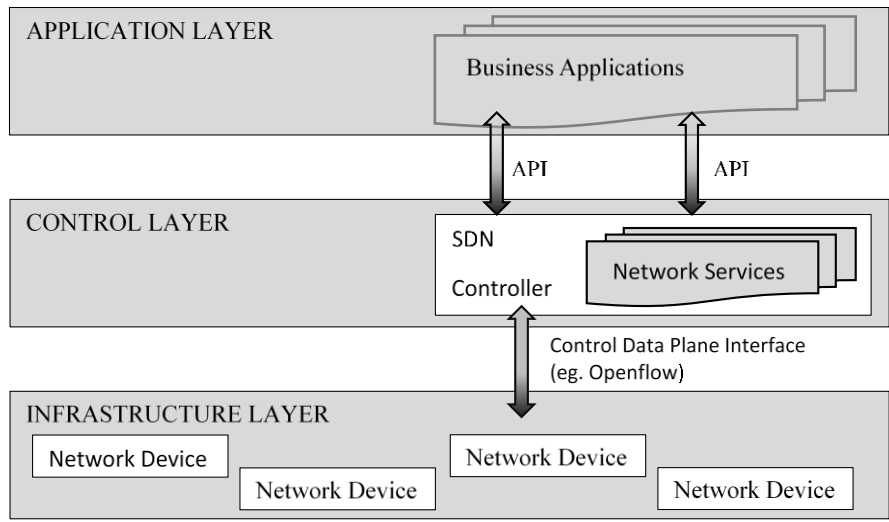
technologies are not able to meet those needs. Factors that limit achieving the growing demand while maintaining profits are [2]:

- Complexity
- Inconsistent policies
- Inability to scale
- Vendor dependence

2.3. SDN architecture

An SDN architecture consists of three layers as shown in Figure 2.2. At the top is the application layer, which includes applications that deliver services, such as switch/network virtualization, firewalls, and flow balancers. These are abstracted from the bottom layer, which is the underlying infrastructure layer. In between lies the SDN controller, the most critical element of SDN. The controller removes the control plane from the network hardware and runs it as software, but must integrate with all the physical and virtual devices in the network.

The infrastructure and control layer are connected via control data plane interface such as OpenFlow protocol, whereas the application layer is connected to the control layer via application programming interfaces (APIs). The nodes at control layer are called as controllers, and they send information such as routing, switching, priority etc. to the data plane nodes associated with them. After receiving the information from control node, the networking devices in the data plane update their forwarding table according to the information received from the control plane [2].



Source: Open Networking Foundation

Figure 2.2: Software Defined Networking Architecture

2.4. OpenFlow

OpenFlow is an open standard that offers controlling the networking equipment programmatically [4]. Several vendors have adopted the OpenFlow protocol, originally developed at Stanford University, as the basis of their SDN strategies. But OpenFlow is not the only way to do SDN and should not be equated with it. The OpenFlow specification is now in version 1.4 and is managed by the Open Networking Foundation (ONF). The goal is to create a common "language" for programming network switches. OpenFlow is used between a controller and a switch to tell the controller about traffic flows and communicates to the switch how to forward those flows.

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. The example of OpenFlow controller are NOX, POX, Floodlight, Opendaylight and Ryu.

2.5. Ryu Controller

Ryu is a component-based software defined networking framework [5].

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 and Nicira Extensions.

CHAPTER THREE: METHODOLOGY

For the infrastructure layer, the choice of the emulator is mininet which is a network emulator which creates a network of virtual hosts, switches, controllers, and links [6] [7]. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking. A looped switch network as shown in Figure 3.1 consisting of three switches and three hosts were emulated in mininet. A topology script file in Python for mininet emulator was developed. The switches are the SDN based Open vSwitch [8]. The RSTP controller controls the behavior of these switches. The three hosts are connected to each switches for the purpose of verifying the connectivity provided by the looped network in both case when there is an RSTP controller and when there is no RSTP controller. Also the ping request from one host to another will generate an ARP message from the source host which will be broadcasted into the network, which is the base of verification of spanning tree convergence.

For the controller layer to infrastructure layer communication OpenFlow Protocol was used. And to implement OpenFlow controller in controller layer Ryu Controller has been chosen as it supports up to the latest version of OpenFlow i.e. OpenFlow v1.4 [5] [9]. Python based RSTP controller script file was developed in Ryu Framework for this purpose. To validate the RST BPDUs generated and to view ARP packet flooding, Wireshark (a network protocol analyzer) is used [10].

A Ping command as shown in Figure 3.1 was requested which generates an ARP packet broadcast into the network. Under the general operation of Layer-2 LAN switch, ARP message broadcast will occur as the destination host is unknown. In a looped network, this process will create a broadcast storm as the same packet circulates round the loop. The SDN application of RSTP controller blocked this undesirable storm by pre-formation of a tree topology of the physically looped network as shown in Figure 3.2. For each switch, the selection of the port roles i.e. Root, Designated and Alternate was determined by the RSTP controller.

The packets were captured by Wireshark in each simulation, without or with the RSTP controller. The capture packets were analyzed in Wireshark for the validation of the RST BPDUs sent by the switches. Also the captured packets were analyzed to verify whether there was Broadcast Storm in the network or not.

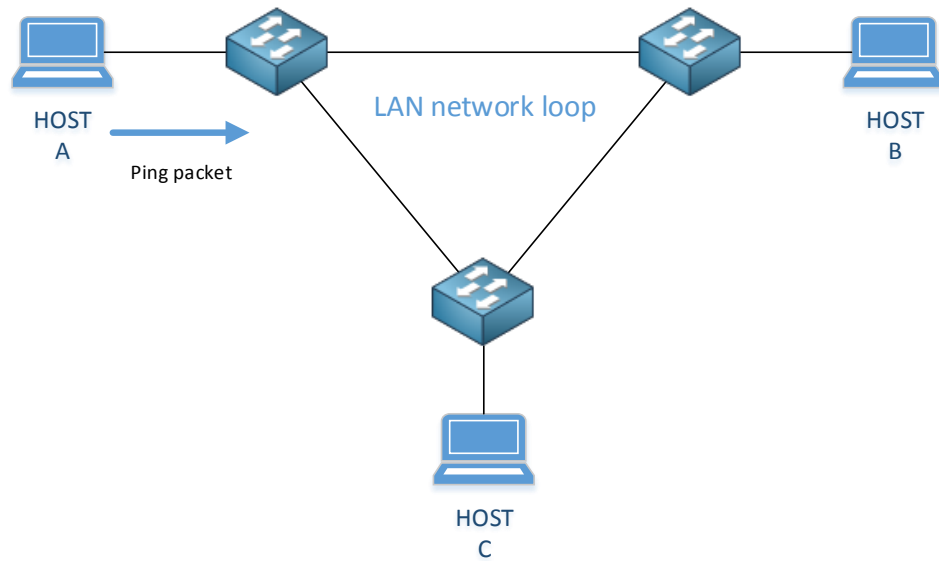


Figure 3.1: Network Topology

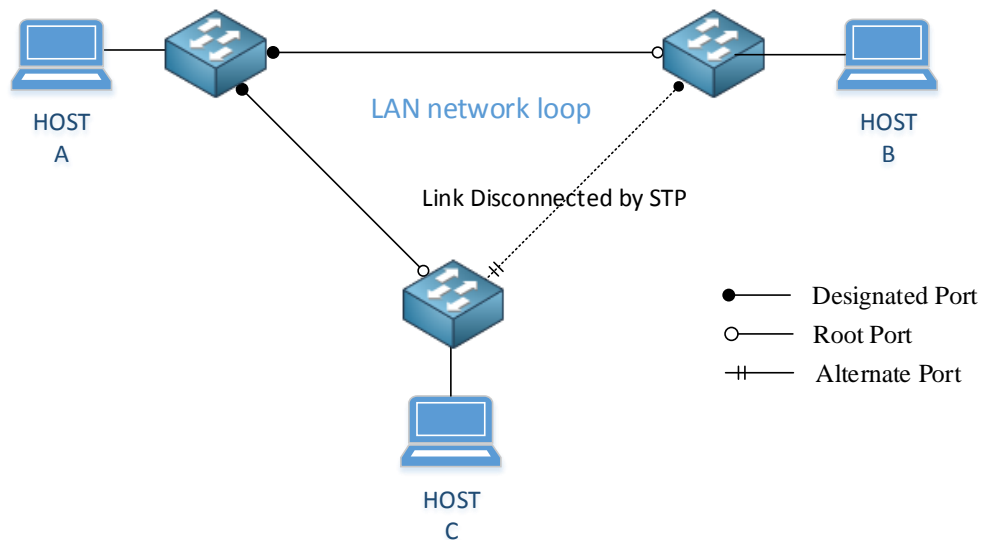


Figure 3.2: Spanning Tree with port roles

Table 3.1: Simulation Tools

SN	Items	Description
1	mininet	Emulator
2	Ryu	SDN controller
3	OpenFlow	Control Data Plane Interface
4	ICMP ping	Ping command to create a broadcast storm
5	Wireshark	For validation of RST BPDU packets and for traffic capture

CHAPTER FOUR: SIMULATION PROCESS

The simulation is based on the mininet emulator and Ryu framework. Mininet emulator emulated the virtual switched network while the Ryu framework simulated the SDN OpenFlow based controller. The simulation was executed in Ubuntu 14.04.4 LTS.

- Network emulator : Mininet v2.2.0
- Controller Framework : Ryu v4.4
- SDN Southbound protocol : OpenFlow v1.3
- Scripting language : Python v2.7.6
- Validation tool : Wireshark v1.10.6

Simulation process incorporates three different sections:

- i. SDN Network implementation in Emulator (Mininet)
- ii. OpenFlow based Controller (Ryu) for monitoring and controlling the traffic flow in the SDN network
- iii. Controller as a RSTP controller for the switch network

Section i deals mainly with the mininet environment. For the Network implementation, a fully functioning mininet script in Python was written. The network topology is consistent with the three switch network shown in Figure 3.1. Apart from the figure, the script incorporates a tcp port connection to a remote controller over port 6633 [4]. The details of the emulated network are:

Table 4.1: Network details

SN	Items	Value
1	Total Switches	3
2	Hosts connected to each switch	1
3	TCP port connection to Remote Controller	6633
4	Host IP format	10.0.0.H where H = host number
5	Host MAC format	00:00:00:00:00:HH where H = host number
6	Switch interface MAC format	00:00:00:00:0S:0I where S = Switch number I = interface number

The Host IP address, Host MAC address and Switch interface MAC address format has been designed for simplicity in the debugging process. For example 10.0.0.1 indicates host h1 in mininet environment. Likewise, MAC ID 00:00:00:00:00:11 indicated MAC ID of host h1 and MAC ID 00:00:00:00:01:01 indicates MAC ID of interface s1-eth1 in mininet environment.

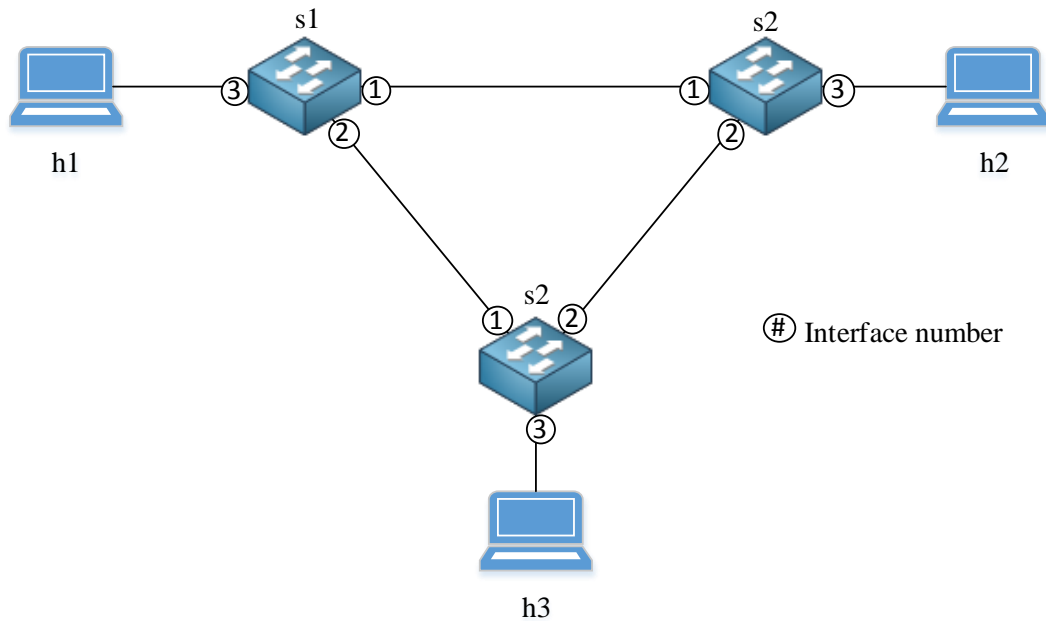


Figure 4.1: Simulated Network in mininet with interface numbers

Section ii includes the Remote controller implementation for which the Ryu controller is chosen. The choice of the controller was made being based on simplicity of Python Programming language and also Ryu controller supporting the latest versions of OpenFlow protocol. The Ryu controller connects to the network switches via tcp port no 6633 to monitor the packet inflows to the switch and to handle those packet processing and forwarding.

Section iii requires building a Ryu app for RSTP controller to the switch network. The controller needs to direct each switch in network to form a RST BPDU packet and multicast to special destination MAC address 01:80:c2:00:00:00. This MAC address is specially reserved by 802.1D Standard as Bridge Group Address for transmitting BPDUs to all other Spanning Tree Protocol Entities.

CHAPTER FIVE: RESULTS AND DISCUSSIONS

The SDN Network implementation in mininet environment was carried in two scenarios. The first scenario was without the RSTP controller running and the second scenario was with the controller running parallel in Ryu.

Three switches loop network was emulated in mininet environment as a prerequisite for both the scenarios. The simulated network in mininet followed the design guideline as listed in Table 4.1 and the topology as shown in Figure 4.1. Also the switches were set to operate in OpenFlow version 1.3 explicitly as default version used by the Open vSwitch in mininet is version 1.0 [8].

Figure 5.1 shows the execution of the python scrip topology file ‘stp_3switch.py’ for mininet environment, where the debug output in mininet shows the steps of the network emulation. And in the mininet CLI prompt. The initial flow-table is checked and verified to be empty.

```
kbn@Trusty:~$ sudo python mininet/custom/stp_3switch.py
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h3 h1 h2
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** s1 : ('ovs-vsctl set bridge s1 protocol=OpenFlow13',)
*** s3 : ('ovs-vsctl set bridge s3 protocol=OpenFlow13',)
*** s2 : ('ovs-vsctl set bridge s2 protocol=OpenFlow13',)
*** set default mac for switch interfaces ***
*** Starting CLI:
mininet> dpctl dump-flows -O OpenFlow13
*** s1 -----
OFPST_FLOW reply (OF1.3) (xid=0x2):
*** s3 -----
OFPST_FLOW reply (OF1.3) (xid=0x2):
*** s2 -----
OFPST_FLOW reply (OF1.3) (xid=0x2):
mininet>
```

Figure 5.1: Running Switched network in mininet with empty flow-table shown

5.1. SDN without RSTP controller

A ping request from host h1 to host h2 as shown in Figure 4.1 was made in mininet CLI prompt shown in Figure 5.2 which created the ARP message broadcast storm in looped network as observed in Figure 5.3.

```
mininet> h1 ping -c 1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

Figure 5.2: Ping request from host h1 to h2 under no RSTP controller running

The ping request from host h1 results in ARP message broadcast starting from host h1 to find the destination MAC address of host h2 or of the gateway interface to host h2. The ARP message broadcasted are rebroadcasted again by the same switch after the broadcast message has been received round the loop. This resulted in the large packet flow and also the required function of ARP message to find the destination MAC address was not full filled. Thus the ping request also was not successful as observed by 100% packet loss in Figure 5.2.

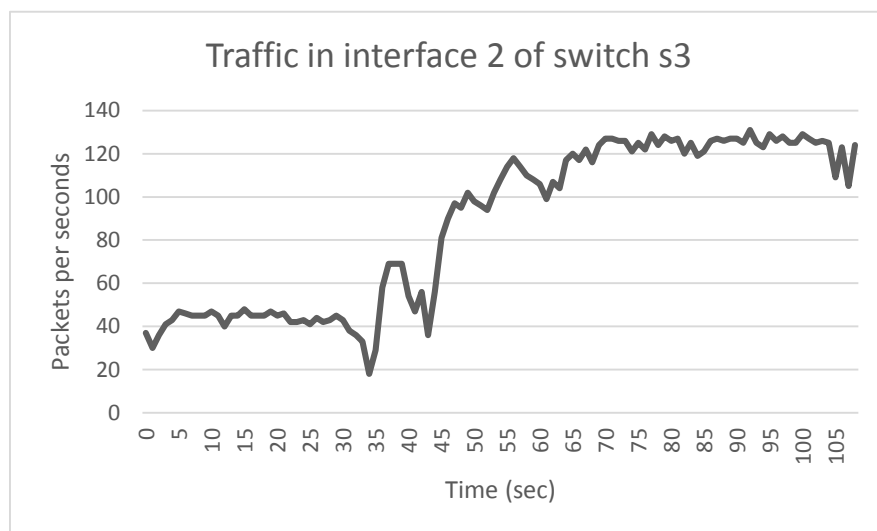


Figure 5.3: Packets flow in the looped network

The Figure 5.3 shows the large packets flow in interface 2 of switch S3 after the ping request was made. Though the ping request was from host h1 to host h2 whose direct shortest path does not include switch s3, still due to prevailing loop in the network ARP broadcast message reach switch s3 from both switches s1 and s2. In response switch s3 rebroadcasts ARP message towards switch s2 and s1.

The certain drops in the graph is merely due to the flow control mechanism of the TCP IP protocol stack. Due to sudden high traffic increase the receiving switch buffers gets full and signals back flow control message to the transmitting switch. And under this condition the broadcast message packet flow was observed to be decreasing. The decreasing pattern is more prominently observed in 35 and 45 seconds where traffic is 40 to 60 packets per seconds but only slightly observed in high packet flow per seconds which is above 120 packets per seconds. This is due to the greater packets flow increasing suddenly in the network.

The large packet flow was validated to be ARP message broadcast from the network protocol analyzer tool, Wireshark by capturing the traffic in the above mentioned interface shown in Figure 5.4. As observed clearly in the Wireshark, the destination address is Broadcast which is actually broadcast address of ff:ff:ff:ff:ff:ff. The protocol of the message is ARP, which indicated the ARP request generated the traffic. The source MAC address of 00:00:00:00:33 indicates the packet is generated by host 3.

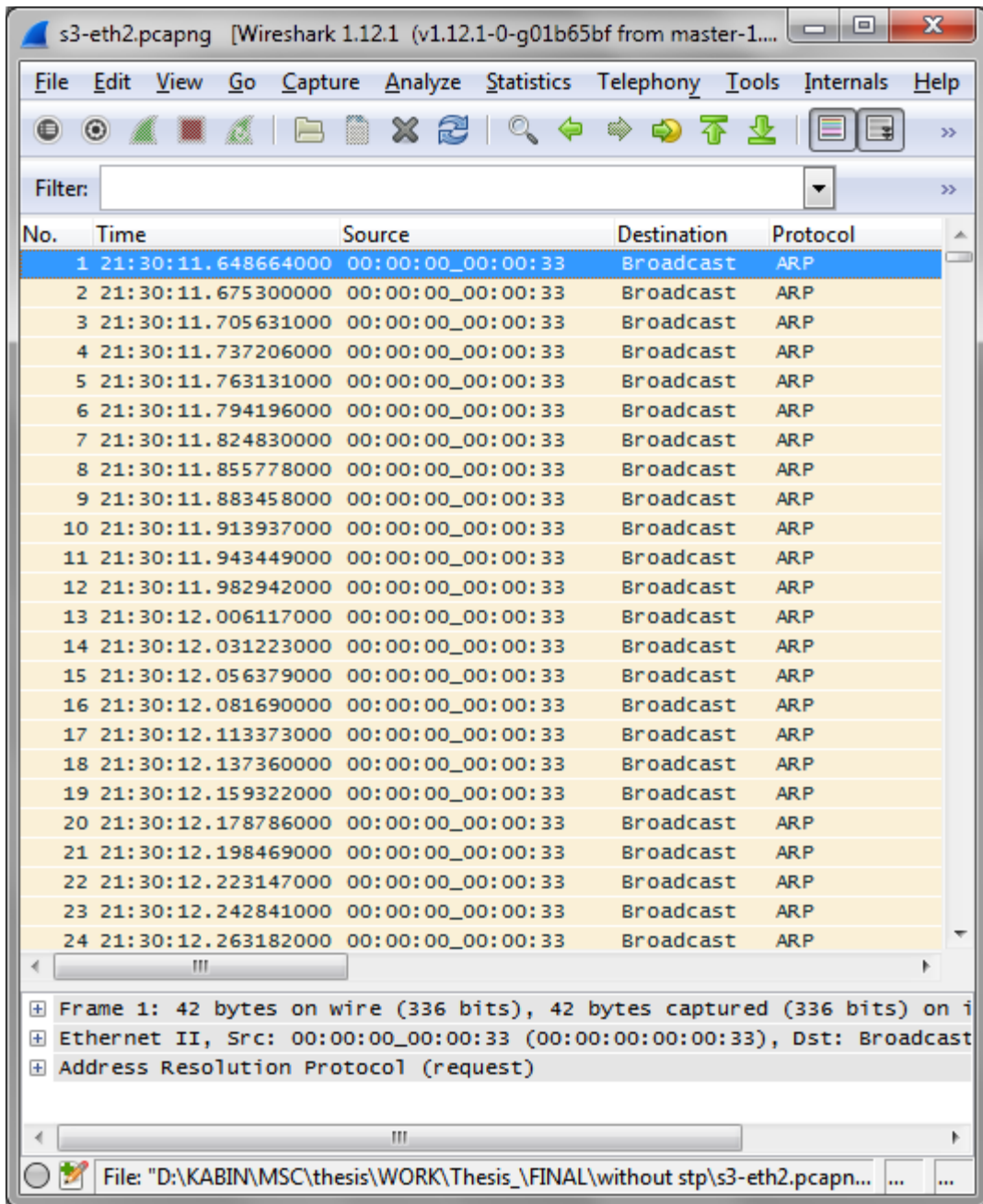


Figure 5.4: ARP message Broadcast storm as observed in Wireshark

5.2. SDN with RSTP controller

The python based controller script file was developed to incorporate the RSTP function in the Ryu Controller. The three switches loop network was emulated in mininet environment along with the RSTP controller running in Ryu framework.

Figure 5.5 shows the execution of RSTP controller scrip file ‘rstp13.py’ in Ryu framework. The debug output also shows a switch in mininet environment joining the STP bridge at the line in Figure 5.5, which already involves the handshaking operations between three SDN switches and Ryu Controller as guided by OpenFlow protocol [4].

```
kbn@Trusty:/usr/local/lib/python2.7/dist-packages/ryu/lib$ ryu-manager rstp13.py
loading app rstp13.py
loading app ryu.controller.ofp_handler
instantiating app None of RStp
INFO: Rstp instance created .....19 sept
creating context rstplib
instantiating app rstp13.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
2016-09-22 23:12:06,897 [RSTP][INFO] dpid=0000000000000003: Join as stp bridge.
```

Figure 5.5: Running Ryu controller

The controller builds a formatted RST BPDU packet for each SDN switch and forwards to the SDN switches for multicasting to Bridge Group Address 01:80:c2:00:00:00. Figure 5.6 shows the RST BPDU packet captured on interface eth2 of switch S3 by Wireshark. After receiving the RST BPDU, switch sends the RST BPDU packet to the controller for processing. Based to information about the Root Bridge in RST BPDU sent by the switch and in the received RST BPDU packet, the controller elects the Root Bridge and the assigns respective roles to the ports of the switch. And with the Forward delay timer expiry, ports goes through different states as seen in Figure 5.7.

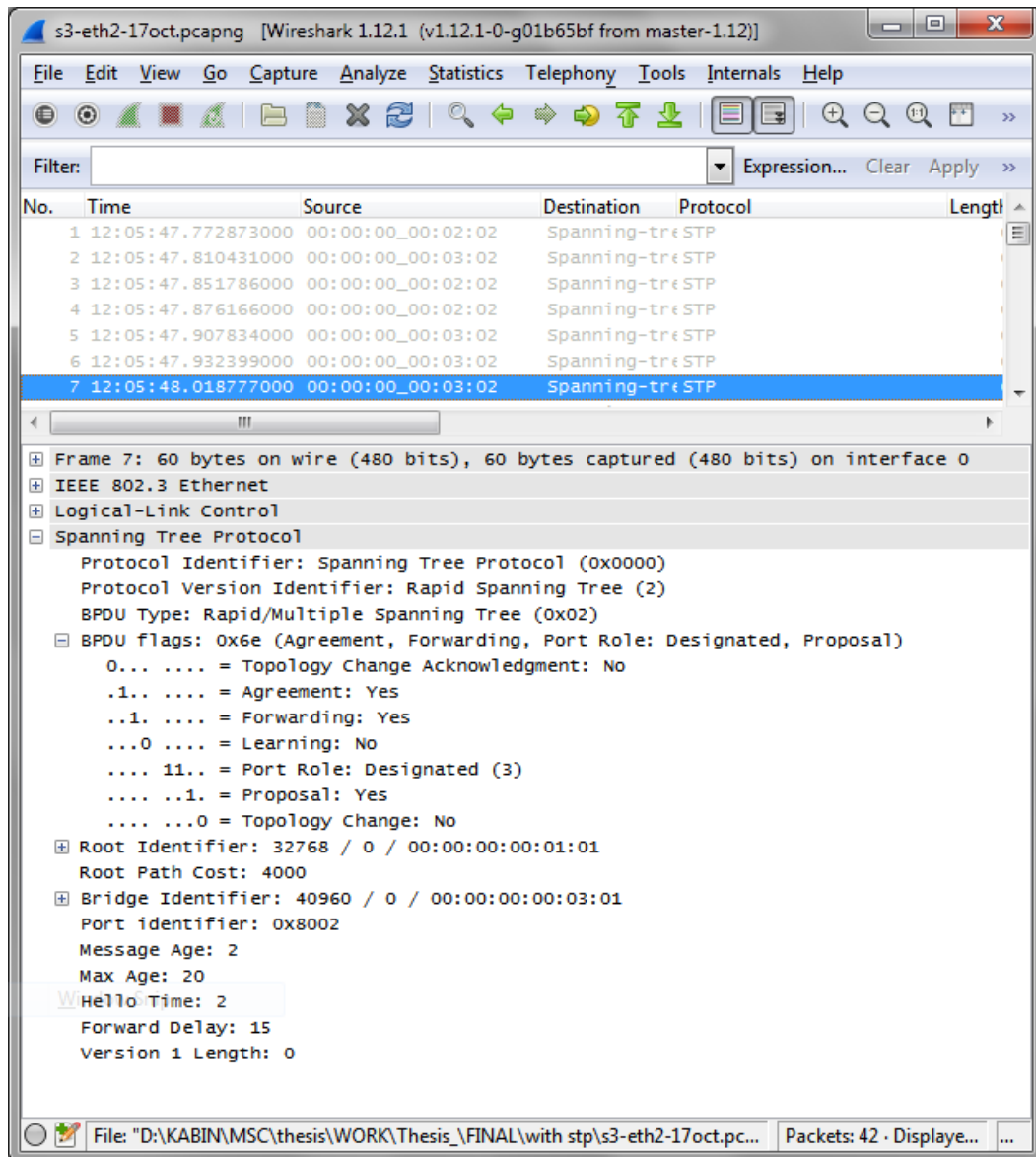


Figure 5.6: RST BPDU packet captured by Wireshark

Figure 5.6 shows an instance of the RST BPDU packet captured at 12:05:48. The details about the packet is shown in lower half of the figure. Under the Spanning Tree Protocol all the fields of the RST BPDU packets can be observed. And Protocol Version Identifier value of 2 signifies that it is RST BPDU packet.

2016-10-20	22:27:04,949	[RSTP][INFO]	dpid=0000000000000001: Root bridge.
2016-10-20	22:27:04,954	[RSTP][INFO]	dpid=0000000000000001: [port=1] DESIGNATED_PORT / DISCARD
2016-10-20	22:27:04,964	[RSTP][INFO]	dpid=0000000000000001: [port=2] DESIGNATED_PORT / DISCARD
2016-10-20	22:27:04,970	[RSTP][INFO]	dpid=0000000000000001: [port=3] DESIGNATED_PORT / DISCARD
2016-10-20	22:27:19,691	[RSTP][INFO]	dpid=0000000000000003: [port=1] ROOT_PORT / LEARN
2016-10-20	22:27:19,699	[RSTP][INFO]	dpid=0000000000000003: [port=2] NON_DESIGNATED_PORT / LEARN
2016-10-20	22:27:19,707	[RSTP][INFO]	dpid=0000000000000003: [port=3] DESIGNATED_PORT / LEARN
2016-10-20	22:27:19,812	[RSTP][INFO]	dpid=0000000000000002: [port=1] ROOT_PORT / LEARN
2016-10-20	22:27:19,821	[RSTP][INFO]	dpid=0000000000000002: [port=2] DESIGNATED_PORT / LEARN
2016-10-20	22:27:19,833	[RSTP][INFO]	dpid=0000000000000002: [port=3] DESIGNATED_PORT / LEARN
2016-10-20	22:27:19,957	[RSTP][INFO]	dpid=0000000000000001: [port=1] DESIGNATED_PORT / LEARN
2016-10-20	22:27:19,976	[RSTP][INFO]	dpid=0000000000000001: [port=2] DESIGNATED_PORT / LEARN
2016-10-20	22:27:19,981	[RSTP][INFO]	dpid=0000000000000001: [port=3] DESIGNATED_PORT / LEARN
2016-10-20	22:27:34,694	[RSTP][INFO]	dpid=0000000000000003: [port=1] ROOT_PORT / FORWARD
2016-10-20	22:27:34,702	[RSTP][INFO]	dpid=0000000000000003: [port=2] NON_DESIGNATED_PORT / BLOCK
2016-10-20	22:27:34,713	[RSTP][INFO]	dpid=0000000000000003: [port=3] DESIGNATED_PORT / FORWARD
2016-10-20	22:27:34,815	[RSTP][INFO]	dpid=0000000000000002: [port=1] ROOT_PORT / FORWARD
2016-10-20	22:27:34,825	[RSTP][INFO]	dpid=0000000000000002: [port=2] DESIGNATED_PORT / FORWARD
2016-10-20	22:27:34,837	[RSTP][INFO]	dpid=0000000000000002: [port=3] DESIGNATED_PORT / FORWARD
2016-10-20	22:27:34,961	[RSTP][INFO]	dpid=0000000000000001: [port=1] DESIGNATED_PORT / FORWARD
2016-10-20	22:27:34,979	[RSTP][INFO]	dpid=0000000000000001: [port=2] DESIGNATED_PORT / FORWARD
2016-10-20	22:27:34,986	[RSTP][INFO]	dpid=0000000000000001: [port=3] DESIGNATED_PORT / FORWARD

Figure 5.7: Final Port States in switches

Figure 5.7 shows the roles of different ports of all the three switches and the states based on those roles. The time information has also been traced out to debug the time information of the port state changes. And the change in time from 19.691 seconds in Learn state to 34.694 seconds in Forward which is 15.003 seconds resembles the forwarding delay time of the RSTP protocol [11].

A ping request from host h1 to host h2 was executed to check the host reachability from mininet CLI prompt, after all the switches have been in communication with Ryu controller and the interfaces of switches went to forwarding state. As certain port has been already assigned the Block state which resembles no traffic in or out flow, the network now does not have any loop. Hence, the ping request is accepted and replied by host h2 as observed in Figure 5.8. For showing the continuity of the ping reply and for making the ping traffic more prominently visible in traffic graph, 10 ping request were sent in this case. The observed Figure 5.8 also show that no packet has been lost.

```

mininet> h1 ping -c 10 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=129 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=3.31 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.104 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.108 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.087 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.123 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.091 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.205 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.486 ms

--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 0.087/13.384/129.233/38.627 ms

```

Figure 5.8: Ping reply from host h2 under RSTP controller running

During the first ping request an ARP packet from host h1 destined to host h2 IP address is broadcasted in the network. But due to the linear network topology the broadcast is no rebroadcasted as in previous case. The ARP request message flooded to Broadcast MAC address ff:ff:ff:ff:ff:ff is also replied back in this scenario by the host h2 as observed in the controller debug log in Figure 5.9.

```

2016-10-20 23-25-14 [CONTROLLER] packet in SW=1 in_port=3 from=00:00:00:00:00:11 to=ff:ff:ff:ff:ff:ff
2016-10-20 23-25-14 [CONTROLLER] packet in SW=3 in_port=1 from=00:00:00:00:00:11 to=ff:ff:ff:ff:ff:ff
2016-10-20 23-25-14 [CONTROLLER] packet in SW=2 in_port=1 from=00:00:00:00:00:11 to=ff:ff:ff:ff:ff:ff
2016-10-20 23-25-14 [CONTROLLER] packet in SW=2 in_port=3 from=00:00:00:00:00:22 to=00:00:00:00:00:11
2016-10-20 23-25-14 [CONTROLLER] packet in SW=1 in_port=1 from=00:00:00:00:00:22 to=00:00:00:00:00:11

```

Figure 5.9: ARP request and reply

The traffic flow graph for this scenario with the presence of RSTP controller can be seen in Figure 5.10. The traffic flow was captured by Wireshark at interface 1 of switch s1. The time interval of the capture was 44 seconds during which the ping request was sent from host h1 at interval between 27 to 36 seconds. Figure 5.10 also shows no occurrence of broadcast storm. Figure 5.10(a) shows the presence of ARP request at 27 seconds, the reply for which is observable at 32 seconds in the same graph. In Figure 5.10(b) the periodic occurrence of RST BPDU packets every 2 sec is also observable, as guided by hello timer. The presence of ping packets can be seen in Figure 5.10(c). In all the graph the line graph resembles the sum of ARP, RST BPDU and Ping traffic.

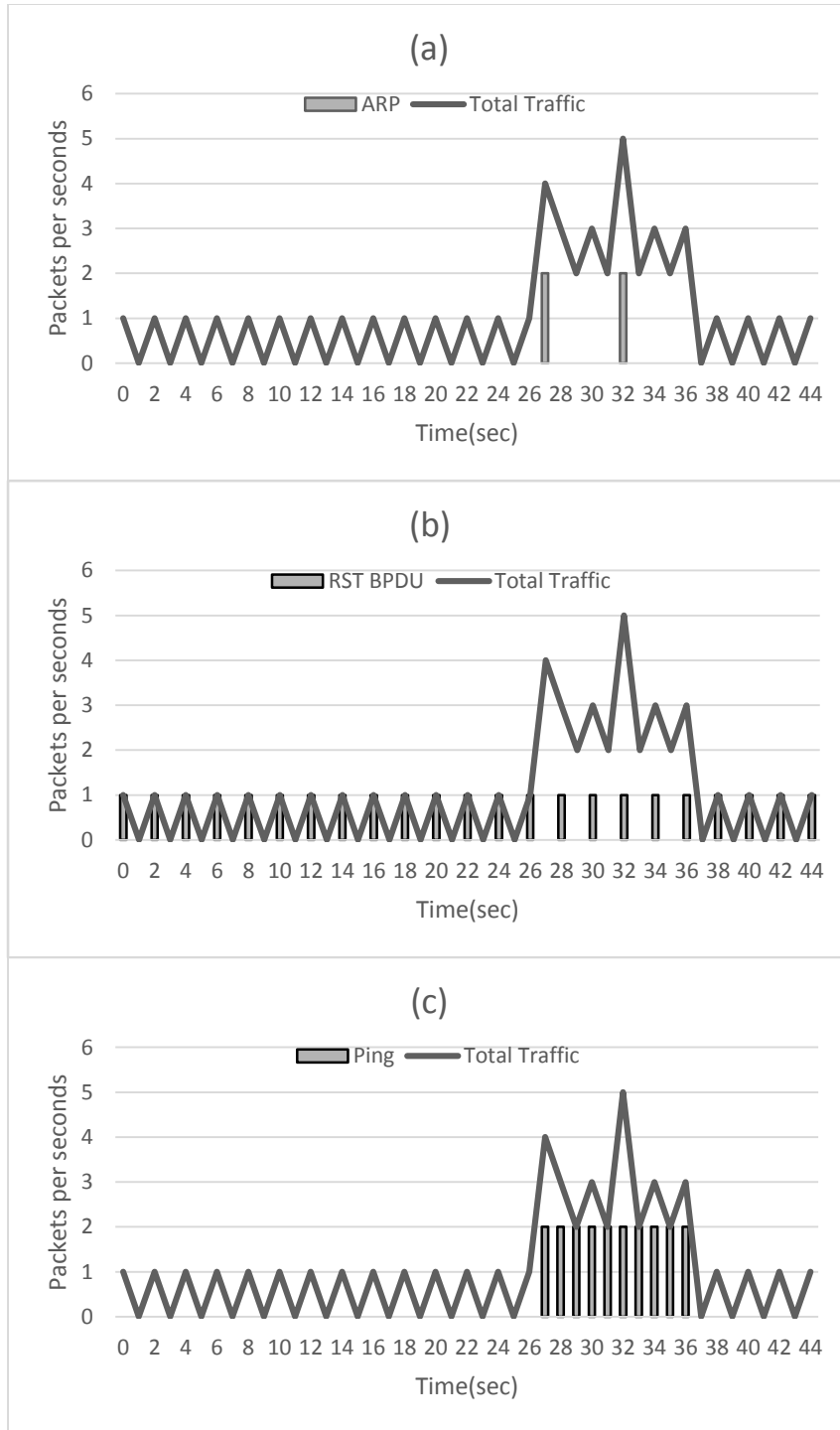


Figure 5.10: Traffic flow in interface 1 of switch s1 after RSTP controller running
 (a) with ARP messages (b) with RST BPDUs (c) with Ping traffic

5.3. Analysis of the Controller

For the simulation process three switch ring network was taken. The no of traffic handled by the controller over the TCP port no 6633 connected to the SDN switches was observed over 300 seconds period duration as captured by Wireshark is shown in Table 5.1.

Table 5.1: Packets to and from Controller

SN	Switch	Total Packets	Total Bytes	Packets to Controller	Bytes to Controller	Packets from Controller	Bytes from Controller
1	S1	1094	127044	535	69118	559	57926
2	S2	934	115722	379	43140	555	72582
3	S3	1119	129348	539	37846	580	91502
Total		3147	372114	1453	150104	1694	222010

With multiple number of switches implemented in similar ring topology, the total traffic handled by the controller increased linearly as shown in the Table 5.2 and clearly visible in Figure 5.11.

Table 5.2: Variation in packets to and from controller with switches

SN	Total switches	Total Packets	Total Bytes	Packets to Controller	Bytes to Controller	Packets from Controller	Bytes from Controller
1	2	1958	235782	1017	102156	941	133626
2	3	3147	372114	1453	150104	1694	222010
3	4	4523	530828	2126	217156	2397	313672
4	5	5534	664244	2506	265480	3028	398764
5	6	7303	859696	3544	362154	3759	497542
6	12	14650	1693642	7117	711944	7533	981698

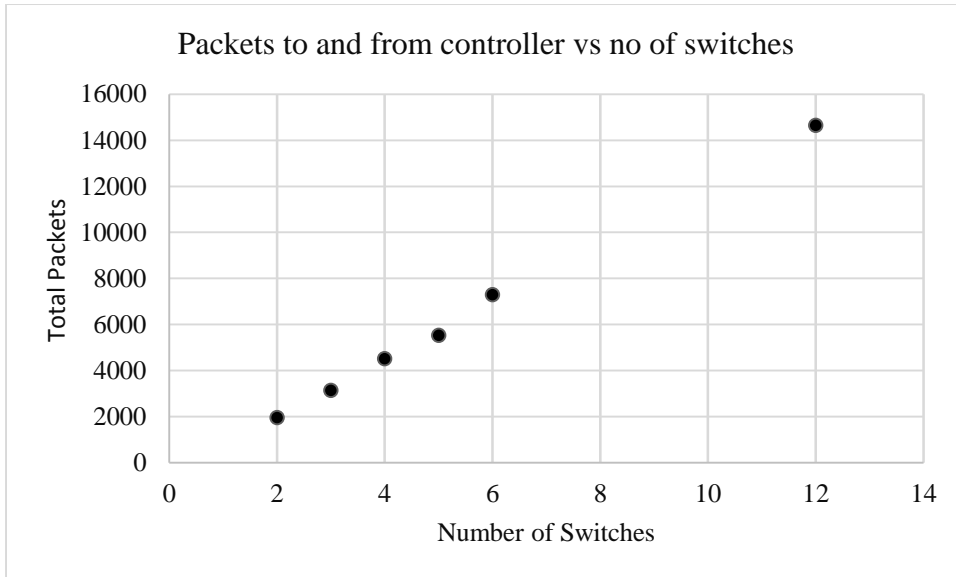


Figure 5.11: Controller Packets vs. No of switches

Hence, though the RSTP controller was seen to avoid the loop structure in the network by assigning ports with various port roles and states, the traffic handled by the controller was observed to be increasing linearly with the no of increasing switching for the ring network.

CHAPTER SIX: LIMITATIONS AND RECOMMENDATIONS

The RSTP controller performed effectively to avoid the loop in the network. But after the analysis of various sized ring networks, the traffic to be handled by the controller was observed to be increasing linearly. However, this linearly increasing packet flow in controller was not analyzed in depth and no measure what so ever had been taken for the minimization of the increasing traffic.

Hence, the further analysis and minimization of load on controller with increasing traffic with the increasing number of switch remains as future works.

CHAPTER SEVEN: CONCLUSION

The thesis was successful in controlling the Broadcast Storm in Software Defined Network. In this thesis, Broadcast Storm prevention by loop avoidance method was used. For the loop avoidance the RSTP protocol as described by IEEE 802.1D-2004 bridging standard was implemented. A RSTP controller script based on Ryu Framework for SDN was developed. The SDN environment was emulated in mininet environment over Ubuntu 14.04.4 LTS.

REFERENCES

- [1] M. Seaman, IEEE Std 802.1 D-2004, MAC Bridges, 2004.
- [2] O. N. Foundation, "Software-defined networking: The new norm for networks," ONF White Paper, 2012.
- [3] H. Zengbin, "Analysis on the Causes & Solutions of Broadcast Storm in LAN [J]," Office Informatization, vol. 10, p. 018, 2010.
- [4] B. Pfaff, B. Lantz and B. Heller, Openflow Switch Specification, Version 1.3.0 (Wire Protocol 0x04), Open Networking Foundation, 2012.
- [5] "ryu Documentation, Release 4.5," ryu development team, 2016.
- [6] K. Kaur, J. Singh and N. S. Ghumman, "Mininet as Software Defined Networking Testing Platform.," in International Conference on Communication, Computing & Systems (ICCCS. 2014), vol. 20, 2014.
- [7] B. Lantz, B. O'Connor and C. Burkard, 2014. [Online]. Available: <http://www.mininet.org>. [Accessed on 2016].
- [8] "Open vSwitch," [Online]. Available: <http://www.openvswitch.org/>. [Accessed on 2016].
- [9] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov and R. Smeliansky, "Advanced study of SDN/OpenFlow controllers," in Proceedings of the 9th central & eastern european software engineering conference in russia, ACM, 2013.
- [10] G. Combs, "Wireshark," [Online]. Available: <https://www.wireshark.org/>. [Accessed on 2016].
- [11] "Understanding and Tuning Spanning Tree Protocol Timers," Cisco Systems, Inc, 2006. [Online]. Available: <http://www.cisco.com/>. [Accessed on 2016].
- [12] F. Ketikci and S. Askar, "Emulation of Software Defined Networks Using Mininet in Different Simulation Environments," in 2015 6th International Conference on Intelligent Systems, Modelling and Simulation, IEEE, 2015, pp. 205--210.
- [13] Python Software Foundation, 2016. [Online]. Available: <https://www.python.org/>. [Accessed on 2016].