Improving existing VPN through Software Defined Networking

BY: GOYTOM MEBRATU

ADVISER: YALEMZEWD NEGASH (PHD)

A Thesis submitted to the School of Electrical and Computer Engineering
Addis Ababa Institute of Technology

in Partial Fulfillment of the Requirements for the Degree of Master of Science in Telecommunication Engineering

Addis Ababa University
Addis Ababa, Ethiopia
November 16, 2018

Declaration

I, the undersigned, declare that the thesis comprises my own work in compliance with internationally accepted practices; I have fully acknowledged and referred all materials used in this thesis work.

Goytom Mebratu

Name

Signature

Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

This is to certify that the thesis prepared by **Goytom Mebratu**, entitled *Improving existing VPN through Software Defined Networking* and submitted in partial fulfillment of the requirements for the degree of Master of Science Telecommunication Engineering complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

Internal Examiner	_____	Signature	_____	Date	_____
External Examiner	_____	Signature	_____	Date	_____
Adviser	Yalemzewd Negash (PhD)	Signature	_____	Date	_____
Co-Adviser	_____	Signature	_____	Date	_____

_____

Dean, School of Electrical and Computer Engineering

ABSTRACT

Currently, telecom service providers are using Multi-Protocol Label Switching (MPLS) technology in their IP core network to provide Virtual Private Network (VPN) services to their customers. However, most of the current IP core network of service providers' network does not have dynamic centralize configuration, provisioning and management mechanism, and also the network is not scalable enough. These limitations are caused by the vertical integration of data and control plane on networking device. In addition, the IP core layers of a service provider networks have redundancy on their network architecture (e.g. ethiotelecom IP core layer has five layers), which causes delay in configuration, provisioning and management. Hence, this thesis work shows how SDN based VPN is better than the existing VPN of a service provider network by comparing two networks performance parameters namely throughput and latency. Therefore, the simulation environment is setup using Linux routers for existing network; and openvswitches for the Software Defined Network (SDN) network. Finally, the simulation results for both networks have been presented on the result and discussion section. Based on the results found, the SDN- VPN throughput is 22.2222% higher than the existing VPN, and also its latency is 70.4675% lower than the existing VPN one.

KEYWORDS

SDN VPN, throughput, latency

ACKNOWLEDGMENTS

I would like to thank God for everything he gave me to complete the thesis work. I deeply grateful to my advisors Dr. Yalemzewud Negash, he gave me useful directions, motivation, moral support and sharing his great experience in the process of this study. Finally, I have to deliver a many thanks message to the School of Electrical and Computer Engineering in AAiT and ethiotelecom for the opportunity they provides me to make my thesis work on how Software Defined Network improve the legacy network performance.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ACM Association for Computing Machinery

API Application Program Interface

CAPEX Capital Expenditure

CE Customer Edge

FIB Forward Information Base

IEEE International Electrical Electronics Engineering

MP-BGP Multi-Protocol Border Gateway Protocol

MPLS Multi-Protocol Label Switching

NMS Network Management Systems

OPEX Operational Expense

OSPF Open Shortest Path First

PE Provider Edge

RAN Radio Access Network

RD Route Distinguisher

RT Route Target

SDN Software Defined Network

VPLS Virtual Private LAN Service

VPN Virtual Private Network

XML Extensive Markup Language

1

INTRODUCTION

1.1 BACKGROUND

Currently, the limitations of traditional architectures are becoming significant: at present, modern networks no longer optimize the costs at all (i.e. the Capital Expenditure (CAPEX) and Operational Expense (OPEX)) [1]. Therefore, most of the existing Service Providers' (SPs') network is characterized as not agile, longer time to market, and not having fast provisioning techniques. Despite these limitations, IP/MPLS services are most dominant in the core of carrier class networks of service providers' network (it is also in case of ethiotelecom) [2]. Hence, ethiotelecom realizes VPN services by using MPLS technology in its IP core network; however, the current IP core network architecture of ethiotelecom does not have dynamic centralized configuration and resource allocation mechanism for provisioning and managing network resources. Therefore, ethiotelecom IP core network resources has been provisioned and managed by vendor specific systems rather than having centralized management. In addition, its network architecture is not easy to scale up. According to [2] SDN based VPN service is highly important to overcome limitations on existing service providers' network such as scalability, expensiveness and complex on network management. Thus, service providers' experiencing such types of problems is highly advised to make an assessment on their network readiness to adopt SDN based VPN service. Therefore, the new SDN paradigm is important to overcome the problems mentioned above, because it needs: i) Overall analysis; ii) Dynamic, rather than static configuration and policy usage; iii) Much greater information feedback than is the case at present [3], [4].

1.2 PROBLEM STATEMENT

Ethiotelecom has providing numerous services for its customers. One of its major carrier grade services is IP/MPLS VPN for connecting geographically distributed sites of its customers. However, the current ethio telecom IP network which provides VPN services has no dynamic centralized management for configuration and resource allocation. Therefore, the current IP core network of ethiotelecom is managed by vendor specific Network Management Systems (NMS), rather than, having one centralize management system. This creates management complexity, scalability limitation and innovation hindrance on the existing IP core network. In addition to the problems mentioned above, the current ethiotelecom IP core network layers namely, ER (Edge Router) and CS (Core Switch) are redundant; which creates service provision delay and increases CAPEX and OPEX of ethiotelecom. Finally, the existing IP core network has limitation on network management, scalability, flexibility and programmability. And, this makes expansion of the network more expensive in addition to increase complexity in management.

1.3 OBJECTIVE

1.3.1 *General Objective*

The general objective of the study is to redesign and demonstrate new SDN VPN architecture to improve the existing limitation in ethio telecom IP core network.

1.3.2 *Specific Objectives*

The specific objectives of this thesis are:

- To extensively review literatures on SDN and IP/MPLS
- To make interview and focused group discussion with ethiotelecom experts

- To study VPN services management complexity, causes of delay in provisioning, maintenance, operation and administration
- To redesign SDN based VPN architecture
- To simulate and demonstrate new architecture and existing architecture
- To make comparison analysis between the new and existing VPN networks
- To recommend ideas based on findings in the study for improving existing VPN services network related limitations

1.4 SCOPE OF THE STUDY

This thesis work is only considered Addis Ababa bole site which is labeled with red color on the existing ethiotelecom VPN architecture shown in Figure 1.1 below.

1.5 METHODOLOGY

The thesis work aims mainly to show how SDN VPN is better on throughput and latency than existing VPN network, which uses the legacy routers. The main hypothesis of the study is SDN VPN throughput and latency is better than that of the existing traditional VPN service provisioning. Specifically, the proposed method changes vendor specific and expensive routers on existing VPN network with OpenvSwitches. Hence, the controlling planes of these OpenvSwitches is handled by a centralize controller called floodlight SDN controller.

1.5.1 *Literature Review*

Extensive literatures were reviewed which are sourced from International Electrical Electronics Engineering (IEEE), Association for Computing Machinery (ACM) and vendors white paper. This literature was highly important to understand

how SDN network is working and what limitations are found on existing service providers VPN network.

1.5.2 *Interview and focus group discussion with ethiotelecom experts*

Interview and focus group discussion were conducted on ethiotelecom experts. It was highly important for investigating network related VPN problems with respect to problems identified in literature review and how the exisiting VPN network is structured.

Figure 1.1: The existing ethiotelecom VPN architecture .

1.5.3 *The new SDN VPN design model*

The limitations of the existing VPN network architecture have been improved by using the SDN model proposed below in Figure 1.2. In this model, the existing routers are replaced by SDN OpenvSwitch where their control logic is controlled by an SDN based floodlight controller.

In Figure 1.2 shown above, the model change the existing network topology architecture which have 5 layers, namely IGW, BR, CR, ER and CS with openvswitch. These switches are running quagga engine in order to provide L3 switch/router functionalities.

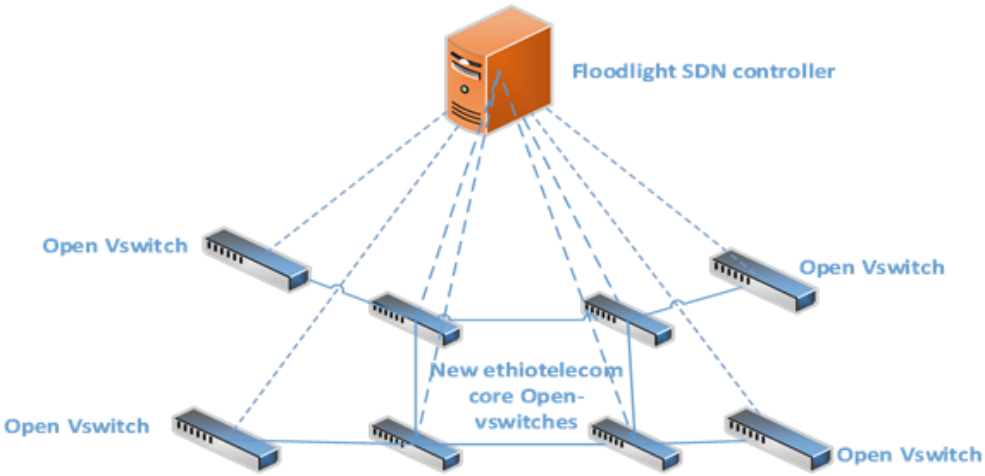

Figure 1.2: New SDN VPN model.

2

LITERATURE REVIEW

2.1 MPLS VPN

MPLS is a technology that optimizes the traffic forwarding in a network by avoiding complex lookups in the routing table. The traffic is directed based on labels contained in an MPLS packet header. The labels define only the local node to node communication and are swapped on every node. This process allows very fast switching through the MPLS core. MPLS relies on traditional IP routing protocols to determine the best routes and to receive topology updates and predetermines the path the packet will take through the network. This process is performed by the MPLS edge router and thus reduces the processing requirements for the core switching routers. These paths are called **LSP!** (LSP!)s.

2.1.1 *Lable*

The MPLS header consists of a stack of 4-byte records where each record has the following structure (depicted in Figure 2.6):

- a label field (20 bits), which carries the label value;

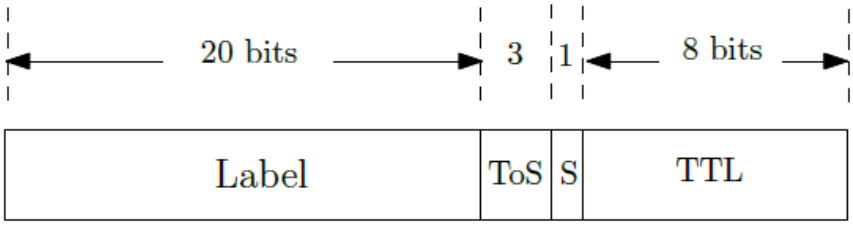

Figure 2.1: Structure of a record in an MPLS header [5] .

2.1.2 *Label Switching*

There are two different approaches to packet forwarding, each mapping to a specific structure of the forwarding table. They are called forwarding by network address and label switching.

2.1.3 *Forwarding by network address*

The most intuitive approach is forwarding by network address, which is an approach of IP. When a packet arrives at a router, the router parses the destination address from the packet header and looks it up in its forwarding table.

2.1.4 *Forwarding by label switching*

An alternative approach is known as label switching. Essentially, while forwarding by network address requires that the egress interface be chosen based on the destination of the packet, label switching requires that such an interface be chosen based on the flow the packet belongs to, where a flow corresponds to an instance of transmission, i.e., a set of packets, from a source to a destination and is identified by a tag (called label) attached to each packet of the flow see Figure 2.2 below.

- a label field (20 bits), which carries the label value;

2.2 BENEFITS OF MPLS VPN

Virtual private network (VPN) services based on MPLS are the key business services of Internet service providers (ISPs) for connecting geographically distributed sites of its customers through L2 and L3 VPN. Hence, MPLS has shown an increasing market since the late 90s [6] and also in today network, and will be in future.

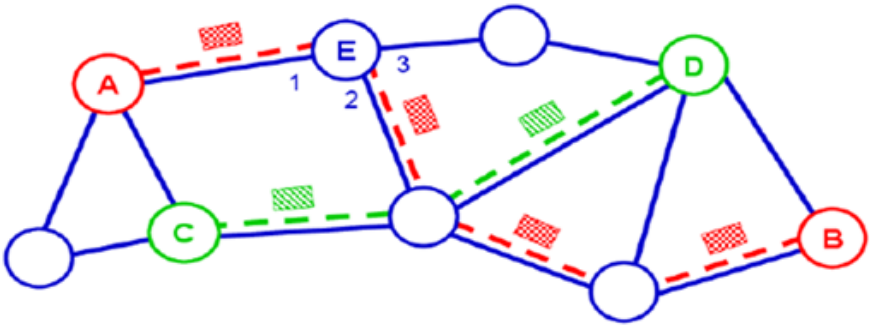

Figure 2.2: A label switching network (where labels are not swapped at each hop) [5] .

According to [5] MPLS VPN is the most deployed and popular once, and the reason for its popularity strongly depends on its ability to meet the demands of customers, providers, and vendors. MPLS VPN are used by customers to connect their geographically distributed site with additional requirements that are: (i) To keep their own IP addressing plan for all the sites; (ii) To logically separate their traffic from other customers that happens when using shared infrastructure; and (iii) To have a given guaranteed quality of service [5]. Providers also benefit from MPLS VPN since they have invested a lot on resource in building the backbone network so they prefer to sell pseudo wires rather than physical connections to their customers [5]. In addition, vendors' benefits from MPLS VPN because it is not possible to easily deploy network technology without meeting the strategies of network device producers and vendors.

Despite, MPLS VPN benefits customers and providers' requirements; the existing MPLS makes providers dependent on vendors' machine. Additionally, according to [6] there are lot of concern regarding to networking technology and protocol involved in its operation (MP-BGP, OSPF, LDP, etc.). Thus, it makes [6]: i) VPNs difficult to provision, configure, manage, and troubleshoot. ii) Hard to cast a predictable behavior from the complex interactions of these technologies, especially considering that configurations are distributed on several devices. And iii) It's controlling to routing and traffic engineering decisions methods to be limited and inconvenient. Generally, the existing MPLS VPN services have limitations like: management complexity, not scalable and also make the network requires expense and vendor specific equipment's network equipment's [7]. In contrast, ISPs seek for simplicity of provisioning and configuration, trouble-free manage-

ment, flexibility in accommodating increasingly complex customer requirements, controllability, and predictability.

2.3 SOFTWARE DEFINED NETWORKING VPN

Service providers' are highly advised to implement SDN technology in their core network. This is because [3] Software-Defined Networking (SDN) is an emerging networking paradigm that gives hope to change the limitations of current network infrastructures. Therefore, SDN helps to improve the existing network limitation since [3] first, it breaks the vertical integration by separating the network's control logic (the control plane) from the underlying routers and switches that forward the traffic (the data plane). Second, with the separation of the control and data planes, network switches become simple forwarding devices and the control logic is implemented in a logically centralized controller (or network operating system 1), simplifying policy enforcement and network (re)configuration and evolution [3].

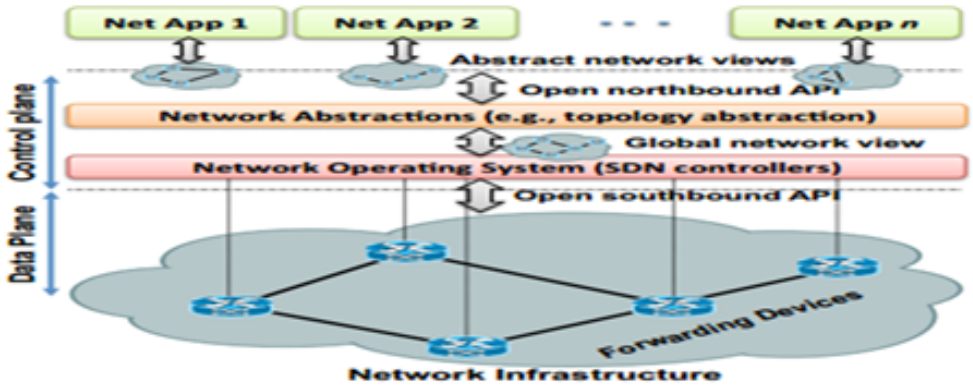

Figure 2.3: Simplified SDN architecture [3].

In the above architecture, according to [3] Forwarding devices are programmed by control plane elements through well-defined SI embodiments. And, the control plane can therefore be seen as the "network brain". In addition, all control logic rests in the applications and controllers, which form the control plane. Hence, the service provider is free to deploy any hardware from any vendor.

According to [8] the existing MPLS networks of acSP architecture is divided in to parts such as core region and the edge (Provider edge). The Provider Edge (PE) devices are where the customers are connected. Whereas, the core network consists of routers using MPLS technology for forwarding traffic among the PEs, and also to connect customer's sites to the provider's network through Customer Edge (CE) devices. However, the existing SPs network architecture which provides MPLS service has its own problems to service provider that are: complex management, not scalability and increase CAPEX (network equipment's supplier are vendors of the network). To solve these problems, new design using the SDN technology called SDxVPN has emerged. Here is below the architecture:

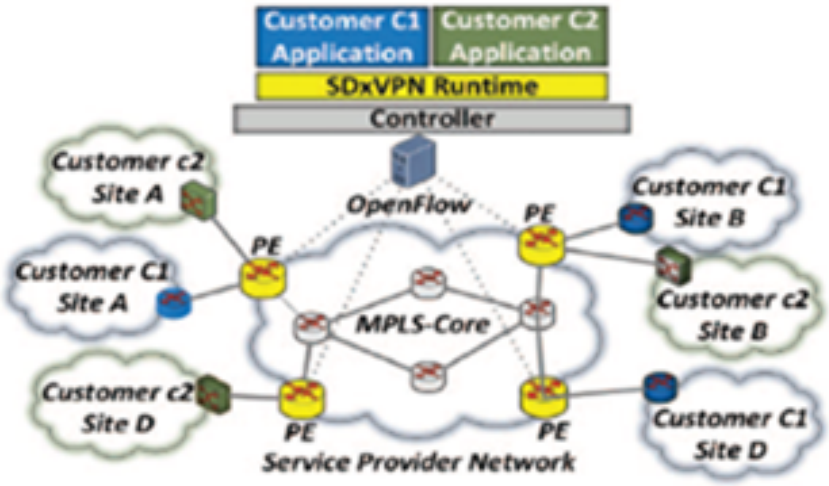

Figure 2.4: Design perspective of SDxVPN [8].

Now SDxVPN helps to resolve the problems mentioned on the existing MPLS VPN and Virtual Private LAN Service (VPLS) services of Service Providers. First, when using SDxVPN approach, each customer has its own network application so that, the customer can simply provision and maintain VPN services by its own. Hence, the service provisioning/maintaining process of service provider which has exposed to problem handled by the customer itself. Second, the control and data planes are separated therefore the SPs are not obligated to buy expensive device for scaling up their network; rather they can use their existing. Third, the network becomes scalable because distributed control planes protocols are running on PEs [8].

In traditional, MPLS VPN network of SP: MPLS relies on a variant of BGP called Multi-Protocol BGP Multi-Protocol Border Gateway Protocol (MP-BGP) to distribute reachability information about customer prefixes [5]. In contrast, in Figure 2.4 shown above; The CE devices are connected to PE device to have VPN network. And, PE routes are change to software-defined switches controlled by logically centralized SDxVPN controller via Open Flow; therefore, PE to PE control plane protocols like MP-BGP and their required attributes such as Route Distinguisher (RD) and Route Target (RT) will be eliminated; rather each customer (C1 and C2 in the Figure 2.4) has a dedicated Software-defined application which operates on top of the SDxVPN runtime and these applications are important for provisioning and managing his/her services [8].

According to [8] the newly SDxVPN architecture of control and data plane which helps to eliminate the problems investigated in the existing MPLS/VPLS VPN architectures such as: management complexity, scalability and require expensive devices. The below control plane architecture presented in Figure 2.5, helps us to avoid the limitation of existing architecture.

In Figure 2.5 shown above, Customer and Core-Mediator applications are implemented independent of the data plane protocol, running in parallel on top of the SDxVPN Runtime, and communicate with the lower layer using Restful APIs. SDxVPN runtime has three main responsibilities: slicing the network for higher-level applications, discovering the topology and providing each application with proper adapters for data plane communication [8]. In this architecture, each customer passes the service specification to his dedicated application in which the appropriate forwarding tables of the virtual devices will be constructed. Later, the forwarding tables will be passed to the Open Flow adapters which are in charge of handling inter-communication between network applications and the Open Flow PE switches [8].

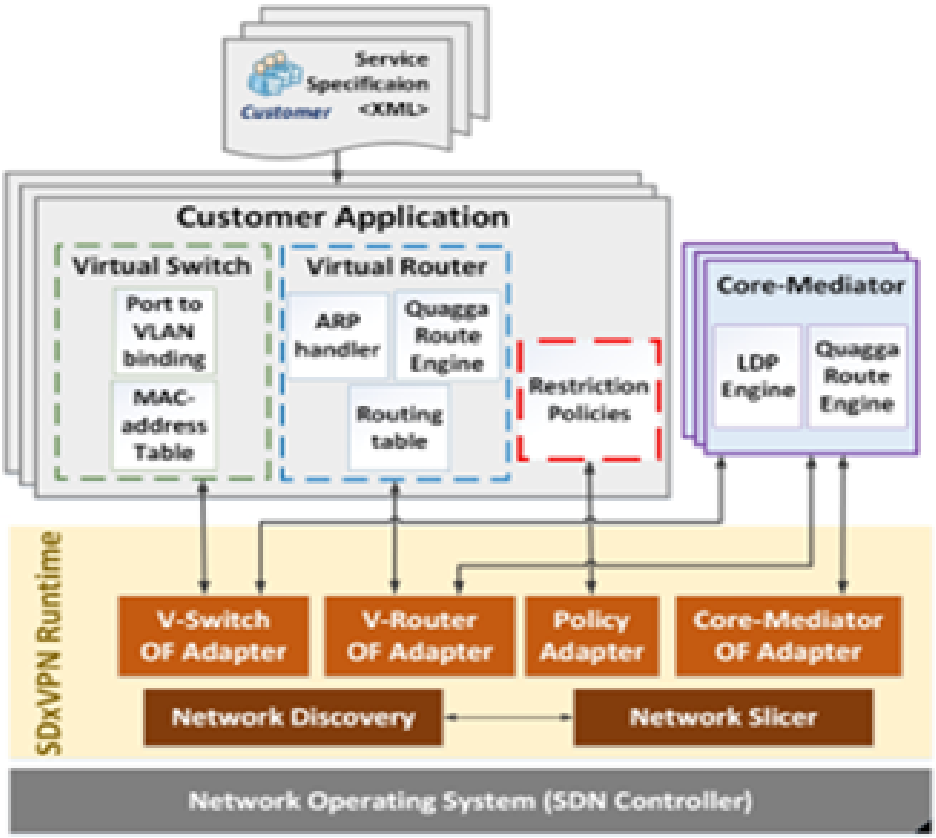

Figure 2.5: SDxVPN controller Architecture [8].

2.4 QUAGGA ROUTING ENGINE

It is a package of Unix/Linux software implementing a number of common network routing protocols like ''Route Information Protocol'' (RIP), Open Shortest Path First, the Border Gateway Protocol (BGP) and IS-IS [9]. The package also includes a routing-information management process, to act as intermediary between the various routing protocols and the active routes installed with the kernel. A library provides support for configuration and an interactive command line interface. Quagga is available under the terms of the GPLv2 license [9].

Quagga consists of a set of processes communicating via Interim Payment Certificates (IPC), as shown in Figure 2.6. Network routing protocols such as BGP, OSPF and IS-IS are run in processes such as bgpd, ospfd, ospf6d, isisd and so on. One daemon process, called "zebra" acts as an intermediary between the kernel's forwarding plane and these routing-protocol processes. In addition, there is an

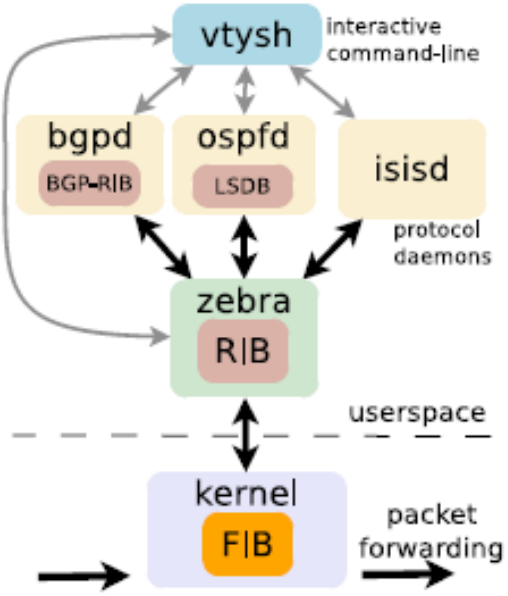

Figure 2.6:]

Quagga Architecture [9].

interactive, command-line tool called "vtysh", which allows network professional to monitor all the above process and modify the configuration.

Table 2.1: Quagga daemons [9]

IPv4	IPv6	
Zebra		FIB/Kernel arbitration
babeld		BABEL wirless mesh protocol
bgpd		BGPv4+
isisd		IS - IS
ospfd	ospf6d	OSPFv2 and OSPFv3
ripd	ripngd	RIPv1/v2, and RIPng

2.4.1 *Architecture*

Zebra process used to maintain a shadow copy of packet forwarding related state such as the network interface and the table of currently active routes (or it is often called Forward Information Base (FIB)). Specifically, the kernel is managing packet forwarding and so the kernel maintains these. However, it is possible to customize zebra to interact with other forwarding engines. Zebra process also collect routing information from routing-protocol process and store these with shadow copy of FIB. It is also responsible for selecting the best route to the destination and updates the FIB to the best route.in addition, this best route information distributed to the protocol daemons. The BGP routing protocol is not usual, so it possible for active **BGP!** (**BGP!**) speakers to not involved in forwarding packets. Hence, bgpd daemon may be run on its own without any zebra daemon. Routing processes communicate with the zebra process through a protocol ZServ. Whereas, Vtysh command line tool communicate with other processes using a simple string passing protocol.

2.4.2 *Support Library*

A C programming language support library called libzebra provides several facilities, both general and networking related. Table 2 below shows overview of modules and the library

The existing Quagga code multi-processing is taken place by using distinct processes with IPC between processes. And each process has co-operation threads with it. The "if" networking interface module can be kept automatically update with the underlying system, if used with the zclient module.

The co-operative threads facility provides support for a number of types of tasks such as Timer-task (for task schedule and reschedule), Read and Write task to access files.

Quagga Library functionality

The zebra daemon can manage FIB in forwarding plane other than the kernel.

Basic Data Structures	linklist	
	hash	Hash table
	pqueue	Priority queue
	vector	Vector / growable array
Checksums / Hashes	checksum	IP header and Fletcher (TCP, IS-IS) checksums.
	jhash	Jenkins' hash
	md5	MD5 cryptographic checksum (obsolete)
Networking Data Structures	if	Network interface state
	prefix	Network address prefix
	stream	Bounded & checked, network byte-order aware, packet buffer
	table	Longest-matching prefix lookup table support
Configuration & UI	command	Configuration & monitoring command definition support
	vty	
I/O	buffer	Low-level buffer for output.
	network	Basic read/write convenience functions
	sockopt	Convenience functions for socket operations
Filtering	filter	IP address/prefix access-list filters
	plist	IP prefix list filters
	routemap	Match/action filtering
Co-operative threads	thread	Event, I/O and timer co-operative threads
	sigvent	Signal handlers
	workqueue	Work queues
Process support	daemon	Process daemonisation
	log	Logging, to UI and/or file.
	memory	Memory allocation, debug & tracking
	pid_output	Locked PID file writing
	privs	Least-privilege support, on certain systems
IPC	agentx	SNMP AgentX interface
	smux	SNMP SMUX interface (obsolete)
	zclient	ZServ client interface

Figure 2.7: Quagga library functionality [9].

In addition, zebra can connect to other agent managing forwarding plane via socket using the protocol described in "fpm/fpm.h". The ospd daemon implements OSPFV2 and announces link state through OSPF-API IPC interface.

Internals overview

The bgpd BGP daemon internally contains all its data structure off a global, refers to event handling sub-system and work queues.

Applications

1. One application is using BDP as route-server acting as a hub for processing BGP routes and so avoiding the need for clients of the route-server to all connect with each other.

2. Another use is the data-logging of routing protocol for later analysis.

3. Can also be used with virtualization (KVM/QEMU, openv2, VMware etc.) to cheaply simulate network scenarios.

3

SOFTWARE-DEFINED NETWORKING

Software Defined Networking (SDN) is the newly proposed paradigm for drastically changing the way the existing networks are working. The basic principle of SDN lies on the separation of the network control and forwarding planes; then an external SDN controller can dynamically inspect rules into SDN capable nodes. Based on these rules, the SDN capable nodes perform packet inspection, manipulation and forwarding. In addition, the underlying SDN capable nodes (can be Open flow switches, OpenVswitch or Virtual Routers etc.) can inspect and modify packet headers at different levels of the protocol stack, from layer 2 to application layer [1].

3.1 SDN PROTOCOLS

3.1.1 *Open Flow*

Open Flow considered as the first SDN standard, and it defines an open protocol that enables an SDN Controller to interact with the forwarding plane of network devices [2]. It is a communications protocol that gives access to the forwarding plane of a network switch or router over the network. SDN Controller pushes down changes to the switch/router flow-table allowing network administrators to partition traffic, control flows for optimal performance, and start testing new configurations and applications [10]. Benefits of Open Flow:

- Programmability
- Enable innovation/differentiation

- Accelerate new features and services introduction
- Centralized Intelligence
- Simplify provisioning
- Optimize performance
- Granular policy management

Abstraction:-Decoupling of Hardware, Software, Control plane, forwarding, and Physical and logical config.

Figure 3.1: Flow-Table Entries [10].

3.1.2 *NETCONF*

NETCONF refers to a device (physical or virtual) configuration and management protocol which allows for configuration, data retrieval and event notification [2]. In addition, NETCONF provides mechanisms to install, manipulate, and delete the configuration of network devices. Its operations are realized on top of a simple Remote Procedure Call (RPC) layer. The NETCONF protocol uses Extensive Markup Language (XML) based data encoding for the configuration data as well as the protocol messages. NETCONF is designed to be a replacement for CLI-based programmatic interfaces, such as Perl and it is expected over Secure Shell (SSH). NETCONF is usually transported over the SSH protocol, using the "NETCONF" sub-system and in many ways it mimics the native proprietary CLI over SSH interface available in the device [11].

3.1.3 *OF- Config*

The Open Networking Foundation (ONF) is a specification for configuration of devices in an Open Flow network. It specifies NETCONF over TLS (Transport Layer Security) with XML data models [2]. Therefore, the Open Flow Management and Configuration Protocol (OF-Config) is a special set of rules that defines a mechanism for Open Flow controllers to access and modify configuration data on an Open Flow switch. One of the most common methods of implementing a software-defined network (SDN) is to decouple the control plane from a physical network and place management in a centralized controller. Often, that controller uses Open Flow as a southbound protocol to direct specific flows between nodes on the network. Although Open Flow determines how packets are forwarded between individual sources and destinations, it doesn't provide the configuration and management functions that are needed to allocate ports or assign IP addresses. Open Flow configuration protocols, like OF-Config, help with this and give network engineers an overall view of every area of the network. It also provides engineers with the ability to set policies and manage traffic across devices. In addition, it also used to manage physical and virtual switches in an Open Flow environment [12].

3.1.4 *XMPP*

Extensible Messaging and Presence Protocol (XMPP) is used by some (i.e. Juniper) as an alternative to Open Flow and Preferred by some due to its maturity as a protocol [2]. XMPP is a set of open technologies for instant messaging, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware, content syndication, and generalized routing of XML data. It was originally developed in the Jabber open-source community to provide an open, decentralized alternative to the closed instant messaging services at that time. These protocols are free, open, public, and easily understandable; in addition, multiple implementations exist in the form of clients, servers, server components, and code libraries.

Internet Engineering Task Force (IETF) has formalized the core XML streaming protocols as an approved instant messaging and presence technology.

3.1.5 *Op-Flex*

A Cisco proprietary protocol proposed as an alternative to Open Flow [2]. OpFlex is an extensible policy protocol designed to exchange abstract policy between a network controller and a set of smart devices capable of rendering policy. OpFlex relies on a separate information model understood by agents in both the controller and the devices. This information model, which exists outside the OpFlex protocol itself, must be based on abstract policy, giving each device the freedom and flexibility to render policy within the semantic constraints of the abstraction. For this reason, OpFlex can support any device, including hypervisor switches, physical switches, and Layer 4 through 7 network services [13].

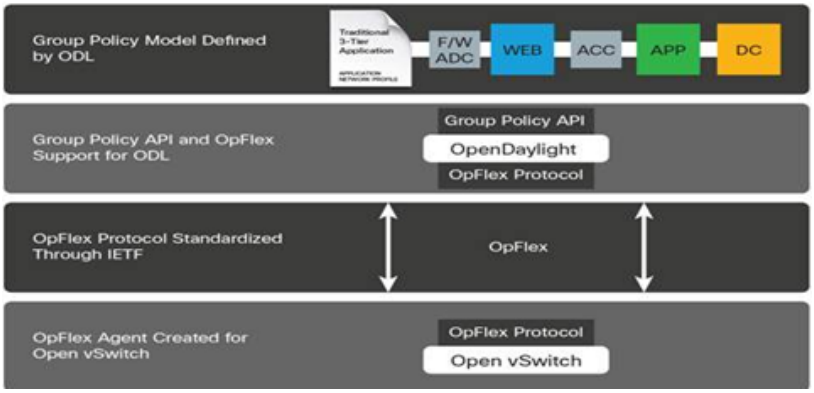

Figure 3.2: Op_flex protocol [13].

3.2 BASIC SDN PRINCIPLES

The Four basic principles for classifying a technology as SDN are:

1. Separation of Control- and Forwarding Plane externalization of the control plane from a network device to an external control plane entity often called the controller [14]

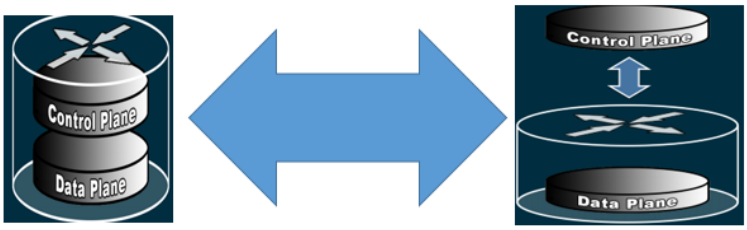

Figure 3.3: Control and data plane **[nagy2014utilizing**.

In Figure 3.3 above, in the traditional network the control and data plane are coupled together with in a device, where as in the SDN technology the control and data planes are decoupled. Therefore, the underlying network device makes only packet forwarding and the controller placed above plays the control role.

2. Logically Centralized Control: The controller of an SDN network is a logically centralized entity: it can consist of multiple physical or virtual instances behave like a single component [14].

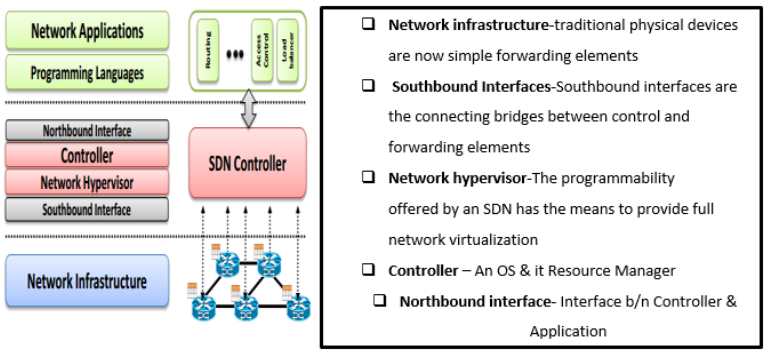

Figure 3.4: SDN controller [14].

3. Open Interfaces: For SDN to reach its full potential in terms of flexibility and adaptability, it is fundamental that its interfaces are and remain open. A closed or proprietary interface limits component exchangeability and innovation [14].

4. Programmability: The most important principle of SDN is the programmability of the network. Ability to treat the network as a single programmable entity instead of an accumulation of devices that have to be configured individually [14].

3.3 BASIC SDN ARCHITECTURE

In the architecture shown in Figure 3.5 below, we see the control layer and the application layer with the northbound and southbound Application Program Interface (API)s (Application Programming Interfaces) between those layers, and the eastbound and westbound APIs with other controllers. The northbound interface facilitates communication between the application level and the controller. Its purpose is to describe the needs of the application and to pass along the commands to orchestrate the network [1].

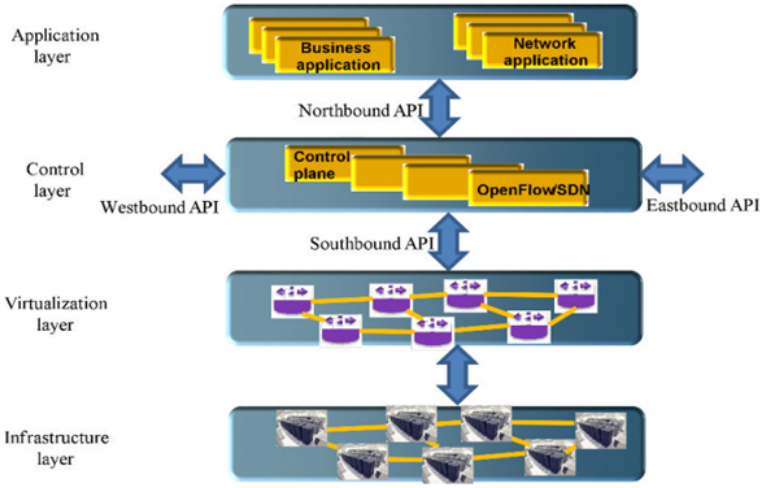

Figure 3.5: SDN architecture [1].

The southbound interface (SBI) describes the signaling necessary between the control plane and the virtualization layer. With this aim in mind, the controller must be able to determine the elements that will make up the software network for which it is responsible. In the other direction, the current network resource consumption must be feedback so that the controller has as full a view as possible of the usage of the resources. The bandwidth necessary for the feeding back of these statistics may represent a few percent of the network's capacity, but this is crucial for optimization which will improve performance by much more than a few percent **[Foukas2016**. In addition to the two interfaces described above, there are also the eastbound and westbound interfaces. The eastbound interface enables two controllers of the same type to communicate with one another and make

decisions together. The westbound interface must also facilitate communication between two controllers, but ones which belong to different sub-networks. The two controllers may be compatible but they may also be incompatible and in this case, a signaling gateway is needed [1].

3.4 MODELS OF DEPLOYMENT FOR SDN

For the practical deployment of SDN, three different possible models can be approached:

1. Switched-based model:- refers to replacing entire traditional network with SDN network, and having a centralized control system for each network element.
2. Overlay model-SDN:- end nodes are virtual devices that are part of hypervisor environment. This model controls virtual switches at the edge of a network, i.e., computing servers that set up path across the network as needed. It would be useful in cases when SDN network responsibility is handled by server virtualization team, and its limitations include debugging problems, bare metal nodes and overhead for managing the infrastructure.
3. Hybrid:-model-combines the first two models, and allows a smooth migration towards a switch-based design. The devices that do not support overlay tunnels such as bare metal servers are linked through gateways in this model [11], [15].

3.5 SDN APPLICATION AREAS: TELECOM PERSPECTIVE

3.5.1 *Traffic engineering*

The wide area network (WAN) that connects the datacenters of cloud providers is critical for the performance of Internet services. WAN links are very expen-

sive, and to guarantee the required performance WAN routers consist of high-end, specialized equipment. To compound the problem, providers are unable to fully leverage their high investment on the infrastructure. Given the extreme lack of efficiency of these inter-datacenter networks they are provisioned with an average utilization of 30-40%. Recognizing this problem, Google and Microsoft have deployed large-scale SDN infrastructures for boosting the utilization of their inter-datacenter links. These systems – Google's B4 and Microsoft's SWAN – leverage on SDN to substantially increase the utilization of their WAN links. In particular, the logical centralization of network control enables centralized traffic engineering and simpler traffic prioritization, making it possible to have these links used up to nearly 100% utilization without impairing the quality of service. Besides avoiding link usage inefficiencies, the network equipment used in these solutions is built from merchant switch silicon, further reducing costs and increasing flexibility [16].

3.5.2 *Mobile Edge Computing (MEC)*

MEC allows developers and content providers to deploy their services over the network edge. This presents many benefits including ultra-low latency, high bandwidth and real-time access to radio network information which can be used by applications for optimization purposes. Such applications are expected to be deployed in a centralized manner and therefore doing this using the conventional LTE architecture becomes a challenging task. Moreover, their deployment assumes the existence of a programmable network, which is naturally enabled by Flex Radio Access Network (RAN) **Foukas2016** .

3.5.3 *RAN Sharing & Virtualization*

A side effect from the densification of cells is the increase of the infrastructural CAPEX and OPEX. This leads to the creation of new business models, where multiple Mobile Network Operators (MNOs) share the same passive infrastructure such as masts and backhaul links in order to save costs. On top of that, a second

level of active sharing can happen, where MNOs share the network equipment as well as provide wholesale access to Mobile Virtual Network Operators (MVNOs), allowing them to provide voice and data services using part of the available resources [17], [18].

4

CHAPTER FOUR: SIMULATION SETUPL

The work aim to show SDN VPN network throughput and latency is better than that of the existing VPN with Legacy routers and switches. The traffic is generated on both networks using client-server architecture.

- The simulation tool used to simulate the existing VPN network is MiniNAM 1.0.1 Open source GUI based tool. The main reason to choose this tool is its capability to simulate the legacy router and legacy switch.
- For the case of SDN VPN mininet 2.3.0d4 Open source emulating tools have been used.

4.1 OPEN SOURCE FLOODLIGHT 1.2 **sdn!** (sdn!) NETWORK CONTROLLER

To controller the underlying SDN VPN network, floodlight 1.2 controllers have been used. Since, floodlight controller is java based and run with java flat form, java 8 is installed with the following specification.

```
goytomsdn@goytomsdn-VirtualBox:~$ java -version
java version "1.8.0_181"
Java(TM) SE Runtime Environment (build 1.8.0_181-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)
```

Figure 4.1: Java –version

```
goytomsdn@goytomsdn-VirtualBox:~$ javac -version
javac 1.8.0_181
```

Figure 4.2: Javac –version

4.2 QUAGGA ENGINE INSTALLATION

To run routing capabilities for openvswitches and Legacy routers for quagga engine install and then run.

- Open ospfd and bgpd in quagga daemons. And also enable zebra because it is interactive to configure all protocols by telnet their process id.

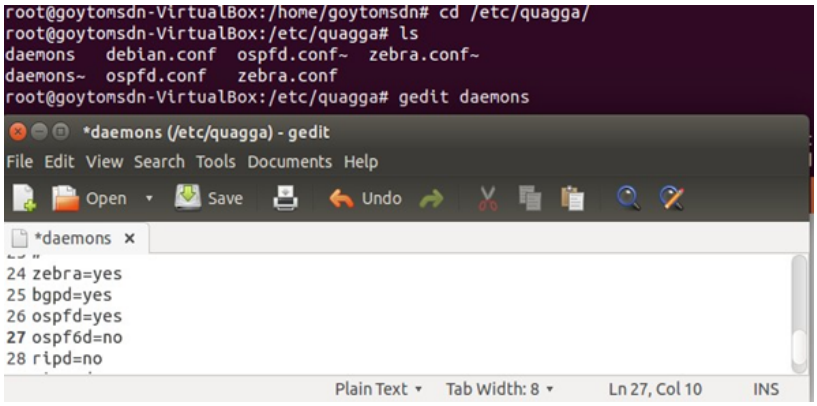

Figure 4.3: Quagga routing engine daemons setting screenshot

- Start the quagga service from /etc/init.d/quagga by using supper user privilege.

Figure 4.4: Quagga routing engine starting screenshot

4.3 EXISTING **vpn!** (vpn!) NETWORK TOPOLOGY

In Figure 4.5 the existing topology below, there are 8 linux routers and 6 switches. The router r0 and the router r1 are the border routers, and the rest routers serve as the Edge routers, with redundant links to border routers.

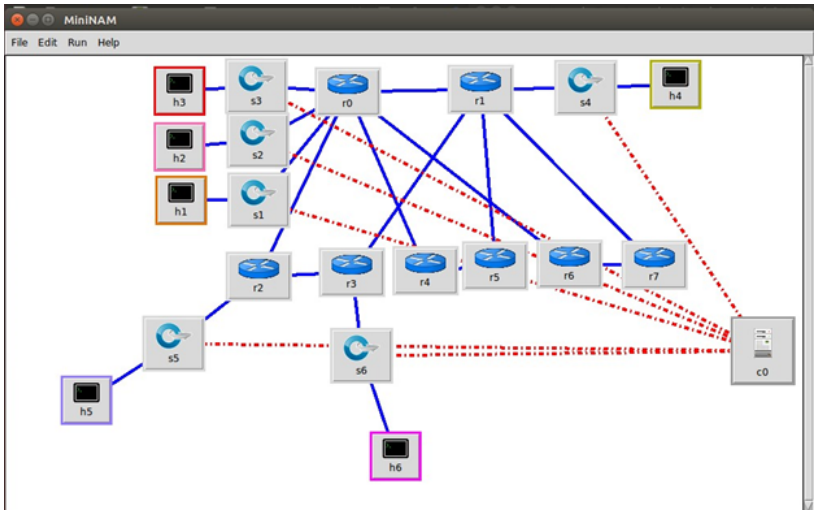

Figure 4.5: Existing topology on MiniNAM

```
goytomsdn@goytomsdn-VirtualBox:~/MiniNAM$ sudo python MiniNAM.py --config conf.c
onfig --custom etnet.py --topo mytopo --controller=remote
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 r0 r1 r2 r3 r4 r5 r6 r7
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (h6, s6) (r0, r1) (r0, r2) (r0, r4)
 (r0, r6) (r1, r5) (r1, r7) (r2, r3) (r3, r1) (r4, r5) (r6, r7) (s1, r0) (s2, r0
) (s3, r0) (s4, r1) (s5, r2) (s6, r3)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 r0 r1 r2 r3 r4 r5 r6 r7
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
*** Starting CLI:
mininet>
```

Figure 4.6: MiniNAM network creating

4.4 THE PROPOSED sdn! vpn! TOPOLOGY

As shown in Figure 4.8, the proposed SDN VPN topology contains 8 openvswitchs, these works as a L3 switches. The L3 switch so and the L3 switch s1 are the border L3 switches and the rest switches server as edge switch in place of edge routers, and also these switches have redundant links with border ones.

Two nodes s1 and s2 works as border L3 switches whereas the rest serves as Edge L3 switches, which have redundant links with the border L3 switches.

Figure 4.9 shown above presents how the mininet code is running to create SDN_VPN network.

```
192.168.1.0     *               255.255.255.0   U     0      0        0 r0-eth1
192.168.2.0     *               255.255.255.0   U     0      0        0 r0-eth4
192.168.11.0    10.10.2.1       255.255.255.0   UG    0      0        0 r0-eth5
192.168.12.0    192.168.2.1     255.255.255.0   UG    0      0        0 r0-eth4
*** Routing Table on Router: r1
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.10.14.1      0.0.0.0         UG    0      0        0 r1-eth3
10.0.0.0        192.168.2.2     255.255.255.0   UG    0      0        0 r1-eth2
10.10.2.0       192.168.2.2     255.255.255.0   UG    0      0        0 r1-eth2
10.10.13.0      192.168.2.2     255.255.255.0   UG    0      0        0 r1-eth2
10.10.14.0      *               255.255.255.0   U     0      0        0 r1-eth3
10.10.15.0      *               255.255.255.0   U     0      0        0 r1-eth5
10.10.17.0      10.10.15.2      255.255.255.0   UG    0      0        0 r1-eth5
10.10.18.0      192.168.2.2     255.255.255.0   UG    0      0        0 r1-eth2
10.10.19.0      *               255.255.255.0   U     0      0        0 r1-eth4
172.16.0.0      192.168.2.2     255.255.255.0   UG    0      0        0 r1-eth2
192.168.1.0     192.168.2.2     255.255.255.0   UG    0      0        0 r1-eth2
192.168.2.0     *               255.255.255.0   U     0      0        0 r1-eth2
192.168.10.0    *               255.255.255.0   U     0      0        0 r1-eth1
192.168.11.0    192.168.2.2     255.255.255.0   UG    0      0        0 r1-eth2
192.168.12.0    10.10.15.2      255.255.255.0   UG    0      0        0 r1-eth5
192.168.18.0    192.168.2.2     255.255.255.0   UG    0      0        0 r1-eth2
*** Routing Table on Router: r2
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.10.2.2       0.0.0.0         UG    0      0        0 r2-eth2
10.0.0.0        10.10.2.2       255.255.255.0   UG    0      0        0 r2-eth2
10.10.2.0       *               255.255.255.0   U     0      0        0 r2-eth2
10.10.13.0      10.10.2.2       255.255.255.0   UG    0      0        0 r2-eth2
10.10.14.0      10.10.2.2       255.255.255.0   UG    0      0        0 r2-eth2
10.10.17.0      *               255.255.255.0   U     0      0        0 r2-eth3
172.16.0.0      10.10.2.2       255.255.255.0   UG    0      0        0 r2-eth2
192.168.1.0     10.10.2.2       255.255.255.0   UG    0      0        0 r2-eth2
192.168.2.0     10.10.2.2       255.255.255.0   UG    0      0        0 r2-eth2
```

Figure 4.7: Sample Routing tables for the nodes created

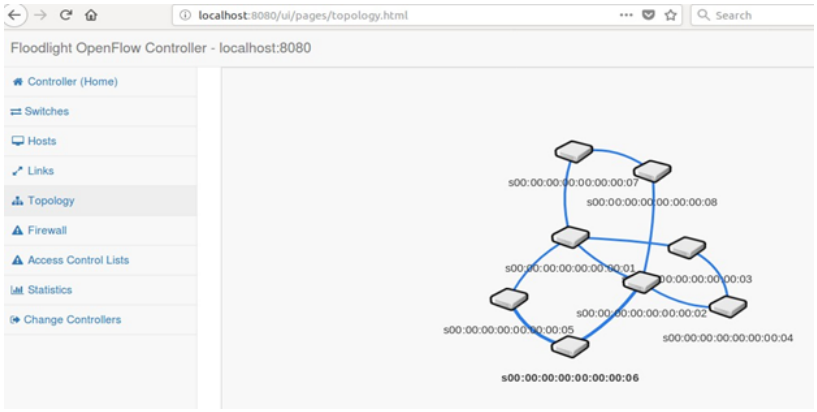

Figure 4.8: The proposed SDN network design architecture

```
goytomsdn@goytomsdn-VirtualBox:~/mininet$ sudo mn --custom ~/mininet/custom/myto
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8
*** Adding links:
(h1, s3) (h2, s4) (h3, s5) (h4, s6) (h5, s7) (h6, s8) (s1, s2) (s1, s3) (s1, s5)

*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 8 switches
s1 s2 s3 s4 s5 s6 s7 s8 ...
*** Starting CLI:
mininet>
```

Figure 4.9: Mininet network creating

5

RESULT AND DISCUSSION

To compare the throughput and latency on both networks, two nodes where selected. The selection criteria were the two nodes should not be adjacent nodes. Finally, using client-server architecture and mininet iperf tool traffic were generating between two nodes for 30 sec.

5.1 sdn! BASED vpn! THROUGHPUT COMPARISON WITH EXISTING vpn! GRAPH

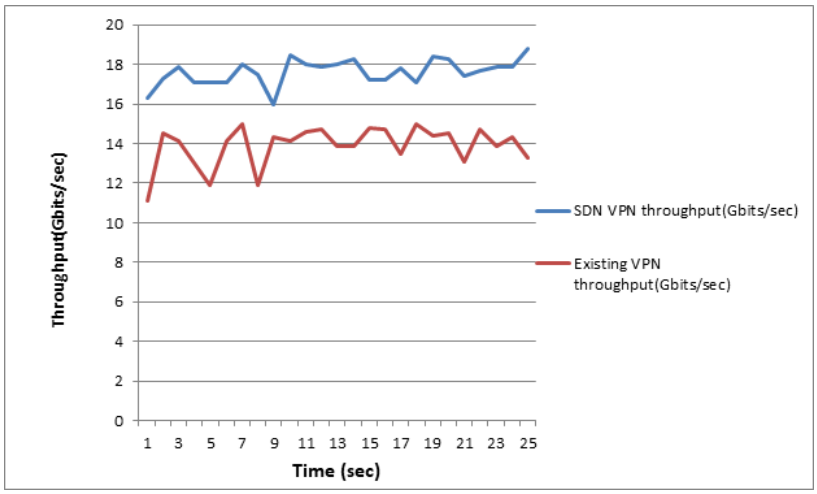

Figure 5.1: SDN VPN throughput vs Existing VPN

In the above Figure 5.1, from 8 switches nodes of the SDN-VPN network, two of them were selected. Following that, the traffic is generated between these nodes using client-server architecture. In the same way two nodes of Linux routers were selected and traffic generated. Finally, by using iperf on mininet Command Line Interface (CLI) the throughput pass on these nodes is extracted for 30 sec. The result shows 26.1 Gigabytes of throughput has been transferred between these nodes.

5.2 LATENCY GRAPH BETWEEN sdn! vpn! AND NON-sdn! vpn! NETWORK

The latency graph extracted by running client-server architecture on two different nodes for the existing and SDN-VPN networks, the using by using iperf on mininet the latency of both networks is extracted. Below is the latency result comparison between to networks.

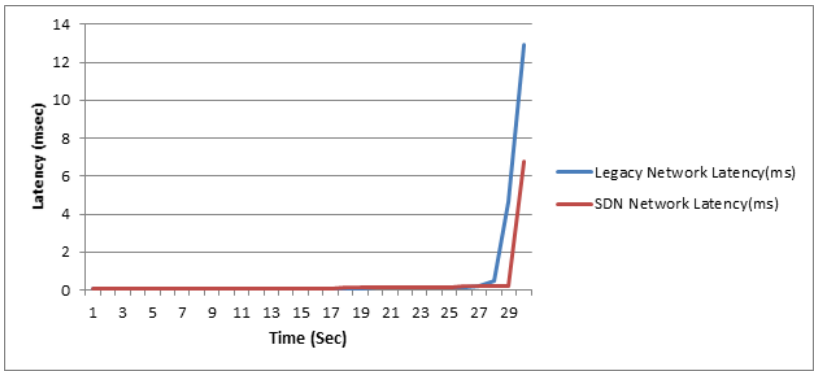

Figure 5.2: Latency comparison between Legacy (Existing VPN) and SDN VPN network

As shown in Figure 5.2 above, the time require to transmit a data sent from one node to the other side node called latency is lower for SDN-VPN network which is red colored, when compared with the existing VPN network.

6

CONCLUSION AND FUTURE WORK

In this thesis the SDN-VPN methodology was proposed, traffic was generated between two nodes on the Existing and SDN-VPN. The throughput gained in Figure 5.1 shows the new SDN-VPN network throughput average is 17.5 Gbit/sec, whereas the existing VPN average throughput is 14.0 Gbit/sec. hence, it show that the SDN VPN throughput is 22.2222% higher than that of the existing VPN network. The main reason, to get better throughput on SDN-VPN is that the network switches on this network become simple forwarding devices; in addition, the control logic is implemented in a logically centralized controller (or network operating system [3].

The routing between any nodes of SDN-VPN has been controlled by a dedicated openflow floodlight controller. Hence, the latency of this network is lower than the existing VPN network by 70.4675%, because the average latency for SDN VPN is 0.328266667 msec which is less than the existing VPN latency 0.6855 msec. In contrast, the existing VPN network has 70.4675% higher latency with respect to the SDN one, this is because every routing has been made independently within the controller plane of each nodes.

In this thesis work, the traffic is generated using client-server architecture between two nodes. Therefore, there is no other connection burden on connected network nodes. The result may be different if all nodes generate traffic between them; hence, it is advisable to generate traffic between all nodes and check the result found with this thesis work. In addition, this works only use throughput and latency parameters to compare the two networks, for future it is advisable to compare these networks using Graphical Network Simulator (GNS3) version 1.5.0 and above for different legacy routers simulation with Virtual machine cloning OpenvSwitch.

REFERENCE

[1] G. Pujolle, *Software Networks: Virtualization, SDN, 5G and Security*. John Wiley & Sons, 2015.

[2] community. (Nov. 12, 2017). Understanding-terminologies-and-protocols-ossdn. [Online; 2017-11-12], [Online]. Available: `https://community.fs.com/blogs/understanding-terminologies-and-protocols-ossdn.html/`.

[3] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[4] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," *Computer Networks*, vol. 75, pp. 453–471, 2014.

[5] L. Cittadini, G. Di Battista, and M. Patrignani, "Mpls virtual private networks," *Recent Advances in Networking*, vol. 1, pp. 275–304, 2012.

[6] S. Salsano, P. L. Ventre, L. Prete, G. Siracusano, M. Gerola, and E. Salvadori, "Oshi-open source hybrid ip/sdn networking (and its emulation on mininet and on distributed sdn testbeds)," in *Software Defined Networks (EWSDN), 2014 Third European Workshop on*, IEEE, 2014, pp. 13–18.

[7] G. Lospoto, M. Rimondini, B. G. Vignoli, and G. Di Battista, "Rethinking virtual private networks in the software-defined era," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, IEEE, 2015, pp. 379–387.

[8] B. Mirkhanzadeh, N. Taheri, and S. Khorsandi, "Sdxvpn: A software-defined solution for vpn service providers," in *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, IEEE, 2016, pp. 180–188.

[9] P. Jakma and D. Lamparter, "Introduction to the quagga routing suite," *IEEE Network*, vol. 28, no. 2, pp. 42–48, 2014.

[10] sdxcentral. (Nov. 12, 2017). What-is-openflow. [Online; 2017-11-12], [Online]. Available: https://www.sdxcentral.com/sdn/definitions/what-is-openflow/.

[11] S. Wallin and C. Wikström, "Automating network and service configuration using netconf and yang," in *Usenix Large Installation System Administration Conference: 04/12/2011-09/12/2011*, USENIX-The Advanced Computing Systems Association, 2011, pp. 267–279.

[12] searchsdn. (Dec. 14, 2017). Of-config-openflow-configuration-and-management-protocol. [Online; 2017-12-14], [Online]. Available: http://searchsdn.techtarget.com/definition/OF-Config-OpenFlow-Configuration-and-Management-Protocol.

[13] xmpp. (Nov. 12, 2017). Technology-overview. [Online; 2017-11-12], [Online]. Available: https://xmpp.org/about/technology-overview.html.

[14] M. Jarschel, T. Zinner, T. Hoßfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, attributes, and use cases: A compass for sdn," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 210–217, 2014.

[15] M. Brandt, R. Khondoker, R. Marx, and K. Bayarou, "Security analysis of software defined networking protocolsopenflow, of-config and ovsdb," in *The 2014 IEEE Fifth International Conference on Communications and Electronics (ICCE 2014), DA NANG, Vietnam*, 2014.

[16] F. M. Ramos, D. Kreutz, and P. Verissimo, "Software-defined networks: On the road to the softwarization of networking," *Cutter IT journal*, 2015.

[17] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "Flexran: A flexible and programmable platform for software-defined radio access networks," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, ACM, 2016, pp. 427–441.

[18] G. Ferro, *Openflow and software defined networking*, 2014.