Comparative Performance Analysis of SDN vs Traditional Networks using SDN Controller

by

Md. Sakib Shahriar
19101361
Jahid Hossain Sabit
19101344
Akib Ahmed
22241175
Amiruzzaman
22141068

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
School of Data and Science
Brac University
January 2023

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.

2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.

3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.

4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Md. Sakib Shahriar

19101361

Jahid Hossain Sabit

19101344

Akib Ahmed

22241175

Amiruzzaman

22141068

Approval

The thesis titled "Comparative Performance Analysis of SDN vs Traditional Networks using SDN Controller." submitted by

1. Md. Sakib Shahriar (19101361)
2. Jahid Hossain Sabit (19101344)
3. Akib Ahmed (22241175)
4. Amiruzzaman (22141068)

Of Fall, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 19, 2023.

Examining Committee:

Supervisor:
(Member)

Amitabha Chakrabarty, PhD

Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi

Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

New difficulties have been presented to the future of the Internet by emerging developments in information and communication technology (ICT) and numerous Internet services, with high priority given to dynamic maintenance and high bandwidth. Considering traditional networking techniques, where device setups are conducted manually and the network infrastructure's capacity is not completely exploited, the high bandwidth required for transmitting enormous amounts of data cannot be attained. SDN, or software defined networking, is a new approach to solving these issues. This study compares and contrasts the performance of conventional networks versus SDN. By contrasting round-trip propagation latency between end nodes and maximum attainable throughput between nodes in traditional as well as Software-defined network environments, a performance study for fundamental and suggested network topologies is conducted using simulations carried out in Mininet.

Keywords: SDN; Networks; Tree; Linear; Packet Drop; Ping; RTT

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.
Secondly, to our advisor Dr. Amitabha Chakrabarty sir for his kind support and advice in our work. He helped us whenever we needed help.
And finally to our parents without their throughout sup-port it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ACL Access Control List

AI Artificial Intelligence

API Application Programming Interface

ASIC Application-Specific Integrated Circuit

BGP Border Gateway Protocol

CLI Command Line Interface

CPU Central Processing Unit

DDoS Distributed Denial of Service

DHCP Dynamic Host Configuration Protocol

DNS Domain Name Service

DOS Disk Operating System

DTE Data Terminal Equipment

GUI Graphical User Interface

ICT Information and Communication Technology

IDS Intrusion Detection system

IOT Internet Of Things

IP Internet Protocol

IPS Intrusion Prevention System

*IPv*4 Internet Protocol version 4

LAN Local Area Network

MPLS Multi Protocol Label Switching

NAC Network Access Control

NIB Network Information Base

$NS3$ Network Simulator 3

NTP Network Time Protocol

ODL OpenDayLight

OF Open Flow

OS Operating System

$OSPF$ Open Shortest Path First

OVS OpenVSwitch

PC Personal Computer

QoS Quality of Service

RTT Round Trip Time

SBI Service Based Interface

SLA Service Level Agreement

SSH Secure Shell Host

TC Traffic Control

TCP Transmission Control Protocol

UAS Unmanned Aerial System

UAV Unmanned Aerial Vehicle

UI User Interface

$VLAN$ Virtual Local Area Network

VM Virtual Machine

VN Virtual Network

VPC Virtual Personal Computer

$WLAN$ Wireless Local Area Network

Chapter 1

Introduction

In every nook and cranny of our life, the Internet plays a significant role. When it comes to TCP/IP transmission, the networking technologies are crucial. To get to their intended recipient, the packets of data traveling from the source pass via the Internet's routers, switches, and so on. These protocols have the responsibility of rapidly identifying connection failure and determining a different path to the target in the event of a connection failure [3].

According to [47], in today's environment, the capacity for failure recognition and remediation has grown crucial due to the rise in unanticipated breakdowns and assaults. In the same way, it is essential to move data from one origin to a certain endpoint whenever a link fails or even when modifications to the topological data take place. Knowing or being able to predict the projected optimal time needed for a system to coincide under specific conditions is essential for the prevention and/or reduction of packet or data failure. [3], [37]. Networks must prioritize the data transmission convergence duration since it is seen as a crucial performance criterion and layout objective for assessing the productivity of the navigation method [16]. Its viability for use in actual deployments increases with the speed at which the protocol's running routers aid the infrastructure in recovering throughout breakdown [35].

Providing accessibility to users so that network activities may be used effectively is one of the key goals of forthcoming Internet-based apps. Although high bandwidth, simple connectivity, and dynamic maintenance are crucial factors for the foreseeable Internet, new developments in smartphones, media platforms, cloud technology, and big data have presented new challenges [36]. The paper [50] proposes that when it comes to massive amounts of data, such as those found in data centers, traditional network solutions that rely on human setup of specialized devices are complicated, susceptible to mistake, and unable to fully leverage the capabilities of physical network infrastructure. It is hard to maintain outdated network management techniques. Configuration, bug-fixing, and adding additional devices are challenges that necessitate a significant amount of human effort. The paper [66] states that traditional networks are more challenging to administer since each network component must be configured separately by the network administrator using particular instructions, some of which may vary from provider to provider. Furthermore, deploying dynamic rules to the network is a difficult task since there is no mechanism for the network to automatically reconfigure itself in the event of load variation and failures. Current conventional networks have a close relationship between the control

plane, which controls network traffic management, and the data plane, which directs traffic in accordance with control plane decisions. The primary cause of traditional networks' architectural intricacy [22], particularly in switches and routers, is this close connection between forwarding devices and the management plane. In paper [13] the authors have indeed made an expenditure in a project that expands networks' development capabilities and lessens the requirement for changing equipment replacement in order to address scalability issues. These criteria prompted the creation of the Software Defined Networking architecture, a novel networking approach (SDN).
The concept "software-defined networking" (SDN) means a networking technique in which APIs or controllers that are based on software and connected to the underlying hardware architecture, regulate network traffic. Networks that are traditional employ traffic with embedded systems (including switches, routers, etc). That is how this paradigm of SDN differs from those networks. The paper [63] states that until 2025, the total number of IoT devices on the planet is expected to reach 25.2 billion. Software-defined networking (SDN), in accordance with [41], permits the development and administration of both virtualized systems and traditional hardware. Software-defined networking supports an innovative approach to maintaining the setup of packets of data via a centralized server, in contrast to network virtualization, which empowers institutions to stratify different simulated systems inside a single physical system or connect equipment on variety of physical systems to establish a single virtual network. [45]. The paper [33] dictates that there is no globally approved flexible network administration that fairly permits flows that goes through switches. switches in any of the present traditional core and access networks, nevertheless. Furthermore, a typical network lacks a global perspective that is necessary to maintain an appropriate throughput ratio as the quantity of flows rises. As a result, once scalability has been reached, it cannot handle new flows or users. By isolating the Control Plane's centralized Controller from the Data Plane, SDN reinvents this troublesome network management as it currently exists [24]. The need for each switch and its present flow demand might be understood due to SDN's global perspective. Thus, a more equitable distribution of immigrant flows might be achieved. In addition to improving network scalability to handle additional flow arrivals, this would maintain appropriate levels of flow throughput for each flow. The benefits of SDN over conventional networks and how it may save costs and enhance network efficiency have been demonstrated by several writers [30]. A pressing and intriguing topic for academics nowadays is the smooth conversion from conventional networks to SDN [64]. The integration connection that was present in a normal network is destroyed by SDN, which separates the control layer from the data layer. Network switches are made simpler as a result of this division and are reduced to simple forwarding devices. A new device dubbed "Controller" now houses the system cognitive plane, also known as the control plane.
The paper [33] dictates that an open-source simulation tool for SDN networks called Mininet supports this field's research and instruction. In addition to enabling software deployments on the topology utilizing an SDN controller like POX or Floodlight, the paper [14] offers a simulated network ecosystem incorporating network apparatus. In conjunction with POX and Floodlight controllers, the paper [18] covers the SDN elements on the Mininet simulation tool. According to them, the tool's functionality is heavily focused on producing outcomes in a real-world setting. Ad-

ditionally, according to prototyping, implementation, and collaboration, the paper [5] details the efficiency of Mininet, particularly in terms of time and materials. It is also simple because of the drag and drop capabilities that makes it simple to design topology. In accordance with [71] Mininet is a solution for orchestrating and simulating SDN networks. For network comprehensive simulation, it offers a portable virtualization system. It has the ability to operate a variety of hosts, switches, routers, and connections on a Linux kernel. The papers [24] and [51] state that the capabilities accessible using Mininet are as follows:

- Customized topologies can be created.
- Engaging with actual Linux OS apps in real time.
- Switches may be configured for packet forwarding.
- It is simple to use and straightforward to provide results.
- It offers a quick and inexpensive way to evaluate networks for the advancement of OpenFlow applications.
- It enables autonomous tasks on a single network topology by different researchers.
- Without even requiring a physical infrastructure, it facilitates evaluation of enormous and complex structures.
- This encompasses tools to troubleshoot and test across the network.
- This also offers straightforward Python APIs for testing.
- It endorses a wide range of topologies and contains a basic collection of topologies.

The paper [66] also proposes that a Linux Secure Shell (SSH) connection may be made to a virtual host on the mininet that functions like a genuine device and can be used to execute many programs, including Linux-installed ones. It uses the network domain system and process-based virtualization, a characteristic of Linux, to build virtual hosts that behave like distinct hardware components. In order to simulate connection speed and latency, packets can be sent through a workflow that resembles a genuine Ethernet by the programs connecting the virtual hosts. A predetermined degree of queueing is used to simulate real Ethernet switches and routers in order to handle packets. The OpenFlow switches software, known as "Open vSwitch," which is used for virtual switches, will be covered later. Utilizing virtual Ethernet pairs that are made available by the Linux Kernel, virtual connections are created between hosts and virtual switches. Link synchronization primitives and virtual Ethernet connections are used in conjunction with the virtualization capabilities of the Linux kernel, as well as CPU speed separation and network namespaces. Instead of using whole virtual machines as emulators do, this approach starts up more quickly and expands to multiple hosts.
In agreement with [66] the elements that make up a Mininet network are as follows:

1. **Isolated Hosts:** A network namespace is occupied by a collection of user-level activities that make up an emulated host. With the use of network subdomains, process groups may privately control their surfaces, ports, and routing tables. Each activity group's share of a CPU is restricted by Mininet via CPU Bandwidth Restriction.

2. **Emulated Links:** Linux Traffic Control (TC), which uses packet schedulers to bend traffic to a defined pace, determines the data rate for each link. Every simulated host has at least one virtual Ethernet port. When two virtual interfaces or virtual switch ports need to be connected, a virtual Ethernet is employed. To pretty much the entire network, including application software, each virtual connection functions as a genuine, working Ethernet port.

3. **Emulation Switches:** To switch packets between several network interfaces, Mininet employs Open vSwitch, which operates in kernel mode.

In accordance with [28] a network operating system, often known as the SDN controller, is the primary element of a software-defined network. The SDN paradigm's characteristics are determined by the controller. It is the part in charge of focusing communication across all of the network's configurable components and offering a single picture of the network. Numerous SDN controllers are available right now. The design and intricacy of the controller's programming language, along with its use in its development, are crucial for creating applications. The paper [23] focuses on a free and controller for OpenFlow and SDN, POX. POX is used to create and prototype innovative network services more quickly. The Mininet virtual machine has a pre-installed POX controller. POX Controller enables you to turn straightforward OpenFlow systems in switch, hub, load-balancing, and firewall hardware. OpenFlow/SDN studies may be carried out simply using the POX controller. To do investigation using testbeds, actual hardware, or a Mininet emulator, POX may be supplied various settings in accordance with real or experimental topologies. According to [11], the TCP dump packet capturing tool is required in order to observe and collect the messages traveling through the POX controller and OpenFlow endpoints.

The functionality of a software application and its hardware architecture are precisely reproduced through emulation, which does not allow for modeling [67]. Several thousand network engineers throughout the world use Mininet to simulate, configure, test, and debug real and virtual networks. Industrial sensor network, composite topology architecture, automated networking, and multi-protocol label switching (MPLS) are some of the notable works that have been done using Mininet. Making the shift to become modern firms that improve consumer experiences and empower employees to execute at the very highest level possible is made simpler by technologies such as VMware. A single Windows or Linux PC may run numerous operating systems as virtualized systems, thanks to VMware. Open vSwitch, sometimes known as OVS, is a global emulated multilayer switch program that is available as an open-source project. Open vSwitch's primary goal is to offer a switching layer for hardware virtualization settings whilst enabling a variety of interfaces and conventions used in computer networks. It comes hand-in-hand with VMware as well. According to the paper [56], it is possible to activate the firewall feature on the SDN switch using the Open vSwitch. In the SDN context, controllers have a significant impact. It is

an application that utilizes the Northbound API to manage all data flow between application modules and switches. The operator which oversees all network-wide managing responsibilities is known as the SDN controller, to put it briefly [46]. The paper [46] claims that a software-defined networking (SDN) solution makes use of the POX and Floodlight Controller, an open group of programmers, majority of them are representing Big Switch Systems, to build a network in the SDN hub that is deployed in a virtual box, the characteristics of POX and Floodlight controllers are evaluated in this study. Multiple controllers are packaged together in the SDN hub. That controller may be created and utilized to manage the established network depending on user needs.
In this paper we will be using Mininet, VMware, Open vSwitch, and POX controllers. Since Mininet is used for evaluating and fixing networks, both real and virtual, it helped us to virtualize real hardware devices. We can operate different topologies with only a few devices. The devices produced while we constructed topologies of traditional and SDN in Mininet utilizing the all-in-one software GUI interface had to be hosted and managed by a server function. This is where VMware came into play for our work. If you want to utilize the Mininet VM, we had two options for running it, directly on your PC through virtualization software like VMware Workstation, VirtualBox, or Hyper-V, or globally on a server via VMware ESXi or perhaps in the cloud. We went with the former. For device routing, we could utilize OpenVswitch (OVS), a virtual switch which joins virtual computers with the aid of virtual connections and ports, because it can connect virtual machines. For connecting the routers, switches, and maintaining traffic, we are using the POX SDN controller. The SDN Controller POX is in responsible of providing the infrastructure facilities with the instructions it requires to regulate traffic, and it is required to keep all network policies current and accurate.

1.1 Research Problems

The effectiveness of the network is impacted by a number of distinct characteristics of SDN controllers. It is unimportant to choose a controller based on only one attribute, therefore choose many to see how they affect network efficacy and productivity. In the study [46], one method of the decision technique is discussed in depth. The papers [14],[9],[24] state that as opposed to smaller networks, where development and implementation may be performed without the need for specialized personnel, Software defined networking infrastructure, according to the article's writers, is not a system that is easily transferrable. In contrast, the paper [18] proposes that organizations operating large-scale data networks must be concerned with implementation in software-defined networks. As a result, unplanned contact with other installed networks may increase the data rate of broadcast traffic generated by devices that are not OpenFlow compatible. The paper [51] proposes that one of the most significant difficulties with SDN is interoperability. It is necessary to provide an interface for communication between SDN and non-SDN control planes, such as MPLS, since this will boost the operational network's scalability. The amount of data flows that the SDN controller can manage depends on the open flow mechanism, which sets up the flow process every time a new flow is launched which can be an issue.
Computer networks are functionally divided into three areas: the data, control, and

management planes. The phrase data plane refers to the networking hardware which is responsible for (essentially) sending information. The paper [61] suggested, the control plane is a representation of the procedures that are employed to populate the routing columns of the data plane elements. The management plane contains the software programs necessary to continuously monitor and modify the control capabilities. The SNMP-based utilities are some examples [2]. The [34] examines that, the management plane defines network protocol, the control plane enacts it, and the data plane puts it into action by transmitting data in accordance with.
According to [29], IP networks' control and data planes have historically been closely related and a part of the same networking infrastructure. The gadgets are extensively distributed throughout the entire corporation. The result is a tremendously intricate and static architecture. Additionally, it is the basic cause of the rigidity of conventional networks. It is very difficult to maintain and regulate. The paper [29] asserts that these two characteristics are largely to fault for a vertically integrated industry that lacks creativity.
According to [68], in the modern world networks, network setup failures and their related issues are very prevalent. Very undesirable network events, such as forwarding loops, packet loss and configuration of unexpected pathways, service disruptions, or contract infringements, may arise from a single incorrectly configured device. In fact, although it happens seldom, a single improperly configured router can impair the proper performance of the entire Internet for several hours. Moreover, to ensure that all queries are processed without interruption, managing a large volume of data in the network necessitates a controller with a high CPU and memory capacity. Few companies provide unique solutions comprising specialized hardware, operating systems, and control software to enable network administration. In [68] it is said that prolonged payback on investment cycles and high capital and operating costs associated with constructing and access control, load balancing, energy efficiency, and traffic engineering are examples of novel features and applications that cannot be implemented with current networking systems.
Controller positioning, scalability, efficiency, safety, compatibility, and reliability are among the SDN difficulties [21]. One controller is responsible for managing all the networks according to the SDN centralized controller paradigm [68]. Because there is no backup controller, if the central controller fails, the entire network is destroyed [57]. According to [27], the goal of software-defined networking is to promote development and provide easy programmatic management of the network data path. Another issue that arises with SDN is scalability, since, with this method, the data and control planes are separate yet may still function autonomously so long as an API links them. From a single perspective, control plane modifications are accelerated by network changes. The necessity for a consistent API for both planes and the SDN controller, which presents a challenge as the network's number of switches and nodes increases, is one of the drawbacks of the decoupling process [40].

1.2 Research Objectives

We are aiming to get results of various networking modules by comparing traditional networks with software-defined networks. From our research, we are able to describe the performance analysis of both SDN and traditional networks.
The objectives of this research are:

- Understanding the differences between traditional and SDN networks.
- Understand and implement both traditional and SDN topologies.
- Comprehensive implementation of tools like Mininet, OpenFlow, Open vSwitch etc.
- Operation of different SDN controllers.
- Understanding the pros and cons of both traditional and SDN networks.
- Comprehension of when to transform from traditional to SDN network.
- To offer a better and more improved analysis.

Chapter 2

Literature Review

Today is the age of the internet. Our everyday work-related devices are all now connected to the internet. So, they all can be considered as IoT devices. For this, everyone needs some kind of network to connect all these devices to one another. Nowadays every industry, organization and institution have its own network. Therefore, network speed and performance need to be faster and more efficient. The traditional network has some major drawbacks. Such as implementation and adding new routers need to be set up manually. Whereas Software Defined Network makes it way much easier. As the control of the network is centralized. Moreover, the complexity of network configuration, operation and maintenance can be highly reduced. So, in terms of performance SDN can be a great solution.

2.1 Software Defined Network (SDN)

Software-defined networking (SDN) is where the network control logic is separated from infrastructure and the hardware that executes the job. This method streamlines the administration of, which may be exclusive to a single business or partitioned to be used by multiple. Controllers in SDN overlay network hardware in the cloud or on-premises. Thus, it can implement the security control on network level rather than user level. SDN helps to control the network's security in a cheaper way but ensures much more security and control over the network. As the network is programmable and software can easily update it is much more convenient than hardware. According to [15] [8], SDN has raised the network structure where the control plane logic is decoupled from the forwarding plane. SDN is structured in such a way that a center will control the whole network according to the settings it needs. [20]

2.2 SDN frameworks and its architecture

SDN's architecture can be divided into 3 layers. So, the layers in a standard SDN are:

1. **Data layer:** This is where every data is received and gets forwarded. Here the IoT devices mainly work.

2. **Control layer:** This is where all the decisions are made. Here, traffic passes through, and data is monitored and controlled by controllers. It serves as the network's brain.

3. **Infrastructures layer:** This is defined by the following switches that build up the data plane and handle data packet transportation.

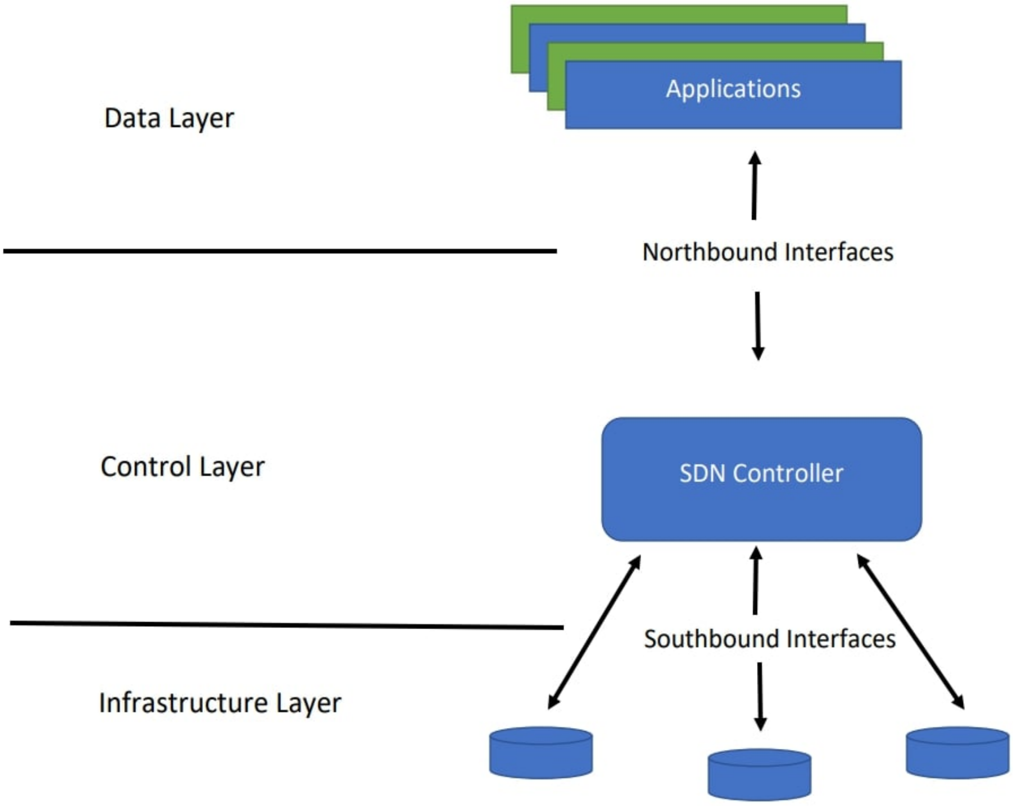

Figure 2.1: SDN Architecture

The framework of SDN consist of 4 things:

1. **SDN enabled Switch:** to use an SDN based network infrastructures every user needs to be connected with an SDN compatible switch and the switch has security policies and rules implemented.

2. **IoT devices:** devices that can access the internet are considered as IoT devices. These devices connected to SDN enabled switch and follow the rules set by SDN.

3. **Cluster SDN Controller:** Based on their geolocation, a small group of end-users is formed. a cluster could be made up of only a few SDN enabled switches. A service provider can build a group of users into a cluster in a certain area to easily enable facility and monitorability in a subspace. [43]

4. **Master SDN Controller:** According to paper [43], at the top of the SDN system, the Master Controller receives instructions only from the network operator. The Master SDN Controller manages all Cluster SDN Controllers under its. This controller can see network traffic flow and access request across the whole network.[43]

2.3 Related Works

This section will review and critique previous major work on the performance difference between traditional network and software defined network. Based on different papers we'll be able to learn the reasons for performance deficit, challenges and how to address them using many procedures and algorithms.
SDN based networks perform exceptionally greater than traditional networks on the basis of managing users as well as reducing lags. Research work [65] shows that the built-in lag of SDN based WLAN with switches is larger than the SDN based WLAN without switches. Although, after the handover, traditional networks have a higher delay than SDN. Traditional based WLANs take more time to send packets than SDN based WLANs both in terms of with switches and without switches. Therefore, the overall delay of the Traditional network is significantly higher. Moreover, the research [65] declares that when the figure of switches boost, SDN structured networks will still perform superior than the traditional network under most data rate and delay.
Traditional network is hardware established while SDN is usually software established. SDN has greater accessibility than traditional networks under most of the handling resources. One of the major benefits of SDN, it offers a platform for software with huge data. Such as virtualization, traffic modeling, simulation, big data, and cross layer architecture. Research [35] shows the comparison with the networks with or without the SDN approach. By using the SDN access, the network contracts packet loss and escalates the bandwidth. As the endowed datagrams were absent 3.3% in SDN endorsed networks. On the contrary, datagram losses in the SDN network were 16.6%. For that reason, the prospected structure can be efficient, scalable, flexible, reusable, as well as robust for contrasting applications. In short, SDN offers effective and automated network control solutions to the network's growing complexity as well as that of many other software domains.
According to paper [35], in comparison with traditional network and SDN, moderate time required for SDN (OpenFlow-enabled network) is low. Because the increasing speed of data communication and improved round-trip propagation delay impacts better speed and maximizes throughput. Again, network efficiency increases significantly by diminishing undesirable traffic and overheads of the data packet. In a traditional network, data plane and control plane are perpendicularly unified on a system core devices and have complicated designs and can be challenging to control and manage. SDN architecture solves the vertical integration issue in traditional networks by centrally controlling the data plane. Also, control plane separation from network core devices helps to manage the network more efficiently. Even for different topologies such as single, linear and tree topology, SDN (OpenFlow-enabled network) performs agile due to its chief controlled construction. The switch will preserve a document of that specific data flow entry for a length of time determined by the OpenFlow controller once it has been made in the flow table of an OpenFlow-

enabled switch for each given flow. If further packets of the cloned data flow arrive at the switch within the specified pace but after a lag, the switch will immediately send the packet to its intended terminal based on the stated flow entry. As a result, time for route calculation is reduced drastically and network performance is improved. Due to the quickest data transmission rate, the OpenFlow-enabled network's overall throughput will be higher than a traditional network.

From research [70], we can see that SDN-based delay measurement approaches are more precise and exact than traditional network strategies when paired with the route information. Additionally, they are more resistant to dynamic route changes. As the route changes can be done comfortably in comparison to traditional networks. Also, by including the delays of each of its subsequent links, the controller may construct a one-way delay route which is not feasible in a traditional network due to lack of route information. Moreover, SDN is more suitable than traditional networks for executing several measurements simultaneously and parallel due to its central control and flexibility. Furthermore, SDN chooses the best nodes and pathways possible to reduce overall overhead while maintaining accuracy. Resulting in a more effective technique to coordinate network-wide measures through the controller. So, the performance of SDN is greater than traditional networks.

According to [35], an SDN based network has advantages of centralizing the data, reducing hardware cost, cloud abstraction, security, automation, flexibility and more. Almohaimed and Asaduzzaman clearly explained [58], a modern engineering for connecting node corners computing to software-based organizing and appearing to make strides in managing with enormous information handling in SDN. That can only be controlled from the central main controller. The matter that has been decreased by SDN's conception of tall weight on the most controller influences the by and large organize yield, driving to longer inactivity as the information estimate increments. They handled a modern show of SDN Edge Controlling that, by handling edge computing advances, conquered the impediments of the execution. In order to curtail the concern on the SDN controller and the control plane and diminish the lag between the control plane and forward plane, the objective is to urge the computer and computing adroitness near to the arranged gear. The test has shown that the most controller's by and large reaction time is decreased by nearly 62 percent per 10,000 demands, and transmission capacity is diminished by nearly 45 percent.

In paper [23], it states that applying SDN to Unmanned Aerial System (UAS) to screen sensor information gives the capacity to oversee sensor systems while running itself in the full capacity and UAV by a centralized SDN controller. They displayed the SD-UAV design system and sensor organizing. They performed the analogy between the systems without SDN and systems utilizing the SDN access. The reenactment discoveries appeared that utilizing the SDN approach inside systems diminishes parcel misfortune and increments the broadcast capacity as transmitted datagrams were misplaced in SDN-assisted systems at 3.3%. In comparison, misfortunes in SDN arrange datagrams were 16.6%. The result too appeared that the proposed framework can be versatile, adaptable, and reusable for distinctive applications.

In the case of [35], bringing a few conceivable bottlenecks that assailants can use to diminish organized productivity or indeed hinder the accessibility of systems. In expansion, a more capable and profitable immersion strike, a table miss assault, is inspected. There comes about more than standard immersion. As protocol-independent security and effective stage for SDN/OpenFlow systems, they proposed SDN Gatekeeper to avoid lost table assaults. They too proposed SDN Gatekeeper. It is found between the switch and other controller arrangements and ensures the organize utilizing four utilitarian modules: The bundle touchy areas preprocessor module, which causes the controller's stream rules to transmit; the risk locator module to caution of the assault flag; a module of activity channel which classifies the focused-on ports; and frequency-based sifting of activity; the law sweeper within the turn stream table for erasing pernicious rules. All SDN Gatekeeper plans comply with OpenFlow, requiring no change of the convention or outside hardware. The evaluation appeared that, in terms of control channel transmission capacity, machine CPU utilization, and exchange stream table with negligible gadget overhead, SDN Gatekeeper seem to viably ease the table-miss assault and ensure arrangement of foundation assets.

The survey and analysis of [35] shows that conventional systems are difficult to regulate and intricate. The majority of the causes for this are vertical coordination between communication and control planes and producer-specificity. By separating the Data plane from the Control plane, increasing the arrangement's adaptability, and centralizing the control, SDN was given the chance to address these lengthy problems. This led to several ideas focusing on SDN and its application rather than traditional organization. Based on the writing survey, each inquiry examined SDN since of distinctive highlights.

From the comparison graph of [35] shows, Using a safe and efficient sustainable energy system of Blockchain-enabled software-defined IoT as opposed to traditional Blockchain will help to maintain the security and integrity of the organization, and the authors of [23] explained, utilizing the SDN approach inside systems diminishes parcel misfortune and increments the transfer speed. But also, the study of [35] presents a systematic, secure SDN platform capable of fending off DoS attacks and spoofing attacks with minimal setup costs.

It is made evident in the study SDN versus traditional network [23] that significant changes in the architecture of present networks are necessary to support ICT's projected future development and current consumption levels. The substitute that is intended to handle present and future difficulties in communication systems is software defined systems. Systems need to be more flexible, adaptable, controllable, and secure when SDN is implemented. The outcomes of this experiment are positive and demonstrate the potential of SDN, particularly in relation to a crucial area like scalability and network expansion adaption. In the studies, it was feasible to see how longer delays were measured in the old network than in the SDN as the network evolved. On the other side, it was discovered that the SDN network also produced good outcomes when the average throughput in both systems was compared. The study also demonstrated the value of simulation in understanding and assessing the performance of communication networks, particularly the open-source application Mininet, which was developed for SDN investigation and improvement. Nevertheless, there is a still far to go.

Research work of [35], states that SDN changes the organizing vision with a noteworthy thought of isolating the organizing control from the information administration equipment and brings unused functionalities such as programmability, flexibility, adaptability, and appropriation capability within the arrange, which are troublesome to think of in conventional inflexible arrange design. Be that as it may, a wide extent of helpless surfaces specifically or in a roundabout way influence the SDN-based system's data security and dispatch different assaults. The paper starts with a set of the preferences of SDN over the conventional organization but, the discoveries of the investigation take off the wraps with respect to vulnerabilities and their results on data security. Subsequently, the risk surfaces are uncovered that exist in SDN design due to frail data security. In expansion, the investigate discoveries too uncover other unmistakable issues independent of data security issues. The consideration is to ring the chime within the most extreme SDN viewpoints and make analysts or experts mindful of current trends of SDN within the best conceivable way. The comprehensiveness of this work is held by specifying each portion of SDN, which makes a difference for the analysts or experts to progress SDN fundamentally or practically.

Liu and Godfrey portraits [71], and explains the differing and changeable organize upper-layer applications and trade necessities and the current steady and unbending conventional organize design. Moreover, the failure to perform clever stream control and visualized arrange status supervision based on organized conditions is additionally an issue that ruins advance improvement for traditional networks. Furthermore, it shows that the programmable network which uses SDN has 3 major advantages and those are challenges confronted by interface/protocol standardization. At display, the control design framework of the SDN centralized control concept isn't bound together, and it is troublesome to attain common operation due to the distinctive degrees of vendors' back for the SDN standard, The center controller of the SDN organize may have security issues such as over the top stack, single point disappointment, and powerlessness to organize assaults. In this manner, it is vital to set up a sensible component to guarantee the safe and steady operation of the complete framework and the existing ASIC chip engineering is based on the conventional IP or Ethernet tending to and sending plan. In this manner, whether the gear beneath the SDN engineering can keep up the hypothetical tall execution remains to be talked about.

In the research of [35] they define that for the control plane, conventional organizing actualizes a dispersed worldview. For each organized gadget, convention works freely. These organize gadgets interface, but no central AI oversees the full organization or makes it shorter.[23][35].

This paper [35]centralize the idea about SDN's positive highlights would essentially encourage collecting, conveying, recovering, and analyzing enormous information. On the contrary side, Big Data would have important recommendations for the design and management of SDN. The developers suggested that SDN seemed to benefit from vast amounts of data, including activity modelling, cross-layer architecture, potential threats removed, and SDN-based intra- and cross centre technologies. Enormous info and SDN joint development will be a good structuring strategy with this data. On the other hand, the research of Achleitner, Bartolini, He, Portaand Tootaghaj [35] explains that SDN gives an instrument permitting the utilization of stream rules to adjust and re-program the information plane effectively. Effective stream rules updates are essential for the development of highly adaptable SDNs that can quickly recover from planned blackouts or respond to evolving needs. The optimized architecture and associated flow configuration calculations, which take into account computing the existing streaming configuration on the controller and the duration of implementation of this establishment on the switches, have been developed to support fast-changing streaming specifics in SDNs. Through point-by-point recreations. The suggested computations have demonstrated that they outperform current, most short-path-based arrangements in the circumstances under consideration by reducing the overall initialization time of the organization by up to 55% while delivering equivalent parcel loss.

They also discovered that, in an arranged arrangement with the partition of a snuffed relationship, computations would reduce the typical time to restore damaged flows by 40%.

For the network routing management, the research of [35] distinguishes that Application-aware directing "routes" each application exclusively by adjusting the application's communication inclinations and ways with comparing properties. This strategy permits different steering techniques to be connected at the same time, successfully growing the extent of applications the arrange can back with the help of SDN. However, traditional networks have no such ways. Moreover, an OpenFlow controller for actualizing an application-aware arrangement, was created as a reference execution. The appropriateness of an application-aware organization was approved by assessing the possibility and common sense of the innovation. Possibility was assessed by comparing the execution of an application-aware organization and a conventional arrangement in basic circumstances. Common sense was assessed by testing an application-aware arrangement with a real-world application and a wide-area organization with the help of different SDN algorithms used for better management.

Chapter 3

Methodology

Entire network topology is affected by the SDN control plane which gives direct and indirect, the overall performance. Different factors knowledge should be gained which is crucial for better performance. Methodology is given below:
A PC with AMD Ryzen (TM) 3600 CPU @3.80GHz. x2 with RAM 16GB,512GB SSD, 2TB HDD with Windows 10 operating system where we are using Ubuntu 22.4 Lts by using VMware software version 16.2.5. We have made an SDN topology using Mininet version 2.3.0 with Open vSwitch that is controlled by Pox as the controller. NS3 GUI is used to build and connect the topology that are written in Python version 3.5/3.7 scripts. Pox is python based CLI interface controller for SDN. For traditional networks we use GNS3 Simulation tools with Cisco C7200 router image and both Windows and Ubuntu VPC. In traditional networks we use Rip v2 for routing and DHCP as well as Static Ip for PCs. For SDN, we use mininet Vterminal for collecting data.

3.1 Topology

For making any kind of network topology is must needed. Computer network topology is how to interconnect different com-ponents of network. There are several topologies. In traditional network system we have used tree topology for 3 routers. Tree topology is where we use the network components in branches like a tree. [6] These 3 routers are placed like a star topology system. Star topology is where components of a network are placed and connected with each other. For SDN we have used only tree topology keeping the open-VSwitch as the controller at the top of the brunch and next brunch is the routers. After those switches are used as one branch and last branch is used for end-users as well as different servers.
For SDN we have used only tree topology keeping the POX as the controller at the centre of some switches. After that switches are used as one branch and last branch is used for end-users as well as different servers.

3.2 Routing

In networking routing is a very important part. For traditional networking we have to assign Ipv4 address for every inter-face with the subnet mask. If every Ipv4 address is unique then we can configure the routers in the console. Here, we are using rip dynamic routing. We need to configure every routing path for this.

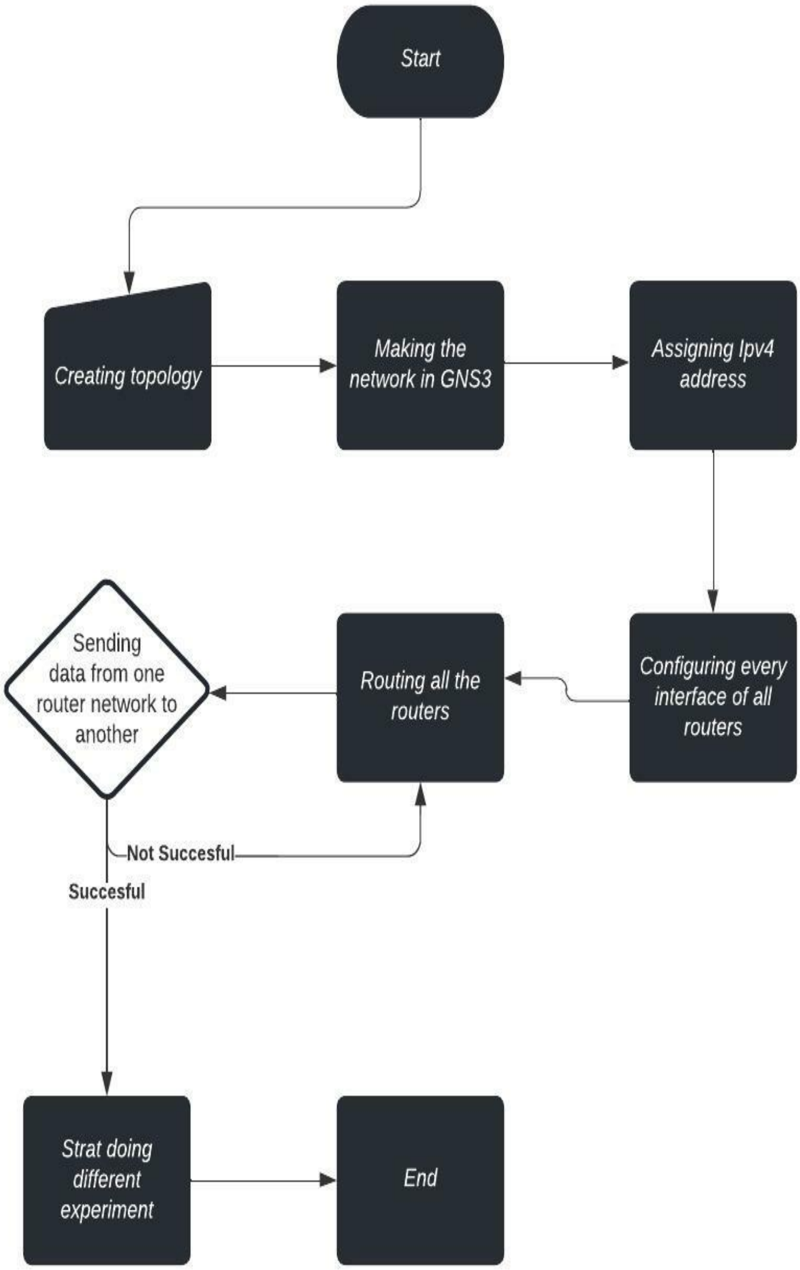

Figure 3.1: Traditional Networks Routing

For SDN routing can be done by the controller. We are using POX as the controller. It has a GUI interface. In the flood light interface there is routing option which is solely purposed SDN network routing.[35]

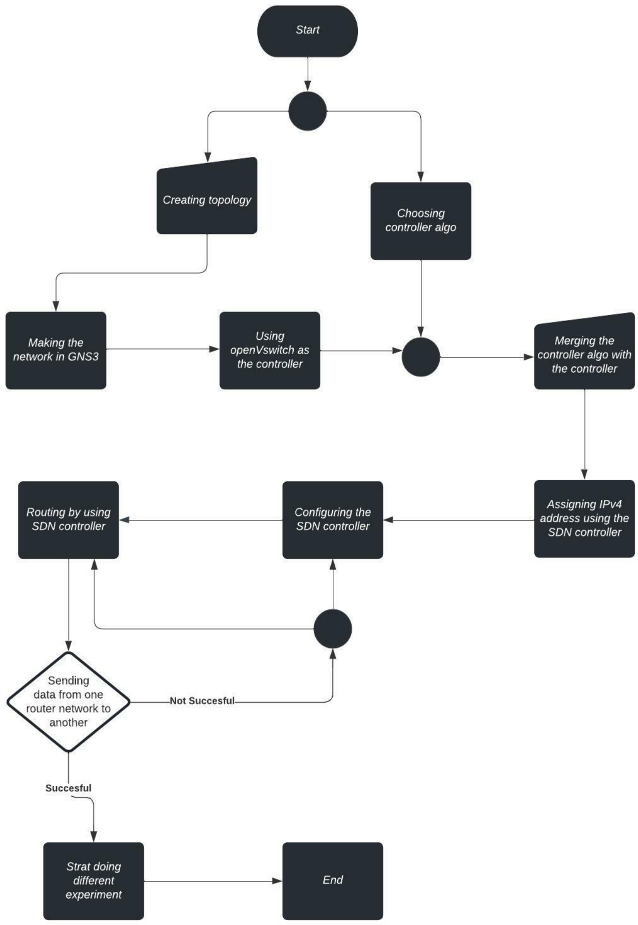

Figure 3.2: SDN Networks Routing

Chapter 4

Implementation

4.1 Traditional Networks

Traditional network is implemented by using GNS3. GNS3 is a Linux operating system-based network simulator. To make it work, we needed to use VMware for GNS3 environment to use it in Windows operating system. We have used 20 routers and 20 switches. One for each router. For router-to-router connection we used serial port as the interface and Serial DTE cable for connections. For router to switch connection, we used ethernet port (RJ-45) as the interface. We have also used some PCs as the end user, 1 DNS server, 1 DHCP server. We used ethernet port (RJ-45) as the interface to connect these components. Moreover, we assigned IPv4 address to some of the PCs separately and the others by DHCP and run DHCP duplication check to check if every single one is unique or not. Then, we configured all the routers and interfaces using router console and used rip dynamic routing for the routing path. After that, we transferred data from one end Pc to another end PC. By repeating data transmission and recording by the parameters we collection our data and analysis them.
We have Used Hybrid architecture to collect data. The reason behind Hybrid architecture is the flexibility to convert into other network types by simply switching links on and off. By connecting to cloud services, it can easily access the internet via VPCs easily.

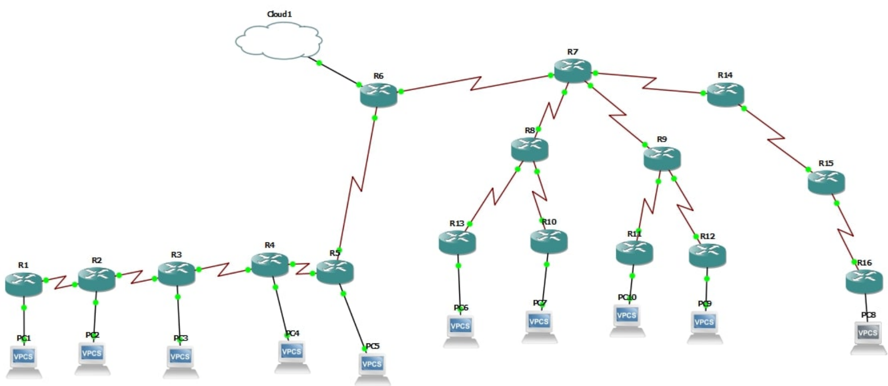

Figure 4.1: Traditional Networks

4.2 SDN Networks

For this we must simulate the network in Mininet a network simulator. The Mininet is a Python based network simulator which needs to Run on in a LINUX distro. For this Linux distribution we choose UBUNTU 18.04 a Debian based Linux distribution by canonical. For UI we used putty 0.78 and Xming 7.7's X11 forwarding method. For controllers we used POX as well as ODL to run the comprehensive tests. POX uses OpenFlow v1.0 and ODL uses OpenFlow v1.3 control system. Both use North bound and South bound interface to maintain the flow. In Mininet we use Open-Vswitch as Interfaces. ODL has a web UI from there all the OpenFlow switches, and routers can be configured. However, POX does not have any Graphical UI to work with, rather we used CLI Interface to RUN the tests. In SDN all the network is controlled through our OpenFlow controller, in our case POX. From the controllers, we assign IP to all the devices, and it automatically routes the network using Shortest Path Algorithm. When we ping the devices across the network result is positive, so the ping is successful. By repeating the transfer of data, we collected results via our parameters.
Again, we have Used Hybrid architecture to collect data. The reason behind Hybrid is to flexibility to convert in.

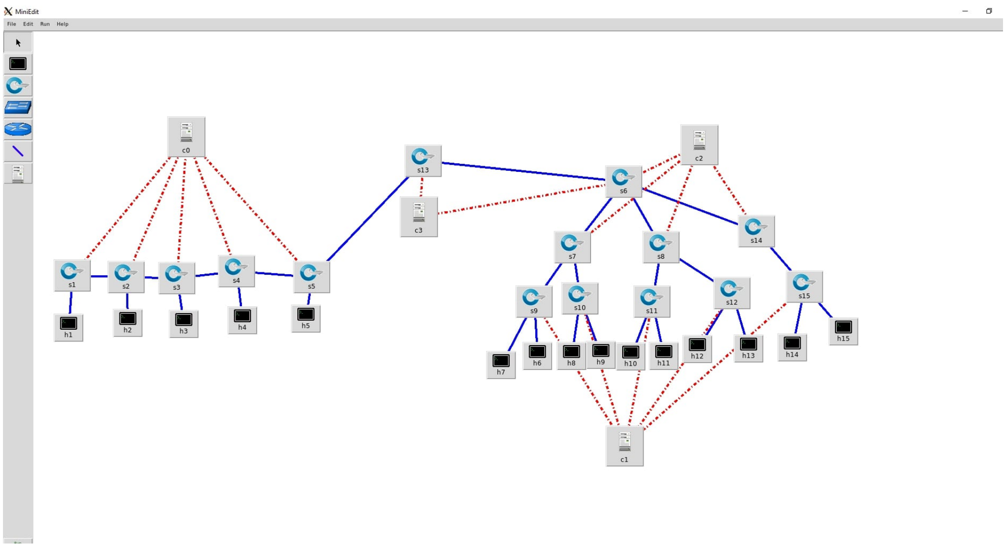

Figure 4.2: SDN Networks

Chapter 5

Result

5.1 Network Monitoring

The process of collecting, analyzing and interpreting data regarding a network's performance and behavior is known as network monitoring. It is an essential part of network management since it enables the detection of issues and the application of fixes to enhance network performance. Network monitoring is essential to network management operations. Trends and patterns in network traffic and devices behavior are a key function of network monitoring [25]. Various specialized hardware and software solutions are commonly used in traditional networks to carry out network monitoring. These tools are made to gather information from different network hardware such as switches and routers. It provides network managers with a general picture of network performance. Traditional network monitoring technologies are not easy to use and could not provide a full picture of the network. SDN offers a more centralized and customizable method of network monitoring. Network administrators receive a thorough overview of the network performance of the network devices thanks to SDN's use of a centralized controller. The controller may also be set up to automatically detect and fix network problems, requiring less operator interaction. The capacity to gather and analyze network data programmatically is one of the main benefits of SDN network monitoring. This enables the development of unique monitoring programs that may be adjusted to the requirements of the network. To evaluate the data and find patterns and trends that may be utilized to enhance network performance, SDNs can also apply machine learning techniques. The capacity to build virtual networks that can be rapidly and readily adjusted to meet demands is another benefit of SDN network monitoring. By allowing for more effective use of network resources, this increases network flexibility and may assist in preventing network problems. For instance, a virtual network can be built for a particular application with strict criteria for low latency and high throughput, like video conferencing.

5.2 Limited Management

Traditional networks, also known as "static networks," rely on manual configuration of network devices. This approach can be more straightforward and easier to manage for organizations with limited technical expertise and resources. However, traditional networks can also be less flexible and less efficient than SDN, as they require manual reconfiguration to make changes to the network infrastructure.

Whereas SDN uses software to centrally control and manage network devices. This approach can provide greater flexibility, automation, and programmability, allowing for faster and more efficient network management.[35] However, it also requires a higher level of technical expertise and specialized hardware and software, which can be costly to purchase and maintain. Additionally, SDN is relatively new technology, lack of industry standards and vendor interoperability can make it difficult for organizations to integrate SDN with their existing network infrastructure.

In the paper [35], it states that for network performance management, power optimization management and resource management different algorithms of SDN are used to make the network better than the traditional network. Because there is no option to modify traditional networks like SDN based networks using different priorities.

In summary, traditional networks may be a better option for organizations with limited technical expertise and resources, while SDN may be a better option for organizations with more advanced technical expertise and resources and are looking for more flexibility and automation.

5.3 Complex Network Management

When it comes to managing complex networks, Software-Defined Networking (SDN) can provide several advantages over traditional networks.
The research of [74] shows that one of the biggest advantages of SDN is its ability to provide a centralized, programmable control plane, which allows for better visibility and control of the network. This can make it easier to manage and troubleshoot complex networks, as well as automate network configuration changes.
Another advantage of SDN is clarified in [23] that the control plane and data plane can be separated, allowing for the usage of various hardware and software on each plane. This can make it easier to scale and adapt the network to changing requirements, as well as improve network security. Additionally, SDN can also provide network virtualization, by which it allows multiple logical networks to run on single physical hardware, which can be useful for multi-tenancy environments [35].
However, SDN also requires specialized hardware and software, as well as a high level of technical expertise to implement and manage. Furthermore, SDN is still a relatively new technology, and the lack of industry standards and vendor interoperability can make it difficult for organizations to integrate SDN with their existing network infrastructure [35].
In conclusion, SDN can provide significant advantages for managing complex networks, but it also requires specialized hardware and software, as well as a high level of technical expertise to implement and manage. Organizations must weigh the benefits against the costs and challenges of implementing SDN before making a decision.

5.4 Latency

Latency is a critical metric in network performance. It determines the overall time needed for a data packet to travel from its source to its destination. According to [54], this measure is crucial because it shows whether a network has issues with managing traffic load, such as blockages, which might be delaying it considerably. In traditional networks, latency is typically introduced by routers and switches. The data must pass through these network devices. In contrast, SDN aims to reduce latency by abstracting the control plane of the network away from the underlying hardware and placing it in software. Another way that SDN reduces latency is by programmatically controlling the flow of traffic through the network. As a result, network resources may be used more effectively, and bottlenecks that could increase latency can be avoided. To further reduce latency, SDNs can employ strategies like traffic engineering to optimize the route that packets take via the network. Routing and switching are proprietary network equipment that are used in traditional networks to manage traffic flow. These devices contain features like firewalls and Quality of Service (QoS) to monitor and regulate traffic. They employ complicated routing protocols to find the optimum path for packets to go via the network. However, because these devices are proprietary, they are difficult to configure and have limited network management. On the contrary, Software-defined networks isolate the network's control plane from the underlying hardware. The data plane is responsible for forwarding the packets and it's a part of the network equipment. However, the control plane which regulates and controls the flow of traffic via the network, is implemented in software. For that, the network is more flexible and programmable as the control plane can be readily upgraded without needing modifications to the underlying hardware.

5.5 Processing Delay

The time it takes for a packet of data to be processed by each network device it travels through end route to its destination is known as processing delay, also known as end-to-end delay [4]. It is an important indicator of network performance since it has an impact on the network's general responsiveness and quality. The many network devices that data must transit through in older networks are usually what cause processing delays. The optimum route for packets to travel via the network is determined by devices like routers and switches. These devices are proprietary and hard to program, which results in a significant processing delay. By abstracting the network's control plane from the underlying hardware and putting it in software, software-defined networks (SDNs) seek to minimize processing time. SDN provides the network additional programmability, flexibility, and control, which can result in networks that are more effective and agile. Utilizing virtualization is another method through which SDNs might lessen processing time. SDNs enable the creation of virtual networks that may be rapidly and easily modified to fit requirements by abstracting the network away from the underlying hardware. By allowing for more effective use of network resources, this increases network flexibility and may assist in shortening processing delays. For instance, a virtual network can be built for a particular application with strict criteria for short processing latency, such as real-time gaming. Despite the continued widespread usage of traditional networks, software-defined networking has the potential to drastically reduce processing latency and boost network performance. SDNs' programmability, flexibility, and control may make networks more effective and nimbler.

In our testing we get that, the processing delay between topologies are not very different with each other in 5hop length traditional network has 7.27ms and 8.1ms delay on an average, but SDN has lower 6.04ms and 7.07ms delay on an average, but if we increase the hop count, the delay keeps on rising. On the contrary, it is always seen that SDN has lower delay than the traditional network.

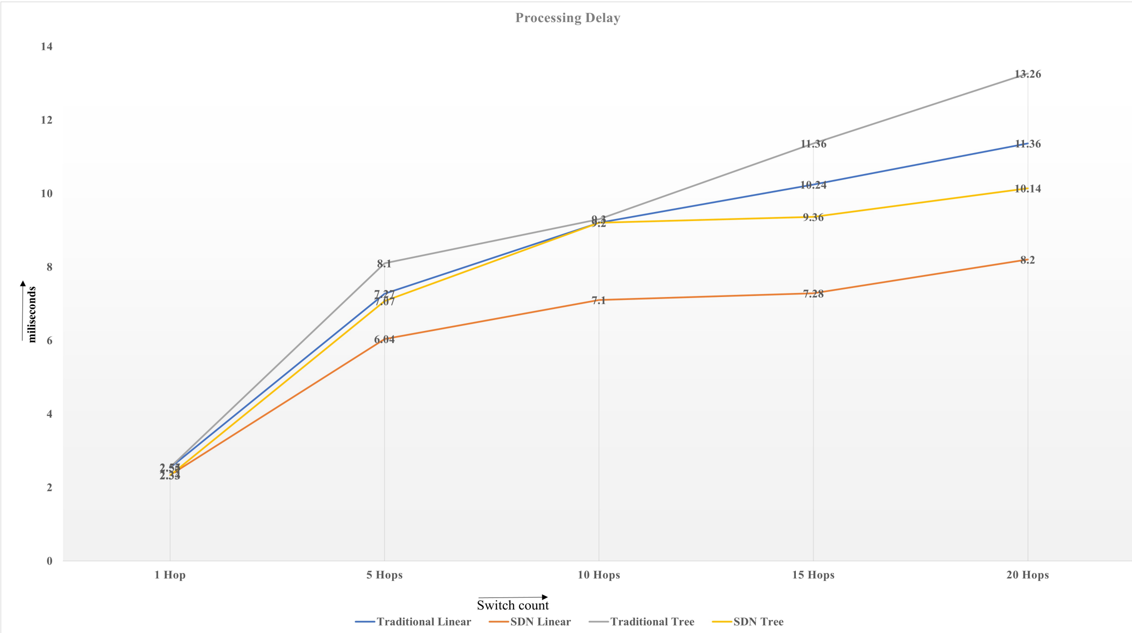

Figure 5.1: Processing Delay

5.6 Round Trip Time

Round trip time counts the amount of time it takes for a data packet to travel from its source to its destination and back to the source [48]. It is a crucial indicator of network performance. Numerous factors such as network congestion, link speed and distance have an impact on it. Due to the numerous network devices that data must transit through and the lack of network traffic flow control in traditional networks, the RTT is frequently high. In contrast, SDN uses less network devices. By abstracting the network's control plane from the underlying hardware and putting it in software, SDN can lower RTT. Furthermore, SDNs provide the network additional programmability, flexibility, and control, which can result in networks that are more effective and agile. Again, SDN uses traffic engineering techniques to lower RTT and optimize the path packets go through. Traffic engineering is the process of managing, routing and analyzing work traffic to enhance and optimize the performance of the network [19]. SDN uses a programmatic approach to manage and direct traffic based on some specific rules and policies. Whereas traffic engineering in traditional networks is based on routing protocols and done manually. Open Shortest Path First (OSPF) and Border Gateway Protocol (BGP) determine the best path for packets to travel through the network [1]. It is done typically by network administrators manually which is a very time consuming and complex process. In our collected data on linear and SDN topologies, we have found that different topologies act differently. The average ping of SDN and Traditional Network in 5 switches are not that different but traditional network has lower than their counter part SDN. When we increase the length of networks to 10 hops, traditional network's linear topology has an average of 1.345ms RTT whereas, SDN has 1.248ms, but in tree topology SDN has 1.969ms while traditional network scoring 1.548ms. It reflects the rest of the testing where SDN linear performs the best and SDN tree performs worst. From the data we get, we can say that SDN linear has been the best.

Average Ping	5 Hops	10 Hops	15 Hops	20 Hops	1 Hop
Traditional linear	0.669	1.345	2.336	4.169	-
SDN linear	0.725	1.248	1.784	2.329	-
Traditional Tree	0.992	1.548	2.557	3.746	-
SDN tree	1.061	1.969	3.157	3.926	-
Traditional Single	-	-	-	-	0.1
SDN single	-	-	-	-	0.06

Table 5.1: Average Ping

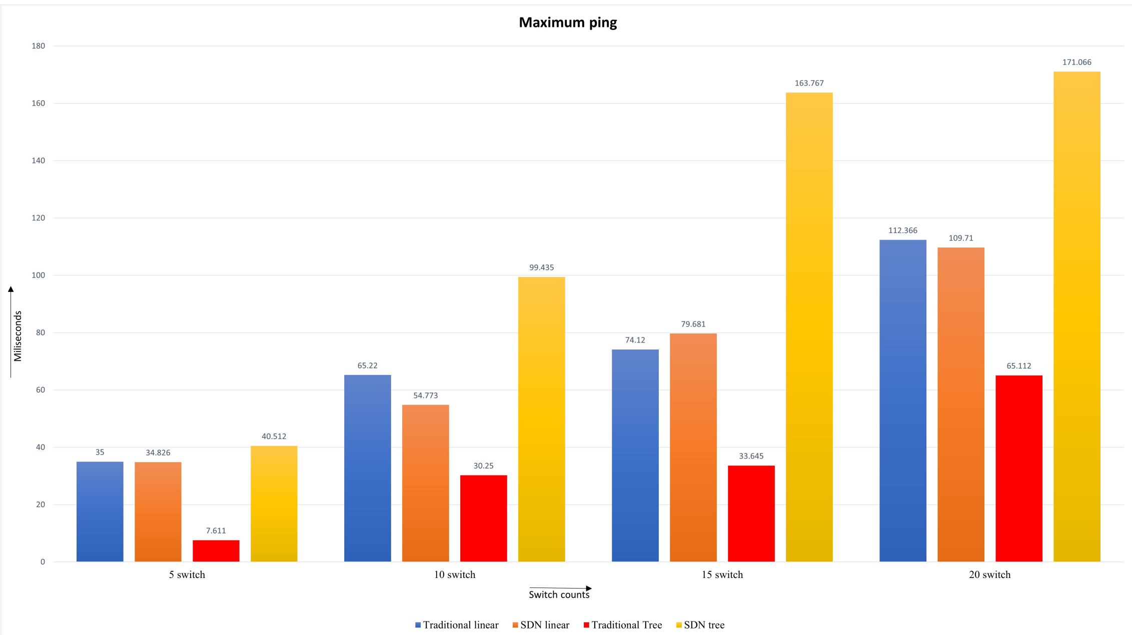

Figure 5.2: Max Ping

When it is about Maximum ping traditional linear has 35ms, 65.22ms ,74.12ms and 112,36ms on our test cases. But SDN linear has slightly lower scores than the other scoring that are 34.826ms, 54.773ms, 79.681ms and 109.71ms. Same reflects on the tree topology as well. But in tree topology maximum pings are way higher than the traditional network. But previously we saw that the traditional tree outperforms the SDN tree on average.

On the other hand, single topology has no significant impact on either SDN or Traditional network on average ping, but for maximum ping, traditional network takes around 7ms which is 5 times higher than SDNs' 1.23ms.

5.7 Packet Drop

Packet drop refers to the process of discarding packets that are being transmitted over a network due to a variety of reasons. Packet drop can occur in traditional networks as well as in Software-Defined Networking (SDN) environments.
In traditional networks, packet drop can occur due to several reasons such as: Overloaded network devices (routers, switches, link-nodes and many more) which May or may not be able to process the amount of traffic they are sending and receiving. Congestion on the network, which can cause packets to be dropped if the network is not able to handle the amount of traffic it is receiving. Configuration error on network devices, such as incorrect access control lists (ACLs), can cause packets to be dropped. Hardware or software failures on network devices can cause packets to be dropped.
In SDN environments, packet drop can occur due to several reasons such as: Overloaded SDN controllers, which overwhelmed the network controller with data causing unable to handle the amount of traffic they are receiving. Congestion on the network, which can cause packets to be dropped if the network is unable to process the amount of traffic it is receiving. Incorrect flow rules or policies on the SDN controller can cause packets to be dropped. Hardware or software failures on the SDN controller can cause packets to be dropped. Packet drop can have a negative impact on network performance and can lead to poor application and service availability. To mitigate packet, drop in traditional networks, network administrators can use techniques such as traffic shaping and rate limiting, while in SDN environments, network administrators can use techniques such as flow-based filtering and proactive monitoring

packet drop	5 switch	10 switch	15 switch	20 switch
traditional linear	1%	2%	4%	7%
SDN linear	0%	0.50%	1%	2%
traditional tree	3%	4.50%	6%	3%
SDN tree	1%	1%	2%	7%
traditional single	0%	0%	0%	0%
SDN single	0%	0%	0%	0%

Table 5.2: Packet Drop

In our testing the packet drop we see there is significantly less packet drop in SDN then linear. In SDN vs traditional networks 5 switches, 10 switches 15 switches and 20 switches test cases.
In the traditional linear topology, as the number of hops increases, the packet drop rate also increases linearly. For example, at 5 switches, the packet drop rate is 1%, at 10 switches rate is at 2% and at 20 switches, the packet drop rate is 7%. This suggests that as the number of switches in the network increases, the packet drop rate also increases.

In contrast, in the SDN linear topology, the packet drop rate is Somewhat lower and remains relatively constant as the number of switches increases. For example, at 5 switches, the packet drop rate is 0%, and at 20 switches, the packet drop rate is 2%. This suggests that in an SDN environment, the packet drop rate is low and less affected by the increasing number of switches adding in the network.
In the traditional tree topology, the packet drop rate increases as the number of switches increases, but it's not linear. For example, at 5 switches, the packet drop rate is 3% and at 20 switches, the packet drop rate is 3%. This suggests that as the number of switches in the network increases, the packet drop rate increases but not in a linear way.
In the SDN tree topology, the packet drop rate is relatively low and remains relatively constant as the number of switches increases. for example, at 5 switches, the packet drop rate is 1% and at 20 switches, the packet drop rate is 2%. This suggests that in an SDN environment, the packet drop rate is lower and less affected by the number of switches in the network.

5.8 Data Flooding

Data flooding happens when large number of data arrives or requested from a network. Most of times data flooding occurs when someone attacks or perform attack in a network. There can be normal users trying to send a big data or some attackers attacking the network. Both Traditional and SDN has data flooding but both types of networks have different system to be flooded and mitigate this.
In traditional networks, Flooding can be happened by sending large amount data than the network capacity which refers to DDOS attack. Alongside it pings using mismatching information and spoofing also cause data flooding. Data amplification also a method to flood a network by DNS and NTP.
In SDN DDoS also can caused data flooding. Moreover, Flood table overflowing and protocol over flowing also causing flooding in SDN networks. Flow table overflow attack is a type of data flooding in which the attacker sends a huge number of packets towards the SDN controller with the intention of overwhelming the flow table, causing the controller to crash or become unresponsive [10]. Protocol flooding attack is a type of data flooding in which the attacker sends a huge number of packets to the SDN controller with the intention of overwhelming the southbound protocol, causing the controller to crash or become unresponsive [10].
To mitigate the data flooding in traditional network in order to prevent network flooding, traffic shaping and rate limiting involve limiting the amount of traffic that can enter the system. Devices like routers and switches that have been set up with traffic shaping and rate limiting policies can be used for this [10].
Attacks that aim to flood servers with data can be recognized and stopped using these systems. IDS/IPS systems can be used to detect and block incoming traffic that matches particular attack signatures, while firewalls can be configured to block incoming traffic from known malicious IP addresses.
By segmenting the network, one can confine the harm a data flooding attack causes to a particular portion of the network. Network access control (NAC) solutions or the creation of virtual LANs (VLANs) can be used to accomplish this.
A significant amount of incoming traffic can be absorbed and filtered out by these services, keeping it from reaching the intended network. Both specialized DDoS pro-

tection appliances and cloud-based DDoS protection services can be used to achieve this.
The use of a load balancer allows for the distribution of incoming traffic among numerous servers. It can aid in distributing the traffic load and guard against a data flooding attack that would overwhelm a single server.
To mitigate the data flooding in SDN traffic shaping and rate limiting policies can be applied more automatically and dynamically using SDN controllers. More precise control over network traffic is possible thanks to the controller's centralized view of the entire network.
When incoming traffic matches certain flow attributes, such as source or destination IP addresses, ports, or protocols, SDN controllers can drop or rate limit the traffic. By dropping or rate limiting traffic that matches specific attack signatures, this can help to stop data flooding attacks [12].
Using SDN controllers, the network can be divided into sections and protected from data flooding attacks. Virtual networks (VNs) and network access control (NAC) software are two methods for achieving this.
Similar to traditional networks, SDN can use specialized DDoS protection appliances or cloud-based DDoS protection services to absorb and filter out a significant portion of incoming traffic, preventing it from reaching the target network.
Real-time monitoring and analysis of network traffic is possible thanks to SDN controllers' centralized view of the entire network. This can be used to quicker and more successfully detect data flooding attacks and respond to them.
Programmable mitigation is made possible by SDN, where software programming languages are used to create the mitigation policies. This enables more dynamic and adaptable mitigation policies and can make it simpler to adjust to shifting network conditions and requirements.

5.9 Scalability

One characteristic that many technologies claim to have been scalability. An issue with many facets, indeed. The fundamental idea may be obvious, but not everyone will immediately associate scalability with the same idea. As a result, neither its concept nor its substance is subject to widespread, exact consensus. Scalability can be interpreted in a variety of ways. Some people describe it as the efficiency of processing capacity to CPUs, while others view it as a metric of application parallelization over several computers. It is a desired quality showing good sense towards a system, design, algorithm, and other things, independent of what it means to someone. According to [49], preemptive rule implementation in SDN switches lowers the burden on the controller, which cuts down on processing time and flow commencement cost in the controller. Nevertheless, this limits the adaptability brought on by the implementation of reactive flow and lessens the controller's and the network's ability to make quick decisions. As another method of reducing the cognitive burden on the controller, controller dispersion also has coordination and integrity issues. Scalability, then, is a mixture of problems that involves exchange to be openly articulated when offering a cure rather than an isolated concern which can be addressed entirely. The paper [49] also proposes that Flow-level forwarding configuration in OpenFlow switches must be put up and destroyed by a controller in SDN either proactively or reactively. In order to avoid having to repeat this step for consecutive packets within the same flow, the flow forwarding status is stored on the OpenFlow switches after it has been configured. A delay is also a part of this configuration procedure. The SDN industry believes that this flow configuration is the most likely culprit for the control plane's efficiency barrier. As a result, while assessing the effectiveness of control plane scalability, the throughput requests processed and flow establishment delay take center stage. Because of this, throughput and flow setup delay are the two measures that best describe scalability, and in the domain of SDN, control plane virtualization specifically. SDN provides the customer with more scalability, which is a beneficial byproduct of centralized provisioning. You may quickly alter your network architecture if you have the flexibility to supply services as needed. When contrasted to a traditional network arrangement, where resources must be manually acquired and managed, the disparity in scalability is astounding. Scalability issues are SDN's key challenges. According to [72] there are two related concerns that may be deduced from this main issue: (A) The controller's scalability (B) The network node's scalability. The paper [73] states that up to 6 million flows can be processed by one controller per second. As a result, in paper [55] [44] it shows how just one controller, or a few controllers, can operate the necessary control plane operations for a significant amount of data forwarding nodes. The theoretically hierarchical controller should therefore be strategically distributed to promote scalability rather than running peer-to-peer [31]. In spite of whether a controller system is distributed or peer-to-peer, according to the study [62], all system components will have the same connectivity problems as the controller.

5.10 Centralized Provisioning

One of SDN's main advantages is its capability to govern networks from a centralized perspective. Simply said, SDN attempts to show the network and data control planes, enabling users to provide both physical and virtual components from a single place. This is quite helpful since managing traditional architecture may be difficult, specifically when there are numerous different components that must be controlled separately. An administrator can dig down and up at will thanks to SDN, which removes this barrier. According to [35], in the traditional architecture, network setup, deployment, and maintenance necessitate highly specialized network and system engineer involvement and operating expenses associated with provisioning and maintaining massive multi-vendor systems. In SDN, a controller that establishes the network policy centralizes control processes. Open-source controller systems include Floodlight, OpenDayLight, and Beacon. There are several layers at which the network may be managed. According to [59] by monitoring the network topological graph within the Network Information Base, the Controller of SDN keeps a centralized perspective of the system (NIB). The Controller introduces forwarding instructions through the so-called Southbound Interface into the nodes that have flow tables in the network in accordance with data in the NIB and regulations set forth by network applications (SBI). OpenFlow (OF) is the SBI that is well known [17]. The packet header data which must match are listed in a flow table entry, along with the activities that should be taken when a match is made. The paper [26] implies that centralization protocols like OpenFlow say that a switch is managed by a controller. In order to allow alternative controllers to take over during the case of a breakdown, OpenFlow permits the connecting of several controllers to a switch.

5.11 Maintaining quality of service

The main difference between maintaining QoS in (SDN) and traditional networks is the way in which QoS is implemented and managed.
In traditional networks, QoS is typically implemented through a combination of hardware and software, using devices such as routers, switches, and firewalls. These devices are configured with various QoS policies and rules, such as traffic shaping, prioritization, and bandwidth management, to manage and prioritize network traffic. However, managing QoS in traditional networks can be complex and time-consuming, and it requires a deep understanding of the underlying network infrastructure and protocols.
In SDN, QoS is implemented through software-based controllers and algorithms. The SDN controller has a centralized view of the entire network and can apply QoS policies and rules in a more automated and dynamic way. This allows for more granular control over network traffic, and it can make it easier to manage and prioritize network traffic. Additionally, SDN allows for real time monitoring and control of the network traffic, which can be used to optimize QoS and resolve issues more quickly.
Additionally, SDN supports Automated QoS enforcing rules. So, when needed Controllers can easily roll out the policies and ensure stability. On the other hand, traditional networks typically manually configured on network devices, and it can be difficult to ensure that the policies are consistently enforced across the entire network. In SDN controller it has a centralized view of entire network and monitor it in real time. But traditional one network cannot see what is happening in other network so resolve issues takes a lot of time.

5.12 Security

Security is an essential element of software-defined networking. SDN safety must be embedded into the design in order to give all components and data usefulness, integrity, and security [69]. Targeting the routes of interaction between separated planes allows one plane to attack another while pretending to be another. There are no standards or explicit criteria that would allow open APIs for programs to administer network support and activities through the control plane, which puts system services, processes, and functions at substantial security risk [7]. According to [32] a centralized decision-making body in SDN is the control plane, which might be an OpenFlow controller. Because of its crucial role, the controller can therefore be specifically attacked for network compromise or criminal activity. According to [17], In SDN, the data plane's security is significantly impacted by the security of the control plane. It implies that if one controller is hacked, the system as a whole, which consists of several data plane nodes, would also be vulnerable. Threats notwithstanding, there are also remedies. It is simple to develop new apps that would receive network information and packet attributes via the controller to create additional security services thanks to the centralized control architecture made possible by SDN.

Additionally, while adhering to security restrictions, access to legitimate apps must be ensured by the control plane. in accordance with their functional needs. Additionally, dangerous apps that may install, alter flow controls in the data route must be prevented from accessing the data plane. For applications that have the ability to alter flow rules, perfectly alright security enforcement techniques like authentication and authorization are utilized. Users are given a comprehensive understanding of their infrastructure and the ability to regulate the security of their whole network despite the fact that doing so makes the SDN controller a threat.

5.13 Reduced Hardware Footprint

An operator can increase productivity by optimizing hardware consumption with SDN. Dynamic hardware can have its functionality changed at any time by the user. This implies that sharing of resources is quite simple. This is superior to a legacy-driven network with restricted hardware use. The paper [52] states that the idea of Software-Defined Networking (SDN) is to transfer communication control from hardware level to the software level. In the context of SDN, routers are transformed into configurable hardware components with the capacity to modify their connections in response to instructions from an SDN Controller. The papers [53] [60] imply that this method encourages a general and concise interaction layout framework, and because the Controller has a high level of understanding about the communication facilities, it also leverages self-adaptivity and multi-objectivity. As a result, the on-chip communication infrastructure is made simpler. simple CS router design that requires no routing or mediation and simply needs configurable facilities for packet forwarding.

In this study, we examine the viability of reducing energy usage by adopting software-defined networks based on OpenFlow. This was carried out in a simulated setting with Mininet. The paper [39] states that being a command-line program, Mininet constructs a virtual network made up of virtual hosts, switches, and a controller; these speeds up response time and simplifies the process to operate. Mininet is employed to emulate a controller and improve a network architecture in order to reduce time, power, and resources. A Flow Table can be virtualized and utilized on the physical network of OpenFlow switches after the network has been simulated and referred to the controller.

5.14 Outcome

By comparing and testing both SDN and Traditional Networks, Software-Defined Networking (SDN) is a network architecture that separates the control plane and data plane of network devices, allowing for more centralized control and dynamic configuration of network devices. This approach allows for more flexibility and ease of management, as network administrators can use software to Overall, SDN allows for more efficient, flexible, and automated management of networks. It also provides an open and programmable interface, which enables integration with other technologies, such as virtualization, cloud computing and IoT. However, it requires a significant investment in terms of infrastructure and expertise [45].

In our tests, at every topology SDN out performs Traditional Static Network and it is also seen that in terms of scalability and monitoring SDN has a lot more advantage than the Traditional network. In terms of Latency, Delay and RTT our tests shows that if the length is small there is no significant difference in SDN and Traditional network. But as the Network length increases SDN has shown a greater success than Traditional network in maintaining the networks. In the test SDN linear topology is the best topology by maintaining networks stability on the other hand Traditional network Tree is the worst. However, average RTT is lower than the SDN tree. But other cases it is not performing well in other cases. In terms of Data flooding mitigation and prevention SDN has more real time measure than Traditional network. So, is more like to use SDN as the future technology in networks. Additionally, it's worth mentioning that some traditional network still has its value in certain use case, for example, networks with high security or criticality requirements, where the centralized control plane of SDN may not be suitable. Confidentiality also reduces in SDN as well.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

It is tough and challenging to modify conventional networks. The development of flexible configurability features, that circumvent this limitation, is indeed a core SDN concept. Traditional networks now in use are difficult to administer and exceedingly sophisticated. One crucial element is the vertical connection between a switch's Control Plane and Data Plane. Additionally, these switches are vendor-specific. They are restricted to certain lines and models of items. In other terms, we may argue that every network switch might have a unique administration and setup plane, that causes a lengthy loop for creating changes. Vendor lock-in issues are caused by all the aforementioned factors, which also severely impede creativity and progress. These old issues may now be resolved in large part because of software defined networking. The key tenet of software-defined networking architecture is, in fact, the separation of the control plane and the data plane. By concentrating on performance instead of data, SDN enables the virtualization of the human intellect accessible to networking. Current networking methods like SDN have been utilized to reinvent a variety of remedies to traditional network concerns, while certain issues still provide difficulties. With the rising complexities of the infrastructure and numerous more software sectors, SDN offers effective and automated network management. The construction and deployment of networks now follow the emerging paradigm of SDN. SDN is built on a versatile, adaptable, and generally simple-to-implement infrastructure that is perfect for the necessities of modern world's networks and services.

Software-defined infrastructure may be the answer for the Internet of the future. Traditional networks have complex architecture and can be hard to manage because of the vertical integration of the control and the data planes on a networking central device. Because the data and control plane are separated as from network base units in SDN technology, the hierarchical unification concern with conventional networks is resolved. This is in contradiction to conventional network architecture, which is decentralized in management.

In this paper, the varied network topologies taken into account allow for a performance comparison of traditional networks with software-defined networks. by contrasting two fundamentally tree-based Mininet network topologies. The collected numerical findings unequivocally show that SDN performs better than conventional networks. The objective of this study was to conduct a quantitative comparison of key network operational indicators, such as Latency, Round-Trip Time, and Throughput between a conventional network design utilizing the POX controller, routing algorithms, and the OpenFlow interface and an SDN network structure employing these same components. In general, improved productivity in the variables under study is confirmed by SDN, confirmed by descriptive statistics, with normalcy and non-parametric testing. Studies have also been done on potential performance penalties in functionally complicated SDN systems. Results disprove this phenomenon. Simulation experiments have proven as expected that network performance is dependent on sources of potential, virtual node density, and network structure. Because of this, experimental outcomes in systems with varying time and bandwidth may occur.

6.2 Future Work

Like all things have their limitations, networks also have theirs. We have faced a lot of issues while working with SDN and traditional networks. The limitations are more in number in terms of SDN. The traditional network is what we are using on a regular basis, but since SDN is a new paradigm, it is hard to get resources for SDN. Nowadays there are hardly any recent resources available online. For example, we have faced a lot of difficulties while only setting up the SDN network alone. Most of the resources that are available online are too old which do not work anymore. According to [38] interactions between processes and programs using the same physical architecture are the root cause of SDN's increasing complexity. The anticipation of these interconnections and the corresponding adaptation of the SDN to give reliability assurances in terms of accessibility and responsiveness are significant challenges that service providers must overcome. The most crucial step in adjusting to shifting client demands and load variations is determining and delivering the required resources to achieve the service level agreement (SLA). Choosing the level of precision at which resources (such as CPU cycles, cluster nodes, increased memory, expanded storage capacity, and bandwidth) should be constantly introduced is also of utmost relevance for the long-term observance of SLA.
We initially started to work on GNS3 for simulation purposes, but later shifted to Mininet. There are no recent updates of Mininet or GNS3. In GNS3, there are hardly any free sources available to work on. Cisco and Ubuntu both had put down their necessary plugins from GNS3, which makes it hard to use GNS3 as a simulator. Since it will be costlier to purchase all the necessary materials, GNS3 is abandoned in a way. We later shifted to Mininet. However, since Mininet does not have a recent update, we were left with a bunch of errors for connecting Mininet with our SDN controller OpenDayLight which we used before the POX controller. We were not able to connect the OpenDayLight controller with Mininet because of the unavailability of the resources. Later we used the POX controller and did our work successfully.
Among all the SDN controllers, only OpenDayLight (ODL) has recent updates. All the other controllers' versions are old. Old versions are not compatible in newer environments in Mininet. So, this was challenging as well. However, the OpenDayLight controller does not have the documentation available which made us not go for ODL. Since the resources are too old, and we had no documentation to follow as well, we chose not to work with ODL. Hence, we chose the POX controller over

all other controllers and finally we were able to get our desired results.
Further comparisons of the SDN network characteristics in Mininet and OpenDaylight might be instructive in illuminating the weaknesses and merits of the two systems. SDN handles the majority of the operations, however security is a key worry. As vital as endpoints are, network security is as crucial. The centralized design of SDN creates a security issue. Stronger infrastructure will thus be used to counter this. The security concerns relating to the control plane are comparable across traditional networks and SDN, but the necessary countermeasures are different. For the purpose of filtering control packets from end nodes, traditional networks can set up a network border. Less beneficial in SDN is this strategy. In the paper [42] it has SDN controllers incorporated into its control plane, that are frequently connected to the network's edge and have connection points that are analogous to end host placements. If a particular control network exists that is separate from user networks, then the answer may change. In order to prohibit intruders from taking part in the control plane, SDN mostly needs cryptographic security. Therefore, security for SDN can be a major field to study in the future. According to [72] numerous specific SDN challenges, such as standardizing the SDN modules and establishing new special SDN processes, still need further research to minimize problems brought about by traditional networks. To develop original ideas for the controllers that function as the SDN architecture's intellect, the research must place a greater emphasis on the control plane. Various security precautions should be taken into consideration because the control plane is a potential point of breakdown for the overall infrastructure. As a result, SDN is crucial in creating countless fixes for traditional network concerns, while certain difficulties still pose a challenge. Additionally, it offers effective and autonomous network control that fulfills the demands of the network's growing intricacy as well as those of several other software fields. Future study will need to take into account shared, diverse communication architectures, the sustainability of the SDN idea for large networks, i.e., control network delay, as well as consequences and methods for security, i.e., susceptibility of the control system through the switches.

Bibliography

[1] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional ip routing protocols," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 118–124, 2002. DOI: 10.1109/MCOM.2002.1039866.

[2] R. Presuhn, *Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)*, RFC 3416, Dec. 2002. DOI: 10.17487/RFC3416. [Online]. Available: https://www.rfc-editor.org/info/rfc3416.

[3] G. Lichtwald, M. Zitterbart, and U. Walter, "Improving convergence time of routing protocols," 2004.

[4] R. Ramaswamy, N. Weng, and T. Wolf, "Characterizing network processing delay," in *IEEE Global Telecommunications Conference, 2004. GLOBECOM '04.*, vol. 3, 2004, 1629–1634 Vol.3. DOI: 10.1109/GLOCOM.2004.1378257.

[5] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp. 1–6.

[6] T. Fencl, P. Burget, and J. Bilek, "Network topology design," *Control Engineering Practice*, vol. 19, no. 11, pp. 1287–1296, 2011.

[7] T. Nadeau and P. Pan, "Software driven networks problem statement," *Network Working Group Internet-Draft, Sep*, vol. 30, 2011.

[8] R. Antonello, S. Fernandes, C. Kamienski, *et al.*, "Deep packet inspection tools and techniques in commodity platforms: Challenges and trends," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 1863–1878, 2012, ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2012.07.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804512001622.

[9] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, "Fabric: A retrospective on evolving sdn," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12, Helsinki, Finland: Association for Computing Machinery, 2012, pp. 85–90, ISBN: 9781450314770. DOI: 10.1145/2342441.2342459. [Online]. Available: https://doi.org/10.1145/2342441.2342459.

[10] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest flooding attack and countermeasures in named data networking," in *2013 IFIP Networking Conference*, 2013, pp. 1–9.

[11] M. Fernandez, "Evaluating openflow controller paradigms," in *ICN 2013, The Twelfth International Conference on Networks*, 2013, pp. 151–157.

[12] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013. DOI: 10.1109/SURV.2013.013013.00155.

[13] A. Gelberger, N. Yemini, and R. Giladi, "Performance analysis of software-defined networking (sdn)," in *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, 2013, pp. 389–393. DOI: 10.1109/MASCOTS.2013.58.

[14] E. Patouni, A. Merentitis, P. Panagiotopoulos, A. Glentis, and N. Alonistioti, "Network virtualisation trends: Virtually anything is possible by connecting the unconnected," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, 2013, pp. 1–7. DOI: 10.1109/SDN4FNS.2013.6702545.

[15] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito, and L. Veltri, "Information centric networking over sdn and openflow: Architectural aspects and experiments on the ofelia testbed," *Computer Networks*, vol. 57, no. 16, pp. 3207–3221, 2013, Information Centric Networking, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2013.07.031. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128613002727.

[16] D. Sankar and D. Lancaster, "Routing protocol convergence comparison using simulation and real equipment," *Advances in Communications, Computing, Networks and Security*, vol. 10, pp. 186–194, 2013.

[17] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 413–424.

[18] A. L. Valdivieso Caraguay, L. I. Barona Lopez, and L. J. Garcia Villalba, "Evolution and challenges of software defined networking," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, 2013, pp. 1–7. DOI: 10.1109/SDN4FNS.2013.6702542.

[19] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in sdn-openflow networks," *Computer Networks*, vol. 71, pp. 1–30, 2014, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2014.06.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128614002254.

[20] 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, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2014.10.015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128614003703.

[21] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, *Software-defined networking: State of the art and research challenges*, 2014. DOI: 10.48550/ARXIV.1406.0124. [Online]. Available: https://arxiv.org/abs/1406.0124.

[22] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 1955–1980, 2014. DOI: 10.1109/COMST.2014.2320094.

[23] S. Kaur, J. Singh, and N. Ghumman, "Network programmability using pox controller," Aug. 2014. DOI: 10.13140/RG.2.1.1950.6961.

[24] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using openflow: A survey," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 493–512, 2014. DOI: 10.1109/SURV.2013.081313.00105.

[25] S. Lee, K. Levanti, and H. S. Kim, "Network monitoring: Present and future," *Computer Networks*, vol. 65, pp. 84–98, 2014, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2014.03.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138912861400111X.

[26] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications surveys & tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[27] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014. DOI: 10.1109/SURV.2014.012214.00180.

[28] L. R. Prete, A. A. Shinoda, C. M. Schweitzer, and R. L. S. de Oliveira, "Simulation in an sdn network scenario using the pox controller," in *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2014, pp. 1–6. DOI: 10.1109/ColComCon.2014.6860403.

[29] V. Sokolov, I. Alekseev, D. Mazilov, and M. Nikitinskiy, "A network analytics system in the sdn," in *2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC)*, 2014, pp. 1–3. DOI: 10.1109/MoNeTeC.2014.6995603.

[30] K. Venkatraman, V. Parthasarathy, S. S. Kumaar, and M. Jayalakshmi, "Supervision of network through software defined networking," in *International Conference on Information Communication and Embedded Systems (ICICES2014)*, 2014, pp. 1–7. DOI: 10.1109/ICICES.2014.7033777.

[31] S. Zeebaree and H. M. Yasin, "Arduino based remote controlling for home: Power saving, security and protection," *International Journal of Scientific & Engineering Research*, vol. 5, no. 8, pp. 266–272, 2014.

[32] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2317–2346, 2015. DOI: 10.1109/COMST.2015.2474118.

[33] M. Erel, E. Teoman, Y. Özçevik, G. Seçinti, and B. Canberk, "Scalability analysis and flow admission control in mininet-based sdn environment," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, 2015, pp. 18–19. DOI: 10.1109/NFV-SDN.2015.7387396.

[34] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, 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. DOI: 10.1109/JPROC.2014.2371999.

[35] J. Panford, K. Riverson, O. Boansi Kufuor, and R. Yehuza, "Comparative analysis of convergence times between rip and eigrp routing protocols in a network," vol. 2, May 2015.

[36] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 27–51, 2015. DOI: 10.1109/COMST.2014.2330903.

[37] H. Zhang and J. Yan, "Performance of sdn routing in comparison with legacy routing protocols," in *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 2015, pp. 491–494. DOI: 10.1109/CyberC.2015.30.

[38] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (sdn): A survey," *Security and Communication Networks*, vol. 9, no. 18, pp. 5803–5833, 2016. DOI: https://doi.org/10.1002/sec.1737. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sec.1737. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1737.

[39] D. B. Rawat and C. Bajracharya, "Software defined networking for reducing energy consumption and carbon emission," in *SoutheastCon 2016*, 2016, pp. 1–2. DOI: 10.1109/SECON.2016.7506640.

[40] V. Shamugam, I. Murray, J. A. Leong, and A. S. Sidhu, "Software defined networking challenges and future direction: A case study of implementing SDN features on OpenStack private cloud," *IOP Conference Series: Materials Science and Engineering*, vol. 121, p. 012 003, Mar. 2016. DOI: 10.1088/1757-899x/121/1/012003. [Online]. Available: https://doi.org/10.1088/1757-899x/121/1/012003.

[41] A. S. Thyagaturu, A. Mercian, M. P. McGarry, M. Reisslein, and W. Kellerer, "Software defined optical networks (sdons): A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 4, pp. 2738–2786, 2016. DOI: 10.1109/COMST.2016.2586999.

[42] A. Abdou, P. C. van Oorschot, and T. Wan, "A framework and comparative analysis of control plane security of sdn and conventional networks," *arXiv preprint arXiv:1703.06992*, 2017.

[43] S. S. Bhunia and M. Gurusamy, "Dynamic attack detection and mitigation in iot using sdn," in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, 2017, pp. 1–6. DOI: 10.1109/ATNAC.2017.8215418.

[44] O. Chippalkatti and S. Nimbhorkar, "An approach for detection of attacks in software defined networks," in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, IEEE, 2017, pp. 1–3.

[45] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A survey on the security of stateful sdn data planes," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1701–1725, 2017. DOI: 10.1109/COMST.2017.2689819.

[46] C. Fancy and M. Pushpalatha, "Performance evaluation of sdn controllers pox and floodlight in mininet emulation environment," in *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, 2017, pp. 695–699. DOI: 10.1109/ISS1.2017.8389262.

[47] D. Gopi, S. Cheng, and R. Huck, "Comparative analysis of sdn and conventional networks using routing protocols," in *2017 International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2017, pp. 108–112. DOI: 10.1109/CITS.2017.8035305.

[48] B. Hale, T. Jager, S. Lauer, and J. Schwenk, "Simple security definitions for and constructions of 0-rtt key exchange," in *Applied Cryptography and Network Security*, D. Gollmann, A. Miyaji, and H. Kikuchi, Eds., Cham: Springer International Publishing, 2017, pp. 20–38, ISBN: 978-3-319-61204-1.

[49] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (sdn)," *Computer Networks*, vol. 112, pp. 279–293, 2017.

[50] M. Lali, R. Mustafa, F. Ahsan, M. Nawaz, and W. Aslam, "Performance evaluation of software defined networking vs. traditional networks," *The Nucleus*, vol. 54, no. 1, pp. 16–22, Mar. 2017. [Online]. Available: http://202.83.167.189/index.php/Nucleus/article/view/95.

[51] S. Mishra and M. A. R. AlShehri, "Software defined networking: Research issues, challenges and opportunities," *Indian Journal of Science and Technology*, vol. 10, no. 29, pp. 1–9, 2017.

[52] Y. E. Oktian, S. Lee, H. Lee, and J. Lam, "Distributed sdn controller system: A survey on design choice," *computer networks*, vol. 121, pp. 100–111, 2017.

[53] M. Ruaro, H. M. Medina, and F. G. Moraes, "Sdn-based circuit-switching for many-cores," in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2017, pp. 385–390. DOI: 10.1109/ISVLSI.2017.74.

[54] V. Jara and Y. Shayan, "Latency measurement in an sdn network using a pox controller," in *2018 IEEE Canadian Conference on Electrical Computer Engineering (CCECE)*, 2018, pp. 1–5. DOI: 10.1109/CCECE.2018.8447647.

[55] A. Jefia, S. Popoola, and A. Atayero, "Software-defined networking: Current trends, challenges, and future directions," in *Proceedings of the International Conference on Industrial Engineering and Operations Management*, 2018, pp. 27–29.

[56] P. Krongbaramee and Y. Somchit, "Implementation of sdn stateful firewall on data plane using open vswitch," in *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2018, pp. 1–5. DOI: 10.1109/JCSSE.2018.8457354.

[57] L. Mamushiane, J. Mwangama, and A. A. Lysko, "Given a sdn topology, how many controllers are needed and where should they go?" In *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2018, pp. 1–6. DOI: 10.1109/NFV-SDN.2018.8725710.

[58] A. Almohaimeed and A. Asaduzzaman, "Dedicated backup units to alleviate overload on sdn controllers," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, pp. 0591–0596. DOI: 10.1109/CCWC.2019.8666452.

[59] M. Amadeo, C. Campolo, G. Ruggeri, A. Molinaro, and A. Iera, "Sdn-managed provisioning of named computing services in edge infrastructures," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1464–1478, 2019. DOI: 10.1109/TNSM.2019.2945497.

[60] S. Ellinidou, G. Sharma, T. Rigas, T. Vanspouwen, O. Markowitch, and J.-M. Dricot, "Sspsoc: A secure sdn-based protocol over mpsoc," *Security and Communication Networks*, vol. 2019, 2019.

[61] E. Kaljic, A. Maric, P. Njemcevic, and M. Hadzialic, "A survey on data plane flexibility and programmability in software-defined networking," *IEEE Access*, vol. 7, pp. 47 804–47 840, 2019. DOI: 10.1109/ACCESS.2019.2910140.

[62] V. Netes and M. Kusakina, "Reliability challenges in software defined networking," in *Conference of Open Innovations Association, FRUCT*, FRUCT Oy, 2019, pp. 704–709.

[63] W. Rafique, X. He, Z. Liu, Y. Sun, and W. Dou, "Cfadefense: A security solution to detect and mitigate crossfire attacks in software-defined iot-edge infrastructure," in *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2019, pp. 500–509. DOI: 10.1109/HPCC/SmartCity/DSS.2019.00080.

[64] L. Csikor, M. Szalay, G. Rétvári, G. Pongrácz, D. P. Pezaros, and L. Toka, "Transition to sdn is harmless: Hybrid architecture for migrating legacy ethernet switches to sdn," *IEEE/ACM Transactions on Networking*, vol. 28, no. 1, pp. 275–288, 2020. DOI: 10.1109/TNET.2019.2958762.

[65] M. Emran and I. Kotuliak, "Performance analysis of traditional and sdn based handovers in wireless lan networks," in *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 2020, pp. 1–6. DOI: 10.1109/IEMTRONICS51293.2020.9216435.

[66] M. Hasan, H. Dahshan, E. Abdelwanees, and A. Elmoghazy, "Sdn mininet emulator benchmarking and result analysis," in *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, 2020, pp. 355–360. DOI: 10.1109/NILES50944.2020.9257913.

[67] S. Helali, "Simulating network architectures with gns3," in *Systems and Network Infrastructure Integration: Design, Implementation, Safety and Supervision.* 2020, pp. 9–25. DOI: 10.1002/9781119779964.ch2.

[68] H. Leqing, "How to realize the smooth transition from traditional network architecture to sdn," in *2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, 2020, pp. 1948–1952. DOI: 10.1109/ICMCCE51767.2020.00427.

[69] S. M. S. A. Abdullah, S. Y. A. Ameen, M. A. Sadeeq, and S. Zeebaree, "Multimodal emotion recognition using deep learning," *Journal of Applied Science and Technology Trends*, vol. 2, no. 02, pp. 52–58, 2021.

[70] D. Chefrour, "One-way delay measurement from traditional networks to sdn: A survey," *ACM Comput. Surv.*, vol. 54, no. 7, Jul. 2021, ISSN: 0360-0300. DOI: 10.1145/3466167. [Online]. Available: https://doi.org/10.1145/3466167.

[71] F. G and R. Anandhi, "Study on sdn with security issues using mininet," in Dec. 2021, ISBN: 9781643682167. DOI: 10.3233/APC210186.

[72] S. H. Haji, S. Zeebaree, R. H. Saeed, *et al.*, "Comparison of software defined networking with traditional networking," *Asian Journal of Research in Computer Science*, pp. 1–18, 2021.

[73] D. H. Maulud, S. R. Zeebaree, K. Jacksi, M. A. M. Sadeeq, and K. H. Sharif, "State of art for semantic analysis of natural language processing," *Qubahan Academic Journal*, vol. 1, no. 2, pp. 21–28, 2021.

[74] T. Jackisch, *Sdn vs traditional network*, Apr. 2022. DOI: 10.13140/RG.2.2.17889.38246.