UNIVERSITY OF EDUCATION, WINNEBA

IMPROVING THE BANDWIDTH OF THE SELF HOSTED LMS DATACENTER BY IMPLEMENTING AN ALGORITHM FOR DYNAMIC BANDWIDTH-AWARE LINK AGGREGATION CONTROL PROTOCOL (LACP).

BRIGHT KWASI KUNU

7191040024

A dissertation in the Department of Information Technology Education, Faculty of Applied Sciences and Mathematics Education, submitted to the School of Graduate Studies in partial fulfilment
of the requirements for the award of the degree of
Master of Science
(Information Technology Education)
in the University of Education, Winneba.

APRIL, 2022

DECLARATION

STUDENT'S DECLARATION

I Bright Kwasi Kunu, declare that this dissertation, except for quotations and references contained in published works which have all been identified and duly acknowledged, is entirely my original work, and it has not been submitted, either in part or whole, for another degree elsewhere.

SIGNATURE: ……………………………..

DATE: ……………………………………..

SUPERVISOR'S DECLARATION

I declare that the preparation and presentation of this work was supervised in accordance with the guidelines for supervision of thesis/dissertation/project as laid down by the University of Education, Winneba.

SUPERVISOR'S NAME: DR. JOSHUA DAGADU

SIGNATURE: ……………………………..

DATE: ……………………………………..

DEDICATION

I dedicate this work to my wife and children for their continuous support and prayers.

ACKNOWLEDGMENTS

I am grateful to my supervisor, Dr. Joshua Dagadu, for his sacrifices in directing and guiding me throughout the writing of this research paper and stirring my confidence to be a good researcher. I also wish to express my sincere appreciation to my pastor, Rev. Isaac Andoh-Krampah for the prayers and moral support.

In addition, I am grateful to Mis Grace Anopong (Headmistress of Boa Amponsem SHS) and her team of assistant heads for granting me permission to enroll and participate in this programme. Mr. Alexander Austin Kotun deserves a special acknowledgment for the instrumental role played in counseling anytime I call on him.

My colleagues, Mr. Muhammed Entsieh, Mr. Seth Mawusi Kpe, Mis Vida Smith, Mis Dorcas Addai, and Mis Mabel Pokoo, all of Boa Amponsem SHS ICT department. I deeply appreciate your efforts in standing in for me when I had to visit campus for this programme.

To those names I cannot mention who helped me in one way or the other, I wish to say, thank you sincerely.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY/ABBREVIATIONS

LMS – Learning Management System

LAG – Link Aggregation Group

LACP – Link Aggregation Control Protocol

SDN – Software Defined Network

QoS – Quality of Service

LA – Link Aggregation

ABSTRACT

Reliable network connectivity is an absolute requirement for running an online Learning Management System. This is particularly important for self-hosted LMS and the growing number of users who need a reliable and uninterrupted connection to the server. Many self-hosted systems are fraught with low bandwidth and its associated poor quality of service. Link Aggregation Control Protocol is one of the techniques that can effectively provide network link reliability and improved bandwidth at a very reduced cost. Link Aggregation Systems combine two or more links into one logical link having a single IP address. Even though an SDN LACP can improve link bandwidth, it often results in link redundancy. Therefore, this research set out to create and implement an algorithm that will allocate network bandwidth according to the dynamic behaviours of the LMS application.

The dynamic bandwidth-aware LACP system was simulated for 2-LAG and 4-LAG setups using the SDN Ryu controller. The simulation results show that LACP is indeed inexpensive yet effective in increasing link bandwidth and fault tolerance. However, the efficiency of the system depended on the number of links aggregated. Whilst the algorithm was excellent in improving bandwidth assurance and QoS for the 2-LAG LACP setup, the results obtained from the 4-LAG LACP setup show that as the number of aggregated links increases, delays and packets retransmission rate also increases. This suggests that the number of links that should form a LAG should be carefully selected according to the SDN controllers' ability to fully utilise all aggregated links to avoid link redundancy as observed in the 4-LAG LACP simulation results.

CHAPTER ONE

INTRODUCTION

A network is a layered system comprising end devices, connecting links, and gateways. The end devices have network applications that rely on the core network components for information exchange. Each layer is dedicated to processing a particular task of the network. Layers isolate network functions into discrete levels which communicate separately with similar levels in remote nodes (Freer, 1988). Layers usually collaborate with neighbouring layers to process tasks. For example, the application layer is at the top of the layered architecture where user interaction occurs. However, for communication from one application node to another to take place, the core network layers work to transmit information between the two end nodes.

Network technology has evolved over the years through various phases of fine-tuning the collaboration among various layers. The most current trend in computing and networking is virtualization technology. Network virtualization is the phenomenon designed for more efficient use of physical network resources. Therefore, virtualization is the technology that drives cloud computing.

1.1 Background to the study

Virtualization abstracts system hardware to act as if it were many devices that can be accessed separately (Campbell and Jeronimo, 2006). This is achieved by decoupling the logical network from the physical network, which serves as the hypervisor. Virtualization is suitable for automating network configuration, management, testing, deployment, monitoring, and operating physical and virtual devices within the network (Adato, 2017). Virtualization simplifies computer system deployment and migration, making room for the high availability of critical services. Additionally, virtualization simplifies IT operations and allows for faster response to business demands. System wide virtualization is an extensively recognized method because it is robust in resource sharing, user isolation, resource aggregation, user mobility and resource reallocation dynamics, and ease of management (Jain and Paul, 2013).

The purpose of automating a network is to improve productivity and efficiency. This is achieved by optimizing speed, reliability, flexibility, and security. When everyday network tasks, functions, and repetitive processes are automated, the true complexity of the network will be disguised (Molina and Jacob, 2018). In addition, network automation stands to improve the availability of services. Automation involves using programmable logic to control and manage network resources and services.

In recent times, Software Defined Networking (SDN) techniques are being studied as a resilient cloud virtualization solution. SDN is a cloud computing approach widely used in building and managing complex networks. SDN is high on automating network components and behaviour using programming languages. The SDN architecture runs a system that separates network services from the hardware components and yet configures the network automatically based on the services specification (Sezer et al., 2013). CISCO (2019), describes SDN-based networks as intent-based. That is, SDN provides a means for programmatic control of the network. Cisco opines that the IT outcomes of an organization should be translated into policies that the network can act on. These policies can then be installed across the physical and logical network through network-wide automation. It also provides the platform to monitor the network, to ensure that the desired intents have been applied to the business and that the outcomes are achieved.

As the concept of virtualization sharpens, a myriad of computer applications are created to change the pattern of human endeavours. Daily activities such as work, schooling, banking, governance, health, etc. have their virtual alternatives. For example, virtual schooling led to the introduction of blended learning approaches. These were later developed into a completely virtual education system called open education. The open education concept is a virtualized educational system where students school from the comfort of their homes or offices by interacting on online learning modules.

The open education system got wide acceptance all over the globe during the spread of the COVID-19 pandemic, which peaked in 2020. The violent spread of the pandemic and the fear of losing too many lives resulted in restrictions on human movement and gathering. Virtualization became the only safe means of human gathering for conducting meetings, training, work, and academic activities. That is, learning went online, meetings took place online, and work was done online.

Educational institutions that had an online presence were able to conclude their curricula at that time. Those institutions without an online presence were forced to introduce the same by quickly deploying the requisite technologies. Only a few institutions in Africa – Ghana, belonged to the class of online ready institutions. Most schools had to painfully endure the introduction of online learning to complete their curriculum. Fortunately, there are many open-source systems for online education management. This system helps to manage the entire education system from a single cloud-based Learning Management System (LMS) (Aldiab et al., 2019).

The cloud-based, open-source LMS is a prominent tool employed to transform the predominantly enclosed education system into an open education system. Thus, a student may learn all relevant courses, acquire skills, and pay fees without physically setting foot at the school or classroom. Consequently, educational media such as multimedia, video conferencing, and real-time simulation tools that are normally used in traditional classrooms had to be transmitted over the internet. This affected the already fragile peak time network connectivity in the country (Ghana). Learners and educators endure delays in remote connectivity to the LMS server.

Internet connectivity is a prerequisite for online virtual learning. Fortunately, GSM network coverage in Ghana has improved over the years. However, the high latency and poor Quality of Service (QoS) of most GMS networks at peak times across the country affected online activities. In addition, poor LAN/WAN setup and configuration hinder smooth remote communication to the LMS server even when the GMS network is at its best performance. Urera and Balahadia (2019) suggest that online education should be run over a networking infrastructure that provides seamless communication network connectivity. Thus, continuous, and smooth network connectivity is paramount in sustaining online learning. When users experience a seamless exchange of resources over the LMS, their interest is sustained in the teaching and learning process.

1.2 Problem statement

The proliferation of educational technologies resulted in making educational content available in varying forms with the availability of media. For example, interactive tutorials, video tutorials, computer-aided instruction, video conferencing, etc. are means by which teaching and learning occur. However, education in Ghana is generally enclosed. Thus, the education system is structured to respond to face-to-face interactions between students and educators. The physical presence of students and teachers in the classroom is so cherished that many educators use regular attendance as a predictor of a students' educational or academic success. This pattern of educational practice influences the way corporate and non-profit organizations conducted their training sessions.

Translating the face-to-face educational experiences to an online virtual learning environment is a nightmare for many Ghanaian students, and lecturers/tutors. One would have thought that the situation would be different at the higher learning institutions. On the Contrary, many higher learning institutions went through a hard time attempting to complete their curriculum, even by virtual means, when it became impossible to have face-to-face lectures during the heat of the COVID-19 pandemic. Those universities that managed with virtual learning during that period had to cope with the stress of slow network connectivity and non-responsive LMS. Some lecturers were forced to resort to sharing a pre-recorded video of their lessons on social media as opposed to using the modules of the LMS.

According to Speedtest global, fixed broadband connection in Ghana is the fastest in Africa and 79th in the world speed index (Ookla, 2021). In their publication, Ookla ranked Ghana's mobile broadband speed to be the 132nd in the world speed ranking. Thus, the fixed broadband connection at the LMS server is expected to be of a high speed as predicted by the global speedtest index. Notwithstanding, the actual speed available for communication on the LMS is determined by the configuration of the institution's LAN or WAN setup. According to Adarkwah (2020) Low bandwidth, or jammed network, accounts for 70% of the low-performance issues in online learning. Jamming in a network is usually caused by the poor configuration of the network (Du et al., 2017).

Solving the challenges that limit the efficiency of the LMS requires the adoption of modern and advanced protocols and services. A few researchers have proposed a dynamic load balancing mechanism as an efficient data transmission mechanism. Explaining that, load balancing is efficient in network congestion control. Queue management and selection of shortest routes are key attributes of load balancing mechanisms. However, load balancing has failed to resolve the bandwidth deficiency in communication links. Bandwidth is the amount of data that can be transferred from one point to another over a transmission link in a specific amount of time (Fisher and Heine, 2022). The transmission capacity of communication links is of the vital determinants of the quality and speed of a network. All the existing load balancing mechanisms that formed the basis for this work have failed to provide a mechanism for dynamic failover. Additionally, when the system is made of one application server connected by a single switch, load balancing by itself cannot provide for high availability application servers that require simultaneous connections.

Consequently, Link Aggregation (LA) mechanisms are introduced to increase the transmission capacity of the network links (IEEE, 2008, 2020). According to the proponents, link aggregation is a method of bundling two or more individual Ethernet links together so that they act as a single logical link. The bundled ethernet channels are called Link Aggregation Group (LAG). The protocol that runs and monitors LA is the Link Aggregation Control Protocol (LACP) (IEEE, 2020). It is responsible for the initial handshake of the LAG members and defines a load-balancing technique in response to link failure between the link aggregated devices (Irawati et al., 2017). LACP was first standardized by IEEE under 802.3ad and was later updated to 802.1ax. Since its first standardization, many network equipment vendors have developed their LACP compatible proprietary solutions. For example, CISCO has first developed the Virtual Switching System (VSS) for LA and later upgraded to the Virtual Port Channel (VPC) solution.

SDN and LACP implementation has gained high popularity in the networking spheres all over the world. However, in Ghana, one could rarely experience a continuous remote network connection to a university LMS without having to endure low bandwidth. More so, all the empirical works the researcher has sighted about SDN-based LACP implementation were focused on two things. (1) selecting an appropriate SDN controller and (2) algorithms that generally improve bandwidth and fault tolerance. No LACP implementation algorithm enforces dynamic bandwidth availability for critical educational applications severs. It is against this background that the researcher found it compelling to conduct this study to fill the knowledge gap.

1.3 Aim of the research

This research seeks to improve the bandwidth and QoS of the self-hosted LMS by proposing and simulating an algorithm for a dynamic bandwidth-aware link aggregation control protocol (LACP).

1.3.1 Specific objectives of the study

i. To propose an algorithm for configuring dynamic bandwidth-aware Link Aggregation Control Protocol (LACP) using an SDN OpenFlow Ryu controller.

ii. To analyze the extent to which the proposed algorithm helps improve the bandwidth allocation and usage of the self-hosted LMS server.

iii. To evaluate whether the proposed algorithm can improve the QoS of the self-hosted LMS site.

1.4 Relevance of the study

Persistent network connectivity is a strong requirement for any engagement in an online classroom. The competition for remote client connections to the online classroom platform is keen that the slightest misjudgment in connection passing can deny users access to the classroom. More so, when the hosting site of the online learning system is lacking in bandwidth, the likelihood of experiencing a series of packet losses is high.

Therefore, this paper proposes a cheap but efficient system that will reduce the complexity of the high-level educational system into low-level IT networking policies that can be implemented using a lightweight algorithm. The proposed algorithm will specifically make the self-hosted educational site dynamically expand its communication links bandwidth and make room for failover in moments of link failure.

The results of this study will be useful for planning and implementing online and open education systems. It will also be useful for educational and corporate institutions that would like to deploy their own LMS for training purposes and blended learning. In addition, the ideas and results of this paper will help businesses to improve bandwidth awareness of their self-hosted production systems.

1.6 Research design and methods

This study will be designed according to the design four-step experimental process for simulating solutions for network problems, proposed by Shamseldin and Bowling (2009). The first step is to gather information about the system and identify the system's metrics. The second step involves a description of the system's attributes and constructing logical simulation diagrams. The third step includes an analysis of the logical simulation diagram to identify ways to optimize the system based on the metrics defined in step one. The fourth stage is dedicated to evaluating the heuristic results based on the simulated logic diagram.

1.6.1 The major steps in the research process

- Provide background information detailing the suitability of SDN for network-wide automation and LACP for improving the bandwidth in self-hosted educational systems.
- Describe the proposed SDN-based LACP implementation algorithm.
- Experimentally simulate the proposed SDN-based LACP implementation algorithm for a self-hosted LMS server.
- Provide an analysis of the proposed algorithm to identify ways to optimize its implementation.
- Evaluate the experimental results from the simulation and suggest ways of improving the algorithm.

By adopting the above measures, this study hopes to address the problem of low bandwidth in the educational system. Using the proposed algorithm, the network resource allocation and management functions of the system will be automated. This intent-based algorithm will help provide high availability communication links for simultaneous transmission of real-time educational content over the LMS.

1.7 Limitations and Delimitations of the study

This study aims at proposing a python3 algorithm that will be suitable for providing network bandwidth assurance for high self-hosted productivity systems. It will aid in the planning and implementation of online virtual classrooms and distance learning.

However, some strenuous conditions could affect the smooth running and implementation of this research idea and results. Such conditions include.

1. Strenuous weather conditions can affect internet availability for both the hosting site and end-user connection.
2. Consistency of bandwidth provided by the service provider to both remotely connected hosts and for the self-hosted site. This is because, internet services in Ghana flip between a high 4G to as low as Edge or no connection at all, depending on the location.
3. Again, varying hardware specifications can alter the results of this research.

1.8 Definition of terms

Software Defined Network (SDN) is a cloud computing approach to network abstraction programmatically.

Link Aggregation Group (LAG) is a name given to the group of physical ethernet links that are aggregated to make a logical one.

Link Aggregation Control Protocol (LACP) is an IEEE 802.3ad standard for link bonding and monitoring.

Self-hosted is a software system installed and maintained by the user on a generic web hosting service.

1.9 Organization of the Study

This study will be organized into five chapters: introduction, literature review, methodology, implementation, discussion of findings, and recommendation.

Chapter 1: Introduction - presents an introduction to the research, comprising background, problem statement, the aim of the study, objectives of the study, research questions, conclusions, and organization of the study.

Chapter 2: Literature Review – provides a review of related research works on LMS implementation, Software Defined Networking (SDN), Link Aggregated Control Protocol (LACP), and other information about improving network bandwidth. The reviews are based on published articles and other resources such as published books, working papers, conference reports, and internet resources among others.

Chapter 3: Methodology – outlines the methods and procedures employed for conducting experiments, techniques used, programming language and algorithms used, and simulation diagrams.

Chapter 4: Implementation, Testing, and Results – this chapter presents the results of the experiments conducted and performance analysis graphs.

Chapter 5: Discussions, Conclusions, and Recommendations – This chapter provides a summary of the research, a discussion of the findings, conclusion, and recommendations based on the findings and related works from the literature reviewed.

CHAPTER TWO

LITERATURE REVIEW

2.0 Introduction

This study is aimed at improving the bandwidth and the Quality of Service (QoS) of self-hosted LMS by proposing an algorithm for implementing an SDN-based dynamic bandwidth-aware LACP. The LMS is a cloud-based application system that is high on simultaneous real-time data transmission. Network burstiness is usually associated with such high-demand application systems. Decongesting bursty systems is a major concern for the networking industry. Therefore, it has attracted considerable research attention over the years. A prominent network decongestion mechanism that has gained recognition among practitioners is load balancing (Boero et al., 2016b).

The ever-in demand for network resources has led to investments into more robust but cheaper ways of developing network solutions. This chapter critically reviews literature in the areas of LMS, Software Defined Networking (SDN), and Link Aggregation Control Protocol (LACP) implementation for improved bandwidth and QoS of high data application systems. The reviews are based on published articles and other resources such as published books, working papers, conference reports, and internet resources among others.

2.1 Communication on the LMS

The LMS can be decomposed into two broad groupings; (1) the components which comprise all the managerial modules and (2) the components which comprise all the modules for teaching and learning contents.

The managerial modules comprise, user accounts creation and login, the database which keeps records and students' marks, course management, students' management, and financial records, etc. These modules do not generate high network traffic. Users will simply log in and perform other specific lightweight tasks on the LMS. Whilst students simply log in to access their learning contents, other users of these modules simply perform some light traffic functions whenever they log in (Patel et al., 2014). Usually, simultaneous access is not mandatory for these modules. Even in the cases of simultaneous access, packets transmitted are not usually high on bandwidth. Therefore, these modules do not need high bandwidth reservation.

On the other hand, the teaching and learning modules generate the highest traffic on the LMS (Patel et al., 2014). After students and lecturers have signed into the LMS, they work with the modules which deliver the teaching and learning materials for a particular lesson. Examples include video conferencing, group discussions, class assignment presentations, examinations, etc. These modules are characterized by a large number of simultaneous access (Chien and Kao, 2005, Maneewongvatana and Maneewongvatana, 2013) and result in high data flow traffic on the network. Therefore, the is usually high congestion and packet loss as users compete for common network resources from their remote locations. Hence, these modules require bandwidth reservation to ease the problem of congestion in the data flow.

The network systems of the self-hosted LMS should accordingly be configured to provide high availability and better user experiences as they interact with the various modules. Denneman et al. (2018) stress the need for reserving network bandwidth to high traffic application systems following their bandwidth behaviours as observed from its different modules.

2.2 Software Defined Network (SDN)

The goal of every network administrator is to correctly map the high-level intent (policy) to the low-level forwarding behaviour of the network (Heller et al., 2013). Efficiently mapping the network policies to its traffic forwarding behaviour is usually confronted with problems of hardware requirements, bandwidth availability, traffic engineering, scalability, resilience against failure, cost, decentralized visibility of hardware, interoperability, etc. These problems are serious research concerns. Software Defined Network (SDN) provides solutions for most of the underlying networking problems (Mishra et al., 2017).

Firstly, the SDN architecture splits the network stack into control and data forwarding planes and slices the underlying network to ease construction, forwarding, and management (Heller et al., 2013). This enables network administrators to compose software programs that manipulate the logical map of each network slice (Yen and Su, 2014). Each slice represents a high-level intent of the operator (CISCO, 2019). A simplified view of the SDN layered architecture is shown in Figure 2.1.

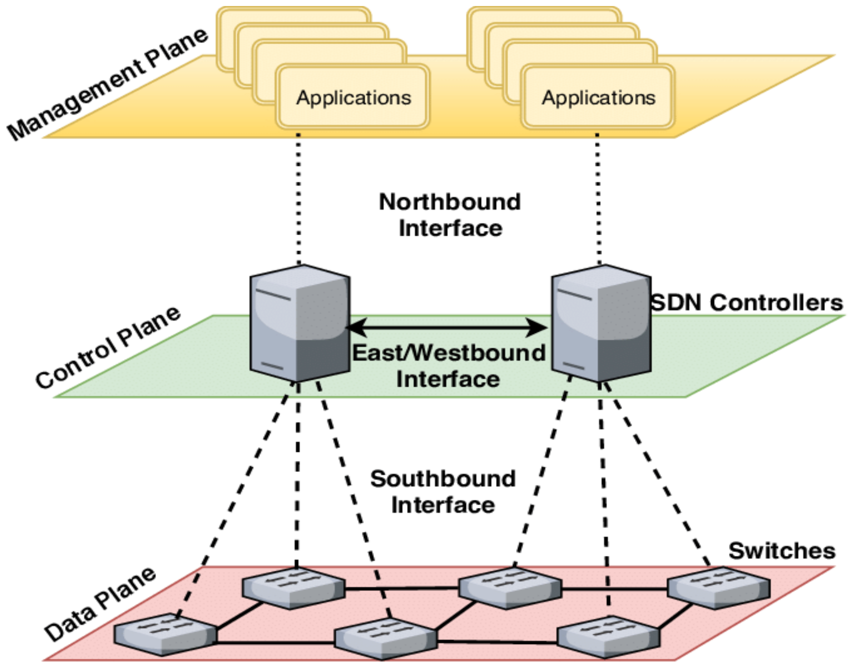


Figure 2.1: Layered view of SDN Architecture Adopted from (Latif and Sharif, 2020)

The SDN data plane/layer describes the underlying network traffic forwarding devices such as switches and routers and their communication links. The control plane is further broken into two layers: the network control plane and the application control plane. Thus, the entire SDN architecture has three layers/planes – the network/data plane, network control plane, and application control plane. The network control plane is the middle layer of the SDN architecture that serves as the hypervisor (Latif and Sharif, 2020). It describes the logical entity that takes instruction from the application layer/plane. It also provides the abstraction of the data forwarding devices. The network control logics are implemented at the control plane which is located in a logically centralized controller network operating system (Kreutz et al., 2015). Thus, the controller can be freely programmed and adapted to the network according to the network administrator's intents (Jarschel et al., 2013). The application plane provides the interface for user interactions with the network. Application programming interfaces (API) are placed southward and northward of the control plane (Latif and Sharif, 2020). The southbound API is used to enforce the regulator capability of the controller over the data plane elements. And the northbound API enforces the hypervisor ability of the controller over the network applications.

The southbound and northbound APIs are essential for running network configuration, supervision, and optimization from the controller (Zhou et al., 2014, Latif and Sharif, 2020). The API is how the controller exercises direct control over the state of the data plane elements. Thus, the network policies, their implementation, and traffic forwarding are all implemented from the controller (ONF, 2015). SDN networks are configured using high-level abstractions provided by SDN applications. By this, SDN developers can decouple global network-wide optimizations from individual network slices (functionalities) and perform configuration from a centralized controller.

2.2.1 SDN Controller Selection

Controllers are no longer a problem for SDN based networks. Many controller platforms have been developed over the years. For example, OpenFlow controller, NOX, POX, Nettle, OpenDayLight, FloodLight, Ryu, etc (ONF, 2015, Yamei et al. and Salman et al., 2016). OpenFlow protocol is developed according to the SDN layered architecture (ONF, 2015). The OpenFlow is usually implemented between the SDN controller and the OpenFlow enabled devices. And it serves as the standard for orchestrating both physical and virtual network devices. These controller platforms help in the development of many applications for network virtualization, energy-efficient networking, load balancing, dynamic access control in the enterprise network, Virtual machine mobility, etc. (Ma et al., 2015).

However, selecting a suitable controller has been a matter of concern to many researchers. The reasons for controller selection concerns are meant to avoid fatal errors in (1) using multiple controllers within a project (Abdullah et al., 2019), (2) interoperability between SDN and non-SDN networks (Mtawa et al., 2021), and (3) flow delays caused by controller placement (Long et al., 2015, Zhang et al., 2017).

Researchers have proposed that SDN controller selection should be according to their features (such as productivity, language, interface, and support given) (Tootoonchian et al., n.d., Khondoker et al. 2014, Abdul et al., 2019), their architecture, and efficiency (such as availability, scalability, fastness, and flow requests handling) (Mishra et al. 2017), and their QoS parameters such as delay and packets loss (Rowshanrad et al., 2016).

2.3 Application Aware Networking

Network transport infrastructure for end-to-end application data transmission is a determinant of how an application is effectively achieving its performance objectives (Al-Shehri et al., 2017). Thus, how well the network understands and serves the application requirements determines the performance of the application (Gatos et al., 2005). The known and monitored network features or metrics that affect application performance include bandwidth, latency, jitter, packet loss, error rate, etc. These determinants help the network orchestrator to define generic policies and allocate network resources required for the application (Jahromi et al., 2018). And the end-to-end transmission medium plays a major role in attaining the application efficiency requirements as reflected in user experiences. However, application requirements can change over time depending on the active use case and the number of concurrent users (Feldmann, A., et al, 2003).

Naïve implementation and monitoring of QoS metrics for application-aware networks are usually problematic. This is usually marked by dedicating network resources to an application unduly and hence requires some expertise to carry out. QoS marking on the IP header is one of the oldest techniques for creating application awareness. However, over the years, QoS marking on IP header had proven to be untrusted and therefore is rejected by network administrators.

SDN provides a simplified approach that ropes even inexperienced people into implementing policies and monitoring from a central location. The programmable northbound interface of the SDN acts as an abstraction layer that let applications declare their networking requirements with minimal networking knowledge (Lee et al., 2014, Soulé et al., 2014, and Gorlatch et al., 2014). For example, SDN OpenFlow controllers are capable of layer 2, 3, and 4 based policy enforcement (Qazi et al., 2013). However, the OpenFlow protocol can be leveraged to automate the data collection and application detection processes to incorporate layer 7-awareness into SDN.

QoS management in high load networks can sometimes be difficult. Jarschel et al., (2013) suggest the adoption of application stations for monitoring path selection (load balancing) (Aziz et al., 2016) mechanisms. These mechanisms can be utilized to receive current application flow information and buffer levels. When the buffer level gets below a certain threshold, the application station will inform the controller that an action is required for a particular flow to maintain the QoS. Therefore, combining application-state information with SDN-based application-aware networks is beneficial in improving control for network management. In addition, implementing QoS methods are better achieved through the reactive flow setup of the SDN controller. This can be implemented in both small and large networks.

2.3.1 Application aware load balancing

Load balancing is a networking technology that helps save power and improve resource utilization in the network. Data-intensive applications, with simultaneous users interactions with massive data, place varying demands on the network infrastructure. However, conventional campus network architectures which are usually without a global view of the network, rely on load balancers that are not precise (Sadiq et al., 2020).

Significant research has been conducted on SDN-based abstraction methods for efficient network traffic engineering which includes load balancing. For example, Application-aware Predictive Intelligent Load Balancer (APRIL), which intelligently combines application-layer metadata with deep learning predictive analytics to create customized load balancing policies (Neghabi et al., 2018, 2019).

Zhou et al. (2014) designed a prototype for SDN based load balancing using the PyResonance controller. PyResonance is an event-driven controller (Kim et al., 2012) which uses the Finite State Machine (FSM) model to define and implement network policy. The FSM model provides a useful abstraction mechanism for sequential circuits with a centralized state of operation. If the flow pattern is not consecutive, the FSM ability to accurately compute the next state of the machine to receive inputs will be affected.

Furthermore, many other known path selection mechanisms have not been efficient when deep learning is employed in SDN (Xu et al., 2018). They found that sampling and classification of traffic data based on application characteristics consumes a lot of I/O and computing resources of the SDN controller. Therefore, they proposed a model which adds a layer called virtual network function (VNF) to the SDN system. The VNF is a virtual machine that was set to perform port listening and traffic classification (Warraich et al., 2020) and finally forwards the information to the SDN controller.

Network administrators have a daunting task in selecting a load balancer for their networks. This is because of the increasing complexity of networks and the high demand for network resources on the go. Over the years, the most common qualitative parameters used by network administrators in selecting a load balancer are throughput, latency, and utilization. Many administrators found out that, load balancer selection based on these parameters was not effective since the business requirements of networks differ (Rajguru and Apte, 2012). More so, the load balancer that works efficiently for one network might be found defective for another. Therefore, many efficient qualitative parameters have been suggested in research. Examples, the average number of synchronization per minute (Neghabi et al., 2018), the accuracy of the algorithm that provides the exact amount of traffic in each queue for load balancing (Boero et al., 2016b), energy consumption (Hu et al., 2017), and degree of load balancing (Rangisetti et al., 2017), etc.

In addition, Neghabi et al. (2018) proposed a mechanism that uses qualitative parameters for load balancing. This method performs load balancing by measuring the average number of controller state synchronization per minute (Guo et al., 2014) and the frequency of traffic in each queue (Boero et al., 2016a) on the network.

In addition, Lee et al., (2014) present a paper at a SIGCOM conference that describes a controller-based abstraction for bandwidth assurance in datacenters. Their algorithm was named Tenant Application Graphs (TAG). This algorithm establishes a differentiation between the actual bandwidth requirement and the specified high availability of the tenant application server. However, the abstraction TAG algorithm has some leakage in assuring link failure for fault tolerance, which is a key factor in high availability.

Also, the LMS has a variety of traffic types it handles. for instance, real-time video conferencing, multimedia streaming, chat rooms, text-based files, etc. Real-time video conferencing and multimedia streaming are bandwidth intensive and require the dedication of end-to-end network resources to avoid link stress (A. Z. Khan et al., 2010). For this reason, Khan et al. (2012) proposed a path selection mechanism called Content Centric Network (CCN). This mechanism proposed bandwidth aggregation by path caching for multiple path selection to the content. However, path caching cannot make up for the high bandwidth requirements of a simultaneous access application. Giving room for a more robust mechanism of traffic management that meets the complexity of application demands on network resources.

2.4 Link Aggregation Control Protocol (LACP)

Link Aggregation is a technique for combining several physical communication links that have different IP addresses into one logical link with one IP address called Link Aggregation Group (LAG) (IEEE, 2020). LAG was defined by IEEE 802.1AX in 2008 for local and metropolitan networks and later revied in 2014 and 2020 respectively. LAG provides two main benefits; (1) increased capacity by balancing traffic across member links to provide aggregated throughput, and (2) link redundancy which provides quick recovery when one or more individual links are down or fail.

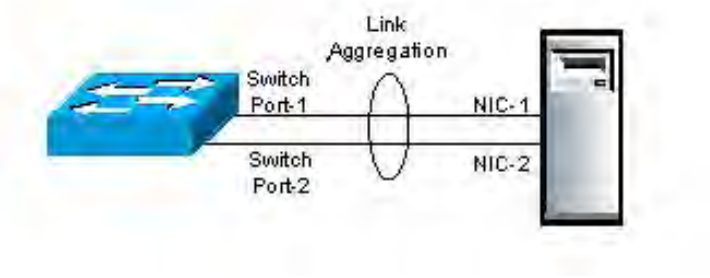


Figure 2.2: Link Aggregation

According to IEEE (2020), Link Aggregation Control Protocol (LACP) is the standard protocol that monitors and manages link aggregation. It allows devices to include or remove individual links from the LAG. Also, it protects the network from misconfiguration by ensuring that links are aggregated into a bundle if they are consistently cabled and configured. Additionally, LACP is responsible for the initial handshake of aggregated devices and defines a load-balancing technique in response to link failure between the aggregated devices (Irawati et al., 2017).

LACP can be configured in one of two ways: (1) Active mode where the device immediately sends LACP message whenever the port comes up, and (2) Passive mode where ports only respond to LACP in a passive negotiation state.

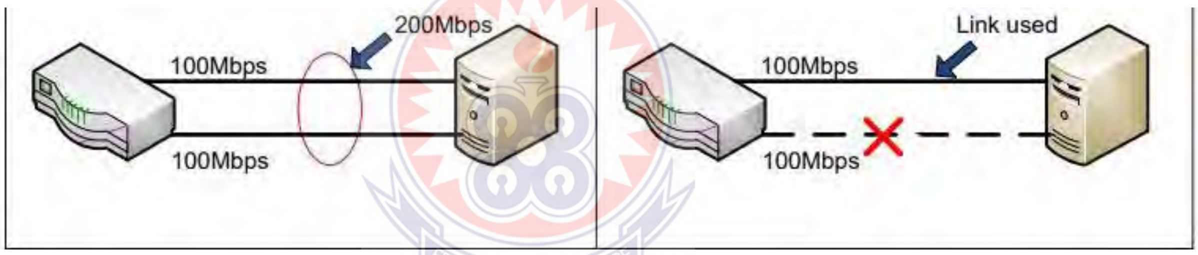


Figure 2.3: Active and passive LACP

2.4.1 The LACP use cases in research

Several researchers have considered LACP implementation in an SDN environment. For example, Irawati et al. (2017) worked on the dynamic LACP configuration for SDN. They used Python programming language to develop SDN based application for OpenFlow (Ryu) controller. After setting up the topology, the researchers configured Link Aggregation (LA) on the network server using a Linux kernel modprobe bonding four. In all, two ethernet interfaces were bonded between the OpenFlow switch and the Linux server. Upon configuring link bonding on the physical ports, the researchers configured LACP functions on the Ryu controller to perform fault tolerance. Wireshark data was collected for throughput and flow speed. Results of the experiment indicated that when the LACP data unit was passed through the topology every 30 seconds, flow speeds up to 10000Mbps were recorded for each of the Ethernet links. This shows that the two aggregated virtual links between the switch and the server will record data flow speeds up to 20000Mbps. However, they recorded an average data flow speed of up to 7413.33Mbps during normal flow. The researchers also found out that the setup used about 90 seconds to recover whenever one of the aggregated links become inactive. Data collected indicates that LACP implementation resulted in bandwidth improvement and a relatively quick recovery for Fault Tolerance.

In a study to explore the detailed implementation of LACP on SDN-based networks, Baskoro et al., (2019) compared LACP implementation in two SDN controllers (Ryu and Opendaylight). They wrote the LACP algorithm in Python and java respectively. Experiments were conducted using Mininet simulations with both the Ryu controller and Opendaylight controller followed by a hardware test-bed demonstration. The Ryu controller simulation was conducted following the method described in (Irawati et al., 2017). Whilst the Opendaylight simulation was according to the method in Opendaylight documentation 2018. Iperf software was used for monitoring traffic throughput on the aggregated host port. They also used Wireshark to observe the switchover time to collect data on the reply time before and after failover. The measurements in each case were repeated 10 times. The results of the experiments show that using a hardware test-bed experiment has higher and more stable throughput than virtual simulation. In addition, the throughput results indicate that Ryu and Opendaylight gave nearly the same throughput aggregation performance. However, the Opendaylight controller showed a faster switchover-time than the Ryu controller.

Baskoro and Risanuri et al., (2019) in another paper discussed an approach to modifying the Ryu LACP application or algorithm using Multiple Flow Tables (MTF). They focused on reducing the size of the flow table by lowering the number of flow entries. The experimental topology has an OpenFlow switch connecting several hosts, of which one is a server and has two link aggregation numbers. The simulation was modeled after the pattern of mininet simulation outlined by Irawati et al. To check for the efficiency of the topology, the researchers create a Single Flow Table (SFT) algorithm and a Multiple Flow Table (MFT) algorithm in the data plane. They then collected throughput and switchover time for both SFT and MFT. The total flow entries were calculated in each case. It was observed from the experiment that SFT produces more flow entries than MFT. This shows that MFT is much more suitable for reducing the number of flow entries. In addition, MFT produced better throughput and failover values than SFT. They explained that the good standard deviation figures of MFT were because MFT only implements LACP in an active mood.

Nurhadi et al. (2020) conducted a study to review LACP as a link redundancy tool in SDN networks. Objectives of the study include (1) giving an overview of Link Aggregation, and (2) experimenting to simulate the influence of LACP to increase bandwidth on SDN networks using Ryu controller. They set up two network topologies with different link aggregation numbers for the simulation. The first topology has two link aggregation numbers, and the second topology has four-link aggregation numbers. Various degrees of simultaneous traffic was passed through the two topologies and the bandwidth behaviour of the aggregated links was observed and recorded. Failover and round-trip times of the two topologies were also collected. Results from the experiment show that the bandwidth was improved if the simultaneous traffic transmission links were equal to or less than the number of aggregated links. However, no improvement in bandwidth was realized whenever the simultaneous traffic transmission links were more than the number of aggregated links. In addition, there was a reasonable delay in failover for the 8 links aggregation topology as compared to the two links aggregation topology. One notable conclusion of the studies was that the Ryu controller is not suitable for 8 links aggregation. They explained that a periodic burst of delay and duplicate packets were observed at the server. However, the work of Nurhadi et al. was silent on the tools used for the simulation, data collection, and methods of data collection.

Khan and Ali (2013) proposed an odd load balancing algorithm for implementing redundant links aggregation. They used an educational network for the experiment where they compared the Data Forwarding Module (DFM) values of the Virtual Router Redundancy Protocol (VRRP) with the DFM values of the proposed algorithm (Redundant Links Aggregation (RLA)). The module provides a traffic forwarding technique for the Odd Load Balance. To evaluate the network performance in peak time, two real-world scenarios were selected and RLA was compared with VRRP. The results show that VRRP failed to satisfy the demand of the network at peak hours. However, RLA was found to respond better to demands at peak hours. The outcome of the research shows that in peak hours the suggested RLA scheme performed better than the standard router redundancy protocol. This is because RLA effectively uses idle links as a backup while using the standard router redundancy protocol. In addition, the feature of adjusting the bandwidth according to the network demand makes it more effective.

Takiguchi et al. (2012) worked on a method for implementing Link Aggregation (LA) on the application layer. This enables LA for wireless and wired links to Android terminals. The LA method was configured for a server connected to a distributed network. A client can communicate to the server through any of the distributed networks. The proposed method was purposed to enable: (1) a dynamic link control function that switches between link aggregation communication mode and single link communication mode. (2) dynamic data distribution balancing ratio function which determines an appropriate distribution ratio between any of the two links. (3) data distribution ratio control function which controls the ratio for temporal link throughput variation during data transfer. The researchers built and configured LA on two experimental networks (Wireless-to-Wireless link and Wireless-to-Wired link) for implementing the proposed method. Link states were varied to measure the throughput values of the wireless link, wired link, and link aggregation. The study recorded that data distribution ratio control was achieved properly. In addition, the throughput graph plotted the recorded high throughput values for link aggregation when TCP packets were transmitted.

Data communication speed and network fault-tolerant enhancement over Software Defined Networking (Mohammad et al., 2018). The researchers set out to simulate link aggregation on SDN networks to improve network speed and performance. They created two experimental setups, one configured with link aggregation and the second one without link aggregation. Link aggregation was configured by enabling Linux link bonding. The topologies were simulated in Mininet for the Ryu controller. Results of data flow on the two topologies were captured using Wireshark. For each result, three performance analysis graphs (throughput graph, time-sequence graph, and round-trip-time graph) were drawn. A comparison of the throughput values of the two scenarios shows that data communication speed in the LACP SDN topology has improved by approximately 31% contrary to the topology without LACP SDN. In addition, the time sequence graph shows that more packets are transmitted with LACP SDN topology compared to without LACP SDN topology. Again, the RTT graph showed an improved network performance with LACP SDN topology than the topology without LACP SDN.

QoS negotiation system was proposed to address the problem of packet scheduling and QoS (Fernandez et al., 2009). The proposed system comprises a Dynamic QoS Negotiation module, Packet Scheduling module, and a Resource Manager module. The resource management module has components that divide the timeslot among users, called bandwidth allocation, and a mechanism to ensure the fairness and scalability in bandwidth allocation called bandwidth aggregation control. The dynamic QoS negotiation module has components for; creating a new QoS negotiation, a component to facilitate user profile exchange, and a component for renegotiating a running QoS session. The packet scheduling module aims to minimize reordering delayed packets at the receiver and the associated packet loss rate. Three video applications were used to transmit data of varying buffer sizes from three wireless terminals simultaneously. Data on QoS packet negotiation and packet loss rates were collected and analyzed. Based on the simulation results, the researchers proposed a TS-EDPF scheduling algorithm (which was a modification of the initial algorithm) that largely mitigates the unhandled issues of packet reordering and packet loss rate.

A critical analysis of the works of (Irawati et al., 2017; Mohammad et al., 2018; Nurhadi et al., 2020), and many others revealed an observed pattern running through the existing literature. Mininet simulation, Ryu controller, and most importantly Linux bonding were the most experimental platforms. Most of the works reviewed used the Linux bonding for Link Aggregation configuration.

Additionally, the trend in the literature shows a gap in knowledge as to designing a network that is aware of application context and responds to the intent of reserving network bandwidth following the observed application contexts. This paper seeks to close this gap by proposing a python3 algorithm that will reserve bandwidth according to application context by forwarding traffic on LA groups as LACP data units.

CHAPTER THREE

METHODOLOGY

3.0 INTRODUCTION

SDN has over the years proven to be a desirable network abstraction. Its controller-based network slicing simplifies and eases network construction, management, and monitoring. Small, medium, and complex distributed networks stand to benefit immensely from the capabilities of the SDN. However, an efficient SDN implementation does not completely eradicate the challenges with network traffic engineering (Sezer et al., 2013). Network administrators must go out of their way to enforce all traffic engineering policies on their networks to harness the full benefits of SDN. Notwithstanding, the SDN provides a fertile ground for introducing innovative software technologies that match network resources' demand to their supply (Nunes et al., 2014). For example, SDN-based LACP is a software solution introduced to reduce network congestions that result from many users competing for the same network resources (IEEE, 2020).

1.1 Research Design

This research is designed to simulate the dynamic bandwidth aware LACP implementation using a Mininet3.0 SDN simulator. This is because, the dynamic behaviour of LACP can only be empirically evaluated to provide a basis for real-life network implementation (Pahlevan, 2020). According to Maria (1997), "simulation modelling methods can be quite powerful in gaining insight into the behaviour of new and complex systems". Thus, the simulation method provides the steps in the design, development, testing, and evaluation of the system implementation. More so, simulation gives the ability to modify models to test system sensitivity and tune its performance (Aguado et al., 2011).

This simulation is set up to model and monitor the aggregated links' behaviour in terms of providing bandwidth assurance for the peak time activities of the LMS system, and its impact on the quality of service (QoS). The output of the simulation will provide numerical data that will be analyzed.

1.2 Proposed System

The goal of this research is to propose a system for dynamic Link Aggregation that will provide bandwidth assurance to the self-hosted LMS in its peak time. Implementation is towards easing congestion on the LMS by expanding the campus network bandwidth and improving end-user experiences on the LMS. The system performs bandwidth reservation according to the data flow context of the LMS. It has an event that collects data about packet flow characteristics of the LMS, another event defines rules to interpret the flow characteristics, and yet another event performs customized actions on the packets.

The proposed algorithm first observes and collects a log of data flow characteristics on the LMS. This includes port negotiations, port status, and bandwidth requirements for the packet flow. If a link is down, it will be captured as the port status for that negotiator. The system then triggers the traffic monitoring event which gives a context inference of the system. Thus, it defines heavy or moderate traffic situations. The rule for heavy traffic is set to 70% throughput rate of the link bandwidth. Based on the context inference, the system will provide bandwidth assurance in satisfaction of the flow context recorded. In this case, the controller will decide whether to transmit the packets as LACP data units or to perform round-robin load balancing on the available physical links.

The self-hosted LMS server is expected to exhibit a peak time behaviour all day long. Thousands of remote simultaneous negotiations for network resources must be processed throughout the day. This typically defines a peak-time operation of the campus network. Therefore, the proposed algorithm provides a traffic shaping mechanism to ensure all users have an overall favourable connection to their learning resources. For example, on a typical lecture day, thousands of users compete for network access to real-time services from the LMS. The Ryu controller will keep a log of all network resource negotiations and will estimate bandwidth requirements for port connection requests it received.

Based on the estimated bandwidth requirement, the controller will sanction traffic flow over the LAG links. This is done by calculating the total throughput rate of the requested link bandwidth and comparing the results to the 70% minimum rule set in the algorithm. If the calculated throughput value is greater than the minimum for a particular link, the controller will simply sanction the packet to be transmitted over the physical links available. However, if the throughput value is less or equal to the minimum for the requested link, then the controller will direct the traffic over the LAG links. Figure 3.1 is the flow diagram showing how the algorithm logically processes packets.

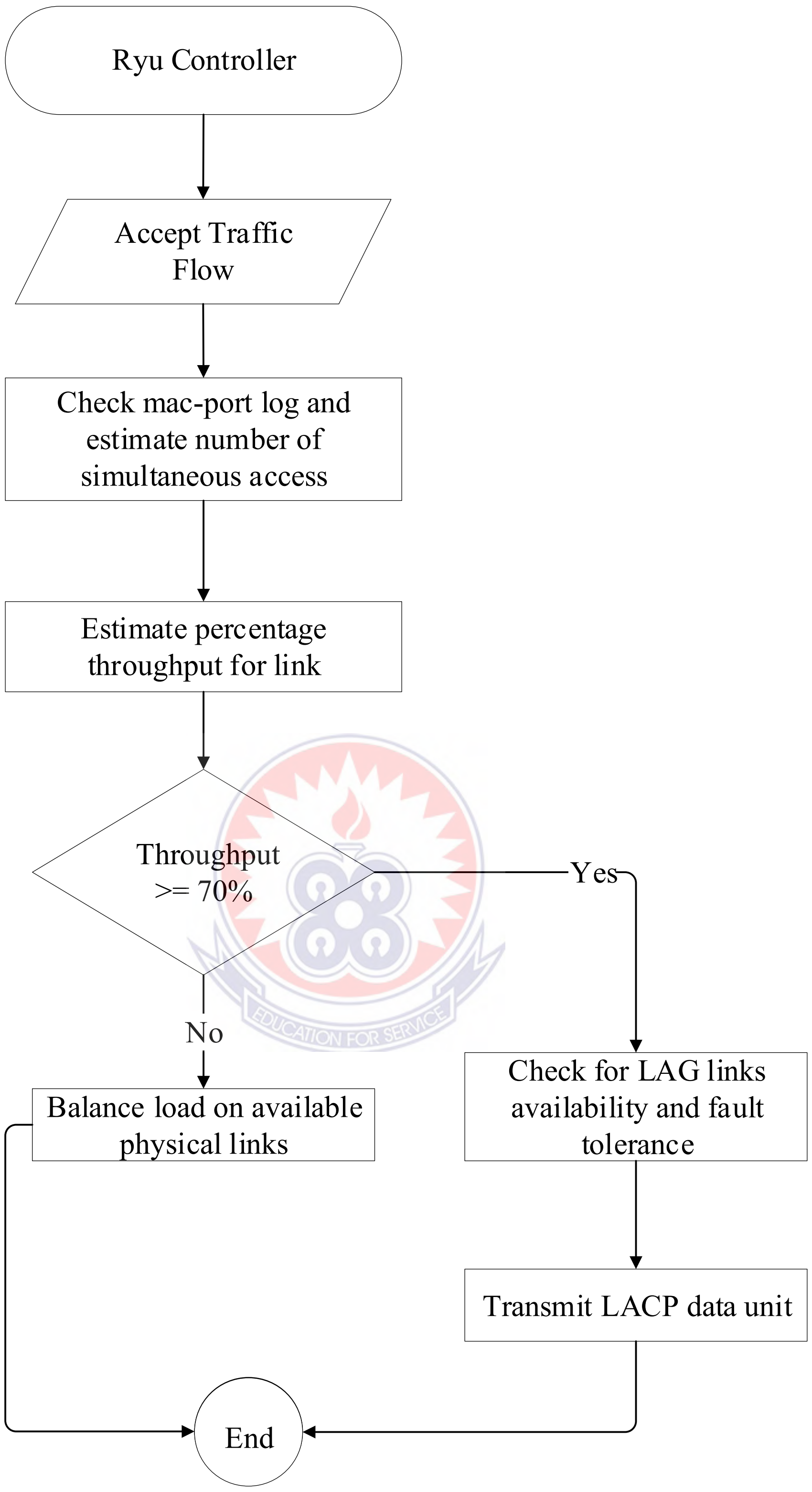

Ryu Controller
Accept Traffic Flow
Check mac-port log and estimate number of simultaneous access
Estimate percentage throughput for link
Throughput >= 70%
Yes
No
Balance load on available physical links
Check for LAG links availability and fault tolerance
Transmit LACP data unit
End

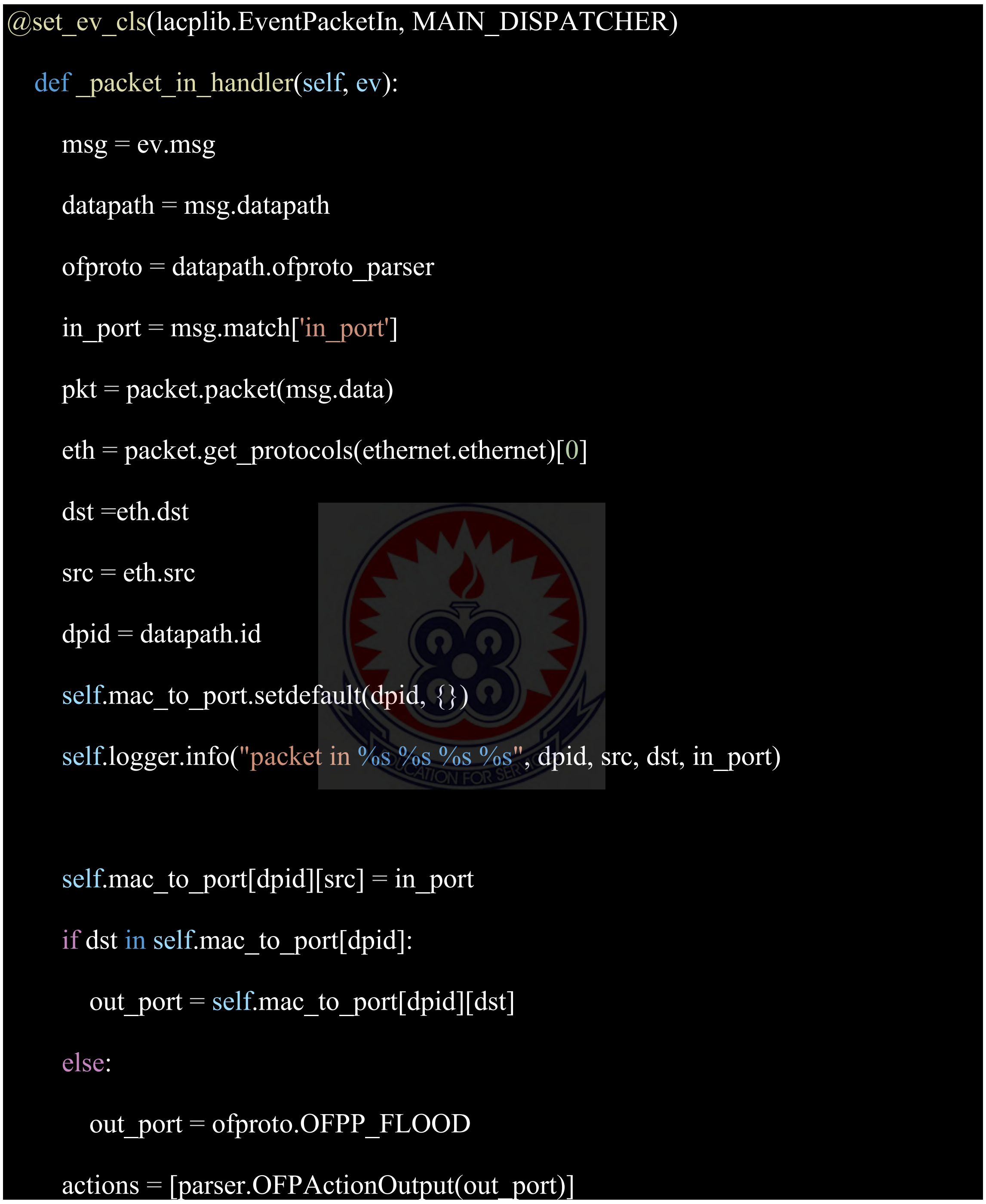
Figure 3.1: A Flow diagram for context aware LACP bandwidth assurance

3.2.2 Context Awareness

This study proposes an SDN algorithm that aggregates two or four ethernet channels according to the IEEE 802.3ad LACP standards. The algorithm has an event handler class that is triggered anytime a new layer two (L2) switch is added to the topology and whenever a switching event is requested. The event handler first installs a flow entries table called table-miss in the switch. The event handler class's helper method defines the priority for adding flow entries to the table-miss table. Any flow requests that did not meet the priority specified by the helper method and are not matched with any entries of the table-miss table are sent to the Ryu controller to decide on how to forward them. The MAC address table (MAC-to-Port table) for the DPID of the current switch is then created. The packet information is logged, and the MAC-to-Port table is finally updated with the packet's source address associated with the port it arrived on. After learning the destination mac address as recorded in the MAC-to-port table, the system sets the output port of the packet to the learned port. If the mac information is not found in the mac-to-port table, the packet is set to flood.

In addition, the algorithm describes a python3 decorator @set_ev_cls in Ryu.controller.handler that monitors the event-port-modification class. This decorator performs the function of keeping port state information and calculating link throughput. In this instance, the LACP data unit forwarding priority was set to 70% throughput rate for a single simultaneous connection to a single LMS application module. At the interval of every ten seconds, the system analyzes the throughput in Mbps. Then the system performs bandwidth calculation on each port of the simulated switch. The results of these calculations help the Ryu controller to make adaptations to the network and links. The main network adaptation provided for in this work is the decision to forward traffic as

LACP data using the LA group bandwidth assurance handler or otherwise. Figure 3.2 shows the python3 script for creating context awareness.



```
@set_ev_cls(lacplib.EventPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto_parser
        in_port = msg.match['in_port']
        pkt = packet.packet(msg.data)
        eth = packet.get_protocols(ethernet.ethernet)[0]
        dst =eth.dst
        src = eth.src
        dpid = datapath.id
        self.mac_to_port.setdefault(dpid, {})
        self.logger.info("packet in %s %s %s %s", dpid, src, dst, in_port)

        self.mac_to_port[dpid][src] = in_port
        if dst in self.mac_to_port[dpid]:
            out_port = self.mac_to_port[dpid][dst]
        else:
            out_port = ofproto.OFPP_FLOOD
        actions = [parser.OFPActionOutput(out_port)]
```

Figure 3.2: Python3 script for creating application awareness in L2 OpenFlow Switch

3.2.3 Dynamic Bandwidth Assurance

The system is configured to function in both active and passive LACP modes. LACP packets are transmitted in active mode whenever the learned ports recorded in the table-miss of the switching hub are matched and packets are forwarded to the learned port. This is done by binding the LAG group dpid to the learned ports in the switching hub. Making the LAG group actively open for monitoring in the intervals of 30 seconds. The LACP settings (modeprobe 4) in the ubuntu system sets the logical EtherChannel (bond0) as the master channel and the physical EtherChannel (h1-eth0 and h1-eth1) for two channels aggregation or (h1-eth0, h1-eth1, h1-eth2, and h1-eth3) for four channels aggregation, are set as slave channels. Whenever the LACP data unit is exchanged, the aggregated physical links will be enabled, and the load will be balanced on them at the combined bandwidth of the logical interface. Packets other than the LACP data unit are learned and transmitted using the switching hub algorithm. If no LACP data unit is received in a specified time interval, the indication is that the physical interface is disabled. This demonstrates how the algorithm monitors the changes that occur in the slave state by the way events of the Ryu controller are executed. Figure 3.3 shows the python3 script for implementing bandwidth assurance for LACP data transmission.

```
@set_ev_cls(lacplib.EventSlaveStateChanged, MAIN_DISPATCHER)
  def _slave_state_changed_handler(self, ev):
    datapath = ev.datapath
    dpid = datapath.id
    port_no =ev.port
    enabled = ev.enabled
```

```
self.logger.info("slave state changed port: %d enabled: %s", port_no, enabled)
if dpid in self.mac_to_port:
    for mac in self.mac_to_port[dpid]:
        match = datapath.ofproto_parser.OFPMatch(eth_dst=mac)
        self.del_flow(datapath, match)
    del self.mac_to_port[dpid]
self.mac_to_port.setdefault(dpid, {})
```

Figure 3.3: Dynamic bandwidth assurance for LACP data unit transmission

In addition, if for any reason one of the links in the LAG fails or is disabled, the lacplib will direct traffic to the active link. So, in a LA group, the bonded physical channels always serve as active backups for each other. The actual bandwidth of the backup link remains the same as the bandwidth of the logical interface (bond0), but the maximum bandwidth that will be utilized will be the actual bandwidth of the backup link(s). For example, in the four-channel LAG, if h1-eth0 is down for any reason, the LACP data unit will be transmitted over the remaining channels (h1-eth1, h1-eth2, and h1-eth3) in the LA Group. However, the actual bandwidth of the transmission is the summation of the bandwidths of the used channels.

The algorithm observes when a slave channel's current state orchestrates an idle timeout for the flow entry that performs packet-in of the LACP data unit. A timeout may occur because of a link failure or when the physical interface is disabled. The system will print the change of state to disabled and then send a FlowRemove message to the switching hub. In this instance, packets-in flow is redirected to the Ryu controller, which decides to transmit the packet over a backup link. Therefore, when the enable/disable state of a physical interface is changed, the FlowRemove event handler will delete all flow entries that use the physical interfaces included in the logical interface to which the disabled physical interface belongs. This makes LACP suitable for constructing highly available network links and provides a quick switch for fault tolerance.

1.3 Experimentation

3.3.1 Experimental Materials and Tools

Firstly, Ubuntu server 21.04 was installed on a VMware Workstation 16 pro. The choice of ubuntu 21.04 was occasioned by the desire to provide the most current stable network operating system environment with advanced features and flexibility for modelling the experiment. The system requirements for installing ubuntu 21.04 make it convenient to mitigate hardware constraints that could influence the results of the experiment. Additionally, the system setup was purposed to improve users' experience in using a self-hosted system comparable to the user experiences using a cloud-hosted system. More so, the goal was to solely use the superior networking and simulation features provided by the ubuntu server.

OpenFlow1.3 was used for communication between the SDN control plane and Open vSwitch data plane in Mininet 3.0 environment (Ferguson et al., 2013). Mininet is a lightweight virtualization environment that uses lightweight Operating System containers (such as Linux shell) for building a complete network within a single OS instance (Lee et al., 2014). And it has its virtual Ethernet interfaces. In addition, Mininet helps to verify the functional correctness of programming codes used in network experiments. Its components provide the basis for verifying performance properties for a much wider range of experiments. Scripts for the algorithm were written in Python3.

Traffic scenario instances of the proposed framework use the Ryu modular SDN controller. Each module of the Ryu controller listens for specific events and catalogues the same. When the Ryu controller receives events, it processes them in a FIFO queue (Ryu project team, 2014). Ryu has an event processor which dequeues the received event queues and calls the appropriate event handler. The proposed framework has a module that tracks the LACP state based on LACP data units exchanged. A module that maintains links utilization and ports status. Another module processes application requests for validity based on network snapshot information and administration policies and evaluates whether the request complies with its criteria. Finally, another module implements the network flow behaviour in the data plane.

3.3.2 Experimental setup

Two network topologies were created in Miniedit for the experiment as shown in figures 3.4 and 3.5 respectively. Figure 3.4 had two link aggregation groups and figure 3.5 had four-link aggregation groups. Both topologies have the same LACP functions configuration. Thus, the proposed algorithm was tested with both topologies. This was to access the extent to which the number of LA groups in an LACP influences bandwidth awareness in a heavy traffic network.

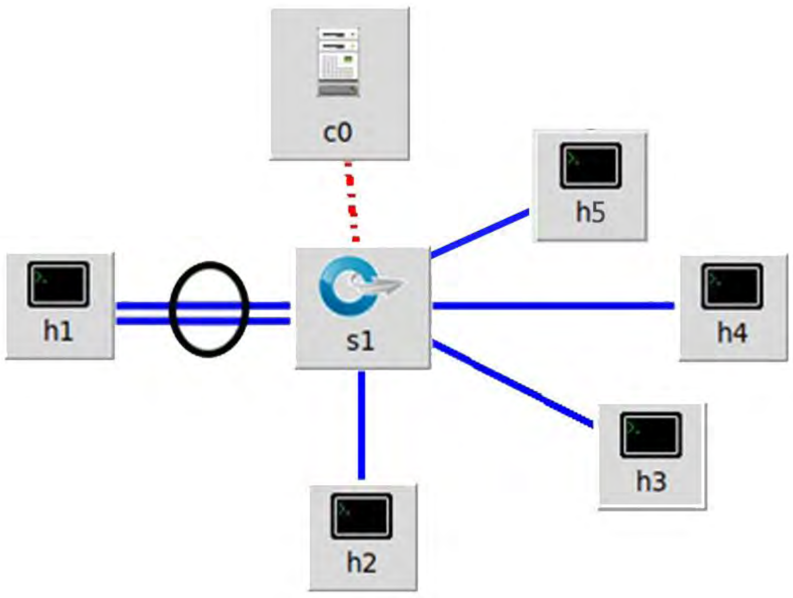


Figure 3.4: Two *links LAG Topology*

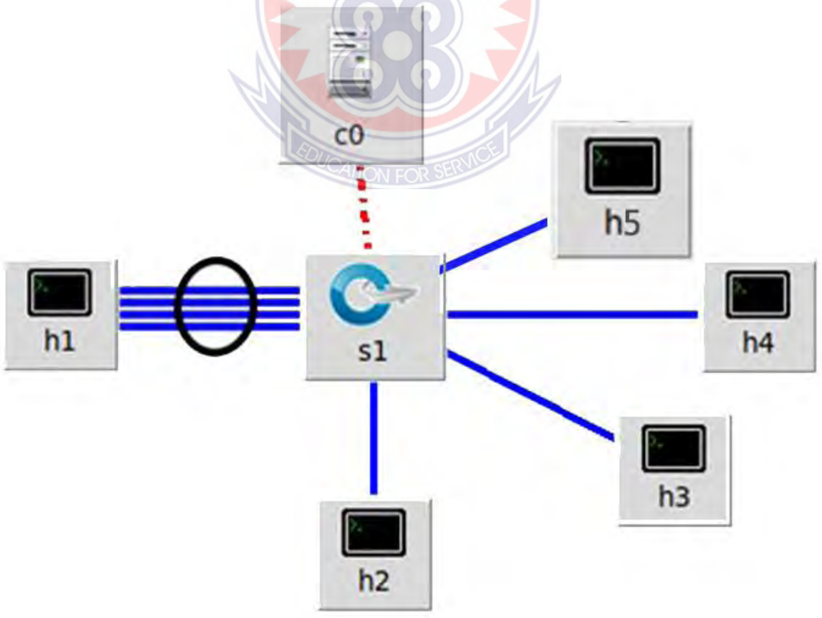


Figure 3.5: Four links LAG Topology

3.3.3 Simulation of the proposed algorithm

Firstly, the data plane, control plane elements, and network topology were programmatically created in python3 and were saved in 2 different .py files. The .py file for the network topology was run in the ubuntu console in an activated mode using the ubuntu script "sudo python3 topo.py" (where topo is the name of the topology file created in python3). This creates the network in Mininet where all the hosts, switches, routers, controllers, and links can be seen. Further configurations are then done on the various devices and APIs can be installed on the controller. For example, in this experiment, the controller and network behaviour are then installed on the host named c0 (controller) by running the .py file in the c0 ubuntu console. Also, a simple web server script was run on h1 so that it can serve as the LMS server in this experiment. The final step in the network configuration was the creation of the virtual link using Linux bonding and then setting the virtual link (bond0) up as the master link for data transmission and the physical links as slaves to be used for failover.

Test traffic was generated to measure bandwidth and other QoS parameters such as latency, jitter, packet loss, and fault tolerance. The testing process was repeated using iPerf3 and ping (ICMP) until the system continuously produces the same results three consecutive times.

3.3.3.1 Bandwidth

Bandwidth of the links between h1 and the other connected hosts h2, h3, h4, and h5 was measured by running the iPerf3 Linux utility. The server side iperf3 was run on h1 (which was set up as the LMS server) and the client-side iperf3 was run from h2, h3, h4, and h5 respectively. Individual and simultaneous client TCP and UDP iPerf3 packets were transmitted from h2, h3, h4, and h5 respectively to measure the bandwidth of the connections to the server (h1).

Each host h2, h3, h4, and h5 took turns to transmit TCP and UDP iPerf3 packets to the server (h1) for 15 seconds and their respective bandwidths were recorded. After which all the connected hosts h2, h3, h4, and h5 transmit TCP and UDP iPerf3 packets to the server (h1) simultaneously and their bandwidths are recorded.

3.3.3.2 Latency

To be sure that bandwidth assurance can improve the QoS of the self-hosted LMS datacenter, it was necessary to know how quickly data travels between the server (h1) to the other hosts (h2, h3, h4, and h5). Thus, the latency of data transmission between the connected hosts was measured for both the two links aggregation and the four links aggregation, and during their fault tolerance testing. This is achieved by transmitting TCP and UDP ping (ICMP) packets from the connected hosts (h2, h3, h4, and h5) to the server (h1). The packets transmission rate was observed in wireshark, and the latency figures were recorded in a .txt file for further analysis.

3.3.3.3 Jitter

It is important to observe the variable delays in end-to-end packet transmission between the hosts (h2, h3, h4, and h5) and the server (h1). Thus, to measure the quality of the packets delivered to the host computers on the network per second. This is done by transmitting 50G of TCP and UDP iPerf3 packets as described in 3.3.3.1.

3.3.3.4 Packet Loss

As described in 3.3.3.2, the UDP and TCP ping (ICMP) packets were recorded in a .txt file and observed in Wireshark for packets that were lost during transmission or could not receive a response in the ping process.

3.3.3.5 Fault Tolerance

After collecting data about how effectively the LACP application has influenced bandwidth reservation and QoS of the LMS, the system was reconfigured by deleting the master status of the virtual link (bond0) to check how the system responds to faults. In this case, traffic is channelled through the available physical links to the sever (h1). The bandwidth and jitter figures were again analysed.

1.4 Evaluation Metrics

The proposed system was evaluated on bandwidth, latency, jitter, packet loss, and fault tolerance. And data collected were analyzed and visualized using originlab 2020.

3.4.1 Bandwidth

Bandwidth is the volume of data that can be transmitted over a network transmission medium per unit time (Fisher and Heine, 2022). Thus, the data transfer rate of a network transmission medium in a unit time is determined by its bandwidth and it is measured in Megabits per second (Mbps). As bandwidth increases, the amount of data that can flow through the transmission medium at a specific time also increases (Norman, 2017). The allowed bandwidth of a network is usually shared among all the hosts accessing resources on the network at a particular time and according to the type of resource being accessed. This is done according to the bandwidth specifications of the network administrator. For example, this research is set up to experiment on dynamic bandwidth reservation for an LMS (which is a bandwidth sensitive application), and it is simulated as described in 3.3.3.1.

3.4.2 Latency

Latency is an expression of how much time it takes to transmit a data packet between two hosts in an end-to-end communication (Gillis, 2020). This describes the time taken for connected hosts to transmit or load data from the LMS server. A low latency is an indication of a faster time for data transmission, and a higher latency is an indication of a slower data transmission time. Gillis added that networks with large bandwidth are likely to produce a lower latency, thus, faster delivery of packets or a good round trip time.

The ultimate aim of this research is to improve end users' experience in accessing data on the LMS datacenter by improving the bandwidth. Therefore, the tools and processes described in 3.3.3.2 were followed to evaluate the latency of the experimental network created.

3.4.3 Jitter

One factor that affects the latency of a network delivery is the jitter. Liang et al. (2021) describe jitter as the time delay in packet transmission over the network. This is observed and measured as described in 3.3.3.3 of this research work.

3.4.4 Packet Loss

Packet loss is the expression for the number of packets that could not make it to their destination (Gillis, 2020). This is observed and measured as described in 3.3.3.4 of this research work.

3.4.5 Fault Tolerance

Fault tolerance is the ability of a network to deliver uninterrupted service, despite the failure of one or more of its components (Mohammad et al., 2018, Kranz, 2021). This research work has a principal focus on accessing the performance of network transmission media. Hence, 3.3.3.5 describes how the system response to faults in the links was accessed and mitigated.

CHAPTER FOUR

RESULTS AND ANALYSIS

The primary concern of any network administrator is to deliver uninterrupted service to the clients. Thus, when a client transmits a packet over the network, it must reach its destination without delay and uninterrupted. Therefore, this section presents results obtained from the implementation of the dynamic LACP algorithm in Mininet as described in chapter three. The results cover the test outcomes for Bandwidth, Latency, Jitter, Packet loss, and fault tolerance.

4.1 Bandwidth

4.1.1 Bandwidth in two-LAG LACP

Table 4.1 presents the network bandwidth recorded as 1GB of iPerf3 UDP stream datagrams was transmitted from each of the connected hosts (h2, h3, h4, and h5) to the server (h1) for 15 seconds.

Table 4.1: Bandwidth for iPerf3 UDP stream datagram in two LAG LACP.

Interval (sec)	h2_Bandwidth (Mb/s)	h3_Bandwidth (Mb/s)	h4_Bandwidth (Mb/s)	h5_Bandwidth (Mb/s)
1	979	992	978	997
2	998	977	973	985
3	961	981	1000	978
4	1030	996	996	965
5	999	989	994	1000
6	1000	995	1010	984
7	982	983	998	1010
8	987	997	1000	994
9	985	1010	996	982
10	987	993	1000	980

11	1010	1000	985	1010
12	991	1000	999	986
13	1000	994	1000	1000
14	995	980	991	1000
15	1000	979	990	951

From table 4.1, bandwidth was allocated for each datagram stream according to the amount of datagram transmitted. An average bandwidth of 11900.4Mbps for the 15 seconds of observed data transmission. Which is 60.5% improvement in bandwidth reservation of a self-hosted system over the work done by Irawati et al. (2017), which recorded an average of 7413.33Mbps for 600 seconds of observed data transmission. Figure 4.1 is a graphical presentation of the UDP stream datagram variations in the lines-point graph.

Figure 4.1: Bandwidth for iPerf3 UDP stream datagram in two LAG LACP.

1GB of iPerf3 TCP icmp packets was then transmitted simultaneously from the connected hosts (h2, h3, h4, and h5) to the server (h1) for 15 seconds as shown in table 4.2.

Table 4.2: Bandwidth for iPerf3 TCP icmp packets in two LAG LACP setup

Interval (sec)	h2_Bandwidth (Mb/s)	h3_Bandwidth (Mb/s)	h4_Bandwidth (Mb/s)	h5_Bandwidth (Mb/s)
1	1000	1000	998	999
2	999	1000	1000	1000
3	1000	1000	1000	1000
4	999	1000	1000	1000
5	1000	1000	1000	999
6	1000	1000	1000	1000
7	1000	999	999	999
8	999	1000	1000	1000
9	1000	999	1000	1000
10	1000	1000	1000	999
11	999	1000	999	1000
12	1000	1000	1000	1000
13	1000	1000	1000	999
14	1000	1000	1000	1000
15	1000	1000	999	1000

Table 4.2 shows maximum reservation of network bandwidth of all streams of icmp flood transmitted simultaneously from the connected hosts to the server. The lowest bandwidth observed for a packet stream was 998Mb/s whilst the highest bandwidth was 1000Mb/s. Thus, providing a highly reliable bandwidth assurance, since 98% of icmp streams came through with the highest reserved bandwidth.

Figure 4.2. shows the bandwidth graph for 15 seconds of iPerf3 TCP packets in two LAG LACP setup

Figure 4.2: Bandwidth for iPerf3 TCP packets in two LAG LACP setup

4.1.2 Bandwidth for four-LAG LACP

Table 4.3 presents the network bandwidth recorded as 1GB of iPerf3 UDP datagram streams were transmitted from each of the connected hosts (h2, h3, h4, and h5) to the server (h1) for 15 seconds in the four LAG LACP experimental setup.

Table 4.3: Bandwidth for 15sec of iPerf3 UDP datagram stream in four LAG LACP setup

Interval sec	Bandwidth_h2 Mb/s	Bandwidth_h3 Mb/s	Bandwidth_h4 Mb/s	Bandwidth_h5 Mb/s
1	264	452	594	693
2	317	885	387	776
3	643	573	258	86
4	417	434	310.3	521
5	205	97.8	250.2	955
6	650	6.35	380.3	649
7	436	282	340.8	482
8	703	903	430.2	663
9	449	477	300.4	546

10	748	700	570.1	359
11	665	1000	390.3	464
12	954	869	340.8	524
13	419	656	410.5	544
14	835	354	150.2	461
15	282	280	340.4	641

From table 4.3, bandwidth was allocated for each datagram stream according to the amount of datagram transmitted. This shows a wider variation in datagram streams according to the volume of data transmitted. This varying and low allocation of bandwidth is indicative that increasing the number of aggregated links does not necessarily increase bandwidth assurance. Therefore, this system is prone to recording out of order packets and a resultant burstiness. Figure 4.3 shows the varying lines-point.

Figure 4.3: Bandwidth for iPerf3 UDP datagrams in a 4-LAG LACP

Table 4.4 shows the results of the simultaneous transmission of 1GB iPerf3 TCP icmp packets from the connected hosts (h2, h3, h4, and h5) to the server (h1) for 15 seconds.

Table 4.4: Bandwidth for iPerf3 TCP icmp packets in four LAG LACP setup

Interval sec	Bandwidth_h2 Mb/s	Bandwidth_h3 Mb/s	Bandwidth_h4 Mb/s	Bandwidth_h5 Mb/s
1	360	500	500	500
2	775	498	500	500
3	500	227	233	500
4	499	1000	814	500
5	414	165	505	499
6	595	1000	500	500
7	500	500	500	499
8	319	500	500	501
9	467	500	500	499
10	866	499	500	501
11	500	500	499	500
12	499	357	314	499
13	500	251	670	500
14	330	900	120.6	500
15	795	540	1000	455

Table 4.4 shows an irregular distribution of network bandwidth for each stream of icmp flood transmitted simultaneously from the connected hosts to the server. This is a confirmation of the observations made on the results from table 4.3. Thus, irrespective of the type of packets transmitted in the 4-LAG LACP and higher number of link aggregation group, poor bandwidth assurance was evident. This also shows that OpenFlow1.3 was inefficient in maximising the bandwidth capabilities when 4 links were aggregated. This is because the system could not utilize the aggregated virtual link for

transmitting packets, since only two of the physical links were active in LACP data transmission.

Figure 4.4. shows the graphical presentation of the results from table 4.4

Figure 4.4: Bandwidth for iPerf3 TCP icmp packets in four LAG LACP setup

Comparing the bandwidth allocation and assurance in both the Two-LAG LACP setup and the Four-LAG LACP setup, it became evident that the Two-LAG LACP setup is more reliable in bandwidth assurance than the Four-LAG LACP setup as shown in figure 4.5.

Figure 4.5: Comparing bandwidth assurance in Two-LAG LACP and Four-LAG LACP

4.2 Latency

4.2.1 Latency for two LAG LACP

1000Mb of ping flood packets were transmitted from the connected hosts (h2, h3, h4, and h5) simultaneously to the server (h1) for 100 seconds and their minimum latency, average latency, maximum latency, mean deviation, interframe spacing, and exponential weighted moving average values were recorded as shown in table 4.5

Table 4.5: Summary of latency report for 100s of ping icmp flood in two LAG LACP setup

Hosts	Min Latency (ms)	Avg Latency (ms)	Max Latency (ms)	Mdev (ms)	Ipg (ms)	Ewma (ms)
h2	0.007	0.093	5.398	0.612	0.11	0.008
h3	0.009	0.102	4.89	0.557	0.124	0.009
h4	0.006	0.499	31.849	3.467	0.374	0.008
h5	0.007	0.113	6.024	0.695	0.135	0.013

Table 4.5 shows that it took an average of 0.807ms to transmit a total of 400 icmp packets in a simultaneous TCP flood of 4000Mb of data transmitted. This indicates a fast end-to-end latency.

The average latency for h4 was the highest for all hosts because there was a relatively high interpacket gap (ipg) observed. However, this ipg was an isolated case and could not result in the transmission of duplicate or out-of-order packets. Figure 4.6 shows a graph that compares average latency and ipg. When the ipg is high, it takes a longer time for the packet to be delivered to the destination.

Figure 4.6: Average Latency and Interpacket gap (ipg) in two LAG LACP setup

4.2.2 Latency for four LAG LACP

Table 4.6 shows the average latencies for simultaneous transmission of 1000M of TCP flood requests and acknowledgments for each connected host (h2, h3, h4, and h5) and the server (h1) for 100 seconds.

Table 4.6: A summary of ping flood results for each host for 100s in four LAG LACP setup.

Hosts	Min Latency (ms)	Avg Latency (ms)	Max Latency (ms)	Mdev (ms)	Ipg (ms)	Ewma (ms)
h2	0.007	0.551	13.162	2.292	0.237	0.01
h3	0.006	0.084	3.224	0.433	0.103	0.008
h4	0.01	0.134	6.774	0.814	0.172	0.024
h5	0.009	0.143	8.201	0.89	0.178	0.015

Table 4.6 shows that it took an average of 0.032ms to transmit a total of 400 icmp packets in a simultaneous TCP flood of 4000Mb of data transmitted. This indicates a faster end-to-end latency when the number of aggregated links was increased to four.

However, an increased interpacket gap (ipg) was observed for all transmitting hosts. This was because acknowledgment for some packets was duplicated. The impact of the ipg on latency is shown in figure 4.7.

Figure 4.7: Average Latency compared with Interpacket gap (ipg) for each host

Comparing the average latencies of the Two-LAG LACP setup and the Four-LAG LACP setup, it was realised that the 2-LAG LACP setup performed better in most cases by delivering packets in a relatively shorter time than the 4-LAG LACP setup, as shown in figure 4.8.

Figure 4.8: Comparing the average latencies of both 2-LAG LACP setup and 4-LAG LACP setup

4.3 Jitter

4.3.1 Jitter for Two LAG LACP

Jitter values obtained when the connected hosts h2, h3, h4, and h5 transmit 1GB iPerft3 UDP stream datagrams to the server (h1) for 15 seconds as shown in table 4.7 and figure 4.9.

Table 4.7: Jitter values for 15s of iPerf3 UDP

Interval (sec)	h2_Jitter (ms)	h3_Jitter (ms)	h4_Jitter (ms)	h5_Jitter (ms)
1	0.029	0.00	0.035	0.006
2	0.00	0.017	0.006	0.018
3	0.001	0.012	0.018	0.027
4	0.00	0.007	0.011	0.00

5	0.007	0.005	0.001	0.006
6	0.005	0.006	0.009	0.011
7	0.009	0.007	0.01	0.005
8	0.001	0.013	0.011	0.009
9	0.007	0.007	0.032	0.00
10	0.014	0.029	0.006	0.003
11	0.008	0.006	0.008	0.008
12	0.022	0.008	0.033	0.018
13	0.001	0.026	0.001	0.01
14	0.00	0.008	0.019	0.008
15	0.002	0.033	0.004	0.385
TOTAL	**0.106**	**0.184**	**0.204**	**0.514**

Figure 4.9: Graph showing jitter for all hosts after 15s of iPerf3 UDP stream

Also, iPerf3 TCP streams recorded a 0ms jitter for all connected hosts. However, the simulation report shows a congestion window (Cwnd) for all connected hosts as shown in table 4.8.

Table 4.8: Congestion window for iPerf3 TCP flood for all hosts

Interval (sec)	h2_Cwnd (KB)	h3_Cwnd (KB)	h4_Cwnd (KB)	h5_Cwnd (KB)
1	290	270	276	263
2	320	270	276	263
3	320	270	293	263
4	320	288	293	263
5	320	288	293	263
6	320	288	293	263
7	320	288	293	263
8	502	288	293	263
9	502	288	293	263
10	502	288	293	263
11	502	288	293	263
12	502	288	293	263
13	502	303	293	263
14	502	318	293	263
15	502	318	293	263
TOTAL	**6226**	**4341**	**4361**	**3945**

These congestion window values were because of the delays in processing the server's (h1) responses to the UDP and TCP flood transmitted simultaneously from all four connected hosts. Thus, it was observed that communication from the clients to the server did not experience congestion but a maximum of 6226KB congestion was created as the server responds to the requests it received from the hosts. These Cwnd values were not large enough so the was 0ms of jitter experienced throughout the communication.

It was also observed that varying sizes of Cwnd were recorded for each transmission. This was due to the systems' attempt to calculate the volume of the incoming load and dynamically assign bandwidth accordingly as shown in figure 4.10.

Figure 4.10: Graph of congestion window for iPerf3 TCP flood.

4.3.1 Jitter for Four LAG LACP

Jitter values obtained when the connected hosts h2, h3, h4, and h5 transmit 1GB iPerft3 UDP datagram stream to the server (h1) for 15 seconds as shown in table 4.9 and figure 4.11

Table 4.9: Jitter values for 15s of iPerf3 UDP stream for four LAG LACP

Interval (sec)	Jitter_h2 (mbps)	Jitter_h3 (mbps)	Jitter_h4 (mbps)	Jitter_h5 (mbps)
1	0.018	0.002	0.10	0.004
2	0.006	0.011	0.153	0.049
3	0.021	0.012	0.011	0.887
4	17.134	0.013	0.04	0.021
5	0.001	0.061	0.063	0.007
6	0.00	55.873	0.022	0.00

7	0.001	63.78	0.02	0.013
8	0.00	0.001	0.053	0.005
9	0.028	0.00	0.034	0.012
10	0.001	0.001	0.832	0.015
11	0.002	0.00	0.042	0.001
12	0.004	0.007	0.064	0.001
13	0.084	0.008	0.022	0.001
14	0.014	0.021	0.015	0.00
15	0.012	43.062	0.041	0.00
TOTAL	**17.326**	**162.852**	**1.512**	**1.016**

Figure 4.11: Graph of jitter for all hosts after 15s of iPerf3 UDP stream

Congestion window (Cwnd) is one of the influencing factors for network performance. When the Cwnd is low, fewer packets will require retransmission which eventually improves jitter and packet loss rate. It was observed that an average of 100 packets were retransmitted for the four hosts. This resulted from the high Cwnd observed for some of the TCP streams as shown in table 4.10 and figure 4.12.

Table 4.10: Congestion window for four LAG LACP iPerf3 TCP flood.

Interval (Sec)	h2_Cwnd (KB)	h3_Cwnd (KB)	h4_Cwnd (KB)	h5_Cwnd (KB)
1	130	180	335	284
2	337	189	335	298
3	337	197	335	298
4	337	1860	2050	298
5	772	1860	2260	298
6	1007	1400	2260	314
7	1007	1400	2260	314
8	1030	1400	2220	314
9	1060	1400	2220	314
10	824	1400	2220	329
11	861	1400	2220	329
12	861	1400	1.41	329
13	861	1400	41	242
14	861	970	2220	242
15	919	977	1660	242

Figure 4.12: Graph of congestion window for four LAG LACP iPerf3 TCP flood

Comparing the jitter for the Two-LAG LACP setup and the Four-LAG LACP setup, it became evident that the was little variation in the jitter performance of the two experimental setups as shown in figure 4.13.

Figure 4.13: Comparing jitter for 2-LAG LACP setup and 4-LAG LACP setup

4.4 Packet Loss

4.4.1 Packets lost in Two LAG LACP

Table 4.11 shows the packets transmitted and the loss rate when 15s of UDP flood datagrams were transmitted from the connected hosts (h2, h3, h4, and h5) to the sever (h1).

Table 4.11: Packets loss rate (%) for two LAG LACP

Hosts	Packets transmitted	Packets lost	Packets Lost Rate (%)
h2	1294925	8298	0.64
h3	1294440	10981	0.85
h4	1294811	6940	0.54
h5	1293079	13352	1.00

4.4.2 Packets lost in Four LAG LACP

Table 4.12 shows the packets transmitted and the loss rate when 15s of UDP flood datagrams were transmitted from the connected hosts (h2, h3, h4, and h5) to the sever (h1).

Table 4.12: Packets loss rate (%) for four LAG LACP

Hosts	Packets transmitted	Packets lost	Packets Lost Rate (%)
h2	1876441	211475	11.27
h3	1858448	211225	11.37
h4	1344379	168925	12.57
h5	1797683	82753	4.6

Comparing the number of packets lost in both the Two-LAG LACP setup and the Four-LAG LACP setup, it was realised that more packets were lost in the Four-LAG LACP setup than in the Two-LAG LACP setup. This is shown in figure 4.14.

Figure 4.14: Comparing the number of packets lost in both Two-LAG LACP setup and Four-LAG LACP setup

4.5 Fault Tolerance

To test for fault tolerance, the Linux code (ip link set h1-eth0 nomaster) was run on the server (h1). This deletes the logical link (bond0), leaving only the physical link that connects h1-eth1 available for data transmission since the physical link that connects h1-eth0 was deleted after setting up the logical link (bond0). In the two LAG LACP experiment, when the h1-eth0 link was deleted, the setup had h1-eth1 as the link available for data communication. The system was tested by transmitting 1000M UDP flood from the hosts (h2, h3, h4, and h5) to the server (h1). Table 4.13 and figure 4.15 show a summary report of the systems' response for bandwidth, jitter, and packet loss rate.

Table 4.13: Fault Tolerance in two LAG LACP

Hosts	Bandwidth	Jitter	Packets Loss Rate
	(Mb/s)	(ms)	(%)
h2	549	0.001	9.7
h3	555	8.027	9.5
h4	603	0.005	17
h5	647	0.001	6

Figure 4.15: Fault Tolerance in two LAG LACP

Similarly, running ip link set h1-eth0 nomaster on the server deletes one link (h1-eth0). This means that three more links (h1-eth1, h1-eth2, and h1-eth3) will be available for transmission. Shows minimum impact on the bandwidth and jitter values during the system testing. Therefore, the code (ip link set h1-eth1 nomaster) was run again to delete h1-eth1, leaving h1-eth2 and h1-eth3 links up for data transmission. The system was tested again by transmitting 1000M UDP flood from the hosts (h2, h3, h4, and h5) to the server (h1). Table 4.14 and figure 4.16 show a summary report of the systems' response for bandwidth, jitter, and packet loss rate.

Table 4.14: Fault Tolerance in four LAG LACP

Hosts	Bandwidth (Mb/s)	Jitter (ms)	Packets Loss Rate (%)
h2	61.8	0.184	82
h3	372	0.717	7.4
h4	242	0.12	33
h5	500	0.001	0.074

Figure 4.16: Fault Tolerance in four LAG LACP

From figure 4.11 and figure 4.12 two LAG LACP is more tolerant of bandwidth assurance than four LAG LACP. All hosts in figure 4.11 had maintained bandwidth assurance in transmitting UDP datagrams. However, figure 4.12 shows that bandwidth depletion can occur per increase in UDP flood datagrams.

Comparing the Two-LAG LACP setup to the Four-LAG LACP setup on their ability to reserve bandwidth during fault tolerance, it was realised that the Two-LAG LACP setup was able to reserve higher bandwidth than the Four-LAG LACP setup. Making the Two-LAG LACP setup to be more fault-tolerant as shown in figure 4.17.

Figure 4.17: Bandwidth reservation at fault tolerance for 2-LAG LACP setup and 4-LAG LACP setup

CHAPTER FIVE

5.0 SUMMARY OF FINDINGS

From the results in chapter four, it was realised that when the setup dynamically reserves high bandwidth in response to TCP and UDP flood transmission, its performance on the other critical QoS parameters was also good. For example, the Two-LAG LACP setup showed a high bandwidth reservation on both TCP and UDP flood. This influenced the shorter time taken to deliver packets, the near insignificant jitter values, and the smaller number of packets lost or out-of-order packets during testing. More so, the Two-LAG LACP setup showed an excellent fault-tolerant which resulted in less packet loss rates as compared with the Four-LAG LACP setup.

The Four-LAG LACP setup on the other hand was poor on bandwidth reservation. This resulted in a huge number of out of order packets transmission and their attendant high packet loss rate.

5.1 CONCLUSION

Although hosts of software systems to improve the bandwidth of campus networks and user experiences on educational systems abound, there has not been a significant improvement in bandwidth and quality of service on the LMS in many Ghanaian schools, especially for online learning. By simulating a dynamic bandwidth-aware LACP system for 2-LAG and 4-LAG, it was evident that LACP is cheaper but effective in increasing link bandwidth and fault tolerance when the aggregated links are not many. For example, 2-LAG LACP was excellent in improving bandwidth and QoS. However, results obtained from the 4-LAG setup show that as the number of aggregated links increases, delays and packets retransmission rate also increases. This suggests that the number of links that should form a LAG should be carefully selected according to the SDN controllers' ability to fully utilise all aggregated links to avoid link redundancy.

5.2 IMPLICATIONS AND RECOMMENDATIONS

While two links aggregation worked perfectly in improving network performance on dynamic bandwidth reservation and other QoS parameters, increasing the number of aggregated links resulted in a lot of uncertainties in the network's ability to process packets. In the LACP simulation, this accounted for the low performance of the Four-LAG LACP setup in the experiment in chapter three.

Therefore, it is important to limit the number of aggregated links and implement a dynamic and application-aware load balancing to maintain bandwidth and QoS for peak-time transmissions. Also, implementing a two-LAG LACP for campus LMS systems stands to reduce the cost of procuring complex networking devices and at the same time improve the bandwidth and QoS performance of the campus network.

5.3 SUGGESTIONS FOR FURTHER STUDIES

Further research will be focused on implementing a north-bound load balancing API for the Ryu controller to improve the bandwidth reservation and QoS performance of 4 and 8 LAG LACP for distributed application systems.

REFERENCES

Abdul, M., Alshehri, R., Mishra, S., Abdul, M., Alshehri, R., & Mishra, S. (2019). Feature Based Comparison and Selection of SDN Controller. *International Journal of Innovation and Technology Management*, *16*(5). https://doi.org/10.1142/S0219877019500299

Abdullah, M. Z., Al-awad, N. A., & Hussein, F. W. (2019). *Evaluating and Comparing the Performance of Using Multiple Controllers in Software Defined Networks*. *August*. https://doi.org/10.5815/ijmecs.2019.08.03

Adarkwah, M. A. (2020). "I'm not against online teaching, but what about us?": ICT in Ghana post Covid-19. *Education and Information Technologies*, *2*. https://doi.org/10.1007/s10639-020-10331-z

Adato, L. (2017). Monitoring and automation: it's easier than you think. *Network Security*, *2017*(4), 5–7. https://doi.org/10.1016/S1353-4858(17)30036-3

Aguado, M., Astorga, J., Toledo, N., & Matias, J. (2011). Simulation-based methods for network design. *International Series in Operations Research and Management Science*, *158*, 271–293. https://doi.org/10.1007/978-1-4419-6111-2_12

Al-Shehri, S. M., Loskot, P., Numanoglu, T., & Mert, M. (2017). Common Metrics for Analyzing, Developing and Managing Telecommunication Networks. *Networking and Internet Architecture*. http://arxiv.org/abs/1707.03290

Aldiab, A., Chowdhury, H., Kootsookos, A., Alam, F., & Allhibi, H. (2019). Utilization of Learning Management Systems (LMSs) in higher education system: A case review for Saudi Arabia. *Energy Procedia*, *160*(2018), 731–737. https://doi.org/10.1016/j.egypro.2019.02.186

Aziz, M., Fazelyh, A., Landi, G., Gallico, D., Christodoulopoulos, K., & Wieder, P. (2016). SDN-enabled Application-aware networking for Data center networks. *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. https://doi.org/10.1109/ICECS.2016.7841210

Baskoro, F., Hidayat, R., & Wibowo, S. B. (2019). Comparing LACP Implementation between Ryu and Opendaylight SDN Controller. *2019 11th International Conference on Information Technology and Electrical Engineering, ICITEE 2019*, *7*, 1–4. https://doi.org/10.1109/ICITEED.2019.8929986

Baskoro, F., Risanuri, H., & Wibowo, S. B. (2019). *LACP Experiment using Multiple Flow Table in Ryu SDN Controller*. 51–55.

Boero, L., Cello, M., Garibotto, C., Marchese, M., & Mongelli, M. (2016a). BeaQoS : Load balancing and deadline management of queues in an OpenFlow SDN switch. *Computer Networks*, *106*, 161–170. https://doi.org/10.1016/j.comnet.2016.06.025

Boero, L., Cello, M., Garibotto, C., Marchese, M., & Mongelli, M. (2016b). BeaQoS: Load balancing and deadline management of queues in an OpenFlow SDN switch. *Computer Networks*, *106*, 161–170. https://doi.org/10.1016/j.comnet.2016.06.025

Campbell, S., & Jeronimo, M. (2006). *Applied Virtualization Technology*.

Chien, T., & Kao, F. (2005). *The Design of Load-Balancing LMS Based on Decomposition Structure*. 1–5. https://doi.org/10.1109/ICALT.2005.264

CISCO. (2019). *Intent-Based Networking Bridge the gap between business and IT*. Cisco Intent Based Networking.

Denneman, F., Epping, D., & Hagoort, N. (2018). *VMware vSphere 6 . 7 Clustering Deep Dive*.

Du, P., Pang, F., Braun, T., Gerla, M., Hoffmann, C., & Kim, J. H. (2017). Traffic optimization in software defined naval network for satellite communications. *Proceedings - IEEE Military Communications Conference MILCOM*, *2017-October*, 459–464. https://doi.org/10.1109/MILCOM.2017.8170766

Feldmann, A., Zitterbart, M., Crowcroft, J., & Wetherall, D. (2003). Proceedings of the 2003 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. *Computer Communication Review*, *26*(4).

Ferguson, A. D., Guha, A., Liang, C., Fonseca, R., & Krishnamurthi, S. (2013). Participatory networking. *ACM SIGCOMM Computer Communication Review*, *43*(4), 327–338. https://doi.org/10.1145/2534169.2486003

Fernandez, J. C., Taleb, T., Guizani, M., & Kato, N. (2009). Bandwidth aggregation-aware dynamic QoS negotiation for real-time video streaming in next-generation wireless networks. *IEEE Transactions on Multimedia*, *11*(6), 1082–1093. https://doi.org/10.1109/TMM.2009.2026086

Fisher, T., & Heine, M. B. (2022). *What is bandwidth? Definition, meaning, and details*. LifewiRe. https://www.lifewire.com/what-is-bandwidth-2625809

Freer, J. R. (1988). Layered network architectures. In *Computer Communications and Networks* (pp. 133–170). Springer US. https://doi.org/10.1007/978-1-4613-1041-9_5

Gatos, L., Group, G. I., & Lin, P. E. W. (2005). *Methods, apparatuses and systems enabling a network services provider to deliver application performance management services*. *2*(12).

Gillis, A. S. (2020). *Latency*. TechTarget. https://whatis.techtarget.com/definition/latency

Gorlatch, S., Humernbrum, T., & Glinka, F. (2014). Improving QoS in real-time internet applications: from best-effort to Software-Defined Networks. *2014 International Conference on Computing, Networking and Communications (ICNC)*, 189–193. https://doi.org/10.1109/ICCNC.2014.6785329

Guo, Z., Su, M., Xu, Y., Duan, Z., Wang, L., Hui, S., & Chao, H. J. (2014). Improving the performance of load balancing in software-defined networks through load variance-based synchronization. *Computer Networks*, *68*, 95–109. https://doi.org/10.1016/j.comnet.2013.12.004

Heller, B., Wundsam, A., Handigol, N., Scott, C., Zeng, H., McCauley, J., McKeownt, N., Whitlock, S., Zarifis, K., Shenker, S., Jeyakumar, V., & Kazemian, P. (2013). Leveraging SDN layering to systematically troubleshoot networks. *HotSDN 2013 -*

Proceedings of the 2013 ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, 37–42. https://doi.org/10.1145/2491185.2491197

Hu, Y., Luo, T., Beaulieu, N., & Wang, W. (2017). An Initial Load-Based Green Software Defined Network. *Applied Sciences*, *7*(5), 459. https://doi.org/10.3390/app7050459

IEEE, S. B. (2020). IEEE Standard for Local and Metropolitan Area Networks - Link Aggregation. *IEEE Xplore*, *2020*.

Irawati, I. D., Hariyani, Y. S., & Hadiyoso, S. (2017). Link aggregation control protocol on software defined network. *International Journal of Electrical and Computer Engineering*, *7*(5), 2706–2712. https://doi.org/10.11591/ijece.v7i5.pp2706-2712

Jahromi, H. Z., Hines, A., & Delaney, D. T. (2018). *Towards Application-Aware Networking : ML-based End-to-End Application KPI / QoE Metrics Characterization in SDN*. 126–131.

Jain, R., & Paul, S. (2013). Network virtualization and software defined networking for cloud computing: A survey. *IEEE Communications Magazine*, *51*(11), 24–31. https://doi.org/10.1109/MCOM.2013.6658648

Jarschel, M., Wamser, F., Thomas, H., Zinner, T., & Tran-gia, P. (2013). *SDN-based Application-Aware Networking on the Example of YouTube Video Streaming*. https://doi.org/10.1109/EWSDN.2013.21

Khan, A. Z., Baqai, S., & Dogar, F. R. (2012). QoS aware path selection in content centric networks. *IEEE International Conference on Communications*, 2645–2649. https://doi.org/10.1109/ICC.2012.6363829

Khan, A. Z., Baqai, S., & Member, S. (2010). A Case for Reducing Link Stress in a Multimedia Streaming Service Backbone. *The IEEE Symposium on Computers and Communications*, 945–947.

Khan, R., & Ali, S. (2013). Conceptual Framework of Redundant Link Aggregation. *Computer Science & Engineering: An International Journal*, *3*(2), 23–31.

https://doi.org/10.5121/cseij.2013.3202

Khondoker, R., Zaalouk, A., Marx, R., & Bayarou, K. (2014). *Feature-based Comparison and Selection of Software Defined Networking (SDN) Controllers*.

Kim, H., Voellmy, A., Burnett, S., Feamster, N., & Clark, R. (2012). *Lithium: Event-driven network control*. http://hdl.handle.net/1853/43377

Kranz, G. (2021). *Fault-Tolerance*. TechTarget. https://www.techtarget.com/searchdisasterrecovery/definition/fault-tolerant?_gl=1*x4787d*_ga*MTIyNDYxODUwMC4xNjQ3NjkxMzgz*_ga_TQKE4GS5P9*MTY0NzY5NjExMC4yLjEuMTY0NzY5Njg1OC4w&_ga=2.244825605.1852185675.1647691383-1224618500.1647691383

Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, *103*(1), 14–76. https://doi.org/10.1109/JPROC.2014.2371999

Latif, Z., & Sharif, K. (2020). Software Defined Networks. *Journal of Network and Computer Applications*, *Volume 156*(102563), 2020. https://doi.org/10.1016/j.jnca.2020.102563

Lee, J., Turner, Y., Lee, M., Popa, L., Banerjee, S., Kang, J.-M., & Sharma, P. (2014). Application-driven bandwidth guarantees in datacenters. *Proceedings of the 2014 ACM Conference on SIGCOMM*, 467–478. https://doi.org/10.1145/2619239.2626326

Liang, G., Li, W., Cui, Q., & Liu, G. (2021). An algorithm to build schedule table for schedule-based fieldbus to reduce communication jitter to its minimum. *ISA Transactions*. https://doi.org/10.1016/j.isatra.2021.08.022

Long, Y., Peilin, H., Wen, Z., Jianfei, L., & Dan, N. (2015). Controller Placement and Flow based Dynamic Management Problem towards SDN. *IEEE ICC 2015 - Workshop on Advances in Software Defined and Context Aware Cognitive Networks 2015*, *2015*, 363–368. https://doi.org/10.1109/ICCW.2015.7247206

Ma, Y.-W., Chen, J.-L., Chang, C.-C., Chiang, C.-M., Xie, Y.-L., & Chen, S.-J. (2015). SDN test cases development and implementation. *17th International Conference on Advanced Communication Technology (ICACT)*, 618–621. https://doi.org/10.1109/ICACT.2015.7224871

Maneewongvatana, S., & Maneewongvatana, S. (2013). Hybrid cloud load prediction model for LMS applications based on class activity patterns. *2013 International Joint Conference on Awareness Science and Technology and Ubi-Media Computing: Can We Realize Awareness via Ubi-Media?, ICAST 2013 and UMEDIA 2013*, 292–297. https://doi.org/10.1109/ICAwST.2013.6765450

Maria, A. (1997). Introduction to modeling and simulation. *Proceedings of the 29th Conference on Winter Simulation - WSC '97*, 7–13. https://doi.org/10.1145/268437.268440

Mishra, S., Abdul, M., & Alshehri, R. (2017). *Software Defined Networking : Research Issues , Challenges and Opportunities*. *10*(August). https://doi.org/10.17485/ijst/2017/v10i29/112447

Mohammad, S. M. S., Alam, B., & Nazrul, M. (2018). Data Communication Speed and Network Fault Tolerant Enhancement over Software Defined Networking. *Wireless Personal Communications*, *5*. https://doi.org/10.1007/s11277-018-5759-5

Molina, E., & Jacob, E. (2018). Software-defined networking in cyber-physical systems: A survey. *Computers and Electrical Engineering*, *66*, 407–419. https://doi.org/10.1016/j.compeleceng.2017.05.013

Mtawa, Y. Al, Haque, A., & Lutfiyya, H. (2021). Migrating From Legacy to Software Defined Networks : A Network Reliability Perspective. *IEEE Transactions on Reliability*, 1–17. https://doi.org/10.1109/TR.2021.3066526

Nadig, D., Ramamurthy, B., Bockelman, B., & Swanson, D. (2019). APRIL : An Application-Aware , Predictive and Intelligent Load Balancing Solution for Data-Intensive Science. *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 1909–1917.

Neghabi, A. A., Navimipour, N. J., Hosseinzadeh, M., & Rezaee, A. (2018). Load balancing mechanisms in the software defined networks : a systematic and comprehensive review of the literature. *IEEE Access*, *3536*(6), 14159–14178. https://doi.org/10.1109/ACCESS.2018.2805842

Norman, T. (2017). Information Technology Systems Infrastructure. In *Effective Physical Security: Fifth Edition* (pp. 311–341). Elsevier. https://doi.org/10.1016/B978-0-12-804462-9.00018-X

Nunes, B. A. A., Mendonca, M., Nguyen, X.-N., Obraczka, K., & Turletti, T. (2014). A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Communications Surveys & Tutorials*, *16*(3), 1617–1634. https://doi.org/10.1109/SURV.2014.012214.00180

Nurhadi, A. I., Petrus, G. B. K., Firdaus, M., & Muhammad, R. (2020). A Review of Link Aggregation Control Protocol (LACP) as a Link Redundancy in SDN Based Network Using Ryu-Controller. *ArXiv*, 1–7. https://doi.org/10.13140/RG.2.2.10801.20329

ONF. (2015). Framework for SDN: Scope and Requirements. In *Open Networking Foundation: Vol. ONF TR-516* (Issue June).

Ookla. (2021). *Global Speeds June 2021*. Speedtest Global Index. https://www.speedtest.net/global-index

Pahlevan, M. (2020). *Time Sensitive Networking for Virtualized Integrated Real-Time Systems* [University of Siegen]. https://doi.org/http://dx.doi.org/10.25819/ubsi/959

Patel, C., Gadhavi, M., & Patel, A. (2014). A survey paper on e-learning based learning management Systems (LMS). *ResearchGates*, *June*.

Qazi, Z., Lee, J., Jin, T., Bellala, G., Arndt, M., & Noubir, G. (2013). Application-Awareness in SDN. *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, 487–488. https://doi.org/10.1145/2486001.2491700

Rajguru, A., & Apte, S. (2012). A Comparative Performance Analysis of Load

Balancing Algorithms in Distributed System using Qualitative Parameters. *International Journal of Recent Technology and …*, *3*, 175–179. http://www.ijrte.org/attachments/File/v1i3/C0278071312.pdf

Rangisetti, A. K., Pasca S., T. V., & Tamma, B. R. (2017). QoS Aware load balance in software defined LTE networks. *Computer Communications*, *97*, 52–71. https://doi.org/10.1016/j.comcom.2016.09.005

Rowshanrad, S., Abdi, V., & Keshtgari, M. (2016). Performance evaluation of sdn controllers: Floodlight and OpenDaylight. *IIUM Engineering Journal*, *17*(2), 47–57. https://doi.org/10.31436/iiumej.v17i2.615

Ryu project team. (2014). *Ryu SDN Framework* (Issue 1.0). https://books.google.com.br/books?id=JC3rAgAAQBAJ&hl=pt-BR

Sadiq, K. A., Ayeni, J. K., & Oyedepo, F. S. (2020). An Optimized Kwara State Polytechnic Campus Networks using VLAN. *International Journal of Computer Applications*, *175*(17), 1–3. https://doi.org/10.5120/ijca2020920671

Salman, O., Elhajj, I. H., Kayssi, A., & Chehab, A. (2016). SDN controllers: A comparative study. *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, 1–6. https://doi.org/10.1109/MELCON.2016.7495430

Sezer, S., Scott-Hayward, S., Chouhan, P., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., & Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, *51*(7), 36–43. https://doi.org/10.1109/MCOM.2013.6553676

Shamseldin, R. A., & Bowling, S. R. (2009). Statistical network optimisation of dynamically complex systems. *International Journal of Experimental Design and Process Optimisation*, *1*(1), 79. https://doi.org/10.1504/ijedpo.2009.028958

Soulé, R., Basu, S., Marandi, P. J., Pedone, F., Kleinberg, R., Sirer, E. G., & Foster, N. (2014). Merlin: A Language for Provisioning Network Resources. *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, 213–226. https://doi.org/10.1145/2674005.2674989

Takiguchi, T., Hidaka, A., Masui, H., Sugizaki, Y., Mizuno, O., & Asatani, K. (2012). A new application-level link aggregation and its implementation on Android terminals. *Wireless Communications and Mobile Computing*, *12*(18), 1664–1671. https://doi.org/10.1002/wcm.2329

Tootoonchian, A., Casado, M., & Sherwood, R. (n.d.). *On Controller Performance in Software-Defined Networks*.

Urera, F. L. J., & Balahadia, F. F. (2019). ICTeachMUPO : An Evaluation of Information E-Learning Module System for Faculty and Students. *International Journal of Computing Sciences Research*, *3*(1), 163–188. https://doi.org/10.25147/ijcsr.2017.001.1.31

Warraich, S. H., Aziz, Z., Khurshid, H., Hameed, R., Saboor, A., & Awais, M. (2020). SDN enabled and OpenFlow compatible network performance monitoring system. *ArXiv*, 1–10.

Xu, J., Wang, J., Qi, Q., Sun, H., & He, B. (2018). DEEP NEURAL NETWORKS FOR APPLICATION AWARENESS IN SDN-BASED NETWORK Jun Xu , Jingyu Wang , Qi Qi , Haifeng Sun , Bo He State Key Laboratory of Networking and Switching Technology Beijing University of Posts and Telecommunications , Beijing , China. *IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING*, *61771068*, 0–5.

Yamei, F., Qing, L., & Qi, H. (2016). Research and comparative analysis of performance test on SDN controller. *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, 207–210. https://doi.org/10.1109/CCI.2016.7778909

Yen, T. C., & Su, C. S. (2014). An SDN-based cloud computing architecture and its mathematical model. *Proceedings - 2014 International Conference on Information Science, Electronics and Electrical Engineering, ISEEE 2014*, *3*, 1728–1731. https://doi.org/10.1109/InfoSEEE.2014.6946218

Zhang, Y., Cui, L., Wang, W., & Zhang, Y. (2017). Author ' s Accepted Manuscript A Survey on Software Defined Networking with Multiple Controllers. *Journal of*

Network and Computer Applications. https://doi.org/10.1016/j.jnca.2017.11.015

Zhou, Y., Ruan, L., Xiao, L., & Liu, R. (2014). A Method for Load Balancing based on Software- Defined Network. *Semantic Scholar*. https://doi.org/10.14257/ASTL.2014.45.09

Abdul, M., Alshehri, R., Mishra, S., Abdul, M., Alshehri, R., & Mishra, S. (2019). Feature Based Comparison and Selection of SDN Controller. *International Journal of Innovation and Technology Management*, *16*(5). https://doi.org/10.1142/S0219877019500299

Abdullah, M. Z., Al-awad, N. A., & Hussein, F. W. (2019). *Evaluating and Comparing the Performance of Using Multiple Controllers in Software Defined Networks*. *August*. https://doi.org/10.5815/ijmecs.2019.08.03

Adarkwah, M. A. (2020). “I’m not against online teaching, but what about us?”: ICT in Ghana post Covid-19. *Education and Information Technologies*, *2*. https://doi.org/10.1007/s10639-020-10331-z

Adato, L. (2017). Monitoring and automation: it’s easier than you think. *Network Security*, *2017*(4), 5–7. https://doi.org/10.1016/S1353-4858(17)30036-3

Aguado, M., Astorga, J., Toledo, N., & Matias, J. (2011). Simulation-based methods for network design. *International Series in Operations Research and Management Science*, *158*, 271–293. https://doi.org/10.1007/978-1-4419-6111-2_12

Al-Shehri, S. M., Loskot, P., Numanoglu, T., & Mert, M. (2017). Common Metrics for Analyzing, Developing and Managing Telecommunication Networks. *Networking and Internet Architecture*. http://arxiv.org/abs/1707.03290

Aldiab, A., Chowdhury, H., Kootsookos, A., Alam, F., & Allhibi, H. (2019). Utilization of Learning Management Systems (LMSs) in higher education system: A case review for Saudi Arabia. *Energy Procedia*, *160*(2018), 731–737. https://doi.org/10.1016/j.egypro.2019.02.186

Aziz, M., Fazelyh, A., Landi, G., Gallico, D., Christodoulopoulos, K., & Wieder, P.

(2016). SDN-enabled Application-aware networking for Data center networks. *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. https://doi.org/10.1109/ICECS.2016.7841210

Baskoro, F., Hidayat, R., & Wibowo, S. B. (2019). Comparing LACP Implementation between Ryu and Opendaylight SDN Controller. *2019 11th International Conference on Information Technology and Electrical Engineering, ICITEE 2019*, *7*, 1–4. https://doi.org/10.1109/ICITEED.2019.8929986

Baskoro, F., Risanuri, H., & Wibowo, S. B. (2019). *LACP Experiment using Multiple Flow Table in Ryu SDN Controller*. 51–55.

Boero, L., Cello, M., Garibotto, C., Marchese, M., & Mongelli, M. (2016a). BeaQoS : Load balancing and deadline management of queues in an OpenFlow SDN switch. *Computer Networks*, *106*, 161–170. https://doi.org/10.1016/j.comnet.2016.06.025

Boero, L., Cello, M., Garibotto, C., Marchese, M., & Mongelli, M. (2016b). BeaQoS: Load balancing and deadline management of queues in an OpenFlow SDN switch. *Computer Networks*, *106*, 161–170. https://doi.org/10.1016/j.comnet.2016.06.025

Campbell, S., & Jeronimo, M. (2006). *Applied Virtualization Technology*.

Chien, T., & Kao, F. (2005). *The Design of Load-Balancing LMS Based on Decomposition Structure*. 1–5. https://doi.org/10.1109/ICALT.2005.264

CISCO. (2019). *Intent-Based Networking Bridge the gap between business and IT*. Cisco Intent Based Networking.

Denneman, F., Epping, D., & Hagoort, N. (2018). *VMware vSphere 6 . 7 Clustering Deep Dive*.

Du, P., Pang, F., Braun, T., Gerla, M., Hoffmann, C., & Kim, J. H. (2017). Traffic optimization in software defined naval network for satellite communications. *Proceedings - IEEE Military Communications Conference MILCOM*, *2017-October*, 459–464. https://doi.org/10.1109/MILCOM.2017.8170766

Feldmann, A., Zitterbart, M., Crowcroft, J., & Wetherall, D. (2003). Proceedings of the

2003 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. *Computer Communication Review*, *26*(4).

Ferguson, A. D., Guha, A., Liang, C., Fonseca, R., & Krishnamurthi, S. (2013). Participatory networking. *ACM SIGCOMM Computer Communication Review*, *43*(4), 327–338. https://doi.org/10.1145/2534169.2486003

Fernandez, J. C., Taleb, T., Guizani, M., & Kato, N. (2009). Bandwidth aggregation-aware dynamic QoS negotiation for real-time video streaming in next-generation wireless networks. *IEEE Transactions on Multimedia*, *11*(6), 1082–1093. https://doi.org/10.1109/TMM.2009.2026086

Fisher, T., & Heine, M. B. (2022). *What is bandwidth? Definition, meaning, and details*. LifewiRe. https://www.lifewire.com/what-is-bandwidth-2625809

Freer, J. R. (1988). Layered network architectures. In *Computer Communications and Networks* (pp. 133–170). Springer US. https://doi.org/10.1007/978-1-4613-1041-9_5

Gatos, L., Group, G. I., & Lin, P. E. W. (2005). *Methods, apparatuses and systems enabling a network services provider to deliver application performance management services*. *2*(12).

Gillis, A. S. (2020). *Latency*. TechTarget. https://whatis.techtarget.com/definition/latency

Gorlatch, S., Humernbrum, T., & Glinka, F. (2014). Improving QoS in real-time internet applications: from best-effort to Software-Defined Networks. *2014 International Conference on Computing, Networking and Communications (ICNC)*, 189–193. https://doi.org/10.1109/ICCNC.2014.6785329

Guo, Z., Su, M., Xu, Y., Duan, Z., Wang, L., Hui, S., & Chao, H. J. (2014). Improving the performance of load balancing in software-defined networks through load variance-based synchronization. *Computer Networks*, *68*, 95–109. https://doi.org/10.1016/j.comnet.2013.12.004

Heller, B., Wundsam, A., Handigol, N., Scott, C., Zeng, H., McCauley, J., McKeownt, N., Whitlock, S., Zarifis, K., Shenker, S., Jeyakumar, V., & Kazemian, P. (2013). Leveraging SDN layering to systematically troubleshoot networks. *HotSDN 2013 - Proceedings of the 2013 ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, 37–42. https://doi.org/10.1145/2491185.2491197

Hu, Y., Luo, T., Beaulieu, N., & Wang, W. (2017). An Initial Load-Based Green Software Defined Network. *Applied Sciences*, *7*(5), 459. https://doi.org/10.3390/app7050459

IEEE, S. B. (2020). IEEE Standard for Local and Metropolitan Area Networks - Link Aggregation. *IEEE Xplore*, *2020*.

Irawati, I. D., Hariyani, Y. S., & Hadiyoso, S. (2017). Link aggregation control protocol on software defined network. *International Journal of Electrical and Computer Engineering*, *7*(5), 2706–2712. https://doi.org/10.11591/ijece.v7i5.pp2706-2712

Jahromi, H. Z., Hines, A., & Delaney, D. T. (2018). *Towards Application-Aware Networking : ML-based End-to-End Application KPI / QoE Metrics Characterization in SDN*. 126–131.

Jain, R., & Paul, S. (2013). Network virtualization and software defined networking for cloud computing: A survey. *IEEE Communications Magazine*, *51*(11), 24–31. https://doi.org/10.1109/MCOM.2013.6658648

Jarschel, M., Wamser, F., Thomas, H., Zinner, T., & Tran-gia, P. (2013). *SDN-based Application-Aware Networking on the Example of YouTube Video Streaming*. https://doi.org/10.1109/EWSDN.2013.21

Khan, A. Z., Baqai, S., & Dogar, F. R. (2012). QoS aware path selection in content centric networks. *IEEE International Conference on Communications*, 2645–2649. https://doi.org/10.1109/ICC.2012.6363829

Khan, A. Z., Baqai, S., & Member, S. (2010). A Case for Reducing Link Stress in a Multimedia Streaming Service Backbone. *The IEEE Symposium on Computers and Communications*, 945–947.

Khan, R., & Ali, S. (2013). Conceptual Framework of Redundant Link Aggregation. *Computer Science & Engineering: An International Journal*, *3*(2), 23–31. https://doi.org/10.5121/cseij.2013.3202

Khondoker, R., Zaalouk, A., Marx, R., & Bayarou, K. (2014). *Feature-based Comparison and Selection of Software Defined Networking (SDN) Controllers*.

Kim, H., Voellmy, A., Burnett, S., Feamster, N., & Clark, R. (2012). *Lithium: Event-driven network control*. http://hdl.handle.net/1853/43377

Kranz, G. (2021). *Fault-Tolerance*. TechTarget. https://www.techtarget.com/searchdisasterrecovery/definition/fault-tolerant?_gl=1*x4787d*_ga*MTIyNDYxODUwMC4xNjQ3NjkxMzgz*_ga_TQKE4GS5P9*MTY0NzY5NjExMC4yLjEuMTY0NzY5Njg1OC4w&_ga=2.244825605.1852185675.1647691383-1224618500.1647691383

Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, *103*(1), 14–76. https://doi.org/10.1109/JPROC.2014.2371999

Latif, Z., & Sharif, K. (2020). Software Defined Networks. *Journal of Network and Computer Applications*, *Volume 156*(102563), 2020. https://doi.org/10.1016/j.jnca.2020.102563

Lee, J., Turner, Y., Lee, M., Popa, L., Banerjee, S., Kang, J.-M., & Sharma, P. (2014). Application-driven bandwidth guarantees in datacenters. *Proceedings of the 2014 ACM Conference on SIGCOMM*, 467–478. https://doi.org/10.1145/2619239.2626326

Liang, G., Li, W., Cui, Q., & Liu, G. (2021). An algorithm to build schedule table for schedule-based fieldbus to reduce communication jitter to its minimum. *ISA Transactions*. https://doi.org/10.1016/j.isatra.2021.08.022

Long, Y., Peilin, H., Wen, Z., Jianfei, L., & Dan, N. (2015). Controller Placement and Flow based Dynamic Management Problem towards SDN. *IEEE ICC 2015 -*

Workshop on Advances in Software Defined and Context Aware Cognitive Networks 2015, *2015*, 363–368. https://doi.org/10.1109/ICCW.2015.7247206

Ma, Y.-W., Chen, J.-L., Chang, C.-C., Chiang, C.-M., Xie, Y.-L., & Chen, S.-J. (2015). SDN test cases development and implementation. *17th International Conference on Advanced Communication Technology (ICACT)*, 618–621. https://doi.org/10.1109/ICACT.2015.7224871

Maneewongvatana, S., & Maneewongvatana, S. (2013). Hybrid cloud load prediction model for LMS applications based on class activity patterns. *2013 International Joint Conference on Awareness Science and Technology and Ubi-Media Computing: Can We Realize Awareness via Ubi-Media?, ICAST 2013 and UMEDIA 2013*, 292–297. https://doi.org/10.1109/ICAwST.2013.6765450

Maria, A. (1997). Introduction to modeling and simulation. *Proceedings of the 29th Conference on Winter Simulation - WSC '97*, 7–13. https://doi.org/10.1145/268437.268440

Mishra, S., Abdul, M., & Alshehri, R. (2017). *Software Defined Networking : Research Issues , Challenges and Opportunities*. *10*(August). https://doi.org/10.17485/ijst/2017/v10i29/112447

Mohammad, S. M. S., Alam, B., & Nazrul, M. (2018). Data Communication Speed and Network Fault Tolerant Enhancement over Software Defined Networking. *Wireless Personal Communications*, *5*. https://doi.org/10.1007/s11277-018-5759-5

Molina, E., & Jacob, E. (2018). Software-defined networking in cyber-physical systems: A survey. *Computers and Electrical Engineering*, *66*, 407–419. https://doi.org/10.1016/j.compeleceng.2017.05.013

Mtawa, Y. Al, Haque, A., & Lutfiyya, H. (2021). Migrating From Legacy to Software Defined Networks : A Network Reliability Perspective. *IEEE Transactions on Reliability*, 1–17. https://doi.org/10.1109/TR.2021.3066526

Nadig, D., Ramamurthy, B., Bockelman, B., & Swanson, D. (2019). APRIL : An Application-Aware , Predictive and Intelligent Load Balancing Solution for Data-

Intensive Science. *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 1909–1917.

Neghabi, A. A., Navimipour, N. J., Hosseinzadeh, M., & Rezaee, A. (2018). Load balancing mechanisms in the software defined networks : a systematic and comprehensive review of the literature. *IEEE Access*, *3536*(6), 14159–14178. https://doi.org/10.1109/ACCESS.2018.2805842

Norman, T. (2017). Information Technology Systems Infrastructure. In *Effective Physical Security: Fifth Edition* (pp. 311–341). Elsevier. https://doi.org/10.1016/B978-0-12-804462-9.00018-X

Nunes, B. A. A., Mendonca, M., Nguyen, X.-N., Obraczka, K., & Turletti, T. (2014). A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Communications Surveys & Tutorials*, *16*(3), 1617–1634. https://doi.org/10.1109/SURV.2014.012214.00180

Nurhadi, A. I., Petrus, G. B. K., Firdaus, M., & Muhammad, R. (2020). A Review of Link Aggregation Control Protocol (LACP) as a Link Redundancy in SDN Based Network Using Ryu-Controller. *ArXiv*, 1–7. https://doi.org/10.13140/RG.2.2.10801.20329

ONF. (2015). Framework for SDN: Scope and Requirements. In *Open Networking Foundation: Vol. ONF TR-516* (Issue June).

Ookla. (2021). *Global Speeds June 2021*. Speedtest Global Index. https://www.speedtest.net/global-index

Pahlevan, M. (2020). *Time Sensitive Networking for Virtualized Integrated Real-Time Systems* [University of Siegen]. https://doi.org/http://dx.doi.org/10.25819/ubsi/959

Patel, C., Gadhavi, M., & Patel, A. (2014). A survey paper on e-learning based learning management Systems (LMS). *ResearchGates*, *June*.

Qazi, Z., Lee, J., Jin, T., Bellala, G., Arndt, M., & Noubir, G. (2013). Application-Awareness in SDN. *Proceedings of the ACM SIGCOMM 2013 Conference on*

SIGCOMM, 487–488. https://doi.org/10.1145/2486001.2491700

Rajguru, A., & Apte, S. (2012). A Comparative Performance Analysis of Load Balancing Algorithms in Distributed System using Qualitative Parameters. *International Journal of Recent Technology and …*, *3*, 175–179. http://www.ijrte.org/attachments/File/v1i3/C0278071312.pdf

Rangisetti, A. K., Pasca S., T. V., & Tamma, B. R. (2017). QoS Aware load balance in software defined LTE networks. *Computer Communications*, *97*, 52–71. https://doi.org/10.1016/j.comcom.2016.09.005

Rowshanrad, S., Abdi, V., & Keshtgari, M. (2016). Performance evaluation of sdn controllers: Floodlight and OpenDaylight. *IIUM Engineering Journal*, *17*(2), 47–57. https://doi.org/10.31436/iiumej.v17i2.615

Ryu project team. (2014). *Ryu SDN Framework* (Issue 1.0). https://books.google.com.br/books?id=JC3rAgAAQBAJ&hl=pt-BR

Sadiq, K. A., Ayeni, J. K., & Oyedepo, F. S. (2020). An Optimized Kwara State Polytechnic Campus Networks using VLAN. *International Journal of Computer Applications*, *175*(17), 1–3. https://doi.org/10.5120/ijca2020920671

Salman, O., Elhajj, I. H., Kayssi, A., & Chehab, A. (2016). SDN controllers: A comparative study. *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, 1–6. https://doi.org/10.1109/MELCON.2016.7495430

Sezer, S., Scott-Hayward, S., Chouhan, P., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., & Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, *51*(7), 36–43. https://doi.org/10.1109/MCOM.2013.6553676

Shamseldin, R. A., & Bowling, S. R. (2009). Statistical network optimisation of dynamically complex systems. *International Journal of Experimental Design and Process Optimisation*, *1*(1), 79. https://doi.org/10.1504/ijedpo.2009.028958

Soulé, R., Basu, S., Marandi, P. J., Pedone, F., Kleinberg, R., Sirer, E. G., & Foster, N.

(2014). Merlin: A Language for Provisioning Network Resources. *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, 213–226. https://doi.org/10.1145/2674005.2674989

Takiguchi, T., Hidaka, A., Masui, H., Sugizaki, Y., Mizuno, O., & Asatani, K. (2012). A new application-level link aggregation and its implementation on Android terminals. *Wireless Communications and Mobile Computing*, *12*(18), 1664–1671. https://doi.org/10.1002/wcm.2329

Tootoonchian, A., Casado, M., & Sherwood, R. (n.d.). *On Controller Performance in Software-Defined Networks*.

Urera, F. L. J., & Balahadia, F. F. (2019). ICTeachMUPO : An Evaluation of Information E-Learning Module System for Faculty and Students. *International Journal of Computing Sciences Research*, *3*(1), 163–188. https://doi.org/10.25147/ijcsr.2017.001.1.31

Warraich, S. H., Aziz, Z., Khurshid, H., Hameed, R., Saboor, A., & Awais, M. (2020). SDN enabled and OpenFlow compatible network performance monitoring system. *ArXiv*, 1–10.

Xu, J., Wang, J., Qi, Q., Sun, H., & He, B. (2018). DEEP NEURAL NETWORKS FOR APPLICATION AWARENESS IN SDN-BASED NETWORK Jun Xu , Jingyu Wang , Qi Qi , Haifeng Sun , Bo He State Key Laboratory of Networking and Switching Technology Beijing University of Posts and Telecommunications , Beijing , China. *IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING*, *61771068*, 0–5.

Yamei, F., Qing, L., & Qi, H. (2016). Research and comparative analysis of performance test on SDN controller. *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, 207–210. https://doi.org/10.1109/CCI.2016.7778909

Yen, T. C., & Su, C. S. (2014). An SDN-based cloud computing architecture and its mathematical model. *Proceedings - 2014 International Conference on Information Science, Electronics and Electrical Engineering, ISEEE 2014*, *3*, 1728–1731.

https://doi.org/10.1109/InfoSEEE.2014.6946218

Zhang, Y., Cui, L., Wang, W., & Zhang, Y. (2017). Author ' s Accepted Manuscript A Survey on Software Defined Networking with Multiple Controllers. *Journal of Network and Computer Applications*. https://doi.org/10.1016/j.jnca.2017.11.015

Zhou, Y., Ruan, L., Xiao, L., & Liu, R. (2014). A Method for Load Balancing based on Software- Defined Network. *Semantic Scholar*. https://doi.org/10.14257/ASTL.2014.45.09