IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 6, June 2020

Dynamic Bandwidth Utilization in Software - Defined Campus Based Networks: A Case Study of the Kwame Nkrumah University of Science and Technology

Dr. James Dzisi Gadze[1], Kobby Asare Obeng[2], Justice Owusu-Agyeman[3]

Senior Lecturer, Department of Telecommunications Engineering, College of Engineering, Kumasi, Ghana[1]

Student Researcher, Department of Telecommunications Engineering, College of Engineering, Kumasi, Ghana[2]

Student Researcher, Department of Telecommunications Engineering, College of Engineering, Kumasi, Ghana[3]

Abstract: The efficient utilization of bandwidth in campus networks is a major traffic engineering issue. It requires a complete knowledge of the underlying physical network architecture as well a means to automate or reactively and proactively program the network. The static nature of traditional network creates a hurdle that must be overcome to achieve the above. The Software Defined Network architecture proposes a novel way to automate, program and dynamically configure computer networks. This work uses the VMware virtualization software and the GNS3 network emulator to emulate a Software Defined-based campus network. A data plane made up software-based replicas of network devices is designed and configured to connect to a controller software. A network application scheme is implemented by leveraging the Hierarchical Token Bucket Queuing Discipline which automatically programs bandwidth allocation at the data plane through the controller based on traffic demands. The functionality of the architecture is tested by carrying out a number parallel-connections to simulate changing traffic patterns. This is done using the Iperf Application. The results show the conversion of a traditional campus network into a Software Defined-based campus network. It also depicts the complete emulation of the entire Software Defined-based campus network. At the data plane of the emulated network, devices are able to forward packets to one another with the most active port forwarding about 9,000 packets. The controller obtains a global view of all 11-network devices in the emulated network. The latency between the controller and the software defined switches at the data plane ranges between 50 and 62.5 milliseconds. The throughput between the controller and the software defined switches at the plane ranges between 2 and 9 Mbps. Application Plane to Control Plane communication in the emulated network is executed in an average of 30 milliseconds and bandwidth utilization occurs in a minimum of 11seconds and peaks at 27.5 seconds. It however becomes steady at 17 seconds as traffic patterns vary.

Keywords: SDN, virtualization, GNS3, Hierarchical Token Bucket, automation.

I. INTRODUCTION

Network Technology is currently going through its third major revolution. The first revolution was the shift from circuit switching to packet switching. This involved the use of the packet as the main means of transmitting a message from one device to another. The second revolution was the shift from the hard-wired mode to the wireless mode of switching which saw the introduction of Wi-Fi technology, 3G,4G and 5G technologies. The third revolution in network technology has to do with a shift from the hardware-based mode of networking to a software-based mode of networking [1]

Figure 1 represents a typical enterprise network that is responsible for providing different technological services such as IP video conferencing, video surveillance, printing, scanning and internet browsing for various users. It provides these services through a set of network devices such as switches which facilitate communication between various users on different parts of the network.

Existing computer networks like the above network have a number of limitations. These include network provisioning complexity, tightly managed network functions, technology specific connections and purpose-built hardware which carry out each network function. Such an environment like the above could also consist of multivendor equipment manufacturers with their own means of orchestrating and controlling specific equipment. Current networks are also limited by the fact that the applications which run on the devices in the network are programmed to suit current network needs [2]

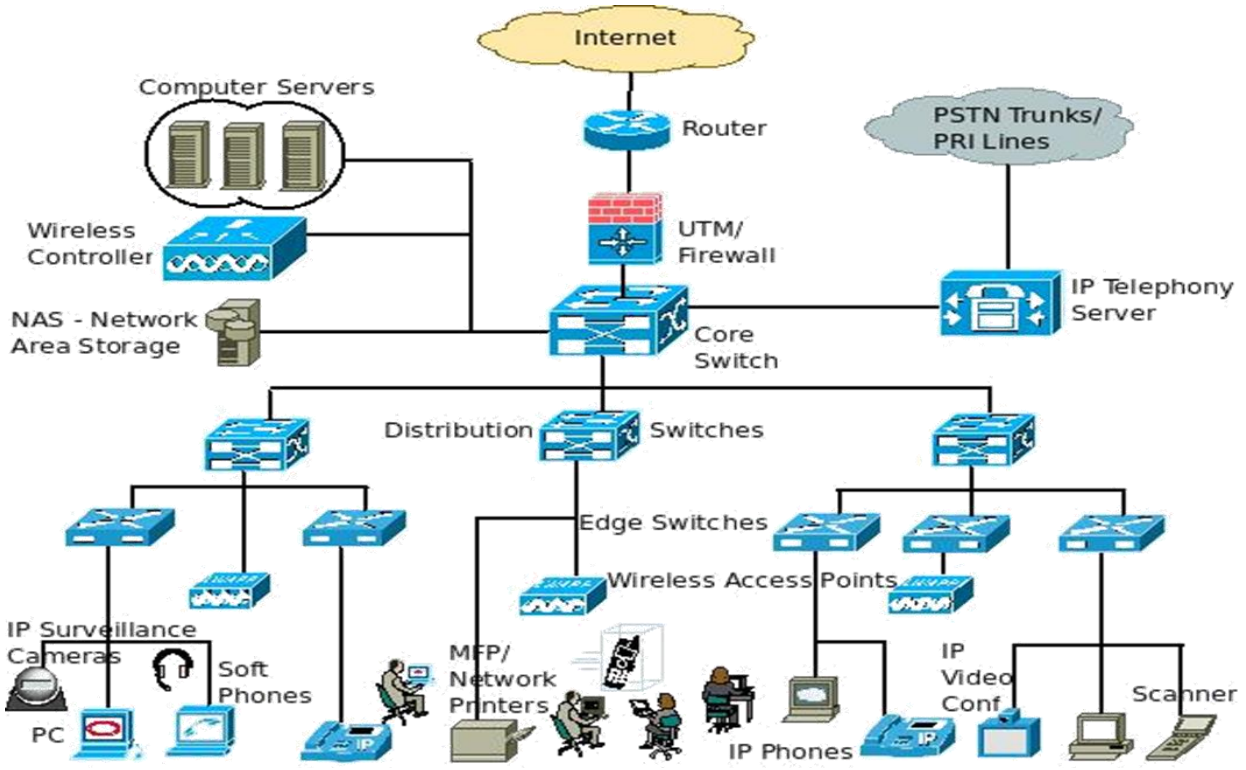


Figure 1 A Typical Enterprise Network
Source: http://www.excitingip.net/27/a-basic-enterprise-lan-network-architecture-block-diagram-and-components/

The Kwame Nkrumah University of Science and Technology plays hosts to a campus enterprise network. [3] The campus network is split up into two main parts; the faculty portion consisting of all the colleges and the residential portion consisting of halls of residence and the residences of the university's workers. Both portions of the University's network have a fixed bandwidth allocation to facilitate networkwide communication. This static assignment of bandwidth for communication creates a problem. Majority of students and lecturers find themselves in the faculty area in the day. During this period, traffic volumes at that portion of the network increases affecting network performance. This is seen in difficulty to browse the web, download content or stream resources for teaching and learning. The reverse is seen in the evening as the concentration of traffic shifts to the halls of residence. The devices that make up the Local Area Network of the school do not have the ability to dynamically prioritize and optimize the allocated bandwidth to facilitate communication in the network. In as much as there can be a manual reconfiguration of these devices it would require systematic planning, addition of new nodes, constant analysis of traffic and vast knowledge from technical experts. The reiteration of such processes every single day would be tiring and costly. Without recourse to increasing resources or overhauling existing infrastructure a way must be found to efficiently use the bandwidth already allocated in a dynamic need-based manner to ensure optimal performance. This situation provides an opportunity for the proposal of a new architecture that will optimize the already existing bandwidth at both portions of the network based on the numbers of students, lecturers and staff members who are accessing the network at any place at any point in time.
Software Defined Networking presents an approach to solve the major limitations that plague existing campus networks. It seeks to provide a programmable open scale approach to designing, building and managing networks.
Through the principle of plane separation, it provides a centralized view of distributed network states.
The Software Defined Networking architecture consists of two planes. These are the control and data planes. The control plane has a central intelligent agent called a controller that carries implements the logical, decision-making aspects of a network through programmed network policies also known as the Northbound Bound Interface protocol. The Data Plane consists of programmable OpenFlow switches that facilitate communication between the controller and network devices using the Southbound Interface Protocol called OpenFlow. A unified and global view of networks as envisaged by the SDN paradigm creates a powerful centralized platform for efficiently managing networks. A centralized, self-provisioning network will have the ability to implement changes without the need for the tortuous planning and anticipation that is needed in carrying out provisioning in current networks. Automating of network protocols provides an easier way to configure, run and change the protocols that make the network devices function. It provides a way to dynamically configure network protocols on a needs-based approach. Traffic Engineering is one major aspect of networking in which Software Defined Networking architecture and principles can be applied. Traffic

Engineering basically seeks to manage the flow of traffic within a network by taking stock of the topology and changing traffic patterns in order to prevent network resources from being constrained. Increasing volumes of traffic in networks places a strain on bandwidth which is one of the most important resources in a network. Software Defined Networking proposes a way of efficiently utilizing bandwidth by using the controller to dynamically allocate capacity on a needs-based basis. The proposition seeks to use the controller's knowledge of the characteristics of the links in the network topology to respond to changes in traffic volumes in various portions of the network by prioritizing the bandwidth required for traffic flow in the network. [4]

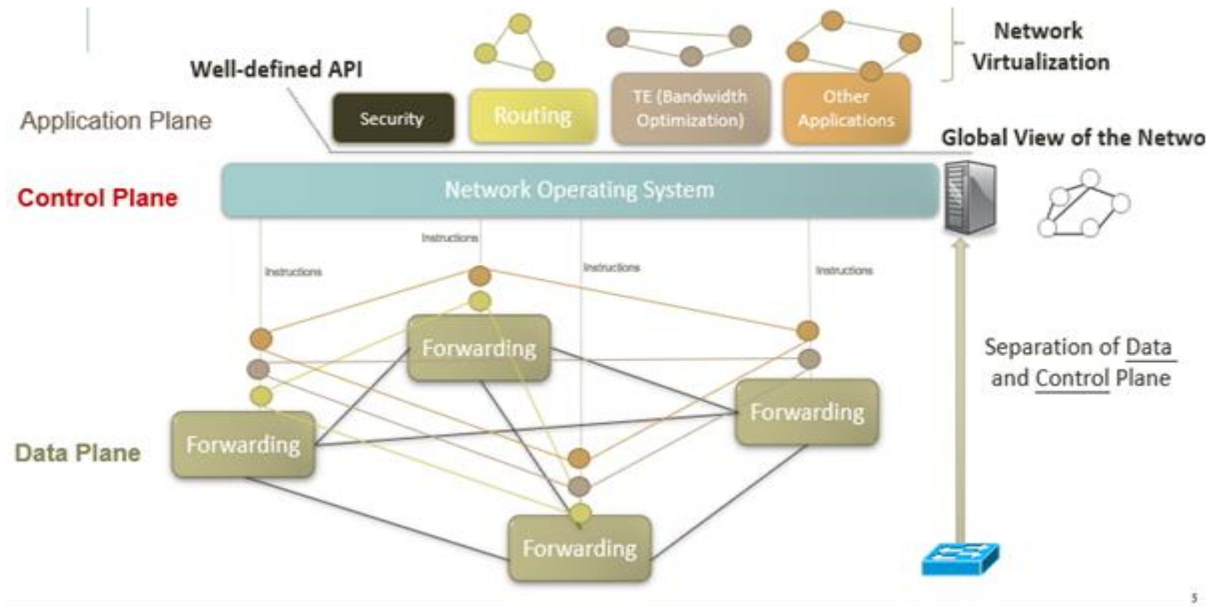


Figure 2 The Software Defined Networking Architecture
Source: The Future of Networking and the Past of Protocols by Scott Shenker et al

This work seeks to propose the Software Defined Networking approach to dynamically utilize the bandwidth allocated to the faculty and residential portions of the Kwame Nkrumah University of Science and Technology campus network. There are seven main sections of this work. The first part is the introduction. The second part takes a look at research carried out by others in the field of Software Defined Networking. The third and fourth parts delve into the theoretical basis for the work and the proposed solution respectively. The methodology followed in carrying out the work is stated in part five. In part six, the emulation environment is explained while part seven discusses the results obtained from the work. Part eight and nine are the conclusion and references respectively.

II. RELATED WORK

There has been a vast amount of research into the implementation of Software Defined Networking concepts in Enterprise and Carrier Networks. The main issues in enterprise networks are security, load balancing, Quality of Service configurations, bandwidth management and traffic control.

This section takes a look at related research in the above fields with respect to the tools used for emulation and the metrices used when carrying out research into Software Defined Networking concepts. In [5], S. Bhelekar et al from the Sardar Patel Institute of Technology in India present a dynamic load balancing strategy in Software Defined Networking. They took into account the number of active connections to a set of individualized servers and the shortest path taken from a specific client to a specific server in order to avoid congestion.

The authors used the Mininet Emulation Tool and VirtualBox Software to simulate a fat tree topology of a modern data center network at the data plane.

One key network functionality that can be implemented in the application plane of a Software Defined Network is security. N. Zope et al in [6] Usha Mittal Institute of Technology, Mumbai, India took a look at replacing a physical switch in a network by virtual switches and the development of a firewall and load balancing application in a Software Defined Network. The work seeks to demonstrate that most of the firewalls in traditional computer networks can be replaced by software firewalls. Using the Floodlight controller and the REST API, a set of firewall rules were constructed and pushed to an independent physical network consisting of an Open vSwitch and two hosts. The firewall module is capable of pushing flows containing the firewall rules to the hosts to either block or allow traffic.

In [7], M. S. Olimjonovich from the Tashkent University of Information Technologies used the Mininet simulator, a topology consisting of 5 switches and 10 host in the Python language to carry out a research on management of network resources in SDNs. Specialized language modules were developed for the hardware switches using Python programming. These software modules were remotely managed via encrypted SSH-channel in CLI mode. He designed a REST API to increase the amount of unused resources of the networks allowing approximately a 30% increase in the efficiency of network management.

Gu et al [8] from Japan designed an application for on-demand bandwidth pricing using the Software Defined Network Architecture. The application was based on a Stackelberg game constructed to analyse the competitive interactions between an ISP and a home network. A pricing strategy was determined using the Nash equilibrium solution of the Stackelberg game. Using the pricing strategy, network subscribers can decide the bandwidth to be reserved in an on-demand basis. The research was carried using an SDN enabled carrier network consisting of an SDN-enabled backhaul, a Controller, OpenFlow switches, a Controller API and a data plane consisting of wireless access points. The simulation was carried out using MATLAB R2016a. All applications were modelled using a Mathematical Network Model Simulation.

Results show that during off-peak, mid-peak, and peak time, the payoff of network subscribers is improved by 388.9%, 134.6%, and 19.8% respectively. The payoff of ISP is improved by 98.5%, 47%, and 7%, respectively. Results show that the optimal price increases with the increase of traffic load. Also, with the increase of traffic load, a large portion of surplus goes to ISP.

Z. Shu et al [9] from China and South Korea, propose a framework responsible for monitoring and analysing real-time network traffic as a prerequisite for traffic management. The authors designed the framework in a hybrid IP/SDN network consisting of two SDN capable switches and a single controller. The proposed framework utilizes the Link Layer Discovery Protocol (LLDP) implemented in an SDN capable switch to gain a complete view of the network while measuring the number of flows generated as the network elements communicate.

D. Satasiya and Raviya Rupal D [10] from the University of Pune, India carried out an analysis of a Software Defined Network implementation of a firewall designed by Karamjeet Kaur et al [11] who created topology including the POX controller, 6 OpenFlow switches and 5 hosts. Karamjeet Kaur et al built a learning switch application and firewall application which restricted or allowed the traffic by proactively placing rules into the network based on key network parameters such as the IP address. into an OpenFlow switch-based source MAC address, destination MAC address (Layer 2), source IP address, destination IP address (Layer 3), network protocol, destination port (Layer 4). Results show the firewall was able to block access to the whole network. It also allowed web access to one host in a scenario where all other communications were on going. In addition to the above, after firewall implantation latency got increased which implied that the firewall introduced overheads, throughput got reduced which means unwanted traffic was reduced.

A stateful firewall implementation was carried out by [12] P. Krongbaramee and Y. Somchit from Chiang Mai University in Thailand. The research focused on the use of the Open vSwitch to configure a stateful firewall using a TCP three-way handshake. The Mininet emulation tool was used with virtual server in the Digital Ocean Cloud. An analysis of the results shows that the average connection time of host in the stateful firewall is 20.06 milliseconds while the average connection time of host in the stateless firewall is 18.05 milliseconds. The stateful firewall increases the time for each connection for only about 2.01 milliseconds or about 11.14% longer. This is attributed to the SDN switch having to update rules when denying or permitting the connection of an external host to the network. The authors believe that performance would be faster when using with a hardware based SDN switch.

To exploit the capabilities of Software Defined Networking when implementing a Bandwidth on Demand service, A. Mendiola et al [13] proposed a framework called DynPaC, which they believe is able to provide efficient switching services based on bandwidth and vlan utilization. They used two Software Defined Networks in two different research labs located in Cambridge and Belgrade to create a multidomain network architecture. Each of the networks run based on the ONOS controller. In the emulation, a client based at the Cambridge campus tries to request two Bandwidth on Demand services from the server located in Belgrade using a portal developed on the DynPaC framework which resides on an intermediary network in Spain. DynPaC calculates the optimal intra-domain path taking into account the amount of bandwidth needed, the VLAN of the service and the service which has already been requested. The authors were able to prove that Bandwidth-on-Demand service provisioning is possible. This includes intra-domain bandwidth

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 6, June 2020

provisioning, limiting of traffic rate based on QoS requirements, link failure reactivity and automatic installation of backup paths.

Quality of Service in Software Defined Networks has gained attention over the past few years, A. O. Adedayo and B. Twala,[14] from University of Johannesburg carried out research into the use of Quality of Service configurations to control network bandwidth, latency and throughput. They leveraged the Mininet emulation tool to create a two-host topology at the data plane connected to an Open vSwitch. The Ryu controller was used as the centralized controller and Hierarchical Token Bucket Queuing discipline was used to ensure each queue in the QoS settings had a number of resources allocated to them. The authors used three scenarios to check for the use of traffic policing on the ports on the switch. They used the IntServ classification to assign bandwidth to certain services DiffServ classification to assign bandwidth to certain services. The results show that the use of DiffServ classification facilitates the scalability of QoS in the network. Also, the IntServ classification is capable of assigning bandwidth to different queues representing different classes of traffic with different priorities. The traffic policing scenario is also able to limit traffic to 10Mbps, while dropping traffic that exceeds 10Mbps.

F. Volpato et al [15] proposed a network application (Autonomic QoS Broker) and a controller module that implements the OpenVswitch Database Management Protocol (OVSDB). These two components used to provide QoS management based on the prioritization of queues in an SDN environment. The Autonomic QoS Broker is a resource and QoS provisioning application that was designed based on the MAPE-K control loop functionality. It was implemented to carry out analysis, planning, execution and monitoring of network resources and works in conjunction with a QoS configuration module. The module performs the configuration of switch’s QoS resources and the management of forwarding rules. The Mininet emulator, Open vSwitch (OVS) (version 2.5.0) and the Floodlight controller were used to carry out the experiment. The Broker improved flows throughput and packet loss rates. Flows in the same network path had the same latency values. A change in path caused an increase in latency values.

The concept of Virtual Local Area Network configuration in Software Defined Networks was explored in [16] by Van-Giang Nguyen and Young-Han Kim in Seoul, Korea. The authors designed and implemented an application for easily managing and flexibly troubleshooting the VLANs in an SDN architecture. They used an all OpenFlow data plane connected to the Floodlight Controller on the control plane. They developed a REST API-based module in the Floodlight controller to create static vlans in the underlying data plane. Two hosts in the vlan 10 network were able to reach each other. The authors also went further to set up a hybrid testbed consisting of an OpenFlow switch connected to an HP switch. Two hosts were connected to the HP switch and one host is connected to the OpenFlow switch. The REST API-based module was able to create vlans in the hybrid testbed as well. They also carried out an implementation of a dynamic vlan application based on the Mininet in out-of-band control mode. They used Floodlight controller in the same Mininet host. The results of the above showed that the time for sending packets and installing the flow modifications were independent of the type of topology. The latency on the switch and the controller were also very similar. An investigation into the operability of Software Defined Networking in wireless local area networks was carried out by A. Amelyanovich et al[17]. The work proposed a solution to the control traffic in wireless local area networks using a simulated version model of the St. Petersburg State University of Telecommunications network. The work sought to prove end-to-end quality of service support using traffic classification and priority-based queuing. The emulation was carried out using a two-switch-two wireless access point topology in Mininet. The OpenDaylight controller was used to facilitate the flow of different streams configured from the two Open vSwitches to the two hosts. The results showed that classification of traffic based on Differentiated Services Code Point (DSCP) values in the IP-headers of packets is possible in wireless networks. It reduces the number of simultaneously operating wireless devices. As A result, the effect of interference is also reduced. Another significant observation is the change in traffic priority with respect to applications which are sensitive to delay. Also, the work showed that the most frequent interval between messages was in the range from 100 µs to 1 ms. This shows that the controller responded to changes in the network.

Z. Shu et al[18] from China and South Korea, propose a framework responsible for monitoring and analysing real-time network traffic as a prerequisite for traffic management. The authors designed the framework in a hybrid IP/SDN network consisting of two SDN capable switches and a single controller. The proposed framework utilizes the Link Layer Discovery Protocol (LLDP) implemented in an SDN capable switch to gain a complete view of the network while measuring the number of flows generated as the network elements communicate.

S.-C. Lin et al[19] from the Georgia Institute of Technology proposed a framework for optimizing QoS and Routing in Software Defined Networks. They used tenant isolation, prioritization and flow allocation in a multitenant network for utilization. They designed network and switch hypervisors to isolate and prioritize tenants to create fine-grained isolation in the network. A dynamic flow allocation was also proposed in their work to enable optimal flow route selection. They also designed an adaptive feedback management tool to combine virtualization and flow allocation.

These three implementations were carried out using algorithms. In analysing the results, it was seen that the network and switch hypervisors were able to isolate three tenant networks in three subnets. The results show that feedback tool was able to maintain the number of shared links at a constant value. It was able to optimize route selection and wisely utilize link capacity for future flows. It was also capable of providing bandwidth for future flows. Umadevi et al.[20] proposed a scheduling algorithm for controlling the incoming data traffic in a Software Defined Network in an effective manner. The simulation was carried out using the OpenFlow package in OMNeT++. The authors constructed a multi-level switching queue. Multiple queues were maintained with varying priority levels. Analysis of the results shows that in case of a normal First Come, First Serve (FCFS) queue, bits are received by the queue only after the 1350 breakpoint. In the multilevel queue, packets were serviced even as the queue size decreases below 1350. Also, the packet drop count value was as high as 1600 in case of a normal FCFS queue. In the multilevel priority queue, packets enter different levels of queues thereby minimizing traffic in a single queue. The number of packets dropped in the case of multilevel queues was as low as 0.

Based on four objectives, Veena et al[21] from PES University in India proposed a way to manage and monitor a network using Software Defined Networking. The work was centred around optimizing the Mininet emulation tool in order to create custom topologies, collecting and preserving historical data from a controller for analysis, the reduction of layer 2 broadcast traffic in data centres and the introduction of Cross Layer Utilization algorithms for better resource utilization in Data Centres. They designed an Abstraction layer between the infrastructure and application planes. Also, Pseudo MAC addresses were used to reduce the ARP broadcast traffic in the emulated data centre. The enhancement of the Mininet Emulation tool, enabled the researchers specify the different link parameters such as bandwidth and latency using a simple text script. They were also able to create varied network topologies to test their ideas and new protocols
Gu et al [22] from Japan designed an application for on-demand bandwidth pricing using the Software Defined Network Architecture. The application was based on a Stackelberg game constructed to analyse the competitive communication between an ISP and a home network. A pricing strategy was determined using the Nash equilibrium solution of the Stackelberg game. Using the pricing strategy, network subscribers can decide the bandwidth to be reserved in an on-demand basis. The research was carried out using an SDN enabled carrier network consisting of an SDN-enabled backhaul, a Controller, OpenFlow switches, a Controller API and a data plane consisting of wireless access points. The simulation was carried out using MATLAB R2016a. All applications were modelled using a Mathematical Network Model Simulation. Results show that during off-peak, mid-peak, and peak time, the payoff of network subscribers is improved by 388.9%, 134.6%, and 19.8% respectively. The payoff of ISP is improved by 98.5%, 47%, and 7%, respectively. Results show that the optimal price increases with the increase of traffic load. Also, with the increase of traffic load, a large portion of surplus goes to ISP.

The key gap identified in all the reviews is the fact most of the simulations make use of the Mininet emulator.

The most significant limitation of Mininet is that it cannot work efficiently at high loads because it has one default scheduler that multiplexes CPU resources. Also, Mininet is not capable of prototyping large-scale networks having many nodes. Mininet does not offer a real-world approximation of network device functions. It also does not present an easy to use and adaptable graphical user interface for configuring networks.

The novelty with this works lies in the fact that the GNS3 Virtual Machine and VMware Workstation 14 are used to carry out the emulation of the entire Software Defined Networking Architecture. The choice of these two tools stem from the fact that they can be used to carry out nested virtualization. Nested virtualization is the ability of a virtualization tool to replicate different operating systems with the same tool. Nested virtualization provides the ability to test Software Defined Networking's proposition of vendor neutrality, openness and device simplification. In addition to the above, the GNS3 Virtual Machine makes use of a graphical interface that facilitates the design, configuration and testing of a virtual network.

III. THEORECTICAL BASIS

A Software Defined Network architecture promises flexibility and programmability in managing computer networks. The proposed SDN-based campus network can thus be programmed to dynamically borrow bandwidth from portions of the network where there is unused bandwidth based on the traffic demands or requirements in the network. The proposed dynamic bandwidth sharing technique is based on the theory of the Hierarchical Token Bucket Queuing Discipline.

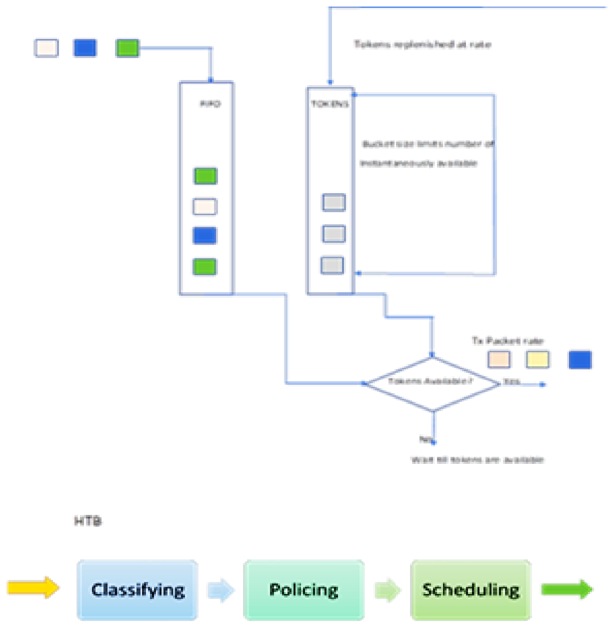

Tokens replenished at rate
FIFO
TOKENS
Bucket size limits number of
instantaneously available
Tx Packet rate
Tokens Available?
Yes
No
Wait till tokens are available
HTB
Classifying
Policing
Scheduling

Figure 3 Hierarchical Token Bucket Queuing Discipline

The Hierarchical Token Bucket Queuing Discipline (HTB) is a class-based queue discipline that controls the use of bandwidth on a given output link and implements efficient resource allocation. It uses the concept of multilevel token buckets to allow for efficient dynamic control of the egress bandwidth on a given link. HTB is based on hierarchical classes made up of three class types known as root, inner and leaf classes. In Hierarchical Token Bucket (HTB) classification is used for traffic control [23] HTB facilitates guaranteeing bandwidth to classes, while also allowing specification of upper limits to inter-class sharing and has the capability of prioritizing classes.[24]The root class represents the minimum and maximum amount of bandwidth (guaranteed bandwidth) that is set for communication between network devices. Every class has an associated Ceiling Rate (CR) and Rate (R). The CR specifies the maximum bandwidth that a class can use while R represents the minimum bandwidth the class can use. When one class requires bandwidth greater than R, it borrows bandwidth from its parent class until CR is reached, when this class reaches CR, the packets are queued until new tokens are available in the token-bucket [25]

IV. PROPOSED SOLUTION

This work seeks to propose the Software Defined Networking approach to dynamically utilize the bandwidth allocated to a campus-based network using the faculty and residential portions of the Kwame Nkrumah University of Science and Technology campus network as a case study. The first task in this work is to map the current KNUST Campus network into an SDN-based campus network as shown below

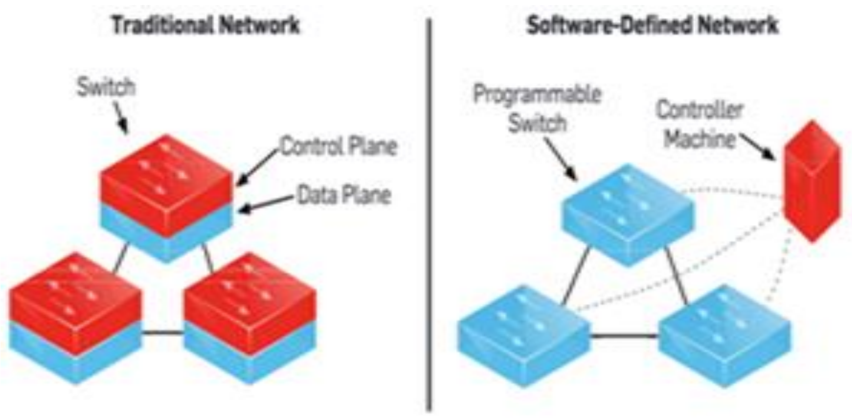


Figure 3 Traditional Network Architecture vs SDN Architecture
Source: The Future of Networking and the Past of Protocols by Scott Shenker et al

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

Vol. 9, Issue 6, June 2020

First the Traditional Network in Figure 1 is mapped into the SDN architecture in Figure 2

The core switch is replaced by an OpenVswitch which is a layer 3 routing switch which is capable of carrying out both switching and routing functions. The three distribution switches are replaced by three OpenVswitches. The five edge switches were replaced by six network devices (either routers or switches) which are connected to end users and the other equipment that form the access portion of the network. The OpenVswitches and network devices were connected to a Control Software to form the data plane or infrastructure layer. The Control Software was connected to a set of network applications which would facilitate configuration of the infrastructure layer creating the Northbound Interface. The resulting network architecture form the conversion is as shown in Figure 4

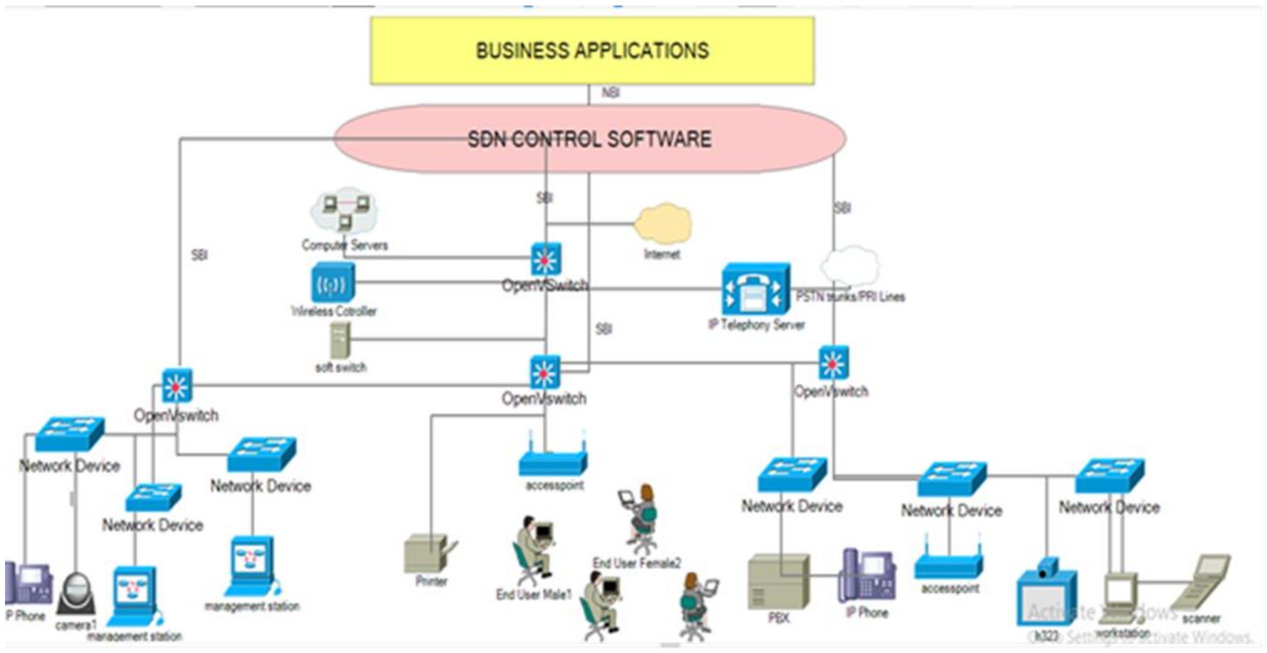


Figure 4: Generic Software Defined Enterprise Network

V. PAGE LAYOUT

The KNUST Campus network is divided into two portions. Portion 1 services the six colleges and the Institute of Distance Learning (IDL).

Portion 2 serves the residential area which is divided into Residential 1 and Residential 2.
Based on the conversion described above, a similar Software Defined Networking Model was developed for the conversion of the KNUST network into a Software Defined Network.

The following process was used:
Five OpenVswitches were used to represent a collapsed core and distribution switches Since an OpenVswitch is a layer 3 switch, it is capable of carrying out the functions of both a core switch and distribution switch. The five OpenVswitches are proposed based on the locations of the six colleges, the Institute of Distance Learning Centre (IDL) and the Residential part of the campus. The six colleges of the university were represented by six network devices each given an IP address. The residential portion of the network was also represented using two network devices. The KNUST campus network has access to the internet via two Internet Service Providers (ISPs). Two network devices were used in representing these. The five OpenVswitches were connected to the OpenDaylight Controller to form the Southbound Interface (SBI). The OpenDaylight Controller was connected to an application that would dynamically configure and optimize bandwidth in the underlying network through an Application Programming Interface (API). This forms the Northbound Interface. (NBI).

The schematic diagram of the proposed KNUST SDN campus network is shown in Figure 5 following the above mapping process

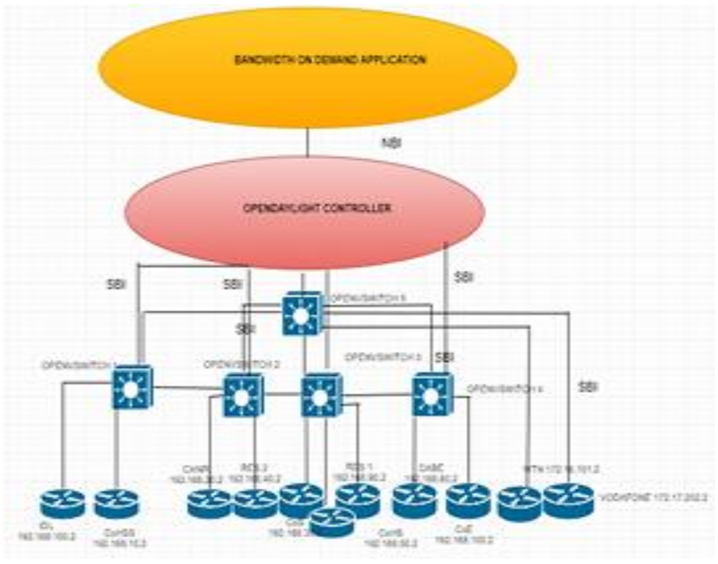


Figure 5: Proposed Software Defined Network Architecture for Kwame Nkrumah University of Science Technology

From the figure above the proposed architecture can be split into three planes. These are

i. The Infrastructure Plane which consists of eleven network devices(routers) representing the various faculty and residential portions of the campus network. This is the access network. This plane also consists of five OpenVswitches which represent the core network.

ii. The Control Plane: The OpenDaylight Controller is implemented at the control portion of the Software Defined-based campus network architecture. Communication between the Control Plane and the Infrastructure Plane is facilitated by connecting the management interfaces of each of the five OpenVswitches to the Controller. Each of the OpenVswitches use the OpenFlow version 1.3 protocol to exchange messages with the OpenDaylight Controller. The connection between the Controller and the switches represents the Southbound Interface (SBI)

iii. The Application Plane: The Bandwidth on Demand Application represents a QoS configuration command the implements the Hierarchical Token Bucket Queuing Discipline on the OpenVswitches using a REST API. The connection between the application and the Controller is the Northbound Interface (NBI)

VI. HTB ADAPTATION FOR SOFTWARE DEFINED-BASED CAMPUS NETWORK

The KNUST Campus link which connects the entire university to the internet is divided into two portions. One portion of the link goes to the faculty and the other portion of the link goes to the residential area. In adapting the HTB for the KNUST Campus an instance is considered where there is the need to allocate 500 Mbps to the College of Engineering (CoE) during the day due to high traffic demand and 200 Mbps to the residential portion during the day due to lower traffic demand.

The 500 Mbps allocated to the College of Engineering (CoE) needs to be subdivided into 100Mbps for the wired access to the office of lecturers and 400 Mbps for the wireless connections used by students. Any unused bandwidth from the 100 Mbps allocated to the lecturers' offices should be given to the wireless connection for students and vice-versa. If the total traffic requested at the College of Engineering (CoE) does not exceed 500 Mbps, the excess will be given to the residential portion. Using the theoretical explanation of HTB given earlier, the 500Mbps allocated for the College of Engineering CoE is the root class. It represents the Ceil Rate for all communication in the College of Engineering. The 400 Mbps and the 100 Mbps represent the inner classes for the College of Engineering (CoE). These values represent the Assured Data Rates for both the wired and wireless connections to the College of Engineering.
Figure 6 below demonstrates an adaption of HTB to the Software Defined-based campus network

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

Vol. 9, Issue 6, June 2020

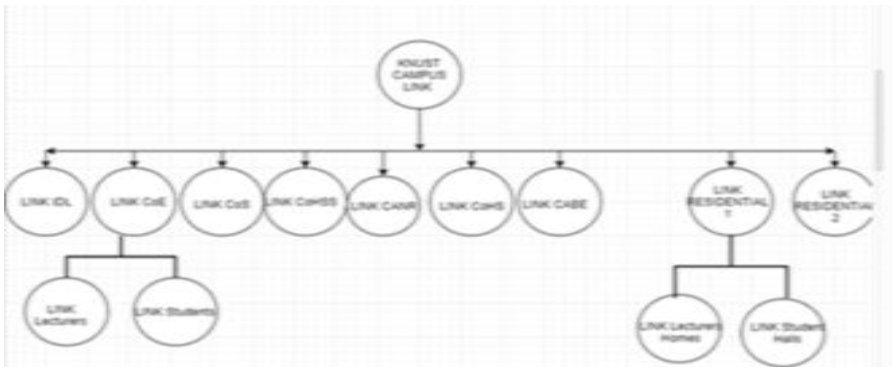


Figure 6 Sample HTB class hierarchy for KNUST SDN-based campus network

VII. METHODOLOGY

This section discusses the process involved in emulating the various parts of the Software Defined-based campus network as proposed in IV.
The VMware Workstation 14 Software is used to provide the platform for the implementation of the Infrastructure Layer and the Control Layer. VMware Workstation 14 is a software that is used to design virtual networks by creating software-based replicas of real network devices such as routers and switches and the links that connect them. It provides an ecosystem for designing virtual networks. The Bandwidth on Demand Application is designed and implemented within by using a Python Program to coordinates between the QoS Configuration, the OpenDaylight Controller and the Open vSwitches.

VIII. INFRASTRUCTURE LAYER IMPLEMENTATION

The Colleges and Residential portions of the network are implemented using virtual instances of Cisco c3600 routers Each of these is given an Ip address using the 192.168.x.x/24 block.

The OpenVswitches that make up the core network are implemented using Docker Containers. The management interfaces of each of the OpenVswitches is given an IP address from a DHCP pool created in VMware Workstation using the 192.168.198.x/24 addressing pool.

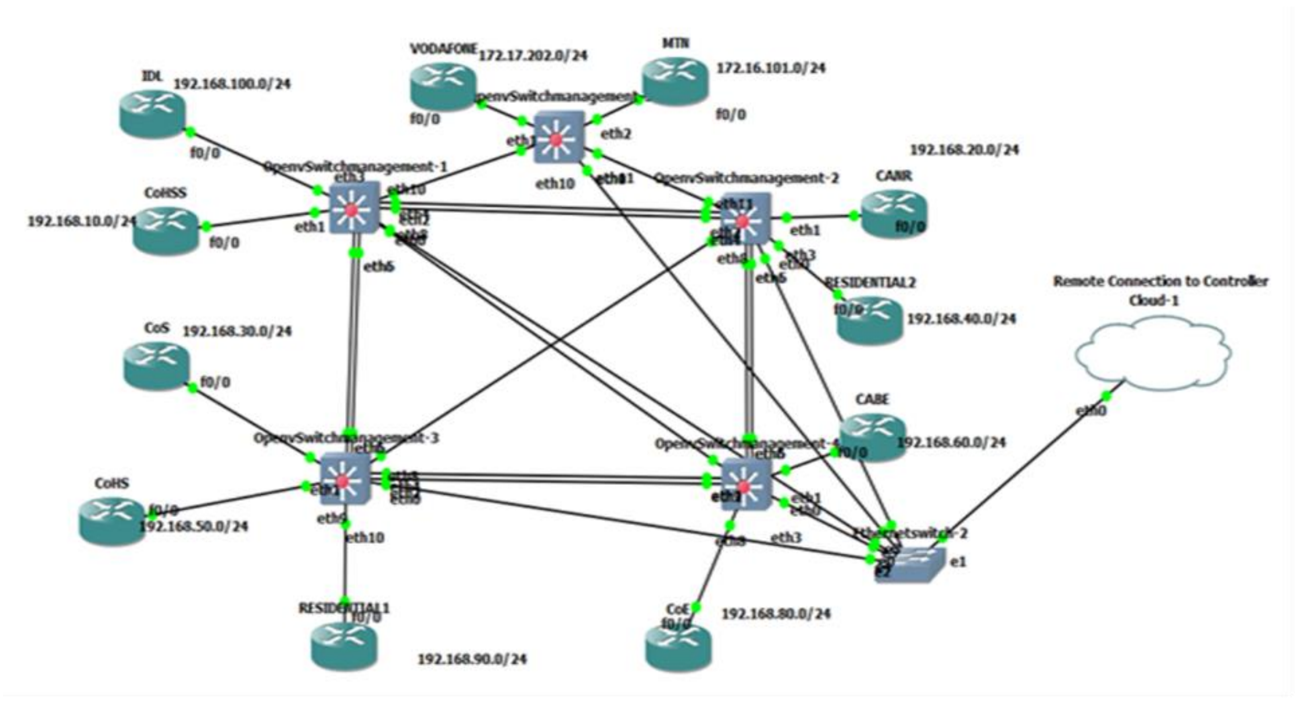
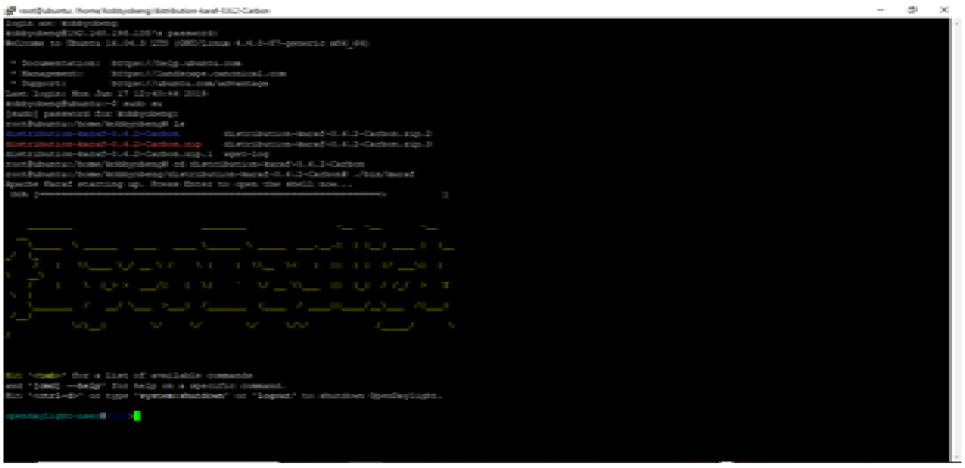


Figure 7: Graphical User Interface Implementation of Infrastructure and Controller in GNS3 and VMware

IX. CONTROLLER IMPLEMENTATION

The OpenDaylight Controller Software is downloaded from the OpenDaylight website, imported into VMware Workstation 14 as an Open Virtual Appliance file and configured with an IP address from the 192.168.198.x/24 pool similar to those of the OpenVswitches. The Open Virtual Appliance file runs an Ubuntu Operating System which contains the carbon edition of the OpenDaylight controller in a zip file. The zip file was extracted and the controller started by running the karaf file stored in the .bin directory of the Ubuntu Operating System.
The Figure below shows the fully installed controller which is described as Connection to Remote Controller in Fig 7



```
[illegible]
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-87-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
[illegible]
[illegible]
[illegible]
distribution-karaf-0.6.2-Carbon              distribution-karaf-0.6.2-Carbon.zip.2
distribution-karaf-0.6.2-Carbon.zip          distribution-karaf-0.6.2-Carbon.zip.3
distribution-karaf-0.6.2-Carbon.zip.1   wget-log
[illegible] cd distribution-karaf-0.6.2-Carbon
[illegible] ./bin/karaf
Apache Karaf starting up. Press Enter to open the shell now...
100% [========================================================================>]

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

opendaylight-user@root>
```

Figure 8: Controller Interface Implementation

X. BANDWIDTH ON DEMAND APPLICATION IMPLEMENTATION

The Bandwidth on Demand Application uses a python program that runs a Main Process. This Main Process uses web sockets to facilitate TCP connections between three sub-process. It coordinates connections to the script containing the QoS definition for the Bandwidth on Demand, the OpenDaylight Controller and the OpenVswitches.

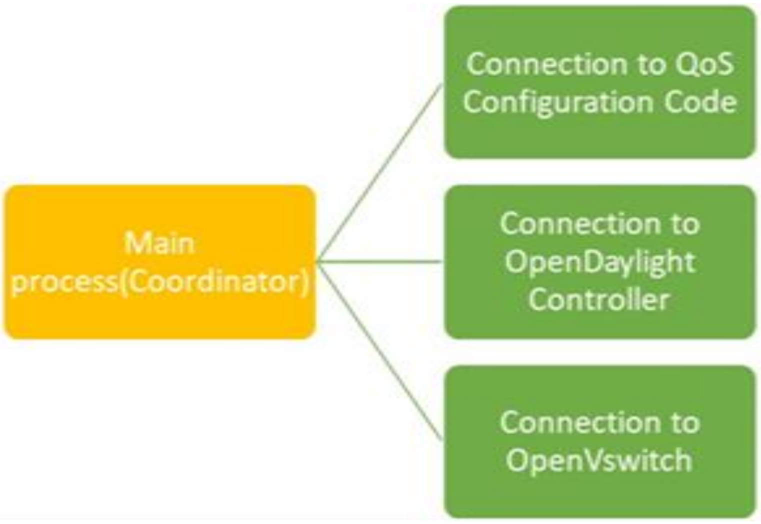

Connection to QoS Configuration Code
Main process(Coordinator)
Connection to OpenDaylight Controller
Connection to OpenVswitch

Figure 9: Implementation of Bandwidth on Demand Application

XI. EMULATION ENVIRONMENT

This section is a presentation of the experimental environment used in obtaining results
In order to obtain the metrices emanating from the interaction between the data plane and the control plane, the Iperf tool was used. Iperf has client and server functionality, and can create data streams to measure the throughput and round-trip time between the two ends in one or both directions.

Two desktop devices running the Ubuntu Operating were connected to OpenVswitch 1 representing IDL and OpenVswitch 4 representing the College of Engineering.
The desktop representing the College of Engineering is the client and is used in generating streams of packets in the form of a set of parallel threads to the desktop representing IDL which is the server.
The client generates an iterated number of parallel threads starting from 1 thread to 18 threads to the server with the root QoS class set at a minimum of 1 Mbps and maximum of 9 Mbps.
The Wireshark Packet Analyzer is used in getting the results for this section

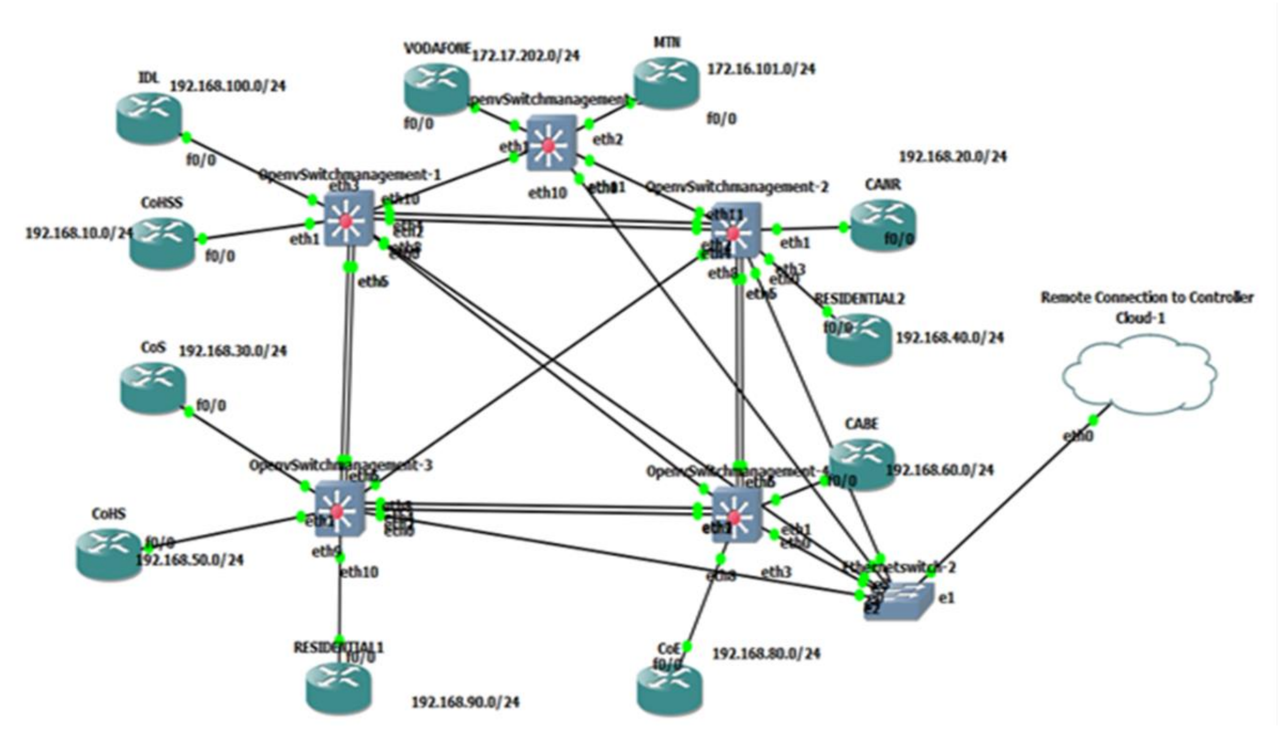


Figure 10: Graphical User Implementation used in GNS3 for obtaining metrices for data plane-control plane interaction

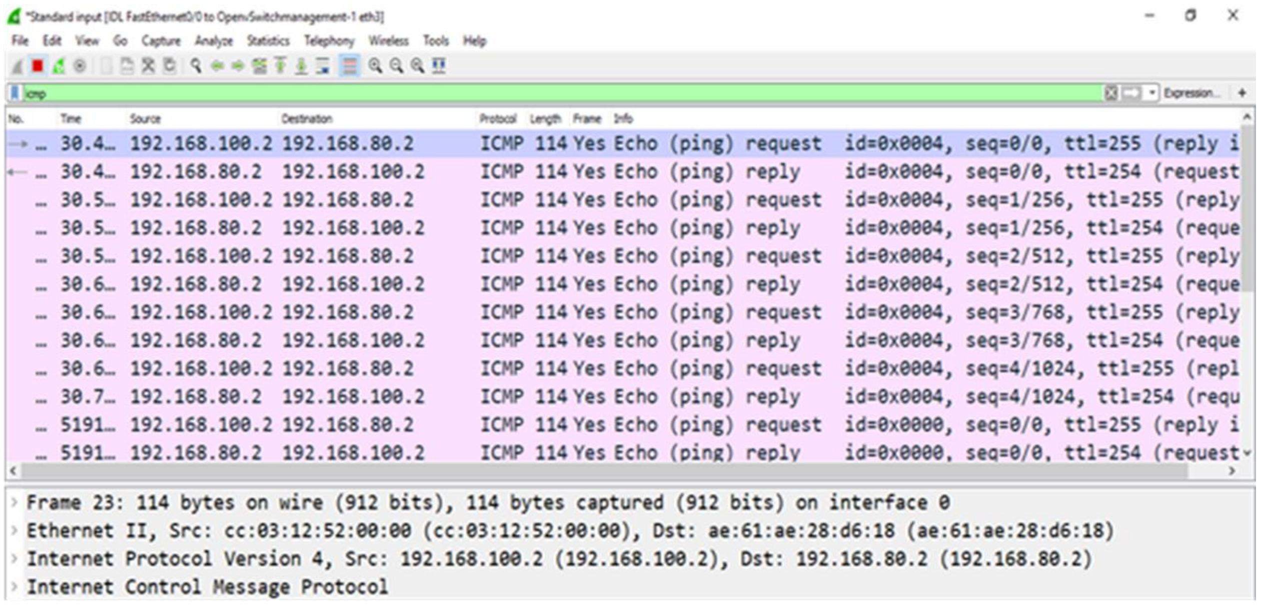
An easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

XII. EXPERIMENTAL RESULTS

This section is a presentation of the results based on the experiment carried out in 6
It is divided into three sections.
i. Results for Data Plane Communication
ii. Results for Data Plane-Control Plane Communication
iii. Results for Control Plane-Application Plane Communication

XIII. RESULTS FOR DATA PLANE COMMUNICATION

In order to check the forwarding of packets from one host to another, the Internet Control Message Protocol is used to carry out a ping trace from one host (192.168.100.2 representing IDL) to another host (192.168.80.2 representing the College of Engineering)



```
*Standard input [IDL FastEthernet0/0 to OpenvSwitchmanagement-1 eth3]
File  Edit  View  Go  Capture  Analyze  Statistics  Telephony  Wireless  Tools  Help
icmp                                                                     Expression...  +

No.  Time    Source          Destination      Protocol Length Frame Info
→ …  30.4…  192.168.100.2 192.168.80.2    ICMP 114 Yes Echo (ping) request  id=0x0004, seq=0/0, ttl=255 (reply i
← …  30.4…  192.168.80.2  192.168.100.2   ICMP 114 Yes Echo (ping) reply    id=0x0004, seq=0/0, ttl=254 (request
  …  30.5…  192.168.100.2 192.168.80.2    ICMP 114 Yes Echo (ping) request  id=0x0004, seq=1/256, ttl=255 (reply
  …  30.5…  192.168.80.2  192.168.100.2   ICMP 114 Yes Echo (ping) reply    id=0x0004, seq=1/256, ttl=254 (reque
  …  30.5…  192.168.100.2 192.168.80.2    ICMP 114 Yes Echo (ping) request  id=0x0004, seq=2/512, ttl=255 (reply
  …  30.6…  192.168.80.2  192.168.100.2   ICMP 114 Yes Echo (ping) reply    id=0x0004, seq=2/512, ttl=254 (reque
  …  30.6…  192.168.100.2 192.168.80.2    ICMP 114 Yes Echo (ping) request  id=0x0004, seq=3/768, ttl=255 (reply
  …  30.6…  192.168.80.2  192.168.100.2   ICMP 114 Yes Echo (ping) reply    id=0x0004, seq=3/768, ttl=254 (reque
  …  30.6…  192.168.100.2 192.168.80.2    ICMP 114 Yes Echo (ping) request  id=0x0004, seq=4/1024, ttl=255 (repl
  …  30.7…  192.168.80.2  192.168.100.2   ICMP 114 Yes Echo (ping) reply    id=0x0004, seq=4/1024, ttl=254 (requ
  …  5191…  192.168.100.2 192.168.80.2    ICMP 114 Yes Echo (ping) request  id=0x0000, seq=0/0, ttl=255 (reply i
  …  5191…  192.168.80.2  192.168.100.2   ICMP 114 Yes Echo (ping) reply    id=0x0000, seq=0/0, ttl=254 (request

> Frame 23: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0
> Ethernet II, Src: cc:03:12:52:00:00 (cc:03:12:52:00:00), Dst: ae:61:ae:28:d6:18 (ae:61:ae:28:d6:18)
> Internet Protocol Version 4, Src: 192.168.100.2 (192.168.100.2), Dst: 192.168.80.2 (192.168.80.2)
> Internet Control Message Protocol
```

Figure 11: Ping Trace Statistics

From the figure above, the ping trace shows a reply from the College Engineering host. This proves that forwarding of packets has been achieved.

XIV. RESULTS FOR DATA PLANE-CONTROL PLANE COMMUNICATION

The results below show communication between OpenVswitch 1 and the OpenDaylight Controller

i. OpenVswitch to Controller Latency (OpenVswitch 1)

From the figure below the latency between the switch and the controller becomes steady at 62.5 milliseconds as the traffic generation simulation is carried out. It peaks at 15.2 seconds after 400 seconds of the simulation.

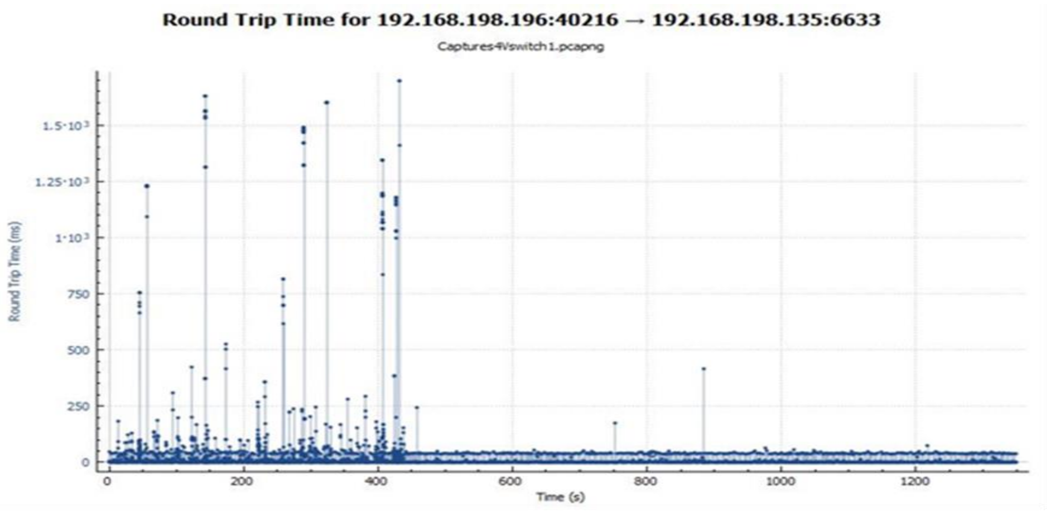


Figure 12: Latency between OpenVswitch 1 and Controller

i. OpenVswitch to Controller Throughput (OpenVswitch 1)

This metric measures the amount of data moved successfully from OpenVswitch 1 to the Controller in a given time period, and typically measured in megabits per second (Mbps)

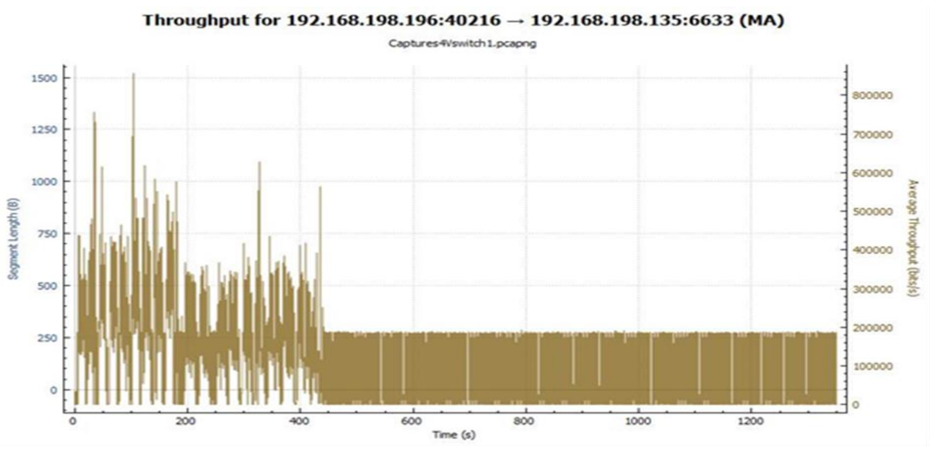


Figure 13: Throughput between OpenVswitch 1 and Controller

From the figure above, there is a steady flow of 2Mbps of data from the OpenVswitch to the Controller. The peak throughput is 8Mbps

XV. RESULTS FOR CONTROL PLANE-APPLICATION PLANE COMMUNICATION

The results below show communication between the OpenDaylight Controller and the Bandwidth on Demand Application. The results below show a record of the Iperf application running on the client device representing the College of Engineering connected to OpenVswitch 4 after the triggering of the QoS Configuration setting in the Bandwidth setting in the Bandwidth on Demand Application via an API call. These include Bandwidth Change per Thread Group and Bandwidth Change per Thread Group per Time.

i. Bandwidth Change per Thread Group

This metric shows the dynamic change of bandwidth based on the QoS Configuration setting which has the root class set to 1Mbps minimum and 9Mbps maximum. Based on classifying, policing, scheduling and borrowing mechanisms, bandwidth is changed dynamically per the parallel thread iteration.

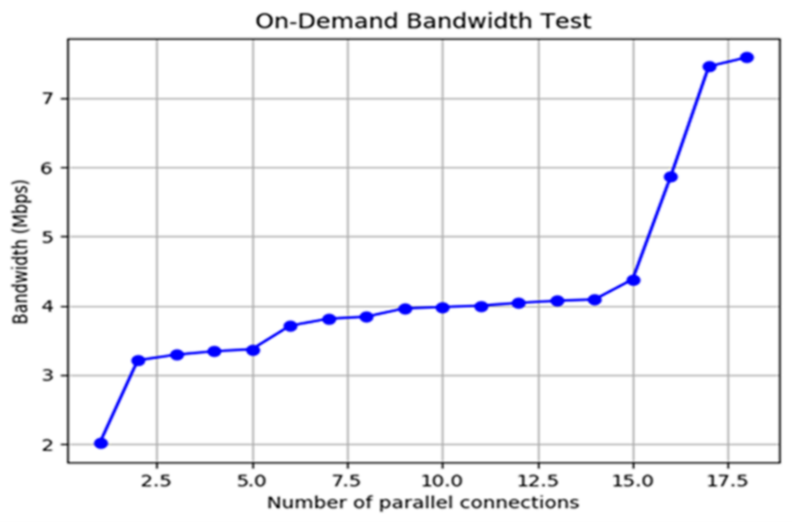


Figure 14: Bandwidth Change per Thread Group

In the figure above, a total of 4Mbps is allocated for the 10-parallel thread iperf request. There is a steady rise in allocated bandwidth from the 1thread iteration to the 18-thread iteration peaking at about 7.8 Mbps for the 18th thread. This clearly shows that bandwidth is borrowed from the root class to an inner class and is shared equally in a leaf class for every single thread iteration satisfying the QoS Configuration definition.

ii. Bandwidth Change per Thread Group per Time

This metric shows the amount of time taken for the QoS setting to dynamically change the bandwidth for the thread-based iteration.

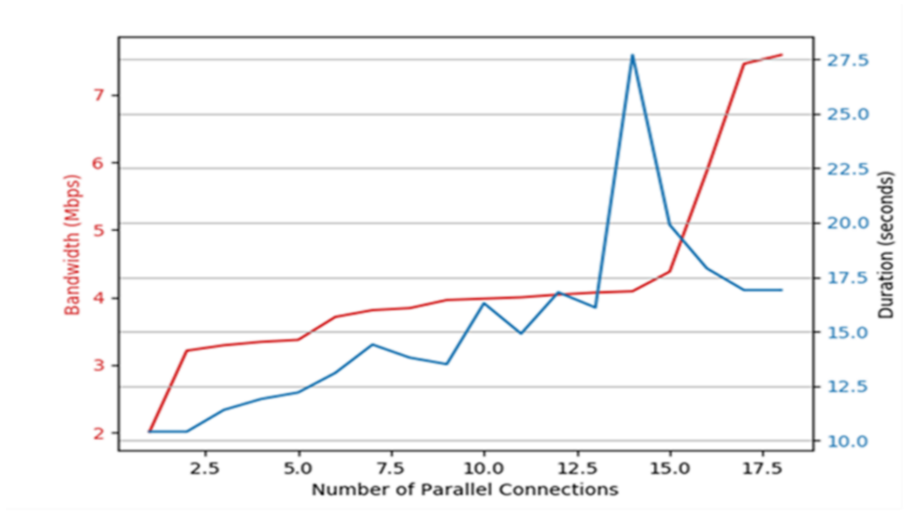


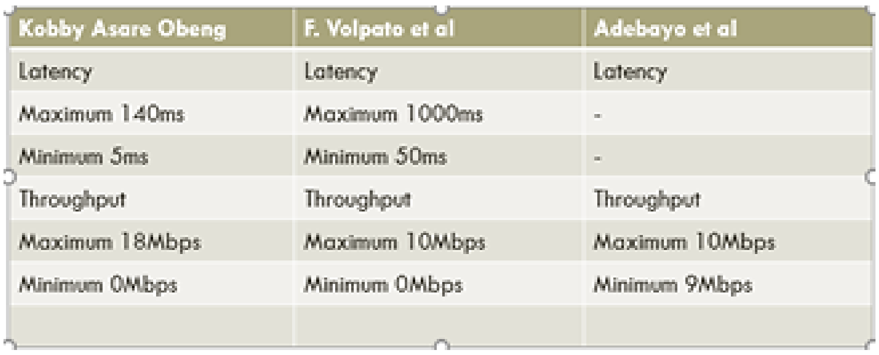
Figure 16: Bandwidth change per thread group per time.

The graph above shows that for 10 parallel connections 4 Mbps is allocated in 17 seconds. It takes 17 seconds to allocate a leaf class for each of the 10 parallel connections.
This allocation peaks at a value of 4.3 Mbps in 27.5 seconds for 13 parallel connections. The graph however takes a downward trend from this peak value of 27.5 seconds. The amount of time for the dynamic allocation to take place reduces steady and tapers to 17 seconds for the 18-thread iteration clearly demonstrating network automation. This implies that after 27.5 seconds, the Software Defined-based campus network dynamically adapts to the QoS configuration changes as the thread iterations increase. The controller dynamically adapts to the changing conditions in the network and carries out the bandwidth changes in a shorter space of time. These results clearly show that GNS3 and VMware can be used to emulate SDNs more effectively as opposed to virtual box and Mininet which are used in other works.

XVI. COMPARATIVE ANALYSIS

Below is a comparative analysis of this work with two reviewed works which took a look at similar metrics using the Mininet Emulator.

It shows that using the GNS3 virtual machine and VMware Workstation software reduces the latency and increases the maximum throughput of the software defined network.



Kobby Asare Obeng	F. Volpato et al	Adebayo et al
Latency	Latency	Latency
Maximum 140ms	Maximum 1000ms	-
Minimum 5ms	Minimum 50ms	-
Throughput	Throughput	Throughput
Maximum 18Mbps	Maximum 10Mbps	Maximum 10Mbps
Minimum 0Mbps	Minimum 0Mbps	Minimum 9Mbps

XVII. CONCLUSION

In this work a Software Defined-based campus network was designed and tested within a virtual environment which better emulates the software running on traditional network devices. Dynamic bandwidth utilization was carried out automatically by leveraging the Hierarchical Token Bucket Queuing Discipline in response to changing traffic patterns in the underlying network.

Funding: This study was not funded by an external grant. Authors used their own resources to carry out the work.

Conflict of Interest: The authors declare that they have no conflict of interest

REFERENCES

1. G. Pujolle, “Software Networks,” p. 262
2. Software Defined Networking for the Utilities and Energy Sector: Fujitsu Network Communications Inc
3. Omollo, Kathleen Ludewig, “Information and Communication Technology Infrastructure Analysis of Kwame Nkrumah University of Science and Technology and University of Ghana”
4. https://www.networkcomputing.com/networking/new-network-management-tactic-bandwidth-demand/592428015
5. S. Bhelekar, M. Iyer, G. Mehta, and S. Chaudhari, “Dynamic load balancing strategy in software-defined networking,” in 2017 International Conference on Trends in Electron-ics and Informatics (ICEI), Tirunelveli, 2017, pp. 875–878.
6. N. Zope, S. Pawar, and Z. Saquib, “Firewall and load balancing as an application of SDN,” in 2016 Conference on Advances in Signal Processing (CASP), Pune, India, 2016, pp. 354–359.
7. M. S. Olimjonovich, “Software Defined Networking: Management of network resources and data flow,” in 2016 International Conference on Information Science and Commu-nications Technologies (ICISCT), Tashkent, Uzbekistan, 2016, pp. 1–3.
8. B. Gu, M. Dong, C. Zhang, Z. Liu, and Y. Tanaka, “Real-time pricing for on-demand bandwidth reservation in SDN-enabled networks,” in 2017 14th IEEE Annual Consum-er Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2017, pp. 696–699.
9. Z. Shu et al., “Traffic engineering in software-defined networking: Measurement and management,” IEEE Access, vol. 4, pp. 3246–3256, 2016.
10. D. Satasiya and Raviya Rupal D., “Analysis of Software Defined Network firewall (SDF),” in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Chennai, India, 2016, pp. 228–231
11. K. Kaur, K. Kumar, J. Singh, and N. S. Ghumman, “Programmable firewall using Software Defined Networking,” p. 5, 2015
12. P. Krongbaramee and Y. Somchit, “Implementation of SDN Stateful Firewall on Data Plane using Open vSwitch,” in *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Nakhonpathom, 2018, pp. 1–5.
13. A. Mendiola *et al.*, “Multi-domain bandwidth on demand service provisioning using SDN,” in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, Seoul, South Korea, 2016, pp. 353–354
14. A. O. Adedayo and B. Twala, “QoS functionality in software defined network,” in *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, 2017, pp. 693–699.
15. F. Volpato, M. P. Da Silva, A. L. Goncalves, and M. A. R. Dantas, “An Autonomic QoS management architecture for Software-Defined Networking environments,” in *2017 IEEE Symposium on Computers and Communications (ISCC)*, Heraklion, Greece, 2017, pp. 418–423
16. Van-Giang Nguyen and Young-Han Kim “SDN-Based Enterprise and Campus Networks: A Case of VLAN Management,” *Journal of Information Processing Systems*, 2015
17. A. Amelyanovich, M. Shpakov, A. Muthanna, M. Buinevich, and A. Vladyko, “Centralized control of traffic flows in wireless LANs based on the SDN concept,” in *2017 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SINKHROINFO)*, Kazan, Russia, 2017, pp. 1–5
18. M. S. Olimjonovich, “Software Defined Networking: Management of network resources and data flow,” in *2016 International Conference on Information Science and Communications Technologies (ICISCT)*, Tashkent, Uzbekistan, 2016, pp. 1–3
19. S.-C. Lin, P. Wang, and M. Luo, “Jointly optimized QoS-aware virtualization and routing in software defined networks,” *Computer Networks*, vol. 96, pp. 69–78, Feb. 2016
20. “Umadevi et al. - 2017 - Multilevel queue scheduling in software defined networks.pdf
21. Veena S, R. P. Rustagi, and K. N. B. Murthy, “Network management and performance monitoring using Software Defined Networks,” in *20th Annual International Conference on Advanced Computing and Communications (ADCOM)*, Bangalore, India, 2014, pp. 29–31
22. B. Gu, M. Dong, C. Zhang, Z. Liu, and Y. Tanaka, “Real-time pricing for on-demand bandwidth reservation in SDN-enabled networks,” in *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, 2017, pp. 696–699
23. S. Ren, Q. Feng, Y. Wang, and W. Dou, “A Service Curve of Hierarchical Token Bucket Queue Discipline on Soft-Ware Defined Networks Based on Deterministic Network Calculus: An Analysis and Simulation,” J. Adv. Comput. Netw, vol. 5, no. 1, 2017.
24. D. G. Balan and D. A. Potorac, “Linux HTB queuing discipline implementations,” in 2009 First International Conference on Networked Digital Technologies, Ostrava, 2009, pp. 122–126.
25. T. Bhattacharjee, V. Gopal, L. N. Ngangoua, and C. Raghunath, “TrafficLight: Network Traffic Monitoring and Allocation,” p. 9