The Relevance of Development and Deployment of Software Defined Networking Solutions for a University Network

J.K. Mensah, J.A. Abandoh-Sam, H. Amankwah, E. Tchouchu

Abstract: *The rapid advancement of technology in today's world necessitates the development and deployment of robust and efficient networking architectures. This paper focuses on the relevant of creation and implementation of software defined networking (SDN) solutions for a university network, which offer improved network reachability and ease of navigation for network technicians. Utilizing the Ubuntu server command line interface within a virtual platform, all software defined networking components were seamlessly installed using a secure file transfer protocol client. To evaluate the effectiveness of the implemented SDN, Mininet was employed to supply OpenFlow switches within the software defined networking environment. This allowed for the integration of commands into a controller for thorough testing and assessment. The main parameters used in this paper are hosts, switches and a controller as the focus is on a single controller network. Because software defined networking centralizes and optimizes computer networks and removes irrelevant packets and frequent assaults for quick network performance, it makes networking intelligent and has several advantages over traditional networks. The successful implementation of software defined networking, as demonstrated in this paper, ensures that network technicians can swiftly identify and address challenges within computer networks of universities. This not only simplifies their tasks but also contributes to the overall efficiency and reliability of the network infrastructure.*

Keywords: *Software Defined Networking; Controller; Traditional Network; Virtualization; Open flow Switches.*

I. INTRODUCTION

Traffic increases every day as far as computer networking is concerned [1]. This is a result of the daily increase in the number of people using digital gadgets. Furthermore, computer networking is already a standard feature for the majority of people and businesses. According to [2], A network is made up of two or more computers that exchange data and resources. They stated that companies are implementing intranets for the sole purpose of enabling them to gather, arrange, and distribute information more quickly and effectively than they could have done in the past. Because of this, most organizations are operating on an intranet to share digital data internally. Time and energy efficient technological infrastructure is needed to handle this traffic in order to deliver dependable and affordable networking. The majority of businesses utilize the internet to promote their goods and services internationally and to be viewed by a global audience. For this to happen, organizations must ensure efficient, reliable, and cost-effective network infrastructure to make their network intelligent. The intelligence of an intelligent switch is limited because it cannot know what happening on the network. Because of the associated costs, most networking infrastructure are not traditionally intelligent, even though the emergence of the internet has increased competition in the business world. In a network that is not intelligent, network engineers do diagnostics from one switch to another and point-to-point troubleshooting to solve network problems. According to [3], increased traffic through the outbreak of mobile devices has caused the need for a robust network infrastructure. This is because the old network infrastructure has the bandwidth compromised. It therefore appears that software defined networking is needed in our networking infrastructure today because it is challenging to manage and apply functions to the traditional network making it difficult to understand. Software defined networking is purely open flow that introduces the separation of the control plane from the data plane so as to control the network from a central location and ensure network reachability [4,5,6]. It is an application that, centralized the network, manages traffic, and makes finding and fixing network problems easier while there are limitations when the traditional approach is used.

This paper is about the importance of creating and implementing software defined networking in institutions and organizations. software defined networking is required when a first layer traditional network (installed devices such as keystones faceplates, switches and laying of network cables) is established [4]. It is suitable for current networking because of the current demand. It implementation and maintenance is also less demanding than the traditional approach [7]. The Second Layer Traditional network (Introduction of managed switch) makes the network very complex to manage. According to [8][34], instead of manually inputting numerous command lines on various network devices, it is now only necessary to accept the instructions from the software defined networking controller.

Manuscript received on 12 January 2024 | Revised Manuscript received on 12 February 2024 | Manuscript Accepted on 15 February 2024 | Manuscript published on 30 March 2024.

**Correspondence Author(s)*

Joseph Kofi Mensah*, Department of Mathematics Computer Lab, University of Cape Coast, Ghana, Cape Coast, Ghana. E-mail: joseph.mensah4@ucc.edu.gh, ORCID ID: 0009-0007-1516-2503

Joseph Ahor Abandoh-Sam, Department of Computer science, Valley View university, Accra, Oyibi, Ghana. E-mail: abandoh@vvu.edu.gh

Henry Amankwah, Department of Mathematics, University of Cape Coast, Ghana, Cape Coast, Ghana. E-mail: hamankwah@ucc.edu.gh

Emmanuel Tchouchu, Department of Management, University of Cape Coast, Ghana, Cape Coast, Ghana. E-mail: temmanuel@ucc.edu.gh

This eliminates the need to understand and interpret thousands of protocols. According to [9], using high-level languages and application programming interface, software defined networking provides a framework that enables network managers to automatically and dynamically manage and control network components. Software defined network is a technology that supports computer network administrators [10]. It enables configuration, security, and optimization of network resources quickly by dynamic automated software defined networking programs that administrators can create themselves because these programs are not dependent on protected software [4].

Figure 1 [8] shows the logical view of OpenFlow [8,11,12] in two planes that clearly shows the separation of the controller from the forwarding device (data plane) through application programming interfaces (API) [13].

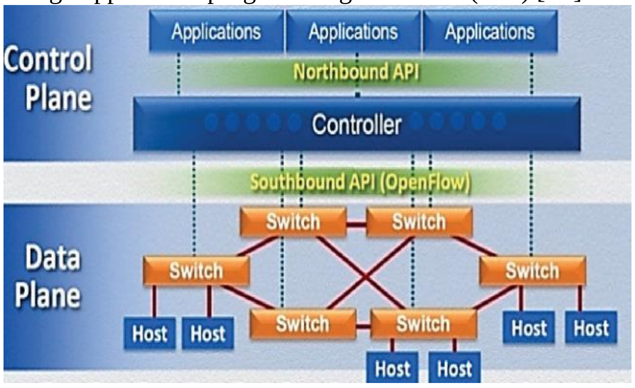


Fig. 1: Logical View of Software Defined Networking Architecture

Here, we see that in the control plane, the applications are connected through the northbound API to the controller. In the data plane, hosts are connected to switches. This data plane is connected to the controller through the southbound API **[14].**

Figure 2 shows the three main layers of the software defined networking architecture as can be found in [3].

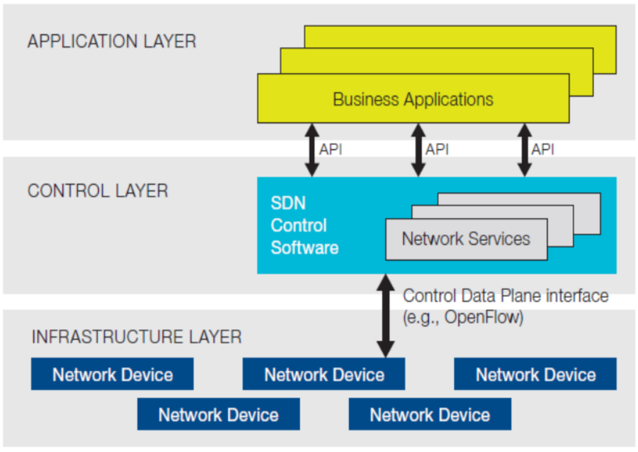


Fig. 2: Main layers of Software Defined Networks

The application layer contains the business applications that are connected to the control layer through APIs. The control layer is embedded with network operating system that contains network services [15]. The infrastructure layer contains network devices (switches) that are connected to the control layer through APIs [16][32][33].

According to a study by [17], the software defined networking controller [18] is a logical entity that sends instructions or requirements to the networking components from the software defined networking application layer. Additionally, the controller gathers data about the network from the hardware networking applications, providing statistics and events about what is happening on the network for decision making. The data processing and forwarding capabilities of the network are under the control of the software defined networking devices [19].

Using protocols like the rapid spanning tree protocol (RSTP), switches in traditional networking decide the network path through which packets should be sent [20]. They also went further to say that, in SDN, the decision-making functions are removed from the switches and handled by a centralized controller software that makes all the decisions for the network. The switches, in turn, receive packet forwarding instructions from the SDN controller. They can now concentrate just on the actual physical forwarding of packets.

The primary and backup paths for each communications flow on the network can be set using the controller. According to [21] every device is instructed about what to do during a network breakdown. There is essentially no delay in forwarding packets when there is a failure. The controller speeds up recovery and reduces packet loss. This is because there is no need to negotiate forwarding paths, as in an RSTP Ethernet network. Packets use the same links in a traditional Ethernet network. However, not all are authorized in the network, else there will be traffic restricting the total bandwidth utilization of the networks slowing the link. Therefore, utilizing SDN allows for the allocation of each application to its path, maximizing network bandwidth [22].

Software defined networking comes with benefits and some of which are as follows:

Network traffic management is optimized by cutting out unauthorized traffic, controlling the entire network path, and having the power to set bandwidth [23].

II. METHODS

The following are the procedures to configure software define networking controller on Windows using the Hewlett Packard Virtual Application Networks (HP VAN) SDN Controller:

a) We set up a virtual platform (hypervisor).

b) Further, we import and set up Ubuntu server operating system on the virtual platform.

c) Next, we set up mininet on the virtual platform.

d) By using series of commands by HP, we install the virtual networking component.

e) By using series of commands by HP, we import the controller Debian package with the help of safe file transfer protocol (SFTP) client (e.g., WinScp or Filezilla) and unpack.

f) By using script, we set up a graphical user interface login of the controller.

g) Finally, by using series of commands by HP, we set up virtual OpenFlow switches [24] from mininet for the controller.

This will connect the switches to the SDN controller. Here, the administrator can have access to the devices on the network.
Figure 3 shows the layout of both networking infrastructure.

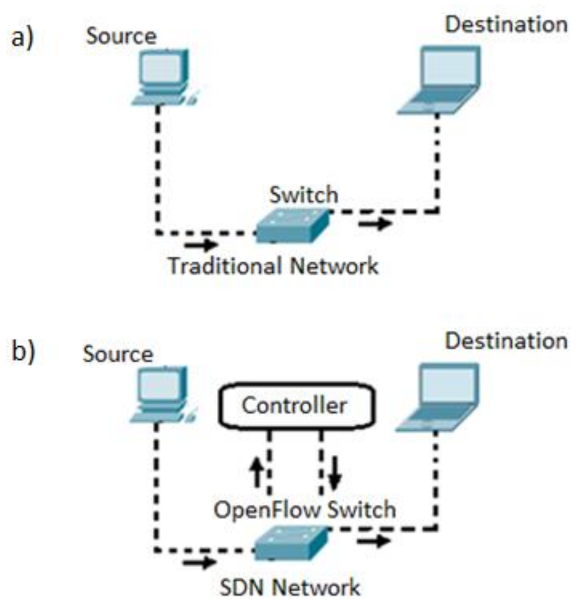
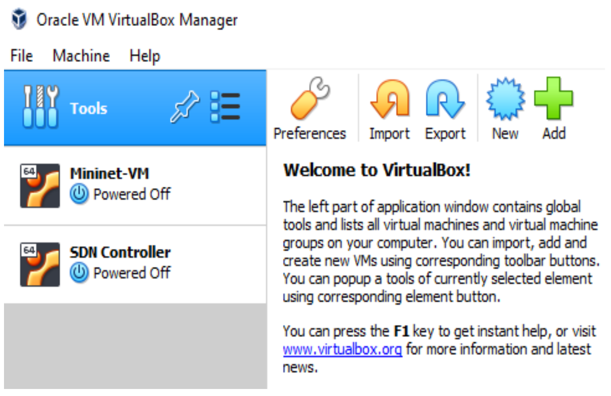


Fig. 3: Layout of a) Traditional and b) SDN Networking Infrastructure

In this figure, a) (traditional network) shows that, packets from the source are manage by the switch to the destination. However, b) (SDN Network) has introduced a controller that manages the flow of packet on it networks [25,26].

A. Network Virtualization

A hypervisor can run several operating systems depending on the processer, memory, and hardisk capacity. It also allows virtualized operating systems to share resources [27]. The type of hypervisor used in this paper is Type 2, that is, a VirtualBox. Figure 4 shows a graphical interface of a Virtualbox.



Welcome to VirtualBox!

The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.

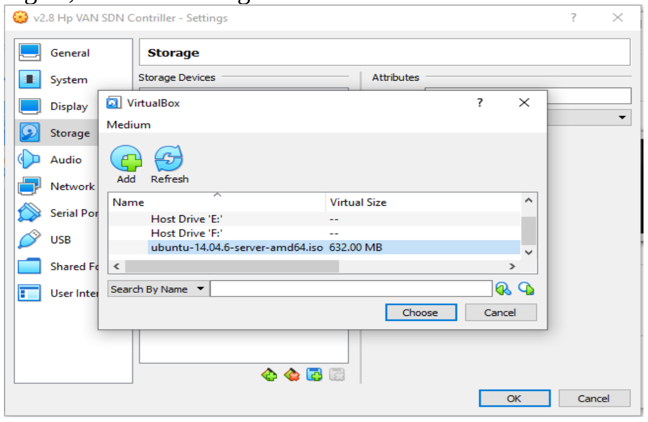
You can press the F1 key to get instant help, or visit www.virtualbox.org for more information and latest news.

Fig. 4: Graphical Interface of the Virtualbox

The figure displays the installed mininet virtual machine and the controller virtual machine. Mininet provides openflow switches for the virtual platform whiles the controller virtual machine allows the installation of network components.

B. Ubuntu Server

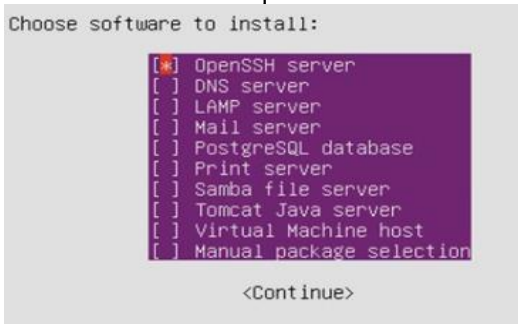
Ubuntu 14.046 server amd64.iso image is the required operating system. On the storage settings, Ubuntu 14.046 is then imported into the virtual machine before installation begins, as shown in Fig. 5.



Name	Virtual Size
Host Drive 'E:'	--
Host Drive 'F:'	--
ubuntu-14.04.6-server-amd64.iso	632.00 MB

Fig. 5: Imported Ubuntu 14.04 Server Operating System

After the importation of Ubuntu server, the operating system type must be linux, and the version should be Ubuntu (64-bit). On the network settings, the bridged adapter is selected to allow the secure shell (SSH) to connect to the Ubuntu server operating system. This provides a duplicate window of the server or operating system in use. The virtual machine is then started to setup the server (Ubuntu 14.04). Figure 6 shows the activated open SSH server.

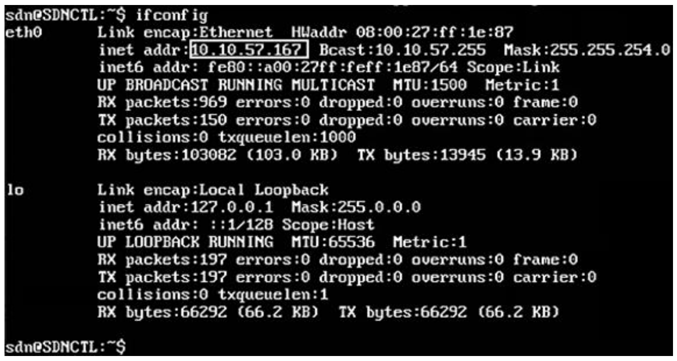


```
Choose software to install:

    [*] OpenSSH server
    [ ] DNS server
    [ ] LAMP server
    [ ] Mail server
    [ ] PostgreSQL database
    [ ] Print server
    [ ] Samba file server
    [ ] Tomcat Java server
    [ ] Virtual Machine host
    [ ] Manual package selection

                <Continue>
```

Fig. 6: Activated Open Secure Shell Server

We continue until the setup process is finished. Note that a user name and a password are relevant to set the login prompt as well as the activation of the open SSH server to allow a connection to putty and a safe file transfer protocol.

Figure 7 is the command line interface of the SDN controller. To configure SSH (putty), the safe file transfer protocol (SFTP), and the network switches, we run ifconfig command to view the controller IP address information.

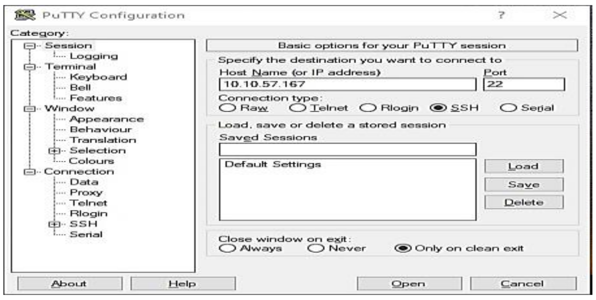


```
sdn@SDNCTL:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:ff:1e:87
          inet addr:10.10.57.167  Bcast:10.10.57.255  Mask:255.255.254.0
          inet6 addr: fe80::a00:27ff:feff:1e87/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:969 errors:0 dropped:0 overruns:0 frame:0
          TX packets:150 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:103082 (103.0 KB)  TX bytes:13945 (13.9 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:197 errors:0 dropped:0 overruns:0 frame:0
          TX packets:197 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:66292 (66.2 KB)  TX bytes:66292 (66.2 KB)

sdn@SDNCTL:~$
```

Fig. 7. Command Line Interface of the Software Defined Networking Controller

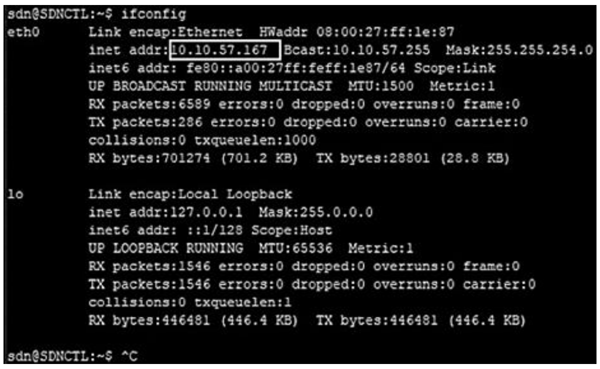
figure shows the internet protocol address of the software defined networking controller for identification. It has been placed in a box on the third line in Fig. 7.

Figure 8 is a successful duplicate of the command line interface of the SDN controller.



Specify the destination you want to connect to
Host Name (or IP address): 10.10.57.167
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Close window on exit: Always Never [✓ Only on clean exit]

a)



```
sdn@SDNCTL:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:ff:1e:87
          inet addr:10.10.57.167  Bcast:10.10.57.255  Mask:255.255.254.0
          inet6 addr: fe80::a00:27ff:feff:1e87/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6589 errors:0 dropped:0 overruns:0 frame:0
          TX packets:286 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:701274 (701.2 KB)  TX bytes:28801 (28.8 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1546 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1546 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:446481 (446.4 KB)  TX bytes:446481 (446.4 KB)

sdn@SDNCTL:~$ ^C
```

b)

Fig. 8: a) Configured Putty and b) Displayed Putty

The login and the display of the server are the same as the putty as shown in Figure 8. Before starting to run the commands to install the components, the system needs to be updated by connecting to Ubuntu cloud with this command: sudo apt-get updates on the SDN controller platform. In SDN implementation, a series of commands are used to install devices and software. These commands are vital information created by [28] to implement software-defined networking. There is also the need to install a SFTP (WinScp) client, and for that matter, winscp software is installed. The installation of the SFTP client and putty are done on the master operating system which is windows. It is vital to have a SFTP to transfer the Debian package (hp-van-sdn-ctrl-2.7.16-x64.zip) from windows platform to Linux (Ubuntu server). This is to activate the controller software on the virtualized platform. Now, the installation of the components can begin and the commands are as follows:

~$ sudo apt-get update:

This command connects the HP VAN SDN controller to the Ubuntu cloud to update to the most recent package. Note that this is the first command.

~$ sudo apt-get install python-software-properties:

Here, there will be a prompt on the hard disk size. Type "Y" to continue.

~$ sudo apt-get install Ubuntu-cloud-keyring

This will install Ubuntu-cloud-keyring.

~$ sudo add-apt-repository cloud-archive: Juno

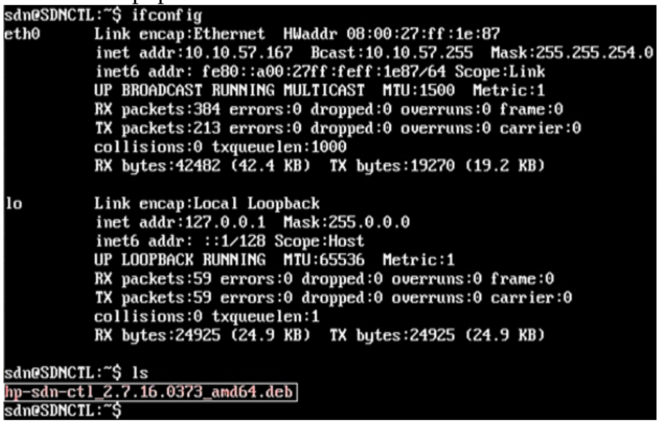
This will provide direction to the Ubuntu cloud archive and display the following prompt:

Press [ENTER] to continue or ctrl-c to cancel. Then, enter to continue.

With these commands, the system connects to the Ubuntu cloud and installs the needed software.

The following commands are responsible for the installation of the host.
~$ sudo apt-get update
~$ sudo apt-get install keystone
During this installation process, a prompt requires an answer before it continues. It may be either Prompt A or Prompt B. If it is Prompt A, type Y to continue, however, if it is Prompt B, type N to continue. Further, the system continues after a selection is made as follows:
Prompt A: Do you want to continue [Y/N]?
Prompt B: Configuration file ==> Modified (by you or a script) since installation. The package distributor has shipped an updated version. What would you like to do about it? Your options are:
Y or I: install the package maintainer's version
N or O: keep your currently-installed version
D: show the differences between the versions
Z: start a shell to examine the situation
The default action is to keep your current version.
*** dmk.sh (Y/I/N/O/D/Z) [default=N]?

After installing the keystone, there is the need to upload the Debian package of the HP VAN SDN controller from the windows platform to the Ubuntu server on the virtual machine. This is done by the using SFTP (WinSCP). Figure 9 is an example of a successful transfer of the Debian package used in this paper.



```
sdn@SDNCTL:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:ff:1e:87
          inet addr:10.10.57.167  Bcast:10.10.57.255  Mask:255.255.254.0
          inet6 addr: fe80::a00:27ff:feff:1e87/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:384 errors:0 dropped:0 overruns:0 frame:0
          TX packets:213 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:42482 (42.4 KB)  TX bytes:19270 (19.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:59 errors:0 dropped:0 overruns:0 frame:0
          TX packets:59 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:24925 (24.9 KB)  TX bytes:24925 (24.9 KB)

sdn@SDNCTL:~$ ls
hp-sdn-ctl_2.7.16.0373_amd64.deb
sdn@SDNCTL:~$
```

Fig. 9: Installed Debian Package

After the transfer, a command: "~$ ls" is entered to confirm a successful transfer of the Debian package, hp-sdn-ctl_2.7.16.0373_amd64.deb, as shown in Figure 9. This package is the main controller component. We next unpack and install the controller components. We now verify whether the computer hardware meets the SDN Controller requirement and install by using the command:
~$ sudo dpkg --unpack hp-sdn-ctl_2.7. 16.0373_amd64.deb.
If it does, it will continue to unpack hp-sdn-ctl_2.7. 16.0373_amd64.deb. If it does not, it fails with several error prompt occurring.
However, the following override this error.
~$ touch /tmp/override.txt

This command instructs the system to accept the system hardware. This overrides the error notice. The following command is now used to unpack the Debian package of the controller:

~$ sudo dpkg --unpack hp-sdn-ctl_2.7. 16.0373_amd64.deb.

This makes the controller ready to be installed.

To install the package of the controller, we use the command: ~$ sudo apt-get install –f.

We use the following command to check the status of the controller:

~$ sudo service sdnc status

This command displays a piece of information that verifies that the controller services have started working. It also displays the follows:

~$ sdnc start/running, process number

The following is the script that runs the controller in a graphical user interface on the webpage application:

~$ sudo /opt/sdn/admin/config_local_keystone

This allows the administrator to login to the controller's graphical user interface that contains the address: http:/0000:8443/sdn/ui/, where 0000 means the SDN controller IP address.

III. RESULTS AND DISCUSSION

The results are presented in this section. Figure 10, for example, shows a traditional network with a break at Switch 4.

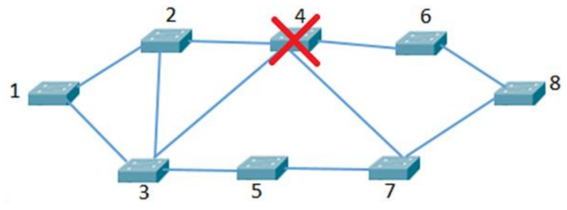


Fig. 10: Traditional Network

The switches of the network in Figure 10 are all intelligent. However, the intelligence of this network is limited because the intelligence of every switch is within itself [29] (it is unaware of the state of the other switches on the network). For example, switch 1 chooses the best path and sends packets to Switch 6. If Switch 1 sends packets to Switch 6, switch, 1 is unaware of the state of Switch 4. The packet may no longer reach its destination and flow across the network if Switch 4 is damaged. This is just one of the reasons for software defined networking.

Figure 11 shows the main layout of SDN.

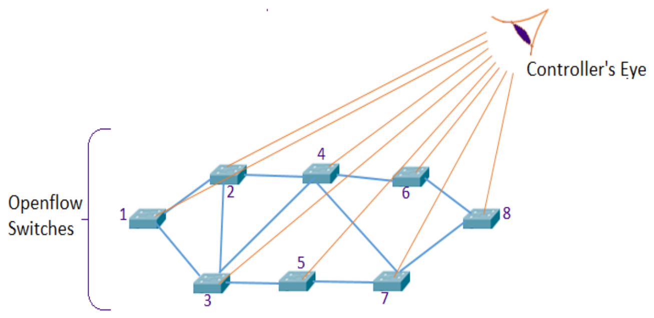


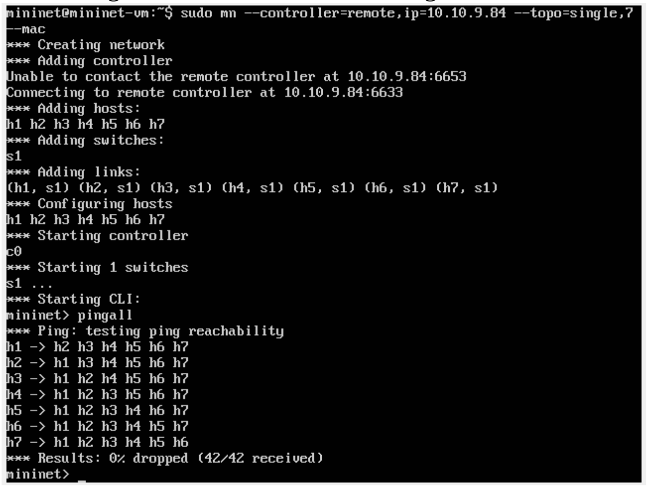
Fig. 11: Main Layout of Software Defined Networks

In Figure 11, the controller has an eye on all activities and every switch on the network. In this case, the SDN controller decides on the correct path to forward traffic and is also aware of the state of every switch on the network. The following parameters provide an SDN environment on a virtual platform.

The command:

$ sudo mn --controller=remote,ip=10.10.9.84 --topo=single,7 –mac

integrates mininet into the controller and creates the networking environment as shown in Fig. 12.



```
mininet@mininet-vm:~$ sudo mn --controller=remote,ip=10.10.9.84 --topo=single,7
--mac
*** Creating network
*** Adding controller
Unable to contact the remote controller at 10.10.9.84:6653
Connecting to remote controller at 10.10.9.84:6633
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7
h2 -> h1 h3 h4 h5 h6 h7
h3 -> h1 h2 h4 h5 h6 h7
h4 -> h1 h2 h3 h5 h6 h7
h5 -> h1 h2 h3 h4 h6 h7
h6 -> h1 h2 h3 h4 h5 h7
h7 -> h1 h2 h3 h4 h5 h6
*** Results: 0% dropped (42/42 received)
mininet> _
```

Fig. 12: Integrating Mininet into Controller

This figure shows one switch, one controller and seven hosts and how they are connected on the network. It also shows that this network should be created on that controller bearing this IP address 10.10.9.84, as shown in Fig. 12. The IP address of the controller may keep changing depending on the type of server in use.

Figure 13 displays the outcome of Fig. 12.

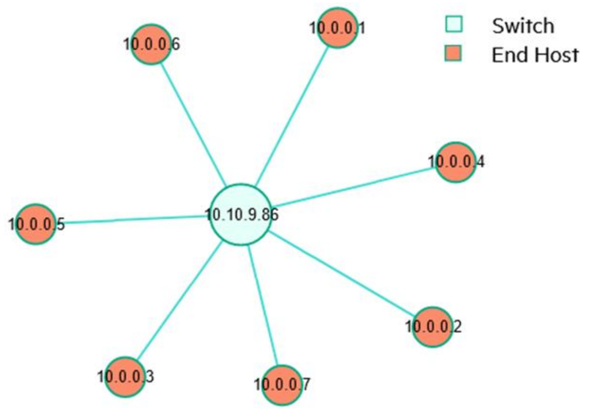
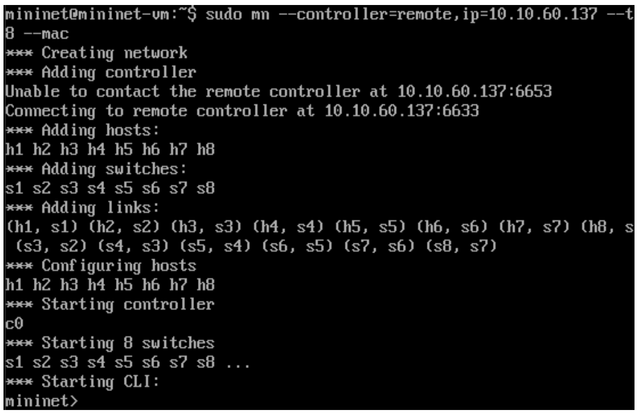


Fig. 13: Controller Network Topology

Figure 13 is the graphical user interface of the result in Fig. 12. The network created can be manipulated according to the demand on the network or the administrator [30,31].

Figure 14 is SDN command line interface of a network with eight switches.



```
mininet@mininet-vm:~$ sudo mn --controller=remote,ip=10.10.60.137 --t
8 --mac
*** Creating network
*** Adding controller
Unable to contact the remote controller at 10.10.60.137:6653
Connecting to remote controller at 10.10.60.137:6633
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (h6, s6) (h7, s7) (h8, s
 (s3, s2) (s4, s3) (s5, s4) (s6, s5) (s7, s6) (s8, s7)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 8 switches
s1 s2 s3 s4 s5 s6 s7 s8 ...
*** Starting CLI:
mininet>
```

Fig. 14: Controllers Graphical User Interface

Here, the command on the first line of Fig. 14 creates the network on Fig. 15.

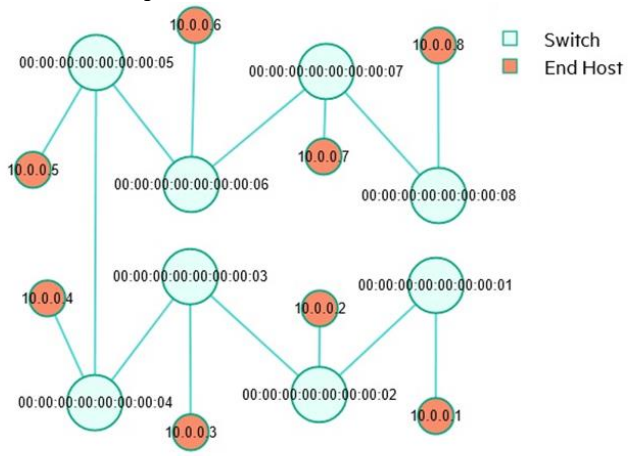


Fig. 15: Software Defined Networking Interface

This figure shows SDN network without a collapse switch as seen in Fig. 11. Should there be a collapse switch, there will be an immediate indication (the path connecting the collapse switch will disapear) and a change in packet path when using a complex topology. On this platform all the nodes connected to this network is visible to the Administrator.

As an example, we consider a campus area network shown in Fig. 16.

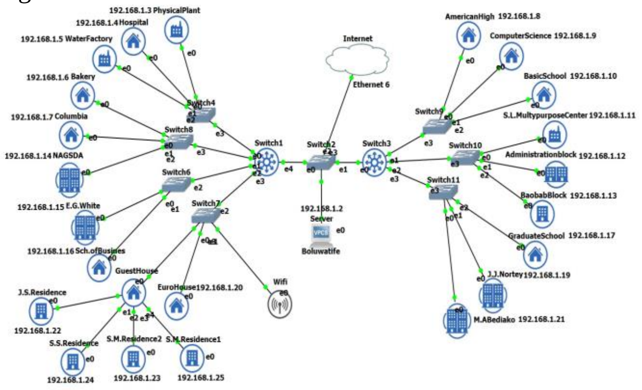


Fig. 16: Traditional Campus Area Network

Since there is no controller, it is difficult to detect network errors in a traditional campus area network as shown in Fig. 16. With SDN, detecting errors can be very easy. Figure 17 shows SDN campus area network.

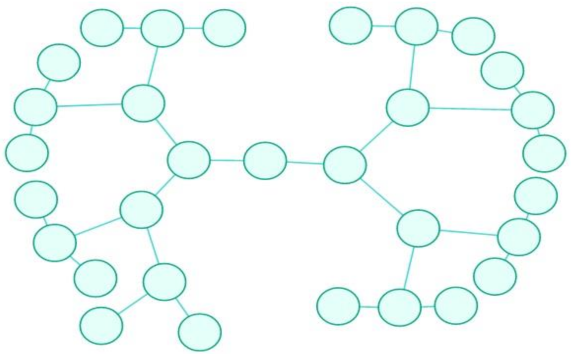
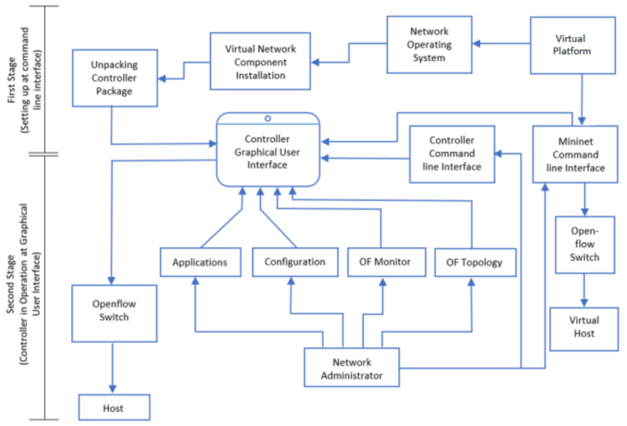


Fig. 17: Software Defined Networking Campus Area Network

With the command $ sudo mn --controller=remote,ip=10.10.9.86 --topo=tree,5 –mac, a network technician does not need to move from switch to switch to detect errors. The platform now shows only the switches on the network as requested by the Administrator. In the figure, SDN indicates and shows where an error is on the network.

A. Stages in Creating and Implementing HP VAN SDN Controller Creation

Figure 18 show the two stages when using the Debian package.



First Stage (Setting up at command line interface)
Second Stage (Controller in Operation at Graphical User Interface)
Unpacking Controller Package
Virtual Network Component Installation
Network Operating System
Virtual Platform
Controller Graphical User Interface
Controller Command line Interface
Mininet Command line Interface
Open-flow Switch
Applications
Configuration
OF Monitor
OF Topology
Openflow Switch
Virtual Host
Network Administrator
Host

Fig. 18 Stages in Creating and Implementing Software Defined Networking with A Debian Package

In this figure, the virtual platform is running two operating system at the same time. The first one which is the network operating system is used to install the network component and unpack the controller package (Debian package) to activate the controller's graphical user interface for the network administrator. The second operating system is mininet. It is used to provide virtual OpenFlow switches and host for testing. The first stage is the creation aspect of the controller and the second stage is the implementation aspect.

IV. CONCLUSION

The use of traditional networking is becoming outmoded in this 21st century because of the increase in population (in terms of networking).

Maintenance of traditional network is costly because detecting faults on a network can be very challenging. Technicians will have to move from switch to switch to locate the faults. This is time consuming.

Software defined networking is a new trend in networking that gives network administrators an appetite for their daily work. Network administrators are able to detect faults from the SDN controller on the network as soon as it occurs. It is affordable and efficient. Because the SDN controller has the mechanism to clean the data path for fast data flow, it makes networking intelligent with numerous advantages over the traditional network.

Software defined networking has been created and tested with the use of the controller and mininet in this paper. Ubuntu 14.04 server command line interface has been used to produce the results.

It is reliable and efficient. It controls the network by controlling traffic flow to relax the network operation and ensure network reachability. We therefore recommend the implementation of Software defined networking in all institutions and organizations to have a feel of its technology.

ACKNOWLEDGMENTS

The authors acknowledge the support of the University of cape coast and Valley View University.

LIST OF ACRONYMS

IP -	Internet Protocol
SDN-	Software Defined Networking
API -	Application Protocol Interface
RSTP -	Rapid Spanning Tree Protocol
SFTP -	Safe File Transfer Protocol
SSH -	Secure Shell
OFT -	OpenFlow Topology
OFM -	OpenFlow Monitor

Availability of Data and Materials

Information for this paper was from HPE VAN SDN Controller 2.7.16 guide.

DECLARATION STATEMENT

Funding	No, I did not receive.
Conflicts of Interest	No conflicts of interest to the best of my knowledge.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval and consent to participate with evidence.
Availability of Data and Material	Not relevant.
Authors Contributions	All authors have equal participation in this article.

REFERENCES

1. Liu Z, Xu G, Liu P, Fu X and Liu Y 2019 Energy-Efficient Multi-User Routing in a Software-Defined Multi-Hop Wireless Network, *Future Internet*. 1-5. https://doi.org/10.3390/fi11060133
2. Kumar M B D and Deepa B 2015 Computer Networking: A Survey. *International Journal of Trend in Research and Development*, India. 2 (5) ,126-130.
3. Balaram V S S S, Mukundha C and Bhutada S 2016 Enhancement of Network Administration through Software Defined Networks, IOSR *Journal of Computer Engineering*,Vol 18(1), 30-36.
4. Chen K, Yu X, Lu Y and Wang J 2018 A SDN-Based Hybrid Electrical Optical Architecture, *International Conference on Communication Technology*. 18, 242. https://doi.org/10.1109/ICCT.2018.8600126
5. Stancu A, Vulpe A, Suciu G and Popovici E 2016 Comparison Between Several Open Source Network Configuration Protocol Server Implementations, *Ceragon Networks, Romania*, 137. https://doi.org/10.1109/ICComm.2016.7528212
6. Caicedo C V, Prieto Y, Pezoa J E, Sobarzo S K and Ghani N 2019 A Novel Framework for SDN Teaching and Research: A Chilean University Case Study, *Telecommunication and Network Engineering Education*. 67-69. https://doi.org/10.1109/MCOM.001.1900261
7. Lin Y H, Yang C W, Chuang T C, Liu M and Chang M C 2019 An Integrated Network Monitoring System for SDN VPN, *Asia-Pacific Network Operations and Management Symposium*. 20, 1-4, https://doi.org/10.23919/APNOMS.2019.8892841
8. Petar C, Dragan E and Sanja M C 2018 Implementation of Software-Defined Networks Using Open-Source Environment, *Technical Gazette*, Suppl. 25, 222-230. https://doi.org/10.17559/TV-20160928094756
9. Sanjeev S and Rakesh K J 2016 A Survey on Software Defined Networking: Architecture for Next Generation Network, *Journal of Network and Systems Management*. 25, 321–374. https://doi.org/10.1007/s10922-016-9393-9
10. Ihsan H. A, Deqing Z, Israa T A, Bin Y and Weiming L 2018 Sec SDN-Cloud: Defeating Vulnerable Attacks Through Secure Software-Defined Networks, *Digital Object Identifier*, China. 8292-8294. https://doi.org/10.1109/ACCESS.2018.2797214
11. Chowdhary A, Huang D, Alshamrani A, Kang M, Kim A and Velazquez A 2019 TRUFL: Distributed Trust Management framework in SDN, 1-3. https://doi.org/10.1109/ICC.2019.8761661
12. Colombo C, Lepage F, Kopp R and Gnaedinger E 2019 SHERPA: A SDN Multipath Approach to Eliminate Resilience Impact on Video Streams, *International Conference on Communication Technology*. 18, 1357-1358. https://doi.org/10.1109/ICCT.2018.8600180
13. Comer D and Rastegarnia A 2018 OSDF: An Intent-based Software Defined Network Programming Framework, *Purdue University USA*, 527-529. https://doi.org/10.1109/CCNC.2018.8319173
14. Deng S, Gao X, Lu Z, Li Z and Gao X 2019 DoS vulnerabilities and mitigation strategies in software-defined networks, *Journal of Network and Computer Applications*. 125, 209-213. https://doi.org/10.1016/j.jnca.2018.10.011
15. Demirci S and Sagiroglu S 2019 Optimal placement of virtual network functions in software defined networks: A survey, *Journal of Network and Computer Applications*, Turkey.147, 1-3, https://doi.org/10.1016/j.jnca.2019.102424
16. Dong S, Abbas K and Jain R 2019 A Survey on Distributed Denial of Service (DDoS) Attacks in SDN and Cloud Computing Environments, *IEEE Access*. Vol 7, 80813-80814. https://doi.org/10.1109/ACCESS.2019.2922196
17. Marlese L and Connor C 2019 SDN 101: Networking Foundations Guide, *SDxCenter*, LLC, USA,
18. Anadiotis A C, Galluccio L, Milardoc S, Morabito G and Palazzo S 2019 SD-WISE: A Software-Defined WIreless SEnsor network Computer Networks, *Elsevier*. 159, 84-85. https://doi.org/10.1016/j.comnet.2019.04.029
19. Coronado E, Garriga E T, Villalón J, Garrido A, Goratti L and Riggio R 2019 SDN@Play: Software-Defined Multicasting in Enterprise WLANs, *Network and Service Management*. 86-87. https://doi.org/10.1109/MCOM.2019.1800502
20. Marcos C, Mauricio S and Ryan U 2019 SDN Advantages for Ethernet-Based Control, *Schweitzer Engineering Laboratories*, Inc.
21. Hantouti H, Benamar N, Taleb T and Laghrissi A 2019 A. Traffic Steering for Service Function Chaining, *IEEE Communications Surveys & Tutorials*, 21(1), 487-490. https://doi.org/10.1109/COMST.2018.2862404
22. Li Z, Lu Z, Deng S and Gao X 2019 A Self-Adaptive Virtual Network Embedding Algorithm Based on Software-Defined Networks, *Transactions on Network and Service Management*, 16(1), 362-364. https://doi.org/10.1109/TNSM.2018.2876789
23. Kim Y, Ahn S, Thang N C, Choi D and Park M 2019 ARP Poisoning attack Detection based on ARP Update state in Software-Defined Network, *Soongsil University Seoul*, Korea, ICOIN,366-370. https://doi.org/10.1109/ICOIN.2019.8718158

24. Chen L, Abdellatif S, Tegueu A F S and Gayraud T 2019 Embedding and re-embedding of virtual links in software-defined multi-radio multi-channel multi-hop wireless networks, *Elsevier*. 145, 161-162. https://doi.org/10.1016/j.comcom.2019.06.012
25. Go S J Y, Festin C A M and Tan W M 2019 An SDN-based framework for improving the performance of underprovisioned IP Video Surveillance networks. *Journal of Network and Computer Applications*, *Elsevier Ltd*. '132, pp49-56. https://doi.org/10.1016/j.jnca.2019.01.026
26. Halloush R, Halloush M, Almalkawi I, Musa A and Salameh H B A 2019 rate-maximizing spectrum sharing algorithm for cognitive radio networks with generic resource constraints, *Special Issue Article, John Wiley & Sons, Ltd*,1-3. https://doi.org/10.1002/ett.3602
27. Alharbi T and Portmann M 2019 The (In)Security of Virtualization in Software Defined Networks, *IEEE Access*, 7, 66585. https://doi.org/10.1109/ACCESS.2019.2918101
28. Packard H 2016 HPE VAN SDN Controller 2.7.16 guide, *Hewlett Packard Development LP*, 9-23.
29. Assefa B G and Özkasap Ö 2019 A survey of energy efficiency in SDN: Software-based methods and optimization models, *Journal of Network and Computer Applications*. 2, 128-129. https://doi.org/10.1016/j.jnca.2019.04.001
30. Cai H, Deng J, Chen S, Wang X, Pack S, and Han Z 2019 Improved Flow Awareness by Spatio-Temporal Collaborative Sampling in Software Defined Networks, *Shanghai, China*, 1-6. https://doi.org/10.1109/ICC.2019.8762093
31. Gotoa Y, Ng B, Seahb W K G and Takahashi Y 2019 Queueing analysis of software defined network with realistic OpenFlow–based switch model, *Computer Networks*, 163, 1-3. https://doi.org/10.1016/j.comnet.2019.106892
32. Arora, S., & Dalal, Dr. S. (2019). DDoS Attacks Simulation in Cloud Computing Environment. In International Journal of Innovative Technology and Exploring Engineering (Vol. 9, Issue 1, pp. 414–417). https://doi.org/10.35940/ijitee.a4163.119119
33. Jain, S., Choudhary, *Anupam, Sharma, A., & Patel, B. (2019). Privacy-Preserving Ddos Attack Detection using Cross-Domain: Challenges and Research. In International Journal of Engineering and Advanced Technology (Vol. 9, Issue 1, pp. 2373–2382). https://doi.org/10.35940/ijeat.f9201.109119
34. Jain, S., Choudhary, *Anupam, Sharma, A., & Patel, B. (2019). Privacy-Preserving Ddos Attack Detection using Cross-Domain: Challenges and Research. In International Journal of Engineering and Advanced Technology (Vol. 9, Issue 1, pp. 2373–2382). https://doi.org/10.35940/ijeat.f9201.109119

AUTHORS PROFILE

Joseph Kofi Mensah is an Information Technology (IT) technician at Department of Mathematics computer laboratory, School of physical Sciences, College of Agriculture and Natural Sciences, University of Cape Coast. He holds a first degree in B. Ed Information Technology from Valley View University with about seventeen years working experience. He is an accomplished educator with demonstrated ability to work effectively, highly self-motivated and result driven person with a demonstrated passion for efficiency and mindset for commitment to work. He enjoys undertaking tasks that pose a challenge to him and works in a very challenging environment. His interest is computer networking and currently on his master's degree program.

Joseph Ahor Abandoh-Sam holds a Cisco CyberOps Associate Certificate from the Cisco Networking Academy and is pursuing a PhD in Computer Science at the University of Ghana. He also has a master's degree in software engineering from Andrews University and a Bachelors in Computer Science from Valley View University, Ghana. He has further enriched his knowledge through various certifications, including a Udacity Nano degree in AI Programming with Python and a Certificate in Cybersecurity from MIT Professional Education. Josephs work experience includes roles as the Dean of the Faculty of Science, Head of the Department of Computer Sciences Engineering, and Senior Lecturer at Valley View University. He facilitated the teaching and learning of various subject areas, such as software engineering, systems analysis, and data communication. He also contributed to developing the BSc Business Information Systems program at Palm Institute Manya-Jorpa and led the curriculum development team for the BSc Computer Science program.

Henry Amankwah was born in Accra, Ghana on 4th July 1969. Attended Cambridge Preparatory School and Wesley Grammar School, all in Accra, Ghana. Enrolled at the University of Cape Coast in 1990 to study Mathematics (BSc) and Mathematics Education (BEd). Completed the program successfully in the year 1994. Was a teaching assistant at the Department of Mathematics and Statistics, University of Cape Coast. He did his National Service period in 1994/95 and was a Senior Research Assistant at the same department in 1995. Now holds PhD degree in Applied Mathematics from Linkoping University, Sweden. Currently, teaching various courses of Mathematics at the University of Cape Coast. His interest is Modelling in areas of Applied Mathematics and Scientific Computing.

Emmanuel Tchouchu is a lecturer at the Department of Management, School of Business, University of Cape Coast, Ghana. He holds a Doctor of Philosophy (PhD) degree in Public Administration and Policy Management from the University of Ghana, Legon. He has over 13 years of teaching experience at the Valley View University, Accra Institute of Technology (AIT), and currently at University of Cape Coast, all in Ghana. He possesses very good research skills and analytical abilities with specific interest in the following areas: Public Administration, Public Policy, Management and Organizational Behavior, Human Resource Management, and innovative technology. He is a reviewer to a couple of international journals.