Statement and Confirmation of Own Work

A signed copy of this form must be submitted with every assignment.
If the statement is missing your work may not be marked.

Student Declaration

I confirm the following details:

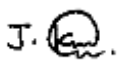
Student Name:	Johnson Kenisha Corera
Student ID Number:	c23020001@cicra.edu.lk
Qualification:	Bachelor in Cyber Security
Unit:	SIT325 Advanced Network Security
Centre:	CICRA Campus
Word Count:	<Word Count>
I have read and understood both *Deakin Academic Misconduct Policy* and the *Referencing and Bibliographies* document. To the best of my knowledge my work has been accurately referenced and all sources cited correctly. I confirm that I have not exceeded the stipulated word limit by more than 10%. I confirm that this is my own work and that I have not colluded or plagiarized any part of it.	
Candidate Signature:	
Date:	03/12/2024

Table of Content

Table of Figures

Table of Acronyms

RSDN	Research Software-Defined Networking
ONOS	Open Network Operating Systems
SDN	Software-Defined Network
TLS	Transport Layer Security

Introduction

- In the context of the Software-Defined Networking (SDN), the main control point in the management of the network is an ONOS (Open Network Operating System) controller. As part of SIT325: Advanced Network Security, and this assignment addresses the following topic: integrating Mininet for instance the widely used SSN emulator for RSDN research with ONOS. The primary purpose of this work is to show that ONOS should be capable of being installed and configured and connected to mininet. Moreover, communications service providers cannot do without ONOS because this system is aimed at providing networks that are high-performance, high-available and scalable. With help of integration of Mininet with ONOS, the manipulation of network topologies is possible and thereby the possibilities as well as the functioning of SDN can be outlined.

- At first, there was a build tool called Bazel which was necessary in order to compile and run ONOS. But due to the issues that appeared in Bazel, instead a different approach had been implemented which was based on Docker. Fortunately, Docker makes the setup process much simpler and the deployment and installation processes are made easier by the provision of readymade ONOS image. The objectives I set were to ensure network was well managed and visualized, to test ONOS interface and to be able to successfully install ONOS in Mininet. From the theoretical perspective, this experience expands knowledge of SDN and its applications while from the practical side it improves the understanding of the setup and basic management of SDN controller.

Part A

- My plan was to deploy and integrate the ONOS Controller with the Minitek. Software Defined Networking or SDN mainly operates on ONOS as the primary control network administration point. In this regard, ONOS was built with Bazel as its build tool to identify its compatibility with Mininet.

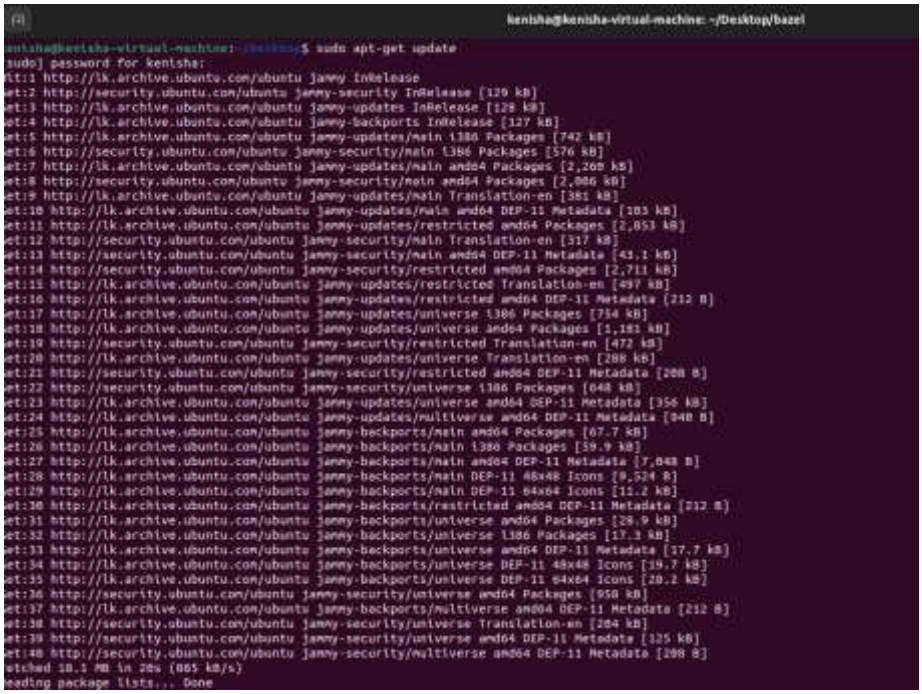


Figure 01: sudo update

- I set up every required dependent package.

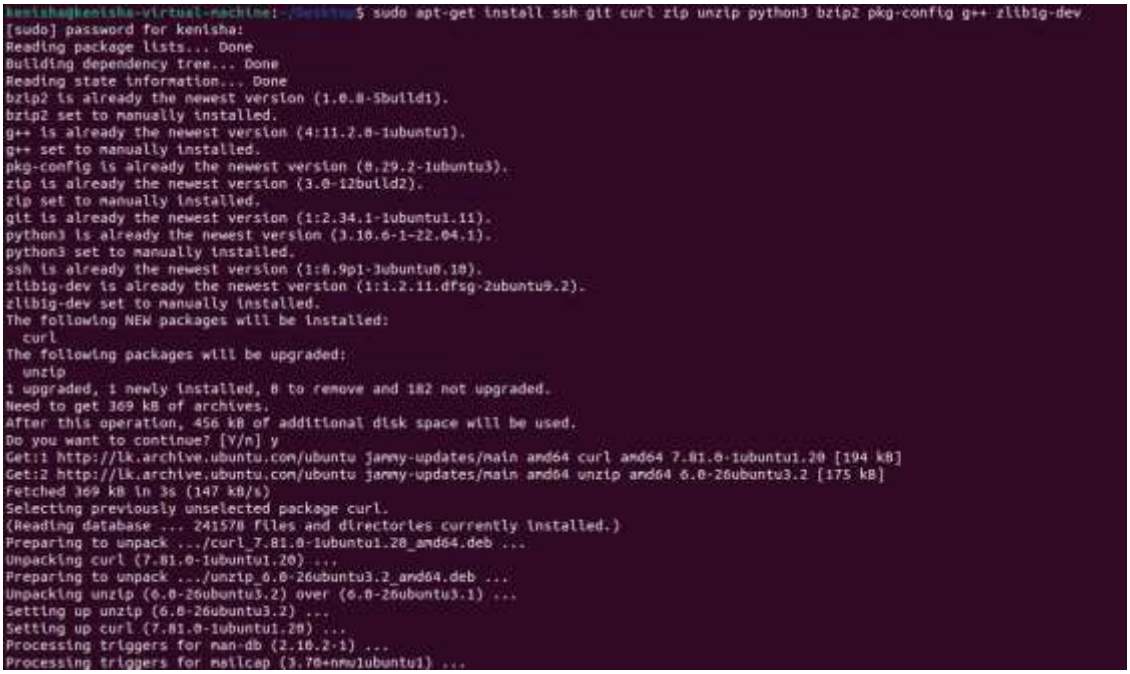


Figure 02: Installation

- Next, I used the command git clone https:To meet the task's requirements and clone Bazel go to https://github.com/bazelbuild/bazel as the installer, I made the program start running after its installation.

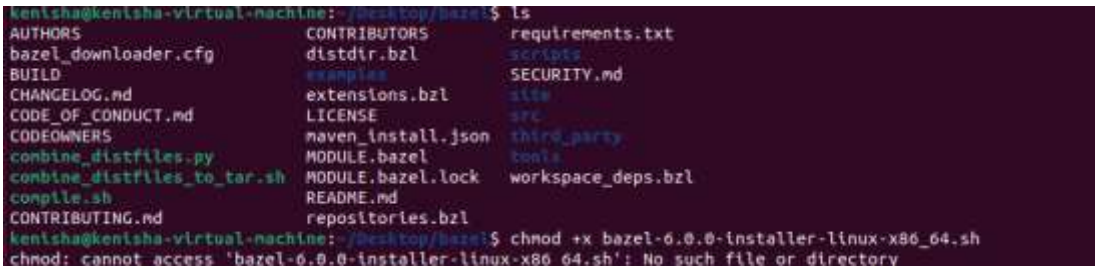


Figure 03: Directory of bazel

Figure 04: Running chmod

- After that, I attempted to build it with Bazel once onos repository was cloned into my machine.

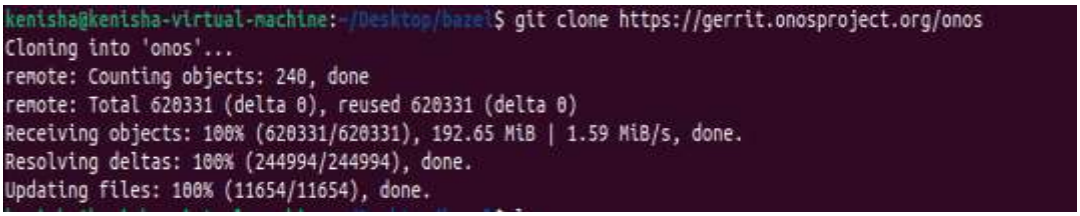


Figure 05: onos cloned

- An effort was made to create an onos environment by using Bazel which was unsuccessful. Alas, such problems occurred at the stage of constructing the procedure. The build failed on attribution of wrong configuration and other errors dealing with missing dependencies. I tried to address these improper work issues, but I failed to work out an ideal solution for the Bazel build issue.

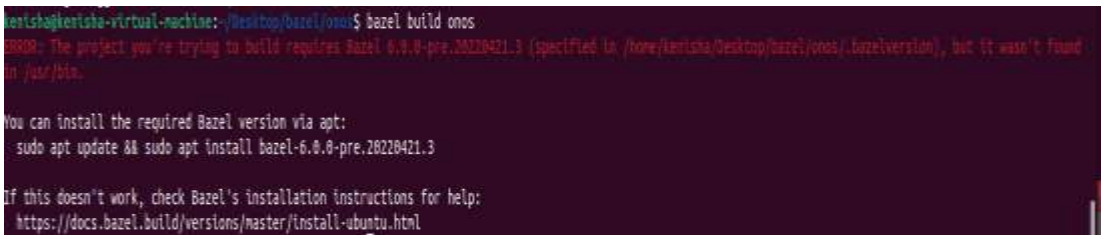


Figure 06: Error in onos

- I preferred to use a different approach since the earlier procedures presented some difficulties when being implemented. The decision was made to install ONOS using a docker based setup. I started by installing Docker and updating the package list using the commands below:

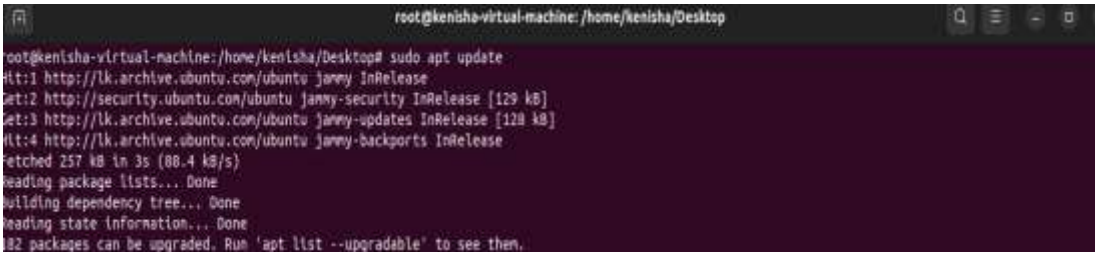


Figure 07: Updating sudo

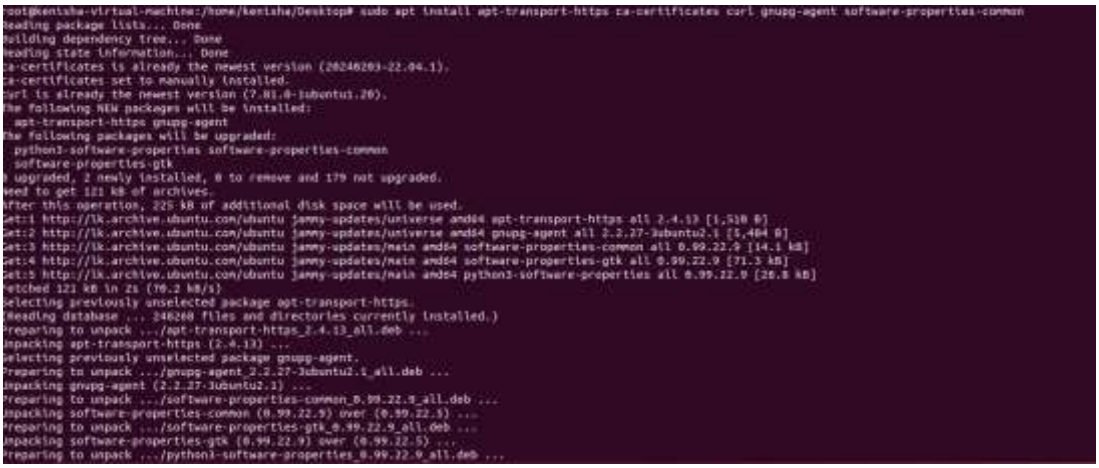


Figure 08: Installation

Figure 09: docker Installation

- I then downloaded and executed the ONOS Docker image.

```
root@kenisha-virtual-machine:/home/kenisha/Desktop# docker pull onosproject/onos:2.7.0
2.7.0: Pulling from onosproject/onos
7595c8c21622: Pull complete
d13af8ca898f: Pull complete
70799171ddba: Pull complete
b6c12202c5ef: Pull complete
a3caae5bc1ad: Pull complete
ce1773b9f958: Pull complete
1f74a437590f: Pull complete
Digest: sha256:bc844aaafd64e6b3834c7043bad83fa1eecc6afb6984e9b4887c12e1a307a7c2
Status: Downloaded newer image for onosproject/onos:2.7.0
docker.io/onosproject/onos:2.7.0
```

Figure 10: Pulling docker image

```
root@kenisha-virtual-machine:/home/kenisha/Desktop# docker run -t -d -p 8181:8181 -p 8101:8101 -p 5005:5005 -p 830:830 --name onos onosproject/onos:2.7.0
d877894947e6180e2bb92a412e3d7193622b3be831c5784bdf1e9f57f5689889
```

Figure 11: docker running

- Finally, I used the URL http:Such that I could now get the ONOS UI on my web browser, from http://localhost:8181/onos/ui/login.html. I then typed the password which was rocks and the username, onos.

Figure 12: login page of onos

- To be certain that the change was effective, I logged into ONOS user interface.

- Once I logged in I enabled a few programs that were needed for a PC with a small internet connection and such.

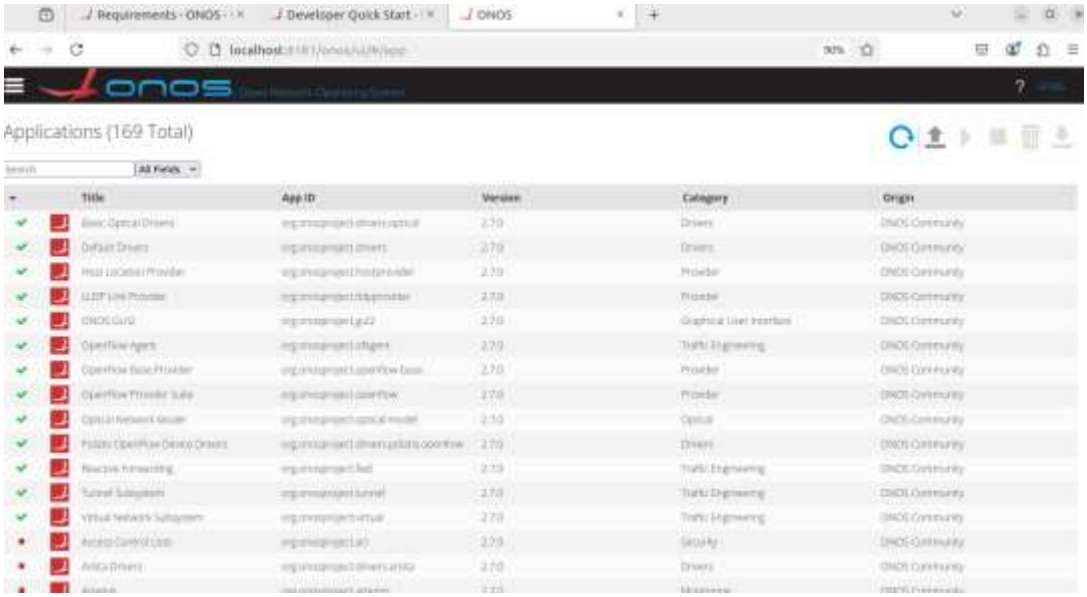


Figure 13: Applications that are enabled

- I then created a mininet with an ONOS controller, verified the topology, and then run dotted line pinging all the hosts. Of course it will reflect my own set up and the port and IP values need to be adjusted accordingly.

Figure 14: Running mininet

- The OS shown all devices connected to the controller by the time I had pinged all the hosts. This connected the devices that were linked and the topology.

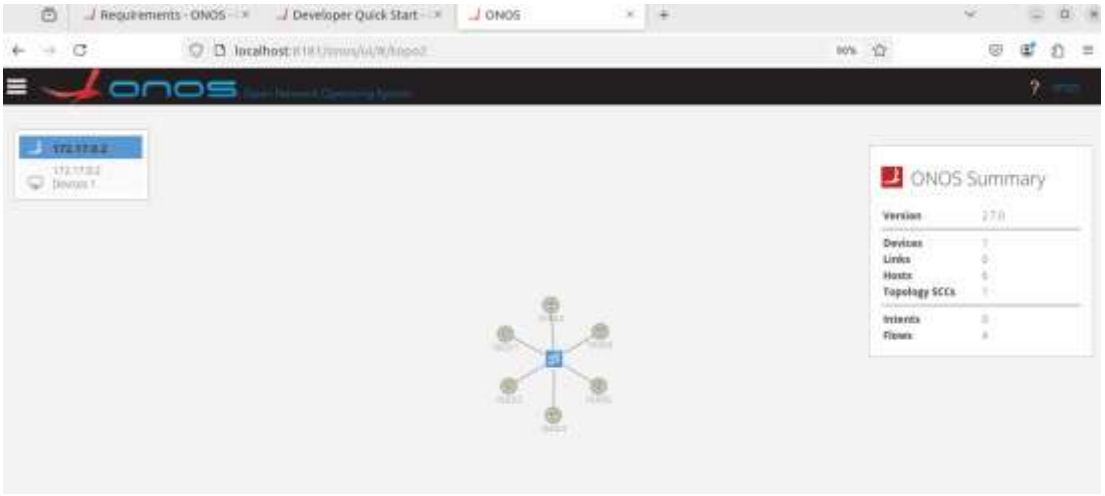


Figure 15: Network Topology

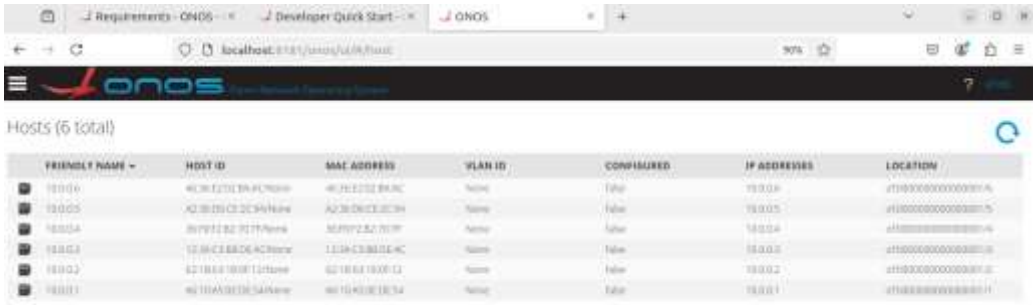


Figure 16: devices that are linked

Part B

Q1. What do you mean by the secure channel in SDN, illustrate and explain.

- A secure channel as SDN is a mechanism that aims to provide the required level of protection in software-defined networks, while the establishment of the secure channel means the definition of a connection. Specialized and protected link that an SDN controller employs to interconnect to network gear, including switches and routers. Besides, the safe route ensures that the In between these components, data encrypted and/or protected from outside intrusion. (Kreutz, 2018)

- The secure channel can mainly be employed to ensure that the actual data being transmitted is confidential, integrated and valid. It protects the controller from third parties because such parties can interfere with the messages that the network devises transmit and receive. Based on the idea of making the channel secure from interception by any third party, the transport layer security TLS is an example of a secure channel.

- The safe communication channel for instance guarantees that messages such as flow entry and configuration commands reach the switches from the controller safely when implementing OpenFlowProtocol under SDN basement. As these messages define how network traffic should be handled, any modifications to these messages would definitely be adversarial to the network. This makes them crucial.

- Since the network control plane is only accessible to authorized controllers and switches, the safe channel is also beneficial for determining the engaging parties. This authentication that is done only allows authorized devices to connect to the network and this goes a long way in eliminating a number of nuisances that may be caused by unauthorized devices. Typically, for the network substrate to be protected and for the control and management, job to be accomplished, often a secure channel is established in SDN.

Q2. What is in-band or out-of-band communication in SDN?

- There are two types of communication that occur between the SDN controller and network devices in Software-Defined Networking (SDN): In Band and Out of Band. Consequently, there is a relation as to how both strategies impact on the performance and management of the network. (Ahmad, 2020)

- One form of communication referred to as “in-band” arises when both the control signals and the payload signals Transmission that occurs at the same period or occupy the same channel in the nervous system of the body. In this case the controller of the occurrence of and distributes the control information. Events in the general substrate undergo, including the one that it uses to transport user traffic. The first advantage is that in-band communication can be deployed at very little cost because it operates on existing hardware.

- Especially, such files may be large or contain expressions that take time to be computed at the end or have stacking facet calls that, if too large, failed; or are created based on low quality sources and contain code that is potentially destructive. Heavy data traffic quantities for instance can jam down control messages that could include network management slowing or even stopping network control.

- Rather, out of band occurs when control traffic hits the channels or network paths and deviates from data traffic all together. It is made more reliable through opened connections between the SDN controller and the network devices not limiting the user data throughputs. Not only does out-of-band communication send control messages in addition to data traffic, but the separation also enhances security since control messages cannot be tampered with or intercepted. However, there is a trade-off: you need infrastructure that is additional in which the cost and the difficulty of implementation are higher.

Q3. In SDN, one of the fundamental actions of the OpenFlow switch is to forward packets to the controller for exception handling. PACKET_IN is generally sent via the secure channel to the controller for handing off this exception processing. Many times, buffer ID is communicated with the PACKET_IN message. Please explain via an example to discuss why BUFFER ID field in the PACKET_IN message is required to send to controller.

- This means that the BUFFER_ID field exists in the PACKET_IN message: in other words, effective packet management is incapable without it. The last member of the structure, the BUFFER_ID, is the identification number of the buffer where the specified packet is to be stored which the switch reserves for itself. It is important to do so for two principal reasons: This is essential for two main reasons:

  - Reducing Packet Loss: The switch on analyzing a packet which it has learned for the first time forwards the packet to the buffer and sends the PACKET_IN message along with the BUFFER_ID to the controller. It also makes it possible for the switch that received the packet to forward it to the controller to give the new instructions on where the packet should go next. However if the packet has to be transmitted and not disappear in the process then the controller can then use the BUFFER_ID to get a copy of it from the switch's buffer. (Stephenson, 2024)
  - Effective Use of Resources: When the switch has the BUFFER_ID, it does not forward the whole put payload to the controller thus minimizing the band width consumed and the load exerted on the controller. Actually, however, the controller applies the get() method of the buffered socket to form the packet using that of the BUFFER_ID, thus accelerating and diversifying the process. For example, a switch will drop a packet and send the controller a PACKET_IN message along with the BUFFER_ID at the onset if it forwards a packet to and unknown destination. To decide if according to the recently learnt flow rules the controller should forward or drop the packet, it analyzes the packet using the BUFFER_ID.

Q4. We have discussed potential drawbacks and limitations of SDN (week 4 lecture). Were the arguments convincing, or do you feel that these SDN limitations are debilitating for the adoption of SDN? What other drawbacks might there be that we have not mentioned?

- The SDN concepts bring a number of advantages, such as the centralized management of the network employed and the obtained network flexibility. However, there are a number of restrictions and disadvantage which has been pointed out as may be hindering its use.
    - **Security Issues:** SDN offers new security challenges. Since the SDN architecture is centralized, the controller becomes a new single point of failure. It has been identified that an attacker can take full control of the given network if the controller is compromised. Further, the communication between the controller and switches, if not protected, can be under assaults. A must consider is the controller and control messages security. (Astuto, 2014)
    - **Problems with Scalability:** Hyper scalability is a major drawback of SDN. Since the scalability of the network means that there is added pressure on the SDN controller, the performance may stutter. This may prove cumbersome especially in large-scale networks where it will be tasksing Control Plane with handling a lot of traffic and making relatively fast decisions. This scaling problem may become an issue which blemishes the reliability and effectiveness of the rooted network.
    - In addition to the foregoing issues, there could be other negative effects that could rarely be spoken about. For example overdependence on a specific controller puts the prestige of the company at risk since it acts as a one-stop-shop. Moreover, what would be viewed by many organizations as a major concern is that shifting to the SDN may require massive investment in new technology and training. (McKeown, 2008)
- Consequently, SDN has many beneficial attributes; however, it also has a number of detrimental ones such as scalability problems, security vulnerabilities, complexity of interoperation, and susceptible to single points of failures. That is why these problems need to be addressed in order for SDN adoption and implementation to work effectively.

Conclusion

- Mininet in combination with ONOS also proves that Software-Defined Networking (SDN) has the possibility to radically redefine the network process and its management. In this paper, the author has brought out the capability of SDN with efficient, effective, and sustainable networks through the implementation, configuration, and its interaction with ONOS and Mininet facilities. The setup process is kept simple and fast by actually deploying SDN using Docker, which in turn assures that people understand the architecture of SDN's and the controllers

- SDN introduce revolutionary advantages like easier topologies' management, network programmability and being controlled from the center. This investigation has however also exposed inherent challenges such as scalability issue, insecurity and centralization risks. However, such difficulties may be off-putting to some organizations, but with proper addressing with strong security protocols, better controller design and careful planning, full potentials of SDN can be achieved.

- Finally, this work underlines the importance of ONOS as an essential core for the SDN ecosystems and the need for further advancement to overcome the challenges and can ensure that SDN is a viable and effective solution for today's network environments.

References

1. Software-Defined Networking: A Comprehensive survey
   https://ieeexplore.ieee.org/document/6994333

2. Security in Software Defined Networks: A survey
   https://ieeexplore.ieee.org/document/7226783

3. Stephenson, R. (2024) 'Understanding Out-of-Band Communication
   Understanding Out-of-Band Communication and its' Role in Cyber Attack Response

4. McKeown, N. et al. (2008b) 'OpenFlow,' ACM SIGCOMM Computer Communication
   https://www.researchgate.net/publication/220195143_OpenFlow_Enabling_innovation_in_campus_networks

5. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks'
   https://www.researchgate.net/publication/260064541_A_Survey_of_Software-Defined_Networking_Past_Present_and_Future_of_Programmable_Networks