CAREER EPISODE 1

INTRODUCTION

CE 1.1 During my Bachelor of Engineering (Telecommunications) from the [redacted] of Engineering and Technology, I worked on a project "Testbed development for Software Defined Networking (SDN) to know its deployment feasibility on Access layer", for my course Computer Communication and Networks in the 6th Semester. I started this project in [redacted] and completed it in [redacted]. In this project, I explored the implementation of SDN based testbed using a Zodiac-FX OpenFlow switch and assessed its performance against the existing network architecture of FEECE at my University. I will explain my work on this project in this episode, along with the results I achieved.

BACKGROUND

CE 1.2 During 6th semester of my degree, as part of my course, Computer Communication and Networks, I worked on a project to know what I learned from theory and how I could apply it to a practical networking scenario. This project focused on enhancing network scalability, performance and manageability using SDN. I explored the limitations of existing network architectures and addressed them by implementing SDN, which I learnt separates the control plane from the data plane to simplify network management and regulatory policies. I used the Zodiac-FX Open Flow switch to develop a testbed and tested the feasibility of SDN at the access layer. I also simulated the existing network of the FEECE at my University and compared its performance with an SDN-based network. This simulation and practical implementation, allowed me to learn that the SDN significantly improves network performance, flexibility and centralizes control compared to the existing architecture.

CE 1.3 I conducted literature review on various aspects of SDN, Network Functions Virtualization (NFV) and their implementation in modern networking. I studied how SDN separates control as well as data planes, allowing centralized management of network and simplifying programmability, which improves network performance and reduces complexity. I studied SDN's architecture, with a focus on the interaction between Northbound and Southbound APIs and how controllers manage network flows using OpenFlow protocols. I also explored several OpenFlow controllers including HPE VAN, RYU and POX and their flexibility in packet processing. I also studied NFV which I learnt allows network functions like load balancers and routers to be virtualized and the relationship between SDN and NFV. I reviewed the IPERF tool for measuring TCP/ UDP performance that provided information on network bandwidth, delay, jitter and data loss. I also studied the contributions of academic institutions like Stanford and UC Berkeley and networking vendors like Cisco in advancing SDN. I further studied SDN's implementation on the access layer, which I learned offers benefits like

scalability, flexibility and efficient resource utilization. This literature review also included the growing need for SDN in response to changing traffic patterns in data centers, the rise of cloud services and the increasing complexity of network management, emphasizing ability of SDN to enhance automation, scalability and security.

CE 1.4 I planned my project by first identifying its objectives and technical scope, then formed a team by working with 2 of my fellow students who shared similar interests in networking technologies. Taking on the role of their leader, I assigned responsibilities based on their strengths and checked that everyone had a clear understanding of the assigned tasks. I held regular meetings with both my team and project Supervisor to discuss progress, address any issues and gather valuable feedback. I guided my team through the implementation phase, maintained documentation and checked that we were aligned with our goals and deadlines. As the project progressed, I compiled my findings, prepared a project report and created a presentation to present my work to the class. I explained the main concepts and results to make them understand the significance of my project.

PROJECT REPORTING HIERARCHY

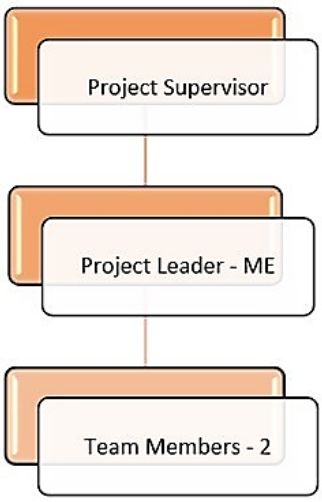

Figure 1: Hierarchy

PERSONAL ENGINEERING ACTIVITIES

CE 1.5 Through my project, I aimed to address the growing need for scalable and high-performance networks. I noticed that existing hardware-based networks, though once reliable have become static, inflexible and lack the agility required to support evolving technologies like cloud computing, mobile computing and IoT. Using my knowledge of telecom systems, I studied how existing networks operate using control, data and management planes and how their distributed architecture poses limitations in dynamic IT environments of today. I planned and executed the development of SDN testbed kit, with the aim to analyze the performance of SDN when deployed on the access layer. I conducted a hardware deployment using Zodiac-

FX OpenFlow switches and compared their performance against existing switches to understand enhancements in scalability, stability, throughput and efficiency. I applied my understanding of telecom protocols and networking principles to configure the SDN setup, bind it to several controllers and monitor flows using tools like Wireshark. I also compared various OpenFlow controllers to assess their role in centralizing control and simplifying network management. I also extended the SDN deployment to the FEECE faculty to show real-world feasibility, which showed my ability to translate my theoretical telecom concepts into practical solutions.

CE 1.6 I designed and implemented a methodology that involved both simulations and hardware-based work for developing an SDN testbed kit. I started with the simulation phase, where I focused on creating virtual environments to match the real networking scenarios. I then installed a virtual machine and configured it to support the Ubuntu OS, which provided a stable and Opensource platform for network experimentations. Using my understanding of network virtualization and system architecture, I installed Mininet, which I learnt is a network emulator widely used for SDN testing and various SDN controllers to analyze their behavior in various scenarios. I developed multiple network topologies on Miniedit which included an existing network topology, an SDN based topology and a FEECE departmental network topology integrated with various controllers. The virtual machine environment helped me simulate both hardware and software layers, using x86 and x64 hardware architecture for emulation, and ISO image files for OS. I knew this simulation was important to check SDN functionality before hardware deployment and it helped me evaluate and configure telecom networks in virtual setting.

CE 1.7 Using my knowledge of telecom and network systems further, I set up a simulation environment using Ubuntu, an Opensource OS known for its reliability and widespread use in networking and cloud infrastructures. I selected Ubuntu as I knew about its powerful capabilities in handling automated infrastructure operations and its compatibility with several cloud services including Amazon AWS, Microsoft AZURE, Google Cloud and IBM. I understood the significance of using an OS backed by Canonical, which confirms flexibility, automation, VLAN Mininet which I knew is a widely used network emulation tool that supports the creation of routers, switches, controllers and hosts within a virtual environment. I preferred installing Mininet on Ubuntu using command line tools, as it gave me better control over dependencies and versions. I used commands like sudo apt-get install git and copied the Mininet repository, checked out the relevant version and executed the installation script to set up the environment. It helped me design custom network topologies, test real network programs written in Python and analyze their performance using Wireshark. I was fully aware of limitation of Mininet like its reliance on CPU resources and inability to emulate the

extremely high throughput network, however, it was still highly effective for checking my network configurations and SDN performance at the access layer.

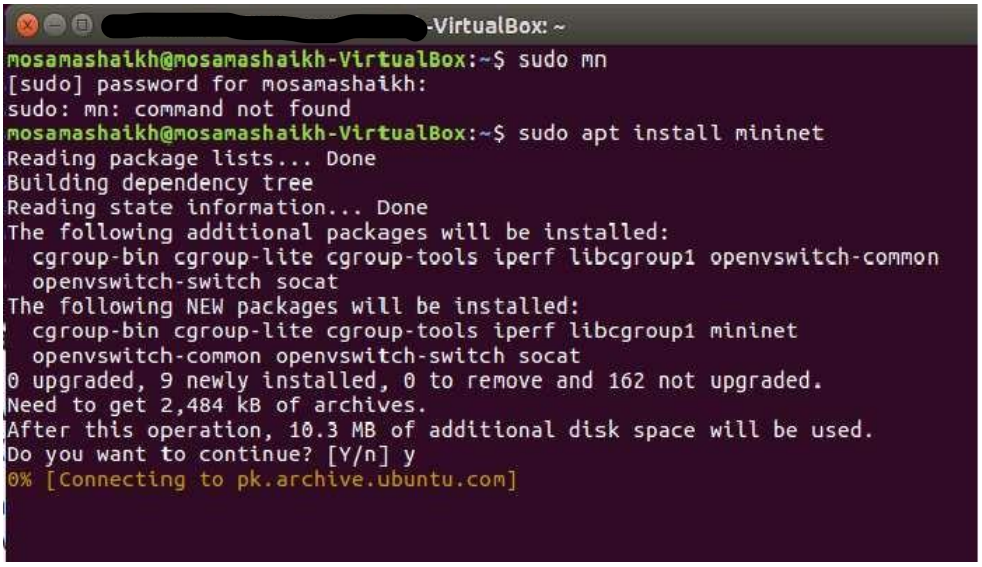

Figure 2: Installation of Mininet through Commands

CE 1.8 After successfully installing Mininet on Ubuntu, I started configuring network topologies both through the command line and using Miniedit which is a GUI based tool added with Mininet. My telecom background helped me understand how Miniedit simplifies topology design by allowing drag and drop functions for switches, routers, controllers, hosts and cabling. I used Miniedit to simulate existing and SDN based network topologies. By understanding the core concept of separating the control plane from the data plane, I was able to simulate fault tolerant SDN design using multiple controllers for resilience which was an important consideration. To automate and program custom SDN behaviors, I used Python programming that helped me write scripts and interact with Mininet's Python API. I knew that Python is widely used in telecom industry for network automation due to its simplicity and powerful libraries. I found it especially useful in creating dynamic network environments and testing real-world Telecom scenarios like traffic redirection, failover testing and load balancing.

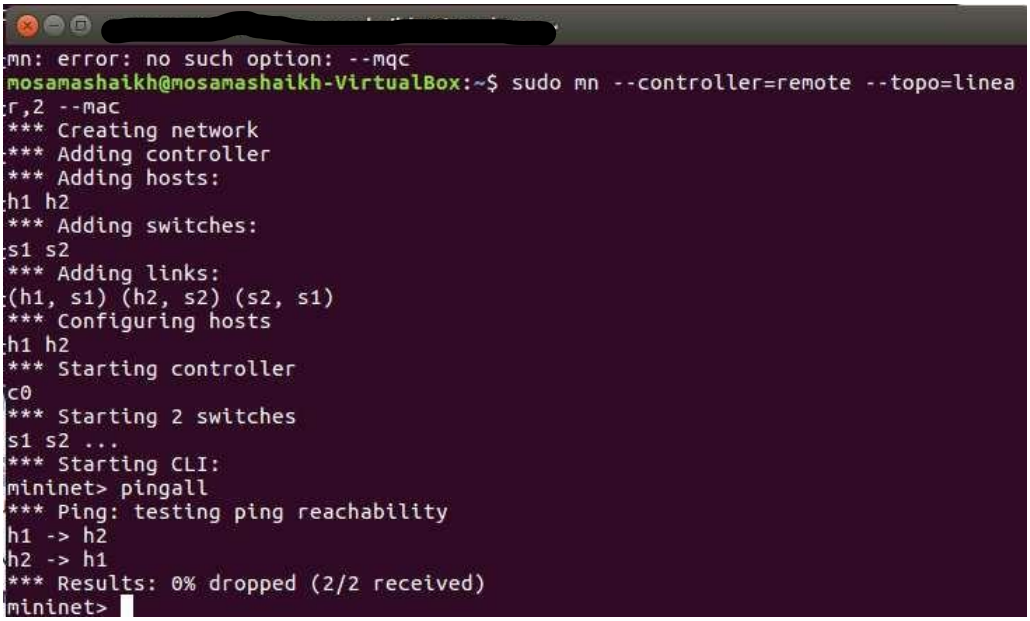

Figure 3: Mininet testing

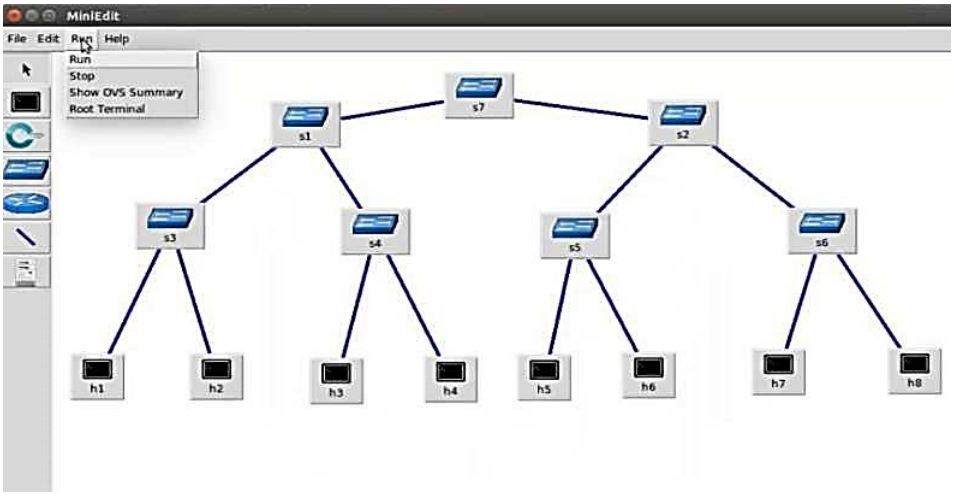

Figure 4: Traditional network on Miniedit

CE 1.9 For remote connectivity and configuration, I used PuTTY which I knew was an essential tool in the telecom domain for secure communication with network devices. It supports various protocols like SSH and Telnet, which I used to remotely access switches, routers and controllers. I also used other components of PuTTY like PSCP and PuTTYgen for secure file transfer. I used WinSCP with PuTTY for secure file transfers between my Windows and Linux environments. It played an important role in moving controller configuration files, logs and scripts during the setup and testing phases. It's SSH based secure transfers provide data integrity, bringing it into line with the security requirements. Further deepening my practical expertise, I installed and configured the HPE VAN SDN controller. After setting up the virtual environment and deploying the controller, I then made necessary configurations like setting the hostname, IP addresses, subnet and gateway settings. I connected the controller to a Zodiac-FX switch using UTP cables. It helped me check controller switch communication. I

noticed the port status transitioned to up, showing successful connection. I also used RYU SDN controller to test SDN applications. I studied its integration with Zodiac-FX by establishing the appropriate physical connections. I found support of RYU for REST, APIs, VLAN configuration and OpenStack integration particularly relevant for modern telecom cloud networks.

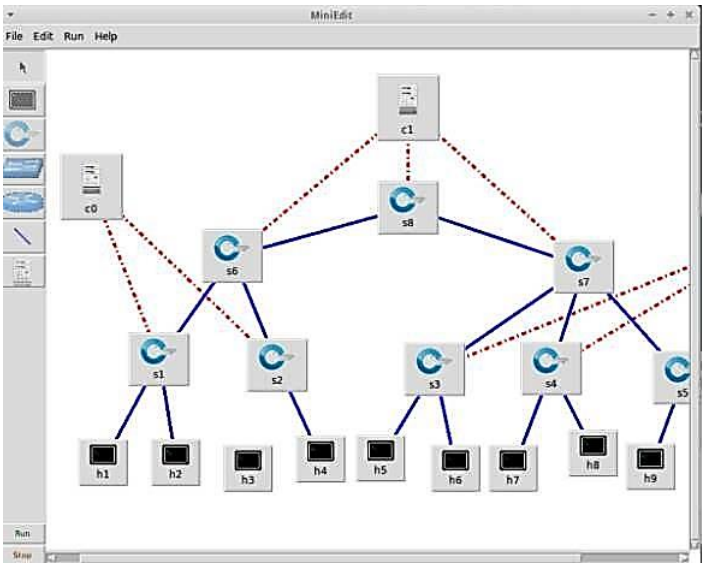

Figure 5: SDN on Miniedit

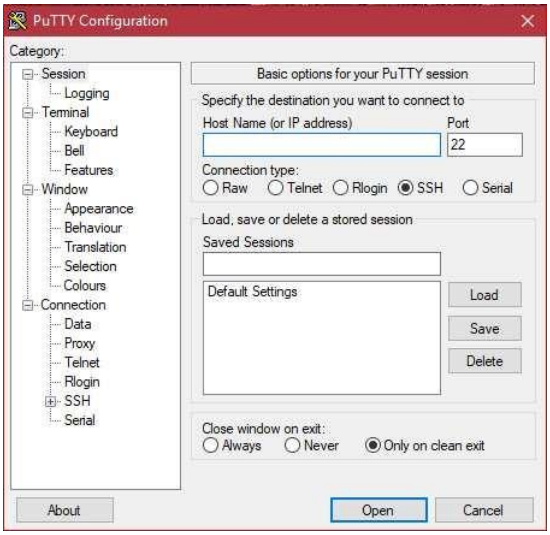

Figure 6: PuTTY

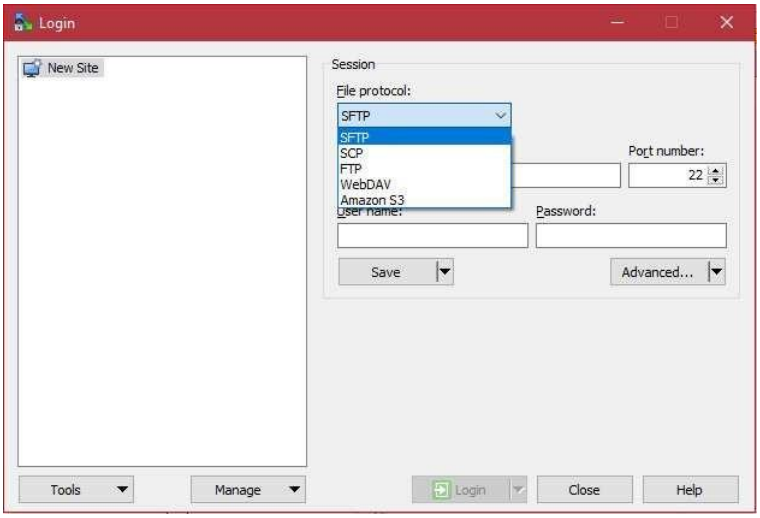

Figure 7: WinSCP interface

CE 1.10 I configured and implemented POX SDN controller and used its APIs to interact with OpenFlow switches. I used its built-in components for host tracking, routing and topology analysis to better understand network behaviors in software defined environment. Through POX, I explored SDN applications like SANE, which I learned treats the network as a file system and Ethane developed at Stanford University, to show centralized security of netwrok. These applications helped me understand how SDN reduces complexity significantly and improves efficiency in network management. I installed the POX controller using a Git-based method and added it with the Zodiac-FX OpenFlow switch. The switch I knew has 4 ports, where port 4 acts as the native uplink to the controller. I connected its port using a UTP cable and monitored the port status to check connectivity. Once the link was established, the status showed 'up' confirming the connection. To further evaluate the SDN performance, I designed and tested a network topology based on the FEECE model using different controllers; HPE VAN, RYU and POX. I started by configuring the Zodiac-FX switch using CLI via PuTTY after finding the correct COM port, I manually set its IP address, subnet, gateway and controller IP for seamless communication with the SDN controllers.

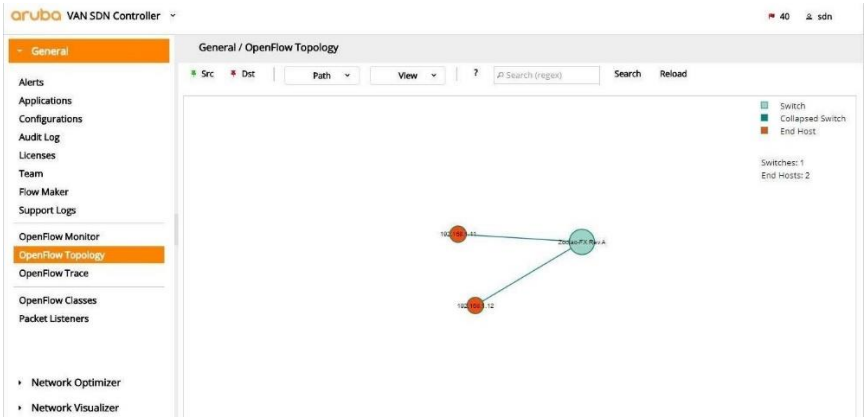

Figure 8: Zodiac FX Topology in HPE Van SDN Controller

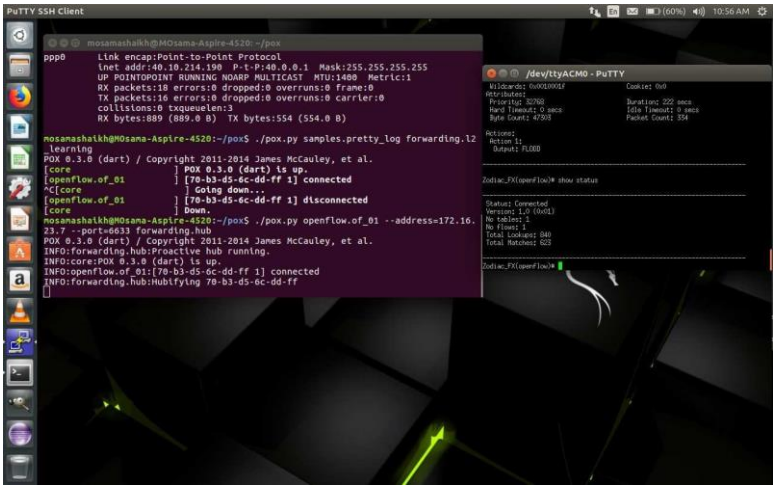

Figure 9: POX and Zodiac-FX Integration

CE 1.11 Once the hardware configuration was complete, I tested the connection by sending pings between the switch and the controller, and successful connectivity helped me run layer 2 and 3 switching applications and add flows, simulating how data traverses in real-world telecom networks. I also gathered OpenFlow statistics using Wireshark, which I used to analyze the packet flow, switching behavior and network efficiency. I gained experience with the Zodiac-FX OpenFlow switch, which enabled me to modify its behavior to suit the specific requirements for firmware development, and used its web interface for intuitive configuration. To extend the physical network, I used TP-Link unmanaged switches to connect multiple Zodiac-FX units to a single controller. This helped me simulate a more complex access network similar to those used in real setups. I powered all devices using 220V to 12V adapters. Finally, I used Wireshark to monitor and trace OpenFlow packet activity in the network. I set network interfaces to promiscuous mode to capture traffic data. It helped me understand how data flows between different network elements and how to troubleshoot issues.

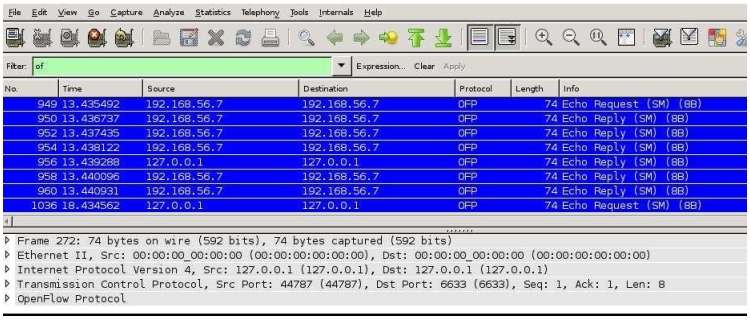

Figure 10: OpenFlow packets tracking on Wireshark

CE 1.12 During the implementation phase of my project, I evaluated and compared various SDNs as mentioned above. This was part of my exploration into optimizing network performance at the access layer, and understanding the practical implications of SDN in a telecom environment. I analyzed the CPU usage of various SDN controllers while they operated in a

live network environment. Using Zodiac-FX kit and an SDN controller's usage tool, I measured and compared the CPU loads. I noticed that the RYU controller constantly used lower CPU resources compared to others, showing superior efficiency. It translated into better scalability and reliability, especially in resource constrained edge devices where efficient processing is important to maintain service quality. I then performed a throughput comparison between SDN and existing networks using the IPERF tool. My findings showed that SDN networks outperform existing ones in terms of throughput. For instance, at a 6ms interval, the SDN setup achieved a throughput of 94.5Mbps compared to 91.5 Mbps in the existing setup.

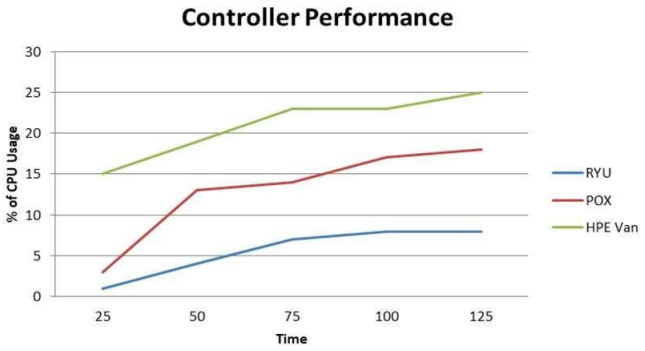

Figure 11: CPU usage v/s time

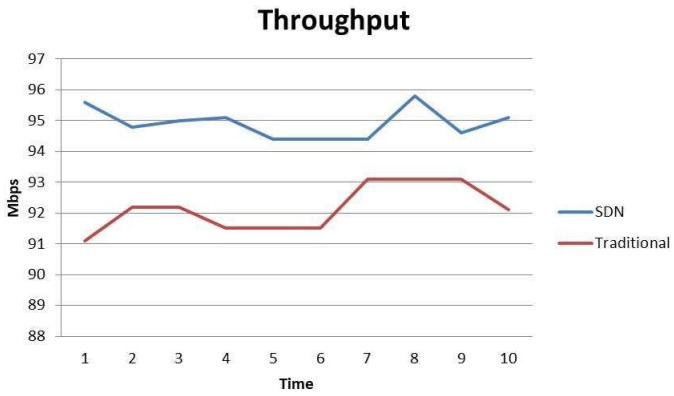

Figure 12: Throughput comparison

CE 1.13 I extended the throughput analysis by examining the impact of user load on throughput. I recreated the FEECE faculty topology and gradually increased the no. of users. I observed that in the existing network, throughput started to decrease as user count rose. At 40 users, the throughput was just 2 Mbps. In comparison, the SDN network maintained higher throughout, reaching 2.9 Mbps at the same user count and sustained better performance even beyond 60 users. I then tested and compared latency in SDN and existing networks by executing ping tests to measure packet delivery times. I learned that SDN switches transferred 10 packets in just 0.6ms, whereas existing switches took between 0.8 to 1.2ms. This reduction in latency showed how SDN enables faster packet forwarding and enhances network responsiveness which are critical factors for telecom applications like voice and real-time video services, where latency can directly affect user experience. All these experiments, helped me validate

the performance gains of SDN over existing networking methods. Using my knowledge, I interpreted these metrics in context of scalability, network efficiency and service quality, all of which I knew were important. My results showed the significance of transitioning towards SDN to meet growing user demands.

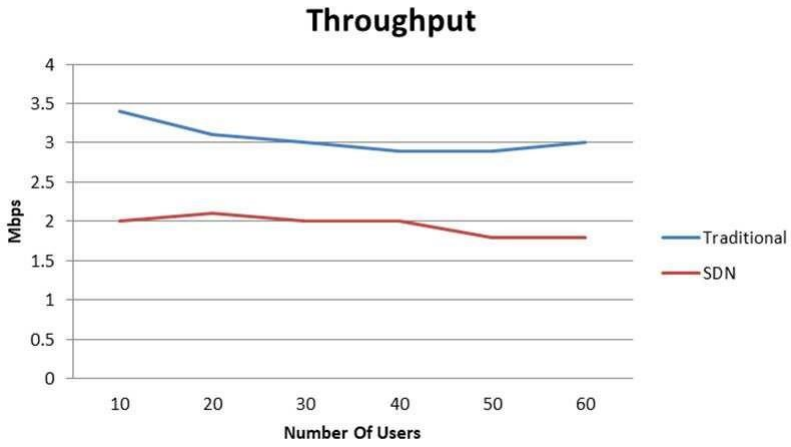

Figure 13: no. of Users V/s Throughput

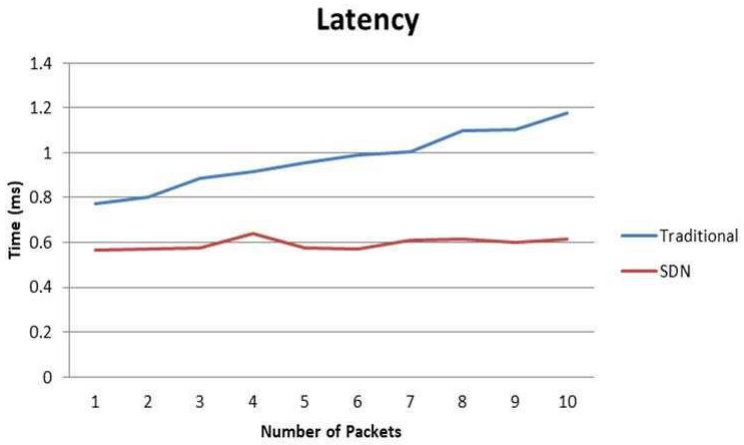

Figure 14: Latency

SUMMARY

CE 1.14 In this project, I developed a testbed for SDN using the Zodiac-FX OpenFlow switch. I tested the feasibility of deploying SDN at access layer and assessed its performance in terms of delay, throughput and scalability. I also simulated the FEECE faculty network of my University using Mininet and experimented with 3 different SDN controllers; HPE VAN, RYU and POX. It helped me compare controller's performance and gain deeper knowledge of practical implementation of SDN in telecom networks. Throughout this project, I adhered to ethical practices by using only authorized tools in controlled test environment without affecting live networks.