Addis Ababa Science and Technology University

College of engineering

Department of Software Engineering

Software Defined System - Case Study

Section B

Group Members Name	ID
Kidist Tadele	ETS0403/12
Melat Gizachew	ETS1020/12
Meron Kedir	ETS1022/12

Submitted To: Instructor Biniam Behailu

April 2024

1) Software-defined networking is a networking method that uses software-based controllers and application programming interfaces (APIs) to communicate with underlying hardware infrastructure (switches and routers) and to direct traffic on a network. With the use of a centralized software-based SDN controller, the network can be controlled and automated via software[1].

SDN consists of three key components: the data plane (switches, forwarding devices), the control plane (SDN controller operates here), and the application layer. The data plane is responsible for forwarding network traffic, while the control plane manages network infrastructure and makes decisions about how network traffic should be handled. The application layer consists of software applications that run on top of the SDN infrastructure[2]. SDN provides centralized network control by separating the control plane and the data plane from both being on routers.

SDN offers several key benefits over traditional networking approaches. For example, SDN allows for more efficient network management, as network administrators can automate many tasks that would otherwise be done manually. SDN also allows for more flexible and customizable network configurations, as network infrastructure can be reconfigured on the fly[2].

Software Defined Networking (SDN) is different from traditional networking approaches in many ways. For example, SDN's key strengths include its programmability, which is absent in non-programmable traditional networks, and its open interface philosophy, which contrasts with closed interfaces in traditional networks. This transformative nature emphasizes agility, centralized management, and openness, redefining the landscape of networking paradigms.

2) **Limited scalability:** Traditional networks rely on physical hardware devices, which can be expensive and time-consuming to scale up as the network grows. Adding new users, devices, or applications often requires manual configuration of each switch and router. This becomes a cumbersome and error-prone process, hindering ABC Corporation's ability to adapt to changing demands.

Network management and control: Traditional networks lack a central point of control. Each device requires individual configuration and management, making it cumbersome to monitor the network's overall health, troubleshoot issues, or implement changes across the entire system.

Complex network configurations: Traditional networks rely on manual configuration of individual switches, routers, and firewalls. As the network grows and becomes more complex, managing these configurations becomes a cumbersome task. Imagine a maze of interconnected roads with each intersection requiring a handwritten traffic sign. Modifying a single route can mean updating signs at multiple locations, a time-consuming and error-prone process.

Lack of flexibility: Traditional networks are not very adaptable. Adding new devices, services, or security policies often requires manual configuration changes on individual devices. This inflexibility makes it difficult for ABC Corporation to respond quickly to changing business requirements or security threats.

3) a. **SDN Controller**: It is the central component of an SDN architecture. It's the brain of the operation, responsible for configuring, managing, and optimizing the entire network. By dynamically adjusting network behavior based on real-time data, the controller optimizes traffic flow, improves resource utilization, and ultimately delivers a more agile, efficient, and secure network.

b. **OpenFlow Protocol**: serves as the communication language between the SDN controller and the SDN-enabled network devices (switches, routers, etc.). Its primary functionality lies in enabling centralized control over the network by allowing the Controller to dynamically program the forwarding behavior of these devices.

c. **Data Plane**: It is like the physical layer of the OSI model, consisting of network elements like physical and virtual devices that deal with the data traffic. It is referred to as the forwarding plane of SDN and is physically responsible for forwarding frames of packets from its ingress to egress interface using the protocols used by control plane[4].

d. **Control Plane**: Centralized configuration and management are the function of the control plane of SDN. It is a logical entity in a software defined network that receives commands/instructions from the application layer and transmits them to the networking components of SDN. The task of the controller is to extract useful information from the hardware devices and communicate back to the SDN applications with an abstract view of the network, including various activities happening in the network. Usually a software solution, the SDN controller resides here to provide centralized control of the router and switch that populates the data plane, removing the control plane from the individual devices[4].

4) There are several steps involved in upgrading to a software-defined networking (SDN) architecture [3]:

1. **Please look over your current network:** The first step in upgrading to SDN is to assess your existing network infrastructure and determine what changes need to be made to support an SDN architecture. This may involve identifying any bottlenecks or performance issues and determining what devices and resources will need to be added or removed to support the new architecture.
2. **Plan the upgrade:** Once you have assessed your current network, you can begin planning the upgrade to SDN. This may involve creating a timeline for the upgrade process, identifying

potential challenges or risks, and determining what resources (such as budget and personnel) will be needed to complete the upgrade.

3. **Implement the SDN controller:** The SDN controller is the central component of an SDN architecture and is responsible for managing and coordinating the flow of data across the network. In order to upgrade to SDN, you will need to implement an SDN controller and integrate it with your existing network devices.
4. **Configure network devices:** Once the SDN controller is in place, you will need to configure your network devices (such as switches, routers, and firewalls) to be controlled by the SDN controller. This may involve updating firmware, installing new software, or making other changes to the devices.
5. **Test and deploy:** Once you have implemented and configured your SDN controller and network devices, you can begin testing the new architecture to ensure it is functioning correctly. Once you have completed testing, you can deploy the new SDN architecture and start using it to manage your network.

5) **Open Interface Vulnerability:** Open interfaces of the SDN network may bring new types of network attacks that may reduce the performance of the SDN. Various D-DOS attacks may down the working of networks. SYN flood attacks, which may bring down the server of any organization by exhausting the queue of the TCP protocol.

Centralized Controller Vulnerability: The SDN controller acts as a single point of failure. If compromised by attackers, they gain control over the entire network and manipulate network traffic, launch denial-of-service (DoS) attacks, or steal sensitive data flowing through the network.

Insecure Communication Channels: The secure communication between the controller and network devices is critical for maintaining network integrity. Weaknesses in the communication protocols, like OpenFlow, could allow attackers to intercept or modify network traffic. Unencrypted communication channels expose network policies and flow entries to potential eavesdropping.

Policy Enforcement and Authorization Issues: The SDN controller enforces network policies based on programmed rules. Misconfigurations or vulnerabilities in policy definitions can lead to unintended consequences, like allowing unauthorized access or blocking legitimate traffic. Inadequate role-based access control (RBAC) could allow unauthorized users to manipulate network policies or gain access to sensitive information.

6) ABC Corporation can assess the effectiveness and efficiency of their SDN deployment using various key performance metrics (KPIs). These metrics provide insights into different aspects of the network's health and performance. Here are some crucial SDN performance metrics and their significance:

- Throughput: This metric measures the amount of data successfully transmitted across the network per unit of time (usually Mbps or Gbps). High throughput indicates efficient data flow and network capacity to handle traffic volume.
- Latency: This metric measures the time it takes for a data packet to travel from its source to its destination. Low latency is essential for real-time applications like video conferencing and online gaming. SDN allows for dynamic routing, potentially reducing latency by optimizing traffic paths.
- Packet Loss: This metric represents the percentage of data packets that don't reach their destination. High packet loss can disrupt communication and lead to unreliable network performance. SDN's programmability can help identify and address bottlenecks that contribute to packet loss.
- Flow Table Size: This metric measures the number of flow entries stored in the switches' flow tables. Each flow entry represents a specific traffic flow and the forwarding instructions for it. Large flow tables can indicate complex traffic patterns or potential scalability limitations. Efficient SDN applications can optimize flow table usage.
- Controller Resource Utilization: This metric measures the CPU, memory, and other resources consumed by the SDN controller. High resource utilization can indicate controller overload and potential performance degradation. It's crucial to monitor controller resource usage to ensure it can handle network demands.
- OpenFlow Message Processing Time: This metric measures the time it takes for the controller to process OpenFlow messages received from switches. Fast processing times are vital for real-time network control and efficient communication between the control and data planes.

7) Security:

- Secure Controller Placement: The SDN controller is a critical component and should be placed in a secure, highly available environment. Implement strong access controls, authentication mechanisms, and regular security audits.
- Network Segmentation: Utilize SDN's capabilities to segment the network into logical zones. This compartmentalizes traffic flow and limits the impact of a security breach in one segment on others.
- OpenFlow Security: While OpenFlow is a powerful protocol, it can introduce vulnerabilities. Implement secure versions of OpenFlow (e.g., OpenFlow with Secure Channel) and leverage role-based access control (RBAC) to restrict access to specific network devices.

Flexibility and Scalability:

- Standardized APIs: Ensure chosen SDN solutions leverage open and standardized APIs like OpenFlow. This allows for integration with diverse network devices and future-proofs the network for incorporating new technologies.
- Automation: Automate repetitive network tasks like provisioning, configuration, and security policy enforcement. This reduces manual errors, improves efficiency, and allows for faster network adaptation to changing demands.
- Policy-based Management: Define network policies based on business needs and user groups. SDN allows for dynamic enforcement of these policies, enabling flexible and granular control over network traffic.

Reliability:

- High Availability (HA): Implement redundant SDN controllers to ensure continued network operation in case of controller failure. Consider geographically distributed controllers for additional resilience.
- Disaster Recovery (DR): Develop a comprehensive DR plan for the SDN environment, including controller backups and procedures for rapid network recovery in case of disasters.
- Testing and Validation: Thoroughly test and validate the SDN solution before deployment. This includes testing network functionality, security posture, and failover mechanisms.

SDN Controller Selection: Choosing the right SDN controller depends on ABC Corporation's specific needs and resources. Here are some popular open-source options to consider:

- OpenDaylight (ODL): ODL is a mature and feature-rich platform offering a modular architecture and support for various southbound protocols (including OpenFlow). It requires a steeper learning curve for deployment.
- Floodlight: Floodlight is a lightweight and easy-to-deploy controller, ideal for smaller networks or proof-of-concept deployments. However, it offers fewer features compared to ODL.
- POX: POX is a Python-based controller known for its flexibility and programmability. It's suitable for developers who want to customize the control plane extensively.

Reference

[1]Nunez, A., Ayoka, J., Islam, M.Z. and Ruiz, P. (2023). A brief overview of software-defined networking. *arXiv preprint arXiv:2302.00165*.

[2] Geeks of geeks. (2023). Difference between Software Defined Network and Traditional Network https://www.geeksforgeeks.org/difference-between-software-defined-network-and-traditional-network/amp/

[3]Aashiq Abdul Jaleel (2022). Introduction to Software-Defined Networking: A Guide to Unleashing the Full Potential of Your Network.
https://www.linkedin.com/pulse/introduction-software-defined-networking-guide-full-abdul-jaleel/
[4]Rana, Deepak & Dhondiyal, Shiv & Chamoli, Sushil. (2019). Software Defined Networking (SDN) Challenges, issues and Solution. INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING. 7. 884-889. 10.26438/ijcse/v7i1.884889.