Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

SBAC-SDN: A Scalable Blockchain-based Access Control in Northbound Interface for Multi-Controller SDN with Load Balancing Mechanism

Zulkarnain Zainal[1], Azizol Abdullah[1,*], Fahrul Hakim[1], Muhammad Daniel Hafiz Abdullah[2]

[1] Department of Communication Technology and Networks, Faculty of Computer Science and Information Technology, University Putra Malaysia, 43300 Serdang, Selangor, Malaysia
[2] Department of Computer Science, Faculty of Computer Science and Information Technology, University Putra Malaysia, 43300 Serdang, Selangor, Malaysia

ABSTRACT

Abstract The rapid adoption of Software-Defined Networking (SDN) has not only revolutionized network management but also presented challenges in scalability and security, particularly in multi-controller environments. This paper introduces a load balancing technique aimed at enhancing the scalability of the Northbound interface, a critical component often susceptible to performance bottlenecks. The experimental setup utilized Mininet and Ryu to create a simulated multi-controller SDN environment, where SBAC-SDN was thoroughly evaluated based on CPU and memory usage, response times, and error rates. The results showed substantial improvements in critical performance indicators, confirming the effectiveness of the load balancing technique in addressing scalability challenges. The findings indicate that the system can be scaled without excessive resource consumption, as CPU usage peaked at 60.3% and memory usage remained below 8 GB even with eight controllers. The stable average response times were consistently below 1 s, and the high throughput, ranging from 4.34 to 4.67 requests per second, further demonstrates the efficiency of the system. The error rate remained 0.0% across all configurations, highlighting the robustness of the implemented security measures. This study contributes to ongoing efforts to improve the scalability and overall robustness of SDNs, aligning with the broader goals of reliability and efficiency in network management.

Keywords:
Software-defined networking; Load balancing; Northbound interface; Scalability; Distributed systems

1. Introduction

Software-Defined Networking (SDN) has revolutionized the networking landscape by decoupling control and data planes, offering unprecedented flexibility and programmability. However, this flexibility comes at the cost of increased vulnerability, particularly at the Northbound interface, where applications interact with the SDN controller. Existing research has highlighted these vulnerabilities, but falls short in offering scalable solutions, especially in multi-controller

* *Corresponding author.*
E-mail address: azizol@upm.edu.my

https://doi.org/10.37934/araset.55.1.2443

environments. This research aims to address these gaps by introducing a load balancing technique and cache mechanism specifically designed for multi-controller SDN settings.

1.1 Background

Software-Defined Networking (SDN) Software-Defined Networking (SDN) has emerged as a transformative technology in network management. SDN offers unprecedented flexibility and programmability by decoupling control and data planes. However, this architectural shift introduced new challenges, particularly in terms of scalability and performance.

This section introduces the background of SDN, including their basic architecture, scalability, and security problems. In addition, we provide an overview of the main concepts and principles of blockchain technology and then discuss the possibility of utilizing blockchain in the SDN security context.

1.1.1 Overview of SDN

Software-defined networks (SDN) have revolutionized network architecture by separating network logic from the underlying forwarding devices. The three-layer SDN architecture shown in Figure 1 consists of an application layer, control layer, and data plane. The Northbound interface connects the controller and applications, whereas the Southbound interface is the communication link between the controller and the physical networking hardware.

The controller, as the brain of the network, directs traffic to its desired destination by receiving routing information from the data plane through flow-rule installation. The OpenFlow protocol facilitates the exchange of messages between the control and data planes.

This centralized control method provides an entire view of the network and simplifies the programming, modification, and management of network configuration. Third-party applications can leverage the management information in the controller to perform various operations such as load balancing and statistics. To communicate with the controller, these applications use a Northbound interface as an intermediate channel. This architectural transformation has revitalized the network layer, enabling flexible security policies and centralized management of networks through the deployment of applications, such as firewalls, proxies, and load balancers, that support the OpenFlow protocol.

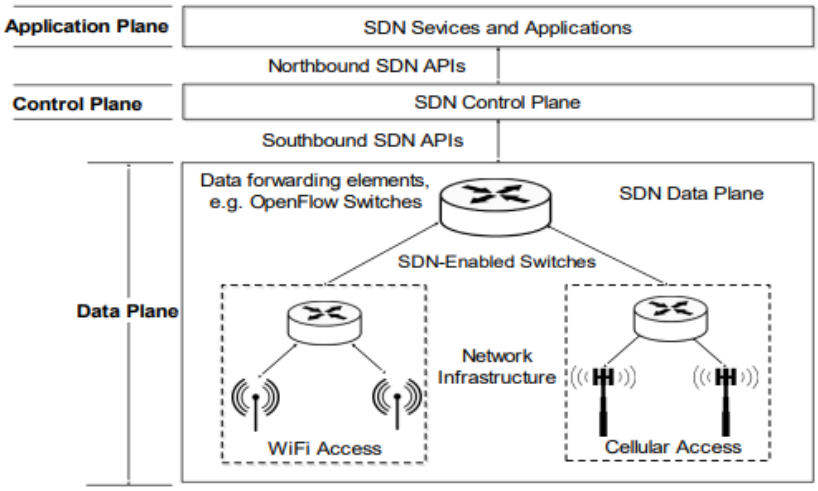

Fig. 1. Overview of SDN architecture [1]

1.1.2 Scalability and security issues from external applications and northbound interface on controller

It is imperative to carefully consider the design of the system to ensure that the controller can handle the traffic and security requirements of external applications and the Northbound interface. This may involve the implementation of additional controllers or utilization of alternative solutions to enhance scalability and security.

There are several reasons why a single controller may not be sufficient to manage scalability and security issues from external applications and the Northbound interface effectively. These include limited resources, complexity, a lack of flexibility, and difficulties in monitoring and troubleshooting. Therefore, it is crucial to evaluate the system design to ensure that the controller can handle traffic and security requirements and to consider implementing additional controllers or alternative solutions [2].

1.1.2.1 Scalability

SDN scalability is multidimensional, with varying interpretations across different systems. In some systems, it refers to the parallel execution of multiple applications on different CPUs, whereas in others, it involves optimizing the system's resources with a dynamic workload. Despite the lack of a standard definition or content, this study focuses on the performance of SDN controllers in handling authentication requests, blockchain queries, and query responses. Centralized controllers aim to achieve scalability through parallelism, whereas distributed controllers aim to achieve scalability using horizontally distributed or hierarchically organized controllers. In hybrid SDN control, scalability is largely dependent on the performance of the central SDN controller and the efficiency of the mechanisms used for interoperability between legacy-distributed and centralized SDN controllers [3].

1.1.2.2 Security

The security of an entire network is compromised when vulnerabilities exist in the SDN controller. To mitigate attacks such as spoofing, tampering, Denial of Service (DoS), and privilege elevation, it is essential to implement measures such as process containment, application permission structure, and resource utilization monitoring within the controller [4]. In a single centralized controller, a malformed OpenFlow header can cause a crash, and frequent flow requests from an attacker can degrade overall performance. Authentication protocols are urgently required in the distributed control architecture to validate and verify controller instances before proceeding with state information exchange. Such protocols can prevent impersonation and DoS attacks that are likely to occur in the hybrid SDN control plane, owing to the combination of the two control planes and diverse devices in the network. In addition, it is crucial to secure exchanges at SDN interfaces with security protocols to mitigate various integrity threats, regardless of the SDN control plane architecture [5].

1.2 Problem Statements

In a multi-controller environment, the Northbound interface serves as the primary communication channel between the controllers and applications. However, the growing size and complexity of networks pose significant challenges, turning the Northbound interface into a bottleneck that restricts system scalability [6-8]. The fixed capacity of the Northbound interface becomes increasingly strained as the number of applications and controllers increases, hindering its ability to manage the escalating volume of traffic generated by various network components.

Security compounds these challenges because the Northbound interface is often inadequately designed, making it susceptible to diverse types of attacks [9,10]. Attackers can exploit vulnerabilities to intercept or modify the control plane state and configuration data, potentially destabilizing the network. Unauthorized access to the control plane via this interface can compromise the entire network.

Existing solutions attempt to address these issues; however, they have limitations that warrant consideration. The RESTful Northbound interface design proposed by Alghamdi *et al.,* emphasized compatibility but may have scalability constraints [11]. Rathnamalala *et al.,* presented a Fully Integrated SDN Testbed; however, the scalability of their solution in multi-controller environments remains an open question [12]. Bai *et al.,* implemented a Northbound interface in a multi-domain control framework, offering insights, but potential scalability challenges have not been thoroughly explored [13].

Moreover, the B-DAC framework by Duy *et al.,* introduced a decentralized access control on the Northbound interface using blockchain [14]. Although innovative, its broader applicability and potential scalability hurdles require detailed examination. Liu *et al.,* propose DACAS, integrating attribute-based access control for Northbound interface security [15], yet the trade-offs and limitations in real-world deployments need further scrutiny.

By summarizing these existing solutions and their limitations, this study aims to delineate the gaps in current approaches and provide a foundation for proposing more effective and scalable solutions to address Northbound interface challenges in multi-controller SDN environments.

1.3 Research Questions and Objectives

The objective of this paper is to investigate how load balancing techniques can be effectively utilized to enhance the scalability of the Northbound interface in multi-controller SDN environments by addressing the research question: "How can load balancing techniques be implemented to improve the scalability of the Northbound interface in multi-controller SDN environments?" The main goal is to develop and evaluate a load balancing approach that optimizes the distribution of incoming requests across multiple SDN controllers.

The paper intends to address the following research question: "How can cache mechanism be employed to enhance the delay and response of Blockchain queries for authentication in multi-controller SDN environments?" In addition, this study aims to develop and assess the performance of caching integration within controllers that optimizes the authentication process across multiple SDN controllers.

Last but not least, the paper seeks to address the following research question: "How can blockchain technology be utilized for the security of the Northbound interface in multi-controller SDN environments?" The subsidiary objective is to develop and test the integration of blockchain technology with a controller that manages authentication requests across multiple SDN controllers.

1.4 Scope and Contributions

The scope of this paper is to present and evaluate a method for improving scalability and security in multi-controller SDN settings through load balancing. Although caching mechanisms and blockchain technology have also been discussed, they are not the main focus of this study. The key contributions of this study consist of an effective load balancing approach and its experimental validation, which demonstrate notable improvements in scalability metrics.

1.5 Paper Organization

The remainder of this paper is organized as follows. In Section 2, we discuss related work. Section 3 outlines our methodology, followed by Section 4, which details the implementation. Section 5 presents and discusses the results. Finally, Section 6 concludes the paper and suggests future work.

2. Related Works

In the context of Software-Defined Networking (SDN), this section provides an integrated discussion of scalability, load balancing, caching, and the potential use of blockchain technology. Previous studies have extensively explored scalability and security issues in SDN, focusing on parameters such as flow entries and controller capacity, which were not on the Northbound interface for examples like in [16-18]. However, the integration of load balancing, caching, and blockchain in addressing these scalability challenges, particularly in multi-controller environments, remains underexplored. Overall, the use of multi-controller environments and load balancing techniques is essential for achieving scalability in SDN, and can help ensure that networks can adapt to changing demands and grow as needed.

2.1 Existing Load Balancing Techniques

Although load balancing is not a novel concept, it has recently emerged as a topic of interest in SDN. Various techniques, including basic round-robin algorithms and advanced adaptive methods, have been proposed. However, many of these techniques are tailored for web servers or cloud computing settings, and may not be immediately adaptable to SDNs, particularly at the Northbound interface.

Network performance in a Software-Defined Networking (SDN) environment with multiple controllers can be optimized through load balancing. Various techniques have been proposed to address the challenges of workload allocation and optimization in SDN. One approach is the use of a Lyapunov-based design for distributed load balancing algorithms [19]. This algorithm dynamically allocates the workload among the available SDN controller instances to optimize network performance, considering factors such as scalability and software-defined networking to improve service quality. Another strategy is the use of deep-learning-based load prediction and switch migration in SDN multi-controller clusters [9]. This approach addresses issues such as the high switch migration cost, load imbalance, and inefficient load balancing. It utilizes a migration switch selection algorithm, a target controller selection algorithm, and a switch migration decision algorithm to optimize the load distribution.

In [20] an adaptive load balancing technique was proposed for scenarios involving multiple SDN controllers. The aim is to optimize resource utilization, enhance scalability, and improve the overall network performance. The study suggests that the proposed technique results in a more efficient load distribution among controllers, reducing the latency and improving the response time of SDN.

A reliable load balancing fault-tolerant multi-SDN controller approach was proposed to enhance the reliability and fault tolerance of SDN. This approach demonstrates superior performance compared with traditional mechanisms, ensuring reliable load balancing in multi-controller environments [21].

Artificial intelligence-based load balancing frameworks have been proposed to balance the load in SDN [22]. These frameworks employ algorithms, such as affinity propagation based on particle swarm optimization (PSO), to optimize the load distribution in multi-domain SDN.

The study in [23] addressed the challenges of adaptability and scalability in the context of distributed SDN controllers. This study introduces a novel approach aimed at dynamically adjusting the network topology to enhance scalability, reduce latency, and improve the overall SDN performance. The findings suggest that the proposed method achieves better scalability and adaptability than traditional SDN controller architectures.

Hybrid SDN networks present their own load balancing challenges, which can be addressed through multi-parameter server load balancing schemes [3]. These schemes aim to optimize load balancing in hybrid SDN environments with minimal SDN device sets.

Various load balancing techniques can be employed in software-defined networks (SDNs) with multiple controllers. These techniques range from Lyapunov-based algorithms to deep learning-based load prediction, artificial intelligence-based frameworks, and nature inspired meta-heuristic algorithms. Each technique targets specific issues and aims to optimize load distribution and network performance in SDN environments. Some of the load balancing techniques that have been explored in SDN include emulation of the POX controller as a load balancer, game theoretic consensus for controller load balancing, resilient placement of SDN controllers exploiting disjoint paths, and traffic management inside software defined data centre networking. In addition, nature-inspired meta-heuristic algorithms have been explored for load balancing in SDN to address the NP-complete nature of load balancing problems in SDN [24].

In summary, the use of load balancing techniques in SDNs with multiple controllers provides a range of solutions to specific challenges and aims to improve the network performance.

2.2 Northbound Interface Security and Performance

The Northbound interface plays a crucial role in software-defined networking (SDN) architecture, as it serves as the communication link between the SDN controller and applications that manage network behaviours. Although previous studies focused on securing this interface, limited attention has been paid to its performance and scalability, particularly in environments with multiple controllers. [8] provided a relevant reference in this regard, providing a comprehensive survey of SDN. This study highlights the significance of the Northbound interface and its ability to help developers create revenue-generating applications without being hindered by the complexities of the underlying networks. This reference reinforces the importance of the Northbound interface in managing the network behaviour.

The paper [25] presented a token-based authentication method was presented as a solution to the threat of network attacks in SDN architectures, wherein only legitimate network applications with valid credentials are granted access to execute operations within the network. Additionally, an authorization method is implemented to enforce permission constraints, and the trust between the controller and network application is evaluated using Subjective Logic Reasoning.

SEAPP [26] is a secure application management framework designed to protect against malicious attacks by managing application permissions and encrypting REST API calls. This framework ensures the security and effectiveness of the system, with minimal impact on CPU and memory resources.

This paper presents a decentralized access control mechanism known as SILedger [27], which utilizes blockchain and attribute-based encryption to address the deficiencies in permission management within software-defined networks for Internet of Things (SDN-IoT) applications. This mechanism offers efficient authorization; maintains a record of interactions; and enables the implementation of charging, analysis, and audit mechanisms for SIApps.

The study in [28] targeted the improvement of access control in SDNs by introducing a fine-grained mechanism based on smart contracts. The focus is on enhancing security by allowing

organizations to specify and enforce precise access policies at the terminal level. These findings suggest that FACSC offers improved access control compared with traditional methods

Although security is crucial, it is also important to consider the performance and scalability of the interface. Although some studies have addressed these issues, there are still studies that highlight the importance of performance and scalability in the design and implementation of the Northbound interface. Additionally, references discuss the security aspects of the interface and propose frameworks for secure and controller-independent Northbound interfaces.

2.3 Blockchain for Network Security

Blockchain technology has garnered significant attention in recent years owing to its vast potential to revolutionize various domains, including network security. The utilization of blockchain technology in network security presents numerous advantages such as decentralization, immutability, and transparency. In this discussion, we delve into the topic of blockchain for network security and draw upon relevant scholarly work to support our findings.

One of the primary sources that we will refer to in this discussion is the paper titled "B-DAC: A Decentralized Access Control Framework on Northbound interface for Securing SDN Using Blockchain" by Phan The Duy [14]. This groundbreaking paper proposes a decentralized access control framework that leverages blockchain technology to enhance the security of Software-Defined Networking (SDN). The framework utilizes smart contracts to enforce access control policies and guarantees the confidentiality and integrity of network data.

The paper "Fine-Grained Access Control Based on Smart Contract for Edge Computing" [29] proposes a Fine-Grained Access Control (FGAC) mechanism for edge computing, leveraging smart contracts. This study focuses on enhancing security by providing precise control over access rights in edge computing environments. The findings suggest that the FGAC mechanism offers improved access control compared with conventional methods.

Two relevant papers have recently been published. "BCNBI: A Blockchain-Based Security Framework for Northbound interface in Software-Defined Networking" [30] presents a blockchain-based security framework for the Northbound interface in SDN. This framework leverages blockchain's decentralized and immutable nature to enhance the security and trustworthiness of the Northbound interface, which is critical in SDN architectures.

"BMC-SDN: Blockchain-Based Multicontroller Architecture for Secure Software-Defined Networks" [31] is another paper that proposes a blockchain-based multi-controller architecture for secure SDNs. The architecture uses blockchain technology to ensure secure communication and coordination between multiple controllers in an SDN environment. This enhances the security and resilience of SDNs against various types of attacks.

The studies examined in this analysis demonstrate the potential of blockchain technology to enhance network security, particularly in the context of Software-Defined Networks (SDNs). By utilizing the blockchain's decentralized and immutable nature, these frameworks offer secure access control, secure communication, and coordination among network components, while ensuring the confidentiality and integrity of network data. In summary, blockchain technology holds promising solutions for network security and can improve the trustworthiness of network systems through its application in access control, communication, and coordination among network components. The papers analysed provide valuable insights and frameworks for implementing blockchain-based security solutions in network environments.

2.4 The Gap in Existing Literature

Previous studies in 2.2, and 2.3, claimed that they have already tried to cover the security of the Northbound interface, and only a few studies have used a multi-controller environment; therefore, this remains a gap for this research.

To elucidate the symbiotic relationship between load balancing, caching, and blockchain in addressing both scalability and security challenges, we present a schematic (Figure 2). This visual representation underscores how the combined approach enhances the resilience and efficiency of SDN networks in a multi-controller environment.

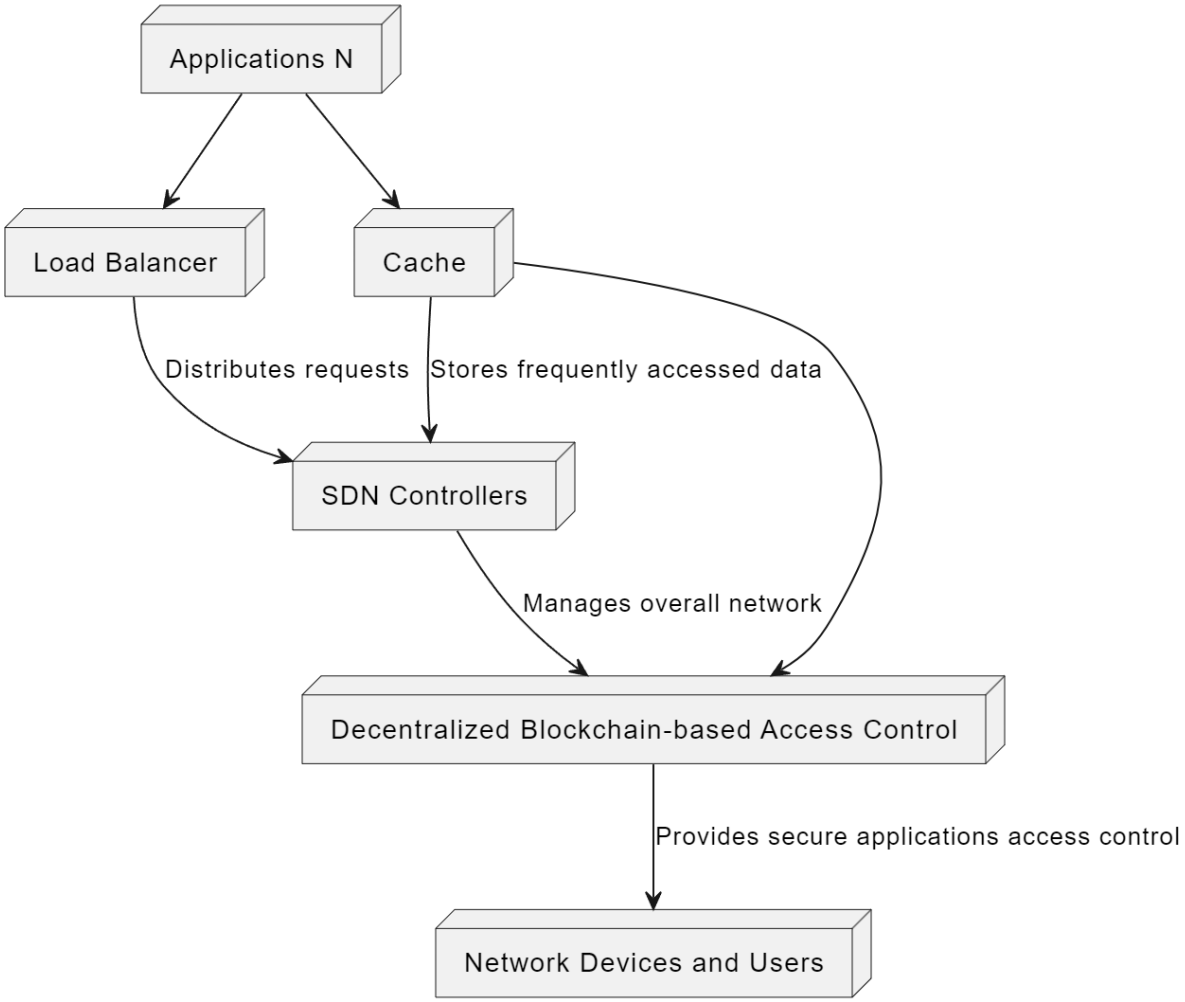

Fig. 2. Schematic diagram

While there is extensive research on individual aspects such as scalability, load balancing, and Northbound interface security, there is a noticeable gap in studies that integrate these elements with a focus on multi-controller SDN environment. This paper aims to fill this gap by focusing on the implementation and evaluation of a load balancing technique designed to enhance both the scalability and reliability of the Northbound interface in multi-controller SDNs.

3. Methodology

The methodology employed in this study aimed to comprehensively evaluate the effectiveness of the proposed load balancing technique within a multi-controller SDN environment. This approach encompasses the research design, tools and technologies utilized, evaluation metrics, and detailed description of the experimentation process.

3.1 Research Design

We adopted an experimental design to rigorously assess the effectiveness of the load balancing technique. This study involved creating a simulated network using Mininet [32] with multiple instances of the Ryu controller to form a dynamic, multi-controller SDN environment. To handle decentralized access control securely and facilitate transparent decision-making, we integrated Hyperledger Fabric [33] and Minifabric.

3.2 Tools and Technologies

This experiment leveraged innovative tools and technologies to ensure a controlled and highly adaptable experimental framework. Mininet served as the foundation for network simulation, whereas Ryu functioned as an SDN controller. Hyperledger Fabric with Minifabric provided a robust solution for decentralized access control. The Hyperledger Fabric SDK (Node.js [34]) acts as the interface connecting the SDN environment to an API server [35], enabling seamless communication between the Ryu controller and the blockchain network.

For load balancing, we employed HAProxy [36], configured for dynamic load balancing using the least connections algorithm supported by caching strategies. Custom Python scripts were developed to collect comprehensive performance metrics, ensuring a holistic evaluation of the load balancing technique's impact.

By combining these technologies, our experimental setup offers a cohesive approach that addresses both scalability and security challenges in multi-controller SDN environments.

3.3 Evaluation Metrics

The effectiveness of the load balancing technique is assessed using a comprehensive set of evaluation metrics. These metrics provide a well-rounded view of the system performance. Key metrics included:

i. CPU and Memory Usage: Monitoring the utilization of CPU and memory resources helps gauge the impact of load balancing on the computational overhead of the controllers.
ii. Response Time: Measuring the response time allowed for the evaluation of how efficiently the load balancing technique managed incoming requests, ensuring minimal delays in the network responsiveness.
iii. Error Rate: The error rate was monitored to determine the ability of the load balancer to handle and redirect traffic without causing errors or disruptions.

These metrics collectively provide valuable insights into the overall performance of the proposed load balancing technique.

3.4 Experimentation

The experimental process involved the execution of multiple scenarios to simulate the diverse network conditions and loads. Each scenario, including the interaction with Hyperledger Fabric, was meticulously tested using custom Python scripts and Hyperledger Fabric SDK (Node.js). A detailed workflow, from authentication to application acceptance, was simulated to comprehensively assess the performance of the load balancing technique, as described in the next section.

This methodology ensures a comprehensive and systematic approach for evaluating the load balancing technique in a multi-controller SDN environment. The combination of robust research design, advanced tools and technologies, well-defined evaluation metrics, and rigorous experimentation allows for a thorough assessment of the effectiveness and implications of the proposed technique for network scalability. An overview of the methodology used in this study is shown in Figure 3.

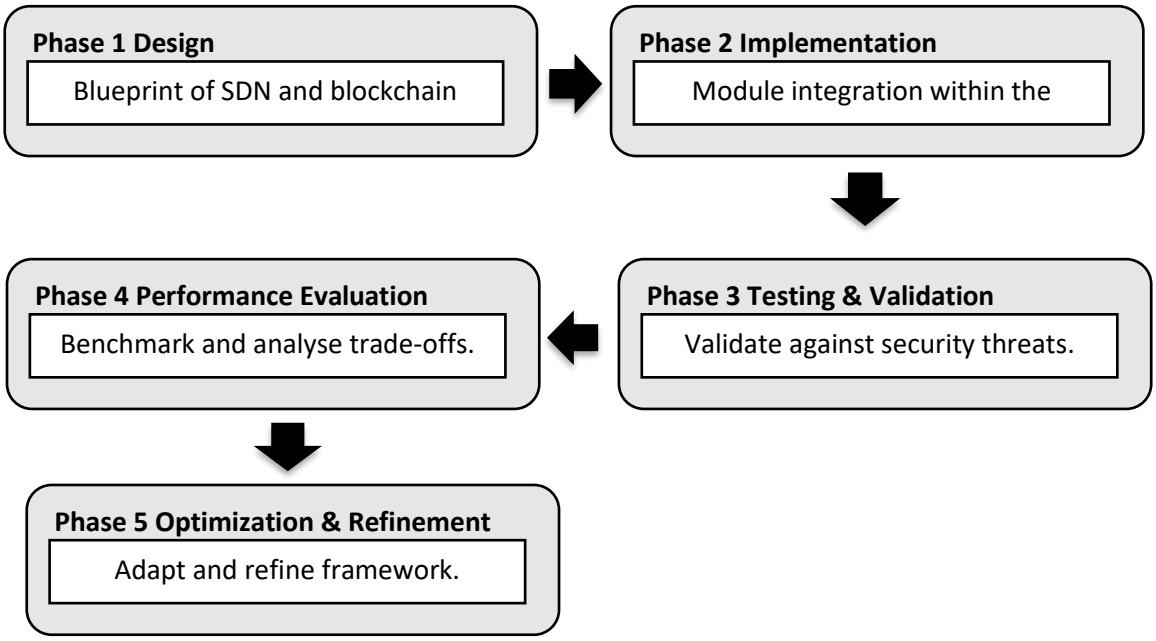

Fig. 3. Methodology Overview

The methodology for this research was meticulously designed to address the challenges posed by the Northbound interface in SDN. Understanding the vulnerabilities inherent to this interface is important for our approach. In addition, we thoroughly examined the compatibility of blockchain with the SDN architecture, recognizing its potential to enhance security and access control.

To concretize our conceptualization, we propose an initial blueprint for the framework. This blueprint delineates the interaction between SDN and blockchain, serving as the foundation for subsequent implementation.

The implementation phase involved leveraging the Mininet virtual environment to emulate the network components under investigation. Within this virtualized setting, the proposed framework was seamlessly integrated as a module within the control layer. This integration allowed the assessment of the practicality and effectiveness of the proposed approach in a controlled environment. The development of scripts plays a pivotal role in facilitating the functionality of frameworks. Utilizing both Node.js and Python, we ensured that the scripts were aligned with the

architecture of the proposed framework, thereby enabling a comprehensive evaluation of its capabilities.

Subsequent to the implementation, our methodology involved deploying the framework in a controlled environment using the Ryu controller. This deployment is crucial for validating the framework operation against a spectrum of potential security threats. Gathering preliminary data on the system performance and security metrics is an integral part of the validation process.

To gauge the relative efficacy of our solution and address the core focus of scalability, we conducted rigorous performance evaluation. Benchmarking the proposed framework against existing methods provides valuable insight into its capabilities. Importantly, we designed scalability tests to assess the performance of the framework in a multi-controller environment. This involved deploying multiple instances of the Ryu controller and analysing the system response to varying workloads and network conditions. The results of these scalability tests play a pivotal role in evaluating the effectiveness of the framework for handling increased demands while maintaining security.

In the optimization and refinement phases, particular attention was paid to the seamless and efficient operation of the integrated blockchain solution. Refining the integration ensured that the framework operated cohesively within the SDN environment, aligning with our overarching objective of maintaining security without compromising the system efficiency.

4. Implementation and Experimental Setup

The implementation of load balancing involves several crucial components and technologies that offer clear and comprehensive understanding of the proposed system architecture. This section includes a detailed explanation of the load balancing technique, multi-controller setup, caching mechanism, and challenges encountered during implementation, along with their solutions.

4.1 Load Balancing Technique

The implementation of the load balancing technique involves configuring the parameters of both the Mininet and Ryu controllers. Mininet, our SDN infrastructure simulation tool, was deployed on a specified platform, whereas Ryu ran on a computer equipped with four CPU cores and 8 GB RAM. The load balancing technique leveraged REST APIs to send various types of requests to the Ryu controller, allowing us to assess its performance under different scenarios. The requests were varied in type, number, and size, thereby providing a comprehensive evaluation of the efficacy of the load-balancing technique.

4.2 Multi-Controller Setup

The Ryu controller, pivotal to our Software-Defined Network (SDN) experiment, was hosted on a computer with four CPU cores and 8 GB of RAM. For precise control, we configured parameters, such as the number of requests, number of controllers, and sample of applications with customized attributes, as shown in Figure 4. In our setup, Mininet is employed to simulate the SDN infrastructure, representing a linear network with 16 switches and 32 connected hosts. The configurations of Mininet and Ryu are bounded with functions for the topology and API endpoints, ensuring the accurate emulation of real-world scenarios. To assess the load balancing technique, we used the REST APIs of the Ryu controller for various requests. These requests included appId, controllerId, resourceURI and httpMethod, aiming to simulate diverse network scenarios and stress test the load balancing mechanism under different conditions. Multiple instances of the Ryu controller were

meticulously configured to function cohesively in a multi-controller environment. This involves HAProxy, which enables seamless coordination among the controllers. Their operations were synchronized through a shared database to ensure consistent and collaborative decision making in network management. To uphold security and decentralization, we utilized the Hyperledger Fabric SDK (Node.js). This framework facilitates robust communication between the Ryu controller and blockchain network, ensuring that decision-making processes are secure, decentralized, and resilient against potential vulnerabilities.

```
10
11  # Define the parameters
12  num_requests = 1000  # Number of requests to send
13  controller_url = 'http://localhost:8081'  # Mock API server URL
14  app_credentials = [  # Replace with your application credentials
15      {'appId': 'APP0001', 'controllerId': 'CNTL001', 'resourceUri': '/api/resourceA', 'httpMethod': ['GET', 'POST', 'DELETE']},
16      {'appId': 'APP0002', 'controllerId': 'CNTL002', 'resourceUri': '/api/resourceB', 'httpMethod': ['GET', 'POST', 'PUT', 'DEL
17      {'appId': 'APP0003', 'controllerId': 'CNTL001', 'resourceUri': '/api/resourceC', 'httpMethod': ['GET', 'PUT', 'DELETE']},
18  ]
19
```

Fig. 4. The parameters of the experiment

The workflow process from authentication to acceptance is described in detail. The Hyperledger Fabric SDK (Node.js) facilitated communication between the Ryu controller and the blockchain network, ensuring secure and decentralized decision-making. The comprehensive workflow considers the entire lifecycle of an application in an SDN environment and provides insights into the impact of the load balancing technique at each stage. The overall setup is illustrated in Figure 5.

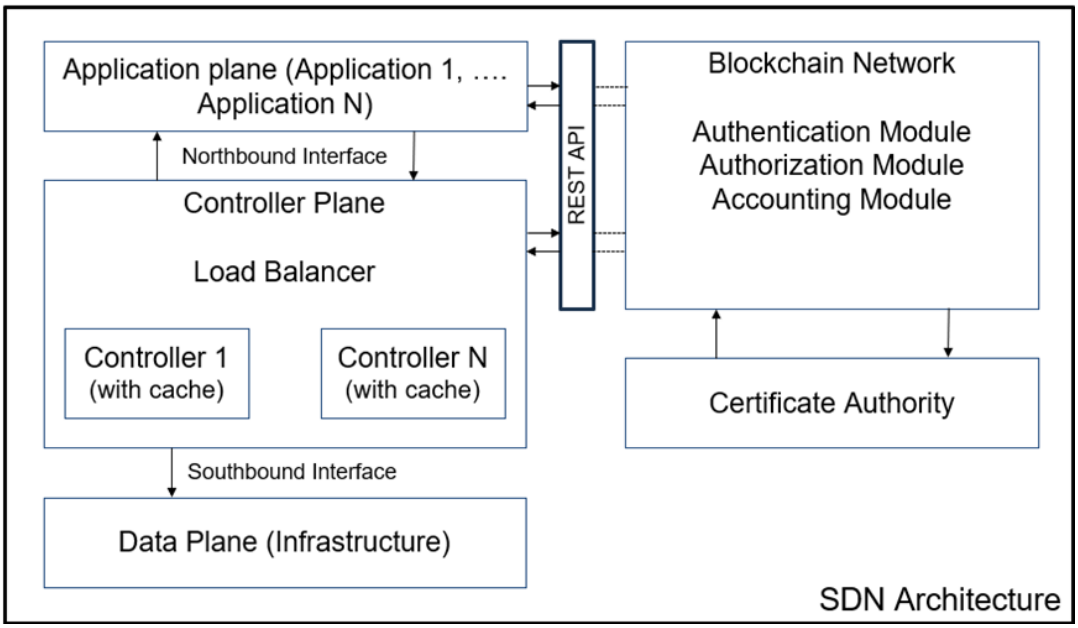

Fig. 5. The overview of experimental framework

4.3 Caching Mechanism

While the primary focus of this study was loading balancing, an additional feature was implemented to enhance network performance. A caching mechanism, including the addition of a trust index in GET requests within the Ryu controller's functions, was introduced to optimize the response times and reduce the load on SDN controllers. This mechanism aims to store previous access control decisions and reduce the frequency of blockchain queries, thereby alleviating the computational burden on the controllers. Although the detailed impact and performance evaluation of this caching mechanism were not the primary focus of this study, they played a crucial role in

optimizing the overall SDN environment. The trust index in GET requests contributes to the effectiveness of the caching mechanism, providing a means of prioritizing and retrieving cached decisions efficiently.

4.4 Blockchain Integration with Hyperledger Fabric

In addition to the load balancing and multi-controller setup, this study incorporated the integration of Hyperledger Fabric, utilizing Minifabric to enhance the functionality of the SDN environment. This subsection provides an overview of the role of Hyperledger Fabric and Minifabric in its implementation.

Hyperledger Fabric, a prominent open-source blockchain framework, was integrated to handle access control decisions in a secure and decentralized manner. Minifabric, a lightweight tool designed to simplify the deployment of Hyperledger Fabric networks, was employed for efficient setup and management.

Specifically, Hyperledger Fabric was configured to serve as a decentralized access control mechanism, complementing the load balancing and multi-controller components. Smart contracts and distributed ledger technology enable secure access control decision making and maintain an immutable record of network transactions. Node.js SDK provides a reliable and efficient interface for communication, allowing flexibility and customization in the development and deployment of blockchain networks. Node.js is a popular run-time environment for scalable and efficient applications. Its integration with the Hyperledger Fabric SDK provides developers with a familiar and powerful toolset for creating blockchain-based solutions. With Hyperledger Fabric SDK, developers can easily interact with the blockchain network, create and manage smart contracts (chaincode), and customize networking rules and consensus protocols according to their specific requirements. The combination of Hyperledger Fabric SDK and Node.js enables developers to leverage the extensive ecosystem of Node.js libraries and frameworks, thereby facilitating rapid development and integration with existing systems.

Minifabric streamlined the deployment process, ensuring a rapid and consistent setup of the Hyperledger Fabric network, thereby minimizing implementation complexities.

The function of the Certificate Authority (CA) in a blockchain network is to authenticate participants, including applications and controllers, when using SDN resources. This helps to prevent spoofing by verifying the identity and trustworthiness of the participants. The CA issues X.509 certificates, which contain identity information and credentials, to participants for certificate-based authentication. This replaces the traditional username-password pairs. CA plays a crucial role in the authentication process of the proposed system. Participants first authenticated the REST API using JSON Web Tokens (JWT) and then uploaded their identity cards (wallets) to the system. These identity cards are used to authenticate the blockchain and to sign transactions. By providing a trusted source of authentication, the CA enhances the security and integrity of the blockchain network, ensuring that only authorized participants can access and interact with the network resources. In summary, the purpose of CA in a blockchain network is to authenticate participants, prevent spoofing, and increase network security.

Blockchain integration aims to provide a comprehensive solution for managing access control decisions, while reducing the computational overhead of SDN controllers. However, it is important to note that the primary focus of this study is the load balancing technique and its impact in a multi-controller SDN environment. A detailed evaluation of the blockchain component, including its performance and implications, was not the main focus of this study.

4.5 Testing and Script Execution

During the testing phase, scripts were executed to simulate the entire workflow process, from authentication to authorization, within the SDN environment. These scripts were designed to emulate the behaviour of mock applications, including monitoring, firewalls, and a combined Monitoring, Firewall application. The scripts initiated 1000 requests for authentication and authorization queries to the blockchain network, providing a robust simulation of real-world network conditions.

The testing scenario involved the use of mock applications representing various network functionalities including monitoring, firewalls, and a combined Monitoring, Firewall application. These applications, both individually and in combination, generate realistic authentication and authorization requests that simulate the dynamic nature of SDN environments. The execution of 1000 requests per application type allowed for a comprehensive assessment of the scalability and reliability of the load balancing technique.

The results obtained from the testing and performance evaluations were systematically analysed to derive meaningful insights. The effectiveness of the load balancing technique in enhancing the overall network performance and its ability to handle blockchain transactions efficiently were thoroughly examined. The performance of individual components, such as the load balancer, Hyperledger Fabric, and the Ryu controllers, was assessed in detail in the next section.

5. Results and Discussion

The Results and Discussion section of this paper presents the findings obtained from the extensive experimentation and analysis conducted to evaluate the effectiveness of the implemented load balancing technique within a multi-controller Software-Defined Networking (SDN) environment. This section is divided into several key components, including performance metrics, scalability improvements, graphical representations, and implications and limitations of the study.

5.1 Scalability Improvements

The results of the performance analysis provide significant insights. Notably, the implemented load balancing technique showed remarkable improvements in system scalability. As the number of controllers increased, CPU and memory usage remained within acceptable limits, indicating the ability of the technique to efficiently manage network resources, even in a multi-controller setting. Furthermore, the response time consistently demonstrated that the system could handle incoming requests adeptly, thereby contributing to an enhanced user experience.

Utilizing the least connections method with multiple controllers and a load balancer fosters scalability by equitably apportioning the workload among the controllers and preventing any individual controller from being overwhelmed with requests. The load balancer continuously monitors the number of active connections to each controller and directs new requests to the controller with the fewest connections, thus optimizing the resource utilization. This strategy also enhances the fault tolerance by providing redundancy. If one controller malfunctions or becomes unresponsive, the load balancer can automatically re-route requests to the remaining controllers, thereby ensuring uninterrupted authentication queries. By disseminating the workload and offering redundancy, this configuration reduces the likelihood of a single point of failure, and improves the overall reliability and availability of SDN systems.

5.2 Graphical Representations

To provide a more comprehensive understanding of the results, this section incorporates various charts and graphs that visually represent the performance metrics shown in Figures 6, 7, 8 and 9.

5.2.1 CPU and Memory Usage

The CPU usage across different numbers of controllers (2, 4, 5, and 8) showed slight variation, ranging from 49.4% to 60.3%. Memory usage also exhibited a moderate increase, with the highest being approximately 7.8 GB for the eight controllers.

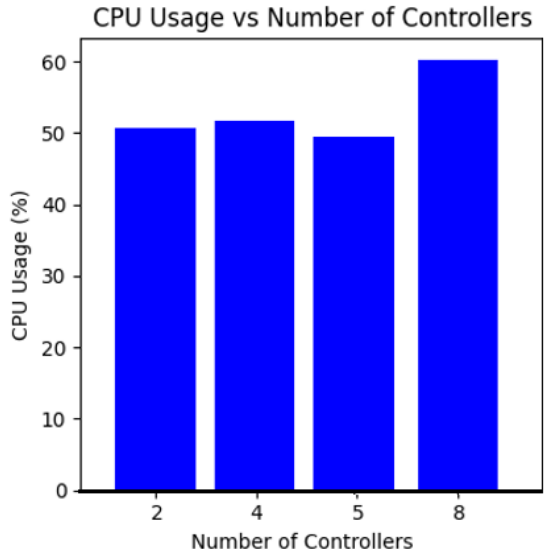

Fig. 6. CPU usage of SBAC-SDN

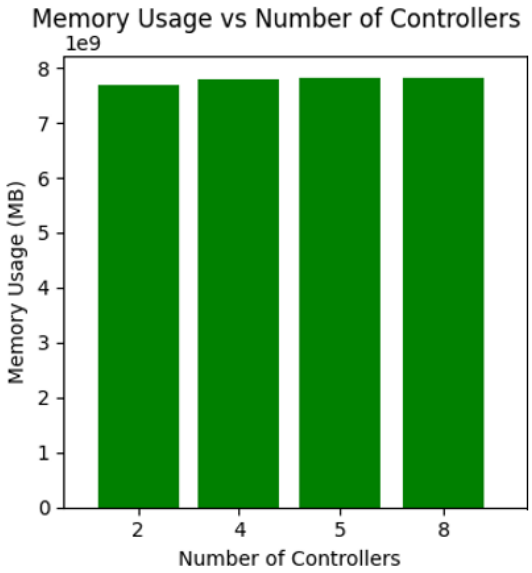

Fig. 7. Memory usage of SBAC-SDN

The CPU and memory usage metrics indicate that the system is not overly resource-intensive. Even with eight controllers, CPU usage peaked at 60.3% and memory usage remained below 8 GB. This is a significant finding because it suggests that the system can be scaled without excessive computational resources.

5.2.2 Response time

The average response time remained fairly consistent across all configurations, ranging from 0.214s to 0.230s. The maximum response time is 0.726s for the eight controllers, whereas the minimum is 0.177s for the two controllers.

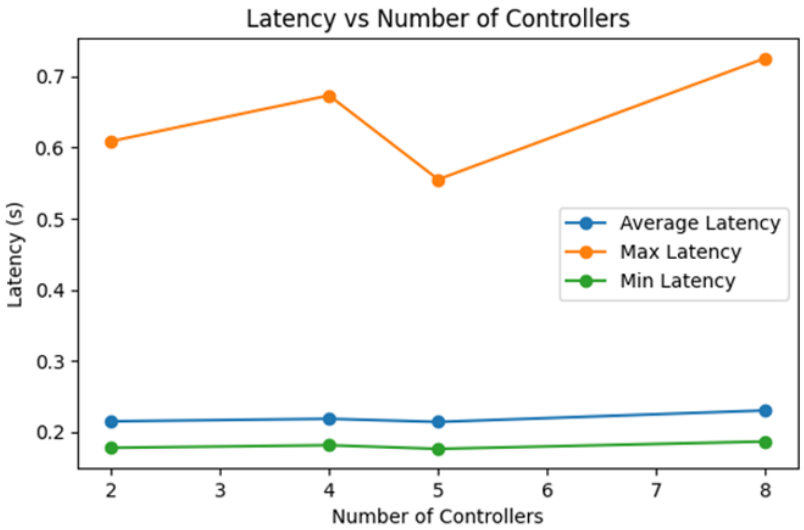

Fig. 8. Latency of SBAC-SDN

Consistency in the average response times across different numbers of controllers is a strong indicator of the system's ability to handle incoming requests efficiently. Even at the peak, the maximum response time remained below 1s, which is within acceptable limits for real-time applications.

The utilization of a cache mechanism in conjunction with a trust index enhances the performance and efficiency of the data exchange by minimizing the need for repeated authentication and verification. The cache mechanism saves the response to a request, allowing the SDN controller to respond immediately to subsequent requests without requiring reauthentication. This caching mechanism not only improves the response time, but also reduces the load on the blockchain-based access control framework, leading to increased efficiency. The trust index, which represents the well-behaved nature of an application, is combined with a cache mechanism to determine whether SDN controllers should trust or deny further application actions. By integrating the trust index, the cache mechanism can update the cached information and trust index accordingly, ensuring that only trusted and authorized applications are granted access to the critical resources.

5.2.3 Throughput and error rate

The throughput ranged from 4.34 to 4.67 requests per second, indicating stable system performance. Importantly, the error rate remained 0.0% across all configurations, confirming the robustness of the implemented security measures.

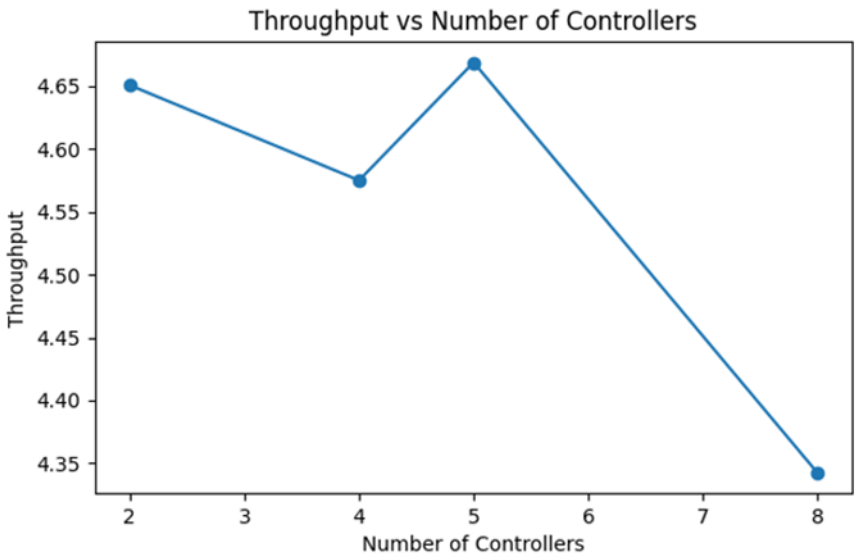

Fig. 9. Throughput of SBAC-SDN

The throughput results further validated the efficiency of the system as it remained consistent across different configurations. The zero-error rate is a critical metric, because it is directly correlated with the reliability and robustness of the system. This is particularly important given that one of the key objectives of this research is to improve the security of the Northbound interface in multi-controller SDN environments.

5.3 Implications and Limitations

The findings of this study have significant implications for SDN and network management. The results strongly suggest that the implemented load balancing technique effectively addresses the scalability challenges faced by multi-controller SDN environments. This demonstrates that even as the complexity of the network increases, the system can maintain efficient resource allocation and handle incoming requests without degrading the performance.

However, it is essential to acknowledge the limitations of the present study. The results are based on experiments conducted in a simulated environment. Although the findings provide valuable insights into the capabilities of the load balancing technique, it is important to recognize that real-world applications may introduce additional complexities and variables that were not accounted for in the simulations. Factors such as hardware limitations, network traffic patterns, and unforeseen anomalies can affect the performance in production environments. Therefore, further testing and validation in real-world scenarios are necessary to fully understand the practical implications of this load balancing technique.

In conclusion, the Results and Discussion section presents a comprehensive assessment of the effectiveness of the load balancing technique, highlighting improvements in scalability and offering visual representations of performance metrics. This study provides a valuable foundation for the implementation of load balancing techniques in multi-controller SDN environments, emphasizing the need for continued research and practical validation in real-world settings.

6. Conclusion

This paper presents a new approach for tackling scalability challenges in multi-controller SDN environments. By implementing a load balancing mechanism, this study has shown how the

Northbound interface, a critical yet often neglected component, can be optimized for better performance. The technique was tested in a simulated setting, and the results indicated significant enhancements in crucial performance metrics including CPU and memory usage, response time, and error rate. The study also briefly touches upon the role of caching mechanisms but keeps the spotlight on the impact of the load balancing technique on scalability. The results of this study contribute to the broader objective of making SDNs scalable, robust, and efficient. Future research will focus on further optimization techniques, including a more in-depth examination of caching mechanisms and the integration of advanced security features. This paper lays the groundwork for future studies aimed at enhancing the scalability, performance, and security of multi-controller SDN settings.

Acknowledgements
We extend our sincere gratitude to our advisors, committee members, and supporting institutions for their invaluable guidance and resources, which significantly aided in the advancement of this research. This research was not funded by any grant.

References

[1] Akhunzada, Adnan, Ejaz Ahmed, Abdullah Gani, Muhammad Khurram Khan, Muhammad Imran, and Sghaier Guizani. "Securing software defined networks: taxonomy, requirements, and open issues." *IEEE Communications Magazine* 53, no. 4 (2015): 36-44. https://doi.org/10.1109/MCOM.2015.7081073

[2] Phaneendra, Y. Sri Deepak, U. Prabu, and Sk Yasmine. "A Study on Multi-Controller Placement Problem (MCPP) in Software-Defined Networks." In *2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, pp. 1454-1458. IEEE, 2023.

[3] Teodor, Malbašić, Petar D. Bojović, Bojović Živko, Šuh Jelena, and Vujošević Dušan. "Hybrid SDN Networks: A Multi-parameter Server Load Balancing Scheme." *Journal of Network and Systems Management* 30, no. 2 (2022). https://doi.org/10.1007/s10922-022-09642-y

[4] Tulloh, Rohmat, Muhammad Iqbal, Mohammad Rifki Baihaqi, and Aditya Bayu Aji Pamungkas. "Load Balancing Implementation Strategy for Various Services in Software Defined Network using ONOS Controller." *International Journal Of Computing and Digital System* (2022). https://doi.org/10.12785/ijcds/1201103

[5] JC, Correa Chica, J. C. Imbachi, and Botero Vega JF. "Security in SDN: A comprehensive survey." (2020). https://doi.org/10.1016/j.jnca.2020.102595

[6] Azodolmolky, Siamak, Philipp Wieder, and Ramin Yahyapour. "Performance evaluation of a scalable software-defined networking deployment." In *2013 Second European Workshop on Software Defined Networks*, pp. 68-74. IEEE, 2013. https://doi.org/10.1109/EWSDN.2013.18

[7] Sokappadu, Bhargava, Avishek Hardin, Avinash Mungur, and Sheeba Armoogum. "Software defined networks: issues and challenges." In *2019 Conference on Next Generation Computing Applications (NextComp)*, pp. 1-5. IEEE, 2019. https://doi.org/10.1109/NEXTCOMP.2019.8883558

[8] Hu, Tao, Zehua Guo, Peng Yi, Thar Baker, and Julong Lan. "Multi-controller based software-defined networking: A survey." *IEEE access* 6 (2018): 15980-15996. https://doi.org/10.1109/ACCESS.2018.2814738

[9] Pietrabissa, Antonio, Lorenzo Ricciardi Celsi, Federico Cimorelli, Vincenzo Suraci, Francesco Delli Priscoli, Alessandro Di Giorgio, Alessandro Giuseppi, and Salvatore Monaco. "Lyapunov-based design of a distributed wardrop load-balancing algorithm with application to software-defined networking." *IEEE Transactions on Control Systems Technology* 27, no. 5 (2018): 1924-1936. https://doi.org/10.1109/TCST.2018.2842044

[10] Rout, Suchismita, Sudhansu Shekhar Patra, Punyaban Patel, and Kshira Sagar Sahoo. "Intelligent load balancing techniques in software defined networks: A systematic review." In *2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC)*, pp. 1-6. IEEE, 2020. https://doi.org/10.1109/iSSSC50941.2020.9358873

[11] Alghamdi, Abdullah, David Paul, and Edmund Sadgrove. "Designing a restful northbound interface for incompatible software defined network controllers." *SN Computer Science* 3, no. 6 (2022): 502. https://doi.org/10.1007/s42979-022-01423-3

[12] Rathnamalala, Damitha Bandara, Hiriyamulla Withanage Pubudu Milan, Karunanayaka Liyanage Inosha Dilshani, and Eranda Harshanath Jayatunga. "Fully integrated software-defined networking (SDN) testbed using open-source platforms." *SN Computer Science* 3, no. 1 (2022): 85. https://doi.org/10.1007/s42979-021-00973-2

[13] Bai, Wenqi, Yihong Hu, Guochu Shou, and Yaqiong Liu. "Implementation of Northbound Interface in Multi-domain Coordinate Control Framework." In *Asia Communications and Photonics Conference*, pp. Su3J-3. Optica Publishing Group, 2018. https://doi.org/10.1109/ACP.2018.8595921

[14] Duy, Phan The, Hien Do Hoang, Anh Gia-Tuan Nguyen, and Van-Hau Pham. "B-DAC: a decentralized access control framework on northbound interface for securing SDN using blockchain." *Journal of Information Security and Applications* 64 (2022): 103080. https://doi.org/10.1016/j.jisa.2021.103080

[15] Liu, Yifan, Bo Zhao, Yang An, and Jiabao Guo. "DACAS: integration of attribute-based access control for northbound interface security in SDN." *World Wide Web* 26, no. 4 (2023): 2143-2173. https://doi.org/10.1007/s11280-022-01130-2

[16] Chung, Joaquin, Eun-Sung Jung, Rajkumar Kettimuthu, Nageswara SV Rao, Ian T. Foster, Russ Clark, and Henry Owen. "Advance reservation access control using software-defined networking and tokens." *Future Generation Computer Systems* 79 (2018): 225-234. https://doi.org/10.1016/j.future.2017.03.010

[17] Saxena, Shivani, and Krishna M. Sivalingam. "DRL-Based Slice Admission Using Overbooking in 5G Networks." *IEEE Open Journal of the Communications Society* 4 (2022): 29-45. https://doi.org/10.1109/OJCOMS.2022.3227591

[18] Jiang, Weiwei, Yafeng Zhan, Guanming Zeng, and Jianhua Lu. "Probabilistic-forecasting-based admission control for network slicing in software-defined networks." *IEEE Internet of Things Journal* 9, no. 15 (2022): 14030-14047. https://doi.org/10.1109/JIOT.2022.3145475

[19] Xia, Rui, Haipeng Dai, Jiaqi Zheng, Hong Xu, Meng Li, and Guihai Chen. "Packet-in request redirection: A load-balancing mechanism for minimizing control plane response time in SDNs." *Journal of Systems Architecture* 129 (2022): 102590. https://doi.org/10.1016/j.sysarc.2022.102590

[20] Praveen, S. Phani, Pappula Sarala, TNS Koti Mani Kumar, Siva Ganesh Manuri, V. Sai Srinivas, and D. Swapna. "An Adaptive Load Balancing Technique for Multi SDN Controllers." In *2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, pp. 1403-1409. IEEE, 2022. https://doi.org/10.1109/ICAISS55157.2022.10010881

[21] Sharathkumar, S., and N. Sreenath. "A Reliable Load Balancing Fault Tolerant Multi-SDN Controller approach in a typical Software Defined Network." *EAI Endorsed Transactions on Internet of Things* 7, no. 26 (2021): e4-e4. https://doi.org/10.4108/eai.2-2-2022.173295

[22] Belgaum, Mohammad Riyaz, Fuead Ali, Zainab Alansari, Shahrulniza Musa, Muhammad Mansoor Alam, and M. S. Mazliham. "Artificial intelligence based reliable load balancing framework in software-defined networks." *CMC—Comput. Mater. Contin* 70 (2022): 251-266. https://doi.org/10.32604/cmc.2022.018211

[23] Kelian, Virakwan Hai, Mohd Nazri Mohd Warip, R. Badlishah Ahmad, Phaklen Ehkan, Fazrul Faiz Zakaria, and Mohd Zaizu Ilyas. "Toward Adaptive and Scalable Topology in Distributed SDN Controller." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 30, no. 1 (2023): 115-131. https://doi.org/10.37934/araset.30.1.115131

[24] Liu, Hean, Xuan Liao, and Baiyan Du. "The applications of nature-inspired meta-heuristic algorithms for decreasing the energy consumption of software-defined networks: A comprehensive and systematic literature review." *Sustainable Computing: Informatics and Systems* (2023): 100895. https://doi.org/10.1016/j.suscom.2023.100895

[25] Aliyu, Aliyu Lawal, Adel Aneiba, Mohammad Patwary, and Peter Bull. "A trust management framework for software defined network (SDN) controller and network applications." *Computer Networks* 181 (2020): 107421. https://doi.org/10.1016/j.comnet.2020.107421

[26] Hu, Tao, Zhen Zhang, Peng Yi, Dong Liang, Ziyong Li, Quan Ren, Yuxiang Hu, and Julong Lan. "SEAPP: A secure application management framework based on REST API access control in SDN-enabled cloud environment." *Journal of Parallel and Distributed Computing* 147 (2021): 108-123. https://doi.org/10.1016/j.jpdc.2020.09.006

[27] Ren, Wei, Yan Sun, Hong Luo, and Mohsen Guizani. "SILedger: A blockchain and ABE-based access control for applications in SDN-IoT networks." *IEEE Transactions on Network and Service Management* 18, no. 4 (2021): 4406-4419. https://doi.org/10.1109/TNSM.2021.3093002

[28] Jiang, Bingcheng, Qian He, Mingliu He, Zhongyi Zhai, and Baokang Zhao. "FACSC: Fine-Grained Access Control Based on Smart Contract for Terminals in Software-Defined Network." *Security and Communication Networks* 2023 (2023). https://doi.org/10.1155/2023/6013270

[29] Zhu, Yong, Xiao Wu, and Zhihui Hu. "Fine grained access control based on smart contract for edge computing." *Electronics* 11, no. 1 (2022): 167. https://doi.org/10.3390/electronics11010167

[30] Algarni, Sultan, Fathy Eassa, Khalid Almarhabi, Abdullah Algarni, and Aiiad Albeshri. "Bcnbi: A blockchain-based security framework for northbound interface in software-defined networking." *Electronics* 11, no. 7 (2022): 996. https://doi.org/10.3390/electronics11070996

[31] Derhab, Abdelouahid, Mohamed Guerroumi, Mohamed Belaoued, and Omar Cheikhrouhou. "BMC-SDN: Blockchain-based multicontroller architecture for secure software-defined networks." *Wireless Communications and Mobile Computing* 2021 (2021): 1-12. https://doi.org/10.1155/2021/9984666
[32] B. Lantz and B. O'Connor. "Mininet Walkthrough - Mininet," Mininet Project.
[33] IBM. "IBM Blockchain based on Hyperledger Fabric from the Linux Foundation," www.ibm.com
[34] Node.js. "About | Node.js," *Node.js Foundation*, (2017).
[35] Express.js. "Express - Node.js web application framework," www.expressjs.com
[36] HAProxy. "HAProxy - The Reliable, High Performance TCP/HTTP Load Balancer," HAProxy.