Journal of Information Systems and Informatics
Vol. 6, No. 1, March 2024 e-ISSN: 2656-4882 p-ISSN: 2656-5935
DOI: 10.51519/journalisi.v6i1.682 Published By DRPM-UBD

An Integrated Framework for Controllers Placement and Security in Software-Defined Networks Ecosystem

Rodney Sebopelo[1], Bassey Isong[2]

[1,2]Computer Science Departement, North-West University, Mafikeng, South Africa
[1]rodney.sebopelo@nwu.ac.za, [2]bassey.isong@nwu.ac.za

Abstract

In the evolving landscape of Software-Defined Networking (SDN), the strategic placement of controllers poses a critical challenge that necessitate a precise balance between network performance and security. This paper presents an integrated framework for enhancing security and performance in SDN by combining controller placement and intrusion detection systems (IDS). Unlike existing solutions which were implemented disjointedly, we propose a holistic approach that leverages the proximity of controllers to network traffic for real-time threat detection, rapid response, and mitigation of security attacks. We employ an advanced clustering model for optimal controller placement, reducing costs and latency while ensuring reliability and balanced loads. In addition, we utilize k-nearest neighbour (KNN) for efficient anomaly detection in our IDS for improved network security. Experimental results confirm the framework's effectiveness in strengthening SDN security and resilience. The enhanced-DBSCAN-based CPP model significantly minimized the cost, and latency, and ensured continuous operation in dynamic SDN environments while the KNN-based IDS shows effectiveness in improving threat detection capabilities, achieving high detection accuracy of 100% on the LAN dataset, outperforming other machine learning models such as Random Forest and Naïve Bayes. The indication is that strategic controller deployment, in conjunction with IDS, can significantly bolster threat detection, response times, and the overall security stance of the SDN environment.

Keywords: SDN, Controller placement, Intrusion detection, Security attacks, Latency, Load balancling, Reliability.

1. INTRODUCTION

The traditional network architecture has become obsolete and inadequate for the current network needs due to its complexity and inherent rigidity in network management and configuration [1, 2]. The rapid development of information technology, such as IoT, cloud computing, big data applications, and so on, has increased the demand for online services that require low latency, high security, and minimal packet loss, leading to the development of a new network architecture called software-defined networking (SDN) [3]. SDN emerged as a revolutionary approach to network management and control as it overcomes the drawbacks of

traditional networks by decoupling the control plane from the data plane. This allows for centralized control and programmability of network devices, paving the way for direct programmability and simplification of the network design, monitoring, and management [4]. The paradigm shift empowers network administrators to effectively manage and allocate network resources which in turn, enhances network performance and flexibility [5]. SDN introduction rids out the need for complex and manual configurations on individual network devices, as network intelligence is delegated to the software-based controllers. The gathering of network intelligence streamlines network management and unlocks the avenues for networking innovation and evolution to support emerging technologies such as IoT, cloud computing, etc. [6, 7]. SDN also introduces new possibilities for network security, as the control plane can make dynamic decisions to detect, mitigate, and prevent network attacks.

SDN is considered a transformative technology that promises to simplify network management, enable innovation, and enhance network security. However, SDNs face challenges such as the controller placement problem (CPP) and security attacks emanating from denial of service (DoS), distributed DoS (DDoS), [8-10], and so on. For the CPP challenge, the SDN control plane can be either centralized with a single controller that oversees multiple data plane devices or distributed with multiple controllers that coordinate with each other [6]. These choices have a huge cost on SDN's reliability, scalability, processing capacity, and security [11]. For instance, centralized control has the drawbacks of a single point of failure, limited processing power, and unsuitability for large-scale networks, while distributed controllers have the advantages of availability, efficiency, scalability, and load balancing, but are confronted with complexity and coordination [12]. Therefore, CPP emanated as the problem of determining the optimal number and location of controllers in a distributed SDN architecture, to achieve the desired performance metrics such as latency, reliability, load balancing, energy efficiency and cost [13-15]. Hence a good controller placement is critical to achieving good quality of service (QoS) and meeting the essential network requirements. Furthermore, security is a critical challenge that affects or hinders mass adoption of SDN. SDN is faced with various security vulnerabilities such as controller vulnerabilities, DoS/DDoS attacks, data privacy concerns, flow table manipulation, insecure southbound communication, inadequate authentication, and authorization [9] and so on. These threats are so severe in hybrid and distributed SDN and require proactive action to address them. For instance, the SDN relies heavily on centralized controllers for network intelligence, and any vulnerabilities in these controllers could expose the entire network to potential attacks or failure, with the centralized control, there's a risk of unauthorized access to sensitive data, such as network configurations and traffic patterns, which could compromise privacy and it is susceptible to DoS/DDoS attacks targeting the control plane, disrupting network functionality by overwhelming the controller with excessive traffic [16].

To address these challenges, a comprehensive approach involving robust and efficient controller placement strategy to enhance SDN's performance, scalability, reliability, and security, as well as intrusion detection systems (IDS), ongoing monitoring and updates to mitigate emerging threats in the evolving landscape of SDN security have been developed and deployed. Particularly, there are various methods and algorithms to solve the CPP, such as integer linear programming (ILP), bio-inspired, heuristic algorithms, and machine learning (ML) techniques [17, 18]. However, an efficient CPP strategy is yet to be achieved. Moreover, over the years several IDS methods have been developed [19-21] to address the security susceptibility of the SDN to threats and attacks. IDS is an essential SDN security component used to monitor network traffic and identify anomalous or suspicious patterns and behaviours [22]. Several types of IDS exist for SDN including signature-based, anomaly-based [19, 23, 24], and hybrid IDS and intrusion prevention systems, which use various techniques, such as ML, deep learning (DL), and even blockchain coupled with effective feature engineering techniques to enhance the accuracy and efficiency of intrusion detection. However, existing solutions to address CPP and IDS in the realm of SDN are disjointed, as both approaches are implemented separately. Previous studies have shown that SDN-based CPP solutions cannot be achieved independently, as they are influenced by other factors such as security issues, controller load balancing issues, and others [10].

Therefore, this paper presents and implements an integrated framework for solving the CPP goals and attack-aware strategy in an SDN-enabled wide area network (WAN) environment. This is to provide a comprehensive solution to SDN challenges involving performance, reliability, scalability, and security. The framework utilizes the enhanced DBSCAN method to optimize the processing latency, load balancing, reliability, and optimal number of controllers needed while the attack-aware strategy uses the KNN to design an IDS. The main contribution of this study is summarized as follows:

1. Highlights the performance and security challenges faced in SDN-enable WAN environments.
2. Proposed an integrated framework that leverages the proximity of controllers to improve SDN security and dependability. The integrated framework has both CPP and IDS modules. While the CPP module ensures optimum controllers' location and allocation, the IDS detects network intrusions and anomalous behaviours in real time.
3. We designed and discussed the components of the proposed framework. The aim is to minimize cost, and latency and maximize reliability and load balancing.
4. We performed simulation experiments and utilized sets of performance measures to evaluate the effectiveness of the proposed framework. We showed promising results in minimizing cost and maximizing security.

The remaining parts of this paper are organized as follows: Section 2 presents the related works, Section 3 presents a literature review on CPP and IDS, and Section 4 presents and discusses the proposed integrated framework. Additionally, Section 5 presents the evaluation and discussion while Section 6 presents the paper's conclusion.

2. RELATED WORKS

This section presents some of the important concepts and related works in terms of existing CPP and IDS approaches in the SDN. We summarize them as follows.

2.1 Controller placement problem

The placement of SDN controllers is a crucial architectural decision that can significantly impact the performance, scalability, and security of the network [25]. The CPP has been considered a challenging optimization problem designed to find the optimal number and location of controllers in the network environment [26]. To achieve the optimal number of controllers to effectively manage the network, factors such as network size, complexity and desired performance metrics are essential while controller location within a given network topology ensures the minimization and maximization of the network objectives but can be constrained by bandwidth limitations, physical infrastructure, security considerations, and so on. Thus, this problem is NP-hard in nature, as there is no efficient algorithm to find the optimal solution in polynomial time. In the SDN, CPP is generally formulated by modelling the networks with multiple nodes and link as an undirected graph, G <S, L, C> to represent the network topology, where S represents the set of switches, L the set of physical links between nodes, where the position of the node is the position of controllers or switches, and C represents the set of controllers. Moreover, n = SUC represents the number of network nodes while k is the number of controllers. Previous studies on SDN mainly focus on finding the optimal value of optimal S-C assignment. Consequently, the predefined objective function is optimized with either a single objective or multiple objectives, such as latency, load balancing, link failure, cost, etc.

In the realm of SDN CPP, a variety of strategies have been developed to achieve optimal placement outcomes. These strategies include heuristic and metaheuristic methods, Integer Linear Programming (ILP), bio-inspired algorithms, and ML-based techniques [19, 27, 28]. The primary objectives of these approaches are to optimize key performance metrics such as latency, reliability, cost, and load balancing [29, 30]. While some methods focus on optimizing a single metric [31, 32], others aim to address multiple metrics simultaneously [33], leading to a spectrum of Pareto-optimal solutions, where improving one metric may compromise another. Minimizing latency is crucial for reducing communication

delays within the network, and it is categorized into average latency for S-C, worst-case S-C latency, and average C-C latency, which includes processing time [34, 35]. Cost minimization is achieved by deploying fewer controllers, thereby enhancing the economic efficiency and feasibility of the network through reduced capital and operational expenditures, as well as lower energy consumption [36]. Maximizing reliability is essential to maintain network functionality in case of controller failures, affecting both network availability and fault tolerance [37]. Strategies to enhance reliability include deploying multiple controllers, establishing multiple control paths, and minimizing control path lengths [22]. Load balancing across controllers is also vital to prevent overloading, which influences the network's scalability, QoS, and robustness [38]. Controller placement is further influenced by traffic patterns and security considerations [39]. Analyzing network traffic patterns helps identify critical points for controller placement to manage traffic effectively. Secure placement of controllers is imperative to mitigate risks such as unauthorized access or control plane attacks.

In recent studies, innovative techniques have been proposed to optimize SDN CPP. Jiang et al. [40] developed a predictive SDN configuration model using neural networks and boosting regression, which proved superior to deep learning models in experimental evaluations. Singh et al. [41] focused on minimizing network latency and enhancing reliability, even with the failure of multiple controllers, by designing a capacitated controller arrangement. Their findings showed that a setup with three controllers could maintain excellent performance despite controller failures. Further, Singh et al. [42] presented a mathematical model for CPP and RCPP, aiming to reduce average latency while considering controller capacity and load as constraints. They compared their varna-based optimization method with established heuristic algorithms like PSO, Jaya, and WOA. Ramya et al. [43] also proposed an ML-based approach to manage network traffic by predicting the optimal number of controllers needed, utilizing the k-medoid algorithm for placement, thereby advancing network automation by integrating SDN and ML technologies.

Benoudifa et al. [44] introduced an intelligent system that employs self-competition, combining tree search with a learned model, to train for optimal game placement in network topologies, focusing on metrics like latency and load balancing. Their experimental benchmarks validated the system's effectiveness. Similarly, Almakdi et al. [45] developed a novel load-balancing method using hierarchical agglomeration clustering and backpropagation neural networks, segmenting network services into groups based on normalized data requirements and evaluating based on network delay, packet loss, and latency. Joshua et al. [46] also applied a clustering technique for optimal controller placement, assessed using the Mininet emulator and silhouette scores to determine the ideal number of controllers for various topologies. In a similar effort, Chen et al. [47] proposed a density-based controller placement algorithm (DCPA) for joint latency

optimization, which segments the network into subnetworks and strategically places controllers to minimize average and worst-case latencies. Tested on eight real network topologies, DCPA demonstrated optimal performance with low time consumption. Moreover, Abeer et al. [48] presented a dynamic mapping based CPP technique for distributed architectures, enhancing system reliability and availability. They combined a heuristic CPP algorithm with metaheuristic particle swarm optimization (PSO) for a hybrid approach that balances reliability and cost-effectiveness. Concurrently, Hui Xu et al. [49] suggested a multi-controller placement strategy using an improved Harris Hawks algorithm, considering local controller load limits, and employing a Sin chaotic map for CPP initialization. Their approach considers total latency, node reliability, link failure rates, and placement costs.

2.2 SDN Security and Intrusion Detection

SDN as a revolutionary network architecture that promotes programmability, flexibility and scalability of network resources is faced with several security challenges and vulnerabilities, both from internal and external threats [3]. These threats target the different layers of SDN architecture. Some of these threats include controller vulnerabilities to attacks such as hijacking, flooding, poisoning, and compromising, etc., designed to take over the network, overload the controller with fake requests, inject malicious rules or commands, or compromise the controller's integrity or availability [50, 51]. The data plane is also susceptible to attacks such as spoofing, tampering, DoS/DDoS, man-in-the-middle, etc. designed to interrupt the network's normal operations, modify, or steal sensitive network/user data, or impersonate legitimate devices or users, etc. [52]. Moreover, the application layer is confronted by attacks such as malicious codes, unauthorized access, data leakage, etc. which aim to illegally access network resources, execute malicious code on the controller or the devices, and even expose sensitive data to unauthorized parties, etc. [53]. Others include the lack of standardized security protocols and practices in different SDN implementations resulting in inconsistencies and breaches in security measures, etc. [54]. Therefore, tackling these challenges in the evolving realm of SDN requires a comprehensive approach that involves robust mechanisms, protocols, tools such as IDSs, and continuing monitoring and updates to mitigate developing threats.

SDN, despite its transformative impact on network resource management through enhanced programmability, flexibility, and scalability, encounters numerous security challenges. These challenges stem from both internal and external threats that exploit vulnerabilities across the SDN's various layers [3]. At the controller level, threats such as hijacking, flooding, poisoning, and compromising attacks aim to seize control of the network, inundate the controller with spurious requests, insert malevolent rules, or undermine the controller's integrity and availability [50, 51]. The data plane is not immune, facing threats like spoofing, tampering, DoS,

and man-in-the-middle attacks, which disrupt normal network operations, alter or pilfer sensitive data, or masquerade as legitimate entities [52]. The application layer is also a target, with risks including malicious code execution, unauthorized access, and data breaches, all of which threaten to illicitly exploit network resources, compromise devices, and leak sensitive information [53]. Compounding these issues is the absence of uniform security protocols across different SDN implementations, leading to security inconsistencies and vulnerabilities [54]. Addressing these multifaceted security concerns necessitates a holistic strategy that incorporates strong security mechanisms, protocols, and tools like IDS, alongside continuous monitoring, and updates to counteract emerging threats. IDSs have been increasingly incorporated into SDN to bolster network security. These systems enable real-time traffic monitoring and analysis, facilitating centralized control, dynamic policy updates, resource optimization, and adaptive security measures [55, 56]. IDS in SDN employ two primary detection methods: signature-based and anomaly-based [57, 58]. Signature-based IDS compares network activities against a database of known attack patterns to identify intrusions [59], but they are ineffective against novel threats. In contrast, anomaly-based IDS detect irregularities by comparing traffic against a baseline of normal activity, offering protection against both known and novel attacks, albeit with a higher risk of false positives [60, 61]. Anomaly detection often utilizes ML techniques, including supervised, unsupervised, reinforcement, and DL to efficiently detect network anomalies [62-66]. ML-based IDS within SDN are designed with modular components that enhance attack detection, including traffic data collection, anomaly identification, mitigation, and reporting functions [67].

In the field of SDN-based IDS, Logeswari et al. [68] introduced the HFS-LGBM IDS, utilizing a two-phase HFS algorithm for optimal feature selection and Light Gradient Boosting Machine (LightGBM) for attack detection. Alzahrani et al. [69] generated datasets with the Ryu controller and Mininet, training various ML algorithms like AdaBoost and DT, with DT achieving a 0.99% F1 score. Shaji et al. [70] developed an intelligent IDS for SDN, utilizing two ensemble ML models to classify DDoS attacks. Their multi-class RF-LR model demonstrated high performance with 99.45% precision and 99.46% sensitivity, outperforming the SVC-RC model. In a similar vein, Seneha et al. [71] introduced the Real-time Anomaly Detection System (RADS), which capitalizes on SDN's capabilities and uses a dynamic threshold for real-time anomaly detection through ML techniques like ARIMA. The system alerts users to malicious activities within 150 milliseconds, using Mininet for SDN topology, Elasticsearch for data storage, and ARIMA, LR, and Prophet models for anomaly detection. Dubem et al. [72] investigated the application of Generative Adversarial Networks (GANs) for anomaly detection in SDN, utilizing the GENI testbed. The study proposed a comprehensive controller-based framework to address common network attacks, demonstrating GANs' potential to identify diverse anomalies. Equally, Yousif et al. [73] combined Ryu, Mininet, and a 1D-CNN to detect and counter DDoS

attacks, achieving a 99.99% detection accuracy rate, outperforming other ML models like LR, RF, SVM, and KNN. Likewise, Mohamed et al. [74] introduced ML-based network intrusion recovery strategy, a novel technique for intrusion recovery in SDN that leverages traffic pattern analysis for strategic backup path selection, significantly reducing recovery time by up to 90% compared to traditional methods.

The studies mentioned represent significant advancements in CPP and IDS within the SDN framework. Although these methods effectively enhance network efficiency, performance, scalability, and security, they are typically deployed in isolation. This separation is a notable issue, and this paper aims to bridge this gap by proposing an integrated framework that concurrently addresses SDN's CPP and security, offering a unified and robust solution.

3. METHODS

This section presents the proposed integrated framework that combines controller placement with intrusion detection in the SDN environment. The development methodology includes theoretical modelling, algorithm creation, and simulation experiments. Initially, a thorough literature review was performed in Section 2, provided insights into current techniques, challenges, and trends in SDN, CPP and IDS. Subsequently, mathematical models were constructed to depict the network, controllers, and IDS, considering factors such as network topology, network traffic, security requirements, and performance metrics. The objective is to ascertain optimal locations for controllers and IDS. To resolve the CPP-IDS integration, ML algorithms were developed, employing clustering for controller placement and supervised learning for detecting network anomalies. These algorithms are intended to enhance SDN performance metrics, including latency, reliability, controller workload, and security response times. Furthermore, the framework was tested in a simulated SDN environment using Mininet to evaluate its performance and effectiveness under various network conditions and attack scenarios. This involved generating network traffic, injecting security attacks, and measuring performance in terms of controller location and allocation, latency, detection accuracy, etc. The framework's structure and its components are depicted in Figure 1 and discussed therein.

As shown in Figure 1, the integrated framework in the SDN environment has the following components: the CPP module, SDN infrastructure and the IDS.

a) CPP module: The CPP module focuses on determining the optimal placement of SDN controllers within the network topology to improve the performance and efficiency of the network. We utilized the enhanced DBSCAN method as the CPP solver for optimum placements. The goal is to minimize both cost and latency as well as balance controllers' load and maximize reliability.

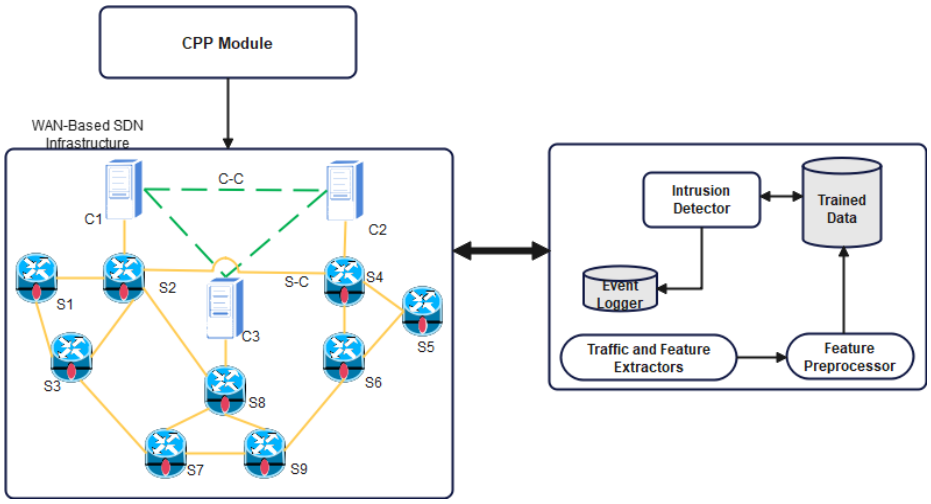

Figure 1. Integrated CPP-IDS architecture

b) SDN Infrastructure: This represents the SDN network infrastructure, consisting of SDN controllers and switches. The SDN controllers take charge of managing the network and communicating with switches using the OpenFlow protocol. They execute control logic and make forwarding decisions or behaviours of the switches. In addition, the network switches are responsible for forwarding traffic based on instructions received from the controller.

c) IDS module: This module is responsible for detecting intrusions and anomalous behaviours within the network traffic. It analyses incoming traffic data to the network to identify potential security threats such as DoS/DDoS, etc. It implements algorithms and techniques/policies for detecting intrusions and security attacks in real time by continuous monitoring of network traffic and data collection from switches. We utilized the KNN algorithm to implement the detection algorithm.

All these components in the framework contribute to optimizing controller placement alongside robust intrusion detection capabilities.

3.1 Controller placement formulation

In the context of SDN, the placement of controllers is a critical architectural decision that significantly impacts network performance, scalability, and security. This section introduces an integrated framework designed to address the challenges of CPP and security within SDN. The framework employs multiple distributed controllers across a WAN to ensure network availability, minimize delays, and manage controller processing loads effectively. It also emphasizes the

strategic placement of controllers in secure locations to reduce the risk of unauthorized access and control plane attacks, thereby maintaining high QoS and user experience. The CPP strategy aims to optimize controller and switch assignments, reduce C-C and S-C latencies, enhance load balancing across controllers, and bolster network reliability to prevent single points of failure. The main goal of the network is articulated and defined to align with these objectives.

$$CPP = min \sum_{k=1}^{n} (C+ L)+ max \sum_{k=1} (C_{LBD}+R+ S) \quad (1)$$

Where:

a) *Cost:* Cost (C) involves the number of controllers to be deployed and the number of switches assigned. The number of controllers is denoted by $\mathbf{N_{ctr}}$ and is achieved via training the collected traffic data to predict the optimal number and location of the controller in the network. It also involved the number of cluster subdomains required in the SDN network.

$$N_{ctr}=min \sum_{k=1}^{n} L(C_k) \quad (2)$$

b) $\mathbf{L(C_k)}$ denotes the optimal number ($\boldsymbol{k}$) and location of the controller in the network, and $\boldsymbol{n}$ = 1,..., N is the number of the placed controllers in the network. Moreover, the number of switch assignments denoted by $N_{Ctr \rightarrow Sw}$ represent the number of data nodes assigned to a specific controller in the network and belong to the members of the specific subdomain network. We maximized the subdomain cluster network from the location of the controller network range for traffic communication and minimized the least members of the core data nodes (β).

$$N_{Ctr \rightarrow Sw}= max \sum_{v \in V}^{k} d(v, z') \quad (3)$$

In Eq.(3), $d(v, z')$ is the maximized range of subdomain network communication from the placed controller to the subdomain cluster members. The least core data nodes are contained within the $d(v, z')$ of the subdomain cluster network without overfitting.

c) *Latency:* In SDN, the latency can be analyzed from different approaches: (1) processing latency is the time both at the controller side (processing requests and generating instructions) and at the switch side (processing and applying instructions). These are measured by S-C – average and S-C – worst and the S-C latency. (2) C-C latency is the communication latency between different SDN controllers in the distributed architecture. We aimed to minimize both S-C and C-C latencies.

d) *Load balancing (C_{LBD}):* This is achieved based on mathematical analysis and comparison of the controllers' load with its capacity. Once the load on the controller exceeds its capacity i.e., $C_{LBD} > \varphi(y)$, the load is shared among other available and active controllers in the network.

$$C_{Capacity} = \frac{Collected\ traffic\ statistics\ \ \varphi(x)}{Oveall\ placed\ controller\ (\varphi(z''))} = \varphi(y) \tag{4}$$

Eq.(4) computes the controllers' capacity $\varphi(y)$, $\varphi(x)$ is the collected traffic data and $\varphi(\mathrm{z}'')$ denoted the overall number of placed controllers in the network.

$$C_{LBD} = \frac{\beta(x) \times d(v,z')}{Collected\ traffic\ statistics\ \ \varphi(x)} \tag{5}$$

Eq.(5) formulates the threshold load on the controller based on the number of the received traffic data, $\beta(x)$ denoted the least core data nodes contained by the $d(v,z')$. Thus, the threshold is exceeded, and the load is balanced using Eq.(2).

e) *Reliability (R):* We computed the C-C and S-C time processing using the multi-path alignment method. This is to forward the load on the failed controller to another active or available controller. This is achieved by minimizing the time required by the controller to control the forwarding load, the time required to forward the load between the controllers should not exceed the required time in the network.

$$MCP_{\ Ctr[1]\rightarrow Ctr[2]} = MCP_{\ Ctr[2]\rightarrow Ctr[3]} = \theta \tag{6}$$

f) *Security:* Security (S) in eq. 1 represents security which is one of the core objectives to maximize in this work. We aim to maximize SDN security by placing controllers strategically to minimize control traffic overhead and within a secure place to mitigate security risks associated with attacks such as DoS/DDoS. This is achieved by the integration of a robust ML-based IDS in the SDN environment.

3.2 Optimum controller placement algorithm

This subsection presents the proposed CPP algorithm that directly impacts the SDN network's performance, scalability, and resilience in terms of maximum security, load balancing, reliability and minimum cost and latency. that is, the placement algorithm satisfies the number of the required controllers, switch assignments, processing latency, C-C and S-C, controller processing latency or load balancing, and reliability. We utilized the enhanced DBSCAN algorithm [75, 76], motivated by some benchmarked performance from the widely known

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

clustering algorithms [77, 78]. Moreover, we decoded the collected LAN traffic statistics ($v_1, v_2, v_3 \ldots . v_k$) $\in$ V using Wireshark software. The proposed optimum placement algorithm is captured in Figure 1.

Algorithm 1: Algorithm for achieving the controller placement

Input: Topology network nodes ($v \in V$)

Output: (N_{ctr}, $N_{Ctr \to Sw}$, C_{LBD} network topology)

1. Read the topology nodes and find the shortest path
2. Compute the N_{ctr}, $N_{Ctr \to Sw}$, C_{LBD} and adjacent nodes
3. Verify the node belonging to the subdomain network
4. Compute the **(d (*x's*, y) == k) on the network nodes**
5. For all of $v \in V$
6. node number = 1
7. else node number = 0
8. If node number = 0
9. Store and calculate the distance of the node = 0
10. Obtain the adjacent topology belonging to the node number = 0
11. Assign each node to the closest cluster centroid
12. Search for N_{ctr}, $N_{Ctr \to Sw}$, C_{LBD} new solution
13. Repeat statement (5)(6)(7)
14. End

Figure 2. Controller placement algorithm

As shown in Figure 2 derived the model technique for the number of the required controllers, switch assignment, load balancing, and reliability. We first find the shortest path between each pair of the nodes calculate the required distance, and find the controller location that minimizes the processing latency, the required number of the controller was formulated using equation 2. Moreover, we derived the processing latency an (i) and b(i) using steps 2 and 3 for the time required to process the packet in the network. We further derived the controller-to-switch assignment which is the assigned path from one controller to the switch that minimizes the processing latency using equations 4 and 5. That is capacity and load on the controller formulated using steps 6 and 7, where *B(x)* is the collected traffic statistics, $d(v, z')$ derived the distance path of the controller to switch, and $(\varphi(z''))$ is the derived optimally placed controllers. Furthermore, we derived the multi-path alignment paths using the computation of equation 6, that is the time required to forward the load between the controllers was compared to the availability of the active and placed controllers. Once the approach is performed, we derived the intrusion detection on the nodes and was derived using algorithm 2 in Figure 3.

3.3 Intrusion detection

This section presents the design of the proposed IDS model for ensuring maximum security in the SDN environment. The proposed detection architecture is anomaly-based flow detection, automatic and modular, leveraged by SDN's architectural functionalities such as the OpenFlow and ML technique. It automates network monitoring in real-time and the detection of anomalous traffic flows or malicious behaviours by alerting the administrators and invoking mitigative actions to thwart the attack in the network. The proposed IDS model in this work provided a positive effect and increased security measurement in the SDN environment [22]. Though several IDS and security countermeasures have been proposed and developed for the SDN, many of these systems are less efficient, have low precision, are not effective in real-time operation and exert so much burden on the controller in terms of network monitoring, flow table information updating, anomalous flow detecting, etc. To address this challenge, our proposed model was based on the application plane during anomaly detection. The IDS model shown in Figure 1 has several interrelated components that contribute to the detection of anomaly flow attacks in the SDN. This includes modules for the collection of traffic statistics, anomaly detection, mitigation, and reporting.

3.3.1 Traffic collection module

In this module, we showed how the traffic was read in the form of $(v_1, v_2, v_3 \ldots . v_k) \in V$ where $\mathbf{V}$ is the overall collected traffic statistics sent and received during the traffic operation. Moreover, v_k is the i^{th} traffic flow selected features extracted as shown in Table 1.

Table 1. Extracted features from the collected V_i

Extracted feature	Description
Deviceid	The device used for the transmission of the traffic flow
Bytes	The bytes associated with the transmitted flow data
Packets	The size associated with the transmitted data
Duration	The time associated with the delivered data
Priority	The priority of the data in the network
TableId	The identity of the table associated with the delivered data
Payload	The payload associated with the data flow

3.3.2 Anomaly detection module

This module deals involves traffic classification for the detection of anomalous flow. It utilizes the KNN algorithm, an ML technique to identify anomalous events in the network by performing deep packet inspection and analysis on collected traffic flow. Detailed information about the injected anomalous flow is

shown in Table 2. Metrics were used to evaluate and correlate classified traffic, and the identified attacks were sent to the anomaly mitigation module.

Table 2. Anomaly attack packet details

Information	Description
ping	Used to send the ICMP echo request packets to the target host or the IP address.
-c	Number of the sent data packets in the network
-s	The sent packet size in the network
Target IP address	The information about the target communication address
Time	Round trip time about the packet details
Packet loss	Information about the transmitted packet

Moreover, the model categorized the detection into normal and anomalous flows. As shown in Table 3, "+/0" denotes the normal traffic, while "-/1"is abnormal in the network. This traffic classification was enabled by adopting the described function and generating descriptive statistics. The employed data were split into training and testing, that is 80% for the training and 20% for testing. This is because the higher the classification accuracy signified the dispersed probability distribution, while the lower denoted the concentration of the data distribution.

Table 3. Normal and anomaly effects on the traffic feature

Type	Affected feature traffic pattern			
	Bytes	Packets	Duration	Priority/ Table id
Normal	+/1	+/1	+/1	+/0
Anomaly	-/0	-/0	+/1	-/1
Anomaly	+/1	+/1	-/0	+/1
Anomaly	-/0	-/0	+/1	-/1
Anomaly	-/0	-/0	-/0	-/1

Furthermore, the trained traffic features were subsequently fed to the anomaly detection module, an efficient anomalous flow detection which uses the KNN algorithm. KNN is a supervised ML algorithm used for classification tasks, where it can assign labels to unlabeled data points based on the labels of neighbouring data points in the training dataset [79, 80]. In this study, we trained and correlated the dataset identifying anomalous events and forwarded the detection report to the mitigation module. For anomalous event detection, the Euclidean distance is applied. We computed and defined the threshold value that determined the anomaly flow attack, and defined the rules to trigger the alert when the traffic exceeds a certain threshold. That is the flow rules above the threshold are considered as the potential attack. Once identified as a potential attack, the controller acts and reports the malicious data. The KNN-based anomaly detection algorithm is shown in Figure 3

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

As shown in Figure 3, we derived our intrusion detection model technique, Finding *k* the number of the network nodes belonging to the topology $v \in V$, we computed the square of the distance of the adjacent nodes belonging to the topology output, we calculated the threshold of the topological network nodes and search for all the network nodes belonging to the anomalous nodes, we obtained the network nodes belonging to the number = 1, that is the DDoS potential attack, we removed the network node = 1 from the overall topological network, we performed the statement 3 and 4 based on the threshold segregating the network topological network nodes, then, we searched for the adjacent network nodes number = 0 and allocated the nodes to the network subdomain, we repeated the statement (2)(3) and (7) on the adjacent and new network nodes.

Algorithm 2: Distributed intrusion detection algorithm

Input: Topology network nodes ($v \in V$)

Output: $v \in V$ based attack topology nodes

1. Compute k belongs to the network nodes
2. Calculate the square distance **d (x's, y) of the adjacent nodes**
3. Calculate the threshold of the adjacent nodes
4. Obtain the adjacent network node number = 1
5. Remove the node = 1 out of the topology network
6. **Repeat statement (3)(4) on the network nodes = 1**
6. Search k for all the network adjacent nodes = 0
7. Allocate the node number = 0 to the adjacent cluster network subdomain
8. Repeat statement (2)(3)(7) for all the new network nodes = 0
9. End

Figure 3. Intrusion detection algorithm

3.3.3 Mitigation module

This presents the applied mitigation actions on the traffic flow associated with anomalous flow detection. The policies include DropFlowData - dropping of the traffic flow data, ExcludeFlowData - exclusion of the flow data, DropEntryFlow-dropping of entry flow, and PushControllerLoad - pushing of the controller's load to the neighbouring controllers. With these actions, DropFlowData ignores and deletes any flow associated with the anomalous traffic flow in the SDN while DropEntryFlow blocks the entry flow communication from the specific host to the specific services in the destination IP address. Moreover, the ExcludeFlowData ignores the entry flow after distributed IDS make an alert about the anomaly attack in the network, and PushControllerLoad pushes the load on the attacked controller to the neighbouring or new controller. To ensure the above actions are enforced correctly, Table 4 summarizes the function policies implemented in Figure 4 on the SDN environment to mitigate detected anomalous flow in the SDN. Figure 4 presents the algorithm that implements the formulated

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

function policy used to manage the effective detection and necessary mitigation actions taken in the SDN network.

Table 4. Implemented function policy

Criteria	Match criteria	Action
DropFlowData	Utilized controller and defined the flow table in switches, the rule dictates how the network traffic is processed.	Drop the flow rule/s associated with the identified DDoS attack. *controller > drop the source IP flow rule associated with the ID anomaly attack.*
ExcludeFlowData	Set the matching packet rule and identify any related packets associated with the attack attributes.	Exclude the identified flow rule associated with the DDoS flow rule required to duplicate the traffic flow rule. *controller > exclude the id flow rule required to mimic the source and destination IP address.*
DropEntryFlow	Dictate any flow rule matching the IP address associated with the anomaly attack in the network.	Drop the source IP address flow matching the entry flow data packet related to the anomaly attack. *controller > drop the source IP address related to the ID anomaly attack*
PushControllerLoad	Evaluate controller failure rate and how flow rules are forwarded from the source to the destination.	Push the load of the affected controller to the target controller.

Algorithm 3: Enforced function policy algorithm

Input: Receive the ***x's*** – traffic attribute flow stats
Output: drop / exclude/push flow data

1. Receive the input traffic flow stats *x's*
2. Characterised *x's* → y - payload
3. Knn fits *y* → *x's* – flow rules
4. if (**x's, y) → d (***x's***, y)** == *k*
5. forward **k** → anomalous storage event *(A)*
6. *while (***d (***x's***, y) !=k**
7. *forward* **d (***x's***, y)** → *N* - x's → normal
8. if (**d (***external'k***, y) ==k)** ∈ *e'k* → *(external input attack)*
9. **drop entry d (***e'k***, y) == k**
10. **forward** *e'k* → anomalous storage event *(e'k)*
11. else if (*d(internal'k, y) ==k)* ∈ *inter'k* → *(internal input attack)*
12. drop flow *d(internal'k, y) == k*
13. forward internal'k → anomalous storage events *(A)*
14. End

Figure 4. Policy enforcement algorithm

3.3.4 Reporting module

Once the anomaly attack is mitigated, a report is generated of the anomalous events, recorded, and stored. The stored events are used to predict any identical anomalous event that wants to invade the resources of the network. Also, any new identical anomaly flow and services associated with the recorded anomalous event are prioritized and mitigation action policy is prioritized before causing any harm to the network or its resources.

4. RESULTS AND DISCUSSION

This section presents the evaluation of the proposed system that integrates CPP and IDS to demonstrate its performance and effectiveness in terms of minimum delay and cost, maximum security, load balancing and fault tolerance in the SDN network. The evaluation was achieved based on simulations, and the results obtained were presented and analyzed.

4.1 Experimental simulation setup

To evaluate the proposed system, implementations were performed on the SDN-enabled environment. The simulations were run on a PC with the Ubuntu 20.04 LTS OS on the VM, a core ™ i7-10610U CPU processor and 16 GB RAM. We designed a network topology consisting of connected switches with multiple host devices, and controllers. We employed the address, with the subnet mask, default gateways, DNS server, and link-local address. The designed network topology is built to simulate the SDN WAN-enabled network that will be using the controllers to monitor the traffic communication, and OpenFlow switches to simulate the SDN virtual environment. We simulated the traffic flow in real time and identified the DDoS attack in the SDN environment. The results were collected based on the run-time simulation process of the traffic nodes' communication. The simulation parameters used are shown in Table 5 and evaluation metrics.

Table 5. Simulation parameters

Simulation Parameter(s)	Values
Simulation area - S_A	SDN enabled environment
Simulators - S_T	Mininet, ONOS web UI, VM, python
Number of nodes N_A	20
Number of the switches and controllers - S_C	6, 3
Controllers - C_N	ONOS
SDN domain - S_D	5
Simulation time - S_T	300s
Traffic type - T_T	UDP and TCP
Size of flow table size - S_T	120 entries

The evaluation metrics used are the accuracy, precision, and F1-score [70], [73] which are all based on the confusion matrix where TP is the true positive, TN the true negative, FP the false positive and FN the false negative in the classification. Accuracy computes the ratio of the number of correctly classified instances (attacks and non-attacks) to the total number of instances. Recall measures the proportion of attacks correctly identified. This represents the system's ability to detect attacks effectively and avoid false negatives. Precision symbolizes the system's ability to avoid FPs and is the proportion of instances identified as attacks/intrusions which are true. F1-score is the harmonic mean of both precision and recall. Based on the confusion matrix, the metrics are defined as shown in Eq. 7 to 10 [70], [73]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

$$F - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{10}$$

4.2 Results and Analysis

This subsection presents the results and analysis of the simulations conducted to evaluate the performance of the proposed CPP-IDS framework. The results are presented twofold: placement of controllers and intrusion detection.

4.2.1 Placement of controllers

This subsection presents the evaluation of the proposed CPP in the SDN environment. The experiment was done on real-time simulation using the Mininet, ONOS web UI network simulation tool etc. The results obtained are based on the number of controllers, C2S assignments, latency, controller load and reliability.

1) *Number of the controllers:* Based on the designed network topology, C_3 optimal controllers were recommended by our model to be deployed in the network. About 20 host devices were required to achieve 18 flow rules and 6 connected flow switches while 2GB CPU memory was utilized based on the generated traffic. Based on the deployed controllers, 30 active links were recorded. Table 6 summarizes the results of the deployed number of the controller consisted of the SDN networks.

Table 6. Deployed number of the controller

Criteria	Value 3	Value 2	Value 1
Total host devices	20	10	5
OvS	6	3	1
Number of flow rules	18	10	3
CPU memory	64GB	64GB	64GB
CPU utilisation	30%	10%	10%
Controller response time	50ms	30ms	20ms
Links	30	20	6
Device status	Reachability	Reachability	Reachability

2) *Controller to switch assignments:* Our model found that 1 assigned switch generated fewer flow rules as compared to 2 and 3 switches. This shows that the higher the number of switches, the higher the generation of the flow rules in the SDN networks. Thus, a single switch recommended a minimum of 3 flow rules, 2 switches recommended a minimum of 10 flow rules and 3 switches recommended 18 flow rules. Table 7 presents the required S-C to be employed in the clustered subdomain network without overfitting the active switches in the network.

Table 7. Summarized controller-to-switch results

Controller container	1	1	1
No. of connected switch	6	2	1
Open flow rules	18	10	5
Connection	True	True	True
Network interface	172.192.10.101	172.192.10.101	172.192.10.101
Container	172.168.10.1	172.168.10.2	172.168.10.3
Links	30	20	6
OpenFlow Port	8181	8181	8181

3) *Latency:* In terms of latency, our model achieved a promising latency. It significantly minimized the C-C and S-C latency which are the time taken for computational tasks between controllers and between controllers and switches. Table 8 presents the results obtained in terms of processing latency and C-C latency during SDN traffic communication.

Table 8. Network latency

Criteria	Processing latency (s)	Values
Inter-latency $a(i)$ = S-C	ONOS to switch	0.958s
Intra-latency *b(i)* = C-C	ONOS to ONOS	1.891s

4) *Load balance:* Our model computed load balancing and recommended a load of about 0.069lb to accommodate 3 flows, while 0.027lb for 10 flows, and

0.833lb for 18 flows. This validated the saying that the load on the controller should not exceed its capacity. When the load on the controller is greater than its capacity the load was shared with other available and active controllers. Table 9 presents the summarised results of the load balancing and the capacity of the controller.

Table 9. Load balancing in the network

Criteria	Value 1	Value 2	Value 3
Controller container	1	2	3
Host devices	5	10	20
Traffic flows	3	10	18
Traffic statistics	72	72	72
Controller capacity	72cc	36cc	24cc
Load balancing	0.069lb	0.027lb	0.833lb
Open flow switch	1	3	6

5) *Reliability:* Reliability was achieved by computing the multi-paths between the controllers. Due to attacks on the controller, the load one controller was reassigned to other available and active controllers. Moreover, we excluded the attacked controller from the traffic communication; thus, the newly placed controller location maximized the distance θ = 1.4m. Table 11 summarises the reliability condition once the controller is affected by an anomaly attack.

Table 10. Summarized reliability results θ

Criteria	Value 1	Value 2	Value 3
Controller container	1	2	3
Host devices	0	10	20
Traffic flows	0	10	18
OVS	0	3	6
Status reliability	0	1	1

4.2.2 Network anomalies detection

This section presents the evaluation of the proposed IDS for maximizing security in the SDN ecosystem. Detection is an anomaly flow-based model that leverages the traffic flow of the network to detect attacks. Based on the designed network topology**,** we generated the traffic that includes the device ID, bytes, packet, length, and payload. The traffic flow stats were grouped into normal and anomalous flows where "+/1" denotes the normal traffic outcome, and "-/0"is the abnormal in the network. Moreover, an attack is detected if the 1.096 threshold value is exceeded in the network. Table 11 presents the obtained results during the traffic classification and detection.

Table 11. Classification and detection results

Criteria	Results
Classification accuracy	100%
Precision	100%
F1 - score	100%
n_neighbor	5
Anomaly	8
Threshold	1.096
Threshold percentile	95%

To further evaluate the effectiveness of the proposed IDS model in the SDN by injecting DDoS attacks into the network. We utilized the Ubuntu environment terminal and ran a set of multiple commands to launch DDoS attacks on the selected nodes of the designed SDN. We evaluated how the launched packets impacted the traffic data communication as well as how the KNN model technique identified the DDoS attacks based on the threshold percentile. We launched different sizes of packets to the various destination IP address nodes. That is 64 bytes of DDoS packet attacks were directed to the destination IP addresses that include: 10.0.0.2, 10.0.04, 10.0.0.3, 10.0.0.5, and 10.0.0.14. Table 12 presents the set of injected DDoS traffic commands in the SDN environment.

Table 12. Summarized DDoS Anomaly Attack details

DDoS attack	Description
ping -c 5 -s 64 10.0.0.2	Send the 5 packets of 64 bytes to the host IP address 10.0.0.2
ping -c 6 -s 64 10.0.0.4	Send the 6 packets of 64 bytes to the DST IP address 10.0.0.4
ping -c 4 -s 64 10.0.0.3	Send the 4 packets of 64 bytes to the host IP address 10.0.0.3
ping -c 2 -s 64 10.0.0.5	Send the 2 packets of 64 bytes to the host IP address 10.0.0.5
ping -c 2 -s 64 10.0.0.14	Send the 2 packets of 64 bytes to the DST IP address 10.0.0.14

The proposed IDS model was effective against all forms of network anomalies. We evaluated mitigation policies based on the predefined policy to accurately identify DDoS packet attacks in the SDN subdomain network. The proposed policy in Table 4 is on the identified DDoS anomaly packet attack. The model performed the mitigation action and monitored the state of our SDN environment in the real-time simulation. The enforced policy was used to take actions against alerted DDoS attacks in the network in terms of the number of data packets suspended, excluded, pushed, and dropped during the failure because of the DDoS attack in the SDN environment. We also evaluated how many flow rules were required to be migrated in the network based on the failed controller. Table 13 presents the results obtained after the mitigation action policy was invoked on the injected flow data.

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

Table 13. Enforced policy results.

Criteria	Results
DropEntryFlow attack	5
ExcludeFlow attack	2
DropFlow attack	2
PushControllerFlow	3
No.of controllers	3

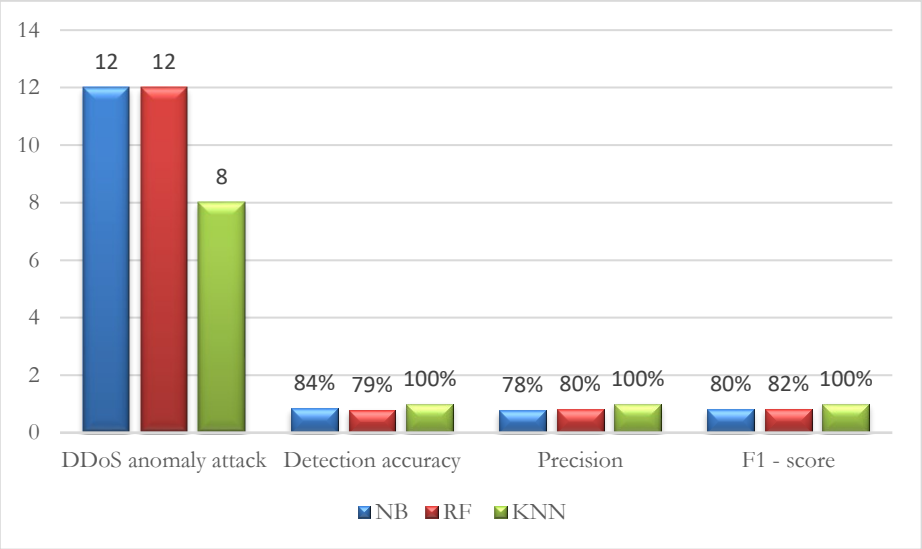

Figure 5. Anomaly detection

Furthermore, the effectiveness of the KNN-based IDS model was compared with two other ML techniques such as naïve Bayes (NB) and random forest (RF). The essence was to evaluate its detection accuracy and the pattern of the traffic monitoring and classification on the assumed SDN data packets. As shown in Figure5, the results show that the KNN algorithm outperformed NB and RF in all three criteria, achieving perfect detection accuracy for DDoS attacks, precision, and F1-score with 100% respectively. NB, on the other hand, performs reasonably well but has slightly lower accuracy, 84%, and precision 79%. Moreover, RF also performs well, especially in terms of precision, with 80% and F1-score with 82%. Other factors such as computational efficiency and scalability are critical and will be considered to further identify the most suitable algorithm. However, in terms of the number of DDoS anomaly attacks detected, both NB and RF detected 12 attacks while KNN detected only 8 attacks. Thus, although KNN is the best in terms of detection accuracy, precision, and F1 score, both NB and RF are more effective in identifying DDoS attacks.

4.3 Discussion

This paper introduces a unified framework for CPP and security within the SDN ecosystem. By integrating CPP with IDS, we optimize network security, enhance threat detection, and improve network performance. Our approach employs ML approaches: a clustering-based CPP and a KNN-based IDS to strategically place controllers, facilitating real-time traffic analysis and more effective anomaly detection. This integration reduces latency in controller-device communication, enabling quicker detection and response to security incidents, thereby minimizing their impact on network performance and user experience. Cost efficiency is also achieved by optimizing resource use and reducing operational expenses related to controller and switch deployment, as well as IDS management. Strategic controller placement, informed by traffic analysis, reduces the number of necessary controllers while extending coverage and detection capabilities. Load balancing is attained by evenly distributing network traffic across controllers, considering traffic volume, processing capacity, and utilization, which enhances computing resource efficiency and network responsiveness. The model also improves fault tolerance and resilience by distributing control functions across multiple controllers, ensuring service continuity in the event of controller failure. Furthermore, the integration facilitates centralized security management, enabling consistent security policy enforcement and more effective threat detection and response, thus improving visibility, control, and coordination of security operations. Improved scalability, although not explicitly addressed, is an additional benefit of this integrated approach.

To assess the proposed framework's efficacy and performance, simulation experiments were conducted, and the results were analyzed. These results affirmed the framework's capability to augment detection accuracy, reducing latency and costs, as well as load balancing and reliability. A comparative analysis with existing literature, detailed in Table 14, focused on controller placement's impact on SDN performance, controller deployment, and anomaly attack detection. The analysis highlighted that current CPP implementations lack security integration. Unlike disjointed IDS and CPP strategies found in these studies, this paper uniquely combines both to offer a comprehensive solution in the SDN environment.

Table 14. Comparison with Related Works

Author	Method & Algorithm	Evaluation	Challenges
R Anusuya et al. [82]	KNN, SVM, DT, and RF	SDN environment	Network scalability & mitigation of DDoS attack
Sultan Zavrak et al. [83]	SAnDet -SDN anomaly detector,	SAnDet, an Area Under the ROC curve	Anomaly-based intrusion detection

Author	Method & Algorithm	Evaluation	Challenges
	neural networks (RNN), EncDecAD		
Avtar S et al. [84]	Hybrid ML model in the SDN network	SDN environment	CPP objective & DDoS detection and mitigation
Najmun Nisa et al. [85]	Packet filtration and ML classification techniques, SVM and KNN	SDN network	CPP and DDoS detection
Jiang Liu *et al.* [86]	Clustering algorithms	SDN optimal and sub-optimal solution	Multiple objectives optimization
Guodong Wang *et al.* [87]	Optimized k means	SDN load balancing, resistance/reliability awareness	Decrease the complexity of the controller placement
Lei *et al.* [88]	Heuristic algorithm	SDN topology, Propagation, transmission delay	Control plane minimization
This study	SDN-based enhanced DBSCAN & KNN IDS technique	SDN topology based open daylight, Ryu, Floodlight's optimal controller	SDN processing latency, reliability, load balancing, optimal controller, controller to switch assignment

5. CONCLUSION

In this paper, we have designed an integrated framework for CPP and IDS in the SDN environment to provide a comprehensive solution to SDN security and performance. The integrated framework utilized clustering and classification techniques to achieve controller placement and to detect the anomaly respectively in the network. This integration is crucial for reducing costs and latency, while maximizing security, load balancing, and fault tolerance. The effectiveness and performance were evaluated based on simulation experiments. The results obtained demonstrated that the integration of ML-based strategies can provide comprehensive solutions in terms of security and performance in the SDN environment. The IDS model was further compared with other ML techniques such as NB and RF where KNN excels in all criteria. Moreover, this work was also compared with CPP solutions in the literature. Based on the findings, show that by strategically deploying controllers closer to critical network slices and integrating them with IDS, organizations can improve threat detection capabilities, improve incident response times, and strengthen overall network security posture in dynamic and evolving environments. Our future work will focus on incorporating additional ML techniques, such as DL and ensemble methods, and improving feature engineering to further improve the model's detection accuracy.

REFERENCES

[1] N. Makondo, H. I. Kobo, and T. E. Mathonsi, "The latest developments in Software Defined Networking: Adoption rate and challenges," in *2023 IEEE AFRICON*, 2023, pp. 1-6: IEEE.

[2] F. Liu, G. Kibalya, S. Santhosh Kumar, and P. Zhang, "Challenges of traditional networks and development of programmable networks," in *Software defined Internet of everything*: Springer, 2021, pp. 37-61.

[3] A. Shaghaghi, M. A. Kaafar, R. Buyya, S. J. H. o. C. N. Jha, C. S. Principles, and Paradigms, "Software-defined network (SDN) data plane security: issues, solutions, and future directions," pp. 341-387, 2020.

[4] A. Yazdinejadna, R. M. Parizi, A. Dehghantanha, and M. S. J. C. N. Khan, "A kangaroo-based intrusion detection system on software-defined networks," vol. 184, p. 107688, 2021.

[5] S. Haider *et al.*, "A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks," vol. 8, pp. 53972-53983, 2020.

[6] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. J. I. I. o. T. J. Bangash, "An in-depth analysis of IoT security requirements, challenges, and their countermeasures via software-defined security," vol. 7, no. 10, pp. 10250-10276, 2020.

[7] X. Hou *et al.*, "Reliable computation offloading for edge-computing-enabled software-defined IoV," vol. 7, no. 8, pp. 7097-7111, 2020.

[8] B. B. Gupta and A. Dahiya, Distributed Denial of Service (DDoS) Attacks: Classification, Attacks, Challenges and Countermeasures. CRC Press, 2021.

[9] K. B. Virupakshar, M. Asundi, K. Channal, P. Shettar, S. Patil, and D. J. P. C. S. Narayan, "Distributed denial of service (DDoS) attacks detection system for OpenStack-based private cloud," vol. 167, pp. 2297-2307, 2020.

[10] M. H. Ali *et al.*, "Threat analysis and distributed denial of service (DDoS) attack recognition in the Internet of things (IoT)," vol. 11, no. 3, p. 494, 2022.

[11] P. Krishnan, K. Jain, A. Aldweesh, P. Prabu, and R. J. J. o. C. C. Buyya, "OpenStackDP: a scalable network security framework for SDN-based OpenStack cloud infrastructure," vol. 12, no. 1, p. 26, 2023.

[12] M. Rahouti, K. Xiong, Y. Xin, S. K. Jagatheesaperumal, M. Ayyash, and M. J. I. A. Shaheed, "SDN security review: Threat taxonomy, implications, and open challenges," vol. 10, pp. 45820-45854, 2022.

[13] A. Singh, G. S. Aujla, R. S. J. S. C. I. Bali, and Systems, "Container-based load balancing for energy efficiency in software-defined edge computing environment," vol. 30, p. 100463, 2021.

[14] M.-L. Chiang, H.-S. Cheng, H.-Y. Liu, and C.-Y. J. C. C. Chiang, "SDN-based server clusters with dynamic load balancing and performance improvement," vol. 24, pp. 537-558, 2021.

[15] K. A. Jadhav, M. M. Mulla, and D. Narayan, "An efficient load balancing mechanism in software defined networks," in *2020 12th international conference on computational intelligence and communication networks (CICN)*, 2020, pp. 116-122: IEEE.

[16] M. R. Belgaum, S. Musa, M. M. Alam, and M. M. J. I. A. Su'ud, "A systematic review of load balancing techniques in software-defined networking," vol. 8, pp. 98612-98636, 2020.

[17] A. Ahmad, E. Harjula, M. Ylianttila, and I. Ahmad, "Evaluation of machine learning techniques for security in SDN," in *2020 IEEE Globecom Workshops (GC Wkshps*, 2020, pp. 1-6: IEEE.

[18] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez, and J. M. J. I. A. Arco, "A survey on machine learning techniques for routing optimization in SDN," vol. 9, pp. 104582-104611, 2021.

[19] Y. Hande and A. Muddana, "A survey on intrusion detection system for software defined networks (SDN)," in *Research Anthology on Artificial Intelligence Applications in Security*: IGI Global, 2021, pp. 467-489.

[20] K. M. Sudar and P. J. I. J. o. I. E. Deepalakshmi, "Comparative study on IDS using machine learning approaches for software defined networks," vol. 7, no. 1-3, pp. 15-27, 2020.

[21] T. Jafarian, M. Masdari, A. Ghaffari, and K. J. I. J. o. C. S. Majidzadeh, "Security anomaly detection in software-defined networking based on a prediction technique," vol. 33, no. 14, p. e4524, 2020.

[22] Y. Maleh, Y. Qasmaoui, K. El Gholami, Y. Sadqi, and S. J. J. o. R. I. E. Mounir, "A comprehensive survey on SDN security: threats, mitigations, and future directions," vol. 9, no. 2, pp. 201-239, 2023.

[23] K. Muthamil Sudar, P. J. J. o. I. Deepalakshmi, and F. Systems, "An intelligent flow-based and signature-based IDS for SDNs using ensemble feature selection and a multi-layer machine learning-based classifier," vol. 40, no. 3, pp. 4237-4256, 2021.

[24] N. Mazhar, R. Salleh, M. A. Hossain, M. J. I. J. o. A. C. S. Zeeshan, and Applications, "SDN based intrusion detection and prevention systems using manufacturer usage description: A survey," vol. 11, no. 12, 2020.

[25] S. Ahmad, A. H. J. J. o. N. Mir, and S. Management, "Scalability, consistency, reliability and security in SDN controllers: a survey of diverse SDN controllers," vol. 29, pp. 1-59, 2021.

[26] M. Ali *et al.*, "Performance and Scalability Analysis of SDN-Based Large-Scale Wi-Fi Networks," vol. 13, no. 7, p. 4170, 2023.

[27] B. Sapkota, B. R. Dawadi, and S. R. J. E. R. Joshi, "Controller placement problem during SDN deployment in the ISP/Telco networks: A survey," vol. 6, no. 2, p. e12801, 2024.

[28] J. P. Martin, "Orchestration Mechanisms for Enabling Distributed Processing In the Fog Computing Environment," National Institute of Technology Karnataka, Surathkal, 2021.

[29] A. J. W. P. C. Javadpour, "Providing a way to create balance between reliability and delays in SDN networks by using the appropriate placement of controllers," vol. 110, pp. 1057-1071, 2020.

[30] F. Chahlaoui and H. J. S. C. S. Dahmouni, "A taxonomy of load balancing mechanisms in centralized and distributed SDN architectures," vol. 1, no. 5, p. 268, 2020.

[31] M. R. Belgaum, Z. Alansari, S. Musa, M. M. Alam, M. J. I. J. o. E. Mazliham, and C. Engineering, "Role of artificial intelligence in cloud computing, IoT and SDN: Reliability and scalability issues," vol. 11, no. 5, p. 4458, 2021.

[32] L. Zhu *et al.*, "SDN controllers: A comprehensive analysis and performance evaluation study," vol. 53, no. 6, pp. 1-40, 2020.

[33] D. Cabarkapa and D. Rancic, "Software-Defined Networking: The Impact of Scalability on Controller Performance," in *2022 IEEE Zooming Innovation in Consumer Technologies Conference (ZINC)*, 2022, pp. 17-21: IEEE.

[34] A. Naseri, M. Ahmadi, and L. J. C. C. PourKarimi, "Placement of SDN controllers based on network setup cost and latency of control packets," 2023.

[35] M. T. Islam, N. Islam, and M. A. J. W. P. C. Refat, "Node to node performance evaluation through RYU SDN controller," vol. 112, pp. 555-570, 2020.

[36] V. H. Kelian *et al.*, "Toward Adaptive and Scalable Topology in Distributed SDN Controller," vol. 30, no. 1, pp. 115-131, 2023.

[37] P. Sun, Z. Guo, J. Li, Y. Xu, J. Lan, and Y. J. I. A. T. o. N. Hu, "Enabling scalable routing in software-defined networks with deep reinforcement learning on critical nodes," vol. 30, no. 2, pp. 629-640, 2021.

[38] Z. Ye, G. Sun, and M. J. I. I. o. T. J. Guizani, "ILBPS: An Integrated Optimization Approach Based on Adaptive Load-Balancing and Heuristic Path Selection in SDN," 2023.

[39] J. C. C. Chica, J. C. Imbachi, J. F. B. J. J. o. N. Vega, and C. Applications, "Security in SDN: A comprehensive survey," vol. 159, p. 102595, 2020.

[40] W. Jiang, H. Han, M. He, and W. J. E. S. w. A. Gu, "ML-based pre-deployment SDN performance prediction with neural network boosting regression," vol. 241, p. 122774, 2024.

[41] G. D. Singh *et al.*, "A novel framework for capacitated SDN controller placement: Balancing latency and reliability with PSO algorithm," vol. 87, pp. 77-92, 2024.

[42] A. K. Singh, S. Srivastava, S. J. J. o. A. I. Banerjea, and H. Computing, "Evaluating heuristic techniques as a solution of controller placement problem in SDN," vol. 14, no. 9, pp. 11729-11746, 2023.

[43] G. Ramya and R. J. T. J. o. S. Manoharan, "Traffic-aware dynamic controller placement in SDN using NFV," vol. 79, no. 2, pp. 2082-2107, 2023.

[44] O. Benoudifa, A. A. Wakrime, R. J. J. o. K. S. U.-C. Benaini, and I. Sciences, "Autonomous solution for Controller Placement Problem of Software-Defined Networking using MuZero based intelligent agents," vol. 35, no. 10, p. 101842, 2023.

[45] S. Almakdi, A. Aqdus, R. Amin, and M. S. J. I. A. Alshehri, "An Intelligent Load Balancing Technique for Software Defined Networking based 5G using Macine Learning models," 2023.

[46] J. Jacob, S. Shinde, D. J. J. o. T. Narayan, and I. Technology, "An Efficient Controller Placement Algorithm using Clustering in Software Defined Networks," no. 4, pp. 9-17, 2023.

[47] D. He, J. Chen, and X. J. T. J. o. S. Qiu, "A density algorithm for controller placement problem in software defined wide area networks," vol. 79, no. 5, pp. 5374-5402, 2023.

[48] A. A. Ibrahim *et al.*, "Reliability-aware swarm based multi-objective optimization for controller placement in distributed SDN architecture," 2023.

[49] H. Xu, X. Chai, and H. J. S. Liu, "A Multi-Controller Placement Strategy for Hierarchical Management of Software-Defined Networking," vol. 15, no. 8, p. 1520, 2023.

[50] E. Calle, D. Martínez, M. Mycek, and M. J. I. J. o. C. I. P. Pióro, "Resilient backup controller placement in distributed SDN under critical targeted attacks," vol. 33, p. 100422, 2021.

[51] G. Hessam, G. Saba, and M. I. J. J. o. C. S. Alkhayat, "A new approach for detecting violation of data plane integrity in Software Defined Networks," vol. 29, no. 3, pp. 341-358, 2021.

[52] S. Yang, L. Cui, Z. Chen, W. J. I. T. o. N. Xiao, and S. Management, "An efficient approach to robust SDN controller placement for security," vol. 17, no. 3, pp. 1669-1682, 2020.

[53] M. S. Tok, M. J. C. Demirci, and Security, "Security analysis of SDN controller-based DHCP services and attack mitigation with DHCPguard," vol. 109, p. 102394, 2021.

[54] T. Hasan, A. Akhunzada, T. Giannetsos, and J. Malik, "Orchestrating sdn control plane towards enhanced IoT security," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 457-464: IEEE.

[55] I. Ahammad, M. A. R. Khan, Z. U. J. S. M. P. Salehin, and Theory, "QoS performance enhancement policy through combining fog and SDN," vol. 109, p. 102292, 2021.

[56] S. Goudarzi, M. H. Anisi, H. Ahmadi, and L. J. I. I. o. T. J. Musavian, "Dynamic resource allocation model for distribution operations using SDN," vol. 8, no. 2, pp. 976-988, 2020.

[57] Y. Otoum, A. J. J. o. N. Nayak, and S. Management, "As-ids: Anomaly and signature based ids for the internet of things," vol. 29, pp. 1-26, 2021.

[58] I. P. Saputra, E. Utami, and A. H. Muhammad, "Comparison of anomaly based and signature based methods in detection of scanning vulnerability," in *2022 9th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2022, pp. 221-225: IEEE.

[59] S. Einy, C. Oz, and Y. D. J. M. P. i. E. Navaei, "The anomaly-and signature-based IDS for network security using hybrid inference systems," vol. 2021, pp. 1-10, 2021.

[60] N. Sahani, R. Zhu, J.-H. Cho, and C.-C. J. A. T. o. C.-P. S. Liu, "Machine Learning-based Intrusion Detection for Smart Grid Computing: A Survey," vol. 7, no. 2, pp. 1-31, 2023.

[61] L. Le Jeune, T. Goedeme, and N. J. I. A. Mentens, "Machine learning for misuse-based network intrusion detection: overview, unified evaluation and feature choice comparison framework," vol. 9, pp. 63995-64015, 2021.

[62] M. Labonne, "Anomaly-based network intrusion detection using machine learning," Institut Polytechnique de Paris, 2020.

[63] U. A. Usmani, A. Happonen, and J. Watada, "A Review of Unsupervised Machine Learning Frameworks for Anomaly Detection in Industrial Applications," in *Science and Information Conference*, 2022, pp. 158-189: Springer.

[64] T.-H. Nguyen, T. T. T. Van Son Nguyen, T. T. Dung, N. L. Le Thi Thanh Thuy, N. M. Dung, and N. J. J. o. S. J. U. Van Ba, "Using Machine Learning And Deep Learning To Improve Anomaly Attack," vol. 58, no. 4, 2023.

[65] P. R. B. N. Tomás, "Using Machine Learning (Ml) For Anomaly Detection Over Traffic Present In Service Mesh Arquitectures," 2022.

[66] I. Martins, J. S. Resende, P. R. Sousa, S. Silva, L. Antunes, and J. J. F. G. C. S. Gama, "Host-based IDS: A review and open issues of an anomaly detection system in IoT," vol. 133, pp. 95-113, 2022.

[67] R. Chaganti, W. Suliman, V. Ravi, and A. J. I. Dua, "Deep learning approach for SDN-enabled intrusion detection system in IoT networks," vol. 14, no. 1, p. 41, 2023.

[68] G. Logeswari, S. Bose, T. J. I. A. Anitha, and S. Computing, "An intrusion detection system for sdn using machine learning," vol. 35, no. 1, pp. 867-880, 2023.

[69] A. O. Alzahrani, M. J. J. C. Alenazi, C. Practice, and Experience, "ML-IDSDN: Machine learning based intrusion detection system for software-defined network," vol. 35, no. 1, p. e7438, 2023.

[70] N. S. Shaji, R. Muthalagu, P. M. J. M. T. Pawar, and Applications, "SD-IIDS: intelligent intrusion detection system for software-defined networks," pp. 1-33, 2023.

[71] M. Sneha, A. K. Kumar, N. V. Hegde, A. Anish, and G. J. I. J. o. I. S. Shobha, "RADS: a real-time anomaly detection model for software-defined networks using machine learning," vol. 22, no. 6, pp. 1881-1891, 2023.

[72] D. A. Ezeh and J. J. I. J. A. P. O. T. S. A. F. I. de Oliveira, "An SDN controller-based framework for anomaly detection using a GAN ensemble algorithm," vol. 15, no. 2, pp. 29-36, 2023.

[73] Y. Al-Dunainawi, B. R. Al-Kaseem, and H. S. J. I. A. Al-Raweshidy, "Optimized Artificial Intelligence Model for DDoS Detection in SDN Environment," 2023.

[74] M. Hammad, N. Hewahi, W. J. A. J. o. B. Elmedany, and A. Sciences, "Enhancing Network Intrusion Recovery in SDN with machine learning: an innovative approach," vol. 30, no. 1, pp. 561-572, 2023.

[75] S. Lal and V. Singh, "Techniques to Enhance the Performance of DBSCAN Clustering Algorithm in Data Mining."

[76] H. Zhang, "Wireless Network Analysis and Optimization Based on the Social Media Data."

[77] A. E. Ezugwu *et al.*, "A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects," vol. 110, p. 104743, 2022.

[78] S. Ray, "A quick review of machine learning algorithms," in 2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon), 2019, pp. 35-39: IEEE.

[79] M. Bansal, A. Goyal, and A. J. D. A. J. Choudhary, "A comparative analysis of K-nearest neighbor, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning," vol. 3, p. 100071, 2022.

[80] J. Yang, X. Tan, and S. J. P. R. L. Rahardja, "Outlier detection: How to Select k for k-nearest-neighbors-based outlier detectors," vol. 174, pp. 112-117, 2023.

[81] R. Santos *et al.*, "Machine learning algorithms to detect DDoS attacks in SDN," vol. 32, no. 16, p. e5402, 2020.

[82] R. Anusuya, M. R. Prabhu, C. Prathima, and J. A. J. J. o. S. i. F. S. Kumar, "Detection of TCP, UDP and ICMP DDOS attacks in SDN Using Machine Learning approach," vol. 10, no. 4S, pp. 964-971, 2023.

[83] S. Zavrak, M. J. N. C. Iskefiyeli, and Applications, "Flow-based intrusion detection on software-defined networks: a multivariate time series anomaly detection approach," vol. 35, no. 16, pp. 12175-12193, 2023.

[84] A. Singh, H. Kaur, and N. J. C. C. Kaur, "A novel DDoS detection and mitigation technique using hybrid machine learning model and redirect illegitimate traffic in SDN network," pp. 1-21, 2023.

[85] N. Nisa, A. S. Khan, Z. Ahmad, and J. J. I. J. o. N. M. Abdullah, "TPAAD: Two-phase authentication system for denial of service attack detection and mitigation using machine learning in software-defined network," p. e2258, 2024.

[86] J. Liu, J. Liu, R. J. C. S. Xie, and I. Systems, "Reliability-based controller placement algorithm in software defined networking," vol. 13, no. 2, pp. 547-560, 2016.

[87] G. Wang, Y. Zhao, J. Huang, Q. Duan, and J. Li, "A K-means-based network partition algorithm for controller placement in software defined network," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1-6: IEEE.

[88] L. Zhu, R. Chai, and Q. Chen, "Control plane delay minimization based SDN controller placement scheme," in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2017, pp. 1-6: IEEE.