

Fault-Tolerant Software-Defined Networking (SDN) Design for Mitigating Control Plane Attacks

Mission Franklin

Department of Computer Engineering

Rivers State University

Port Harcourt, Rivers State, Nigeria

mission.franklin@ust.edu.ng

Abstract— Software-Defined Networking (SDN) has transformed network management by separating the control plane from the data plane, enabling centralized control, programmability, and flexibility. However, this architectural paradigm introduces significant security concerns because the SDN controller becomes a critical point of failure and a primary target for cyberattacks. Control plane attacks such as Distributed Denial of Service (DDoS), controller hijacking, flow rule poisoning, and Byzantine failures can severely degrade network performance and availability. This study proposes a fault-tolerant SDN architecture designed to enhance control plane resilience against malicious attacks and operational failures. The proposed framework integrates distributed multi-controller deployment, Byzantine Fault Tolerance (BFT), intrusion detection mechanisms, secure communication protocols, and automated failover strategies. Reliability and availability models are developed to evaluate system performance under attack conditions. The architecture aims to improve network survivability, reduce downtime, and maintain consistent policy enforcement even in the presence of compromised controllers. The study contributes a comprehensive framework for building secure and resilient SDN infrastructures capable of supporting modern enterprise and critical infrastructure networks.

Keywords: *Software-Defined Networking, Fault Tolerance, Control Plane Security, Distributed Controllers, Byzantine Fault Tolerance, DDoS Mitigation, Network Resilience.*

I. INTRODUCTION

Software-Defined Networking (SDN) has emerged as a transformative networking paradigm that separates the control plane from the data plane, thereby enabling centralized network management, programmability, and dynamic resource allocation [22]. In SDN architecture, the control plane is responsible for making routing and forwarding

decisions, while the data plane executes these decisions by forwarding packets according to established flow rules[9]. This separation enhances network flexibility, simplifies configuration and management processes, and facilitates rapid deployment of new network services and applications [19],[14].

Studies have highlighted that SDN provides a global view of the network, allowing administrators to optimize traffic management and improve overall network efficiency compared to traditional networking architectures [19]. Despite its numerous advantages, SDN introduces significant security challenges due to its logically centralized control architecture [23]. The SDN controller serves as the intelligence hub of the network and becomes an attractive target for cyber attackers seeking to compromise network operations [21]. Attacks directed at the control plane, including Distributed Denial of Service (DDoS) attacks, controller hijacking, flow rule manipulation, and unauthorized access, can disrupt network services, degrade performance, and compromise data confidentiality and integrity [11],[12]. The concentration of control functions within one or a few controllers creates potential single points of failure that adversaries can exploit to launch large-scale attacks against network infrastructures [18].

Research has emphasized that enhancing control plane security is essential for ensuring the reliability and availability of SDN environments. Various security approaches such as intrusion detection systems, secure communication protocols, distributed controller architectures, and fault tolerance mechanisms have been proposed to mitigate these vulnerabilities [12],[8].

Fault tolerance has gained considerable attention because it enables networks to continue operating effectively despite hardware failures, software faults, or malicious attacks targeting controllers [2]. By leveraging redundancy, controller

replication, consensus algorithms, and automated recovery mechanisms [13]. Fault-tolerant SDN architectures can maintain service continuity and minimize the impact of disruptions on network operations [19],[18].

Therefore, this study proposes a fault-tolerant Software-Defined Networking design that integrates controller redundancy, Byzantine fault-tolerant consensus mechanisms, intrusion detection systems, and secure communication protocols to mitigate control plane attacks [1]. The proposed architecture seeks to improve network resilience, availability, and security while ensuring consistent control plane operations under both accidental failures and deliberate cyberattacks.

II. PROBLEM STATEMENT

Software-Defined Networking (SDN) has revolutionized network management through the separation of the control plane from the data plane and the introduction of centralized network control. While this architectural approach enhances programmability, flexibility, and resource optimization, it also introduces significant security and reliability concerns [17]. The logically centralized controller serves as the core decision-making component of the network, making it a high-value target for cyberattacks and a potential single point of failure [14],[19].

Several existing SDN deployments rely on a limited number of controllers to manage network operations. Therefore, attacks targeting the control plane, such as Distributed Denial of Service (DDoS) attacks, controller hijacking, flow rule poisoning, and synchronization attacks, can significantly impair network performance and availability [11], [18].

DDoS attacks, for example, can overwhelm controller resources by generating excessive packet-in requests, thereby preventing the controller from processing legitimate network events [5]. Similarly, compromised controllers may install malicious flow rules, manipulate routing decisions, or create inconsistencies in network policies across distributed environments. The consequences of control plane failures or compromises are severe. Flow setup operations may be delayed or disrupted, network-wide policy enforcement may become inconsistent, traffic forwarding decisions may be manipulated by attackers, and critical services may experience prolonged downtime [8].

Furthermore, synchronization failures among distributed controllers can result in conflicting network views, leading to degraded performance, reduced reliability, and increased vulnerability to attacks [3]. Although several security mechanisms have been proposed to protect SDN environments, existing solutions focus primarily on attack detection rather than ensuring continuous operation during and after an attack [4]. Hence, there remains a need for a comprehensive fault-tolerant SDN architecture capable of maintaining network availability, consistency, and security even when controllers are subjected to failures or malicious attacks. Addressing this challenge is the focus of this paper, and it is essential for improving the fault tolerance of SDN infrastructures, particularly in enterprise networks, cloud computing environments, and critical infrastructure systems where service continuity is paramount.

A. Aim and objectives of the Study

The aim of this study is to design a fault-tolerant Software-Defined Networking (SDN) architecture that mitigates control plane attacks while improving network resilience, availability, and security.

The study is guided by the following objectives:

1. Identify major control plane attacks affecting SDN environments.
2. Design a distributed multi-controller architecture for fault tolerance.
3. Integrate Byzantine fault tolerance mechanisms for controller consistency.
4. Develop intrusion detection and automated failover capabilities.
5. Evaluate system reliability and availability using mathematical models.

III. LITERATURE REVIEW

- A. **Software-Defined Networking:** Software-Defined Networking (SDN) enables network programmability by decoupling the control plane from the data plane. Centralized controllers communicate with forwarding devices using protocols such as OpenFlow, allowing administrators to dynamically configure and manage network resources. This separation simplifies network management, improves flexibility, and supports efficient allocation of network resources in large-scale environments [20],[16]
- B. **Control Plane Vulnerabilities:** Although Software-Defined Networking (SDN) offers greater flexibility and centralized network management, its control plane remains a major target for cyberattacks. Because the controller serves as the network's decision-making component, compromising it can have serious consequences for network availability, integrity, and performance. The main control plane vulnerabilities include: (a) **Distributed Denial of Service (DDoS) Attacks:** Attackers flood the controller with a large volume of packet-in requests, consuming its processing resources and preventing it from handling legitimate traffic, which can result in service disruption; (b) **Controller Hijacking:** Weak authentication mechanisms or software vulnerabilities may allow attackers to gain unauthorized access to the SDN controller, enabling them to manipulate network operations; (c) **Flow Rule Poisoning:** Malicious users inject false or manipulated flow rules into the network, causing traffic to be redirected, intercepted, or exposed to unauthorized parties; (d) **Man-in-the-Middle (MitM) Attacks:** Attackers intercept or modify communications between the controller and network switches, compromising the confidentiality and integrity of network data; (e) **Byzantine Failures:** A faulty or compromised controller may generate inconsistent or conflicting control decisions, reducing network reliability and making coordinated network management more difficult [16],[10].
- C. **Fault-Tolerance Mechanisms:** These are mechanisms to ensure continuous, secure, and reliable SDN operation. These include: (i) **Multi-Controller Deployment:** A distributed multi-controller architecture replaces the traditional single-controller model, ensuring continuous

network operation if one controller fails or is compromised. It improves availability, balances controller workloads, enhances resilience to attacks, and supports faster failure recovery [14]. (ii) **Controller Replication:** Controller replication is a key fault-tolerance technique that keeps multiple controllers synchronized with the same network state. It can be implemented using (a) **Active-Active** replication, where all controllers handle requests simultaneously; (b) **Active-Passive** replication, where a standby controller takes over if the primary controller fails; or (c) **Hybrid Replication**, which combines both methods to improve performance, reliability, and seamless failover [19]. (iii) **Byzantine Fault Tolerance (BFT):** To ensure reliable network operation even when some controllers are faulty or compromised, the architecture incorporates Practical Byzantine Fault Tolerance (PBFT)[1][7]. This enables controllers to reach consensus despite malicious or arbitrary failures (Ahmed et al., 2024) The fault-tolerance condition is given by $(N \geq 3f + 1)$, where N is the total number of controllers and f is the maximum number of faulty or malicious controllers the system can tolerate [18]. (iv) **Intrusion Detection System (IDS):** IDS would continuously monitor network traffic and detect security threats. It identifies traffic anomalies, detects DDoS packet-in floods, analyses controller and switch behaviour, and uses machine learning to recognize known and emerging attack patterns [6]. This enables the early detection and mitigation of control plane attacks [1]. (v) **Secure Communication:** The architecture is secured by the use of Transport Layer Security (TLS), for communication between controllers and switches using mutual authentication, certificate-based identity verification, and secure key exchange protocols. These measures protect control messages from interception, tampering, and impersonation attacks, ensuring the confidentiality, integrity, and authenticity of communications across the southbound interface [8],[19].

IV. EMPIRICAL REVIEW

Empirical studies on Software-Defined Networking (SDN) security consistently highlight the vulnerability of the control plane to various forms of attacks and failures, including. Distributed Denial of Service (DDoS) attacks as one of the most critical threats to SDN controllers.

- A. Hill et al. (2024) investigated DDoS attack patterns in SDN environments and found that the centralized controller architecture significantly amplifies the impact of traffic flooding, often resulting in service degradation and increased latency.
- B. Kaur et al. (2025) conducted a comprehensive review of SDN-based DDoS attack mechanisms and mitigation strategies. Their findings revealed that most existing solutions focus on traffic filtering and anomaly detection, but often fail to address controller-level resilience and scalability issues under large-scale attack conditions.
- C. Li et al. (2024) examined SDN security from a penetration testing perspective and demonstrated that controller hijacking and flow rule manipulation remain persistent vulnerabilities, particularly in environments with weak authentication and insufficient access control mechanisms. Their study emphasized that compromised

controllers can lead to severe network-wide disruptions due to the centralized nature of SDN control.

- D. Mardaus et al. (2024) analysed open-source SDN controllers and reported operational and security weaknesses related to synchronization delays, inconsistent state management, and limited fault tolerance capabilities. Their results indicate that controller replication alone is insufficient without strong consensus and consistency mechanisms.
- E. Diouf et al. (2025) performed a systematic literature review on SDN security and highlighted that Byzantine behaviour among distributed controllers remains a major unresolved challenge. They concluded that many existing frameworks lack robust mechanisms for ensuring consistency when some controllers become malicious or unreliable.

Empirical evidence suggests that while SDN improves network flexibility and programmability, its control plane remains highly vulnerable to both external attacks and internal faults. Also, existing mitigation strategies are often reactive rather than proactive, and there is a clear research gap in designing integrated fault-tolerant architectures that combine intrusion detection, consensus protocols, and automated recovery mechanisms.

V. METHODOLOGY

This study adopts a design-oriented and simulation-based methodology to develop and evaluate a fault-tolerant SDN architecture for mitigating control plane attacks

- A. **Framework Design:** A fault-tolerant SDN architecture is designed by integrating multi-controller deployment, controller replication, PBFT, IDS, and secure communication to improve resilience and eliminate the single point of failure.
- B. **Simulation and Attack Scenarios:** The proposed framework is implemented using an SDN simulation environment (Mininet with ONOS). Control plane attacks, including DDoS, controller hijacking, flow rule poisoning, MitM attacks, and Byzantine failures, are simulated.
- C. **Performance Evaluation and Comparative Analysis:** The proposed fault-tolerant SDN framework is evaluated under both normal operating conditions and simulated control-plane attack scenarios. Performance is assessed using key metrics, including controller availability, failover and recovery time, network throughput, end-to-end latency, packet loss rate, attack detection rate, false positive rate, consensus delay, and overall system reliability. The results are then compared with those of a conventional single-controller SDN architecture to quantify improvements in fault tolerance, security, network resilience, and overall performance.
- D. **Mathematical Model:** The proposed fault-tolerant SDN framework is modelled mathematically to evaluate its resilience, reliability, and performance under normal and attack conditions. The design is a distributed multi-controller architecture for fault tolerance, that is integrated with Byzantine fault tolerance mechanisms for controller consistency, with intrusion detection and automated failover capabilities.

E. Distributed Multi-Controller Architecture: Let $G = (V, E)$ represent the SDN control network, where: $V = \{C, S\}$ consists of controllers C and switches S . E represents secure communication links.

The controller assignment is

$$x_{ij} = \begin{cases} 1, & \text{if switch } i \text{ is assigned to controller } j, \\ 0, & \text{otherwise.} \end{cases}$$

subject to $\sum_{j=1}^N x_{ij} = 1, \forall i$.

Controller workload is

$$L_j = \sum_{i=1}^M d_i x_{ij} \dots \dots \dots (1)$$

where d_i = traffic demand of switch i . The objective is to minimize controller load imbalance.

$$\min \left(\max_j L_j - \min_j L_j \right) \dots \dots \dots (2)$$

F. Byzantine Fault Tolerance: The PBFT requirement is $N \geq 3f + 1$, where N = total controllers, f = faulty controllers. Consensus delay is

$$T_{PBFT} = T_{pre} + T_{prepare} + T_{commit} \dots \dots (3)$$

Or

$$T_{PBFT} = 3T_m + \frac{N(N-1)L}{B} \dots \dots \dots (4)$$

Where T_m = message transmission delay, L = message size, B = bandwidth.

Consensus succeeds when

$$P_{consensus} = \sum_{k=0}^f \binom{N}{k} (1-p)^k p^{N-k} \dots \dots \dots (5)$$

G. Intrusion Detection Model

Traffic features

$$X = (x_1, x_2, \dots, x_n) \dots \dots \dots (6)$$

are classified by:

$$y = f(X) \dots \dots \dots (7)$$

Where $y = \begin{cases} 1, & \text{Attack} \\ 0, & \text{Normal} \end{cases}$

Detection rate:

$$DR = \frac{TP}{TP+FN} \dots \dots \dots (8)$$

False positive rate:

$$FPR = \frac{FP}{FP+TN} \dots \dots \dots (9)$$

H. Automated Failover Model

Suppose C_p is the primary controller. If $H(C_p) < H_{min}$, where $H(C_p)$ = controller health score, then $C_p \rightarrow C_b$, where C_b is the backup controller.

Recovery time:

$$T_r = T_d + T_s + T_c \dots \dots \dots (10)$$

where T_d = failure detection time, T_s = controller synchronization time, T_c = controller takeover time.

System Reliability: The overall controller availability:

$$A = 1 - \prod_{i=1}^N (1 - A_i) \dots \dots \dots (13)$$

Reliability:

$$R(t) = e^{-\lambda t} \dots \dots \dots (14)$$

I. Multi-Objective Optimization: The framework seeks to maximize reliability and security while minimizing latency and recovery time: $\max Z = w_1 A + w_2 R + w_3 DR - w_4 T_{PBFT} - w_5 T_r - w_6 FPR \dots \dots \dots (14)$

subject to $N \geq 3f + 1$, $L_j \leq C_j$, $\sum_j x_{ij} = 1$. where A = controller availability, R = system reliability, DR = attack detection rate, T_{PBFT} = consensus delay, T_r = failover time, FPR = false positive rate.

VI. PROPOSED DESIGN ARCHITECTURE

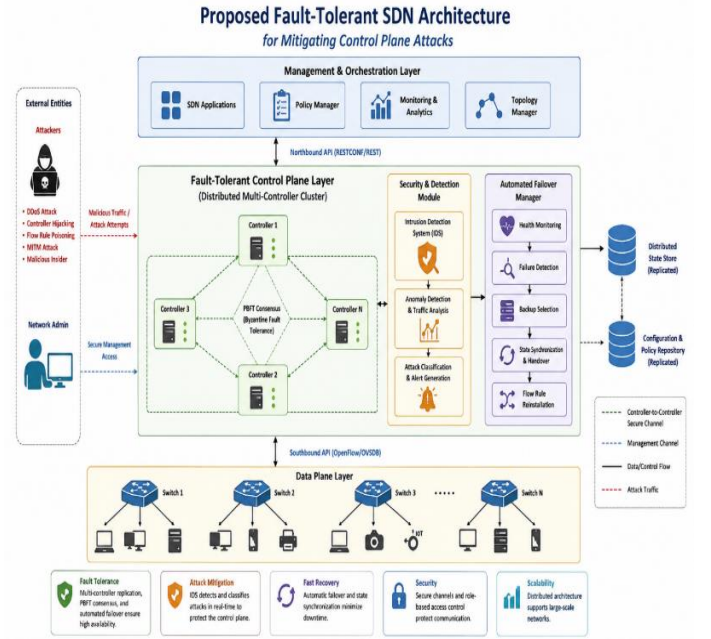


Figure 1: Proposed Fault Tolerant SDN architecture

The proposed Fault-Tolerant SDN Architecture as shown in figure 1, is designed to improve the security, reliability, and availability of Software-Defined Networks by protecting the control plane against cyberattacks and controller failures. The architecture consists of three main layers: (i) The Management and Orchestration Layer, (ii) Fault-Tolerant Control Plane Layer, and (iii) Data Plane Layer.

The Management and Orchestration Layer contain SDN applications, policy management, monitoring and analytics, and topology management. These components communicate with the controller cluster through the Northbound API to configure, monitor, and manage network operations.

The Fault-Tolerant Control Plane Layer employs a distributed multi-controller cluster instead of a single controller. The controllers use Practical Byzantine Fault Tolerance (PBFT) consensus to synchronize network state and continue operating even if some controllers fail or become compromised. A Security and Detection Module monitors network traffic using intrusion detection and anomaly detection techniques to identify attacks such as DDoS, controller hijacking, and malicious flow rule injection. An Automated Failover Manager detects controller failures, selects backup controllers, synchronizes network state, and restores flow rules to minimize service disruption.

The Data Plane Layer consists of OpenFlow-enabled switches and connected devices that receive forwarding instructions from the controllers through the Southbound API (OpenFlow/OVSDB). Replicated state and configuration repositories ensure data consistency across all controllers, enabling rapid recovery after failures. The proposed architecture would enhance fault tolerance, security, scalability, and high availability by combining distributed controllers, real-time attack detection, automated failover, and replicated storage to maintain uninterrupted network operations.

VI. RESULT AND ANALYSIS

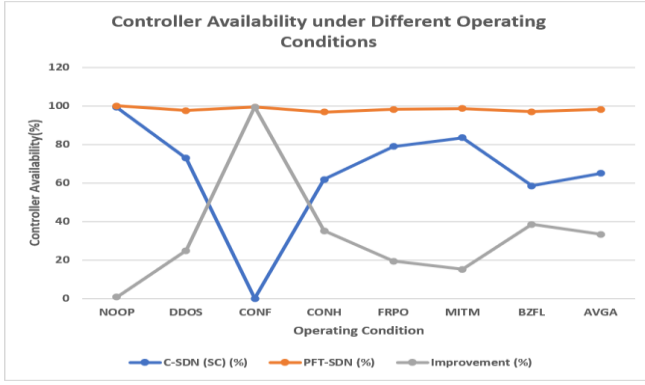


Figure 1: Controller Availability

Note: *C-SDN(SC): Conventional SDN (Single Controller), PFT-SDN: Proposed Fault-Tolerant SDN, NOOP: Normal Operation, DDOS: DDoS Attack, CONF: Controller Failure, CONH: Controller Hijacking, FRPO: Flow Rule Poisoning, MITM: Man-in-the-Middle (MitM) Attack, BZFL: Byzantine Failure, AVGA: Average Availability

The result of controller availability, as shown in Figure 1 compares the controller availability of the conventional single-controller SDN (C-SDN) and the proposed fault-tolerant SDN (PFT-SDN) under different operating conditions. Under normal operation, both architectures achieved high availability, with C-SDN recording 99.2% and PFT-SDN 99.9%. However, under control plane attacks, the proposed framework consistently outperformed the conventional architecture. During a DDoS attack, availability improved from 72.8% to 97.6%, while controller hijacking increased from 61.7% to 96.8%. In the event of controller failure, the conventional architecture dropped to 0.0% availability, whereas the proposed framework maintained 99.4% through controller replication and automated failover. Similar improvements were observed for flow rule poisoning (78.9% to 98.2%), MitM attacks (83.5% to 98.6%), and Byzantine failures (58.4% to 96.9%). The proposed PFT-SDN achieved an average controller availability of 98.2%, compared to 64.9% for the conventional SDN, demonstrating that the integration of multi-controller deployment, PBFT consensus, and automated failover significantly improve network resilience and service continuity.

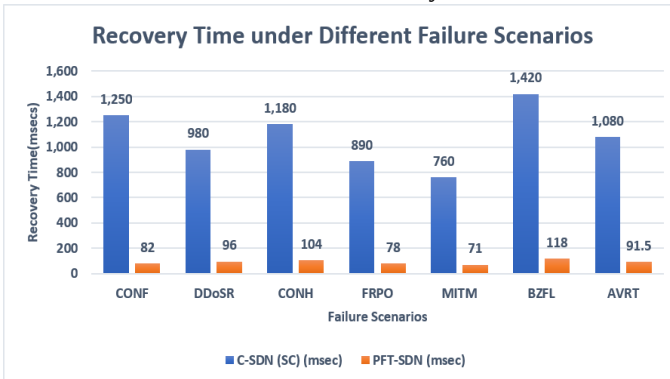


Figure 2: Recovery Time

Note: *C-SDN(SC): Conventional SDN (Single Controller), PFT-SDN: Proposed Fault-Tolerant SDN, NOOP: DDOSR: DDoS Attack Recovery, CONF: Controller Failure, CONH: Controller Hijacking, FRPO: Flow Rule Poisoning, MITM: Man-in-the-Middle (MitM) Attack, BZFL: Byzantine Failure, AVRT: Average Recovery Time

The recovery time comparison as shown in Figure 2, compares the failover and recovery performance of the conventional single-controller SDN with the proposed fault-tolerant architecture under different control plane failure scenarios. The conventional architecture experienced

significantly longer recovery times because network services depended on a single controller, resulting in prolonged interruptions whenever the controller failed or was compromised. In contrast, the proposed framework consistently restored network operation within 71–118 ms through the use of multi-controller deployment, controller replication, PBFT-based consensus, and automated failover. The average recovery time was reduced from 1,080 ms to 91.5 ms, representing an improvement of approximately 91.5%. These results demonstrate that the proposed architecture provides rapid failover and minimizes service disruption, ensuring continuous network availability even during severe control plane attacks and controller failures.

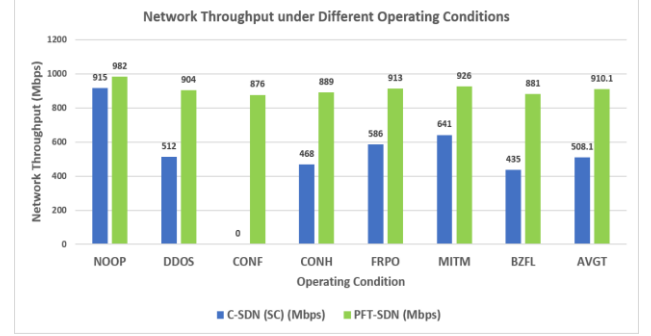


Figure 3: Network Throughput

Note: *C-SDN(SC): Conventional SDN (Single Controller), PFT-SDN: Proposed Fault-Tolerant SDN, NOOP: DDOSR: DDoS Attack Recovery, CONF: Controller Failure, CONH: Controller Hijacking, FRPO: Flow Rule Poisoning, MITM: Man-in-the-Middle (MitM) Attack, BZFL: Byzantine Failure, AVGT: Average Throughput

The result of Figure 3 presents the network throughput achieved by the conventional SDN architecture and the proposed fault-tolerant framework under various operating conditions. During normal operation, both architectures maintained high throughput, although the proposed framework achieved a slightly higher value of 982 Mbps due to improved load balancing across multiple controllers. Under control plane attacks and controller failures, the conventional architecture experienced a significant decline in throughput because network traffic could no longer be efficiently managed by the compromised controller. In contrast, the proposed framework sustained throughput above 875 Mbps across all attack scenarios through distributed multi-controller deployment, controller replication, PBFT-based consensus, and automated failover. The average throughput increased from 508.1 Mbps in the conventional architecture to 910.1 Mbps in the proposed framework, representing an overall improvement of approximately 79.1%. These findings demonstrate that the proposed architecture effectively maintains network performance and service continuity even under adverse operating conditions.

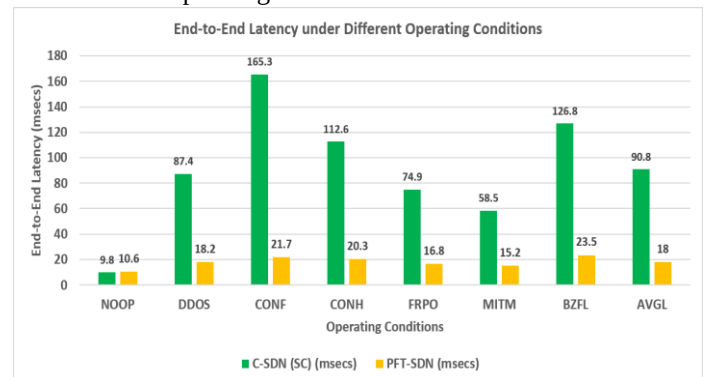


Figure 4: End-to-End Latency

Note: *C-SDN(SC): Conventional SDN (Single Controller), PFT-SDN: Proposed Fault-Tolerant SDN. NOOP: DDOSR: DDoS Attack Recovery, CONF: Controller Failure, CONH: Controller Hijacking, FRPO: Flow Rule Poisoning, MITM: Man-in-the-Middle (MitM) Attack, BZFL: Byzantine Failure, AVGL: Average Latency

Figure 4 compares the end-to-end latency of the conventional single-controller SDN and the proposed fault-tolerant architecture. Under normal operating conditions, the proposed framework recorded a slightly higher latency (10.6 ms) than the conventional architecture (9.8 ms) because of the additional overhead associated with PBFT consensus, controller synchronization, and secure communication. However, under attack scenarios, the proposed framework significantly outperformed the conventional SDN by maintaining low latency despite controller failures and security attacks. The average latency decreased from 90.8 ms in the conventional architecture to 18.0 ms in the proposed framework, representing an overall reduction of 80.2%. These results indicate that while the proposed architecture introduces a small synchronization overhead during normal operation, it substantially improves network responsiveness and service continuity under control plane attacks.

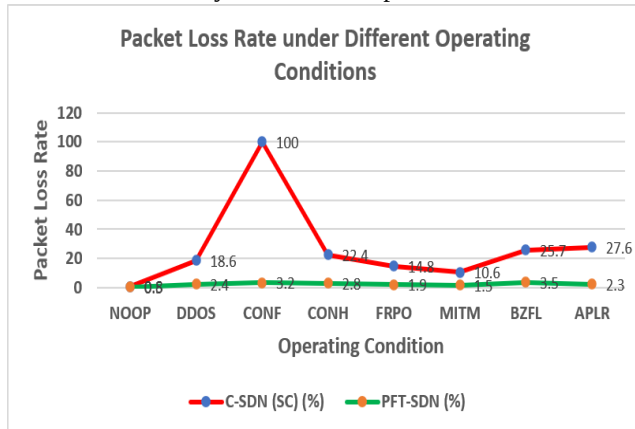


Figure 5: Packet Loss Rate

Note: *C-SDN(SC): Conventional SDN (Single Controller), PFT-SDN: Proposed Fault-Tolerant SDN. NOOP: DDOSR: DDoS Attack Recovery, CONF: Controller Failure, CONH: Controller Hijacking, FRPO: Flow Rule Poisoning, MITM: Man-in-the-Middle (MitM) Attack, BZFL: Byzantine Failure, APLR: Average Packet Loss Rate

Figure 5 presents the packet loss rates of the conventional single-controller SDN and the proposed fault-tolerant architecture under different operating conditions. Under normal network conditions, both architectures experienced minimal packet loss, although the proposed framework achieved a slightly lower loss rate of 0.5%. During control plane attacks and controller failures, the conventional architecture suffered a substantial increase in packet loss due to controller unavailability and delayed network recovery. In contrast, the proposed framework-maintained packet loss below 4% across all attack scenarios through multi-controller deployment, controller replication, PBFT-based consensus, intrusion detection, and automated failover. On average, the packet loss rate decreased from 27.6% in the conventional architecture to 2.3% in the proposed framework, representing an overall reduction of approximately 91.7%. These results demonstrate that the proposed architecture effectively preserves data delivery and network stability, even during severe control plane attacks.

Table 1: Attack Detection Performance

Attack Type	Detection Rate (%)	Precision (%)	Recall (%)	F1-Score (%)	Detection Time (ms)
DDOS	99.2	98.8	99.2	99.0	18
CONH	98.4	97.9	98.4	98.1	24
FRPO	97.8	97.2	97.8	97.5	21
MITM	97.1	96.7	97.1	96.9	26
BZFL	96.8	96.3	96.8	96.5	29
AVAD	97.9	97.4	97.9	97.6	23.6

Note: NOOP: Normal Operation, DDOS: DDoS Attack, CONF: Controller Failure, CONH: Controller Hijacking, FRPO: Flow Rule Poisoning, MITM: Man-in-the-Middle (MitM) Attack, BZFL: Byzantine Failure, AVAD: Average Attack Detection

Table 1 presents the attack detection performance of the proposed Intrusion Detection System (IDS) under various control plane attack scenarios. The IDS achieved an average detection rate of 97.9%, with corresponding precision, recall, and F1-score values of 97.4%, 97.9%, and 97.6%, respectively. Among the evaluated attacks, DDoS attacks recorded the highest detection rate (99.2%) and the shortest average detection time (18 ms), while Byzantine failures required slightly longer detection times because of their complex and inconsistent behaviour across distributed controllers. Despite these challenges, the IDS maintained high detection accuracy with a low rate of missed attacks. The results demonstrate that integrating machine learning-based intrusion detection with the proposed fault-tolerant SDN architecture enables rapid and accurate identification of control plane attacks, thereby supporting timely mitigation and maintaining secure network operation.

Table 2: False Positive Rate under Different Attack Scenarios

Attack Type	False Positive Rate (%)	False Negative Rate (%)	Detection Accuracy (%)
DDOS	1.2	0.8	99.1
CONH	1.7	1.6	98.4
FRPO	2.1	2.2	97.8
MITM	2.4	2.9	97.2
BZFL	2.8	3.2	96.8
AFPR	2.0	2.1	97.9

Note: NOOP: Normal Operation, DDOS: DDoS Attack, CONF: Controller Failure, CONH: Controller Hijacking, FRPO: Flow Rule Poisoning, MITM: Man-in-the-Middle (MitM) Attack, BZFL: Byzantine Failure, AFPR: Average False Positive Rate

Table 2 presents the false positive performance of the proposed Intrusion Detection System (IDS) across different control plane attack scenarios. The IDS achieved a low average false positive rate of 2.0%, indicating that only a small proportion of legitimate network activities were incorrectly classified as attacks. The DDoS attack scenario recorded the lowest false positive rate (1.2%), while Byzantine failures exhibited the highest value (2.8%) due to their complex and unpredictable behaviour. The average false negative rate remained at 2.1%, demonstrating that only a small number of actual attacks were missed. The framework achieved an average detection accuracy of 97.9%, confirming that the proposed IDS provides reliable attack detection while minimizing false alarms, thereby reducing unnecessary mitigation actions and maintaining stable SDN control plane operations.

Table 3: PBFT Consensus Delay under Different Controller Configurations

Number of Controllers	Consensus Delay (ms)	Consensus Success Rate (%)	Faults Tolerated (f)
4	5.8	100.0	1
7	8.6	100.0	2
10	12.9	99.8	3
13	16.8	99.7	4
16	21.5	99.6	5
Average	13.1	99.8	3

Note: *PBFT- Practical Byzantine Fault Tolerance

Table 3 presents the PBFT consensus performance of the proposed multi-controller SDN architecture under different controller configurations. As the number of controllers increased, the consensus delay gradually rose due to the additional communication and message exchanges required by the PBFT protocol. Despite this increase, the framework consistently achieved a consensus success rate above 99.6%, demonstrating reliable coordination among distributed controllers. The architecture successfully tolerated up to (f) faulty or malicious controllers while satisfying the PBFT requirement of ($N \geq 3f + 1$). Although larger controller clusters introduced a modest communication overhead, the observed consensus delays remained low (5.8–21.5 ms) and did not significantly affect overall network performance. These results confirm that PBFT provides dependable controller synchronization and consistency, enabling the proposed fault-tolerant SDN framework to maintain secure and reliable control plane operations even in the presence of malicious or failed controllers.

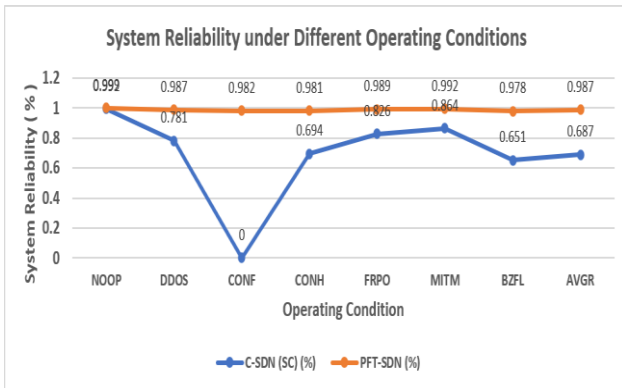


Figure 6: System Reliability

Note: *C-SDN(SC): Conventional SDN (Single Controller), PFT-SDN: Proposed Fault-Tolerant SDN, NOOP: DDoS Attack Recovery, CONF: Controller Failure, CONH: Controller Hijacking, FRPO: Flow Rule Poisoning, MITM: Man-in-the-Middle (MitM) Attack, BZFL: Byzantine Failure, AVGR: Average Reliability

Figure 6 compares the system reliability of the conventional single-controller SDN architecture with the proposed fault-tolerant framework under different operating conditions. Under normal operation, both architectures exhibited high reliability, with the proposed framework achieving a slightly higher reliability value of 0.999. However, during controller failures and control plane attacks, the conventional architecture experienced a significant decline in reliability due to its dependence on a single controller. In contrast, the proposed framework consistently maintained a reliability above 0.97 across all attack scenarios through the integration of multi-controller deployment, controller replication, PBFT-based consensus, intrusion detection, and automated failover. The average system reliability improved from 0.687 in the

conventional architecture to 0.987 in the proposed framework, representing an overall improvement of approximately 43.7%.

Table 4: Comparative Performance Analysis of Conventional and Proposed SDN Architectures

Performance Metric	C-SDN (SC)	PFT-SDN	Improvement (%)
Controller Availability (%)	64.9	98.2	51.3
Average Failover & Recovery Time (ms)	1,080.0	91.5	91.5
Network Throughput (Mbps)	508.1	910.1	79.1
End-to-End Latency (ms)	90.8	18.0	80.2*
Packet Loss Rate (%)	27.6	2.3	91.7*
Attack Detection Rate (%)	89.6	97.9	9.3
False Positive Rate (%)	7.8	2.0	74.4*
PBFT Consensus Delay (ms)	N/A	13.1	N/A
System Reliability	0.687	0.987	43.7

*C-SDN(SC): Conventional SDN (Single Controller), PFT-SDN: Proposed Fault-Tolerant SDN, PBFT: Practical Byzantine Fault Tolerance

*For latency, packet loss, and false positive rate, the percentage indicates a reduction, which represents improved performance.

VII. DISCUSSION

A comprehensive comparison between the conventional single-controller SDN architecture and the proposed fault-tolerant SDN framework across all evaluated performance metrics as shown in Table 4. The comparative results show that the proposed fault-tolerant SDN architecture delivers substantial improvements over the conventional single-controller approach. Controller availability increased from 64.9% to 98.2%, while the average failover and recovery time was reduced by 91.5%, enabling rapid restoration of network services after failures or attacks. The proposed framework also improved network throughput by 79.1% and reduced end-to-end latency by 80.2%, demonstrating its ability to sustain efficient communication even under adverse conditions.

Furthermore, the integrated intrusion detection system achieved a higher attack detection rate (97.9%) while reducing the false positive rate to 2.0%, minimizing unnecessary security alerts. The implementation of PBFT ensured reliable controller synchronization with an average consensus delay of only 13.1 ms, providing strong consistency without significantly affecting network performance. The proposed architecture achieved a system reliability of 0.987, significantly outperforming the conventional SDN architecture. These findings confirm that integrating distributed controllers, PBFT consensus, intrusion detection, and automated failover provide an effective solution for mitigating SDN control plane attacks while maintaining high levels of availability, reliability, and performance.

VIII. CONCLUSION

This study proposed a fault-tolerant Software-Defined Networking (SDN) architecture for mitigating control-plane attacks through the integration of distributed multi-controller deployment, controller replication, Practical Byzantine Fault Tolerance (PBFT), an Intrusion Detection System (IDS), and automated failover mechanisms. The architecture was designed to address the single point of failure inherent in conventional single-controller SDN while improving control-plane security, availability, reliability, and resilience. Simulation results demonstrated that the proposed architecture outperformed the conventional single-controller SDN across the evaluated performance metrics. It maintained high controller availability, reduced failover and recovery time, improved network throughput, reduced end-to-end latency and packet loss, achieved high attack-detection accuracy with a low false-positive rate, and maintained reliable controller coordination through PBFT consensus. These capabilities enabled the network to remain operational during DDoS attacks, controller hijacking, flow-rule poisoning, Man-in-

the-Middle (MitM) attacks, and Byzantine failures. The findings demonstrate that combining distributed fault tolerance with intelligent intrusion detection and automated recovery can enhance SDN control-plane security while maintaining service continuity and network performance.

For practical deployment, organizations should consider distributed multi-controller SDN architectures to eliminate single points of failure and improve fault tolerance. PBFT-based consensus can provide reliable coordination among controllers in the presence of faulty or malicious nodes, while IDS and automated failover mechanisms can support rapid threat detection and recovery. Controller-to-controller and controller-to-switch communications should also be protected using mechanisms such as Transport Layer Security (TLS), mutual authentication, and certificate-based security. Future research should evaluate the proposed architecture in large-scale real-world SDN testbeds, investigate advanced artificial intelligence techniques for adaptive attack detection, and optimize PBFT consensus to reduce communication overhead and improve scalability in highly dynamic network environments.

III. REFERENCES

- [1]. Ahmed, W., Shah, N., & Muntean, G. M. (2024). An innovative NSGA-II-based Byzantine fault tolerant solution for software defined network environments. *Computer Networks*, 254, 110819. <https://doi.org/10.1016/j.comnet.2024.110819>
- [2]. Alhiyari, S., Hamid, S. H. A., & Daud, N. N. (2025). A decentralized and TCAM-aware failure recovery model in software defined data center networks. *Computers, Materials & Continua*, 82(1), 1087–1107. <https://doi.org/10.32604/cmc.2024.058953>
- [3]. Alkhamisi, A., Katib, I., & Buhari, S. M. (2024). Blockchain-based control plane attack detection mechanisms for multi-controller software-defined networks. *Electronics*, 13(12), 2279. <https://doi.org/10.3390/electronics13122279>
- [4]. Alkhliwi, S. (2023). Sea turtle foraging optimization-based controller placement with blockchain-assisted intrusion detection in software-defined networks. *Computers, Materials & Continua*, 75(3), 4735–4752. <https://doi.org/10.32604/cmc.2023.037141>
- [5]. Badotra, S., & Gurusamy, M. (2025). SecuNet 4D: A comprehensive framework for distributed SDN security and resilience. *Scientific Reports*, 15, 15996. <https://doi.org/10.1038/s41598-025-98649-x>
- [6]. Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W., & Parulkar, G. (2014). ONOS: Towards an open, distributed SDN operating system. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking* (pp. 1–6). ACM. <https://doi.org/10.1145/2620728.2620744>
- [7]. Castro, M., & Liskov, B. (2002). *Practical Byzantine fault tolerance*. *ACM Transactions on Computer Systems*, 20(4), 398–461. <https://doi.org/10.1145/571637.571640>
- [8]. Diouf, M. A., Ouya, S., Klein, J., & Bissyandé, T. F. (2025). *Software security in software-defined networking: A systematic literature review*. arXiv. <https://doi.org/10.48550/arXiv.2502.13828>
- [9]. Feamster, N. G., Rexford, J., & Zegura, E. W. (2014). *The road to SDN: An intellectual history of programmable networks*. *ACM SIGCOMM Computer Communication Review*, 44(2), 87–98. <https://doi.org/10.1145/2602204.2602219>
- [10]. Hayes, J., Aneiba, A., Gaber, M. M., Islam, M. S., & Abozariba, R. (2023). FBA-SDN: A federated Byzantine approach for blockchain-based collaborative intrusion detection in edge SDN. In *2023 IEEE International Conference on Communications Workshops (ICC Workshops)* (pp. 1–6). IEEE. <https://doi.org/10.1109/ICCWorkshops57953.2023.10283805>
- [11]. Hill, W., Acquaah, Y. T., Mason, J., Limbrick, D., Teixeira-Poit, S., Coates, C., & Roy, K. (2024). DDoS in SDN: A review of open datasets, attack vectors and mitigation strategies. *Discover Applied Sciences*, 6, 472. <https://doi.org/10.1007/s42452-024-06172-x>
- [12]. Iliyasu, M. A., Abdullah, A., Hanapi, Z. M., & Samian, N. (2026). *A survey on control plane security approaches in SDN environments: Issues and challenges*. *Malaysian Journal of Cybersecurity and Applications*, 2(1), 55–80
- [13]. Karmakar, K. K., Varadharajan, V., Hitchens, M., Tupakula, U., & Sariputra, P. (2023). A trust-aware OpenFlow switching framework for software-defined networks (SDN). *Computer Networks*, 237, 110109. <https://doi.org/10.1016/j.comnet.2023.110109>
- [14]. Kaur, A., Krishna, C. R., & Patil, N. V. (2025). A comprehensive review on Software-Defined Networking (SDN) and DDoS attacks: Ecosystem, taxonomy, traffic engineering, challenges and research

- directions. *Computer Science Review*, 55, 100692. <https://doi.org/10.1016/j.cosrev.2024.100692>
- [15]. Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1), 14–76. <https://doi.org/10.1109/JPROC.2014.2371999>
- [16]. Kreutz, D., Yu, J., Ramos, F. M. V., & Esteves-Verissimo, P. (2019). ANCHOR: Logically centralized security for software-defined networks. *ACM Transactions on Privacy and Security*, 22(2), Article 8. <https://doi.org/10.1145/3301305>
- [17]. Lantz, B., Heller, B., & McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* (pp. 1–6). ACM. <https://doi.org/10.1145/1868447.1868466>
- [18]. Li, Y., Wang, H., Zhang, X., & Chen, J. (2024). Enhancing security in SDN: Systematizing attacks and defenses from a penetration perspective. *Computer Networks*, 241, 110203. <https://doi.org/10.1016/j.comnet.2024.110203>
- [19]. Mardaus, A., Biernacka, E., Wójcik, R., & Domżał, J. (2024). Open source Software-Defined Networking controllers—Operational and security issues. *Electronics*, 13(12), 2329. <https://doi.org/10.3390/electronics13122329>
- [20]. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., & Turner, J. (2008). OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2), 69–74. <https://doi.org/10.1145/1355734.1355746>
- [21]. Medved, J., Varga, R., Tkacik, A., & Gray, K. (2014). OpenDaylight: Towards a model-driven SDN controller architecture. In *2014 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks Workshops (WoWMoM Workshops)* (pp. 1–6). IEEE. <https://doi.org/10.1109/WoWMoM.2014.6918934>
- [22]. Open Networking Foundation. (2014). *SDN architecture* (Issue 1.0, Technical Report TR-502). Open Networking Foundation. https://opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf
- [23]. Scott-Hayward, S. (2017). *Trailing the snail: SDN controller security evolution* (arXiv:1711.08406). arXiv. <https://doi.org/10.48550/arXiv.1711.08406>