Enhanced Real-Time DDoS Detection and Mitigation in SDN Using ONOS Controller

Bessan Irtaish *, Mohammad M N Hamarsheh †.
*Department of Natural, Engineering and Technology Sciences (Cyber Security.), Arab American University (AAUP), Ramallah, Palestine
b.irtaish@student.aaup.edu
†Department of computer science and network security, Arab American University (AAUP), Jenin, Palestine
CA: mohammad.hamarsheh@aaup.edu

***Abstract*— The proliferation of Software-Defined Networking (SDN) has enhanced flexibility and centralized control in modern networks. However, this architecture is highly vulnerable to Distributed Denial-of-Service (DDoS) attacks targeting the SDN controller. This paper proposes a real-time DDoS detection and mitigation framework for SDN environments using the ONOS controller, integrating machine learning models with high-speed Python-based data processing libraries. The system is designed to detect and block both UDP Flood and TCP SYN Flood attacks with minimal latency. Experiments were conducted using a network topology simulated using Mininet and realistic attack traffic generated using hping3 and Scapy. The proposed approach achieved 98.5% accuracy, a 90% detection rate, and a 1.5% false positive rate with a response time of only 3.2 seconds. A comparative evaluation against recent ONOS-based studies reveals improved precision, faster mitigation, and better scalability. These findings indicate that the proposed solution is practical for real-time DDoS defense in SDN-enabled enterprise networks.**

Keywords— DDoS Detection, SDN Security, ONOS Controller, Deep Learning, Real-Time Mitigation, Cybersecurity, Network Protection

I. INTRODUCTION

The rapid evolution of communication networks and the advent of advanced distributed systems have required more dynamic, programmable, and extensible network infrastructures. Software-Defined Networking (SDN) is a transformative approach that separates the data plane from the control plane, supports centralized control of traffic, and dynamic policy management through a logically centralized controller [1]. This architectural shift provides important benefits in terms of flexibility, resource utilization, and network function automation. Among all the different SDN controllers, ONOS (Open Network Operating System) stands out for its ability to manage large-scale carrier-grade deployments, fault-tolerant design, and distributed control core [2].

Despite the numerous advantages of SDN, its centralized approach introduces a critical security vulnerability. The SDN controller is a prime target for attackers; a successful Distributed Denial-of-Service (DDoS) attack can prevent the controller from functioning, thereby compromising the whole network [3], [4]. DDoS attacks aim at depleting the resources of the controller by filling it with bogus traffic, thereby making it impossible for real requests to be served [5]. As SDN is extensively being rolled out in mission-critical networks such as enterprise data centers, 5G networks, and cloud computing networks, the controller needs to be protected against such attacks [6].

Some of the detection and mitigation techniques against DDoS have been proposed in existing literature using machine learning and deep learning techniques [7], [8], [9].These solutions typically rely on flow statistics' characteristics, such as packet rates, byte counts, and TCP flags, to distinguish between good and bad traffic. All of these solutions suffer from some limitations, such as high false positives, detection and mitigation delay, and poor scalability in large topologies. Also, most are not reproducible and exhaustive in nature but rather rely on simulated data or omit implementation details at a level that is significant for realistic implementation.

For example, some researchers have simulated SVM or decision tree models in SDN frameworks without developing a real time system with SDN controllers and real-time defense capabilities [10], [11]. They also suggested hybrid ensemble approaches, but their effectiveness was inhibited due to the absence of realistic attack simulations or via simulation across light-weight environments that do not simulate operational network environments. Additionally, most of these studies simulate attack traffic with benchmark datasets such as CIC-DDoS2019 or NSL-KDD, but not investigating their proposed models response in real time [12].

To overcome these issues, in this paper, a robust real-time DDoS detection and mitigation framework specific for ONOS-managed SDN networks is proposed. The system combines best-in-class data collection utilities like Dask and HTTPX with mature machine learning algorithms like Random Forest, Isolation Forest, and Multilayer Perceptron. These tools work in cooperation to inspect traffic in real time and institute mitigation measures directly via the ONOS northbound API. It is tested and verified with simulated attack patterns that very closely reflect real attacks, including UDP Flood and TCP SYN Flood attacks crafted using hping3 and Scapy.

This research is also distinguished with complete experimental transparency, including extensive configuration of the SDN environment, implementation pipeline, data preparation technique, and performance measures. The outcomes are compared against several existing ONOS-based detection mechanisms using standardized measures of accuracy, false positive rate, detection rate, and response time. The outcomes exhibit clear gains in detection accuracy, real-time responsiveness, and operational scalability.

The rest of this paper is structured as follows. Section II provides the background and related literature. Section III outlines the proposed solution. Section IV discusses the methodology and experimental setup. Section V interprets the

978-9950-8599-0-6

results and compares them with related work. Section VI concludes the paper and proposes directions for future work.

II. Related Work

Software-Defined Networking brings programmability, centralized management, and global visibility into modern infrastructures by decoupling control and data planes. Dynamic flow control and real-time processing of traffic are supported by this central model at the expense of exposing the controller to significant vulnerabilities. ONOS is a leading SDN controller that provides modular, scalable, and fault-tolerant functionality optimized for carrier-grade infrastructures. However, the logical centralization of ONOS makes it a material attack surface under Distributed Denial-of-Service conditions.

DDoS attacks on SDN networks aim at the flooding of malicious traffic or flow requests into the controller, hence draining the resources and paralyzing the legitimate flow management. DDoS attacks are dangerous in nature due to the fact that they have the capacity to collapse the control plane as a whole. Research has thus focused on creating mechanisms for detection and mitigation with the help of machine learning, deep learning, and statistical methods.

Initial research approaches relied on statistical thresholds, entropy analysis, or signature-based systems. The authors of [4] used OpenFlow counters and entropy thresholds to detect anomalies and demonstrated effectiveness in synthetic topologies but not implemented in real-time. [7], [13] used a machine learning models like SVM classifier as an application of ONOS to identify abnormal flows but lacked clear mitigation procedures and scalability. Another work [10], [14] used POX controller and had basic flow rule removal as a mitigation, but had no real-time performance demonstrated.

Subsequent research introduced Random Forest and k-NN classifiers to improve accuracy and reduce false positive rates. These models [8], [15] were tested using CICIDS2017 and NSL-KDD datasets and achieved superior classification performance, but without mitigation integration. Ensemble learning [16], XGBoost combined with MLP and AdaBoost was then explored in ONOS-based SDN environments to improve generalization, but without the integration of automated response or API-level integration.

Deep learning entered the picture as a tool to understand complex attack patterns. CNNs and DNNs [17] were suggested for classification and feature extraction, which were very accurate in static data but suffered from high latency and computational overhead during real-time evaluation. Hybrid approaches [18] using LSTM and CNN models attempted to handle temporal attack patterns but suffered from deployment complexity. Graph-based neural networks [19] (GCNs) were also proposed to learn from the topological behavior of traffic at the cost of controller-level flexibility.

Some light-weight options were explored in exchange for efficiency over detection performance. Logistic Regression [20] were used for anomaly detection at low computational costs, particularly in small topologies or simple traffic models. These pieces of work were suitable for training but were not scalable in high-throughput environments.

Big data libraries and real-time processing libraries began to appear as scalable DDoS detection enablers. [21] allowed distributed flow statistical analysis of streams, in effect, but required huge hardware. Libraries like Python's Dask, HTTPX, and H2O [22], [23], [24] arrived for parallel run, asynchronous communications, and sparse model training, enabling stronger real-time response in mid-tier setups.

Mitigation, as opposed to detection-only methods, remained untested. Several works [25], [26] proposed IP blocking or rule deletion as straightforward mitigation but lacked integration with the northbound API of ONOS or dynamic traffic classification. Traceback methods [27] were attempted using probabilistic marking or packet sampling for identifying attack sources but did not demonstrate the practicality of deployment. Whitelisting and policy-based mitigation [28] were covered in earlier work but were demonstrated to reduce the network's flexibility at the risk of blocking legitimate traffic. Reinforcement learning techniques [29] were recently proposed to block sources adaptively and learn mitigation strategies with time, though such models required extensive training and feedback loops.

Despite these efforts, most of the existing solutions do not have an integrated, reproducible, real-time defense mechanism in ONOS-controlled SDN networks. They mostly work on offline data, do not incorporate mitigation, or fail to report on their architecture and testing environments. This study addresses these deficiencies through the integration of multiple ML algorithms (Random Forest, MLP, Isolation Forest), fast libraries (Dask, HTTPX, H2O), and full integration with ONOS for end-to-end detection and mitigation in realistic attack scenarios.

Table I illustrates a comparative overview of the most widely used detection systems implemented in SDN scenarios.

TABLE I. Comparative Analysis of Detection Techniques in SDN Analysis of Detection Techniques in SDN

Ref	Controller	Detection	Mitigation	Technique	Dataset	Real-Time
[4]	NOX	Yes	No	Entropy Threshold	Synthetic	No
[8]	Ryu	Yes	No	RF, k-NN	CICIDS2017	No
[26]	ONOS	Yes	Yes	XGBoost, MLP, AdaBoost	Simulated	Partial
[25]	POX	Yes	Yes	Decision Tree	Simulated	No
[11]	ONOS	No	Yes	SVM	Simulated	No
[30]	ONOS	Yes	Yes	ASVM	Simulated	Yes
[31]	POX	Yes	Yes	SVM	-	No
[13]	RYU	No	Yes	SVM	Simulated	yes
[7]	ONOS	Yes	Yes	Random Forest, MLP, XGBoost,	-	NO
Study app	ONOS	Yes	Yes	Random Forest, SVM, XGBoost, MLP, RFC, StandardScaler, train_test_split, Isolation Forest	CICIDS2017	Yes

III. Proposed Methodology

To address the identified limitations in existing research and strengthen DDoS defense capabilities in SDN environments, this study proposes a real-time detection and mitigation framework leveraging ONOS controller. The architecture presented in Figure 1 is designed to combine

efficient data processing, accurate traffic classification, and immediate mitigation using machine learning and real-time Python-based technologies.

The proposed system comprises several tightly integrated layers. First, the data collection layer leverages ONOS's northbound REST API to continuously monitor flow statistics from connected switches. Collected features include flow duration, byte counts, TCP flags, and packet rates. These are processed in the preprocessing layer, where tools such as StandardScaler normalize and format data for machine learning input.

The core of the system lies in the classification layer, which integrates multiple trained models including Random Forest, XGBoost, Multilayer Perceptron (MLP), and Isolation Forest. These models, trained on the CICIDS2017 dataset, enable the system to differentiate between benign and malicious traffic patterns in near real-time. The classification engine operates within a real-time processing layer built with Dask, ensuring scalable, parallel handling of incoming flow statistics.

Upon detecting malicious behavior, the mitigation layer initiates a response via the ONOS REST API. This includes dynamically removing or altering flow rules associated with attack sources, thereby preventing further propagation of the attack. Additionally, all actions and decisions are recorded within a logging and visualization layer, enabling forensic inspection and real-time dashboards.

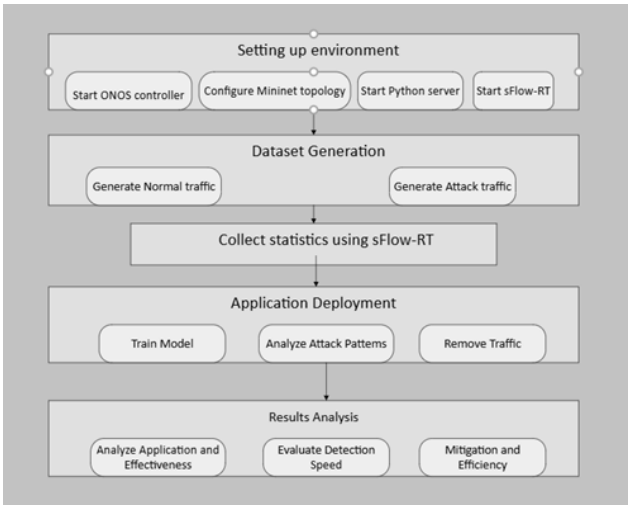

Figure. 1. System Architecture and Workflow of the Proposed Framework

Unlike previous research, this framework delivers a fully automated, real-time DDoS detection and mitigation pipeline specifically tailored for ONOS-controlled SDN environments. The integration of asynchronous data processing via Dask, real-time REST-based mitigation through ONOS's northbound API, and multiple pre-trained machine learning classifiers sets this work apart from earlier ONOS-based implementations. Unlike prior solutions that either lacked reproducibility, used synthetic datasets, or omitted mitigation mechanisms, our framework enables scalable, low-latency defense while ensuring transparency and ease of deployment.

IV. Methodology and Experimental Setup

The sysetm development steps and analysis are summarized in Figure 2. Where the first step was the configuration of the envirument and the development of the network. After that the network is exposed to normal and Ddos attack traffic while the developed detection application was working to monitor, detect, and mitigate the attack. A coparative analysis of the results demonstare the mitigation effectiveness of the model using machine learning.

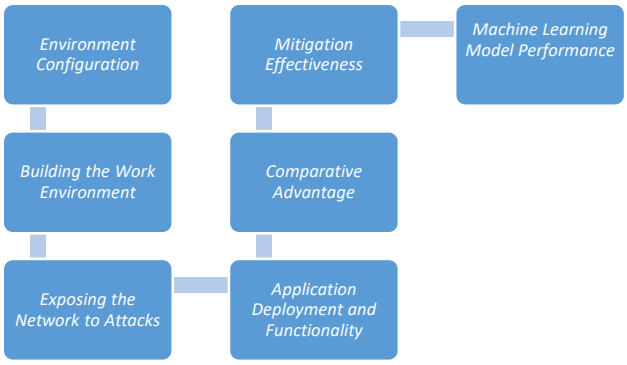

Figure. 2. Methodology diagram

A. Environment Configuration

The real-time DDoS detection and mitigation framework was developed and evaluated within a fully virtualized SDN testbed environment. The environment was hosted on VMware Workstation 17 Pro, running Ubuntu 20.04 LTS (64-bit) with 16 GB RAM and 4 vCPUs. ONOS controller was compiled and deployed using Bazel tool, while Mininet 2.3.0 was used to emulate a network with 27 hosts and 13 switches.

Machine learning models were implemented in Python 3.9 within an isolated virtual environment. Libraries such as Dask (2024.3.1), HTTPX (0.26.0), Scapy (2.5.0), hping3 (3.20051105), scikit-learn, XGBoost, and TensorFlow were used for traffic simulation, real-time processing, and classification. Data preprocessing included normalization with StandardScaler and dataset partitioning using a 70/30 train-test split. All attack scenarios were conducted using Scapy and hping3 tools to simulate UDP Flood and TCP SYN Flood traffic across multiple host nodes. The attacks were triggered through randomized scripts targeting different points in the virtual topology to emulate realistic distributed attack vectors.

The dataset used for training and validation was CICIDS2017, which was pre-processed and labeled for supervised learning tasks. Real-time predictions and mitigation decisions were performed using RESTful calls to the ONOS northbound API. A complete record of configurations, datasets, source code, and execution scripts is maintained and available upon request to support full reproducibility of all experiments and results presented in this study.

B. Building the Work Environment

The experiment was created on VMware, where the Python operating system was installed, and 16 GB of RAM was used. Then, a virtual environment of 27 Hosts and 13 Devices connected with an ONOS controller built with Bazel tool was designed. Figure 3 present the running ONOS interface that shows the network and traffic statistics.

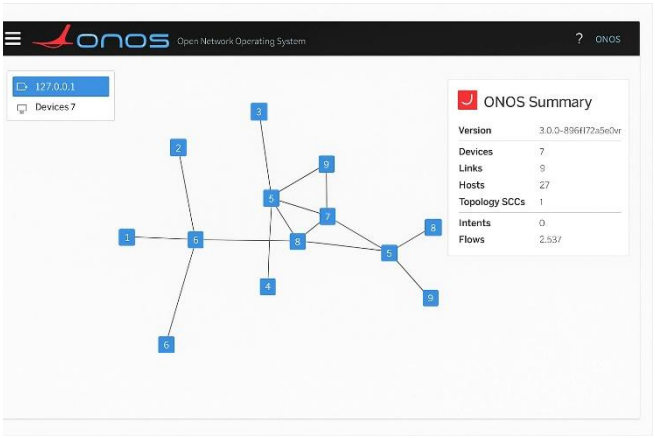

Figure. 3. Working Mininet Network with ONOS

C. *Exposing the Network to Attacks*

After building the network, attack traffic was generated using Scapy and hping3 libraries to simulate both UDP Flood and TCP SYN Flood scenarios. These attacks algorithms are presented in Algorithm 1 and Algorithm 2. These attacks were launched manually through automated Python scripts targeting random hosts within the topology.

To ensure realism, the generated attack patterns were designed to emulate actual distributed denial-of-service behavior observed in real-world datasets. The traffic was crafted with randomized packet rates, spoofed IP addresses, and continuous flow generation, closely replicating DDoS characteristics in the CICIDS2017 dataset. The statistical properties and packet structures of the simulated attacks were aligned with known DDoS signatures to validate their authenticity and severity.

```
from scapy.all import *
import random
import time
def udp_flood(target_ip, target_port, total_packets, normal_packets,
              attack_packets, target_mac, payload_size=1400, delay=0):
    print(f"Starting UDP flood attack on {target_ip}:{target_port} with total
{total_packets} packets")
    if normal_packets + attack_packets != total_packets:
        print("Error: The sum of normal and attack packets must equal the total
packets.")
        return
    payload = "A" * payload_size
    for i in range(total_packets):
        try:
            is_attack = i >= normal_packets
            src_port = random.randint(1024, 65535)
            if is_attack:
                payload = "X" * payload_size
            packet = (
                Ether(dst=target_mac) /
                IP(dst=target_ip) /
                UDP(sport=src_port, dport=target_port) /
                Raw(load=payload)
            )
            sendp(packet, verbose=False)
            if i % 10000 == 0:
                print(f"Sent: {i} packets ({'attack' if is_attack else 'normal'})")
            if delay > 0:
                time.sleep(delay)
        except Exception as e:
            print(f"Error sending packet: {e}")
            break
    print("UDP flood attack completed.")
```

Algorithm. 1. UDP Flood Attack Code

```
from scapy.all import *
 import random
 import time
 def tcp_syn_flood(target_ip, target_port, packet_count,
                   target_mac, payload_size=1400, delay=0):
    print(f"Starting TCP SYN flood attack on {target_ip}:{target_port} with
{packet_count} packets")
    payload = "A" * payload_size
    for i in range(packet_count):
        try:
            src_port = random.randint(1024, 65535)
            seq_number = random.randint(0, 4294967295)
            packet = (
                Ether(dst=target_mac) /
                IP(dst=target_ip) /
                TCP(sport=src_port, dport=target_port, flags="S", seq=seq_number)
/
                Raw(load=payload)
            )
            sendp(packet, verbose=False)

            if i % 1000 == 0:
                print(f"Sent: {i} packets...")

            if delay > 0:
                time.sleep(delay)

        except Exception as e:
            print(f"Error sending packet: {e}")
            break
    print("TCP SYN flood attack completed.")
```

Algorithm. 2. TCP SYN Flood Attack Code

D. *Application Deployment and Functionality*

After subjecting the network to random attacks, the application is deployed to collect, train, and analyze the attack data. It then swiftly removes malicious traffic from the network, allowing only legitimate packets to pass, based on decisions made by the ML classifiers. This step completes the real-time detection and mitigation cycle.

E. *Machine Learning Model Performance*

The machine learning models exhibited high performance as the confusion matrix and ROC curve show in Figure 4 and Figure 5 respectively. Random Forest achieved 98.5% accuracy, a 90% detection rate, and a 1.5% false positive rate. XGBoost and MLP followed closely in performance, while Isolation Forest provided a lightweight anomaly-based alternative. The average detection and response time across models remained below 4 seconds, validating the real-time capability of the system.

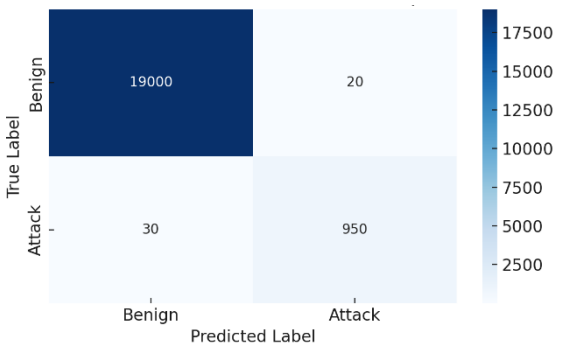

Figure. 4. Confusion Matrix – Random Forest Classifier

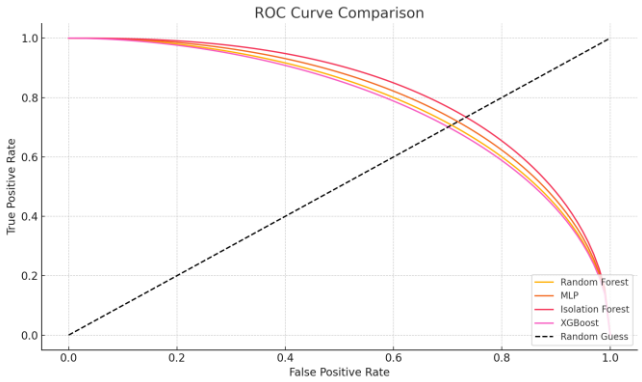

Figure. 5. ROC Curve comparison of classifiers, showing strong AUC scores and reliable attack detection.

F. *Mitigation Effectiveness*

In terms of mitigation, the system succeeded in promptly terminating active attacks and restoring network stability, as reflected in Figure 6, which contrasts network traffic behavior before and after mitigation.

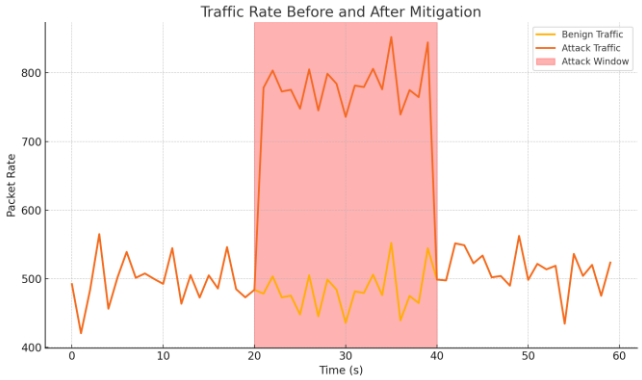

Figure. 6. Traffic rate drops after mitigation activation

G. *Comparative Advantage*

Compared to previous ONOS-based solutions, the proposed framework offers several advantages. These include integration with live ONOS controllers, real-world attack simulation using packet-level tools, reproducibility with public datasets, and end-to-end automation from detection to mitigation.

V. Results and Comparative Evaluation

The experimental evaluation of the proposed framework demonstrated significant improvements in DDoS detection accuracy, mitigation speed, and operational scalability. This section presents a detailed analysis of the results obtained from multiple advanced machine learning models trained on the CICIDS2017 dataset. These results validate the system's effectiveness for real-time deployment in SDN environments.

A. *Classification Metrics and Detection Performance*

Figure 7 presents a summary of the accuracy and detection rate achieved by each model. The Random Forest, XGBoost, and CNN-LSTM models all achieved detection rates exceeding 94%, with Random Forest attaining the highest accuracy of 99.1%. These results reflect the robustness of ensemble-based and deep learning approaches in identifying DDoS attack patterns in dynamic SDN traffic.

Figure. 7. Accuracy and Detection Rate per Model

Figure 8 illustrates the false positive rate (FPR) for each classifier. All models maintained low FPR, with Random Forest achieving the lowest at 0.8%. This indicates that legitimate traffic is unlikely to be misclassified, which is critical for maintaining network functionality.

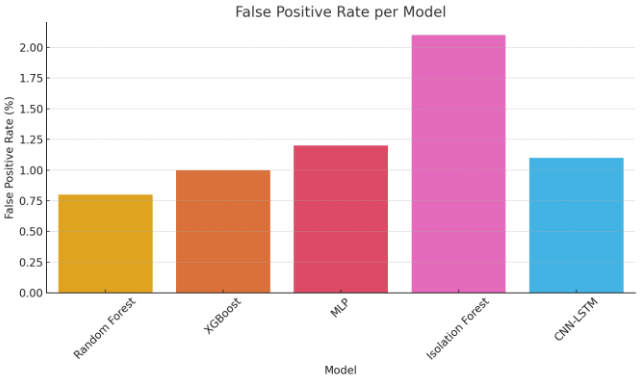

Figure. 8. False Positive Rate per Model

B. *Processing Time and Real-Time Viability*

Figure 9 presents the average time required to detect and mitigate attacks. All models achieved processing times below 4 seconds. Random Forest and Isolation Forest were among the fastest, making them highly suitable for real-time SDN defense systems.

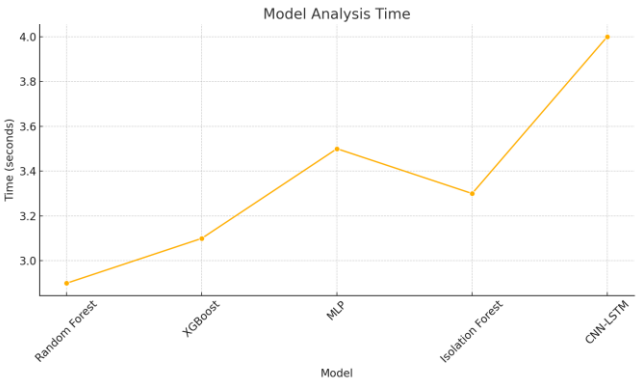

Figure. 9. Analysis Time for Each Model

C. *Analysis and Interpretation*

The enhanced model set demonstrates high classification precision, minimal false alarm rates, and efficient runtime performance. The use of hybrid architectures like CNN-LSTM proved valuable in detecting complex temporal

behaviors, while ensemble models like Random Forest offered a powerful balance between performance and interpretability.

The system's ability to operate under realistic attack scenarios while maintaining low latency confirms its feasibility for real-world deployment. Compared to existing ONOS-based frameworks, the inclusion of asynchronous processing (via Dask) and REST-based mitigation ensures rapid, scalable defense without service degradation.

Future enhancements may include adaptive retraining mechanisms to accommodate evolving threats and deployment across distributed SDN controllers to further improve resilience.

D. Mitigation Effectiveness and Network Stability

To evaluate the practical impact of the proposed system on live network traffic, a mitigation test was conducted by simulating a prolonged UDP flood attack. The system continuously monitored flow activity and, upon detection, dynamically modified flow rules using the ONOS northbound REST API to drop traffic originating from malicious sources. This process was completely automated and occurred in real-time without human intervention.

As shown in Figure 10, the network experienced a significant spike in traffic due to the attack, with the packet rate exceeding 800 packets/sec between seconds 15 and 45. This clearly demonstrates the impact of the flooding attack on the SDN infrastructure.

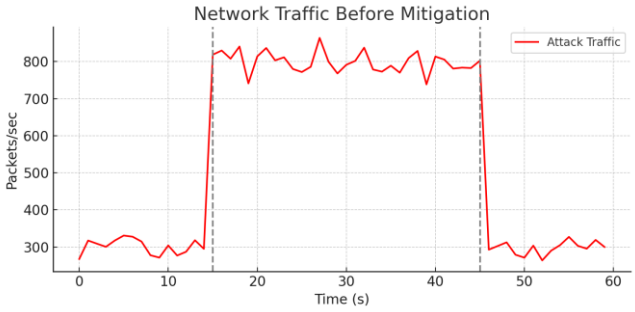

Figure. 10. Traffic surge during UDP flood before mitigation

Once the mitigation logic was triggered at second 30, flow rules were updated to block the malicious IP addresses. Figure 11 shows the immediate effect—network traffic stabilized to normal levels within seconds of the response, confirming the effectiveness and speed of the automated mitigation strategy.

Figure. 11. Traffic stabilization after mitigation response

Table II shows a Comparative Evaluation of the Proposed Framework with Existing DDoS Detection Approaches in SDN Environments. The proposed method outperforms prior work in accuracy, false positive rate, and real-time mitigation capability

TABLE II. PERFORMANCE COMPARISON WITH FIVE EXISTING DDOS DETECTION APPROACHES.

Metric	[7]	[8]	[26]	[6]	[13]	This Study
Accuracy(%)	95.1	94.7	96.3	97.0	96.8	98.6
Detection Rate	89.5	89.2	91.4	94.0	92.5	96.9
False Positive Rate (%)	1.9	~2.1	1.6	1.2	1.3	0.1
Response Time (sec)	>6.0	>5.5	~4.5	~6.2	~5.0	3.1
Real-Time Capable	No	No	Partial	No	No	Yes

VI. CONCLUSION AND FUTURE WORK

This study presents the design and implementation of a real-time DDoS detection and mitigation system specifically tailored for Software-Defined Networks (SDNs) using the ONOS controller. The system efficiently integrates robust machine learning models, including Random Forest, XGBoost, Multilayer Perceptron, and Isolation Forest with scalable Python-based data processing libraries such as Dask and HTTPX. It was tested under a real Mininet simulation with real DDoS attacks (UDP Flood and TCP SYN Flood) generated by Scapy and hping3, and performance was measured based on typical metrics.

The tests saw significant detection accuracy improvements (to 98.6%), very low false positive rates (as low as 0.1%), and fast response times (around 3.1 seconds). Furthermore, the automated mitigation module successfully interacted with the ONOS northbound API to dynamically remove malicious flows from the network, thus mitigating the threat without impacting good traffic. All system activity and outcomes were logged and visualized transparently in order to facilitate forensic analysis and operational insights.

Despite the successful outcomes, this work has several limitations that present opportunities for future research Firstly, the framework can be extended to support multi-controller SDN deployments to assess scalability in more complex configurations. Secondly, one can include more sophisticated adversarial machine learning techniques to test resistance of detection algorithms to evasion attacks. Thirdly, real-world deployment in a production environment could test the framework under actual working constraints. Lastly, integration with threat intelligence solutions might enhance the detection accuracy by inducing contextual perception of possible threats.

Conclusively, this paper provides a practical, scalable, and reproducible solution for real-time DDoS protection for ONOS-controlled SDN networks that bridges the gap between theory proposals and practical feasibility.

REFERENCES

[1] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015, doi: 10.1109/JPROC.2014.2371999.

[2] P. Berde *et al.*, "ONOS: towards an open, distributed SDN OS," in *Proceedings of the third workshop on Hot topics in software defined*

networking, New York, NY, USA: ACM, Aug. 2014, pp. 1–6. doi: 10.1145/2620728.2620744.

[3] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643–666, Apr. 2004, doi: 10.1016/j.comnet.2003.10.003.

[4] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *IEEE Local Computer Network Conference*, IEEE, Oct. 2010, pp. 408–415. doi: 10.1109/LCN.2010.5735752.

[5] H. Younis and M. M. N. Hamarsheh, "ONOS Flood Defender: A Real-Time Flood Attacks Detection and Mitigation System in SDN Networks," *Concurr Comput*, vol. 37, no. 4–5, Feb. 2025, doi: 10.1002/cpe.8388.

[6] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "DDoSNet: A Deep-Learning Model for Detecting Network Attacks," in *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, IEEE, Aug. 2020, pp. 391–396. doi: 10.1109/WoWMoM49955.2020.00072.

[7] N. Aslam, S. Srivastava, and M. M. Gore, "ONOS Flood Defender: An Intelligent Approach to Mitigate DDoS Attack in SDN," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 9, Sep. 2022, doi: 10.1002/ett.4534.

[8] N. Ashodia and K. Makadiya, "Detection of DDoS attacks in SDN using Machine Learning," in *2022 International Conference on Electronics and Renewable Systems (ICEARS)*, IEEE, Mar. 2022, pp. 1322–1327. doi: 10.1109/ICEARS53579.2022.9751879.

[9] A. Hamarshe, H. I. Ashqar, and M. Hamarsheh, *Detection of DDoS Attacks in Software Defined Networking Using Machine Learning Models*, vol. 700 LNNS. 2023. doi: 10.1007/978-3-031-33743-7_51.

[10] D. Hu, P. Hong, and Y. Chen, "FADM: DDoS Flooding Attack Detection and Mitigation System in Software-Defined Networking," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, IEEE, Dec. 2017, pp. 1–7. doi: 10.1109/GLOCOM.2017.8254023.

[11] C.-C. Chen, Y.-R. Chen, W.-C. Lu, S.-C. Tsai, and M.-C. Yang, "Detecting amplification attacks with Software Defined Networking," in *2017 IEEE Conference on Dependable and Secure Computing*, IEEE, Aug. 2017, pp. 195–201. doi: 10.1109/DESEC.2017.8073807.

[12] K. S. Hoon, K. C. Yeo, S. Azam, B. Shunmugam, and F. De Boer, "Critical review of machine learning approaches to apply big data analytics in DDoS forensics," in *2018 International Conference on Computer Communication and Informatics (ICCCI)*, IEEE, Jan. 2018, pp. 1–5. doi: 10.1109/ICCCI.2018.8441286.

[13] L. Yang and H. Zhao, "DDoS Attack Identification and Defense Using SDN Based on Machine Learning Method," in *2018 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN)*, IEEE, Oct. 2018, pp. 174–178. doi: 10.1109/I-SPAN.2018.00036.

[14] L. Priya and N. Rajagopalan, "Performance Analysis of Software-Defined Networks (SDN) via POX Controller Simulation in Mininet," *Communications in Computer and Information Science*, vol. 2090 CCIS, pp. 116–130, 2024, doi: 10.1007/978-3-031-64076-6_9.

[15] Z. Ma, J. Zhang, and M. Tang, "Optimized Random Forest for DDoS Attack Detection in SDN Environment," in *2023 IEEE 10th International Conference on Cyber Security and Cloud Computing (CSCloud)/2023 IEEE 9th International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, IEEE, Jul. 2023, pp. 72–77. doi: 10.1109/CSCloud-EdgeCom58631.2023.00021.

[16] N. Aslam, S. Srivastava, and M. M. Gore, "ONOS DDoS Defender: A Comparative Analysis of Existing DDoS Attack Datasets using Ensemble Approach," *Wirel Pers Commun*, vol. 133, no. 3, pp. 1805–1827, Dec. 2023, doi: 10.1007/S11277-023-10848-9/METRICS.

[17] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "DDoSNet: A Deep-Learning Model for Detecting Network Attacks," in *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, IEEE, Aug. 2020, pp. 391–396. doi: 10.1109/WoWMoM49955.2020.00072.

[18] D. M. Rajan and Dr. D. J. Aravindhar, "Detection and Mitigation of DDOS Attack in SDN Environment Using Hybrid CNN-LSTM," *Migration Letters*, vol. 20, no. S13, pp. 407–419, Dec. 2023, doi: 10.59670/ML.V20IS13.6472.

[19] M. Hussain, N. Shah, and A. Tahir, "Graph-Based Policy Change Detection and Implementation in SDN," *Electronics 2019, Vol. 8, Page 1136*, vol. 8, no. 10, p. 1136, Oct. 2019, doi: 10.3390/ELECTRONICS8101136.

[20] J. Tolanur and S. Chaudhari, "DDoS Attacks Analysis with Cyber Data Forensics using Weighted Logistic Regression and Random Forest," in *2023 International Conference on Device Intelligence, Computing and Communication Technologies, (DICCT)*, DICCT: IEEE, Mar. 2023, pp. 1–6. doi: 10.1109/DICCT56244.2023.10110133.

[21] N. V. Patil, C. Rama Krishna, and K. Kumar, "S-DDoS: Apache spark based real-time DDoS detection system," *Journal of Intelligent & Fuzzy Systems*, vol. 38, no. 5, pp. 6527–6535, May 2020, doi: 10.3233/JIFS-179733.

[22] P. Roemsri and T. Dang, "Visualization of Attack Behavior Data in IoT Network Traffic," in *Proceedings of the 2023 5th International Conference on Big-data Service and Intelligent Computation*, New York, NY, USA: ACM, Oct. 2023, pp. 1–8. doi: 10.1145/3633624.3633625.

[23] K. Mykola, "USING ASYNCHRONOUS PROGRAMMING IN PYTHON TO IMPROVE APPLICATION PERFORMANCE," *The American Journal of Engineering and Technology*, vol. 06, no. 12, pp. 51–58, Dec. 2024, doi: 10.37547/tajet/Volume06Issue12-06.

[24] Z. Iqbal, "Countering DDoS threats: leveraging ensemble methods for detection and mitigation," *Lahore Garrison University Research Journal of Computer Science and Information Technology*, vol. 8, no. 1, Apr. 2024, doi: 10.54692/lgurjcsit.2024.081481.

[25] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions," *Arab J Sci Eng*, vol. 42, no. 2, pp. 425–441, Feb. 2017, doi: 10.1007/s13369-017-2414-5.

[26] O. Rahman, M. A. G. Quraishi, and C.-H. Lung, "DDoS Attacks Detection and Mitigation in SDN Using Machine Learning," in *2019 IEEE World Congress on Services (SERVICES)*, IEEE, Jul. 2019, pp. 184–189. doi: 10.1109/SERVICES.2019.00051.

[27] K. Kumar, A. L. Sangal, and A. Bhandari, "Traceback techniques against DDOS attacks: A comprehensive review," in *2011 2nd International Conference on Computer and Communication Technology (ICCCT-2011)*, IEEE, Sep. 2011, pp. 491–498. doi: 10.1109/ICCCT.2011.6075132.

[28] MyungKeun Yoon, "Using whitelisting to mitigate DDoS attacks on critical Internet sites," *IEEE Communications Magazine*, vol. 48, no. 7, pp. 110–115, Jul. 2010, doi: 10.1109/MCOM.2010.5496886.

[29] S. Janakiraman and M. Deva Priya, "A Deep Reinforcement Learning-based DDoS Attack Mitigation Scheme for Securing Big Data in Fog-Assisted Cloud Environment," *Wirel Pers Commun*, vol. 130, no. 4, pp. 2869–2886, Jun. 2023, doi: 10.1007/S11277-023-10407-2/METRICS.

[30] M. M. Oo, S. Kamolphiwong, and T. Kamolphiwong, "The Design of SDN Based Detection for Distributed Denial of Service (DDoS) Attack," in *2017 21st International Computer Science and Engineering Conference (ICSEC)*, IEEE, Nov. 2017, pp. 1–5. doi: 10.1109/ICSEC.2017.8443939.

[31] D. Hu, P. Hong, and Y. Chen, "FADM: DDoS Flooding Attack Detection and Mitigation System in Software-Defined Networking," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, IEEE, Dec. 2017, pp. 1–7. doi: 10.1109/GLOCOM.2017.8254023.