A Lightweight Flow-Statistics-based DDoS Detection and Mitigation Mechanism in SDN

Yang Yang
Qujing Municipal Bureau of State Secrets Protection
Qujing, China

Shu Jiang
Yuxi E-Government Intranet Information Technology Center
Yuxi, China

Hong Ye
Yuxi Municipal Archives
Yuxi, China

Yanyi She
Yuxi Municipal Target Assessment and Evaluation Center
Yuxi, China

Ming Liu
Yunnan Normal University
Kunming, China

Yuefeng Li
China Mobile Group Yunnan Co., Ltd. Yuxi Branch
Yuxi, China

ABSTRACT
Software-defined networking (SDN) provides centralized control and programmable flow management, making it suitable for dynamic detection and mitigation of distributed denial-of-service (DDoS) attacks. This paper proposes a lightweight flow-statistics-based DDoS detection and mitigation mechanism for SDN. The controller periodically collects OpenFlow flow statistics and extracts four interpretable indicators: aggregated packet rate, source count, flow count, and normalized source-address entropy. A destination is first marked as a candidate attack target when all four traffic conditions are simultaneously satisfied, and a DDoS attack is confirmed only when the suspicious state persists for a predefined duration, thereby suppressing alarms caused by transient traffic bursts. Once an attack is confirmed, the controller dynamically installs high-priority source-specific drop rules and uses an OpenFlow barrier mechanism to confirm mitigation-rule processing. Experiments were conducted in a Mininet environment using Ryu, Open vSwitch, and OpenFlow 1.3. Low-, medium-, and high-rate UDP DDoS scenarios were independently repeated three times, with observed packet rates of 481.81 ± 1.09 pps, 964.00 ± 0.98 pps, and 1606.10 ± 1.08 pps, respectively. The attack-confirmation duration remains approximately 1 s, while the mean barrier-confirmed mitigation delay ranges from 0.7938 ms to 0.8878 ms. A 0.5-s multi-source burst satisfies the traffic-feature conditions but is not confirmed as a DDoS attack, demonstrating the effectiveness of the suspicious-duration mechanism. Additional experiments show that TCP SYN and ICMP flooding attacks can also be successfully detected and mitigated. The installed source-specific rules effectively block attack traffic while preserving legitimate communication with the victim host.

General Terms
Security, Algorithms, Network Management

Keywords
Software-defined networking; DDoS detection; flow statistics; normalized entropy; OpenFlow; dynamic mitigation

1. INTRODUCTION
Distributed denial-of-service (DDoS) attacks remain a serious threat to modern networks. By generating large volumes of traffic from distributed sources, attackers can exhaust the bandwidth, processing capacity, or service resources of a victim host. In traditional networks, DDoS detection and mitigation often rely on dedicated security devices, manually configured filtering rules, or additional monitoring systems, which may limit the flexibility and timeliness of attack response.

Software-defined networking separates the control plane from the data plane and provides centralized and programmable network control[1,2]. Through OpenFlow, an SDN controller can periodically obtain flow-level statistics from switches and dynamically install forwarding or filtering rules[1]. This centralized visibility and programmability make SDN particularly suitable for building reactive security mechanisms.

Existing SDN-based DDoS detection approaches mainly include statistical or rule-based detection, entropy-based detection, and machine-learning-based classification[3,4]. Machine-learning-based approaches can provide sophisticated traffic classification capabilities, but they normally require additional feature engineering, training datasets, offline model training, or online inference. For small-scale or resource-constrained SDN deployments, a mechanism with low implementation complexity and directly interpretable decision logic remains practically useful.

To address this issue, this paper proposes a lightweight flow-statistics-based DDoS detection and mitigation mechanism for SDN. The proposed method does not inspect packet payloads and does not require offline training or online machine-learning inference. Instead, the controller periodically obtains OpenFlow flow statistics and jointly analyzes aggregated packet rate, source count, flow count, normalized source-address entropy, and suspicious-state duration. A destination is first marked as a candidate attack target when all traffic-feature conditions are satisfied. The attack is confirmed only when the suspicious state persists for a predefined duration. Once confirmed, source-specific high-priority drop rules are dynamically installed at the OpenFlow switch.

The main contributions of this paper are summarized as follows:

1)A lightweight DDoS detection mechanism based solely on OpenFlow flow statistics is developed without packet-payload inspection, offline model training, or online machine-learning inference.

2)A two-stage detection mechanism combining aggregated packet rate, source count, flow count, normalized source entropy, and suspicious-state duration is designed to distinguish persistent multi-source attacks from single-source high-rate traffic and short-lived multi-source bursts.

3)A source-specific mitigation mechanism based on high-priority OpenFlow drop rules is implemented and evaluated under repeated UDP DDoS attacks and additional TCP SYN and ICMP flooding scenarios, demonstrating selective attack blocking while preserving legitimate communication.

2. RELATED WORK

Early SDN-based DDoS detection studies mainly exploited the centralized visibility provided by OpenFlow. Braga et al. [5] proposed a lightweight flooding-attack detection method based on flow-level traffic characteristics collected through NOX/OpenFlow. Mousavi and St-Hilaire[6] employed traffic entropy for early detection of DDoS attacks targeting SDN controllers. Giotis et al.[7] combined OpenFlow and sFlow to provide scalable anomaly detection and mitigation. These studies demonstrated that statistical and information-theoretic characteristics can provide useful attack indicators without packet-payload inspection.

Subsequent studies have investigated more sophisticated statistical, entropy-based, and learning-based detection mechanisms. Sahoo et al.[8] focused on low-rate DDoS attacks in SDN-based data-center networks using information-distance metrics. Cui et al.[9] combined flow-table characteristics, entropy, and machine learning for attack detection and defense. Dong and Sarem[10] proposed an improved KNN-based DDoS detection method, while Manso et al.[11] developed an SDN-based intrusion-detection system supporting early detection and mitigation. More recently, Ahalawat et al.[12] employed Rényi entropy together with packet-drop-based mitigation for low-rate DDoS attacks, while Jia et al.[13] proposed a lightweight SDN detection scheme that uses coarse-grained port monitoring to selectively trigger fine-grained detection. These studies show that statistical and lightweight traffic indicators remain useful for reducing detection complexity while supporting timely response.

Machine-learning and hybrid approaches have also been widely investigated to improve traffic discrimination. Liu et al.[14] introduced a two-level architecture in which information entropy is first used for coarse-grained detection and deep learning is subsequently applied to suspicious traffic. Eliyan and Di Pietro[15] proposed DeMi, which integrates attack detection and mitigation to maintain SDN availability. More recent studies have combined entropy with machine-learning-based clustering[16] or employed deeper feature-learning architectures for SDN DDoS detection[17]. Although these approaches can improve traffic characterization and classification capability, they generally introduce additional feature-processing, training, or inference requirements.

Compared with these approaches, this paper focuses on a lightweight and directly deployable detection-mitigation loop without offline model training or online machine-learning inference. Only OpenFlow flow statistics are required, and the decision is based on four interpretable indicators: aggregated packet rate, source count, flow count, and normalized source entropy. In addition, a suspicious-duration confirmation stage is introduced to prevent transient multi-source bursts from immediately triggering mitigation. Once an attack is confirmed, source-specific high-priority drop rules are installed directly at the OpenFlow switch. Therefore, the proposed mechanism emphasizes implementation simplicity, interpretability, transient-burst suppression, and integrated mitigation rather than complex traffic classification.

3. PROPOSED METHOD

3.1 System Architecture

The proposed system consists of an SDN controller, an OpenFlow switch, normal hosts, attack hosts, and a victim host. The controller is implemented using the Ryu SDN framework, while the data plane is constructed using Open vSwitch. The controller communicates with the switch through OpenFlow 1.3 and periodically requests flow statistics from the switch. In the experimental topology, h1, h2, and h3 are configured as attack hosts, h4 and h5 as legitimate hosts, and h6 as the victim host. All six hosts are connected to a single OpenFlow switch.

As illustrated in Figure 1, the controller periodically collects flow statistics from the OpenFlow switch and extracts traffic features for each destination. Traffic satisfying the joint feature conditions enters a suspicious state and is confirmed as a DDoS attack only if the suspicious state persists for the predefined duration. Once confirmed, source-specific high-priority drop rules are installed to mitigate the attack while preserving legitimate communication.

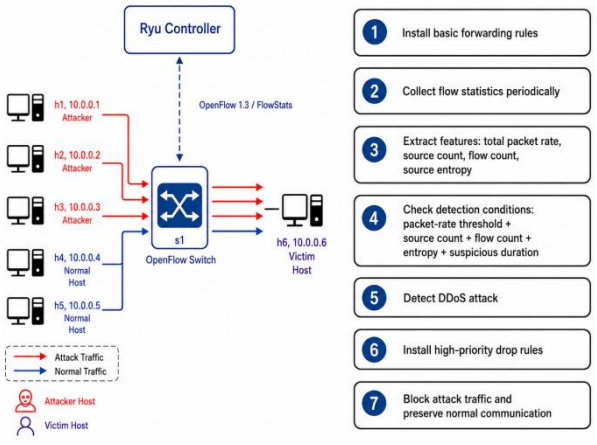


Figure 1. System architecture and workflow of the proposed SDN-based DDoS detection and mitigation mechanism.

3.2 Flow Statistics and Feature Extraction

The controller periodically sends OpenFlow flow-statistics requests to the switch. For each relevant IPv4 forwarding flow, the controller obtains the packet count, byte count, source IP address, destination IP address, and IP protocol.

The packet rate of the i-th flow is calculated using the difference between two consecutive statistics samples:

$$R_i(t) = \frac{P_i(t) - P_i(t - \Delta t)}{\Delta t} \quad (1)$$

where $P_i(t)$ represents the packet count of the i-th flow at time t, and Δt represents the statistics-collection interval.

To reduce the influence of stale or nearly inactive flow-table entries, only flows whose current packet rate exceeds a predefined active-flow threshold are included in subsequent traffic aggregation.

For each destination host d, all active flows having the same destination IP address are aggregated. The total packet rate toward destination d is calculated as

$$R_d = \sum_{i \in F_d} R_i \quad (2)$$

where F_d denotes the set of active flows whose destination address is d.

In addition to the aggregated packet rate, the controller calculates the number of active source IP addresses N_{src}, the number of active flows N_{flow}, and the normalized source-address entropy.

For each source j, its traffic proportion is defined as

$$p_j = \frac{R_j}{R_d} \quad (3)$$

where R_j represents the aggregated packet rate generated by source j toward destination d.

The conventional Shannon source entropy is

$$H_d = -\sum_{j=1}^{N_{src}} p_j \log_2 p_j \quad (4)$$

To make the entropy value comparable when the number of active sources changes, normalized source entropy is used:

$$H_d^{norm} = \begin{cases} \frac{H_d}{\log_2 N_{src}}, N_{src} > 1 \\ 0, \quad N_{src} \leq 1 \end{cases} \quad (5)$$

The normalized entropy is therefore bounded between 0 and 1. A value close to 1 indicates that traffic is relatively evenly distributed among multiple active sources. In the three-source flooding scenarios used in this study, the attack traffic is nearly evenly distributed, resulting in normalized entropy values close to 1.

3.3 Candidate Detection, Duration Confirmation, and Mitigation

The proposed mechanism adopts a two-stage attack-detection procedure. In the first stage, a destination is classified as a candidate attack target only when all four traffic-feature conditions are simultaneously satisfied:

$$R_d \geq T_R, \; N_{src} \geq T_{src}, \; N_{flow} \geq T_{flow}, \; H_d^{norm} \geq T_H \quad (6)$$

where T_R, T_{src}, T_{flow} , and T_H denote the thresholds of aggregated packet rate, source count, flow count, and normalized source entropy, respectively.

In the second stage, meeting the conditions in (6) alone does not confirm a DDoS attack. Instead, the destination enters a suspicious state and the controller starts a suspicious-state timer. An attack is confirmed only when

$$D_s \geq T_D \quad (7)$$

where D_s represents the duration for which the candidate conditions remain continuously satisfied, and T_D is the suspicious-duration threshold. If any condition in (6) becomes false before T_D is reached, the suspicious-state timer is reset. Therefore, a transient traffic burst may satisfy the traffic-feature conditions temporarily but will not be confirmed as a DDoS attack unless the suspicious state persists for the required duration.

Once a DDoS attack is confirmed, the controller installs one high-priority OpenFlow drop rule for each detected attack source. In the prototype, the mitigation rule matches the input port, IPv4 source address, and victim destination address. The rule contains no forwarding action and therefore directly drops matched packets at the switch.

The mitigation rules are assigned a priority of 200 and an idle timeout of 60 s. Consequently, attack-specific rules have a higher priority than ordinary forwarding rules and are automatically removed after the corresponding attack traffic becomes inactive.

To obtain a more reliable measurement of mitigation latency, the controller issues an OpenFlow Barrier Request after sending the mitigation FlowMod messages. The mitigation delay is measured from the issuance of the mitigation-rule batch until receipt of the corresponding Barrier Reply. Thus, the reported value represents the barrier-confirmed mitigation delay rather than only the local execution time required to send a FlowMod message.

In the prototype, the flow-statistics polling interval is 1 s. The aggregated packet-rate threshold is set to T_R = 400 pps. The minimum source-count and flow-count thresholds are both set to 2, the normalized entropy threshold is T_H = 0.8, and the suspicious-duration threshold is T_D = 1 s. The minimum active-flow rate is set to 1 packet/s.

The packet-rate threshold T_R = 400 pps was empirically selected based on preliminary traffic observations in the experimental topology. Boundary tests confirmed that multi-source traffic below this threshold did not enter the candidate state, whereas traffic exceeding the threshold was marked as suspicious when the remaining source-count, flow-count, and normalized-entropy conditions were simultaneously satisfied.

4. EXPERIMENTS AND RESULTS

4.1 Experimental Setup

The experimental data plane is implemented using Open vSwitch[18], the network topology is emulated using Mininet[19], and the controller is implemented using Ryu[20].

The topology contains one OpenFlow switch and six hosts, where h1, h2, and h3 act as attack sources, h4 and h5 as legitimate hosts, and h6 as the victim. To evaluate detection performance under different attack intensities, three UDP flooding scenarios are configured. In each scenario, h1, h2, and h3 simultaneously transmit UDP traffic toward h6 at 150, 300, and 500 pps per source for the low-, medium-, and high-rate scenarios, respectively. The low-rate scenario is configured slightly above the aggregate packet-rate threshold, while the medium- and high-rate scenarios progressively increase the offered traffic load.

Each attack-intensity experiment was independently repeated three times. The observed packet rate, attack-confirmation duration, and barrier-confirmed mitigation delay are reported as mean ± standard deviation. The reported packet rates were measured directly by the controller from OpenFlow flow statistics rather than inferred from the configured sending rates.

Table 1. Attack scenario configuration

Scenario	Attack Sources	Victim	Offered Rate
Low-rate attack	h1, h2, h3	h6	150 pps × 3
Medium-rate attack	h1, h2, h3	h6	300 pps × 3
High-rate attack	h1, h2, h3	h6	500 pps × 3

4.2 Repeated Detection Results under Different Attack Intensities

The experimental results of the three attack-intensity scenarios are shown in Table 2.

Table 2. Detection results under different attack intensities

Scenario	Total PPS	Src. /Flow	H_{norm}	Confirmation Duration (s)	Mitigation Delay (ms)
Low-rate attack	481.81 ± 1.09	3/3	1.000	1.0011 ± 0.0002	0.8280 ± 0.1805
Medium-rate attack	964.00 ± 0.98	3/3	1.000	1.0007 ± 0.0005	0.7938 ± 0.0627
High-rate attack	1606.10 ± 1.08	3/3	1.000	1.0010 ± 0.0004	0.8878±0.2715

The mechanism successfully detects the attack in all three runs at each intensity, corresponding to nine successful detections across the three UDP flooding scenarios. The measured total packet rate increases consistently with the configured attack intensity, from 481.81 ± 1.09 pps in the low-rate scenario to 1606.10 ± 1.08 pps in the high-rate scenario. The relatively small standard deviations indicate stable traffic generation and OpenFlow-based measurement across the repeated runs.

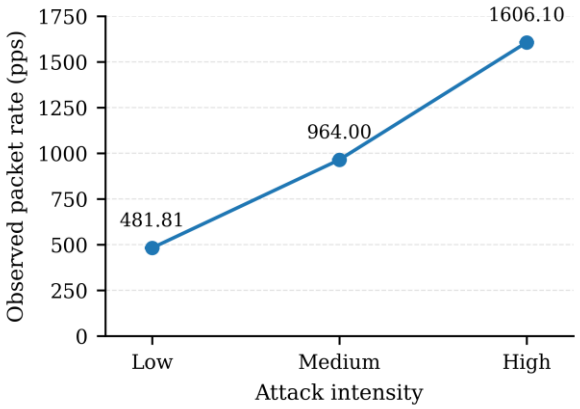


Figure 2. Observed packet rates under different attack intensities.

As shown in Figure 2, the observed packet rate increases consistently with the configured attack intensity, while the relatively small error bars indicate stable measurements across the repeated runs.

In all three scenarios, three attack sources and three active attack flows are identified, while the normalized source entropy remains approximately 1.0. This is expected because the generated attack traffic is nearly evenly distributed among the three sources. Therefore, the entropy feature primarily reflects the balanced multi-source traffic distribution rather than the absolute attack intensity. Increasing the packet rate does not substantially change the entropy value as long as the relative traffic contribution of the attacking hosts remains similar.

The attack-confirmation duration remains close to 1 s across all three attack intensities. This behavior is consistent with the proposed two-stage mechanism, because confirmation is primarily governed by the suspicious-duration threshold $T_D = 1$ s after the traffic-feature conditions have been satisfied. Consequently, a higher attack rate does not substantially shorten the confirmation time once the candidate conditions are met. The barrier-confirmed mitigation delay also remains below 1 ms and does not increase monotonically with packet rate, indicating that, in the current single-switch Mininet environment, the mitigation-rule processing delay is not strongly affected by the offered attack intensity.

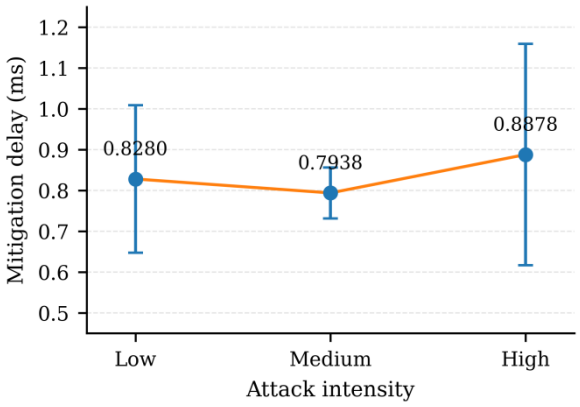


Figure 3. Barrier-confirmed mitigation delays under different attack intensities.

As shown in Figure 3, the barrier-confirmed mitigation delays remain within a narrow sub-millisecond range across the three attack intensities, with no clear increasing trend as the packet rate rises.

4.3 Control and Suspicious-Duration Verification

Table 3. Control and duration-confirmation verification

Scenario	Observed PPS	Src. /Flow	H_{norm}	Candidate	Confirmed
Normal traffic	Below threshold	—	—	No	No
Single-source high-rate	> 400	1/1	0	No	No
0.5s multi-source burst	949.28	3/3	1.000	Yes	No
Persistent multi-source attack	2076.80	3/3	1.000	Yes	Yes

Under normal traffic, the aggregate packet rate remains below the configured threshold and no candidate or confirmed attack state is generated. In the single-source high-rate scenario, the packet-rate threshold is exceeded, but the traffic is rejected at the candidate stage because only one active source and one active flow are observed. For the 0.5-s multi-source burst, the observed packet rate reaches 949.28 pps with three active sources, three active flows, and a normalized entropy of 1.000. Although this traffic satisfies the four feature conditions and enters the candidate state, it is not confirmed because the suspicious state lasts for less than 1 s. In contrast, the persistent multi-source attack reaches 2076.80 pps and remains suspicious long enough to be confirmed and mitigated.

These results demonstrate the different roles of the traffic-feature conditions and the duration-confirmation mechanism. A high packet rate alone is insufficient to trigger an alarm, as demonstrated by the single-source case. In addition, even multi-source traffic satisfying all four feature conditions does not immediately cause a DDoS alarm when the abnormal state is only transient. Therefore, the proposed mechanism considers

both traffic-distribution characteristics and temporal persistence.

The control experiments further illustrate the complementary roles of the feature conditions and temporal persistence. In the single-source high-rate case, the packet-rate threshold is exceeded, but the source-count and flow-count requirements are not satisfied; therefore, the traffic is not considered a candidate distributed attack. In contrast, the 0.5-s multi-source burst satisfies all four traffic-feature conditions and temporarily enters the candidate state, but it is not confirmed because the abnormal condition does not persist for the required duration. These two cases demonstrate that neither a high packet rate nor a transient multi-source pattern alone is sufficient to trigger mitigation.

By combining traffic intensity, source diversity, flow diversity, normalized entropy, and suspicious-state duration, the proposed mechanism introduces multiple decision constraints before installing blocking rules. This design helps reduce unnecessary mitigation caused by isolated high-rate flows or short-lived traffic bursts while still allowing persistent multi-source flooding attacks to be confirmed and mitigated.

4.4 Detection under Different Flooding Protocols

The main attack-intensity experiments use UDP flooding. To examine whether the flow-statistics-based detection logic is also applicable to other common flooding attacks, additional TCP SYN and ICMP flooding scenarios are evaluated using the same three attack hosts and victim. As summarized in Table 4, the TCP SYN flood reaches an observed packet rate of 1241.75 pps and is confirmed after 1.0010 s, with a barrier-confirmed mitigation delay of 0.7496 ms. Similarly, the ICMP flood reaches 1241.41 pps, is confirmed after 1.0013 s, and exhibits a mitigation delay of 0.9331 ms. Both attacks are successfully detected and mitigated.

Table 4. Detection results under different flooding protocols

Attack Type	Observed PPS	Confirmation Duration(s)	Mitigation Delay(ms)	Result
UDP Flood	1606.10 ± 1.08	1.0010 ± 0.0004	0.8878 ± 0.2715	Detected and mitigated
TCP SYN Flood	1241.75	1.0010	0.7496	Detected and mitigated
ICMP Flood	1241.41	1.0013	0.9331	Detected and mitigated

These results demonstrate that the same flow-statistics-based decision logic can detect the UDP, TCP SYN, and ICMP flooding attacks evaluated in this study. The confirmation durations remain close to the configured 1-s threshold across all three protocols, while the barrier-confirmed mitigation delays remain below 1 ms. This consistency is expected because the proposed mechanism relies primarily on aggregated packet rate, source count, flow count, and source-distribution characteristics rather than packet payloads or protocol-specific application signatures. Therefore, different flooding protocols can be detected when they exhibit similar persistent multi-source traffic patterns toward the same victim. Nevertheless, the current experiments cover only three common flooding protocols, and broader protocol independence requires further validation.

4.5 Mitigation Verification

After an attack is confirmed, the controller installs priority-200 source-specific drop rules for the identified attack sources. These rules take precedence over ordinary forwarding entries and directly discard traffic from the detected sources to the victim.

```
ming@ubuntu:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows s1 | grep priority=200
 cookie=0x0, duration=70.151s, table=0, n_packets=28160, n_bytes=8391680, idle_timeout=60,
priority=200,ip,nw_src=10.0.0.3,nw_dst=10.0.0.6 actions=drop
 cookie=0x0, duration=70.151s, table=0, n_packets=28156, n_bytes=8390488, idle_timeout=60,
priority=200,ip,nw_src=10.0.0.1,nw_dst=10.0.0.6 actions=drop
 cookie=0x0, duration=70.151s, table=0, n_packets=28158, n_bytes=8391084, idle_timeout=60,
priority=200,ip,nw_src=10.0.0.2,nw_dst=10.0.0.6 actions=drop
ming@ubuntu:~$
```

Figure 4. Installed high-priority source-specific drop rules after DDoS confirmation.

As shown in Figure 4, the installed rules match the detected source addresses and victim destination and contain no forwarding action. Their non-zero packet counters confirm that the rules actively match and discard attack packets.

The impact of mitigation on end-to-end communication is further evaluated using ping tests. Traffic from h1, h2, and h3 toward h6 experiences 100% packet loss after the source-specific mitigation rules are installed, whereas legitimate hosts h4 and h5 continue to communicate with h6 with 0% packet loss. This result indicates that all three identified attack sources are successfully blocked, while both legitimate hosts retain normal connectivity with the victim. Therefore, the proposed mitigation strategy achieves selective traffic suppression rather than destination-wide isolation, reducing the impact of mitigation on legitimate communication.

5. CONCLUSION

This paper presents a lightweight flow-statistics-based DDoS detection and mitigation mechanism for software-defined networking. The controller periodically collects OpenFlow flow statistics and jointly analyzes aggregated packet rate, source count, flow count, and normalized source-address entropy. A two-stage decision process is adopted: traffic satisfying all feature conditions is first marked as a candidate attack, and the attack is confirmed only when the suspicious state persists for a predefined duration. Once confirmed, source-specific high-priority drop rules are dynamically installed at the OpenFlow switch.

Experiments in a Ryu-Mininet-Open vSwitch environment show that the mechanism consistently detects low-, medium-, and high-rate UDP flooding attacks across three independent runs per scenario. Attack confirmation remains close to the configured 1-s duration threshold, while the mean barrier-confirmed mitigation delay remains below 1 ms across all three attack intensities.

Control experiments further show that single-source high-rate traffic and a 0.5-s multi-source burst do not trigger confirmed DDoS alarms, whereas persistent multi-source traffic is successfully detected.

Additional TCP SYN and ICMP experiments demonstrate applicability to multiple flooding protocols. Source-specific mitigation blocks the identified attack sources while preserving legitimate communication with the victim.

The current evaluation is limited to a small single-switch Mininet topology and fixed detection thresholds. Future work will investigate adaptive threshold selection and evaluate the proposed mechanism under larger network topologies, more diverse traffic patterns, and additional DDoS attack types.

6. REFERENCES

[1] N. McKeown, T. Anderson, H. Balakrishnan, et al., "OpenFlow: Enabling innovation in campus networks," ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, 2008.

[2] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, et al., "Software-defined networking: A comprehensive survey," Proc. IEEE, vol. 103, no. 1, pp. 14–76, 2014.

[3] Q. Yan, F. R. Yu, Q. Gong, et al., "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," IEEE Commun. Surveys Tuts., vol. 18, no. 1, pp. 602–622, 2015.

[4] J. Singh and S. Behal, "Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions," Comput. Sci. Rev., vol. 37, Art. no. 100279, 2020.

[5] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in Proc. IEEE Local Computer Network Conf., 2010, pp. 408–415.

[6] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in Proc. 2015 Int. Conf. Computing, Networking and Communications (ICNC), 2015, pp. 77–81.

[7] K. Giotis, C. Argyropoulos, G. Androulidakis, et al., "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," Comput. Netw., vol. 62, pp. 122–136, 2014.

[8] K. S. Sahoo, D. Puthal, M. Tiwary, et al., "An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics," Future Gener. Comput. Syst., vol. 89, pp. 685–697, 2018.

[9] J. Cui, M. Wang, Y. Luo, et al., "DDoS detection and defense mechanism based on cognitive-inspired computing in SDN," Future Gener. Comput. Syst., vol. 97, pp. 275–283, 2019.

[10] S. Dong and M. Sarem, "DDoS attack detection method based on improved KNN with the degree of DDoS attack in software-defined networks," IEEE Access, vol. 8, pp. 5039–5048, 2019.

[11] P. Manso, J. Moura, and C. Serrão, "SDN-based intrusion detection system for early detection and mitigation of DDoS attacks," Information, vol. 10, no. 3, Art. no. 106, 2019.

[12] A. Ahalawat, K. S. Babu, A. K. Turuk, et al., "A low-rate DDoS detection and mitigation for SDN using Renyi entropy with packet drop," J. Inf. Secur. Appl., vol. 68, Art. no. 103212, 2022.

[13] K. Jia, C. Liu, Q. Liu, et al., "A lightweight DDoS detection scheme under SDN context," Cybersecurity, vol. 5, no. 1, Art. no. 27, 2022.

[14] Y. Liu, T. Zhi, M. Shen, et al., "Software-defined DDoS detection with information entropy analysis and optimized deep learning," Future Gener. Comput. Syst., vol. 129, pp. 99–114, 2022.

[15] L. F. Eliyan and R. Di Pietro, "DeMi: A solution to detect and mitigate DoS attacks in SDN," IEEE Access, vol. 11, pp. 82477–82495, 2023.

[16] A. I. Hassan, E. A. El Reheem, and S. K. Guirguis, "An entropy and machine learning based approach for DDoS attacks detection in software defined networks," Sci. Rep., vol. 14, no. 1, Art. no. 18159, 2024.

[17] M. Yue, H. Yan, R. Han, et al., "A DDoS attack detection method based on IQR and DFFCNN in SDN," J. Netw. Comput. Appl., vol. 240, Art. no. 104203, 2025.

[18] B. Pfaff, J. Pettit, T. Koponen, et al., "The design and implementation of Open vSwitch," in Proc. 12th USENIX Symp. Networked Systems Design and Implementation (NSDI 15), 2015, pp. 117–130.

[19] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in Proc. 9th ACM SIGCOMM Workshop on Hot Topics in Networks, 2010, pp. 1–6.

[20] K. Rui, F. Tomonori, A. Yuji, et al., "Ryu SDN framework—Open-source SDN platform software," NTT Tech. Rev., vol. 12, no. 8, pp. 18–22, 2014.