Balancing Timeliness and Accuracy: A Hybrid Data-Control Plane Framework for Volumetric DDoS Defense in IoT

Jiahang Pu, Hongyu Ye, Jing Cheng, Feng Shan, *Member, IEEE*, and Runqun Xiong, *Member, IEEE*

***Abstract*—Resource-constrained IoT devices in Industrial Internet environments are highly vulnerable to DDoS attacks due to infrequent security updates and insufficient built-in protection mechanisms. Existing defense solutions primarily rely on external filtering servers or programmable switches, but these approaches fail to simultaneously meet the stringent real-time performance and high accuracy requirements of industrial applications. To address these limitations, we propose a novel cross-plane defense framework that exploits the temporal invariance characteristics of attack traffic patterns. In the data plane, an adaptive variance threshold mechanism immediately mitigates high-volume, low-variance traffic flows, while a bidirectional dual-hash table captures low-collision flow features for efficient export to the control plane. The control plane constructs temporally-enhanced flow sequences that enable deep learning models to perform accurate attack detection, subsequently directing the data plane to block identified malicious sources. We implemented and evaluated a prototype of this framework on a software switch platform using both real-world attack datasets and custom-generated traffic patterns. Experimental results demonstrate that our framework successfully mitigates 86% of attack traffic within milliseconds and achieves complete source blocking within 52 seconds. Compared to baseline methods, our framework can effectively counter both DoS and DDoS attacks without generating false positives on benign traffic.**

***Index Terms*—Distributed denial-of-service attack, attack detection, attack defense, P4, deep learning.**

I. Introduction

THE rapid development of the Industrial Internet has led to the widespread adoption of IoT devices in industrial settings. These devices act as a critical bridge between traditionally closed Operational Technology (OT) networks and open Information Technology (IT) networks, enabling applications like smart manufacturing, remote monitoring, and fine-grained management. The number of connected IoT devices reached 16.6 billion by the end of 2023, a 15% increase from 2022 [1]. However, their inherent characteristics—low performance, infrequent updates, and lack of self-protection measures—make them the weakest link in Industrial Internet security and a primary target for attackers. Distributed Denial of Service (DDoS) attacks represent the most significant network threat in this domain [2]. The low attack tolerance of these devices, combined with the high stability requirements of production control, imposes strict demands on the real-time performance and accuracy of any defense system.

To achieve high detection accuracy, many works employ server-based machine learning approaches, training models on large-scale datasets [3], [4], [5], [6], [7], [8], [9], [10]. While these methods can leverage prior knowledge, they are detached from the packet processing pipeline. The process of packet uploading for feature extraction and model inference introduces significant latency, rendering them incapable of meeting the real-time demands of the Industrial Internet, where the resulting delay is sufficient to paralyze resource-constrained IoT devices.

To meet the high-speed, low-latency processing requirements, many recent studies have utilized programmable switches to build defense systems [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21]. These devices support the P4 language, enabling efficient, line-rate packet processing and making them well-suited for high-speed, low-power traffic analysis and defense. However, a key limitation is their reliance on small on-chip memories (tens of MBs), which restricts the complexity of real-time data processing. Existing P4-based works have focused on optimizing detection mechanisms [15], [16] or embedding simplified ML models [17], [18], [19], [20], [21]. The former often require tens of seconds to analyze traffic distribution patterns, failing to provide a timely response to bursty attacks, while the latter must compress models to fit into Match-Action tables, thereby sacrificing accuracy.

Furthermore, the Industrial Internet is highly heterogeneous, utilizing diverse traffic forms and specialized protocols. While current AI-driven defenses achieve high accuracy in detecting

Received 7 December 2025; revised 26 March 2026; accepted 10 May 2026. Date of publication 14 May 2026; date of current version 20 May 2026. This work is supported by the National Key R&D Program of China with Grant number 2021YFB2900100; National Natural Science Foundation of China under grants 62572115 and 62232004; Shenzhen's Industrial Application Research on "Key Technologies and Systems for Intelligent Cloud-Edge Collaboration in the Industrial Internet" (Grant No. CJGJZD20230724092759006) funded by Shenzhen Municipal Science and Technology Innovation Commission; Jiangsu Provincial Key Laboratory of Network and Information Security under grant BM2003201; Key Laboratory of Computer Network and Information Integration of the Ministry of Education of China under grant 93K-9; and the Collaborative Innovation Center of Novel Software Technology and Industrialization. The associate editor coordinating the review of this article and approving it for publication was E. E. Tsiropoulou. *(Corresponding author: Runqun Xiong.)*

The authors are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: j.pu@seu.edu.cn; hy.ye@seu.edu.cn; chengjing@seu.edu.cn; shanfeng@seu.edu.cn; rxiong@seu.edu.cn).

Digital Object Identifier 10.1109/TNSM.2026.3693266

threats within specific protocols [22], deploying these models across the entire industrial protocol stack is computationally prohibitive. Such protocol-specific approaches fail to provide comprehensive coverage and cannot meet the strict low-latency constraints of resource-constrained IoT devices.

To address these challenges, our work is based on a key insight derived from analyzing public DDoS and benign traffic datasets: attack and benign traffic exhibit a high degree of discrimination in both the magnitude and temporal patterns of their packet-level and flow-level features. While previous work often used simple data plane thresholds or sampled statistics, we posit that leveraging the high-frequency, low-variance temporal patterns of attack traffic can reduce initial detection latency without compromising accuracy. Furthermore, using sequences of time-windowed statistics provides a more comprehensive foundation for detection models.

Based on this insight, we propose a cross-plane collaborative framework that combines real-time primary defense and feature extraction in the data plane with flow sequence detection in the control plane. Specifically, our framework utilizes three modules to address the following challenges:

- **Lightweight Primary Defense:** To provide a timely response to high-volume attacks and mitigate their impact, we propose a lightweight primary defense mechanism. By tracking traffic frequency and feature variance, it immediately rate-limits high-speed, low-variance flows, achieving real-time mitigation with minimal overhead.
- **Low-Collision Feature Extraction:** To ensure complete statistical records for suspicious traffic, we design a low-collision feature extraction mechanism. It uses a dual-hash table to maintain statistics for bidirectional flows and employs a time-windowed storage and collection policy to guarantee low-collision recording.
- **Temporal Sequence-Based Detection:** To guarantee detection accuracy on attack sources, we employ a CNN-GRU-Attention model to analyze the temporal evolution of traffic data using constructed temporal flow sequences. It accurately identifies attack sources by detecting the high-volume, low-variance characteristics of attack flows.

We implemented a software switch-based prototype of the framework and evaluated its performance using real-world attack traffic and custom attack scripts. The results show that the primary defense stage detects over 99% of attack packets, reducing attack traffic by approximately 86% within 25ms. Subsequently, the temporal model, with a PR-AUC exceeding 99%, detects and completely blocks the attack sources within approximately 50 seconds. Compared to baseline methods, our framework effectively covers both DoS and DDoS attacks while ensuring that benign traffic is not falsely classified.

II. Background and Motivation

A. Problem Scope

This paper focuses on DDoS network attacks targeting IoT devices. The network topology may include multiple IoT devices, such as cameras and sensors, that send packets to a controller or gateway either periodically or based on specific trigger conditions. These devices often have simple functions, resulting in limited software, hardware, and resource capabilities, which gives them a low tolerance for high-volume attacks. The goal for these victim devices is to prevent service disruption and subsequent losses from such attacks. The defense posture assumes that all traffic within the topology passes through a device where the defense framework is deployed, and that both the victim and the framework have no prior knowledge of the attack's source, type, or scale.

TABLE I
Analyzed Traffic Dataset

Dataset	Traffic Type	#Source IP	#Total
Booter-1 [23]	DNS DDoS	4486	131220
CIC-IDS-2017 [24]	Benign	9709	529918

Attackers are assumed to have access to the network topology containing these IoT devices. They can leverage a controlled botnet to send high-speed, high-volume DDoS traffic aimed at exhausting the victim's bandwidth or computational resources. The attacker may employ various DDoS attacks, including UDP, DNS, and NTP floods, and can use multiple hosts to send a massive number of packets to a single target.

B. Key Observation

1) Feature Selection: To identify the most effective features for distinguishing between normal and attack traffic, and to better align with the multi-source nature of DDoS attacks, datasets with a large number of source IPs—a real-world DNS attack dataset [23] and a simulated benign background traffic dataset [24]—were used to analyze feature distribution and temporal variation. The selected datasets are shown in Table I.

Since the framework focuses on a binary classification problem with multiple continuous features, Analysis of Variance (ANOVA) is used to select highly discriminating features by checking for significant differences between the means of normal and attack traffic. To visually represent the feature differences, the test set was formatted for binary classification, and ANOVA was used to select top 10 key features for simplified analysis. These features were then standardized using Eq. 1, where $\bar{x}$ is the sample mean, S is the sample standard deviation, and z is the standardized value.

$$z = \frac{x - \bar{x}}{S} \tag{1}$$

As depicted in Figure 1, there are clear differences in the network characteristics of normal and DDoS traffic. Features such as packet transmission rate, byte rate, packet length, and packet interval are particularly suitable for distinguishing malicious traffic. This analysis provides a guiding reference for designing the data plane thresholds, selecting the types of features to extract, and designing the temporal model.

2) Magnitude Difference: An analysis of the two datasets was conducted by selecting three features: flow bytes per second, flow packets per second, and average packet size, shown in Figure 2.

At a macroscopic level, attack traffic is characterized by large and high-speed packet transmissions, as the attacker's

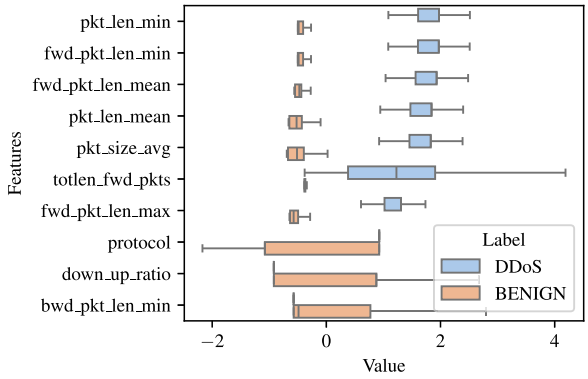
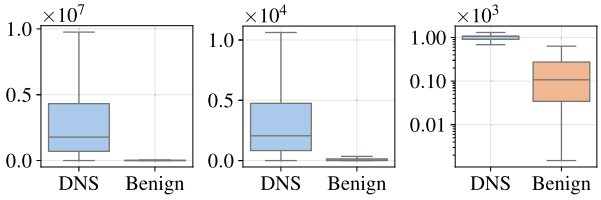


Fig. 1. Standardized values of selected features.



(a) Flow Byte/s (b) Flow Packet/s (c) Avg. Packet Size

Fig. 2. Statistical feature distribution of DNS DDoS and benign traffic.

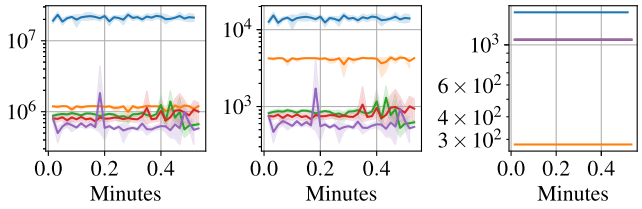
TABLE II
MEDIAN VALUE OF SELECTED FEATURES

Feature	DNS DDoS	Benign
Flow Byte/s	1611782.41	1926.15
Flow Packet/s	1821.18	32.41
Forward Packets	15	3
Forward Length	16020	96
Avg. Packet Size	1068	101

goal is to rapidly saturate network bandwidth and server resources. At a microscopic level, the median values for flow bytes per second, flow packets per second, and average packet size differ significantly. As Table II shows, there is a gap of more than two orders of magnitude between attack and normal traffic for these features, allowing for clear differentiation. This suggests that a lightweight thresholding algorithm can be implemented in the data plane to identify malicious sources based on flow rate and packet size distribution, enabling an initial rate-limiting defense.

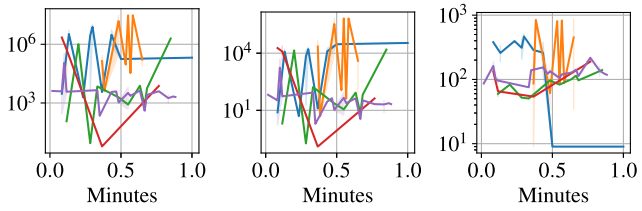
3) Temporal Difference: To observe how traffic characteristics change over time, the five most active flows from each dataset were analyzed. The results show significant differences in the magnitude, distribution, and duration of statistical data between attack and normal traffic, which primarily stem from their respective packet generation processes.

a) Attack traffic: The packet rate and size of attack traffic show little variation and are sent continuously, as shown in Figure 3. This uniformity arises because DDoS attacks are typically launched via scripts that control multiple hosts with predefined parameters (e.g., protocol, packet size), resulting in traffic with stable and homogenous statistical features.



(a) Flow Byte/s (b) Flow Packet/s (c) Avg. Packet Size

Fig. 3. Time series changes of DNS DDoS traffic features.



(a) Flow Byte/s (b) Flow Packet/s (c) Avg. Packet Size

Fig. 4. Time series changes of benign traffic features.

b) Normal traffic: In contrast, normal traffic from diverse sources like sensors and user interactions results in packets with varying sizes and irregular transmission patterns, as shown in Figure 4. This leads to statistics with significantly higher variance and more dynamic characteristics compared to the continuous, uniform nature of attack traffic.

Traditional lightweight machine learning models, such as Multi-Layer Perceptrons (MLPs) or Random Forests (RFs), evaluate traffic features within isolated time windows. This static approach struggles in IoT environments, where transient benign bursts (e.g., event-triggered sensor data) closely resemble attack traffic within a single time slice. Our temporal analysis reveals that DDoS attacks are characterized not just by high volume, but by temporal invariance, with continuous, low-volatility transmission rates over extended periods. Because simple models lack temporal memory to capture these sequential dependencies, extracting feature sequences across multiple time windows and employing a temporal-aware model is essential to accurately distinguish sustained attacks from benign bursts and minimize false positives.

III. OVERVIEW

A. Design

The main contributions of the framework are divided into three parts, which perform primary rate limiting, flow feature extraction, and temporal model detection.

1) Adaptive Lightweight Primary Defense: This module uses the EWMA-based method to adaptively adjust detection thresholds based on normal traffic conditions, enabling low-overhead differentiation of malicious traffic. It employs a sketch-based data structure to record suspicious sources with minimal storage, which triggers flow-level feature extraction. A meter mechanism actively limits packet forwarding rates for a rapid response that reduces the impact on benign traffic.

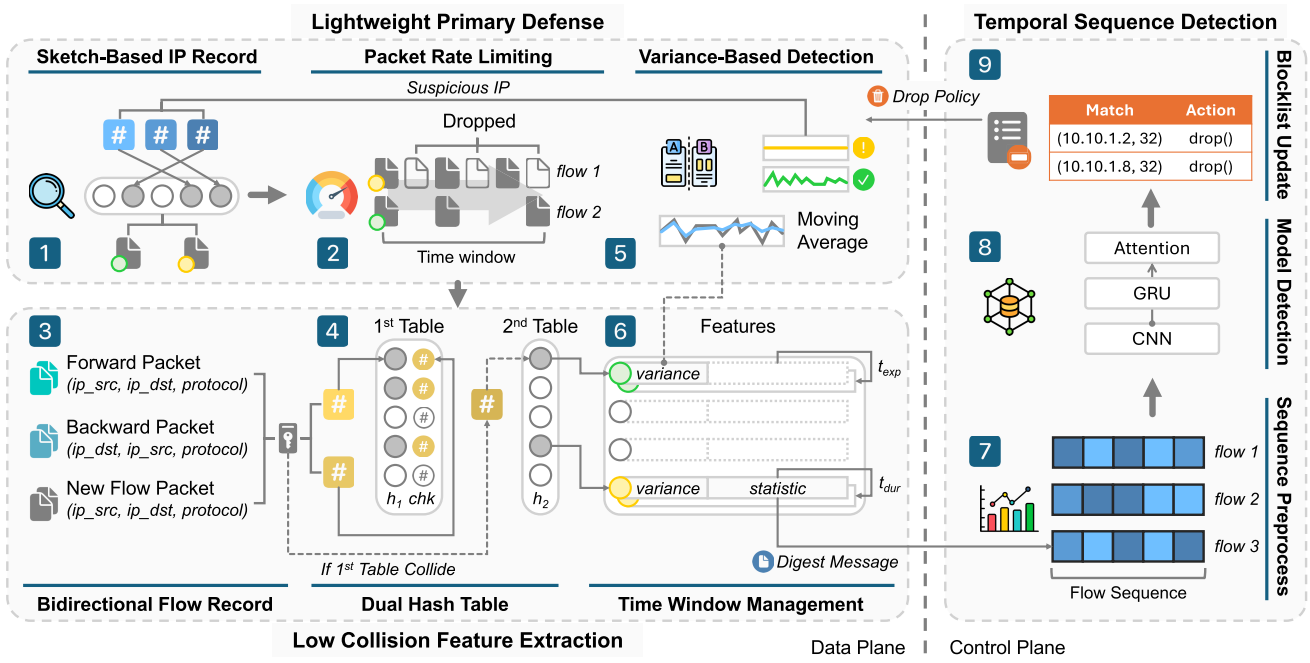


Fig. 5. Overview of proposed framework.

2) Low-Collision Bidirectional Feature Extraction: To reduce the number of required register entries, this module maps bidirectional traffic from a given host-protocol pair to a single hash value. Its dual hash table design automatically engages a secondary table when a collision occurs in the primary table, increasing the capacity for concurrent flow tracking. A time-window-based mechanism automatically exports statistical data and clears entries, maximizing the efficient use of register space by minimizing memory footprint.

3) High-Accuracy Temporal Model Detection: In the data preprocessing stage, enhanced temporal features are computed to capture the dynamic changes in data sequences. A CNN-GRU-Attention model effectively identifies attacks by recognizing the temporal invariance characteristic of malicious traffic. Using the P4Runtime API, the control plane receives flow feature data via digests, updates blocklist entries, and supports integration with existing feature extraction frameworks.

B. Workflow

The framework's workflow is depicted in Figure 5.

1) Data Plane: The data plane supports two modes, which can be modified in real-time by the control plane:

Pre-detection Mode. All packets passing through the switch are forwarded without additional feature extraction or data recording. In this mode, the data plane adaptively updates the detection threshold used by the primary defense mechanism based on the observed benign traffic.

Detection Mode. Incoming packets are first checked against a blocklist, which is updated in real-time by the control plane's detection model. Packets from sources not on the blocklist are processed by a sketch to determine if the source has been previously marked as suspicious.

For non-suspicious packets, their corresponding lightweight statistics are updated in real-time. If the flow's variance exceeds the primary defense threshold, its source IP is used to update the sketch. For suspicious packets, they are immediately rate-limited. The full bidirectional feature extraction process is enabled for these flows. At the end of each time window, the collected statistics are sent to the control plane, and the corresponding data plane entries are reset.

2) Control Plane: The control plane uses a streaming channel to receive statistical data for individual flows in real-time, maintaining a data sequence of a fixed window size for each flow. Once a sequence window is full, the control plane computes enhanced temporal features and feeds the sequence into the CNN-GRU-Attention model for detection. The source IPs of flows identified as malicious are then added to the data plane's blocklist, completing the closed-loop process for attack detection and mitigation.

IV. Design Detail

A. Lightweight Primary Defense

The primary defense process is executed entirely within the data plane, leveraging its high processing rate while circumventing the limitations of the P4 programming language and hardware resources. Specifically, the module implements lightweight attack traffic identification, suspicious IP recording, and traffic rate limiting to achieve a timely and effective response during the initial stages of an attack with minimal computational and storage overhead.

1) Variance-Based Detection: The analysis in the previous sections, which showed an order-of-magnitude difference in packet size between normal and DDoS traffic, provides a basis for classification. However, there are challenges in both

the statistical principles for correctly distinguishing traffic and their implementation in the data plane.

At the principle level, some existing work continuously collects packet sizes and rates within a time window in the data plane, marking traffic as an attack when a predefined threshold is reached. However, this approach has two problems. First, manually set thresholds require domain knowledge and make it difficult to distinguish between different traffic types. Second, even among normal traffic, characteristics can vary numerically depending on the operating scenario and device type. For instance, some IoT sensors transmit data periodically, resulting in traffic with similarly sized packets, while other event-triggered IoT devices or user interactions produce intermittent traffic with varied packet sizes.

At the implementation level, common methods for distinguishing malicious traffic are based on transmission size and rate, calculated per second. While these operations are simple to implement in a control plane language like Python, the P4 data plane has significant operational constraints. For example, calculating an average transmission rate by summing bytes and dividing by a timestamp is not feasible, as P4 specifications prohibit division and floating-point operations. Even converting calculations to multiplication and comparison is problematic due to limitations on the bit-width of operands, which can lead to data overflow.

To address these issues, this framework uses variance instead of a simple size threshold to improve generality, and it uses the Exponentially Weighted Moving Average (EWMA) to avoid direct average calculations, lowering implementation requirements for the data plane. EWMA assigns greater weight to recent observations, better reflecting current trends without needing to store past values, thus reducing the computational load to meet data plane constraints.

The framework maintains the variance of the packet size EWMA and a count of threshold triggers to judge traffic type. Because the P4 data plane prohibits floating-point operations, EWMA and variance calculations must be approximated using integer bit-shift operations. Quantifying the inevitable quantization error from this precision loss is crucial. The ideal floating-point pipeline is formulated as:

$$\mu_n = (1-\alpha)\mu_{n-1} + \alpha x_n, \tag{2}$$

$$\sigma_n = (1-\alpha)\sigma_{n-1} + \alpha x_n^2, \tag{3}$$

$$Var_n = \sigma_n - \mu_n^2 \tag{4}$$

where μ_n and σ_n are the EWMAs of the packet size and its square, respectively, x_n is the current packet size, Var_n is the true variance, and $\alpha = 2^{-s}$ (with $s = 3$) is the weight parameter. The corresponding fixed-point bit-shift pipeline, employing the floor function, is:

$$\hat{\mu}_n = \hat{\mu}_{n-1} + \left\lfloor \frac{x_n - \hat{\mu}_{n-1}}{2^s} \right\rfloor, \tag{5}$$

$$\hat{\sigma}_n = \hat{\sigma}_{n-1} + \left\lfloor \frac{x_n^2 - \hat{\sigma}_{n-1}}{2^s} \right\rfloor, \tag{6}$$

$$\hat{Var}_n = \hat{\sigma}_n - \hat{\mu}_n^2. \tag{7}$$

Integer truncation during right-shifts discards the remainder, introducing a truncation error $\epsilon \in [0, 7/8)$ per update. The maximum accumulated absolute error is bounded by the geometric series:

$$|E_{total}| \le \sum_{k=0}^{\infty} \left(\epsilon_{max} \left(\frac{7}{8} \right)^k \right) = \frac{\epsilon_{max}}{1 - 7/8} = 7. \tag{8}$$

Thus, the steady-state maximum absolute error for both $\hat{\mu}_n$ and $\hat{\sigma}_n$ is 7 (i.e., $|E_\mu| \le 7$ and $|E_\sigma| \le 7$). Substituting these into the variance error formula yields:

$$\begin{aligned} E_{var} &= Var_n - \hat{Var}_n \\ &= (\hat{\sigma}_n + E_\sigma) - (\hat{\mu}_n + E_\mu)^2 - (\hat{\sigma}_n - \hat{\mu}_n^2) \\ &= \hat{\sigma}_n + E_\sigma - \hat{\mu}_n^2 - 2\hat{\mu}_n E_\mu - E_\mu^2 - \hat{\sigma}_n + \hat{\mu}_n^2 \\ &= E_\sigma - 2\hat{\mu}_n E_\mu - E_\mu^2. \end{aligned} \tag{9}$$

Consequently, the maximum absolute error of the variance is bounded by:

$$|E_{var}| \le 7 + 14|\hat{\mu}_n| + 49 = 14|\hat{\mu}_n| + 56. \tag{10}$$

This derivation indicates the variance error is linearly correlated with the packet length EWMA ($\hat{\mu}_n$). Our evaluations confirm that for $\alpha = 2^{-3}$, the relative variance error is merely 1.37%, which is entirely sufficient to support accurate threshold-based anomaly detection without causing misclassifications.

Algorithm 1 EWMA-Based Suspicious IP Detection

Require: Packet length L, flow hash h, variance threshold θ
Ensure: Suspicious IP marking decision
Phase 1: EWMA update
$\mu_{prev}, \sigma^2_{prev} \leftarrow ReadEWMA(h)$
$\Delta \leftarrow (L - \mu_{prev}) \gg 3$ $\{\alpha = 0.125\}$
$\mu_{next} \leftarrow \mu_{prev} + \Delta$
$\Delta^2 \leftarrow (L^2 - \sigma^2_{prev}) \gg 3$
$\sigma^2_{next} \leftarrow \sigma^2_{prev} + \Delta^2$
$WriteEWMA(h, \mu_{next}, \sigma^2_{next})$
Phase 2: Variance calculation
$Var_{flow} \leftarrow \sigma^2_{next} - \mu^2_{next}$
$\mu_{benign}, \sigma^2_{benign} \leftarrow ReadBenignEWMA()$
$Var_{benign} \leftarrow \sigma^2_{benign} - \mu^2_{benign}$
Phase 3: Anomaly counting
$count \leftarrow ReadExceedCount(h)$ {Low variance indicates attack}
if $Var_{flow} < Var_{benign}$ **then**
  $count \leftarrow count + 1$
  $WriteExceedCount(h, count)$
end if
Phase 4: Suspicious marking
if $count > \theta$ **then**
  $MarkSuspicious(IP)$
  $ResetEWMAStatistics(h)$
end if

In pre-detection mode, the framework forwards normal traffic and updates the normal traffic variance in real-time, requiring no manual configuration. In detection mode, the detection process is shown in Algorithm 1. Since P4 does not support floating-point arithmetic, the multiplication by α is

implemented as a bit-shift operation. The calculated variance is compared to the normal variance. If it is lower, a trigger counter is incremented. If this counter exceeds a threshold within a time window, the source IP is deemed suspicious, activating rate limiting and full feature extraction. Experiments confirmed that using an EWMA-based variance and a hit-count threshold effectively filters malicious packets while avoiding false positives on benign traffic.

2) Sketch-Based IP Recording: IPs whose traffic exceeds the variance threshold are considered suspicious and must be recorded for mitigation. A naive approach of using a large hash table in the data plane is infeasible due to memory constraints, while recording in the control plane would incur significant communication overhead, a problem exacerbated by the multi-source nature of DDoS attacks.

To resolve this, the framework implements a Bloom filter as low-overhead method to track suspicious IPs. A Bloom filter is a probabilistic data structure that efficiently tests for set membership. Our implementation uses a 1-bit register array of length 2^{18} and three different hash functions. When a flow's variance triggers the threshold, its source IP is hashed three times, and the bits at the three corresponding positions in the array are set to 1. When a new packet arrives, it is checked against the filter by hashing its source IP and performing an AND operation on the bits at the three positions. A result of 1 indicates the IP was previously marked as suspicious, triggering rate limiting and feature extraction.

3) Proactive Rate Limiting: Packets from suspicious sources are immediately rate-limited to reduce their impact on the target's bandwidth and services, using the P4 meter mechanism. This mechanism is based on a Single Rate Three Color Marker (srTCM), which defines a Committed Information Rate (CIR) and a Burst Size to mark packets according to their conformance to traffic profiles.

The framework uses a filtering table in conjunction with the meter's color markings to enforce rate limits. When a packet from a suspicious IP arrives (i.e., it hits the Bloom filter), a hash function maps it to one of 2^{15} meter instances. This meter updates a rate marker based on preset CIR and Burst Size values configured from the control plane. The filtering table then inspects this marker to determine if the packet exceeds the rate limit. If so, the packet is dropped, thus maintaining the high-volume traffic in a low-speed forwarding state.

B. Low-Collision Feature Extraction

The entire feature extraction process is handled within the data plane to capture the characteristics of attack traffic while avoiding storage-location conflicts. Specifically, this module implements bidirectional flow recording, statistical feature extraction, and a dual hash table mapping scheme. This low-collision design allows for the storage of diverse statistical data in a manner that is compatible with both data plane constraints and the requirements of the temporal detection model.

1) Bidirectional Flow Recording: Traffic between a pair of hosts is inherently bidirectional. This framework records the features for both directions of a host pair in a single, unified entry. This approach reduces the number of required hash values, which in turn lowers the hash table load and minimizes collision probability. This design is based on the key insight that DDoS traffic exhibits a significant order-of-magnitude difference between the statistics of its two directions—attackers send a high volume of traffic to victims who typically have not previously communicated with them. In contrast, this statistical gap is less pronounced in normal traffic. This asymmetry provides a strong data foundation for the subsequent model to leverage for detection.

Algorithm 2 Bidirectional Flow Hash Calculation With Collision Detection

Require: Packet with src_ip, dst_ip, $protocol$
Ensure: Flow hash h and direction flag dir {Assume forward direction}
$dir \leftarrow FORWARD$
$h, chk \leftarrow Hash(src\_ip, dst\_ip, protocol)$
$cnt \leftarrow ReadCount(h)$
{No flow found, try reverse}
if $cnt = 0$ **then**
  $dir \leftarrow BACKWARD$
  $h, chk \leftarrow Hash(dst\_ip, src\_ip, protocol)$
  $cnt \leftarrow ReadCount(h)$
end if
{Flow exists, check collision}
if $cnt > 0$ **then**
  $stored\_chk \leftarrow ReadStoredCheck(h)$
  if $stored\_chk \neq chk$ **then**
    $use\_second\_table \leftarrow TRUE$
    Resubmit packet for second hash table
    return
  end if
end if{Initialize new flow}
if $cnt = 0$ **then**
  $dir \leftarrow FORWARD$
  $h, chk \leftarrow Hash(src\_ip, dst\_ip, protocol)$
  $InitializeFlow(h)$
  $StoreCheck(h, chk)$
end if
return h, dir

The first step is to designate one direction as "forward". A simple method like comparing IP addresses is insufficient. Instead, we leverage the observation that a benign server will not initiate a connection to a previously unknown malicious host, but not vice versa. Therefore, the first packet in a DDoS attack will reliably be sent from the malicious host to the benign server. Based on this, the framework assumes any new flow is initiated in the "forward" direction. As detailed in Algorithm 2, it checks for a record in both the forward and reverse directions. If no record exists, one is initialized. After a record is found, it must be checked for a hash collision, a process handled by the dual hash table mechanism.

2) Dual Hash Table: DDoS attacks are characterized by a large number of source IPs, which can lead to hash collisions when storing flow features. To balance the trade-off between collision rates and memory overhead, the module employs a dual hash table solution instead of a single, larger table. This design splits a hash table of a given width into two tables of

TABLE III
EXTRACTED FEATURES FOR FLOWS

	Feature	Description
Bidirectional	Packets	Number of packets recorded in flow.
	Bytes	Number of bytes in packets recorded in flow.
	First packet timestamp	Arrival time of the first packet.
	Last packet timestamp	Arrival time of the last packet.
Forward	Packets	Number of packets under src. IP.
	Bytes (min/max/sum.)	Minimum, maximum, and sum of bytes in packets under src. IP.
Backward	Packets	Number of packets under dst. IP.
	Bytes (sum.)	Sum of packet bytes under dst. IP.
	IAT	Sum of inter-arrival times of packets under dst. IP.

half that width; if a collision occurs in the primary table, the backup table is used.

To detect collisions, a secondary hash function computes a checking value for each new flow, which is stored alongside the flow's primary record. When a subsequent packet maps to that same location, its checking value is compared with the stored one. A mismatch indicates a collision. The packet is then marked and re-injected into the processing pipeline via the resubmit mechanism, which transfers the new flow's record to the backup hash table. Experiments show that when storing 10,000 flows, two 2^{15} sized hash tables yield a collision rate that is only 18.86% of that from a single 2^{16} sized table.

3) Time Window Feature Extraction: The calculated hash value maps to the same index across multiple register arrays, which store the various features extracted for a single flow. Based on the earlier analysis, the features with the highest discriminative power are selected as input for the temporal model, as shown in Table III.

To manage memory, flow statistics are maintained within a time window. If the inter-arrival packet time or total flow duration exceeds their corresponding threshold, the data plane entry is cleared, and its statistics are uploaded to the control plane via a digest message. The policy for this process differs based on the flow's status, as shown in Algorithm 3.

Non-suspicious flows use a shorter time window to periodically update their variance data. Suspicious flows (those flagged by the primary defense) use a longer time window to collect more detailed, flow-level statistics for the control plane. Notably, if the packet triggering an update belongs to the reverse flow, the 5-tuple in the digest is swapped to ensure that attack traffic is always represented in the "forward" direction for correct processing by the detection model.

C. Temporal Flow Sequence Detection

The flow detection process is executed entirely in the control plane. It uses the statistical data uploaded from the data plane to form traffic sequences, which are then fed to a detection model to identify and block packets from malicious IP sources. This module implements temporal data preprocessing and a temporal detection model to accurately locate attack IPs by leveraging the temporal invariance of their traffic features.

1) Temporal Data Preprocessing: The control plane establishes a P4Runtime streaming channel to continuously receive and parse digest messages into 5-tuple information and bidirectional statistics. These messages are processed via a FIFO queue and grouped by source IP, destination IP, and protocol into a double-ended queue of length L. Once a flow reaches this sequence length, its temporal features are preprocessed and fed into the detection model. To ensure temporal relevance, flow records are removed after a timeout. Due to the operational constraints of the data plane, certain statistics (e.g., flow bytes per second) are calculated in the control plane using the raw sums and timestamps provided by the data plane.

To better capture the dynamics of traffic sequences, the framework computes several enhanced time-series features:

Difference Features. Eq. 11 calculates the first-order difference of the byte rate and packet rate time series. This feature captures the instantaneous rate of change between consecutive time steps, highlighting sudden accelerations or decelerations in traffic volume. In the equation, ΔB_t and ΔP_t are the resulting difference features for the byte and packet rates at the current time step t, respectively, calculated from the rates at the current step (B_t, P_t) and the previous step (B_{t-1}, P_{t-1}).

$$\Delta B_t = B_t - B_{t-1} \quad \text{and} \quad \Delta P_t = P_t - P_{t-1} \tag{11}$$

Moving Average Features. To reveal the underlying trend of a traffic flow, Eq. 12 computes a 3-point moving average to smooth the time series and reduce the impact of random noise. By averaging over a short window, this feature dampens insignificant, short-term fluctuations. Here, $MA_3(X_t)$ is the moving average of a given time-series feature X at time step t, calculated by averaging the feature's value over the last three

Algorithm 3 Flow Time Window Management and Collection Policy

Require: Flow metadata $meta$, timestamps $t_{current}$, t_{last}, t_{start}
Ensure: Flow cleanup or data collection decisions
$\Delta t_{idle} \leftarrow t_{current} - t_{last}$
$\Delta t_{duration} \leftarrow t_{current} - t_{start}$ {First stage cleanup policy}
if $suspicious = FALSE$ **then**
  if $\Delta t_{idle} > T_{expired}$ **or** $\Delta t_{duration} > T_{ewma}$ **then**
    $ResetFlowStatistics()$
    $ClearHashCheck()$
    return
  end if
end if{Suspicious flow collection policy}
if $suspicious = TRUE$ **then**
  if $\Delta t_{idle} > T_{expired}$ **or** $\Delta t_{duration} > T_{max}$ **then**
    if $direction = BACKWARD$ **then**
      $SwitchFlowDirection()$
    end if
    $SendFlowDigest()$
    $ResetFlowStatistics()$
    $ClearHashCheck()$
    return
  end if
end if

time steps ($i = 0, 1, 2$).

$$MA_3(X_t) = \frac{1}{3}\sum_{i=0}^{2} X_{t-i} \tag{12}$$

Ratio Features. The ratio feature in Eq. 13 is calculated to reflect traffic density by determining the average bytes per packet for a given time step. In the equation, R_t represents the ratio feature at time step t, calculated from the byte rate B_t and packet rate P_t for that step. A small smoothing factor $\epsilon = 10^{-10}$ is added to the denominator to prevent division by zero in cases where the packet rate is momentarily zero.

$$R_t = \frac{B_t}{P_t + \epsilon} \tag{13}$$

These methods effectively capture the dynamic characteristics of network traffic, including instantaneous changes, smoothed trends, and density.

2) Model Design: While intuitively fast, using simple static models (e.g., MLP or RF) on single traffic time slices risks generating false positives on transient bursty benign traffic. Conversely, applying simple models across multiple time slices with trigger thresholds fails to leverage the spatio-temporal invariance of attacks and offers negligible latency advantages. To address this, we employ a CNN-GRU-Attention architecture to detect sequences of traffic time slices. Unlike lightweight models, it simultaneously captures spatial patterns and long-term temporal dependencies, effectively eliminating false positives on benign IoT traffic. The marginal increase in inference complexity is heavily outweighed by the critical gains in detection accuracy and overall system reliability.

The core architecture fuses a 1D Convolutional Neural Network (CNN), a Gated Recurrent Unit (GRU), and an attention mechanism. This three-stage design achieves efficient detection by capturing complex spatio-temporal patterns and key discriminative information from the traffic data. The model input is a sequence of traffic statistics from multiple time windows, represented as $X \in \mathbb{R}^{B\times L\times F}$, where $B = 64$ is the batch size, $L = 5$ is the sequence length (time steps), and $F = 21$ is the feature dimension.

Local Pattern Extraction. A two-layer 1D CNN extracts local patterns and temporal features. The input X is processed through two blocks of Conv1D, BatchNorm, and GELU activation. This process is formally described in Eq. 14 and 15, where Conv1D_k denotes a 1D convolution with k output channels, a kernel size of 3, and a stride of 1.

$$H^{(1)} = \text{GELU}(\text{BatchNorm}(\text{Conv1D}_{128}(X))) \tag{14}$$

$$H^{(2)} = \text{GELU}(\text{BatchNorm}(\text{Conv1D}_{256}(H^{(1)}))) \tag{15}$$

A residual connection is included to preserve original feature information and mitigate potential vanishing gradients. A 1×1 convolution W_{res} maps the input features to the output dimension of the second convolutional layer for summation. The final output of the CNN module, H_{final}, is computed by adding the output of the second convolutional layer to the transformed input, as shown in Eq. 16.

$$H_{final} = H^{(2)} + W_{res} \cdot X \tag{16}$$

Temporal Dependency Modeling. To capture long-term dependencies, the output from the CNN module, H_{final}, is fed into a 2-layer stacked bidirectional GRU. The BiGRU's gates effectively control information flow, allowing it to model long-range temporal relationships. The state update for the BiGRU at each time step t is represented as:

$$h_t = \text{BiGRU}(h_{t-1}, H_{final}[:, t]). \tag{17}$$

The term h_{t-1} is the hidden state from the previous time step, and $H_{final}[:, t]$ is the feature vector from the CNN module for the current time step. The GRU module produces the final sequence of hidden states $H_{gru} \in \mathbb{R}^{L\times B\times 256}$.

Attention Focusing. A multi-head self-attention mechanism is applied to the GRU outputs to enhance the model's focus on key discriminative features. As shown in Eq. 18, the attention scores are calculated using scaled dot-product attention.

$$\begin{aligned} A &= \text{MultiHeadAttention}(Q, K, V) \\ &= \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \end{aligned} \tag{18}$$

The query (Q), key (K), and value (V) matrices are derived from H_{gru}. he model uses 4 attention heads, and the scaling factor d_k is the dimension of the key vector.

Classification Decision. The output of the self-attention layer, A, is aggregated into a single context vector c using an attention-based weighted pooling mechanism. First, attention weights α are computed for each time step's output using a small feed-forward network, as shown in Eq. 19.

$$\alpha = \text{softmax}(W_2 \cdot \tanh(W_1 \cdot A)) \tag{19}$$

These weights, which reflect the importance of each time step, are then used to compute the final context vector c as a weighted sum of the attention outputs, as described in Eq. 20.

$$c = \sum_{i=1}^{L} \alpha_i A_i \tag{20}$$

This context vector c is then passed to a 3-layer MLP classification head, $f_c : \mathbb{R}^{256} \to \mathbb{R}^1$, which outputs the final prediction.

3) Optimization Strategies: Given the imbalanced nature of the dataset, a weighted Binary Cross-Entropy loss function (BCEWithLogitsLoss) is used. The loss function is formally defined in Eq. 21.

$$\begin{aligned} L_{BCE} = -\frac{1}{B}\sum_{j=1}^{B}[&w \cdot y_j \cdot \log(\sigma(z_j)) \\ &+ (1 - y_j) \cdot \log(1 - \sigma(z_j))] \end{aligned} \tag{21}$$

In this equation, $y_j \in \{0, 1\}$ is the true label, z_j is the model's raw output (logit) for sample j, $\sigma(\cdot)$ is the sigmoid function, and w is a weight applied to the positive class, calculated as the ratio of negative to positive samples N_{neg}/N_{pos} to counteract the imbalance.

The OneCycleLR policy is employed to dynamically adjust the learning rate lr_t during training, which is calculated according to the schedule in Eq. 22 at any given epoch t.

$$lr_t = lr_{max} \cdot \frac{1 + \cos(\pi \cdot (t/T - 0.1)/0.9))}{2} \tag{22}$$

This policy linearly increases the learning rate from an initial value of 5×10^{-4} to a peak of $lr_{max} = 1.25 \times 10^{-3}$ over the first 10% of epochs, and then anneals it down using a cosine schedule for the remaining epochs. Here, T is the total number of epochs.

V. Evaluation

A. Setup

1) Implementation: We prototyped the system using P4 (1,300 LoC) and Python (2,500 LoC), executing data plane on the bmv2 software switch [25] using default configuration for reproducibility. The data plane comprises a primary defense module (utilizing Bloom filter for IP screening and meter for rate-limiting) and a feature extraction module (employing dual-hash tables with `resubmit` mechanism for collision handling). To accommodate massive concurrent flows, we configured the dual-hash table with 2^{15} registers. The variance detector requires three arrays per table (32-bit packet EWMA, 32-bit squared EWMA, and 8-bit trigger counter), resulting in a highly lightweight memory footprint of 471.8 KB ($2 \times 2^{15} \times 72$ bits), which is easily deployable on resource-constrained programmable switches.

The Python-based control plane uses P4Runtime for table management and PyTorch for the temporal detection model. A streaming channel was implemented to continuously receive digest messages containing feature data from the data plane.

2) Testbed: The system was deployed on a server VM with a 10-core CPU, 24GB of memory, and Ubuntu 24.04. The model was trained on an Nvidia GTX 1650Ti GPU and run on the CPU during system evaluation.

3) Datasets: The datasets were partitioned into specific roles:

- Benign Traffic: To accurately model the baseline behavior of modern IoT environments, benign traffic data was sourced from the CIC-IoT-2023 [26], CIC-DDoS-2019 [27], IEEE-TMC-2018 [28], CIC-IDS-2017 [24], and CIC-UNSW-NB15 [29] dataset. This data is used for model training, false positive rate evaluation, and baseline setting for data plane thresholds.
- Attack Traffic: Various volumetric attacks (e.g., SYN, UDP, SSDP, NTP) from the CIC-DDoS-2019 dataset were utilized. To align with practical application scenarios, we also introduced the Booters [23] dataset, which contains in-the-wild DDoS traffic (Chargen and DNS-based attacks) captured from real-world attack services. For model training, SYN/UDP attack types from the CIC-DDoS-2019 dataset were strictly employed.

4) Metrics: Evaluation metrics included the primary defense's True Positive Rate (TPR) for marking suspicious packets, the hash collision rate, the detection model's accuracy/ AUC/ F1-score, attack throughput reduction, and system response time.

5) Parameters: Default parameters are shown in Table IV.

TABLE IV
Experiment Parameters

Parameter	Value
EWMA present data weight α	0.125
Primary defense hit count threshold τ	200
Primary defense collection time window	1s
Suspicious IP packet rate	10pps
Maximum packet interval	5s
Maximum flow duration	10s

B. Data Plane Performance

The data plane's accuracy and efficiency is evaluated by analyzing its baseline throughput, numeric quantization error, parameter sensitivity, and processing latency.

1) Throughput and Pps: To establish a baseline processing capacity for the bmv2 switch in our virtualized environment, we measured direct packet forwarding between two hosts using `iperf` (5 runs, 30s each). The software switch sustained a maximum throughput of 86.40 ± 0.48 Mbps and a packet rate of 7455.23 ± 42.03 pps.

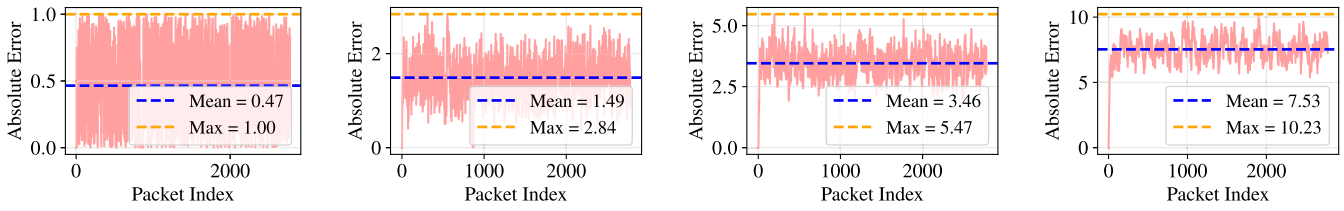
2) Numeric Quantization Error: Because P4 lacks floating-point support, the primary defense module approximates EWMA and variance using integer bit-shifts, introducing truncation errors. We empirically compared this fixed-point logic against an ideal floating-point implementation in Python using CIC-IoT-2023 benign traffic traces. The experiment evaluated different weight parameters, specifically $\alpha \in \{0.5, 0.25, 0.125, 0.0625\}$.

We measured the absolute error of the packet size EWMA and the corresponding relative error of the variance, as shown in Fig. 6 and 7. The results demonstrated that $\alpha = 0.125$ yields a maximum absolute error of 5.47 for the packet size EWMA, which complies with our theoretical analysis bounding the absolute error below 7. Furthermore, under this α, the EWMA algorithm introduces a minimal mean relative variance error of 1.37% among tested α values. Because our detection logic differentiates between the near-zero variance of automated DDoS flows and the massive variance of benign traffic, this minute quantization error is entirely absorbed by the threshold, introducing zero misclassifications.

3) EWMA Parameter Sensitivity Analysis: The primary defense identifies suspicious IPs when their moving variance stays below a baseline threshold for a predefined hit count (τ). Before the experiment, the EWMA data update time window was set to 1s, and the normal variance threshold was automatically established using truncated benign traffic Pcaps from CIC-IoT-2023 (replayed at 5 Mbps).

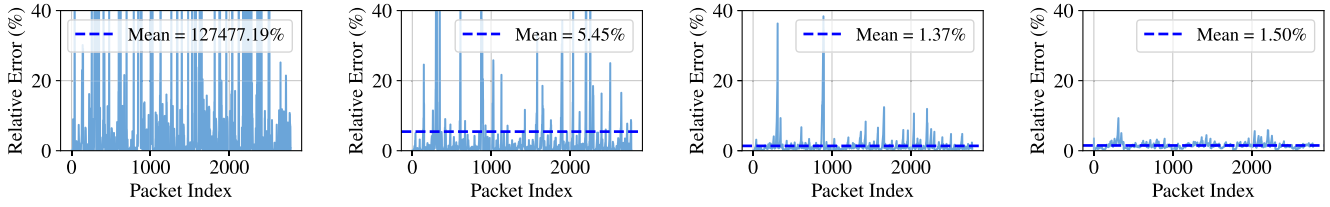
To minimize false positives on benign traffic, we analyzed the sensitivity of the EWMA weight (α) and trigger threshold (τ) using heterogeneous benign traffic from the CIC-IoT-2023 and TMC-IoT-2018 datasets. Fig. 8 shows the False Positive Rate (FPR) across $\tau \in [25, 200]$. Settings of $\alpha \geq 0.125$ and $\tau \geq 100$ yielded notably lower FPRs.

Integrating our quantization error findings, we fixed $\alpha = 0.125$ and evaluated the True Positive Rate (TPR) using 1000 pps UDP/TCP attacks (Fig. 9). The operable region (TPR $\geq 99.5\%$, FPR $\leq 0.5\%$) highlights where the defense maximizes attack interception while sparing benign traffic. We selected $\alpha = 0.125$ and $\tau = 200$ as the optimal operating point. Here, the system reliably filters highly dynamic benign



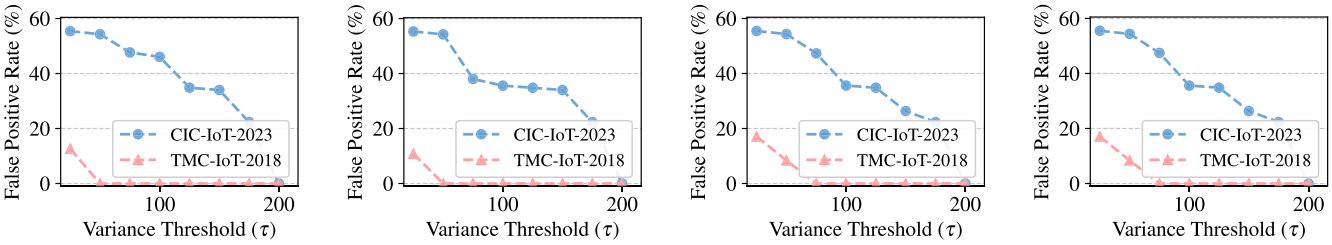
(a) $\alpha = 0.5$ (b) $\alpha = 0.25$ (c) $\alpha = 0.125$ (d) $\alpha = 0.0625$

Fig. 6. Absolute error of packet size EWMA under different weight.



(a) $\alpha = 0.5$ (b) $\alpha = 0.25$ (c) $\alpha = 0.125$ (d) $\alpha = 0.0625$

Fig. 7. Relative error of packet size variance under different weight.



(a) $\alpha = 0.5$ (b) $\alpha = 0.25$ (c) $\alpha = 0.125$ (d) $\alpha = 0.0625$

Fig. 8. FPR of benign traffic packets under different EWMA weight α.

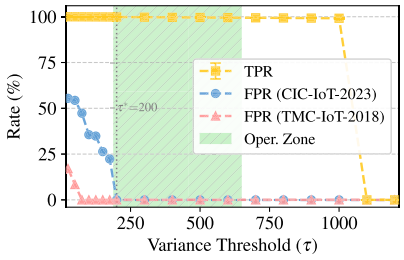
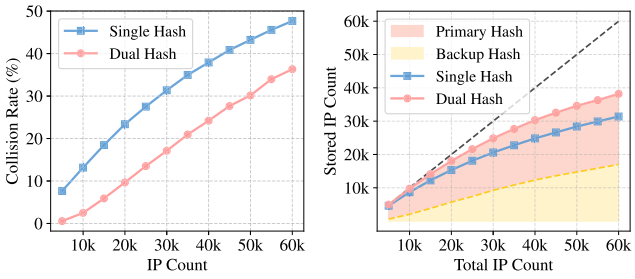


Fig. 9. Positive rate of traffic packets under different variance threshold τ.



(a) Collision Rate (b) Hash Table Usage

Fig. 10. Comparison of collision situations between single and dual hash table.

bursts (0% FPR) while successfully mitigating continuous attack flows with minimal latency.

4) Hash Collision Rate: The framework employs a dual hash table to increase its capacity to track concurrent flows from multi-source DDoS attacks. Experiments compared a single 65536-entry hash table with a dual-table configuration (two 32768-entry tables), as shown in Figure 10(a). With 10,000 concurrent flows, the dual-table design's collision rate was only 18.86% of the single table's rate.

Since the use of a backup hash table involves the recirculation of related data packets at the switch end, it introduces additional forwarding delay. Therefore, forwarding should prioritize maximizing the use of the primary hash table. Figure 10 showed the primary table remains highly utilized, storing 78.81% of flows in the 10,000-flow scenario.

5) Packet Processing Latency: To handle hash collisions, the proposed dual-hash design uses the P4 `resubmit` mechanism, which restarts processing exclusively at the ingress pipeline, avoiding the higher overhead of full `recirculate`. The processing latency for a single packet was calculated as the difference between the timestamp of exiting and entering the processing queue. To quantify this, we compared the latency distribution of individual packets between a single hash

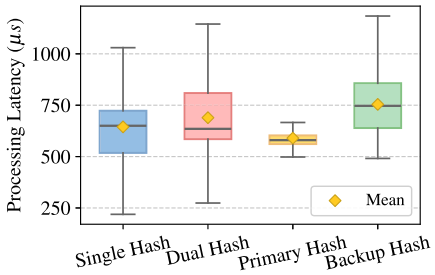


Fig. 11. Packet processing latency distributions for single-hash and dual-hash table configurations.

TABLE V
MODEL DETECTION RESULTS

Dataset	Type	$L=3$	$L=5$	$L=8$	$L=12$
CIC-IoT-2023	Benign	98.68	99.09	97.07	87.46
TMC-IoT-2018	Benign	93.09	96.03	92.16	91.80
Booter	CharGen	99.10	100	100	100
	DNS	92.26	95.76	100	96.21
CIC-DDoS-2019	UDP	62.68	99.80	99.96	94.24
	SSDP	61.04	98.99	99.98	91.97
	NTP	80.73	99.53	99.70	93.59
	NetBIOS	1.16	98.27	99.99	20.36
	Portmap	1.11	96.30	99.72	13.61
	SNMP	0.14	84.10	99.96	2.54
	MSSQL	1.16	83.4	99.65	7.39
	LDAP	0.65	76.79	99.99	16.13

table (sized 2^{16}) and a dual-hash table (each sized 2^{15}) using traffic from 60,000 distinct flows.

As shown in Fig. 11, the mean latencies for the single-hash and dual-hash configurations are 644.6 μ s and 689.29 μ s, respectively, with a minimal overhead of just 44.68 μ s. Within the dual-hash setup, primary and backup table mean latencies are 589.80 μ s and 754.00 μ s (a difference of 164.2 μ s). Because the dual-hash design heavily utilizes the faster primary table while drastically reducing collision rates, the median processing time is actually lower for the dual-hash approach (635.00 μ s vs. 650.00 μ s). Thus, the dual-hash structure effectively maps high-throughput traffic with negligible latency penalties.

C. Control Plane Performance

1) Model Training and Testing: The detection model was trained on 5,361,343 attack and 5,680,779 benign traffic records. To align with data-plane collection windows, datasets were processed into feature vector sequences of length L (max flow duration 10s, idle timeout 5s) and split into training (64%), validation (16%), and test (20%) sets. Models utilizing sequence lengths $L \in \{3, 5, 8, 12\}$ all achieved $>$ 99.93% accuracy and $>$ 0.9992 F1 scores on the test set.

2) Sequence Length Ablation: To evaluate model generalization, we conducted an ablation study varying $L \in \{3, 5, 8, 12\}$. The tests incorporated unseen reflective attacks (e.g., SSDP, NTP) alongside real-world attacks (Booter), despite the model being trained exclusively on direct SYN/UDP floods. Results shown in Table V demonstrated that $L = 5$ maximizes the True Negative Rate (TNR) for benign IoT traffic (up to 99.09%), while $L = 8$ optimizes the True Positive Rate (TPR) for attack traffic ($>$ 99.6%). Attack TPR initially increases with sequence length as temporal context enriches, but subsequently decreases, likely because excessive lengths exceed the GRU's optimal capacity and induce long-term memory decay.

Based on the dataset detection results, we plotted the PR/ROC curves shown in Figure 12 and Figure 13, respectively. Due to class imbalance, we prioritize the PR-AUC metric, which reaches 0.9977 at $L = 5$. Balancing detection accuracy and preprocessing latency, we ultimately selected $L = 5$ for the system's sequence detection mechanism. It provides optimal benign traffic identification while maintaining a $>$ 96.3% TPR across most attack types. Furthermore, under the 10-second window setting, $L = 5$ guarantees the control plane can identify and block threats in under one minute, confirming it achieves the optimal balance between high accuracy and agile system response.

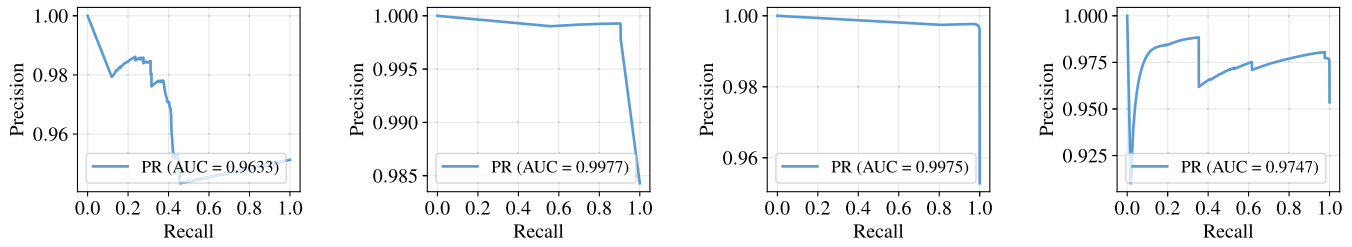
3) Resource Overhead: Flow state tracking relies on a lightweight double-ended queue (length 5) storing a 21-dimensional statistical vector. To evaluate the scalability of maintaining temporal sequences, we measured the overall system memory usage of the control plane processes under scenarios with no traffic digests and with attack traffic digests. By comparing the incremental memory consumption across different scenarios, the experiment confirmed that the runtime control plane RAM usage is 386 MB, with the amortized memory per active suspicious flow being approximately 7.73 KB. This demonstrates that the control plane can continuously process a massive number of concurrent suspicious flows without exceeding standard server memory limits.

D. System Performance

1) Detection Performance: To evaluate the performance of the proposed primary defense mechanism at detecting attack traffic while correctly handling benign traffic, we conducted experiments using benign traffic from the IEEE-TMC-2018 and CIC-IoT-2023 datasets, and custom-built UDP/TCP attack traffic replayed for 120 seconds. The attack traffic was configured with 1, 5, and 10 source IPs, with each source maintaining a sending rate of 1kpps.

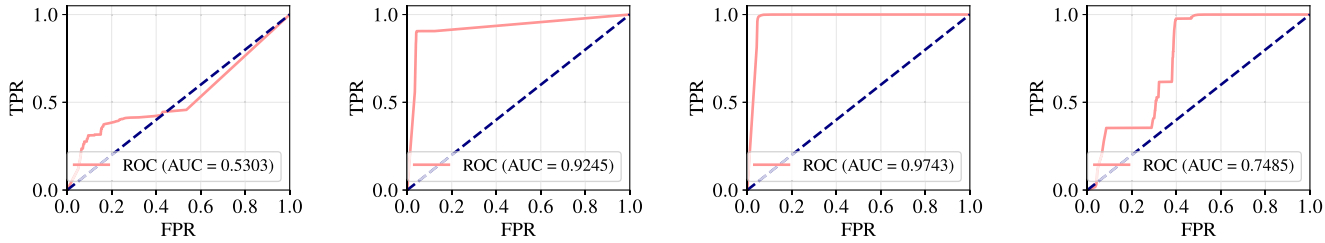
We compared the detection performance of our method against two state-of-the-art P4-based approaches: Euclid [30], which uses the Shannon entropy of IP address distributions to detect anomalies, and P4RTHENON [31], which identifies attacks based on the packet count difference between forward and reverse traffic directions. To specifically isolate and evaluate the effectiveness of our primary defense module, we also conducted an ablation study by testing our framework with the subsequent flow-level feature extraction disabled.

Each method was configured according to its specification. Our method used a trigger threshold of 200 after being baselined with 10k packets of benign traffic. Euclid used an observation window of 2^{14} packets and was baselined with 40k benign packets. P4RTHENON used a trigger threshold of 200 and a 1s window size.



(a) $L = 3$ (b) $L = 5$ (c) $L = 8$ (d) $L = 12$

Fig. 12. PR curve of models on the collected dataset.



(a) $L = 3$ (b) $L = 5$ (c) $L = 8$ (d) $L = 12$

Fig. 13. ROC curve of models on the collected dataset.

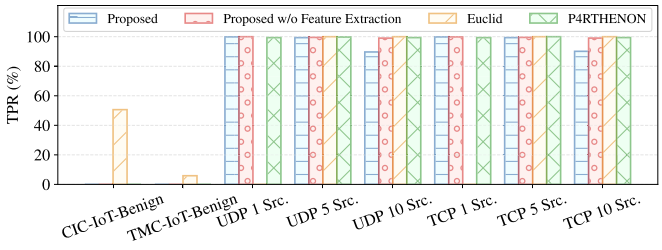
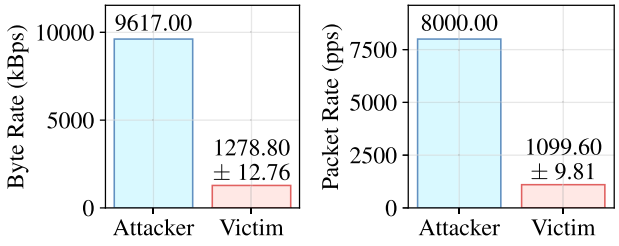


Fig. 14. Detection performance of frameworks under different traffic types.



(a) Byte Rate (b) Packet Rate

Fig. 15. Mitigation performance under chargen DDoS attack.

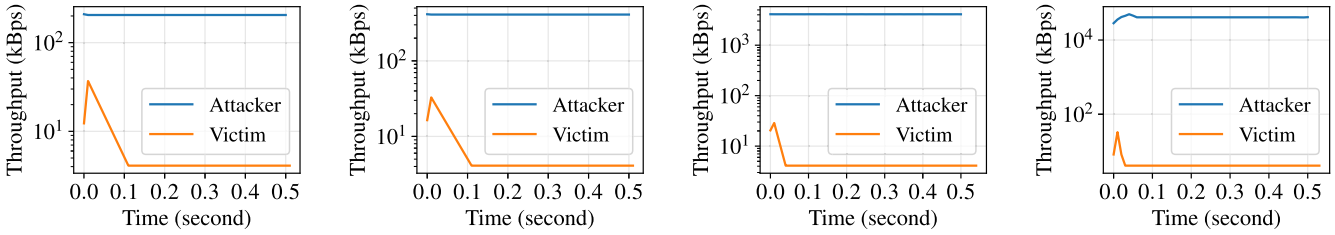
The results, shown in figure Figure 14, evaluate the proportion of packets marked as suspicious. Our proposed method achieved a detection rate of over 99% for attacks of all intensities while generating zero false positives on benign traffic. When the full framework including feature extraction was enabled, the detection rate for the 10-IP attack dropped to approximately 90%. This is attributable to packet loss in the resource-constrained software switch environment and is considered a limitation of the testbed, not the method itself.

In comparison, while Euclid also achieved a detection rate above 99% for multi-source attacks, it failed to detect single-source DoS attacks and exhibited a very high false positive rate, flagging 50.61% of the benign CIC-IoT-2023 traffic as malicious. This is due to a fundamental mismatch between its entropy-based mechanism and the traffic patterns of DoS and IoT networks. First, a single-source DoS attack does not generate enough flow diversity to trigger Euclid's IP frequency thresholds. Second, normal IoT network behavior often involves many devices communicating periodically with a single gateway (a many-to-one pattern), which creates entropy fluctuations that Euclid misinterprets as an attack.

P4RTHENON performed similarly to our primary defense, achieving a detection rate above 99% with no false positives. However, a key distinction is that P4RTHENON lacks an in-network mechanism for extracting flow-level features. After flagging suspicious traffic, it only samples packet-level features to send to an external collector. This design choice prevents the use of more sophisticated detection models that rely on the rich, contextual information provided by flow-level statistics, which our framework is designed to support.

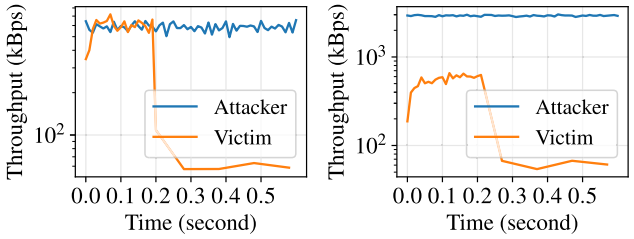
2) Mitigation Performance: The mitigation rate is strictly defined as the percentage difference between attack traffic volume (bytes and packets) sent by the attacker versus successfully received by the victim, measured via synchronized Pcap captures. To evaluate real-world resilience, we replayed a Chargen-based DDoS attack (8000 pps, 70s) against background CIC-IoT-2023 benign traffic.

As shown in Figure 15, the primary defense immediately identified the low-variance attack and applied a 10 pps rate limit, dropping the attack volume and packet rate by $86.70 \pm 0.13\%$ and $86.25 \pm 0.12\%$, respectively, effectively mitigating the attack traffic bursts on downstream devices.



(a) UDP, 5kpps (b) UDP, 10kpps (c) UDP, 100kpps (d) UDP, 1000kpps

Fig. 16. Mitigation performance under UDP DoS attack.



(a) Random-sized UDP, 1kpps (b) Random-sized UDP, 5kpps

Fig. 17. Mitigation performance under random size UDP DoS attack.

TABLE VI
LATENCY ANALYSIS OF PROCESSING ATTACK TRAFFIC

Pkt Rate (kpps)	Flow Rate	First Pkt (ms)	First Digest (s)	Block Policy (s)
5	210kBps	6.38	11.79	51.78
10	404kBps	5.58	11.75	51.83
100	3725kBps	4.22	11.75	51.78
1000	40MBps	7.21	11.76	51.79

3) Latency Performance: To evaluate the system's response time to sustained attacks, the experiment continuously looped a Pcap file containing 5000 UDP attack packets, evaluating various attack rates from 5 to 1000 kpps, while simultaneously simulating a benign IoT device transmitting background telemetry data at a rate of 1 UDP packet per 0.5 seconds. As shown in Figure 16, the system identifies and rate-limits the packet rate of the corresponding IP source at the millisecond level during the early stages of the attack, while ensuring that the packet drop rate for normal traffic remains at 0%.

To evaluate resilience against adversarial evasion, we randomized attack payload sizes (500-700 bytes, 10,000 packets) to inflate variance and mimic benign dynamics. As illustrated in Figure 17, while this artificial variance temporarily delays the EWMA trigger, incurring approximately 100 ms of additional latency compared to fixed-size attacks, the sustained high packet rate inevitably forces the moving variance below the benign baseline. The primary defense successfully mitigates the threat, proving the robustness of the variance-rate thresholding mechanism against dynamic evasions.

Furthermore, the experiment evaluated the response time for attack detection and mitigation within the system, as presented in Table VI. The total time from attack inception to blocklisting was approximately 51 seconds. This delay is primarily dictated by the control plane's need to collect a full sequence of 5 feature vectors, with each vector generated from a 10-second time window (5 * 10s = 50s, plus processing time).

VI. DISCUSSION

The experimental evaluation demonstrates that our framework effectively balances real-time performance and accuracy requirements for DDoS defense in IoT environments. The system significantly mitigates attack traffic in its initial stages while providing accurate source localization and blocking. The adaptive variance-based threshold automatically adjusts its statistical basis, enabling effective defense against various high-volume DDoS attacks without manual parameter tuning across different deployment scenarios. Additionally, by utilizing traffic sequences as input, the detection model successfully distinguishes the high-rate, high-volume, low-variance characteristics of attack traffic from legitimate traffic, thereby improving detection accuracy and minimizing false positives. Nevertheless, several limitations in the proposed approach warrant future investigation.

A. Defense Against Dynamic, Low-Rate Attacks

The current data-plane mechanism fundamentally relies on the low-variance characteristics of sustained volumetric attacks, making it vulnerable to evasive tactics. In a pulsing DoS attack, adversaries alternate traffic bursts with silent periods to artificially inflate statistical variance, while in a low-rate DoS attack, they transmit slowly to avoid triggering count thresholds. Countering these evasions requires integrating frequency-domain features, periodicity tracking, and advanced bio-inspired computing paradigms [32] to establish a higher-precision source identification mechanism.

B. Model Generalization and Adaptation

While the offline-trained CNN-GRU-Attention model generalizes well to several unseen reflective attacks, its detection rate drops when encountering highly irregular or zero-day protocols. Ensuring long-term practical feasibility in dynamic industrial deployments requires shifting toward adaptive architectures. Future iterations could establish unsupervised anomaly detection baselines (e.g., temporal autoencoders trained exclusively on benign IoT features) and implement lightweight, online domain adaptation routines to rapidly update model weights using newly collected statistical digests.

C. Real-World Environment Validation

While prototyped on a software switch, translating this framework to a physical programmable ASIC (e.g., Intel Tofino) introduces strict hardware constraints. Complex operations like EWMA multiplication must be adapted using hardware-specific MathUnit externs within stateful ALUs, and stateful metadata must be meticulously allocated across limited pipeline stages to satisfy memory access restrictions. Although hardware deployment will drastically reduce data-plane processing latency from milliseconds to microseconds, the overall control-plane mitigation latency will fundamentally remain bounded by the configured flow monitoring time slices.

VII. Related Work

A. ML-Based Detection Methods

To leverage the extensive information within public network traffic datasets, machine learning (ML) models are widely employed in network anomaly detection [4], [5], [7], [8], [33], [34], [35]. These models excel at mining traffic features to build detection systems with high accuracy and generalization, establishing this as a mainstream research direction. While some studies use unsupervised clustering methods to defend against unknown attacks [3], [16], the majority of works utilize labeled datasets to train supervised ML models for accurate attack type classification.

For instance, Tang et al. [8] developed a lightweight, real-time framework to detect and mitigate LDoS attacks in SDN; it uses a GBDT model to detect the presence of an attack and a time-frequency analysis algorithm to identify traffic bursts. The work in [5] trains interpretable decision trees that achieve detection performance comparable to closed-box models, using pruning to balance fidelity and complexity, thereby making pre-trained models more understandable. Wichtlhuber et al. [7] introduces a system that learns DDoS traffic characteristics from adjacent Autonomous Systems (ASes) at Internet Exchange Points (IXPs), using BGP signals to blackhole and sample DDoS traffic, which enables learning new attack vectors without carrier intervention. Wang et al. [4] proposed a method that constructs a digital twin of the physical device topology. It identifies IoT attack behaviors by parallelly extracting and fusing spatio-temporal features using an attention mechanism.

Moreover, the landscape of AI-driven security has actively evolved to address advanced threats hidden within specialized channels [22]. While these protocol-specific AI methods excel at isolating complex and stealthy attacks within targeted payloads, they are structurally designed for deep, localized inspection. Applying such resource-intensive, targeted inference across the immense diversity of traffic forms in industrial networks is largely infeasible for real-time volumetric defense.

Despite achieving high detection accuracy through complex models and feature engineering, these approaches share a core challenge: their complete reliance on external servers introduces unavoidable communication overhead and processing delay. The process of uploading traffic features or raw packets to a server for analysis and then relaying decisions back to network devices fails to meet the stringent real-time response requirements of scenarios like the Industrial Internet of Things (IIoT). Our cross-plane framework complements these deep-inspection methods by providing a lightweight, protocol-agnostic first line of defense, filtering out volumetric, low-variance attacks directly in the data plane before they can exhaust network resources. Only the features of initially screened, suspicious traffic are sent to the control plane server for more sophisticated temporal analysis. This approach reduces unnecessary feature transmission overhead without sacrificing final detection accuracy.

B. Programmable Switch-Based Detection Methods

Research in this area primarily focuses on optimizing the speed and accuracy of malicious traffic detection. Work based on programmable switches [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21] aims to fully leverage the hardware's line-rate, high-throughput packet processing capabilities while circumventing the limitations of register resources and data plane programming languages.

1) Efficient Resource Management: These studies design mechanisms for updating, sharing, and migrating data to ensure real-time dynamic recording despite the limited size of registers used for monitoring tasks. For example, the work in [13] dynamically allocates and shares registers between monitoring tasks and offloads low-frequency data records to the control plane to overcome resource shortages. Mirnajafizadeh et al. [14] proposed a serverless in-network data collection framework that uses a distributed protocol for dynamic monitoring task migration, avoiding the overhead caused by negotiation during task allocation and ensuring full traffic measurement coverage.

2) Optimized Recognition Mechanisms: These works focus on distinguishing between normal and malicious traffic through attack source and packet feature distribution patterns. For instance, Misa et al. [15] utilizes the prefix aggregation characteristic of attack IPs to dynamically maintain the granularity of traffic monitoring entries, thereby reducing resource consumption. Another, Alcoz et al. [16], designs an in-hardware distance function to perform online clustering of packets for attack inference. Although these works can identify attack traffic in the data plane with high accuracy, their corresponding detection response times are in the tens of seconds, which fails to meet the real-time response requirements of resource-constrained IoT devices.

3) ML-Enabled Data Plane: These work focuses on enabling hardware with ML models by mapping them into the rule-matching tables available for packet processing. For example, Yan et al. [17] implemented RNN feature embedding, GRU units, and softmax layers in hardware. Jafri et al. [18] supported the representation of arbitrary decision tree structures. Dong et al. [19] generated hardware-based access list matching rules from unsupervised isolation forest models. Zhou et al. [20] reduced table entry consumption for tree and forest models. Akem et al. [21] implemented mapping strategies for Random Forests (RF). While such approaches can leverage switch tables to process packets at line rate—achieving low latency and high throughput unattainable by the control plane—they are limited by the complexity

(e.g., number of layers, trees) of the models they can map. Furthermore, models deployed in hardware are difficult to update dynamically; the downtime required to redeploy a new model creates a window of opportunity for attackers.

C. Programmable Switch-Based Mitigation Methods

1) Modular Policy Design: Some studies provide simple, modular policy abstractions that shield developers from underlying hardware complexities, allowing them to use primitive APIs to write defense scripts for various attacks. For example, Zhang et al. [36] proposed a framework that records packets in a sketch structure and uses statistical primitives to collect feature data in real-time. Liu et al. [37] proposed an ISP-level defense against high-volume DDoS, abstracting primitive APIs into filtering, analysis, and updating components. Xing et al. [38] built a decentralized defense network with custom operators for traffic measurement, capturing network-wide records to form a panoramic view. Zhou et al. [39] designed a distributed defense system against LFA attacks, providing custom collaborative defense APIs to synchronize information among device groups when predefined conditions are met.

2) Refactoring Existing Defenses: Other works refactor traditional defense mechanisms, improving them with modern, higher-performance hardware to enhance their efficacy. For example, Yoo et al. [40] refactors the traditional SYN cookie into a collaborative switch and server proxy, where the switch performs high-speed, secure cookie checks to defend against SYN floods. The work by Alcoz et al. [16] improves the ACC mechanism with an online aggregation module in the data plane, using a heuristic-based distance function to aggregate similar packets and control the forwarding rate of high-bandwidth clusters to defend against pulse-wave DDoS attacks. Compared to the aforementioned works, the framework proposed in this paper does not merely optimize a single aspect within the data plane. Instead, it aims to build a complete, phased defense system where the data plane and control plane work in synergy. This approach achieves a better balance between detection timeliness, accuracy, and resource overhead, satisfying the defense requirements of IoT environments.

VIII. Conclusion

In this paper, we proposed a two-stage detection and defense framework targeting DDoS attacks in IoT environments. It deploys a lightweight primary defense in the data plane to handle suspicious traffic and, when necessary, records fine-grained, flow-level features to enable more precise detection by the control plane. In the data plane, EWMA-based variance detection effectively identifies low-variance attack traffic, while a bidirectional dual hash table architecture records features with low hash collision rates. In the control plane, a temporal sequence-based detection model accurately identifies malicious behavior. Our evaluation demonstrates that the framework can mitigate the vast majority of suspicious attack traffic within milliseconds and can accurately locate and completely block attack sources within a minute.

As future work, we plan to extend the proposed solution to detect and mitigate attacks beyond the high-volume DDoS category, such as low-rate attacks and link-flooding attacks. This will involve developing timely rate-limiting responses and high-precision source identification for these new threats, likely requiring further advances in statistical methods and machine learning techniques.

References

[1] *State of IoT summer* 2024. Accessed: Jul. 16, 2025. [Online]. Available: https://iot-analytics.com/product/state-of-iot-summer-2024/

[2] *Nozomi Networks*. Accessed: Jul. 16, 2025. [Online]. Available: https://www.nozominetworks.com/

[3] C. Fu, Q. Li, K. Xu, and J. Wu, "Point cloud analysis for ML-based malicious traffic detection: Reducing majorities of false positive alarms," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2023, pp. 1005–1019.

[4] H. Wang, X. Di, Y. Wang, B. Ren, G. Gao, and J. Deng, "An intelligent digital twin method based on spatio-temporal feature fusion for IoT attack behavior identification," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3561–3572, Nov. 2023.

[5] A. S. Jacobs, R. Beltiukov, W. Willinger, R. A. Ferreira, A. Gupta, and L. Z. Granville, "AI/ML for network security: The emperor has no clothes," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2022, pp. 1537–1551.

[6] D. Wagner et al., "United we stand: Collaborative detection and mitigation of amplification DDoS attacks at scale," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 970–987.

[7] M. Wichtlhuber et al., "IXP scrubber: Learning from blackholing traffic for ML-driven DDoS detection at scale," in *Proc. ACM SIGCOMM Conf.*, Aug. 2022, pp. 707–722.

[8] D. Tang, Y. Yan, S. Zhang, J. Chen, and Z. Qin, "Performance and features: Mitigating the low-rate TCP-targeted DoS attack via SDN," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 428–444, Jan. 2022.

[9] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic DDoS defense," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 817–832.

[10] T. Cai, T. Jia, S. Adepu, Y. Li, and Z. Yang, "ADAM: An adaptive DDoS attack mitigation scheme in software-defined cyber-physical system," *IEEE Trans. Ind. Informat.*, vol. 19, no. 6, pp. 7802–7813, Jun. 2023.

[11] H. Namkung, Z. Liu, D. Kim, V. Sekar, and P. Steenkiste, "Sketchovsky: Enabling ensembles of sketches on programmable switches," in *Proc. 21st USENIX Symp. Networked Syst. Design Implement.*, 2023, pp. 1273–1292.

[12] R. Das and A. C. Snoeren, "Memory management in ActiveRMT: Towards runtime-programmable switches," in *Proc. ACM SIGCOMM Conf.*, Sep. 2023, pp. 1043–1059.

[13] H. Zhou and G. Gu, "Cerberus: Enabling efficient and effective in-network monitoring on programmable switches," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2024, pp. 4424–4439.

[14] S. M. M. Mirnajafizadeh, A. R. Sethuram, D. Mohaisen, D. Nyang, and R. Jang, "Enhancing network attack detection with distributed and in-network data collection system," in *Proc. 33rd USENIX Security Symp.*, 2024, pp. 5161–5178.

[15] C. Misa, R. Durairajan, A. Gupta, R. Rejaie, and W. Willinger, "Leveraging prefix structure to detect volumetric DDoS attack signatures with programmable switches," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2024, pp. 4535–4553.

[16] A. G. Alcoz, M. Strohmeier, V. Lenders, and L. Vanbever, "Aggregate-based congestion control for pulse-wave DDoS defense," in *Proc. ACM SIGCOMM Conf.*, Aug. 2022, pp. 693–706.

[17] J. Yan et al., "Brain-on-switch: Towards advanced intelligent network data plane via NN-driven traffic analysis at line-speed," in *Proc. 21st USENIX Symp. Networked Syst. DesignImplement.*, 2024, pp. 419–440.

[18] S. U. Jafri, S. Rao, V. Shrivastav, and M. Tawarmalani, "Leo: Online ML-based traffic classification at multi-terabit line rate," in *Proc. 21st USENIX Symp. Networked Syst. Design Implement.*, 2024, pp. 1573–1591.

[19] Y. Dong et al., "HorusEye: A real-time IoT malicious traffic detection framework using programmable switches," in *Proc. 32nd USENIX Security Symp.*, 2023, pp. 571–588.

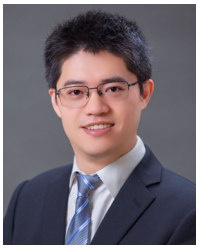
[20] G. Zhou, Z. Liu, C. Fu, Q. Li, and K. Xu, "An efficient design of intelligent network data plane," in *Proc. 32nd USENIX Secur. Symp.*, 2023, pp. 6203–6220.

[21] A. T.-J. Akem, M. Gucciardo, and M. Fiore, "Flowrest: Practical flow-level inference in programmable switches with random forests," in *Proc. IEEE Conf. Comput. Commun.*, May 2023, pp. 1–10.
[22] B. Ali and G. Chen, "Next-generation AI for advanced threat detection and security enhancement in DNS over HTTPS," *J. Netw. Comput. Appl.*, vol. 244, Dec. 2025, Art. no. 104326.
[23] J. J. Santanna et al., "Booters—An analysis of DDoS-as-a-service attacks," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 243–251.
[24] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.
[25] *Behavioral Model (BMV2)*. Accessed: Jul. 16, 2025. [Online]. Available: https://github.com/p4lang/behavioral-model
[26] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," *Sensors*, vol. 23, no. 13, p. 5941, Jun. 2023.
[27] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2019, pp. 1–8.
[28] A. Sivanathan et al., "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1745–1759, Aug. 2019.
[29] H. Mohammadian, A. Habibi Lashkari, and A. A. Ghorbani, "Poisoning and evasion: Deep learning-based NIDS under adversarial attacks," in *Proc. 21st Annu. Int. Conf. Privacy, Secur. Trust (PST)*, Aug. 2024, pp. 1–9.
[30] A. D. S. Ilha, Â. C. Lapolli, J. A. Marques, and L. P. Gaspary, "Euclid: A fully in-network, P4-based approach for real-time DDoS attack detection and mitigation," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 3, pp. 3121–3139, Sep. 2021.
[31] A. A. Sadi, M. Savi, A. Melis, M. Prandini, and F. Callegati, "Unleashing dynamic pipeline reconfiguration of P4 switches for efficient network monitoring," *IEEE Trans. Netw. Service Manage.*, vol. 21, no. 3, pp. 3482–3497, Jun. 2024.
[32] B. Ali and G. Chen, "Neuromorphic quantum adversarial learning (NQAL): A bio-inspired paradigm for DNS over HTTPS threat detection," *EURASIP J. Inf. Secur.*, vol. 2025, no. 1, p. 35, Dec. 2025.
[33] J. Holland, P. Schmitt, N. Feamster, and P. Mittal, "New directions in automated traffic analysis," in *Proc. ACM IGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 3366–3383.
[34] X.-H. Nguyen and K.-H. Le, "Robust detection of unknown DoS/DDoS attacks in IoT networks using a hybrid learning model," *Internet Things*, vol. 23, Oct. 2023, Art. no. 100851.
[35] G. O. Anyanwu, C. I. Nwakanma, J.-M. Lee, and D.-S. Kim, "RBF-SVM kernel-based model for detecting DDoS attacks in SDN integrated vehicular network," *Ad Hoc Netw.*, vol. 140, Mar. 2023, Art. no. 103026.
[36] M. Zhang et al., "Poseidon: Mitigating volumetric DDoS attacks with programmable switches," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020, pp. 1–9.
[37] Z. Liu et al., "Jaqen: A high-performance switch-native approach for detecting and mitigating volumetric DDoS attacks with programmable switches," in *Proc. USENIX Secur. Symp.*, 2021, pp. 3829–3846.
[38] J. Xing, W. Wu, and A. Chen, "Ripple: A programmable, decentralized link-flooding defense against adaptive adversaries," in *Proc. 30th USENIX Security Symp.*, 2021, pp. 3865–3881.
[39] H. Zhou, S. Hong, Y. Liu, X. Luo, W. Li, and G. Gu, "Mew: Enabling large-scale and dynamic link-flooding defenses on programmable switches," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2023, pp. 3178–3192.
[40] S. Yoo, X. Chen, and J. Rexford, "SMARTCOOKIE: Blocking large-scale SYN floods with a split-proxy defense on programmable data planes," in *Proc. 33rd USENIX Security Symp.*, 2024, pp. 217–234.

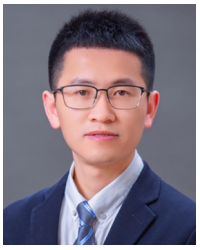
Jiahang Pu received the B.S. degree in software engineering from Southeast University in 2023, where he is currently pursuing the combined master's and Ph.D. degree with the School of Computer Science and Engineering. His research interests include network security and privacy and the Internet of Things.

Hongyu Ye received the B.S. degree from Sichuan University in 2023. He is currently pursuing the master's degree in computer science and technology with Southeast University. His research interests include network security and privacy and the Internet of Things.

Jing Cheng received the B.S. degree from Hefei University of Technology and the M.S. degree from Southeast University, where she is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. Her research interests include LoRa, the Internet of Things, and game theory.



Feng Shan (Member, IEEE) received the Ph.D. degree in computer science from Southeast University, Nanjing, China, in 2015. He visited the School of Computing and Engineering, University of Missouri–Kansas City, Kansas City, MO, USA, from 2010 to 2012. He is currently an Associate Professor with the School of Computer Science and Engineering, Southeast University. His research interests include the areas of Internet of Things, wireless networks, swarm intelligence, and algorithm design and analysis.



Runqun Xiong (Member, IEEE) received the Ph.D. degree in computer science from Southeast University, China. He was with European Organization for Nuclear Research as a Research Associate for the AMS-02 Experiment from 2011 to 2012. He is currently an Associate Professor with the School of Computer Science and Engineering, Southeast University, where he is involved in AMS-02 data processing with the AMS Science Operations Center. His current research interests include the Internet of Things, cyber-physical systems, and wireless networks. He is a member of ACM and China Computer Federation.