FAPM: A Fake Amplification Phenomenon Monitor to Filter DRDoS Attacks With P4 Data Plane

Dan Tang, Xiaocai Wang, Keqin Li, *Fellow, IEEE*, Chao Yin, Wei Liang, and Jiliang Zhang, *Senior Member, IEEE*

***Abstract*—Distributed Reflection Denial-of-Service (DRDoS) attacks have caused significant destructive effects by virtue of emerging protocol vulnerabilities and amplification advantages, and their intensity is increasing. The emergence of programmable data plane supporting line-rate forwarding provides a new opportunity for fine-grained and efficient attack detection. This paper proposed a light-weight DRDoS attack detection and mitigation system called FAPM, which is deployed at the victim end with the intention of detecting the amplification behavior caused by the attack. It places the work of collecting and calculating reflection features on the data plane operated by "latter window assisting former window" mechanism, and arranges complex identification and regulation logic on the control plane. This approach avoids the hardware constraints of the programmable switch while leveraging their per-packet processing capability. Also, it reduces communication traffic significantly through feature compression and state transitions. Experiments show that FAPM has (1) fast response capability within seconds (2) a memory footprint at the KB level and communication overhead of 1 Kbps, and (3) good robustness.**

***Index Terms*—Attack mitigation, distributed reflection denial-of-service, fake amplification phenomenon, P4.**

I. Introduction

DISTRIBUTED Reflection Denial-of-Service (DRDoS) attacks [1] are a common but highly threatening type of denial-of-service attack, showing a growing trend, mainly manifested in the huge volume and endless protocol loopholes. In 2018, a Memcached-based DRDoS attack overwhelmed the code hosting website GitHub with a peak traffic of 1.35 Tbps [2]. A tool named AMPFUZZ [3] has discovered several unreported protocol vulnerabilities in Debian network services that can be exploited for DRDoS attacks. The endless protocol vulnerabilities are also an important factor in promoting the development of DRDoS attacks, and therefore can impact various network architectures, such as vehicular networks [4], cloud computing [5], Software-Defined Networking [6], and the wider Internet of Things [7].

DRDoS attackers control a botnet to send a series of protocol-specific request packets, with the source IP spoofed to match the target host. By exploiting vulnerabilities in the protocol, the attack can generate an amplification effect. Upon receiving the IP-spoofed requests, the involved servers will send a large volume of response packets to the victim host to destroy it, which may not even be using the service related to the protocol.

Because DRDoS attacks violate the normal request-response relationship, many researches detect them by establishing mappings in a session. RALF [8] logs attributes of request packets, only allowing responses if these attributes match, and triggers an alert if mismatches exceed a threshold. WisdomSDN [9] can detect and mitigate DNS amplification attacks in SDN. After learning each DNS request packet, the switch installs a corresponding flow table rule, which indicates a one-to-one mapping of the DNS response packet by swapping source and destination IP addresses and ports. The response packets are forwarded or discarded based on the flow table, and each rule has a short timeout interval because of limited capacity. Due to cache limitations and reliance on empirical knowledge, these methods are often protocol-specific and can only filter known protocol attack packets.

Another category of methods introduces the authentication mechanism lacking in the IP protocol and its upper-layer UDP protocol to identify DRDoS attack packets. When receiving a request packet, a server deployed with ESFD [10] first initiates a TCP connection with the host that sent the request to preliminarily confirm its validity. If the connection is successfully established, the server then queries the host for the timestamp of the previous request packet. If the host can provide the correct timestamp, the previous request packet is considered legitimate and can be forwarded or responded to. For the purpose of reflection and amplification, DRDoS attackers forge the source IP of request packets. Although the IP address can be changed arbitrarily, the actual forwarding and routing hops required for the packet in the network are fixed. A method called HCF [11] filters IP spoofing packets based on their hop count. The server establishes and maintains a mapping table of IP addresses and corresponding hop counts

Manuscript received 22 January 2024; revised 31 July 2024; accepted 12 August 2024. Date of publication 26 August 2024; date of current version 20 December 2024. This work was supported in part by the National Natural Science Foundation of China (#62472153), the YueLuShan Center Industrial Innovation Project (#2023YCII0115), and the Science and Technology Key Projects of Changsha City (#kq2208038). The associate editor coordinating the review of this article and approving it for publication was G. Schembra. *(Corresponding author: Jiliang Zhang.)*

Dan Tang and Xiaocai Wang are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: Dtang@hnu.edu.cn; xiaocaiwang@hnu.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, NY 12561 USA, and also with Hunan University, Changsha 410082, China (e-mail: lik@newpaltz.edu).

Chao Yin is with the School of Computer and Big Data Science, Jiujiang University, Jiujiang 332000, China (e-mail: david_yin@jju.edu.cn).

Wei Liang is with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411199, China (e-mail: wliang@hnust.edu.cn).

Jiliang Zhang is with the College of Semiconductors (College of Integrated Circuits), Hunan University, Changsha 410082, China (e-mail: zhangjiliang@hnu.edu.cn).

Digital Object Identifier 10.1109/TNSM.2024.3449889

in advance. Upon receiving a request packet, the TTL field is extracted to infer the number of hops the packet has traversed. If this count is inconsistent with the mapping table, the packet is identified as an IP spoofing packet. As long as the IP deception packets can be identified and filtered, the enlarged path of DRDoS attacks can be destroyed.

However, most of the methods mentioned above are deployed on servers, which adds to the burden on servers that are already handling a large volume of responses, exacerbating processing delays and potential downtime. In the face of high-speed traffic requirements, transferring attack detection tasks from end hosts [12] to the data forwarding planes [13] is an emerging trend. P4 programmable switches with ASIC chips provide an opportunity for this strategy due to their line-speed and flexible processing capabilities.

Existing methods that use P4 switches to detect DRDoS attacks have some limits. N ETHCF [14] filters IP spoofing packets based on hop count. This method has the defects of low detection accuracy and high communication overhead between the data plane and the control plane. DIDA [15] is completely located in the data plane with the cost of deployment on multiple edge routers which has high requirements on the underlying equipment and causes a lot of detection delay. To this end, we present FAPM, a DRDoS attack detection and mitigation system, as a light-weight deployment solution with the P4 data plane. The "latter window assisting former window" technique accumulates the packet length of the bi-directional flow on the edge switch of the victim end. The amplification factor is calculated and uploaded using a calculator designed specifically for this scenario. According to the uploaded information, the control plane can determine whether there is a fake amplification phenomenon brought on by DRDoS attacks, that is, whether there is a large reflection factor between the packet length of the bi-directional flow. The main flow collection and feature calculation work is deployed in the data plane, and the more complex judgment and decision-making work is completed by the control plane. The system offers reduced communication overhead and low resource usage while ensuring the response effect due to its clever design and implementation details.

The main contributions of this paper are summarized as follows:

- We deploy the main part of FAPM on the data plane, which introduces the "latter window assisting former window" storage mechanism and amplification factor calculator to collect and calculate reflection features rapidly.
- We retain the decision-making rights for the controller, which guides the switch to install mitigation strategies when abnormal features are identified, and dynamically adjust the window size to ensure the performance and resilience of the system.
- We carry out experiments in the BMv2 [16] software switch, define a metric to evaluate attack detection and mitigation systems, and verify the necessity and superiority of the detailed design. The results show that FAPM has a small occupation of SRAM and TCAM resources in the switch, and has low average detection and mitigation delays which is 5.23s.

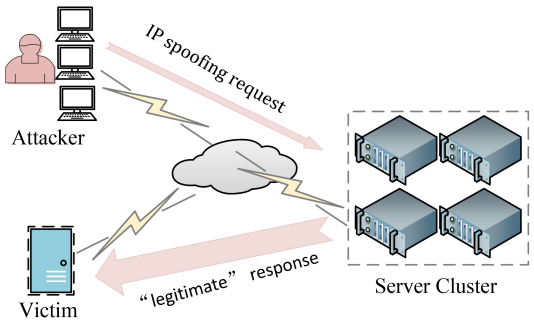


Fig. 1. Principle of DRDoS attack.

The remainder of this paper is organized as follows. Section II provides background on DRDoS attacks and P4. Section III details the design and deployment of the FAPM system. Section IV presents extensive experiments validating the performance of FAPM. Section V discusses related work, and Section VI summarizes the contributions of this paper.

II. Background

In this section, we introduce the threat model of DRDoS attacks, review existing methods, and provide an overview of P4 and programmable data planes. Furthermore, we clarify the challenges associated with implementing the FAPM system.

A. Threat Model and Existing Methods

DRDoS attacks are also known as amplification attacks due to the caused amplification effect. For example, if a user sends a MONLIST request to an NTP server, it will encapsulate the 600 most recently synchronized IP addresses into 100 response packets and send them back. As shown in Fig. 1, attackers use port scanning to find servers with open services as reflectors, then send spoofed requests from zombie hosts to these servers, which in turn flood the victim host with response packets. This is so that the server will not verify the sender after receiving the request since DRDoS attacks frequently employ UDP services whose response packets vastly outnumber the request packets. For the victim host, the response packets reflected by each server are pouring in, which will soon crush its link. The superiority of DRDoS attacks stems from the fact that the vast volume of data flow originally used for the attack is diluted via hundreds of servers before converging into a tremendous flood at the target, making it as difficult to be isolated and blocked as a slow-rate attack [17], [18]. Also, the forged source IP makes it more challenging to track down and find the attacker. The application layer services commonly exploited in amplification attacks are summarized in Table I.

We categorize existing detection methods for DRDoS attacks in Table II. The methods performed on servers can be divided into two categories. The first type, represented by RALF [8] and WisdomSDN [9], establishes mapping relationships between requests and responses to filter out incorrectly matched response packets. Because of the limitation of cache and the use of empirical knowledge, this approach is protocol-related and can only detect amplification attacks on known protocols. Hence, such techniques will be wholly invalid when hackers adopt a protocol that is not included. The second kind of method focuses on the authentication of the packet, diagnosing the

TABLE I
APPLICATION LAYER PROTOCOLS USED IN AMPLIFICATION ATTACKS

Application Protocol	Port	Application Factor	Exploited Functionality
NTP	123	556.9	monlist command
DNS	53	28-54	query type=ANY
LDAP	389	46-55	searchRequest command
MSSQL	1434	25	abuse of MC-SQLR
NetBIOS	137	2.56-3.85	name resolution query
SNMP	161	6.3	GetBulk request
SSDP	1900	30.8	SEARCH request
TFTP	69	60	block and retransmission mechanism

TABLE II
COMPARISON WITH PRIOR WORK

Prior Works	Protocol-independent	Line-speed	Light-weight deployment
RALF [8], WisdomSDN [9]	✗	✗	✗
ESFD [10], HCF [11]	✓	✗	✗
NetHCF [14], DIDA [15]	✓	✓	✗
FAPM	✓	✓	✓

source validity of the request packet through the hop count [11] and time stamp information [10]. However, these methods deployed on servers increase the functional complexity of server kernels, introducing higher packet processing latency.

NETHCF [14] and DIDA [15] are deployed on programmable switches with line-speed processing capabilities to detect DRDoS attacks in high-speed traffic forwarding scenarios. NETHCF [14] implements hop-count-based filtering for IP deception packets in P4 switches, but this requires maintaining a mapping table of the recently used IP addresses and hops on the data plane. The control plane holds the complete table and updates the cache of the data plane frequently. If the hop count of a packet is inconsistent with the record or missing in the mapping table, the packet needs to be mirrored to the control plane for further processing, which may cause pressure on the control channel. DIDA [15] is a distributed in-network DRDoS attack detection system that needs the collaboration of multiple edge routers and access routers in ISPs in order to share information and find unsolicited response packets. It places significant demands on bottom-layer devices and causes substantial detection delays due to the exchange of information between switches.

B. Programmable Data Plane and P4

Traditional network chips from major manufacturers are inflexible, with fixed functionalities and inconsistent specifications. Currently, ASIC chips offer flexible programming with Tbps throughput, allowing network operators to extend network core functionalities using the P4 programming language [19]. The Portable Switch Architecture (PSA) enables other manufacturers to manufacture chips that support P4 programming [20], [21]. As shown in Fig. 2(a), manufacturers must provide PSA definitions and P4 compilers, while users interact only with the P4-written programs.

The abstract forwarding logic provided by P4 is depicted in Fig. 2(b). Network functions are expressed in this abstract logic and programmed into the data plane using P4. The parser extracts packet header information and determines the next state based on specific fields, enabling layer-by-layer parsing. The switch performs actions based on the Match-Action (M-A) table. The deparser reverses this process by repackaging headers for transmission. A buffer between the ingress and egress pipelines stores packets awaiting processing, allowing for packet replication and queue management [22].

C. Challenges

The high throughput of P4 switches is ensured under strict limitations. Specifically, this presents three challenges for the implementation of FAPM.

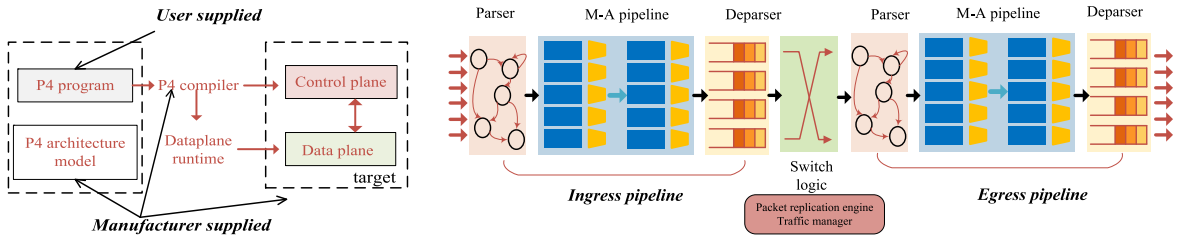
- Optimized feature calculation within a constrained instruction set. Even if there are many effective features, the M-A abstraction is insufficient for complex calculations [21]. The computational model of the P4 switch is extremely constrained and does not support loops or multiplication and division operations, which limits the combination of multidimensional complex features. Choosing simple features and adjusting the computation method are the implementation strategies of FAPM.
- Stateful flow storage and in-time updating. Attack detection relies on flow characteristics, necessitating stateful per-flow information storage. However, there are only 10+MB of SRAM resources and limited TCAM in a pipeline. Considering the large number of network flows, we need to segment time windows for statistics and achieve real-time updates of statistical information within the constraints of register read-write operations.
- Efficient coordination with the control plane. Although the switch can process packets at line rate, migrating traffic to external devices for further analysis inevitably causes a forwarding bottleneck. Therefore, avoiding traffic migration while collaborating with the controller to implement an efficient and dynamic detection system is a crucial consideration in the design of FAPM.

III. DESIGN AND DEPLOYMENT

A prominent feature of DRDoS attacks is the imbalance between a large number of reflection packets and request packets. We choose to install the detection system at the victim end to track this amplification occurrence in order to increase its deployment flexibility. In order to adapt to the pipeline architecture of the programmable data plane, we design the storage mechanism of the "latter window assisting former window" and the amplification factor calculator to complete the extraction and calculation of reflection features. The controller is in charge of result evaluation, state transition, and dynamic parameter modification, all of which call for more sophisticated and complex logic. Furthermore, due to our design, communication between the data plane and the control plane will not be overly burdensome.

A. Fake Amplification Phenomenon

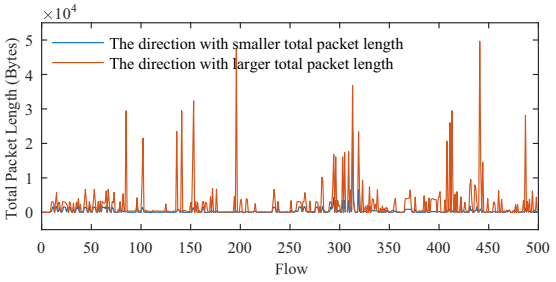
Normal host communication typically involves two-way message transmission between two hosts. However, during



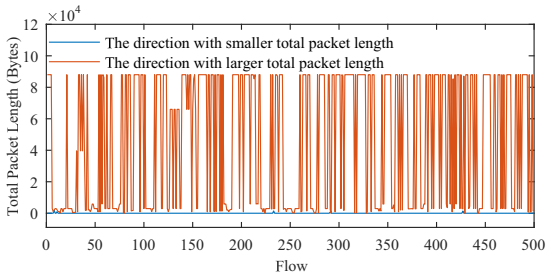
(a) P4 programming workflow.

(b) The abstract forwarding logic of P4.

Fig. 2. P4 overview.



(a) Benign traffic of CIC-IDS2017 [23].



(b) DRDoS-DNS attack traffic of CIC-2019DDoS [24].

Fig. 3. Total packet length comparisons of bi-directional flows.

a DRDoS attack, a large number of response packets from servers converge on the victim network, even though the victim host is not accessing services from these servers. Therefore, a large number of one-way transmission packets can be sniffed in the victim network. Although the victim may occasionally request services from some servers, the number of response packets far exceeds the requests. We call this phenomenon observed during attacks "fake amplification phenomenon". "Amplification" describes a situation in which the victim transmits few packets or no packets at all, but the server side sends out a lot of packets. "Fake" means that while the amplification attribute of a DRDoS attack involves both the intermediate servers and the attackers, the amplification referred to here is an indirect amplification relationship analyzed from the victim's perspective.

A bi-directional flow is defined as follows in this paper. The packets belonging to the same bi-directional flow meet three conditions: (1) the source IP and destination IP belong to the same set $\{IP_A, IP_B\}$ (2) the source port and destination port belong to the same set $\{Port_A, Port_B\}$ (3) the transport protocol is the same. For example, if (source IP, destination IP, source port, destination port, transport protocol) forms a 5-tuples of flow ID, then the packets whose 5-tuples are $(IP_A, IP_B, Port_A, Port_B, TCP)$ or $(IP_B, IP_A, Port_B, Port_A, TCP)$ belong to the same bi-directional flow. Further, specify that a uni-directional flow whose source IP field value is less than the destination IP field value is a forward flow of the two ones in a set of bi-directional flow, and the other uni-directional flow is a backward flow. When the hosts in the network communicate normally with each other, the total packet payload difference between the two of a set of bi-directional flows will not be particularly large. For victims of DRDoS attacks, there may be only one uni-directional flow transmitting packets (responses reflected by the server). Or even if the flow in both directions has data streaming, the total payload of packets is also extremely unbalanced, and one direction occupies absolute dominance. Fig. 3 is a comparison of the total packet length of 500 groups of bi-directional flows under benign situations and DRDoS-DNS attacks. In a normal network, the load of the flow in the two directions is close, and there is no clear amplification relationship. However, under DRDoS attacks, the packet load of one direction is overwhelmingly larger than the one of the other direction, and there is an obvious amplification relationship. Therefore, the fake amplification phenomenon observed on the victim side can be used as the judgment basis for the occurrence of DRDoS attacks.

To detect this fake amplification phenomenon at the victim end, we design FAPM and deploy it on the edge P4 switch. Specific design details will be discussed later. Compared to deploying the detection method on the victim server, which increases kernel processing complexity, deployment on the P4 switch offers benefits such as lower hardware latency and jitter, and faster attack response. It also eliminates the need for redundant strategies on each server, saving computing and storage resources.

B. Modules of FAPM in the Data Plane

The advantage of the data plane lies in the low latency of the hardware and the fine-grained information collection through direct contact with the packets. Based on the properties of each packet processing, we deploy the feature extraction in attack detection on the data plane. With the traffic window as the detection unit, the switch counts the total length of packets in both directions of each bi-directional flow and calculates the corresponding amplification factor, and reports it to the controller as the characteristic basis for attack judgment. A traffic window is composed of a fixed-time window and two fixed-number windows. The mechanism of "latter window assisting former window" is adopted to store the packet length of the flows. The related notations are summarized in Table III. When calculating the amplification factor, taking into account the variable division not supported by P4, we implement an

TABLE III
NOTATIONS USED IN "LATTER WINDOW ASSISTING FORMER WINDOW"

Notation	Definition
IP_1, IP_2	$IP_1, IP_2 \in$ {source IP, destination IP} The field value of $IP_1 <$ the field value of IP_2.
$Port_1, Port_2$	$Port_1, Port_2 \in$ {source Port, destination Port} The field value of $Port_1 <$ the field value of $Port_2$.
W_T	The sub-window with fixed duration time T.
W_{N1}, W_{N2}	The sub-window with N packets.
S, S'	Two storage sketches swapping roles.
$index_1$-$index_K$	Additional registers to store length saving pointers.
K	The number of hash algorithms and the number of register groups.
N	The length of the sketch.
L	Each register unit stores $2L$-long bits.

operation framework suitable for this scenario with the help of shift operation.

1) "Latter Window Assisting Former Window": Because a flow ID might take on a wide range of IP addresses and port numbers, on the premise that P4 supports a long list of pre-defined CRC algorithms, we hash the 5-tuple of the flow ID to aggregate the packet length of the same flow to the same storage unit. The count-min sketch [25] can effectively reduce the counting error caused by hash collision, mainly by increasing the number of hash functions and counting cells. The minimum count among the cells mapped by these functions approximates the true count. Set up a set of registers with size $K \times N$. In order to map each packet of a bi-directional flow to the same location, take (IP_1, IP_2, $Port_1$, $Port_2$, *Protocol*) as the input of the hash functions. As shown in Fig. 4, the switch extracts the 5-tuple of a packet and determines the K storage locations of the bi-directional flow. The IP field value is then compared to determine the precise direction, and the packet length is then stored in either the high L bits (forward) or low L bits (backward) of each unit.

In order to monitor traffic in real-time, the switch counts and reports information in units of windows. One problem introduced is at the end of each window, how to report the statistical information in the window to the controller and clear the sketch for the next window. Performing these tasks at the end of each window in the pipeline of the final packet is impractical due to the excessive workload, which can disrupt high-speed packet forwarding and cause performance oscillations. Additionally, assigning such a heavy task solely to the last packet violates fairness principles.

To avoid task concentration, this paper delays the processing of statistical data from one window to the next. At the end of a window, tasks are not completed immediately but are deferred to the next window. Because statistical data needs to be retained for one more window, two sets of storage structures S and S' are actually required. At any given time, one structure retains the statistical data from the earlier window (referred to as the reserving sketch), while the other counts the packet length of bi-directional flows in the current window (referred to as the working sketch). At the end of each window, the two roles are swapped.

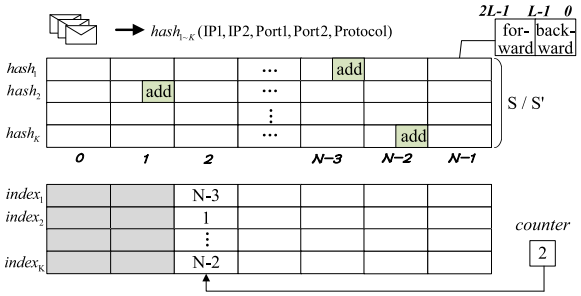


Fig. 4. Accumulation process of packet length.

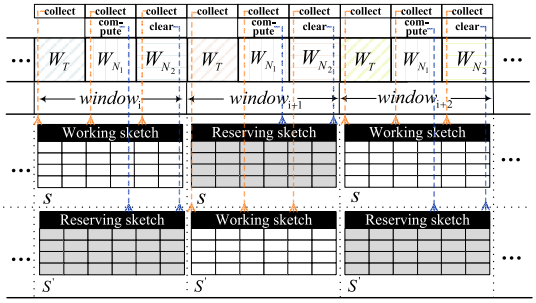


Fig. 5. Schematic diagram of the "latter window assisting former window" mechanism.

The arrival of the first packet of a new flow is indicated by all values at its K storage positions being zero. When a new flow starts, its corresponding K storage positions in the working sketch called the length saving pointers, also need to be recorded. As shown in Fig. 4, these pointers are stored in K additional registers $index_1$-$index_K$ whose length is N (assuming that the number of flows in a window does not exceed N). A counter tracks the number of flows, incrementing by 1 for each new flow index stored. In the next window, when the working sketch is converted to the reserving sketch, a column of $index_1$-$index_K$ can indicate the K positions where the packet length of the same flow is stored, which is convenient to locate and find the minimum count value for further amplification calculation.

Different from the general way, a detection window of FAPM is composed of W_T (fixed duration time T), W_{N1} (N packets), and W_{N2} (N packets), as shown in Fig. 5. The same work in each packet pipeline in the three sub-windows is to store the packet length in the working sketch as above and record the length saving pointers for each flow in $index_1$-$index_K$. The difference is that additional calculation and cleaning of the data in the reserving sketch will be performed within the two fixed-number windows W_{N1}, W_{N2}. Specifically, in each packet pipeline of W_{N1}, the switch examines the value α of the counter. If α is greater than or equal to 0, it retrieves the length saving pointers of a flow from the α_{th} column of $index_1$-$index_K$. It then reads the packet lengths from the corresponding K positions in the reserving sketch, selecting the minimum value as the final packet length for that flow. The amplification factor calculator, introduced later, is used to compute the packet

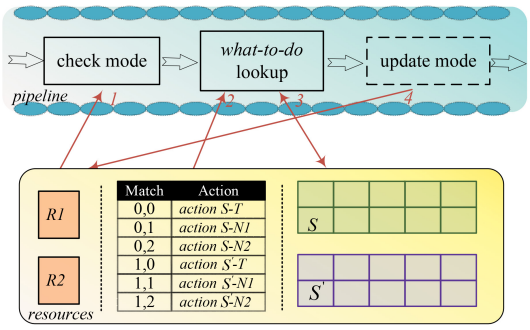


Fig. 6. The workflow of the "latter window assisting former window".

length amplification factor between the forward and backward flows. Finally, α is decremented by 1. Therefore, at the end of the N packets of W_{N1}, the information left by the previous window has been calculated and can be reported using the cloning mechanism in P4.

Immediately after that, in each packet pipeline of W_{N2}, the register units in the reserving sketch are cleared column by column. If the arrival order of a packet is β, clear the β_{th} column in the reserving sketch, that is, make the value stored be zero. As the number of fixed packets in the window is equal to the length of the sketch, which is N, all cells in the reserving sketch are just cleared at the end of W_{N2}. The task is decomposed and distributed to each packet pipeline by delaying information in one window to the next. This storage and window operation mechanism is referred to as the "latter window assisting former window".

Due to the alternating roles of two storage structures (S, S') and three sub-windows (W_T, W_{N1}, W_{N2}), there are six packet processing modes. Register $R1$ tracks the current storage structure (0: S is the working sketch, 1: S' is the working sketch), and $R2$ tracks the window state (0: W_T, 1: W_{N1}, 2: W_{N2}). The above-mentioned tasks are converged into actions in P4. For example, the *action S_N1* indicates that the current working sketch is S and that it is located in W_{N1}. Therefore, in the current packet pipeline, the switch should record its packet length in S and assist the former window in calculating the amplification factor of one flow left in the reserving sketch. By querying the table *what-to-do* with the values of $R1$ and $R2$, the switch can specify the processing mode for each packet. $R2$ is updated at the end of W_T or W_{N1}, and both $R1$ and $R2$ are updated at the end of W_{N2}, maintaining the six processing modes. Fig. 6 illustrates the packet workflow in the "latter window assisting former window" mechanism.

2) Amplification Factor Calculator: To calculate the reflection factor of packet length in the former window, the direct approach would be to divide the accumulated packet lengths of forward and backward flows. However, due to the lack of division support in programmable switch hardware, this method is not feasible. Our primary goal is not to obtain exact values but to differentiate between normal and attack flows. Fortunately, there is a significant difference in the reflection factor between these two cases, allowing us to calculate a fuzzy amplification factor under restricted conditions while still achieving effective discrimination.

The numbers in P4 are represented as bitstrings. As detailed in Algorithm 1, assuming x and y are the packet length (32bits)

Algorithm 1: Amplification Factor Calculator

Input: Two integers of 32 bits x, y
Output: The amplification factor of x and y

$Index_H \leftarrow \max(Top1\_index(x), Top1\_index(y))$;
$Index_L \leftarrow \min(Top1\_index(x), Top1\_index(y))$;
if $Index_H - Index_L \leq 10$ **then**
  return $Index_H - Index_L$ ▷ Amplification factor
else
  return 10 ▷ Limit the maximum factor to 2^{10}
Function `Top1_index(a)` ▷ a is a 32-bit integer
  $n \leftarrow 1$; ▷ Count the number of leading zeros
  if $a == 0$ **then return** -1 ;
  if $a \gg 16 = 0$ **then** $n \leftarrow n + 16$; $a \leftarrow a \ll 16$;
  if $a \gg 24 = 0$ **then** $n \leftarrow n + 8$; $a \leftarrow a \ll 8$;
  if $a \gg 28 = 0$ **then** $n \leftarrow n + 4$; $a \leftarrow a \ll 4$;
  if $a \gg 30 = 0$ **then** $n \leftarrow n + 2$; $a \leftarrow a \ll 2$;
  $n \leftarrow n - (a \gg 31)$;
  return $31 - n$

of the forward and backward flows, only the position of the highest bit 1 is relevant, while the other lower bits are ignored. Then the amplification factor can be expressed as an exponent of 2, with the exponent being the difference between the index of the highest bit 1 in x and y. For example, the amplification factor for 111 (7D) and 110 (6D) is 2^0, while it is 2^4 for 100000001 (257D) and 11110 (30D). If the index difference exceeds 10, it is capped at 10, indicating that a 2^{10} amplification factor is sufficient to embody the reflection effect. Because loop logic is not supported in P4, the highest bit 1 cannot be found by iterative shifting. Instead, a binary search approach is used to determine whether the highest 1 is in the high 16 bits or the low 16 bits, and subsequently in the high 8 bits or the low 8 bits of those 16 bits. This process is repeated until the highest 1 is located.

Although the results calculated in this manner have a significant inaccuracy compared to those obtained using a division ALU, it does not affect the distinction of features. This is because the amplification factors in normal and attack states differ significantly in magnitude, similar to the difference between 2 and 200 rather than 2 and 4. A set of registers $\{0_r, 1_r, 2_r, 3_r, 4_r, 5_r, 6_r, 7_r, 8_r, 9_r, 10_r\}$ is used, where the subscript r denotes the amplification factor in exponential form (base 2). Every time a factor is computed, the value in the corresponding position of the register is incremented by 1. When finished, the register stores the number of flows for each amplification factor. This information is then encapsulated in cloned packets and reported to the control plane, completing the extraction and computation of reflection features. Our amplification factor calculation algorithm simplifies operation and reduces reported payloads. Instead of providing exact numerical values, it categorizes the factor into 11 discrete classes from 0_r to 10_r, compressing the reported information. With these 11 values, we can assess window states and detect reflection attacks effectively.

We compare other division implementation methods in Table IV. A common approach for implementing division

TABLE IV
COMPARISON OF FLEXIBLE PACKET PROCESSING, P4ENTROPY AND AMPLIFICATION FACTOR CALCULATOR (Y FOR YES, N FOR NO)

	Flexible packet processing [26]	P4Entropy [27]	Amplification factor calcutor
TCAM consumption	Y	N	N
High relative error	N	N	Y
Numerous ALU instructions	N	Y	N
Control plane's participation	Y	N	N
Complex logic	N	Y	N

indirectly is through logarithmic and exponential operations, as $A/B = exp(\log(A) - \log(B))$. However, P4 lacks support for these operations. One method [26] involves using lookup tables for precomputed results, which consumes significant TCAM resources. Another method [27] avoids TCAM by decomposing the computation into integer and decimal parts, but it requires more complex calculations. Both methods can achieve less than 1% relative error when parameters are set correctly, which our method does not achieve this level of precision. However, high precision is not crucial for amplification factor calculations, allowing for some flexibility in accuracy. In other words, the amplification factor calculator can be regarded as a division tool specially tailored for this case.

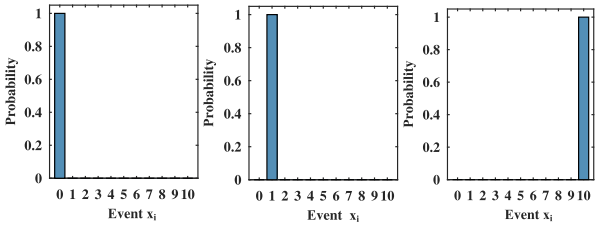
C. Modules of FAPM in the Control Plane

As the control plane has access to more abundant storage and computing resources, it is responsible for adjusting the system according to the dynamic changes in the network, as well as making correct state judgments and transitions.

1) Accuracy: Anomaly Checking: When the controller receives window information reported by the data plane, it must be able to make a quick and accurate judgment on whether an attack has occurred, which is crucial for effective mitigation. One idea is to assign scores from 0_r to 10_r in ascending order and multiply them by the corresponding flow count to obtain a total score which is used to judge the window status. However, this approach will undoubtedly be affected by the flow count in the window, which means a window containing many low-factor flows will have a score similar to the window with only one high-factor flow. Additionally, it is not easy to regulate the exact scores of different amplification factors. To this end, a normalization operation is first performed to calculate the proportion of flow counts corresponding to each amplification factor relative to the total number of flows in the window. Consider "the amplification factor of the flow is i_r" as a random event x_i, and $P(x_i)$ can be regarded as the probability of occurrence of the event. Then, the information reported in a window can be transformed into a probability distribution form $\{P(x_i)\}$, i=0,1,2,3,4,5,6,7,8,9,10 and $\sum_{i=0}^{10} P(x_i) = 1$.

Kullback-Leibler (KL) divergence [28] is a common method for measuring the similarity between probability distributions. It is computed using Eq. (1):

$$D_{KL}(P\|Q) = \sum_i P(x_i) \log \frac{1}{Q(x_i)} - \sum_i P(x_i) \log \frac{1}{P(x_i)} \quad (1)$$



(a) $P1$. (b) $P2$. (c) $P3$.

Fig. 7. Three typical probability distributions.

KL divergence reflects the similarity between probability distributions P and Q by calculating the increased length when coding Q using the code of P. This method ignores the geometric properties of the probability distributions. However, the differences between events such as x_0 and x_1, x_0 and x_{10} should not be considered as consistent.

The Wasserstein distance [29], also known as the Earth Mover's Distance (EMD), measures the similarity between probability distributions by depicting them as piles of earth and calculating the minimum cost to move one set of piles into the other. Let d_{ij} represent the distance from the i_{th} pile of P to the j_{th} pile of Q, which is the known cost, and let f_{ij} represent the amount of earth moved from the i_{th} pile of P to the j_{th} pile of Q, which is the variable to be optimized. Thus, EMD is the minimum cost of moving P to Q expressed by Eq. (2) by finding an optimal solution in all possible F:

$$D_{EMD}(P\|Q) = \frac{inf}{F = [f_{ij}]} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij} f_{ij} \right) \quad (2)$$

If the distance matrix is set as $d_{ij} = |i - j|$, with the cost of movement being positively correlated with the difference between the amplification factors, the KL divergence and EMD are calculated for $P1$ and $P2$, as well as for $P1$ and $P3$, based on three typical probability distributions shown in Fig. 7. The results are $D_{KL}(P1\|P2) = 10$, $D_{KL}(P1\|P3) = 10$, $D_{EMD}(P1\|P2) = 1$ and $D_{EMD}(P1\|P3) = 10$. As can be seen, the actual distribution of $P2$ is closer to $P1$, and $P3$ is farther away from $P1$, but the KL divergence measure cannot reflect this characteristic. Therefore, in this paper, EMD is used to measure the similarity of the reflection characteristics of the window. The control plane calculates the probability distribution of the window after receiving 11 flow counts. Then, the EMD is calculated between the probability distribution of the current window and the representative probability distributions of the normal and attack windows which are trained in advance. If it is closer to the attack window, the window will be regarded as abnormal.

2) Lightness: State Transition and Mitigation: To minimize the impact on normal users, the controller does not immediately conclude a DRDoS attack upon receiving reflection features identified as abnormal. Instead, it treats the network state as suspicious. Upon detecting the first suspicious window, the controller instructs the switch to record the length saving pointers and the 5-tuple flow ID (IP_1, IP_2, $Port_1$, $Port_2$, *Protocol*) in the subsequent window. Unlike before, where information was reported only at the end of W_{N1}, the amplification factor and flow ID are now reported immediately

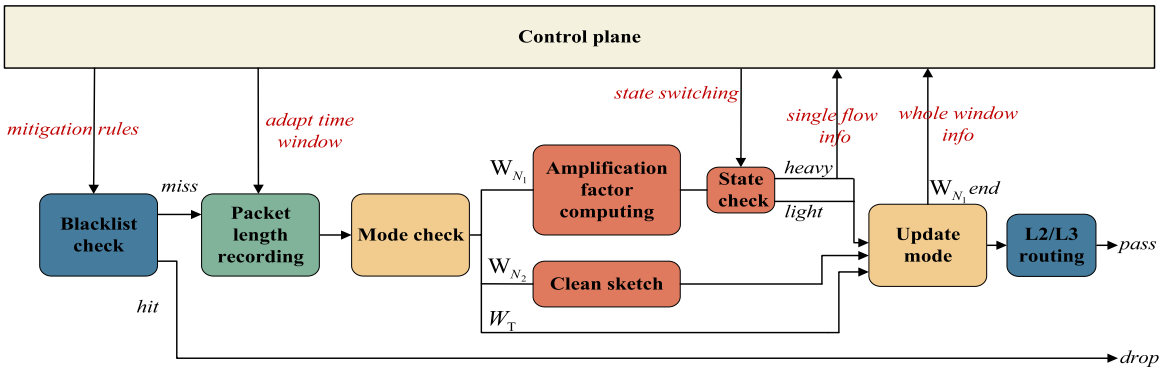


Fig. 8. The architecture of FAPM.

each time the factor is calculated, thereby increasing the workload of the switch.

As shown in Fig. 9, the controller uses the data reported by the switch to determine its state. Typically, the switch operates in a light-weight mode, handling basic statistics and reporting. If the controller detects an abnormal window, it instructs the switch to enter a heavy-weight mode and cache the reported information. After three consecutive abnormal distributions, the system confirms a DRDoS attack and initiates defense measures. Otherwise, the switch reverts to light-weight mode. The port with the highest occurrence is identified as the open service port, and the most frequent IP is identified as the victim host. The controller then sends a mitigation entry with this port and IP to the switch and instructs it to discard packets with matching source ports and destination IPs. The switch then immediately returns to light-weight mode.

This state transition mechanism allows the switch to remain in light-weight mode for fast packet processing. It only turns to heavy-weight mode during network fluctuations to monitor subsequent windows, then quickly returns to light-weight mode once the fluctuation or attack flow is resolved. This approach ensures that attack flow location and defense occur with minimal overall cost.

3) Flexibility: Dynamic Window: After the switch allocates the required registers for recording the packet length, generally this configuration will not be changed arbitrarily. However, changes in flow rate can affect system performance. Specifically, as the flow rate increases, more flows in a window heighten hash collision probability, reducing attack detection accuracy. To alleviate this negative impact, the size of the window W_T can be reduced. On the other hand, as the flow rate decreases, the number of flows in a window also decreases. To minimize communication overhead while ensuring counting accuracy, the size of W_T can be increased appropriately. Therefore, if the size of W_T can be flexibly adjusted based on the flow rate passing through the switch, the robustness of the system can be fundamentally guaranteed.

Assuming the length of the sketch is *N*, as mentioned earlier, we expect that a window will have at most *N* flows. Since the number of flows can reflect the flow rate, the control plane can calculate a total flow number f of a window after receiving the flow counts corresponding to 11 amplification factors. The two thresholds are TH_{high} and TH_{low}. When f reaches TH_{high}, it indicates a high flow rate and a high likelihood of hash collisions, which needs to cut the time window W_T in half. When f is less than TH_{low}, it means that the flow rate at this time is very low and collisions are not easy to occur, so W_T can be doubled to reduce the communication frequency. When f is between the two thresholds, the size of W_T is adjusted only when the difference between f and the total flow count f' of the previous window exceeds a certain value *range*. This is to avoid unnecessary frequent adjustments of the window size. The overall goal is to keep the total number of flows in a window stable within the range $[TH_{low}, TH_{high}]$. The formula for calculating the new window size T_{new} based on the current window size T_{curr} is as follows, where *trunc*() represents rounding towards zero:

$$T_{new} = \begin{cases} 2 \times T_{curr}, & f < TH_{low} \\ T_{curr} \times \left(1 - \frac{trunc\left(\frac{f-f'}{range}\right) \times range}{N}\right), & TH_{low} \le f \le TH_{high} \\ 0.5 \times T_{curr}, & f > TH_{high} \end{cases} \quad (3)$$

D. Overall Framework

Fig. 8 illustrates the system architecture of FAPM. The greatest advantage of the data plane is its hardware-enabled efficient packet processing capability, which allows it to quickly parse packets and extract length fields at a fine-grained level. Complex computing models that the data plane cannot directly execute are delegated to the intelligent control plane. By compressing window reflection features into 11 flow counts, communication bandwidth is conserved, enabling a deployment strategy that leverages the strengths of both planes.

IV. Experiment and Evaluation

Our experiments aim to answer several design and performance-related questions regarding FAPM, including:

- Does the design "latter window assisting former window" truly optimize its effectiveness?
- Can the time window be dynamically adjusted based on the flow rate to improve system performance?
- How can we evaluate its performance in the detection and mitigation of DRDoS attacks?

A. Experimental Setup

On a virtual machine (Ubuntu 20.04, Intel i5-7500, 16 GB mem), we build a simulation platform and run the topology shown in Fig. 11 on Mininet [30]. The switches used in this

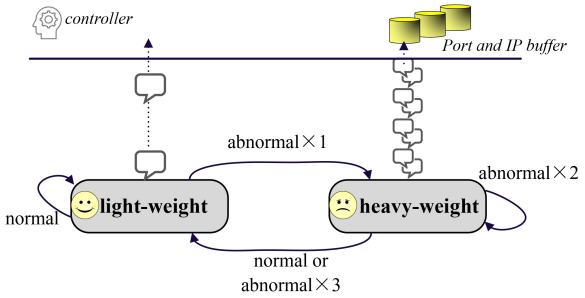


Fig. 9. State transition mechanism of FAPM.

topology are BMv2 [16]. The FAPM prototype[1] in the data plane is implemented with P4 and deployed on switch S3 at the victim end. All traffic destined for the victim will pass through this edge switch, which is connected to the control plane through P4Runtime.

B. "Latter Window Assisting Former Window"

We conduct two sets of experiments under the same conditions. The first set uses a typical window partition method, which calculates all the information and clears all registers in the last packet pipeline of the window. The second set uses the "latter window assisting former window" method designed in this paper, which distributes the work evenly across each packet pipeline. Fig. 12 shows the ingress and egress timestamps of packets in the switch under two scenarios. Accumulating a large amount of work until the last packet results in oscillations at the end of each window, with increasing delays in processing some packets, which affects network performance. In contrast, the design of this paper can effectively avoid these disadvantages, ensuring smooth and consistent packet forwarding.

C. Adjustment of Time Window

This experiment is carried out to test the function of FAPM in dynamically adjusting the time window to match the flow rate. As shown in Fig. 10(a), traffic is sent to the destination host with a bandwidth of 5Mbps, 10Mbps, 15Mbps, and 20Mbps respectively, each lasting for 15s. The initial time window W_T is set to 1s, with TH_{high} at 80 and TH_{low} at 20. The yellow line indicates that the window's duration can be adjusted to avoid the high hash collision rate as the flow speed continues to rise.

We make the comparison between opening and closing the time window tuning module on the control plane. As shown in Fig. 10(b), at low flow rates from 0-15s, there is no issue with hash collisions, so the window size is increased to reduce the reporting frequency and communication overhead. After 15s, as the flow counts in the reported window increase, the window size begins to shrink for the purpose of preventing hash collisions. At this time, the growth of communication costs has become a secondary concern.

Fig. 10(c) is a bubble chart with the horizontal axis as the number of flows and the vertical axis as the number of hash collisions in a window. The size of the bubble is positively correlated with the length of the time window. Blue bubbles, indicating the tuning module is disabled, are uniformly sized (1s) and clustered in high collision areas. In contrast, yellow bubbles, with the tuning module enabled, vary in size and are predominantly found in lower collision areas. The tuning module effectively controls the number of flows in most windows within the range $[TH_{low}, TH_{high}]$, which is consistent with our design objective.

[1]Available at https://github.com/wxc397/FAPM.

TABLE V
THE COMPOSITION OF ATTACK TRAFFIC

Protocol	Number of Packets	Proportion (%)	Protocol	Number of Packets	Proportion (%)
NTP	403859	11.98	NetBIOS	698626	20.73
DNS	464582	13.79	SNMP	208532	6.19
LDAP	464118	13.77	SSDP	472600	14.02
MSSQL	308970	9.17	TFTP	348694	10.35

TABLE VI
THE DETECTION METRICS UNDER THREE DISTANCE MATRICES ("ACC" STANDS FOR ACCURACY, AND "PRE" STANDS FOR PRECISION)

	$d_{ij}=\lvert i-j\rvert$			$d_{ij}=2^{\lvert i-j\rvert}$			$d_{ij}=\lvert i-j\rvert^2$		
	Acc(%)	Pre(%)	Recall(%)	Acc(%)	Pre(%)	Recall(%)	Acc(%)	Pre(%)	Recall(%)
K1	73.55	63.69	99.26	82.41	72.40	100.00	79.51	69.30	99.80
K2	94.54	89.41	100.00	95.93	91.89	100.00	95.52	91.16	100.00
K3	83.27	97.89	65.15	90.09	99.16	79.20	84.72	99.02	67.56
K4	83.06	98.36	64.35	91.05	99.26	81.20	87.07	99.09	72.64
K5	91.03	98.40	81.92	97.32	99.16	95.00	95.04	98.90	90.26
Avg	**85.09**	**89.55**	**82.14**	**91.36**	**92.37**	**91.08**	**88.37**	**91.49**	**86.05**

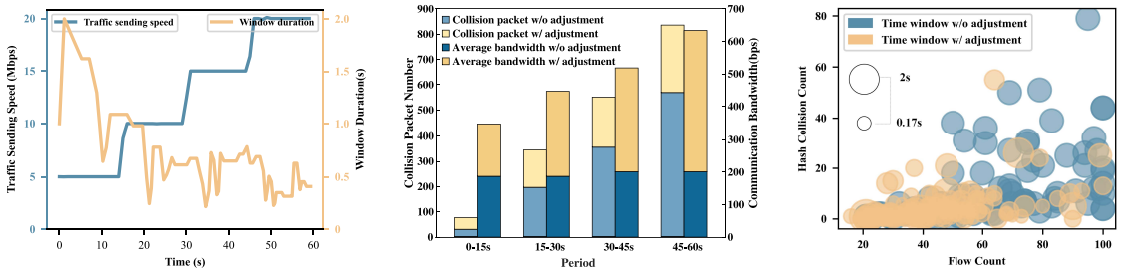
D. Attack Detection

CIC-DDoS2019 [24] is a DRDoS attack traffic dataset that contains multiple protocols. In this paper, we replay a portion of this dataset (3369981 packets, as detailed in Table V) and benign traffic (4500000 packets) in experiments conducted with a window size of W_T=0.1s, W_{N1}=W_{N2}=100, and two hash functions. Thus, we obtain the probability distributions related to the reflection feature for each window.

Set the distance metric in k-means to EMD for training. As shown in Table VI, cross-validation is conducted under three distance matrices: $d_{ij} = |i-j|$ which is linear, $d_{ij} = 2^{|i-j|}$ which is exponential, and $d_{ij} = |i-j|^2$ which is square. The results demonstrate that the exponential distance matrix yielded the best performance, achieving the highest detection effectiveness among the five experiments. The average detection accuracy is 91.36%, the average precision is 92.37%, and the average recall is 91.08%. Fig. 13 illustrates the obtained clustering centers, where the amplification factors of attack windows are concentrated at higher positions, while the amplification factors of benign windows are dispersed at lower positions.

E. Attack Mitigation

1) Metric Definition: In order to evaluate the overall level of an attack mitigation system, we define an evaluation metric called mitigation efforts (ME). T_0 represents the time when the attack begins, T_1 indicates the moment when the system



(a) Window size adjusted with flow rate. (b) Comparison of collision counts and average bandwidth. (c) Effectiveness of time window adjustment.

Fig. 10. Opening tuning module vs. closing tuning module of time window.

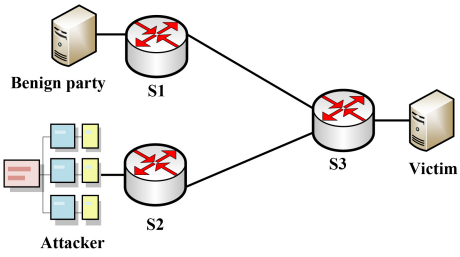
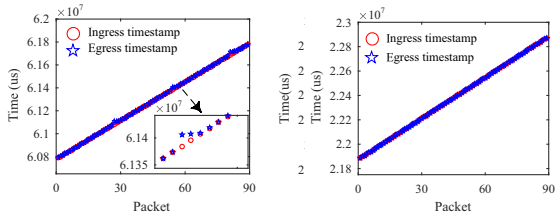
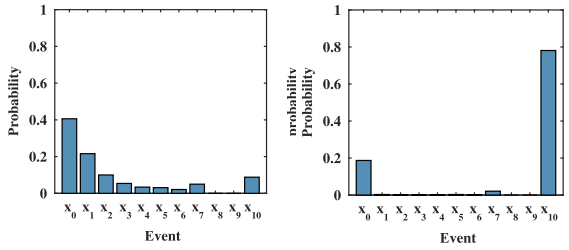


Fig. 11. Topology of the experiment.



(a) General windowing approach. (b) Mechanism of "latter window assisting former window".

Fig. 12. The ingress and egress timestamps of packets.

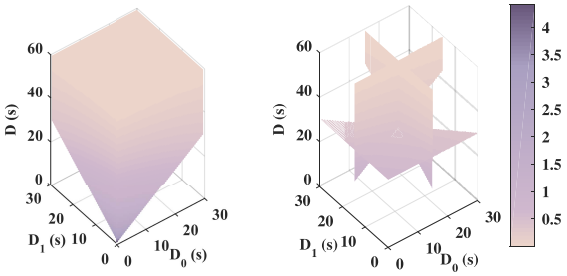


(a) Benign window. (b) DRDoS attack window.

Fig. 13. Clustering centers of probability distributions.

detects an abnormality, which is the moment when the heavy-weight mode is first switched to in this paper, T_2 indicates the moment when the attack flow is discarded, and T_3 is the time when the available bandwidth is fully restored to its normal level. Then, $D = T_3 - T_1$ reflects the entire response time, $D_0 = T_1 - T_0$ reflects the sensitivity of attack detection, and $D_1 = T_2 - T_1$ reflects the efficiency of deploying mitigation rules. To directly evaluate an attack detection and mitigation system overall, we express *ME* using the following equation:

$$ME = -ln\left(\frac{D}{60}\right) \times \left(1 - \frac{D_0}{60}\right) \times \left(1 - \frac{D_1}{60}\right) \quad (4)$$



(a) 3D view. (b) Sectional view of $D_0 = 15$, $D_1 = 15$, $D = 30$.

Fig. 14. Value of *ME* with different D_0, D_1, D.

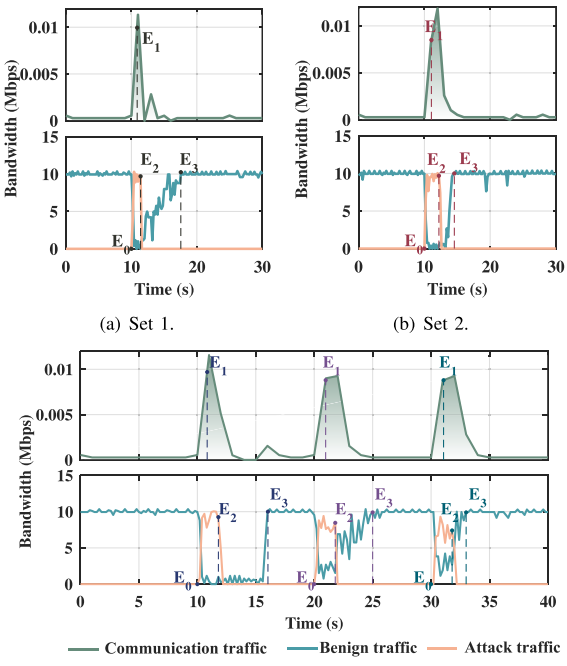
TABLE VII
TIME POINTS FOR MITIGATING CORRELATED EVENTS AND *ME* METRICS

	T_0 (s)	T_1 (s)	T_2 (s)	T_3 (s)	D_0 (s)	D_1 (s)	D (s)	*ME*
Set 1	10.00	10.88	11.40	17.55	0.88	0.52	7.55	2.02
Set 2	10.00	11.06	12.20	14.55	1.06	1.14	4.55	2.48
	10.00	10.84	11.80	16.04	0.84	0.96	6.04	2.23
Set 3	20.00	20.98	21.80	25.00	0.98	0.82	5.00	2.41
	30.00	31.07	31.80	33.00	1.07	0.73	3.00	2.90

If D is smaller and the proportions of D_0 and D_1 in D are smaller, *ME* is higher and the system performance is better. Moreover, when D exceeds 60s, the performance of the system is considered extremely poor, and the value of *ME* becomes negative. As shown in Fig. 14, systems with *ME*>2 can be considered good.

2) Performance Testing: We conduct three sets of experiments in total. The first set launches an amplification attack using only one specific service at 10s. The second set launches an amplification attack using multiple services simultaneously at 10s, which is uncommon in reality but possible. The third set launches attacks continuously at 10s, 20s, and 30s to test the stability of the system. Fig. 15 shows the traffic of the victim link and the communication traffic caused by the cloned packets reported by the data plane. $E_0 \sim E_3$ represent the corresponding events at $T_0 \sim T_3$. The results show that FAPM can quickly respond to and block attack flows, with extremely low communication overhead, and an average bandwidth of less than 1 Kbps.

The detailed numerical statistics are listed in Table VII, which shows that the *ME* of FAPM exceeds 2 under all different conditions, and D is still the dominant factor determining the system performance.



(a) Set 1. (b) Set 2.

(c) Set 3.

Fig. 15. Network traffic after deploying FAPM.

TABLE VIII
PERFORMANCE COMPARISON BETWEEN FAPM AND OTHER METHODS

Method	Resonse Time	Memory Consumption	Communication Overhead
NETHCF [14]	2.5s (Barefoot Tofino)	4MB	huge spoofing traffic
DIDA [15]	20s (BMv2)	150KB	550Kbps
FAPM	5.23s (BMv2)	KB level	1Kbps

3) Method Comparison: Compared with existing methods that use P4 data planes for DRDoS attack detection [14], [15] in Table VIII, the greatest advantage of FAPM lies in its lightweight deployment. The data plane only needs to use registers to record packet length and does not require caching a large amount of information. This greatly saves the valuable SRAM (tens of megabytes) and TCAM (a few megabytes) of the switch. Additionally, the compression of the window features reduces communication overhead to an extremely low level.

V. RELATED WORK

Sketch-based network measurement in P4 data plane. Network measurement based on sketches significantly contributes to attack detection, optimized scheduling, and intelligent operation and maintenance. Currently, significant research focuses on deploying advanced sketch counting structures in constrained programmable data planes, such as FCM-Sketch [31] and CocoSketch [32]. However, some real-time applications, such as real-time attack detection systems, do not belong to the landmark window models which need the overall network traffic data from the start time to the present moment. Instead, they are more concerned with sliding window data, which requires the sketches to be able to promptly update outdated cumulative data that is no longer meaningful.

One mainstream strategy is to adopt a partitioning strategy, such as maintaining a large sketch and small sketches [33] for different time intervals, or using a queue [34] to maintain a series of partitions working in the current window. However, implementing sketch segmentation and queue maintenance in P4 switches is a challenge. Another approach is to maintain a timestamp list [35] to precisely evict outdated packet statistics, but this imposes a significant memory burden on the switch. Moreover, this update mechanism operates asynchronously on a per-flow basis, which means that it cannot obtain all the statistics for a specific window at a single window-end point. The "latter window assisting former window" mechanism proposed in this paper adds a replicated sketch and index recording structure. It records the packet information of the current window while simultaneously calculating and clearing the complete data of the historical window. By introducing a one-window delay, this mechanism efficiently supports feature extraction and computation within the real-time window and report the features to the controller on a per-window basis.

Attack detection with P4 data plane. Many studies have explored deploying attack detection systems in P4 forwarding devices. One approach is to use machine learning models to implement an intelligent data plane for online inference, allowing for line-rate classification of benign and malicious traffic. Planter [36] and NetBeacon [37] have implemented tree-based models, but the calculation and number of statistical features these models rely on are constrained by the stage numbers and the computation capability of the switch. This significantly reduces the classification accuracy of the model. Brain-on-Switch [38] has successfully deployed RNN models by designing a binary activation function and sliding execution inference time steps. However, a small portion of traffic with low classification confidence needs to be forwarded to external servers for further analysis, resulting in forwarding delays.

Dedicated in-band detection systems emphasize rule-based detection and filtering. Jaqen [39] and POSEIDON [40] designed a series of monitoring and defense primitives to cope with dynamically changing DDoS attacks, but this requires significant changes to the switch configuration. SECAP [41] and N ETHCF [14] perform authentication in the data plane to verify legitimate connections, consuming substantial memory for rule storage. P4DDLe [42] and Euclid [43] collect and compute traffic features in the data plane. P4DDLe [42] needs to store many packet-level features for the classifier, while Euclid [43] uses IP entropy as a feature, requiring a large amount of SRAM for the longest prefix match table entries to complete the pre-computed function. Our solution only needs to record the packet lengths and perform simple shift operations to obtain reflection features. Additionally, it compresses the window information into 11 category features, reducing storage resource occupation and upload communication.

Deployment of DRDoS attack detection methods. Due to the unique nature of DRDoS attacks, their detection and mitigation strategies can be deployed at the source, intermediary, and victim ends. At the source end, a threshold-based detection method [44] on the SDN gateway filters request packets and samples traffic to identify attacks. Intermediary reflection servers can serve as deployment points, as they can determine the legitimacy of the request before generating

and delivering the response, as discussed in [10] and [11]. Unfortunately, due to the vast number of servers, implementing this method universally is challenging.

Deployment at the victim end is relatively flexible and convenient. A distributed system proposed in [45] is located at the edge switches and uses aggregated sketches to detect DRDoS attacks by examining request-response packet mappings. Although deployment on the victim side is flexible and on-demand, its reaction is relatively delayed compared to the source and intermediary sides. Additionally, it cannot prevent the bandwidth waste caused by reflected packets before they reach the target. Thus, some methods designed the architecture to detect DRDoS attacks in the border of an ISP network. In D^2AMA [46], a control agent communicates with the border routers associated with the affected subnet, enabling the shielding of attack packets through fuzzy association rules. DIDA [15] facilitates the exchange of request and response packet counts between access routers to identify DRDoS attacks. Access control lists are then applied and maintained on the routers to filter out malicious traffic. To respond earlier at the upstream switches and reduce the bandwidth consumption of attack traffic along the path, the FAPM system proposed in this paper can be enhanced in two ways. First, after the controller makes an attack determination, it notifies the upstream switches about the attack event and the attack source, prompting them to start discarding the malicious traffic. Second, the system can be deployed in a distributed manner across multiple ISP access points, which requires designing a specific information exchange protocol and meeting higher demands on the underlying devices.

VI. Conclusion

In this paper, we explore the advantages of fast and fine-grained programmable data planes and propose a light-weight DRDoS attack detection and defense system called FAPM. In the data plane, it collects the information and calculates features for the fake amplification phenomenon through a storage mechanism called "latter window assisting former window" and an amplification factor calculator. As for the control plane, it is responsible for decision-making and dynamic regulation. At the same time, FAPM ensures that communication between the two planes involves only a small number of cloned packets. Multiple sets of experiments verify the agility, flexibility, and low overhead of FAPM.

In the future, we aim to enhance FAPM to address a broader range of attack threats and assess its performance when deployed on hardware switches. Additionally, we will investigate the optimal balance between the data plane and control plane in implementing network functions to further refine the design.

References

[1] Y. Li, Q. Wang, F. Yang, and S. Su, "Traceback DRDoS attacks," *J. Inf. Comput. Sci.*, vol. 8, no. 1, pp. 94–111, 2011.

[2] K. Singh and A. Singh, "Memcached DDoS exploits: Operations, vulnerabilities, preventions and mitigations," in *Proc. IEEE 3rd Int. Conf. Comput., Commun. Security (ICCCS)*, 2018, pp. 171–179.

[3] J. Krupp, I. Grishchenko, and C. Rossow, "AmpFuzz: Fuzzing for amplification DDoS vulnerabilities," in *Proc. 31st USENIX Secur. Symp. (USENIX Secur.)*, 2022, pp. 1043–1060.

[4] X. Li, T. Liu, M. S. Obaidat, F. Wu, P. Vijayakumar, and N. Kumar, "A lightweight privacy-preserving authentication protocol for VANETs," *IEEE Syst. J.*, vol. 14, no. 3, pp. 3547–3557, Sep. 2020.

[5] Q. Liu, Y. Peng, J. Wu, T. Wang, and G. Wang, "Secure multi-keyword fuzzy searches with enhanced service quality in cloud computing," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 2046–2062, Jun. 2021.

[6] 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.

[7] X. Li, S. Liu, F. Wu, S. Kumari, and J. J. P. C. Rodrigues, "Privacy preserving data aggregation scheme for mobile edge computing assisted IoT applications," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4755–4763, Jun. 2019.

[8] P. M. Priya, V. Akilandeswari, and S. M. Shalinie, "Detecting DRDoS attack by log file based IP pairing mechanism," *WSEAS Trans. Comput.*, vol. 13, pp. 538–548, 2014.

[9] Z. A. El Houda, L. Khoukhi, and A. S. Hafid, "Bringing intelligence to software defined networks: Mitigating DDoS attacks," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2523–2535, Dec. 2020.

[10] L. Kavisankar, C. Chellappan, P. Sivasankar, A. Karthi, and A. Srinivas, "A pioneer scheme in the detection and defense of DrDoS attack involving spoofed flooding packets," *KSII Trans. Internet Inf. Syst.*, vol. 8, no. 5, pp. 1726–1743, 2014.

[11] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: An effective defense against spoofed DDoS traffic," in *Proc. 10th ACM Conf. Comput. Commun. Secur.*, 2003, pp. 30–41.

[12] D. Tang, X. Wang, X. Li, P. Vijayakumar, and N. Kumar, "AKN-FGD: Adaptive Kohonen network based fine-grained detection of LDoS attacks," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 273–287, Jan./Feb. 2023.

[13] D. Tang, S. Wang, B. Liu, W. Jin, and J. Zhang, "GASF-IPP: Detection and mitigation of LDoS attack in SDN," *IEEE Trans. Services Comput.*, vol. 16, no. 5, pp. 3373–3384, Sep./Oct. 2023.

[14] M. Zhang et al., "NetHCF: Filtering spoofed IP traffic with programmable switches," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 1641–1655, Mar./Apr. 2023.

[15] X. Z. Khooi, L. Csikor, D. M. Divakaran, and M. S. Kang, "DIDA: Distributed in-network defense architecture against amplified reflection DDoS attacks," in *Proc. 6th IEEE Conf. Netw. Softwarization (NetSoft)*, 2020, pp. 277–281.

[16] "Behavioral model version 2." Accessed: Feb. 23, 2022. [Online]. Available: https://github.com/p4lang/behavioral-model

[17] D. Tang, Y. Yan, C. Gao, W. Liang, and W. Jin, "LtRFT: Mitigate the low-rate data plane DDoS attack with learning-to-rank enabled flow tables," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 3143–3157, 2023.

[18] B. Liu, D. Tang, J. Chen, W. Liang, Y. Liu, and Q. Yang, "ERT-EDR: Online defense framework for TCP-targeted LDoS attacks in SDN," *Expert Syst. Appl.*, vol. 254, Nov. 2024, Art. no. 124356.

[19] P. Bosshart et al., "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

[20] S. Ibanez, G. Brebner, N. McKeown, and N. Zilberman, "The P4—>NetFPGA workflow for line-rate packet processing," in *Proc. ACM/SIGDA Int. Symp. Field-Programm. Gate Arrays*, 2019, pp. 1–9.

[21] P. Bosshart et al., "Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 99–110, 2013.

[22] A. Sivaraman, C. Kim, R. Krishnamoorthy, A. Dixit, and M. Budiu, "DC.p4: Programming the forwarding plane of a data-center switch," in *Proc. 1st ACM SIGCOMM Symp. Softw. Defin. Netw. Res.*, 2015, pp. 1–8.

[23] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, 2018, pp. 108–116.

[24] 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. Security Technol. (ICCST)*, 2019, pp. 1–8.

[25] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *J. Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[26] N. K. Sharma, A. Kaufmann, T. E. Anderson, A. Krishnamurthy, J. Nelson, and S. Peter, "Evaluating the power of flexible packet processing for network resource allocation," in *Proc. NSDI*, 2017, pp. 67–82.

[27] D. Ding, M. Savi, and D. Siracusa, "Estimating logarithmic and exponential functions to track network traffic entropy in P4," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, 2020, pp. 1–9.
[28] J. M. Joyce, "Kullback-Leibler divergence," in *International Encyclopedia of Statistical Science*. Berlin, Germany: Springer, 2011, pp. 720–722.
[29] S. S. Vallender, "Calculation of the wasserstein distance between probability distributions on the line," *Theory Probab. Appl.*, vol. 18, no. 4, pp. 784–786, 1974.
[30] "Mininet." Accessed: Feb. 23, 2022. [Online]. Available: http://mininet.org/
[31] C. H. Song, P. G. Kannan, B. K. H. Low, and M. C. Chan, "FCM-sketch: Generic network measurements with data plane support," in *Proc. 16th Int. Conf. Emerg. Netw. Exp. Technol.*, 2020, pp. 78–92.
[32] Y. Zhang et al., "CocoSketch: High-performance sketch-based measurement over arbitrary partial key query," in *Proc. ACM SIGCOMM Conf.*, 2021, pp. 207–222.
[33] S. Matusevych, A. Smola, and A. Ahmed, "Hokusai-sketching streams in real time," 2012, *arXiv:1210.4891*.
[34] R. Ben-Basat, G. Einziger, R. Friedman, and Y. Kassner, "Heavy hitters in streams and sliding windows," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2016, pp. 1–9.
[35] Y. Chabchoub and G. Heébrail, "Sliding HyperLogLog: Estimating cardinality in a data stream over a sliding window," in *Proc. IEEE Int. Conf. Data Min. Workshops*, 2010, pp. 1297–1303.
[36] C. Zheng et al., "Automating in-network machine learning," 2022, *arXiv:2205.08824*.
[37] G. Zhou, Z. Liu, C. Fu, Q. Li, and K. Xu, "An efficient design of intelligent network data plane," in *Proc. 32nd USENIX Security Symp. (USENIX Secur.)*, 2023, pp. 6203–6220.
[38] 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. Netw. Syst. Design Implement. (NSDI)*, 2024, pp. 419–440.
[39] Z. Liu et al., "Jaqen: A high-performance switch-native approach for detecting and mitigating volumetric DDoS attacks with programmable switches," in *Proc. 30th USENIX Secur. Symp. (USENIX Secur.)*, 2021, pp. 3829–3846.
[40] M. Zhang et al., "Poseidon: Mitigating volumetric DDoS attacks with programmable switches," in *Proc. 27th Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2020, pp. 1–18.
[41] D. Smyth, S. Scott-Hayward, V. Cionca, S. McSweeney, and D. O'Shea, "SECAP switch—Defeating topology poisoning attacks using P4 data planes," *J. Netw. Syst. Manage.*, vol. 31, no. 1, p. 28, 2023.
[42] R. Doriguzzi-Corin, L. A. D. Knob, L. Mendozzi, D. Siracusa, and M. Savi, "Introducing packet-level analysis in programmable data planes to advance network intrusion detection," *Comput. Netw.*, vol. 239, Feb. 2024, Art. no. 110162.
[43] A. da 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 Manag.*, vol. 18, no. 3, pp. 3121–3139, Sep. 2021.
[44] S.-N. Nguyen, V.-Q. Nguyen, G.-T. Nguyen, J. Kim, and K. Kim, "Source-side detection of DRDoS attack request with traffic-aware adaptive threshold," *IEICE Trans. Inf. Syst.*, vol. 101, no. 6, pp. 1686–1690, 2018.
[45] X. Jing, J. Zhao, Q. Zheng, Z. Yan, and W. Pedrycz, "A reversible sketch-based method for detecting and mitigating amplification attacks," *J. Netw. Comput. Appl.*, vol. 142, pp. 15–24, Sep. 2019.
[46] X. Yang, W. Yang, Y. Shi, and Y. Gong, "The detection and orientation method to DRDoS attack based on fuzzy association rules," *J. Commun. Comput.*, vol. 3, no. 8, pp. 1–10, 2006.

Dan Tang received the B.S., M.S., and Ph.D. degrees from the Huazhong University of Science and Technology in 2014. He is currently an Associate Professor with the College of Computer Science and Electronic Engineering, Hunan University. His research interests include network security, information security, and programmable network.

Xiaocai Wang received the B.E. degree in computer science and technology from Hunan University in 2022, where she is currently a Postgraduate with the College of Computer Science and Electronic Engineering. Her research directions include cyberspace security and programmable network.

Keqin Li (Fellow, IEEE) is a SUNY Distinguished Professor of Computer Science with the State University of New York and also a National Distinguished Professor with Hunan University. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficiency computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He is an AAIA Fellow. He is also a member of Academia Europaea.

Chao Yin received the B.S., M.S., and Ph.D. degrees from the Huazhong University of Science and Technology in 2014. He is currently an Associate Professor with the School of Computer and Big Data Science, Jiujiang University. His research interests include information security, big data, cloud storage, and erasure codes.

Wei Liang received the Ph.D. degree in computer science and technology from Hunan University in 2013. He was a Postdoctoral Scholar with Lehigh University, Bethlehem, PA, USA, from 2014 to 2016. He is currently a Professor with the School of Computer Science and Engineering, Hunan University of Science and Technology. His research interests include blockchain security technology, network security protection, embedded system and hardware IP protection, fog computing, and security management in wireless sensor networks.

Jiliang Zhang (Senior Member, IEEE) is currently a Full Professor with the College of Integrated Circuits, Hunan University. He is the Vice Dean of the College of Integrated Circuits, Hunan University. He has authored more than 80 technical papers in leading journals and conferences. His current research interests include hardware security, integrated circuit design, and intelligent system. He was the recipient of CCF Integrated Circuit Early Career Award, and the winner of Excellent Youth Fund of NSFC. He was a CCF Distinguished Speaker. He has been the Program Committee Member for a number of well-known conferences, such as DAC, ASP-DAC, GLSVLSI, and FPT. He is the Secretary-General of CCF Fault-Tolerant Computing Professional Committee.