Detection and Mitigation of Volumetric DDoS Attacks using Adaptive Rate-Limiting in P4-DPDK

Samia Choueiri, Ali Mazloum, Sergio Elizalde, Elie F. Kfoury, Jorge Crichigno
University of South Carolina, USA
{choueiri, amazloum, elizalds, ekfoury}@email.sc.edu, jcrichigno@cec.sc.edu

***Abstract*—This paper presents a high-performance, adaptive system for detecting and mitigating volumetric Distributed Denial-of-Service (DDoS) attacks using P4-DPDK. The proposed system operates entirely in the user space, leveraging multicore CPUs and SmartNICs to process traffic at line rate while enabling flexible and efficient control plane operations. DDoS detection is implemented in a linear prediction model that forecasts traffic based on historical throughput and dynamically adjusts rate-limiting thresholds. The system is evaluated on the FABRIC testbed using both synthetic and real-world MAWI traffic traces. Experimental results demonstrate robust attack mitigation across various attack types, adaptability to real traffic, and significantly lower packet loss compared to Suricata-DPDK implementations under high-throughput approaching 100 Gbps with multi-core parallelism. The findings affirm the system's scalability, sensitivity tuning via hyperparameter optimization, and its suitability for deployment in cloud-native and large-scale network environments.**

***Index Terms*—DDoS, DPDK P4, PNA, SmartNIC, Linear Prediction, Differential Evolution Optimization.**

I. Introduction

Network environments are progressively being targeted by large-scale Distributed Denial of Service (DDoS) attacks [1]. These attacks are conducted by malicious individuals who manage a network of compromised hosts. Over the recent years, there has been a significant increase in both the size and complexity of DDoS attacks, with traffic volume having doubled over the last year [1].

Current methods to mitigate these attacks fall into software and hardware-based categories. Software solutions are highly flexible and can be easily implemented across different networks, but they also encounter difficulties in managing high traffic volumes, particularly during DDoS events [2]. This limitation arises because they use the standard network stack, which results in significant latency due to the overhead associated with packet processing. Software solutions include Artificial Intelligence (AI) and Machine Learning (ML) techniques, which can analyze traffic patterns in real-time. However, these approaches often require training using specialized hardware and substantial computational resources, making them less practical for high-speed deployments [3].

On the other hand, hardware-based solutions like P4-programmable data plane switches or Field-Programmable Gate Arrays (FPGAs) offer speed advantages over software methods [4], [5]. However, they have several drawbacks; 1) They lack the flexibility necessary for complex packet processing logic. 2) They demand specialized skills for programming, operation, and maintenance, complicating their integration into networks largely dominated by fixed-function devices. 3) Their large-scale deployment can be costly, and they are often unsupported in cloud environments, which is problematic as many organizations have moved to the cloud.

A balanced approach is to leverage the flexibility and simplicity of deploying software solutions without significantly compromising performance. The Data Plane Development Kit (DPDK) strikes this balance by bypassing the kernel and eliminating the packet processing overhead of the traditional network stack. It efficiently uses hardware to distribute and balance packet loads across CPU cores. Moreover, DPDK is P4 programmable and compatible with most modern Network Interface Cards (NICs) [6], providing an accessible option for high-performance packet processing.

This paper presents a high-performance, adaptive system for detecting and mitigating a wide range of volumetric DDoS attacks using P4-DPDK. The system operates entirely in user space, leveraging multicore CPUs and SmartNICs to enable flexible, high-speed packet processing. It employs a lightweight linear prediction model to forecast traffic trends based on historical observations and dynamically adjusts rate-limiting thresholds to detect floods without relying on hardcoded values. The DDoS detection and mitigation logic is implemented in P4 and compiled into a DPDK packet processing pipeline using the p4c-dpdk backend. The system is evaluated on the FABRIC testbed using both synthetic and real-world traces. The results demonstrate high adaptability and reduced packet loss under attack. The source code of the system is publicly available to facilitate further research and development [7].

The contributions of the paper are summarized as follows:

- Designing a predictive DDoS mitigation system based on a linear prediction model integrated into a P4-DPDK pipeline.
- Optimizing the detection sensitivity through hyperparameter tuning of the linear model, ensuring accurate flood detection with minimal false alerts.
- Evaluating the system's adaptability to real-world network traces using datasets from a backbone network, and testing its scalability on the FABRIC testbed.
- Demonstrating superior performance compared to Suricata-DPDK in terms of packet loss and scalability across increasing CPU core counts.

II. Related Work

This section provides an overview of current DDoS detection and mitigation methods that leverage both hardware and software acceleration, with a particular emphasis on those using P4-programmable data planes, DPDK, and methods incorporating machine learning.

A. DDoS Defenses over P4 Programmable Data Planes

DDoS protection measures that utilize P4 programmable data planes operate at the line rate and are capable of scaling up to several Tbps. Thanks to P4's adaptability, detection and mitigation strategies can be adjusted in response to innovative attack strategies. Current P4-based solutions generally employ probabilistic data structures to efficiently monitor flow statistics at scale (e.g., MV-Sketch [8], POSEIDON [9], and Jaqen [10]). The primary drawback of these solutions is their demand for infrastructure upgrades, which adds cost and complexity. Additionally, such systems are not suitable for deployment in cloud environments where resources are dynamically allocated.

B. DDoS Defenses over DPDK

DPDK effectively addresses the challenges encountered in deploying P4 programmable data planes. Today, most high-end NICs come equipped with DPDK support [6]. The capabilities of DPDK have been investigated in various DDoS applications [11]–[14], demonstrating notable performance improvements. Despite these advancements, there are limitations. Implementations are often in C, a general-purpose language, which does not offer the same adaptability as P4 for adding new functionalities, especially as the complexity of the program increases. Since DPDK bypasses the kernel and is executed in the application layer, many functions that are carried out by the kernel in conventional approaches have to be introduced in the code. Developers designing DDoS applications should account for these low-level functionalities such as port initialization and memory allocation in addition to implementing the DDoS defense mechanism.

C. DDoS Defenses using Artificial Intelligence

While AI-driven approaches for DDoS attack detection and mitigation have shown promising results, they also come with notable disadvantages compared to traditional kernel bypass techniques such as DPDK. One of the key limitations of AI-based methods is their high computational overhead, which can lead to delays in real-time traffic analysis and decision-making, particularly during large-scale attacks [15], [16]. Machine learning models, such as Convolutional Neural Networks (CNNs) require substantial processing power and memory resources to train and deploy effectively, potentially slowing down the response time and reducing the scalability of detection systems [17].

III. Background

A. P4 Language and Portable NIC Architecture (PNA)

Networking devices known as programmable data planes offer customizable packet processing functions. P4, which stands for Programming Protocol-independent Packet Processor, is a specialized language designed for programming these devices [18]. The Portable NIC Architecture (PNA) provides the P4 language with structures and essential functionalities tailored for SmartNICs [19]. It has three main programmable components: the parser, the main control block, and the deparser. The parser's role is to extract both standard and custom headers from incoming packets. The main control unit processes these extracted headers along with any intermediate computations. Lastly, the deparser is responsible for reassembling packets and preparing them for transmission.

B. Data Plane Development Kit (DPDK)

The Data Plane Development Kit (DPDK) is a software acceleration framework optimized for packet processing. It uses kernel bypass to enable direct interaction between the NIC and the DPDK Poll Mode Driver (PMD) in the user space. This eliminates the kernel interrupts and polls incoming packets in batches. This packet processing acceleration technique reduces the overhead that is usually caused by the kernel's networking infrastructure in traditional approaches [20]. DPDK is conventionally developed in C, requiring developers to write detailed and efficient code to set up packet processing pipelines. Although this method provides significant performance and adaptability, it requires a strong knowledge of low-level networking and the DPDK API.

C. P4-DPDK

P4-DPDK acts as a link between P4 and DPDK, enabling developers to write a P4 code that is subsequently converted into a DPDK pipeline. This integration merges P4's expressiveness with DPDK's performance advantages. The p4c compiler [21], uses a DPDK backend known as p4c-dpdk, and converts the P4 code into a specification file (.spec) that is needed to program the DPDK Software Switch (SWX) pipeline [22]. The DPDK libraries are used to build the packet processing pipeline and generate the necessary files for running the application.

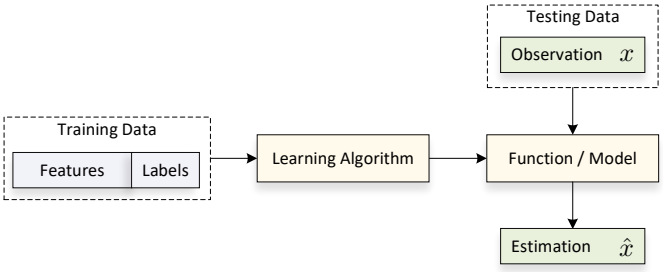

Fig. 1. The structure of a Linear Prediction Model [23].

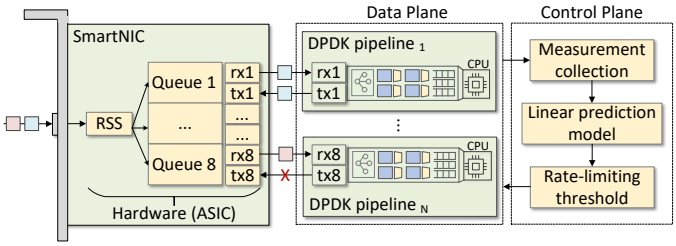

Fig. 2. Proposed system architecture. DDoS mitigation pipelines are deployed on CPU cores. Based on the implemented logic in the pipelines and the calculated rate-limiting threshold, the packets are either allowed or dropped.

TABLE I
LIST OF ATTACKS MITIGATED BY THE PROPOSED SYSTEM.

Protocol	DDoS attack	Description
UDP	UDP flood	The attacker generates fake UDP packets and sends them to the victim server at high rate
TCP	SYN flood	The victim server is flooded with SYN packets
	SYN-ACK flood	The victim server is flooded with SYN-ACK packets
	ACK flood	The victim server is flooded with ACK packets
	FIN/RST flood	The attacker generates fake FIN or RST packets that do not belong to an open connection
	Heavy hitter	The attacker generates a heavy flow to overwhelm the resources of the server
ICMP	ICMP flood	The victim server is flooded with fake ICMP echo-request packets

D. Linear Prediction

A linear system is one in which the output is generated as a linear combination of its current and past inputs, as well as its previous outputs. This means that the system follows the principles of superposition and scaling, ensuring that its behavior can be described using linear equations. Linear prediction is a widely used technique in time series analysis. It relies on linear prediction coefficients, which describe how past values in a time series can be used to estimate future values [24]. The structure of the linear Prediction Model is described in Fig. 1. The system relies on features of training data that are known to fall into a specific category. After this portion of the data is used in the learning algorithm, the parameters of the Linear Prediction Function are tuned. At this stage, the function takes as an input the features of the current observation denoted by x and computes the next predicted value $\hat{x}$.

IV. Proposed System

Consider Fig. 2 which shows a high-level overview of the proposed system. The system employs a DDoS attack detection and mitigation pipeline operating in the user space on multicore CPUs. Using DPDK, packets bypass the kernel entirely. When a packet is received, the SmartNIC directs it to a designated pipeline running on a specific CPU core using the in-hardware Receive Side Scaling (RSS) hashing module. As the packet reaches the pipeline, it undergoes processing by the algorithm in the data plane which also writes the count of packets to registers. Simultaneously, The control plane continuously reads the number of packets received every second and uses these measurements to forecast the number of packets in the next observation. Based on the expected value, the allowed number of packets to be processed is pushed to the data plane which will drop the packets that exceed the allowed range. The implemented code detects and mitigates seven different DDoS attacks listed in Table I.

A. Throughput forecasting Using the Linear Prediction Model

The following Variable and constants are considered for the mathematical representation of the model:

- X : Set of observed packets per second.
- k : Number of past samples used for prediction
- α : Linear prediction coefficient

A sequence of the last k observed packets per-second counts is denoted in X:

$$X = [x_1, x_2, \ldots, x_k]$$

The aim is to predict the upcoming element of $\hat{x}$ by calculating the average of the differences of the observations in X and summing it with the last observation x_k as follows:

$$\hat{x} = \frac{1}{k-1} \sum_{i=1}^{k-1} (x_{i+1} - x_i) + x_k$$

The linear prediction coefficient α is a constant value ranging between zero and one. Based on the predicted value $\hat{x}$ and α, the allowed number of packets per second x_{allowed} is determined by increasing the predicted amount of packets by a certain offset as shown in the following equation:

$$x_{\text{allowed}} = \hat{x} * (1 + \alpha)$$

The calculated amount of packets allowed x_{allowed} is needed for two reasons: 1) to make sure that the network is able to observe a normal increase in traffic, and 2) to limit the rate of processed packets when a flood is detected. This value is written to a register and pushed to the data plane to make sure that the processed number of packets in the pipeline will not exceed the allowed amount x_{allowed}.

B. DDoS Attack Detection and Mitigation Methods

The system checks for a DDoS attack by calculating the error ratio of the predicted value of $\hat{x}$ to the next observed value x is calculated as follows:

$$e = \frac{\hat{x}}{x}$$

The value of e is used to calculate how close the expected value $\hat{x}$ is to the actual value x. Therefore a good prediction will result in a value of e that is close to one which also indicates a normal behavior in the network.

When an attack is initiated, the observed value x will be large compared to the expected value $\hat{x}$ resulting in a sudden drop in the value of e. If $e < (1 - \alpha)$, a flood is detected. At this stage, only the allowed number of packets x_{allowed} is

processed by the P4-DPDK pipeline in the data plane and the remaining packets are dropped. The attack is considered to be over when the ratio e recovers and satisfies the inequality $e > (1 - \alpha)$. For this to happen, the observed number of packets per second x has to drop below the allowed number of packets x_{allowed} and the network will recover to its normal behavior, and the linear prediction resumes.

C. Optimization Model of the Linear Prediction Model Hyperparameter

The linear prediction model relies on two parameters that influence the overall accuracy of the system. The number of previously observed samples, denoted by k affects the accuracy of the predicted value $\hat{x}$. Considering more historical data results in a more informed prediction of future observations. However, a higher value of k introduces a trade-off, as it delays the point at which the system can begin generating predictions. The linear prediction coefficient α plays an important role in calibrating the system's sensitivity. The model needs to be sensitive enough to trigger alerts every time a flood is detected. An overly sensitive system would observe fluctuations in legitimate traffic as floods, therefore triggering false alerts that lead to increased packet loss. To find the most suitable parameters, an optimization problem is modeled as follows.

$$\min_{\alpha,k} \quad \left(\frac{1}{T} \sum_{t=1}^{T} |x_t - \hat{x}_t| - \frac{1}{\alpha} + \beta\, k \right)$$
$$\text{subject to} \quad k \geq 3, \quad 0 < \alpha < 1, \quad \forall e < 1 - \alpha$$

The considered decision variables are α and k. The objective function to be minimized consists of three components that contribute to the overall cost: 1) The mean Absolute Error (MAE). It is calculated by averaging the absolute value of the difference between the observed value x and the predicted value $\hat{x}$ every second t over an interval of time T measures in seconds. This component has to be minimal to ensure that the system produces the best predictions. 2) The smallest value of α ranging from 0 to 1 while the ratio e remains less than $1-\alpha$. This ensures the system's ability to detect attacks while avoiding any false positive alerts. Finally, 3) The smallest amount of previous observations k being at least equal to three which is the minimal number of observations needed to calculate a prediction. k is multiplied by β, which is a weight that can be calibrated to specify the impact of the value of k on the cost function.

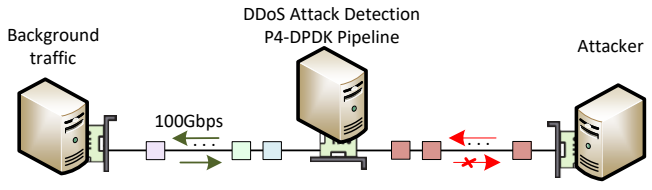

Fig. 3. Experiment topology. The P4-DPDK pipeline detects the flood traffic generated by the attacker, while it processes background traffic at a high rate.

V. Implementation and Evaluation

The system is implemented and deployed on FABRIC [25], an international testbed funded by the NSF for large-scale research and experimentation. The FABRIC infrastructure comprises a network of distributed equipment located at different sites. It offers resources including servers, NICs, high-speed links, and more.

The implemented topology, shown in Fig. 3, consists of three nodes, each equipped with an NVIDIA ConnectX-6 NIC (100Gbps). The servers are located in the same FABRIC site. DPDK-pktgen [26] is used to generate both background traffic and attack traffic. The background traffic consists of packets from a range of flows (i.e., unique 5-tuple). The P4-DPDK pipeline reflects all incoming processed packets. This allows measuring the performance of the system by comparing the number of sent packets with the number of received (reflected) packets. The attacker node generates different types of attacks that will be detected and mitigated by the P4-DPDK pipeline.

A. Rate-limiting Based on Linear Prediction

In this experiment, the system continuously monitors network behavior and uses a linear prediction model to calculate the allowed margin of throughput. This model analyzes historical traffic patterns to predict the expected number of packets under normal conditions. An attack is detected by comparing the predicted allowed margin with the most recent observed packet count. For this experiment, a SYN flood attack is evaluated among six other attacks that the system is able to mitigate in a similar way. The throughput prediction is calculated from the throughput history of size k that is equal to 4 in this example. The allowed margin and the detection of the flood are in function of α which is set to 0.2 in this example. Consider Fig. 4. The system starts by detecting the normal behavior of the system based on the benign traffic received in the first 10 seconds as represented by the blue solid line. All packets were also processed as represented by the blue dotted line. In the tenth second, a flood is detected. While the received flood traffic (solid red line) is above the allowed range (shaded blue area), the system only processes a number of packets (dotted red line) that matches the allowed range. The attack is claimed to be finished when the floor traffic drops below the allowed range as demonstrated in the twentieth second. At this

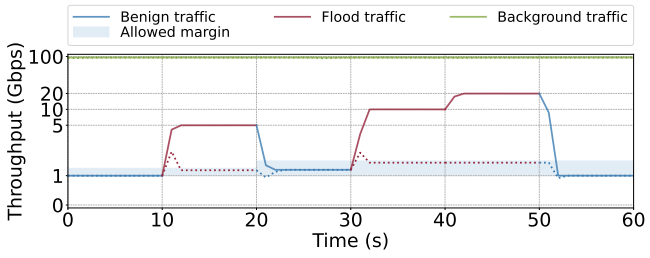

Fig. 4. Detection and mitigation of a flood attack using the linear prediction model. The solid lines refer to the generated traffic, and the dotted lines refer to the processed (reflected) traffic.

stage, the system will resume its normal activity and continue the prediction process and the estimation of the allowed range. Another flood is demonstrated at the thirtieth second and ends at the fiftieth second. The proposed system detects the attacks while it processes legitimate traffic at a high rate approaching 100 Gbps of throughput.

B. Evaluating the Impact of the Linear Prediction Coefficient

The aim of this experiment is to clarify the effect of the Linear Prediction Coefficient α on the function of the system. Fig. 5 describes the behavior of the system when subject to a flood attack initiated at the tenth second with different values of α (0.05, 0.1, 0.25, and 0.5). It is concluded from this experiment, that α is a parameter that specifies the sensitivity of the system to sudden increases in the throughput. For a small value for α such as 0.05, the allowed range is a small offset above the predicted value, and a small rise in the observed throughput is sufficient to claim an attack. Whereas, for a bigger value of α such as 0.5, the allowed range is a remarkable offset above the predicted value, and a more aggressive rise in the observed throughput is needed to claim an attack. Therefore, the smaller the value of α, the higher the sensitivity of the system. It is highly required that the system would be sensitive enough to detect the flood in the system, but a highly sensitive system would generate a great amount of false detections. Ideally, the value of α should be well calibrated based on throughput recorded during the system's normal activity taking into account the fluctuations in the throughput.

C. Evaluating the Adaptability of the System with Real-traffic

This experiment evaluates the system with open-source real traffic captured from different Measurements and Analyses on the WIDE Internet (MAWI) datasets [27]. The MAWI packet traces are publicly available datasets of network traffic data. The considered traces are captured over 15-minute time slots during different times of the day, morning, noon, and night. The traces are processes to calculate the observed number of SYN packets per second and are plotted in Fig. 6. Each processed trace is then passed to the optimization algorithm

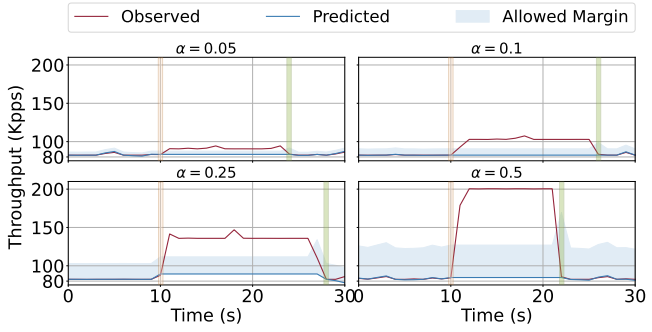

Fig. 5. Behavior of the proposed system with different values of α. A SYN Flood attack is initiated at the tenth second shaded in red and ends the time sample shaded in green.

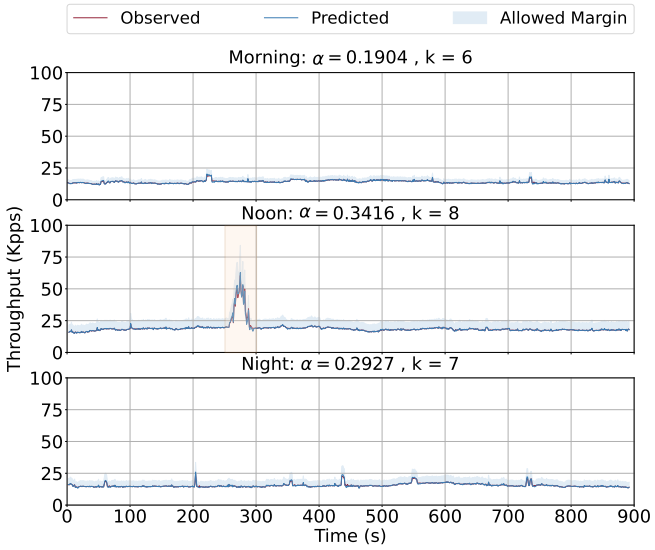

Fig. 6. Optimization of the values of α and k based on the observed MAWI traces at different times of the day.

which converges to the values of α and k that minimize the cost function. Finally, the values resulting from the optimization model are considered, and the proposed system calculates the predicted throughput and the allowed margin. It is shown in the results that the predicted and observed throughput measurements approximately overlap, which indicates that the system functions properly with minimal error. The values of α and k are the highest at noon. This is because during work hours the network tends to be busier and observes sudden rises in throughput (shaded in red) compared to the remaining time slots during the day.

D. Comparing Packet-loss in P4-DPDK and Suricata-DPDK

Suricata [14] is a well-known open-source network Intrusion Detection System (IDS) and Intrusion Prevention System (IPS). It is designed to scale across a range of CPU cores and leverage DPDK for acceleration and bypassing the kernel. It includes a set of approximately 20 thousand rules to mitigate attacks that target networks. The Suricata rules to be loaded can be written and specified by the user. However, the parameters will always depend on hard-coded thresholds that have to be set. On the other hand, the proposed system avoids any hard-coded values for thresholds and adapts to the network requirements. The experiment involves implementing seven DDoS attack mitigation through specified Suricata rules. To ensure a fair comparison, we disable Suricata's wide suite

```
alert tcp any any -> any any (msg:"SYN_packet"; flags:S;
    noalert; sid:123;)
```

```
rate_filter gen_id 1, sig_id 123, track by_dst, count
    1000000, seconds 1, new_action drop, timeout 30
```

Fig. 7. Suricata rule and configuration to mitigate a SYN flood attack.

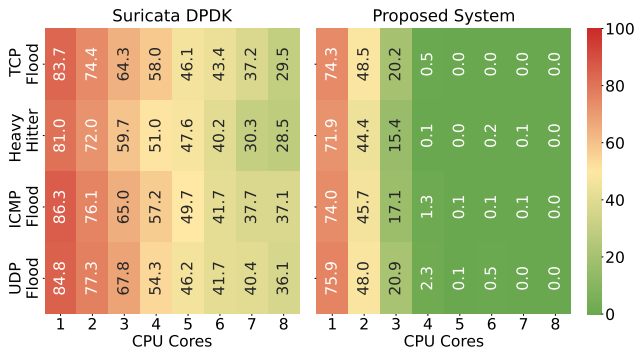

Fig. 8. Packet loss in Suricata-DPDK (a) and the proposed system (b) while mitigating DDoS attacks. Packet loss decreases with more CPU cores.

of predefined rules, loading only our configured rules, and activating DPDK when running Suricata. Fig. 7 shows an example of an attack deployed using a Suricata rule. The first block of code is a rule that detects a SYN packet by monitoring TCP packets with the SYN flag set. The second block of code is a threshold and filter configuration. It creates a filter that restricts the number of received SYN packets sent to the same source IP to a maximum of one million per second, dropping any packets exceeding this threshold.

The results in Fig. 8 (a) show that the packet loss decreases on average from 83% to 31% as we increase the number of CPU cores from one to eight.

To evaluate the performance of our system, we measured the packet loss in the processed background traffic while the network was under various DDoS attacks, and the P4-DPDK pipeline was detecting and mitigating the attacks. The measurements shown in Fig. 8 (b) indicate that using more CPU cores, which also involves running more P4-DPDK pipelines in parallel, significantly reduces packet loss. With four or more CPU cores, the packet loss is nearly eliminated.

VI. Conclusion and Future Work

This work introduces an approach to DDoS detection and mitigation that combines the adaptability of the P4 language with the performance benefits of DPDK, deployed across multicore CPUs. By leveraging a lightweight linear prediction model, the system dynamically computes rate-limiting thresholds in real-time, achieving high detection accuracy without reliance on hardcoded parameters. Evaluations conducted highlight the system's capability to process traffic at nearly 100 Gbps, adapt to varying traffic patterns using real-world MAWI traces, and outperform Suricata-DPDK in terms of packet retention during attack scenarios. The system's performance scales effectively with added CPU cores, reducing packet loss and maintaining service availability. Future work will focus on extending detection capabilities and implementing the system entirely in the SmartNIC pipeline for further packet processing acceleration.

Acknowledgment

The authors would like to acknowledge the National Science Foundation (NSF) for supporting this work, under awards 2346726 and 2403360. The authors would also like to acknowledge the FABRIC [25] team.

References

[1] David Warburton, "2024 DDoS attack trends." [Online]. Available: https://tinyurl.com/35sammzj, 2024. Accessed: Jan. 1, 2025.
[2] S. Mansfield-Devine, "The evolution of DDoS," *Computer Fraud & Security*, vol. 2014, no. 10, pp. 15–20, 2014.
[3] T. E. Ali, Y.-W. Chong, and S. Manickam, "Comparison of ml/dl approaches for detecting ddos attacks in sdn," *Applied Sciences*, vol. 13, no. 5, p. 3033, 2023.
[4] Kfoury et. al., "A comprehensive survey on SmartNICs: Architectures, development models, applications, and research directions," *IEEE Access*, 2024.
[5] Kfoury et. al., "An exhaustive survey on P4 programmable data plane switches: taxonomy, applications, challenges, and future trends," *IEEE Access*, vol. 9, pp. 87094–87155, 2021.
[6] DPDK, "NICs." [Online]. Available: https://tinyurl.com/yzzeujtm. Accessed: Jan. 1, 2025.
[7] samiachoueiri Github, "P4-DPDK-Security." [Online]. Available: https://github.com/samiachoueiri/p4-dpdk_sec.git.
[8] Tang et. al., "A fast and compact invertible sketch for network-wide heavy flow detection," *IEEE/ACM Transactions on Networking*, 2020.
[9] Zhang et. al., "Poseidon: mitigating volumetric DDoS attacks with programmable switches," in *The 27th NDSS*, 2020.
[10] Liu et. al., "Jaqen: A {High-Performance}{Switch-Native} approach for detecting and mitigating volumetric {DDoS} attacks with programmable switches," in *30th USENIX Security Symposium*, pp. 3829–3846, 2021.
[11] Yang et. al, "Heavy hitter detection and identification in software defined networking," in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–10, IEEE, 2016.
[12] Basat et. al., "Volumetric hierarchical heavy hitters," in *2018 IEEE 26th International Symposium on MASCOTS*, pp. 381–392, IEEE, 2018.
[13] Scherrer et. al., "LOFT: an efficient and scalable algorithm for detecting overuse flows," in *2021 40th International SRDS*, IEEE, 2021.
[14] Suricata OISF, "Using Capture Hardware - DPDK." [Online]. Available: https://tinyurl.com/5t43ftv3. Accessed: Apr. 1, 2024.
[15] Hamad et. al., "Deep learning algorithms for detecting and mitigating ddos attacks," *The Indonesian Journal of Computer Science*, vol. 13, no. 2, 2024.
[16] Yungaicela-Naula et. al., "A flexible sdn-based framework for slow-rate ddos attack mitigation by using deep reinforcement learning," *Journal of network and computer applications*, vol. 205, p. 103444, 2022.
[17] Shorubiga et. al., "Cnn-based model for the http flood attack detection," in *2023 International Conference for Advancement in Technology (ICONAT)*, pp. 1–6, IEEE, 2023.
[18] Bosshart et. al., "P4: Programming Protocol-independent Packet Processors," *ACM SIGCOMM Computer Communication Review*, 2014.
[19] Simon et. al., "High-performance match-action table updates from within programmable software data planes," in *Proceedings of the Symposium on Architectures for Networking and Communications Systems*, 2021.
[20] Zhu, *Data Plane Development Kit (DPDK): A Software Optimization Guide to the User Space-Based Network Applications*. CRC Press, 2020.
[21] P4lang, "p4c." [Online]. Available: https://tinyurl.com/4jfwr9bd. Accessed: Jan. 1, 2025.
[22] DPDK, "DPDK Pipeline Application." [Online]. Available: https://tinyurl.com/227m8n85. Accessed: Jan. 1, 2025.
[23] Awaliyatul Hikmah, "Understanding Supervised Learning: The Basics of Linear Regression." [Online]. Available: https://tinyurl.com/yc7bzfwu. Accessed: Apr. 1, 2024.
[24] A. O'Cinneide, D. Dorran, and M. Gainza, "Linear prediction: The problem, its solution and application to speech," 2008.
[25] Baldin et. al., "FABRIC: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, 2019.
[26] Pktgen, "Pktgen-DPDK." [Online]. Available: https://tinyurl.com/2kcyajcz. Accessed: Jan. 1, 2025.
[27] MAWI Working Group, "Packet Traces from WIDE Backbone." [Online]. Available: https://mawi.wide.ad.jp/mawi/. Accessed: Apr. 1, 2024.