Volumetric DDoS Attack Detection and Mitigation using 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*—Distributed Denial of Service (DDoS) attacks continue to be a major threat to network environments, with their scale and complexity increasing over time. Conventional methods, such as firewalls and scrubbing centers, struggle to cope with the high traffic volumes associated with such attacks. Other recent software and hardware approaches face significant limitations in terms of scalability, flexibility, and real-time attack mitigation. This paper presents a DDoS attack detection and mitigation system that leverages the Data Plane Development Kit (DPDK) for high-speed packet processing. The system integrates a P4-based pipeline for efficient detection and mitigation directly within the data plane. The system is evaluated in a testbed environment, while varying parameters such as the attack volume, the packet size distribution, the CPU core allocations, and others. The results show that the proposed system effectively detects and mitigates DDoS attacks while maintaining high throughput and low latency, even at a high traffic rate approaching 100Gbps. Additionally, the system is compared against a widely-used open-source IDS/IPS solution.**

***Index Terms*—DPDK, DDoS, P4, PNA, SmartNIC.**

I. Introduction

Network environments are increasingly targeted by large-scale Distributed Denial of Service (DDoS) attacks [1]. These attacks are orchestrated by malicious actors controlling a network of compromised hosts. In recent years, both the scale and complexity of DDoS attacks have surged, with traffic volume doubling in the past year [1].

Existing methods to counteract these attacks can be categorized into software and hardware-based approaches. Software-based solutions offer great flexibility and can be easily deployed in various networks. However, they struggle to handle large volumes of traffic, particularly during DDoS attacks [2]. This limitation arises because they use the standard network stack, which introduces significant packet processing overhead.

In contrast, hardware-based solutions, such as those using P4-programmable data plane switches or Field-Programmable Gate Array (FPGAs), offer significantly higher processing speed than software-based approaches [3], [4]. However, they come with several limitations. First, they lack the flexibility of devising complex packet processing logic. Second, they require specialized expertise to program, operate, and maintain, making integration challenging in networks largely dominated by fixed-function devices. Third, they can be expensive to deploy at scale and are not typically supported in cloud-based environments, which poses a challenge as many businesses and enterprises have shifted to the cloud.

An ideal approach is to leverage the flexibility and ease of deployment of software-based solutions, without significantly sacrificing performance. The Data Plane Development Kit (DPDK) serves as a middle ground. DPDK bypasses the kernel and avoids the packet processing overhead of the traditional network stack. It also uses the hardware to efficiently distribute and load balance packets across CPU cores. Additionally, DPDK is supported on most modern Network Interface Cards (NICs) [5], making it an accessible solution for high-performance packet processing.

This paper presents a system that detects and mitigates a variety of DDoS attacks, leveraging DPDK for high-speed packet processing. The proposed system uses various techniques and data structures (e.g., Count-Min Sketch (CMS), connection tracking) to optimize the resources used by the pipeline. The mitigated attacks were chosen based on their prevalence in the current threat landscape [6]. The system's logic is expressed in a P4 code and compiled into a P4-DPDK packet processing pipeline. The system was tested and evaluated in a large scale testbed, taking into account several parameters, including packet size distribution, the number of CPU cores, and the volume of attacks. 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:

- Implementing a DDoS attack detection and mitigation solution using P4-DPDK.
- Analyzing the impact of mitigation of the attack on the performance of the network in terms of throughput, latency, and packet loss.
- Testing the system's performance on FABRIC [8], an international testbed used for research and experimentation.
- Comparing the performance of the system against a popular open-source IDS/IPS that detects similar attacks.

The structure of the paper is as follows. Section II discusses related work. Section III offers a brief overview of the technologies utilized in this system. Section IV presents the proposed system. Section V outlines the implementation details and provides an evaluation of the system. Finally, Section VI concludes the paper.

II. Related Work

This section provides an overview of existing DDoS detection and mitigation schemes that utilize hardware and software

acceleration, specifically focusing on those that employ P4-programmable data planes and DPDK.

A. DDoS Defenses over P4 Programmable Data Planes

DDoS defenses implemented on P4 programmable data planes run at line rate and can scale to multiple Tbps. With P4's flexibility, it is possible to adapt detection and mitigation mechanisms based on recent attacks. Existing P4 approaches typically use probabilistic data structures to track flows' statistics at scale (e.g., MV-Sketch [9], POSEIDON [6], and Jaqen [10]). The main issue with these approaches is that they require networks to upgrade their infrastructure, which results in cost and complexity challenges. Also, such systems are not deployable in cloud environments where resources are dynamically allocated.

B. DDoS Defenses over DPDK

As a software-based acceleration technique, DPDK overcomes the deployment challenges faced by P4 programmable data planes. Most of the Network Interface Cards (NICs) today already support DPDK [5]. The potential of DPDK has been explored by multiple DDoS applications [11]–[13] showing significant performance gains. However, these approaches have some limitations. They are implemented in a general-purpose programming language (C), lacking the flexibility of P4 for integrating new functionalities, particularly as the program's complexity grows. Also, they incur unnecessary processing overhead because they are integrated with other functionalities of the Open vSwitch (OVS). These functionalities may not be needed by most networks. Finally, they are not validated on large scale testbed with real traffic traces.

III. Background

A. P4 Language and Portable NIC Architecture (PNA)

Programmable data planes are networking devices with customizable packet processing behavior. P4, or Programming Protocol-independent Packet Processor, is a domain-specific language for programming these devices [14]. The Portable NIC Architecture (PNA) is a specification that provides the P4 language with structures and common capabilities for SmartNICs [15]. It has three main programmable blocks: the programmable parser, the main control block, and the programmable deparser. The parser is responsible for extracting standard and user-defined headers from the incoming packets. The main control block is responsible for processing the extracted headers and intermediate results. The deparser is responsible for reassembling the packets and serializing them for transmission.

B. Data Plane Development Kit (DPDK)

The Data Plane Development Kit (DPDK) is a software acceleration framework optimized for packet processing. Consider Fig.1. Standard packet processing operates through the kernel space, where packets traverse the network stack using the kernel's networking infrastructure and drivers. This method introduces latency due to context switching between kernel and user space, and the overhead of traversing the entire network stack. In contrast, packet processing acceleration using kernel bypass, as demonstrated by the DPDK framework, enables 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, reducing overhead and enabling high-speed packet processing [16].

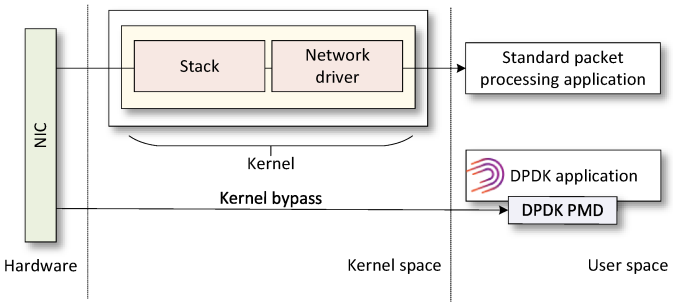

Fig. 1. DPDK packet processing vs. standard packet processing. DPDK bypasses the kernel avoiding the packet processing overhead [17].

DPDK is traditionally programmed in C, requiring developers to write detailed and optimized code to configure packet processing pipelines. While this approach offers high performance and flexibility, it demands a deep understanding of low-level networking and the DPDK API.

C. P4-DPDK

P4-DPDK is a technology that bridges P4 and DPDK. It allows developers to create P4 code which is then translated into a DPDK pipeline, combining the expressiveness of P4 with the performance benefits of DPDK. Fig. 2 shows the process of converting a P4 code into a DPDK pipeline application. The p4c compiler [18] with a DPDK backend (p4c-dpdk) accepts the P4 code and compiles it into a specifications file (.spec) needed to program the DPDK Software Switch (SWX) pipeline [19]. The DPDK libraries are used to build the pipeline that consists of three main components; (1) input ports attached to the receiving packet queues, (2) the match action tables that are determined by the logic in the P4 application, (3) the output ports attached to the transmitting packet queues. Subsequently, a C code is generated including C functions corresponding to each action and control block. A C compiler

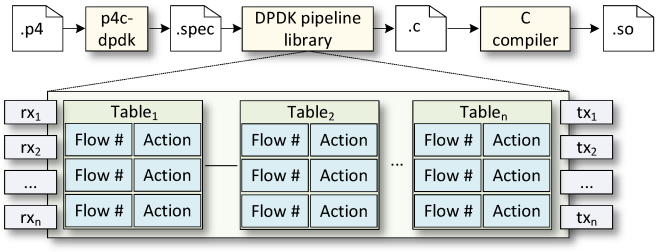

Fig. 2. The p4c-dpdk workflow and the structure of the DPDK packet framework pipeline [16].

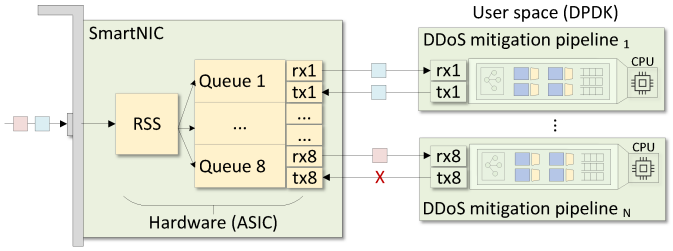

Fig. 3. Proposed system architecture. DDoS mitigation pipelines are deployed on CPU cores. Based on the implemented logic in the pipelines, the packets are either allowed or dropped.

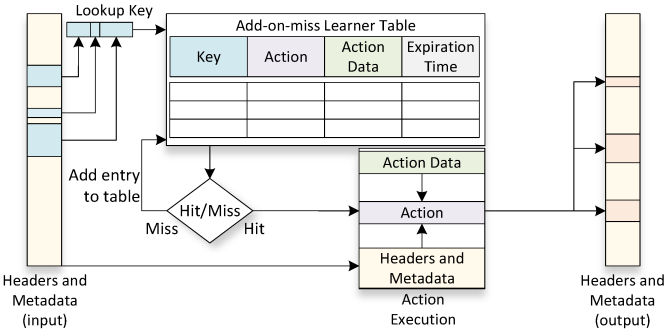

Fig. 4. The add-on-miss feature supported by the PNA in P4.

then generates a shared object (.so) that is needed to execute the application.

IV. Proposed System

Consider Fig. 3 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 which specifies whether a packet is allowed or dropped. The implemented code detects and mitigates seven different DDoS attacks listed in Table I.

A. DDoS Attack Detection

The proposed system uses three detection methods: 1) thresholding based on the total packet count, 2) connection tracking, and 3) thresholding based on the per-flow packet count. The UDP flood attack along with the SYN, SYN-ACK, and ACK TCP flood attacks are detected by considering the total amount of packets for the different packet types. The packet-count is stored in a single-cell register dedicated to each attack. The value that these registers hold is compared to an adjustable threshold, and when exceeded, an attack is detected.

TABLE I
List of attacks mitigated by the proposed system.

Protocol	DDoS attack	Description	Detection	Mitigation
UDP	UDP flood	The attacker generates fake UDP packets and sends them to the victim server at high rate	Total packet count	Rate limiting
TCP	SYN flood	The victim server is flooded with SYN packets	Total packet count	Rate limiting
	SYN-ACK flood	The victim server is flooded with SYN-ACK packets	Total packet count	Rate limiting
	ACK flood	The victim server is flooded with ACK packets	Total packet count	Rate limiting
	FIN/RST flood	The attacker generates fake FIN or RST packets that do not belong to an open connection	Connection tracking	Blocking out-of-session flows
	Heavy hitter	The attacker generates a heavy flow to overwhelm the resources of the server	Per-flow packet count	Blacklisting the flow
ICMP	ICMP flood	The victim server is flooded with fake ICMP echo-request packets	Per-flow packet count	Blacklisting the flow

The ACK packets received indicate an open TCP connection. To keep track of the open TCP sessions, these flows are populated in a table referred to as the "add-on-miss" table. This property is a distinctive feature supported by the P4 PNA [20]. It enables dynamic rule addition to a table when no matching entry is found, eliminating the need for direct intervention from the control plane. It is important to note that this feature is exclusively applicable to tables using exact match operations. When enabled, the default action executed during a miss automatically adds a new entry into the table associating the keys with an action, action parameters, and an expiration time as shown in Fig. 4. Consequently, this newly added entry ensures that the next incoming packet with the same characteristics will result in a match as long as the rule is not expired and removed from the table. A FIN flood is detected if the packet is received from a flow that does not have an established TCP connection recorded in the table.

The heavy-hitter and ICMP flood attacks are detected by considering the per-flow packet count and comparing the individual measurements to an adjustable threshold. In this case, a scalable approach is needed to be able to track the

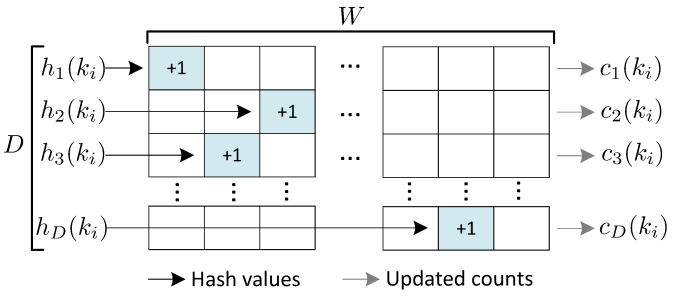

Fig. 5. Count-min Sketch (CMS) for estimating the per-flow packet counts.

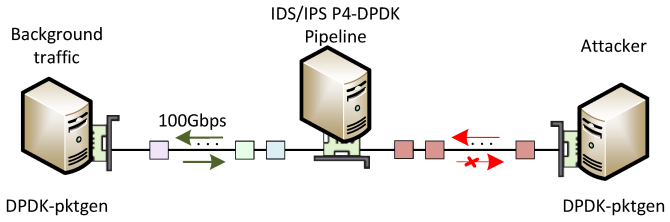

Fig. 6. Experiment topology. The malicious traffic is generated from a single or randomized 5-tuple based on the type of attack.

count of packets from a large number of flows. The Count-min Sketch (CMS) data structure efficiently handles packet stream updates while reducing memory usage compared to exact counting methods. The CMS structure shown in Fig. 5, is represented by a two-dimensional array with D rows (depth) and W columns (width). k_i identifies the i-th packet's flow based on the 5-tuple. The key is the input to a hash function $h(k_i)$ to locate the counter in the sketch and increment it by one. A flow is detected as a heavy hitter if the minimum value among its estimated packet counts $c(k_i)$ exceeds a predefined threshold ϕ.

B. DDoS Attack Mitigation

As soon as an attack is detected, the system mitigates it in one of three different ways to prevent the attack from overwhelming the victim: 1) Rate limiting, 2) Blocking out-of-session flows, and 3) Blacklisting the flow. The UDP flood attack along with the SYN, SYN-ACK, and ACK TCP flood attacks are mitigated by dropping packets probabilistically within a predefined and adjustable percentage to limit the sending rate of packets. Whereas, the FIN flood attack is mitigated by blocking out-of-session packets. In this case, only FIN packets that aim to close a previously open TCP connection recorded in the table will be allowed. Finally, the heavy hitter and ICMP flood attacks are mitigated by blacklisting flows that are detected as heavy flows. Therefore, any blacklisted flow is dropped entirely.

V. Implementation and Evaluation

The system is implemented and deployed on FABRIC [8], 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. 6, consists of three nodes, each equipped with an NVIDIA ConnectX-6 NIC (100Gbps). The servers are located in the same FABRIC site. DPDK-pktgen [21] 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.

Fig. 7. The system processes 100Gbps of background traffic while (a) mitigating DDoS simultaneous attacks and (b) mitigating heavy hitters from different flows and rates. The solid lines refer to the generated traffic, and the dotted lines refer to the processed (reflected) traffic.

A. Analyzing the Effect of DDoS Attacks Detection and Mitigation on Background Traffic

1) Analyzing the Effect of Simultanous DDoS Attacks: This experiment evaluates the ability of the pipeline to detect and mitigate different types of DDoS attacks in the presence of high-rate traffic. Fig. 7 (a) shows the throughput over time. The solid lines correspond to the generated traffic and the dotted lines correspond to the processed (reflected) traffic. The background traffic packets were all processed by the pipeline using four CPU cores. The attacks are generated one after the other. The attackers generate 1500-byte packets at 20Gbps from a range of flows or a single flow depending on the attack. The UDP attack and all TCP attacks, except for the FIN flood and heavy hitter attacks, are detected when register counters exceed a threshold value of one million packets. These attacks are mitigated by reducing and limiting the sending rate each by a different percentage: SYN 25%, SYN-ACK 50%, ACK 75%, and UDP 50%. The packets generated by the FIN flood attack are all dropped since none of them aim to end a previously open connection. Finally, the heavy hitter and ICMP flood attacks, which are generated from single flows, are detected and blocked by the pipeline. The flows are dropped when the flow packet count exceeds a value of 100,000 packets for a TCP heavy hitter and 200,000 for an ICMP heavy hitter. The system promptly mitigated the detected attack while processing traffic at a high rate.

It is worth noting that the thresholds and the limiting rates are dynamically configurable; the values discussed here are just for demonstration purposes.

2) Detecting Multiple Heavy Hitters with High Traffic Rate: This experiment tests the system's ability to detect different

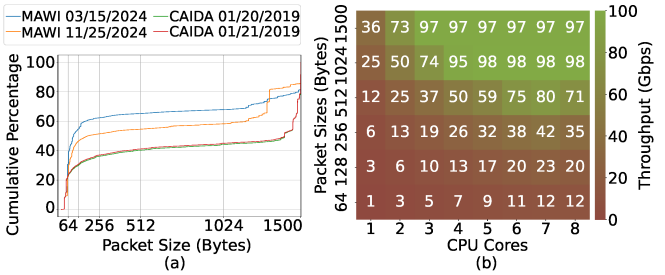

Fig. 8. Real traffic CDF (a) and throughput at the designated packet sizes at different numbers of CPU cores (b).

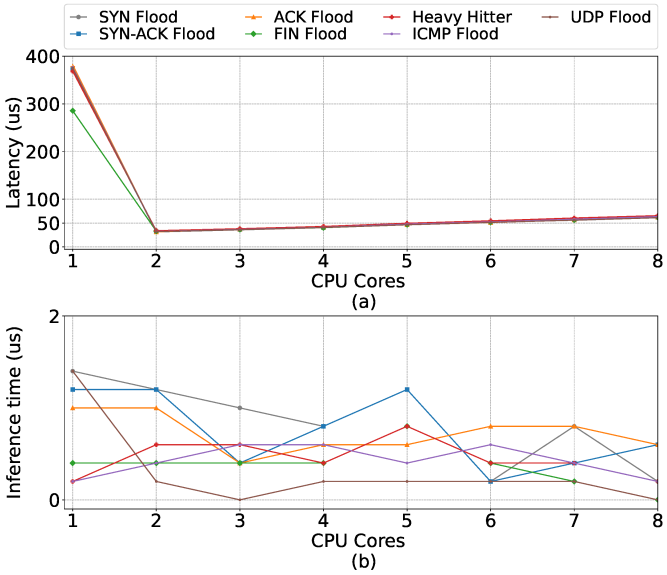

Fig. 9. (a) Latency and (b) inference time vs. the number of CPU cores.

heavy hitters in the presence of high traffic rates. The background traffic generator is sending at a 100Gbps rate from a range of flows. Fig. 7 (b) shows the results of this experiment. The background traffic packets were all processed by the pipeline and reflected. Five heavy hitters have been detected in this experiment. Each heavy hitter sends packets at different rates from different networks. The results also show that the slower the rate of the attack, the longer it takes to exceed the threshold. When the packet counter of a flow exceeds the predefined threshold value ϕ, which is a million packets in this case, a heavy hitter is identified and blocked within less than one microsecond, even while 100Gbps of background traffic is processed by the pipeline using four CPU cores.

B. Evaluating the Impact of the Number of Pipelines and Packet Size on the Throughput

This experiment studies the impact of the number of running DPDK pipelines (one pipeline per CPU core) and the packet sizes on the maximum achievable throughput. The packet sizes considered are based on the analysis of open-source real traffic captured on different dates from two datasets: Measurements and Analyses on the WIDE Internet (MAWI) [22] and Center for Applied Internet Data Analysis (CAIDA) [23]. The results in Fig. 8 (a) show the packet size Cumulative Distribution Function (CDF) of the four different datasets. The results show that for some datasets, over 50% of the packets were larger than 512 bytes. Fig. 8 (b) shows the throughput achieved with various packet sizes. The results show that as the number of cores and packet size increase, the throughput increases. Therefore, these three parameters are directly proportional. For the smallest packet size considered (64 bytes), the maximum throughput attained is 12Gbps running on eight cores. Using four cores is enough to achieve approach line-rate while considering a minimum packet size of 1024 bytes.

C. Analyzing the Overall Latency and Inference Time in the P4-DPDK Pipeline

The first part of the experiment shows measurements over the lifetime of legitimate packets in the system as they reach the NIC, undergo processing in the pipeline, and leave the NIC. Two timestamps were captured with the help of an external P4 Tofino switch, which provides nanosecond granularity. The measurements, shown in Fig. 9 (a), are collected as the pipeline is processing the legitimate flows at line-rate while it mitigates the seven different attacks. The results indicate that with one CPU core, the latency would be very high as all incoming packets are exhausting the same queue and one CPU core is processing this high load of traffic. While increasing the number of cores to two reduces the latency significantly, as we continue to increase the number of cores, the latency increases slightly. As the packets are hashed to the different cores, the PMD will experience a slight delay in polling the packets since more time is needed to have an amount of packets in each queue that meet the configured burst size. A potential solution is to reduce the Receive (RX) and Transmit (TX) queue size along with the burst size as the number of cores increases to accelerate polling of the packets and reduce the queuing delay.

The second part of the experiment in Fig. 9 (b), shows the inference time for all attacks. The inference time is the time the system takes to detect an attack in any of the operating pipelines. With a higher number of cores, the packet processing algorithm is parallelized in the pipelines assigned to each core. While increasing the number of CPU cores enhances the performance of the system in terms of throughput as seen in Fig. 8 and in terms of packet loss as seen in Fig. 11, it causes a delay in inference time if the considered thresholds are not divided by the number of CPU cores. The results in Fig. 9 (b) show that all attacks are detected within two microseconds.

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

This experiment compares the packet loss of the proposed system against Suricata [24] which is a widely used open-source network Intrusion Detection System (IDS) and IPS. It can be scaled to run on multiple CPU cores and is able to leverage DPDK for acceleration and kernel bypass. In this experiment, a DDoS attack mitigation logic is implemented in Suricata rules. The extensive set of rules made available by

```
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. 10. Suricata rule and configuration to mitigate a SYN flood attack.

Suricata is disabled and only the configured rules are loaded enabling DPDK while running Suricata for a fair comparison. An example of an implemented attack in a Suricata rule is shown in Fig. 10. 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. 11 (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. 11 (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 paper discusses a system that uses P4-DPDK to identify and mitigate volumetric DDoS attacks. It employs a variety of approaches based on which attacks are mitigated. Each attack is mitigated by either dropping the flow entirely or limiting the sending rate. Experiments reveal that increasing the number of CPU cores improves throughput and prevents packet loss. As a trade-off, the latency might be slightly affected. Using four pipelines has been proven to be a fair setting to which the system can be scaled and reserve the performance considering the different aspects. The results indicate that detecting all attacks is effective while the system is also able to process packets at a high traffic rate of 100Gbps. Future work includes 1) implementing AI/ML-based detection techniques for adaptive and more accurate DDoS attack identification; 2) adapting the system for edge deployment for SmartNIC-based acceleration; and 3) Implementing self-adjusting rate limiting mechanisms based on attack severity.

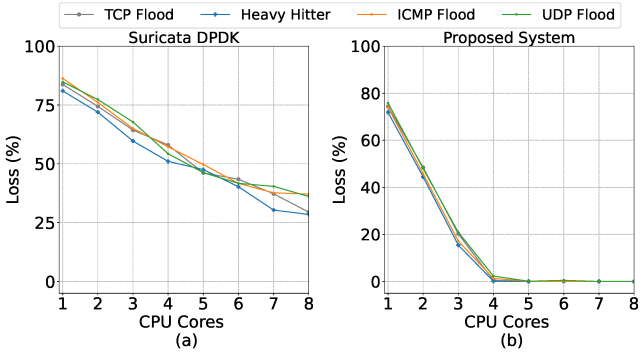

Fig. 11. Packet loss in Suricata-DPDK (a) and the proposed system (b) while mitigating DDoS attacks. Packet loss decreases with more CPU cores.

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 [8] 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] Kfoury et. al., "A comprehensive survey on SmartNICs: Architectures, development models, applications, and research directions," *IEEE Access*, 2024.
[4] 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.
[5] DPDK, "NICs." [Online]. Available: https://tinyurl.com/yzzeujtm. Accessed: Jan. 1, 2025.
[6] Zhang et. al., "Poseidon: mitigating volumetric DDoS attacks with programmable switches," in *The 27th NDSS*, 2020.
[7] samiachoueiri Github, "P4-DPDK-Security." [Online]. Available: https://github.com/samiachoueiri/p4-dpdk_sec.git.
[8] Baldin et. al., "FABRIC: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, 2019.
[9] Tang et. al., "A fast and compact invertible sketch for network-wide heavy flow detection," *IEEE/ACM Transactions on Networking*, 2020.
[10] Liu etl.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] Bosshart et. al., "P4: Programming Protocol-independent Packet Processors," *ACM SIGCOMM Computer Communication Review*, 2014.
[15] 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.
[16] Zhu, *Data Plane Development Kit (DPDK): A Software Optimization Guide to the User Space-Based Network Applications*. CRC Press, 2020.
[17] R. Donato, "What is DPDK?." [Online]. Available: https://tinyurl.com/yfc73h7c, May 2018. Accessed: Jan. 1, 2025.
[18] P4lang, "p4c." [Online]. Available: https://tinyurl.com/4jfwr9bd. Accessed: Jan. 1, 2025.
[19] DPDK, "DPDK Pipeline Application." [Online]. Available: https://tinyurl.com/227m8n85. Accessed: Jan. 1, 2025.
[20] P4 Language Consortium, "P4 Portable NIC Architecture (PNA)." [Online]. Available: https://tinyurl.com/rny8bn8t. Accessed: Jan. 1, 2025.
[21] Pktgen, "Pktgen-DPDK." [Online]. Available: https://tinyurl.com/2kcyajcz. Accessed: Jan. 1, 2025.
[22] MAWI Working Group, "Packet Traces from WIDE Backbone." [Online]. Available: https://mawi.wide.ad.jp/mawi/. Accessed: Apr. 1, 2024.
[23] CAIDA, "CAIDA Data Server: Index of datasets." [Online]. Available: https://data.caida.org/datasets/. Accessed: Jan. 1, 2025.
[24] Suricata OISF, "Suricata." [Online]. Available: https://github.com/OISF/suricata. Accessed: Apr. 1, 2024.