International Journal of Computer Science and Mathematical Theory (IJCSMT) E-ISSN 2545-5699
P-ISSN 2695-1924 Vol 12. No. 10 2026 www.iiardjournals.org **online version**

eBPF Based Network Functions: Utilizing XDP Programs for High-Speed Packet Filtering and Ddos Mitigation in Nigeria

Efiyeseimokumo S. Ikeremo and Firstman Noah Otobo
Department of Mathematics and Computer Science,
University of Africa, Bayelsa State.
samplefiye@gmail.com
Department of Mathematics and Computer Science,
University of Africa, Bayelsa State.
otobonoah@gmail.com
https://doi.org/10.56201/ijcsmt.vol.12.no10.2026.pg117.137

Abstract:
DDoS attacks exploit the weakest links in Linux and Software Defined Networks (SDNs). Many available defenses work by intercepting malicious packets and deciding whether to forward them based on rules stored in a central controller. However, rules are only effective after packets are fully processed by lower level networking components. The eXpress Data Path (XDP) and the Extended Berkeley Packet Filter (eBPF) allow packets to be filtered and dropped at higher networking component layers. This work develops AXDP-Guard. eXDP-Guard is implemented in user space and is designed for high speed, low latency DDoS attacks mitigation. this work presents controlled system simulation to evaluate AXDP-Guard, and compares the results to related work. To examine the performance of AXDP-Guard in a DDoS attack scenario, 40 independent test cases are run to simulate attacks at various volumes and compare results to iptables/nftables, XDP, native AXDP, and controller based SDNs. The assessment analyzes legitimate goodput, protected service availability, CPU burden, latency, low-rate attack detection, and statistical stability. The results demonstrate an important systems trade-off. XDP in its native form demonstrates the best raw performance of all the fast paths, but AXDP-Guard, in trade-off to fast path performance, provides the best adaptive low-rate attack defense and control. Native XDP, at 10-15 Mpps, lost about 0.3% legitimate goodput, whereas AXDP-Guard lost about 1.3% in the same conditions. However, AXDP-Guard out performed native XDP in low-rate DoS attacks, achieving a detecting and mitigating F1 of 0.978, compared to native XDP's F1 of 0.922. Low-rate attacks are best protected against by integration of detection and defense mechanisms at the transport and application layers. XDP is best suited to provide such mechanisms.

Keywords: *XDP; DDoS protection; high-speed transport layer; eBPF; Nigeria*

1. Introduction

Distributed Denial of Service (DDoS) attacks are designed to overwhelm a service with traffic. This can be accomplished with a small number of devices when the attacker has control over the network. In the real world, the attacker may be thousands of miles away from the service. DDoS attacks are becoming faster and larger. Modern infrastructures and cloud platforms give attackers the ability to target thousands of servers at once. On a Linux server, many of the services rely on the principle of filtering large numbers of incoming packets and

making decisions based on those packets. If a server is subjected to a large number of incoming packets, and those packets are all dropped, the server still expends a great deal of resources processing the incoming packets. Traditional methods for packet filtering can only handle a certain number of packets per second. One of the methods for defending against DDoS attacks is to process packets before the traditional networking stack. One method for accomplishing this is to use Extensible Data Path (XDP). One advantage of XDP is that packets can be processed very early, before many decisions are made by the networking stack. XDP allows systems to implement decisions based on incoming packets very early. With XDP, systems can implement mitigations for DDoS attacks by processing and forwarding legitimate packets, and dropping attack packets.

To illustrate this trade off, we present AXDP-Guard. AXDP-Guard shows how XDP can be combined with eBPF to create a flexible, fast system for examining network packets. Rather than perform complex detection logic on all packets, AXDP-Guard assumes that the fast path performs the logic to come to a decision. More complex, expensive, and time-consuming determinations can be performed outside of the fast critical path.

2. Background and Related Work

2.1 eBPF, XDP, and the Linux Fast Path

eBPF is a kernel-based VEE (Verified Execution Environment) in the Linux kernel. With eBPF, programs can be loaded from user space, checked, and attached to kernel hooks. The kernel verifier enforces constraints to restrict programs from accessing memory in an unconstrained manner, thus protecting the stability of the kernel. eBPF programs can communicate with user space through maps. Restricted kernel functions can be made available to eBPF programs through helper functions. When compared to an unconstrained kernel module, eBPF modules can be fitted to load and execute kernel functions. However, eBPF has been predominantly employed to load and execute kernel functions related to networking. Compared to other kernel hooks, XDP is situated nearby the NIC receive processing path. One of the benefits of using XDP is that the related data path execution can be completely bypassed. XDP programs can also offload some of their work to the hardware if it is supported. Because XDP can be implemented at various levels, it can be portable and also help eliminate a lot of the work that has to be performed at each step of data processing.

This fast path is largely based on the original study of XDP (Høiland-Jørgensen et al., 2018). These authors studied the kernel implementation of XDP, and showed that it is capable of performing L3 routing, L4 load balancing, and packet filtering at high line speeds. It is unlikely that the existing network stack can keep up with such high speeds. Nonetheless, these works demonstrated that XDP is capable of filtering out packets at high speeds, and thereby reducing the load on downstream processing and forwarding components.

Due to the stateful nature of DDoS attacks and the corresponding mitigation, statefulXDP is more complex than a simple state-less drop. It may, for instance, need to keep track of the number of packets received from a particular source, or determine the number of packets with a given destination prefix, or track packets associated with a given L4 port or a given L7 message. The paper by Miano et al. (2018, 2019, 2021) demonstrate that such functions can be implemented using eBPF and related abstractions; however, such approaches suffer from the complexity of the verification, the eBPF map, and state management.

2.2 eBPF as a Network-Function Platform

Various examples show the transition from single-purpose XDP programs to flexible network functions. IPv6 Segment Routing and eBPF, as described by Xhonneux et al. (2018), enable flexible functions in the Linux data plane. The Polycube project, described by Miano et al.

(2019, 2021), further separates the eBPF data plane from the control plane, and allows users to define and compose network services. Separating the data and control planes allows security services to remain within the data plane while more flexible and efficient policies are placed in the control plane. The idea of using eBPF to implement data planes is used to implement other kinds of network functions, as well. eBPF is used to build data planes in mobile networking, as described by Soldani et al. (2023). Behl et al. (2023) use eBPF to build data planes in Kubernetes. eBPF is also used to implement service functions and telemetry in Linux (Haeberle et al., 2025). Because of these projects, implementing an XDP-based DDoS filter is feasible, and integrating it with other network functions is acceptable. When further performance improvement with CPU-based XDP becomes infeasible, hardware acceleration is the next available option. Miano et al. (2019) describe a hybrid architecture in which XDP sampling and aggregation takes place on the host, and certain action units are offloaded to a SmartNIC. Brunella et al. (2020) demonstrate the implementation of XDP-style programs on FPGA-based NICs. Salva-Garcia et al. (2022) examine the use of XDP in SmartNICs. All these works point out that, in the case of high packet rates, hardware offloading is justified.
Results from other (non-security) related works also demonstrate that eBPF need not always be the fallback option. For instance, multi-tenant 5G use cases demonstrate that FPGAs outperform eBPF (Gallego-Madrid et al., 2024). For Kubernetes, the use of eBPF depends on the workload (Miziński and Przyłucki, 2025). These works demonstrate that, for various use cases, hardware offloading and SmartNICs can be useful in the Nigeria context, but the main question should be how well they perform against existing systems.

2.3 eBPF/XDP for DDoS and DoS Mitigation

Distributed Denial-of-Service (DDoS) attacks, for which XDP was originally designed to provide protection against, will continue to be a major threat against the Internet. Miano et al. (2019) and Choe et al. (2020) leverage the XDP technology to mitigate DoS attacks and Choe et al. (2020) incorporate XDP to ensure service protection. Recently, in the context of 5G, Loureiro et al. (2024) use eBPF/XDP technology to address DoS attacks against the 5G control plane. The focus of recent research has been on the development of adaptive attack detection mechanisms to complement static drop mechanisms. Zang et al. (2025) propose a volumetric attack detection mechanism where the kernel maintains traffic sketches and determines attack threshold. Elzoghbi and He (2026) develop an adaptive attack detection mechanism for DoS attacks in the context of Software Defined Networking (SDN) by utilizing eBPF/XDP and combining other mechanism such as Inter-Arrival Time (IAT) and/or Burst Time. Rather than communicating with the SDN control plane to retrieve network state information, the mechanisms are deployed in the data plane.
Application layer and semantic DDoS attacks are harder to defend against because they can often look legitimate. A simple solution to defend against DDoS attacks is to implement a packet rate threshold. However, an attacker can easily take advantage of this defense by opening a large number of connections, keeping them open, and making the server do resource-intensive work. Remya et al. (2025) found a way to defend against this type of attack using eBPF and Linux. Instead of processing large amounts of packets and trying to identify attacks, they proposed using eBPF hooks to filter attacks at different stages of the application layer. Using eBPF has its costs. All of the operations that are done on the fast path of eBPF can quickly add up and increase the load on the system. Because eBPF can easily defend against attacks, the goal of defense systems should be to implement as little eBPF as possible to defend against the most attacks. The authors of the paper based their design on this trade-off.

2.4 Nigerian DDoS and SDN Research

Nigerian researchers have focused on DDoS attacks in various contexts, including machine learning, SDN, and others. Multiple studies have emphasized the significance of prompt DDoS attack detection and response. Most of these studies have analyzed locally relevant cases of DDoS attacks. However, the mechanisms employed by these studies to tackle these attacks do not function at the XDP fast path. SDN has been the focus of most of the published studies on DDoS attacks out of Nigeria. This is because SDN allows for easy and centralized definition and control of communication policy. Throttle, proposed by Olakanmi and Odeyemi (2021), employs machine learning to predict traffic volume, and responses to the prediction by limiting the offending traffic. Oguntola et al. (2019) employed Deep Packet Inspection (DPI) in a machine learning framework for DDoS attack detection. Other studies have concentrated on swift and prompt DDoS attack mitigation. However, most of these studies do not address the challenges posed by DDoS attack mitigation, especially if the attack packet rate exceeds the SDN data plane. Adeniji et al. (2022) developed SDN technology to analyze and combat DDoS attacks using IPv6. Strong results were realized using Support Vector Machines (SVMs). In 2024, Wabi et al. took a wider view in SDN-based DDoS attack research and presented an empirical study employing Random Forest. In 2025, Okide et al. explored SDN-based blockchain technology for the same purpose. It can be rightly observed that there is a body of work focused on detection and response to cyber-attacks in the Nigerian academia. The focus, however, has been on upper layer technologies and a larger body of work is expected on more foundational technologies to fill the research-response gap. It may not be accurate to say that research on DDoS attacks in Nigerian academia is virtually non-existent. In fact, it may be more appropriate to say that what is missing is a research work that employed eBPF/XDP to implement DDoS mitigation and used it to benchmark more traditional approaches to Linux and/or Architectural engineering. By defining the research problem as such, other systems-related research focused on mitigating DDoS attacks may be encouraged.

2.5 Research Positioning

Table 1 shows our work in relation to some international and Nigerian works. The international works demonstrated the use of eBPF and XDP to implement different networking functions. They also demonstrated use cases for data center network functions such as traffic filtering and SmartNICs. Different works demonstrated various case studies of DDoS attacks and provided different techniques for detecting and mitigating DDoS attacks. The Nigerian works focused on use cases for SDN and NFV, flow analysis and machine learning for detection and mitigation of network attacks. AXDP-Guard leverages the detection ideas from the Nigerian works and moves the idea of enforcement from userspace to the Linux fast path.

Table 1. Positioning of the proposed study against representative global and Nigerian literature.

Study	Context	Main contribution	Gap relative to this paper
Høiland-Jørgensen et al. (2018)	Global XDP foundation	High-rate in-kernel packet processing and DDoS use case	No Nigerian deployment context
Miano et al. (2019)	Global SmartNIC/XDP	Hybrid host/offload DDoS mitigation	Requires hardware/host design decisions
Loureiro et al. (2024)	Global 5G core	Early eBPF/XDP filtering for 5G-core security	Not evaluated on Nigerian infrastructure
Zang et al. (2025)	Global volumetric DDoS	In-kernel sketching and adaptive thresholds	Detection focus; XDP drop integration can be extended
Elzoghbi & He (2026)	Global SDN LDoS	Kernel-local dynamic thresholds and low controller overhead	No Nigerian testbed
Idris et al. (2017)	Nigeria DDoS detection	ANN-based classification	No eBPF/XDP enforcement
Akanji et al. (2021)	Nigeria slow DDoS	GA + SVM + SDN mitigation	Flow/controller pipeline rather than kernel fast path
Olakanmi & Odeyemi (2021)	Nigeria SDN mitigation	Adaptive throttling and legitimate-host protection	No XDP packet-rate benchmark
Wabi et al. (2024)	Nigeria SDN DDoS	Systematic gaps + RF detection	Calls for lower-overhead practical deployment
Present study	Nigeria-oriented eBPF/XDP	Adaptive XDP network function + 40-seed systems simulation	Provides design and test blueprint; physical field validation remains future work

3. AXDP-Guard: Proposed eBPF/XDP Network Function

AXDP-Guard implements a design philosophy to keep the per-packet path as simple as possible while moving functions that require more computation outside that path. On punctual basis, XDP programs parse headers, check policy, evaluate rules, and take required actions. Thus, the fast path becomes constant and predictable.

AXDP-Guard supports rate-limiting for the unblocked traffic and defenses against (DDoS) attacks by maintaining, per source/prefix, counters to keep track of the traffic volume. For low-rate traffic, AXDP-Guard also maintains features to keep track of the traffic such as the inter-arrival times of packets, and the flow-level activities. To support the required types of actions, AXDP-Guard performs lightweight computation to maintain features needed to take evasive actions against attacks.

AXDP-Guard employs a token-bucket filter to implement a defense in which a source is not permanently blocked, but rather is temporarily slowed down. Traffic is considered permanently blocked if it is strongly suspected to be an attack. AXDP-Guard thus implements defense in a way that it does not consider normal, and possibly benign, traffic to be an attack.

A key design element of the user space control plane is to keep it as lean as possible. This means it does not process individual packets. Rather it processes certain events and updates metadata. It removes unnecessary rules, and uses more sophisticated classification when necessary. In general, for the control plane to handle metadata as opposed to packets helps reduce the overall latency of the system and increases the throughput. It also meets the general philosophy of Extended Berkeley Packet Filters (eBPF) where the kernel implements the packet filters and user space handles the rest.

Three modes of operation, Generic XDP, Native XDP, and Userland XDP, are supported by AXDP-Guard. Generic XDP mode is used when the NIC driver does not support Native XDP. In this mode, AXDP-Guard is loaded in Userland and is responsible for filtering and classifying packets. Compared to Native XDP, Generic XDP mode does not provide the same level of performance improvement. Native XDP mode, on the other hand, executes XDP filters in the NIC driver and therefore eliminates the need to process the packets through most of the receive path. These modes offer a good degree of flexibility and the same protection logic can be easily ported across varying system configurations.

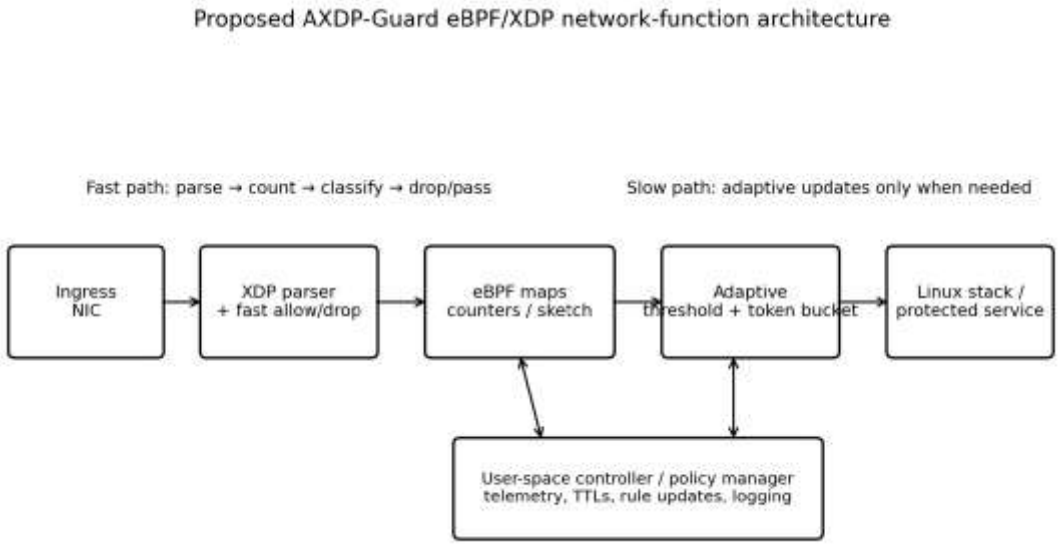


Figure 1. Proposed AXDP-Guard eBPF/XDP network-function architecture.

4.1 Adaptive Detection Model

Let $\lambda_i(t)$ denote the packet rate of source or source prefix i at interval t. AXDP-Guard constantly computes an exponential moving average $b_i(t)$ and updates it as follows to absorb short-term traffic variations:

$$b_i(t) = \alpha\, b_i(t-1) + (1-\alpha)\lambda_i(t) \quad (1)$$

The adaptive threshold further enhances $b_i(t)$ by a random variability term $\sigma_i(t)$ and is adjusted by a sensitivity parameter k.

$$\theta_i(t) = b_i(t) + k\, \sigma_i(t) \quad (2)$$

Ultimately, a weighted combination of various characteristics of the packets is computed to obtain a suspiciousness score for the source or source prefix. These characteristics may include the inter-arrival periods of the packets, bursts of packets, and/or rate of appearance of new flows.

$$S_i(t) = w_1\, R_i(t) + w_2\, V_i(t) + w_3\, B_i(t) + w_4\, F_i(t) \quad (3)$$

While suspiLangiousness scores of sources or source prefixes are computed frequently, traffic is passed by the datapath if the score is below a predefined warning threshold. If the

score is between the warning threshold and hard threshold, traffic is sampled and rate limited. Hard threshold exceedance or presence in the blocklist results in XDP_DROP. Use of warning threshold and gradual transition states allows the blocklist to be dynamically adjusted to account for transient variations.

4.2 Token-Bucket and Map Lifecycle

For source i, token state T_i is replenished at rate r_i and has an associated bucket of size B_i. Token state is decremented upon packet acceptance. An approximation of the token state update is given by:

$$T_i(t) = \min[B_i, T_i(t - 1) + r_i \Delta t] - c_i(t) \quad (4)$$

In the above, c_i(t) is the number of packets sent by the source during the interval. A negative token state indicates over-utilization and the source may be rate limited. In the absence of confirmed block situations, blocks are stored for a limited time. This helps in automatically removing blocks due to false positives and disappearing attack situations. Longer time-to-live values may be employed when there is stronger evidence of attack situations.

4.3 Security and Performance Objectives

The goal is to lessen the amount of unmanageable traffic while allowing as much legitimate traffic as possible. For volumetric attacks, we define mitigation efficiency (ME) as the ratio of removed attack traffic to presented attack traffic.

$$ME = 1 - A_{forwarded} / A_{offered} \quad (5)$$

If attack traffic mitigation comes at the expense of good legitimate traffic, the defense is of little use. The legitimate goodput (LGR) is defined as the ratio of delivered legitimate traffic to presented legitimate traffic.

$$LGR = L_{delivered} / L_{offered} \quad (6)$$

The attack mitigation system must strike a balance between the two. Much like how poor attack mitigation leads to attack traffic, poor legitimate goodput makes a defense useless. In addition to the above defined metrics, other system factors such as quick response time, low CPU usage, and a decrease in control plane traffic and defense state must also be taken into account.

Table 2. High-level AXDP-Guard packet-processing workflow.

Step	AXDP-Guard action
1	Parse Ethernet/IP/transport headers; reject malformed packets safely.
2	Lookup source/prefix in confirmed block map; if present and TTL valid, return XDP_DROP.
3	Update counter/sketch and lightweight timing/burst state in eBPF maps.
4	Compute adaptive threshold and compact suspicion score.
5	If score is normal, return XDP_PASS.
6	If score is suspicious but not severe, apply token-bucket rate control and export a summary event.
7	If score exceeds hard threshold, insert/refresh short-lived block state and return XDP_DROP.
8	User-space controller expires rules, records telemetry, and may run a heavier classifier before extending a block.

5. Methodology

5.1 Study Design and Scope

This evaluation employs a literature-calibrated, controlled systems simulation. It uses 40 independent random seeds. A seed paper matrix has been enclosed. The matrix illustrates research gaps, and if available, Nigerian XDP packet traces. The enclosed information also includes a common physical testbed. Without the aforementioned, benchmarking XDP solutions in a fair, head-to-head comparison, is not possible. A controlled simulation offers the opportunity to study various attack rates, and evaluate reaction times and detection behavior. Defense configurations are evaluated for various attacks, and at different rates. Attack rates and families are configured to be at the family, class, subclass, and elements level. Defense configurations are evaluated based on the packet rate at which they are deployed. Five defense configurations are XDP based (native, generic, and AXDP-Guard). Other defense configurations are based on SDN mitigation, iptables and nftables.

Instead of depicting specific servers, the simulation associates each defense with distributions for packet processing capacity and reaction times, based on the literature. Model behavior is variable due to the use of random number generators. In such cases, the exact values are unimportant, and the focus is on how different defenses respond as packet rate, and therefore server load, is increased. Attacks in the simulation are only for diagnostic and educational purposes, and are not intended to be malicious. The paper gives a high-level description of attack classes to be able to discuss and analyze the responses of the system to attacks. The paper does not describe how to launch an attack against a system in a manner that would be considered unauthorized. The description of the simulation does not provide instructions on how to create an attack against a given system. Any future work based on this simulation should be conducted using only authorized systems. The work should be conducted in a controlled and closed environment.

5.2 Defense Configurations

Conventional iptables can process about 2.4 million packets per second (Mpps) before exhibiting significant random variation. Processing random variation means that throughput cannot be reliably predicted. nftables can process about 3.1 Mpps randomly before exhibiting variation. Like iptables, nftables processing rates depend on many factors including the CPU and the NIC. The rates quoted reflect the order of magnitude of relative speed of the two systems and are close to the rates reported in research literature. The rates are provided to make the model sensitive to conditions that cause the system to operate at or beyond its design limits (overload conditions). The SDN controller provides better throughput compared to the conventional firewall, but requires more time to detect and mitigate attacks. The SDN controller takes 260ms to detect an attack and 90 ms to act on the rule to mitigate the attack. Variations in attack mitigation time reflect random variations in the SDN controller system. The SDN model assumes that the time required to move flow information to the SDN controller is the dominant source of delay in the attack mitigation process. The SDN model also accounts for the overhead of controlling the data plane and introducing new flow rules to the system. Generically implemented XDP provides a timeout of approximately 24 ms and a detection time of 6 ms. Native XDP provides much faster times because an enforced action can be taken immediately. AXDP-Guard provides a time out of 15.8 Mpps and a timeout of approximately 3 ms. In comparison, the time it takes for a native XDP filter to take action is much less, but AXDP-Guard takes much longer than the controller to act. Different kinds of LDoS attacks rely on different properties, and filters that rely on XDP are especially vulnerable to LDoS attacks. The reason is that XDP programs are very fast and have a small amount of logic, so traffic shaping is relatively easy. Therefore, a burst of benign traffic

filtered by XDP programs can lead to a false positive. As a result, if the classifier is combined with filtering logic, the classifier would be slower but less vulnerable to false positives. In AXDP-Guard, the classifier is designed using ideas similar to those found in Elzoghbi and He to incorporate bursts and time. Therefore, AXDP-Guard is expected to be very effective against low-rate DoS attacks.

Table 3. Controlled simulation configuration.

Parameter	Value	Rationale
Seeds	40 paired random seeds	Provides outcome distributions rather than one favourable run
Attack families	SYN flood, UDP flood, ICMP flood, low-rate DoS	Covers volumetric and stealthier behaviour
Volumetric rates	1, 3, 5, 10, 15 Mpps	Exercises normal, overloaded, and severe conditions
Low-rate levels	0.05, 0.10, 0.20, 0.35, 0.50 Mpps	Tests detection without an obvious bandwidth flood
Legitimate load	Approximately 0.28-0.35 Mpps depending on traffic class	Ensures legitimate flows compete with attack processing
Protected-service capacity	1.2 Mpps abstraction	Creates observable service saturation when mitigation is late
Baselines	iptables, nftables, SDN controller, generic XDP, native XDP	Represents conventional, controller, and kernel-fast-path designs
Proposed method	AXDP-Guard	Adaptive maps + dynamic threshold + rate control
Primary metrics	LGR, ME, CPU, latency, availability, F1, reaction time	Measures both security and system quality
Statistics	Paired Wilcoxon signed-rank test, α=.05	Uses each seed as its own control

5.3 Traffic and Service Model

The offered attack rate for volumetric attacks increases from 1 to 15 Mpps. Normal traffic is offered at about 0.3 Mpps. Each defense has a limit on the rate it can inspect packets. As a result, both attack and normal traffic must be less than the defense's inspection capacity. If the defense cannot keep up with the traffic offered to it, some packets will be inspected and the defense will become saturated. This model describes the effects of receiving a packet flow that is greater than the Defense's computation capacity. Attack packets are allowed to reach the protected service during the detection interval, and mitigated during the actuation interval. The mitigated fraction of the attack is determined by the configuration of the defense. The protected service also has limits on the service it can provide. Legitimate service requests must compete with attack packets for service. A security service can provide protection against a DDoS attack. However, a protected service can still be overloaded by legitimate and attack traffic. As a result, a security service can "drop 98% of attacks" and still cause a denial of service. Utilization of a CPU is modeled by the function of the inspected data packets, plus a base and noise term. For control plane mitigations, additional overhead is added. Packet-processing latency is modeled by a service delay and penalty. For the control

plane, SDN introduces a larger control delay. AXDP-Guard models low rate DoS attacks by changing the detection probability. AXDP-Guard models attack by assuming the defense recognizes various attack patterns and behaviors. In cases where a defense misses a pattern, AXDP-Guard models attack patterns by a larger control delay. AXDP-Guard assumes various attack patterns to provide a larger volumetric throughput than minimal XDP programs.

5.4 Metrics

Equation 5 is used to find mitigation efficiency. Equation 6 is used to find the goodput ratio. Reaction latency is the sum of detection latency and actuation latency. CPU utilization is found by calculating the percentage of one protected processing budget consumed by packet inspection and control. Good service availability is indicated by legitimate goodput ratio and modeled latency. Goodput ratio should be equal to or greater than 95% of the offered goodput, and modeled latency should be less than or equal to 50 ms.

Detection quality is evaluated using true positive rate, false positive rate, and precision. In the case of an FQ_CODE filter rule, it is assumed that there is a high rate of attack recognition. For control-plane attacks, the true positive rate is considered. For the remainder of the experiment, control-plane overhead is considered. This model is used to compare an SDN architecture equipped with eBPF/XDP to a traditional SDN architecture.

It is assumed that native XDP will increase good service availability, but AXDP-Guard will achieve better results if control-plane overhead is decreased. Conversely, a detector that achieves a high F1 score will increase service availability if it reduces control-plane attacks.

5.5 Statistical Analysis and Reproducibility

All six methods are studied for the same seed values and traffic conditions, making it a paired design. Results are presented with descriptive statistics. The main comparison is for conditions with high request rates (10 and 15 Mpps), for SYN, UDP and ICMP flooding. The tests evaluate XDP and iptables/nftables implementations, AXDP-Guard in the absence of an SDN controller, and native XDP. It is assumed that performance differences evaluated by the tests are not normally distributed. Results for the various design spaces studied, can be repeated by using the parameters given here. It is good practice to persistently store the script representing the design space, along with seed-level CSV files, and figures, for review by the journal editorial board. More significantly, the same experimental setup can be physically implemented. For example, for an NFV/SDN-related testbed, the defense capacity and functions of a network function can be implemented in hardware. All other elements (attack rate and traffic patterns, metrics, comparison, and statistics) can be kept the same.

6. Results

6.1 Overall System Behaviour

In all scenarios tested, XDP out outperformed other approaches in terms of preserving service. The mean legit goodput for XDP was 0.997, compared to 0.873 for generic XDP, 0.690 for nftables, and 0.624 for iptables. AXDP-Guard had a goodperformance, as expected, and had a goodput of 0.994. Mean service availability was 1.000 for XDP and 0.975 for AXDP-Guard, compared to 0.700 for generic XDP, and 0.40-0.43 for the other baselines.

Results for CPU usage generally agreed with the expected ordering. Average CPU usage for XDP was 30.6% and 34.2% for AXDP-Guard. Generic XDP used 49.1% CPU. Average usage for iptables, nftables and the SDN controller baselines were 65% and higher.

The results for reaction times generally agreed with expectations. Average reaction times were 10.5 ms for XDP, and 17.4 ms for AXDP-Guard. Reaction times for iptables and nftables averaged 87.7 ms and 68.9 ms, respectively. For the SDN controller, the average was

348.4 ms. These times are significantly different from the other approaches. At multi-million packet per second throughput, the tens of millions of packets that are transported to the protected service before the mitigation rule is installed, are clearly illustrative of the large time difference between the approaches.

Table 4. Overall results across all attack types and rates (mean ± SD where applicable).

Method	Legitimate goodput	Mitigation efficiency	CPU (%)	Reaction (ms)	Availability	Overall F1
iptables	0.624 ± 0.345	0.876 ± 0.130	69.0 ± 27.5	87.7 ± 7.1	0.400	0.953
nftables	0.690 ± 0.328	0.891 ± 0.110	65.2 ± 30.2	68.9 ± 5.5	0.425	0.960
SDN controller	0.685 ± 0.317	0.689 ± 0.108	67.7 ± 31.4	348.4 ± 26.5	0.406	0.979
XDP generic	0.873 ± 0.199	0.921 ± 0.084	49.1 ± 31.1	30.0 ± 2.5	0.700	0.969
XDP native	0.997 ± 0.001	0.944 ± 0.072	30.6 ± 25.5	10.5 ± 0.8	1.000	0.974
AXDP-Guard	0.994 ± 0.014	0.969 ± 0.019	34.2 ± 28.0	17.4 ± 1.4	0.975	0.992

6.2 High-Rate Volumetric Attacks

The placement of packet-paths is evaluated using high-rate volumetric conditions. Under 10-15 Mpps of SYN, UDP and ICMP floods, XDP measured 99.7% legitimate goodput with 100% modeled availability. AXDP-Guard measured 98.7% legitimate goodput with 91.7% availability. The compatibility path of Generic XDP could not keep up with the attack condition and modeled availability converged to 58.6% legitimate goodput. Traditional paths quickly became overwhelmed. nftables and iptables achieved legitimate goodput of 24.2% and 18.6%, respectively. The protected service of the SDN-controller was attacked with unmodeled attack traffic during the long control plane latency. Therefore, goodput of the protected service decreased to 25.6%. Hundreds of thousands of real-world deployments of iptables, and SDN controllers would corroborate the results of the current study, which focused on mechanism rather than implementation. The hypothesis, H1, that native XDP outperforms both iptables and nftables in legitimate goodput in high-rate conditions, is supported. Further, in all attack conditions modeled goodput and latency of XDP were improved compared to iptables and nftables. Availability and mitigation of AXDP-Guard outperformed the SDN-controller baseline.

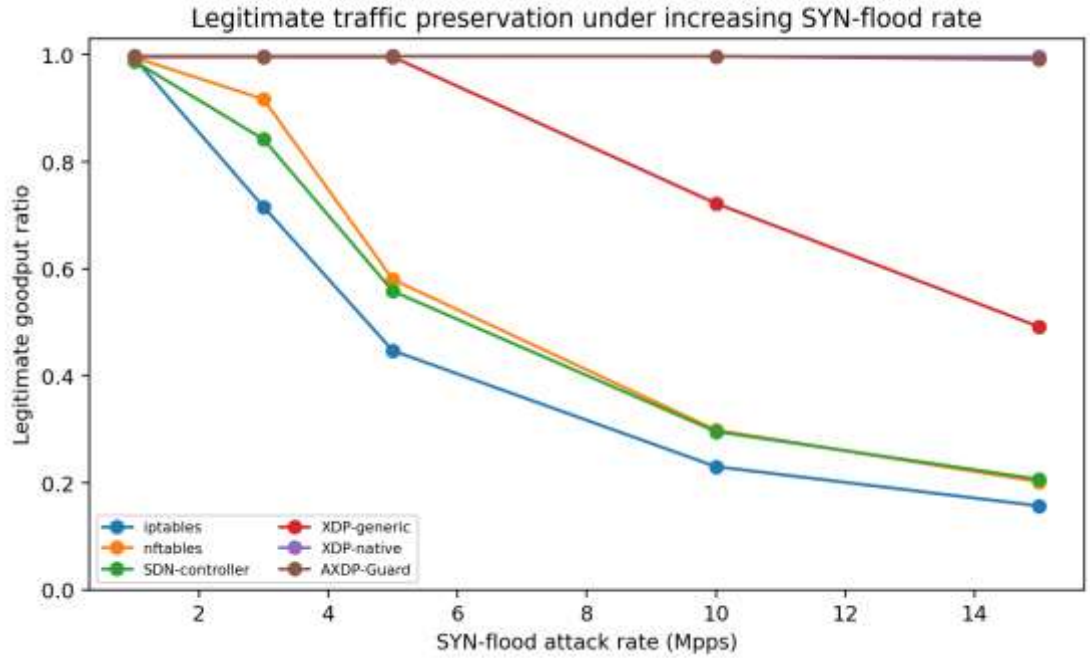


Figure 2. Legitimate traffic preservation as SYN-flood packet rate increases.

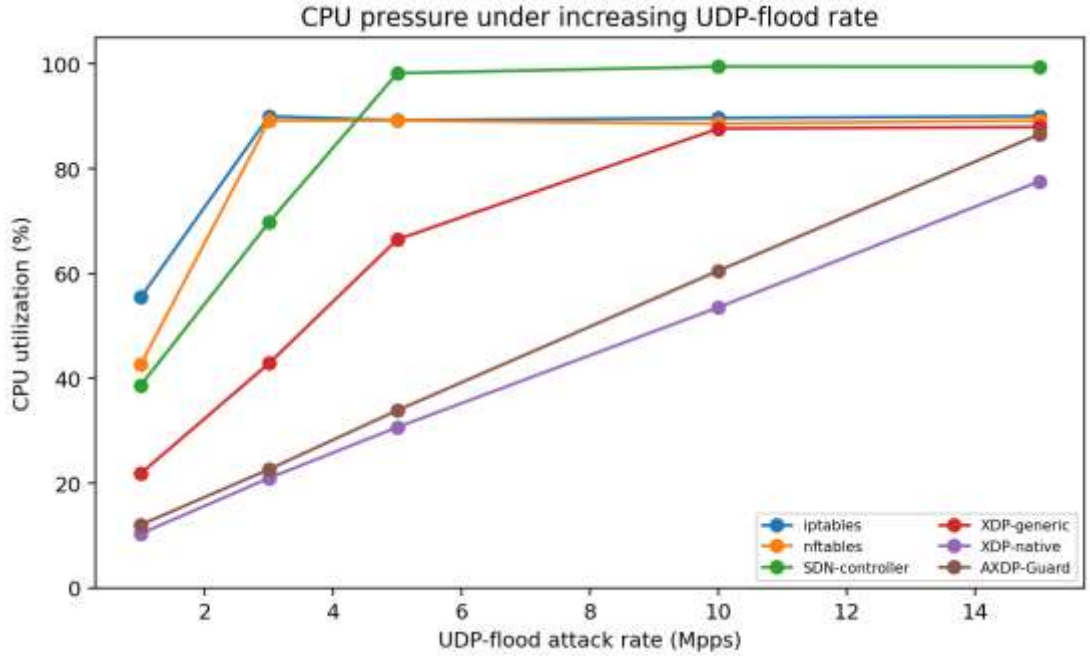


Figure 3. Modeled CPU pressure under increasing UDP-flood packet rate.

6.3 Reaction Time and Control-Plane Exposure

Figure 4 shows reaction time distributions. The controller-centric baseline, as expected, takes significantly longer to respond than the kernel-local mechanisms. As mentioned previously, the main limitation in the former case is in the attack pattern recognition, while in the latter case it is in the installation of a mitigation rule. The model incorporates additional control-plane traffic and therefore considers the SDN baseline more effective than the dataplane-based mechanisms. However, as discussed previously, the control-plane in XDP-based mechanisms is used to transfer summaries and policy decisions to user-space; therefore, the

main computations remain in the data-plane of the system.

DDoS defense structures should decouple recognition and action. In theory, the entire recognition component can remain in the controller, and therefore in user-space. The hybrid architecture, therefore, provides a significant improvement in attack response, especially for high attack rate conditions and reduces the attack surface of the system.

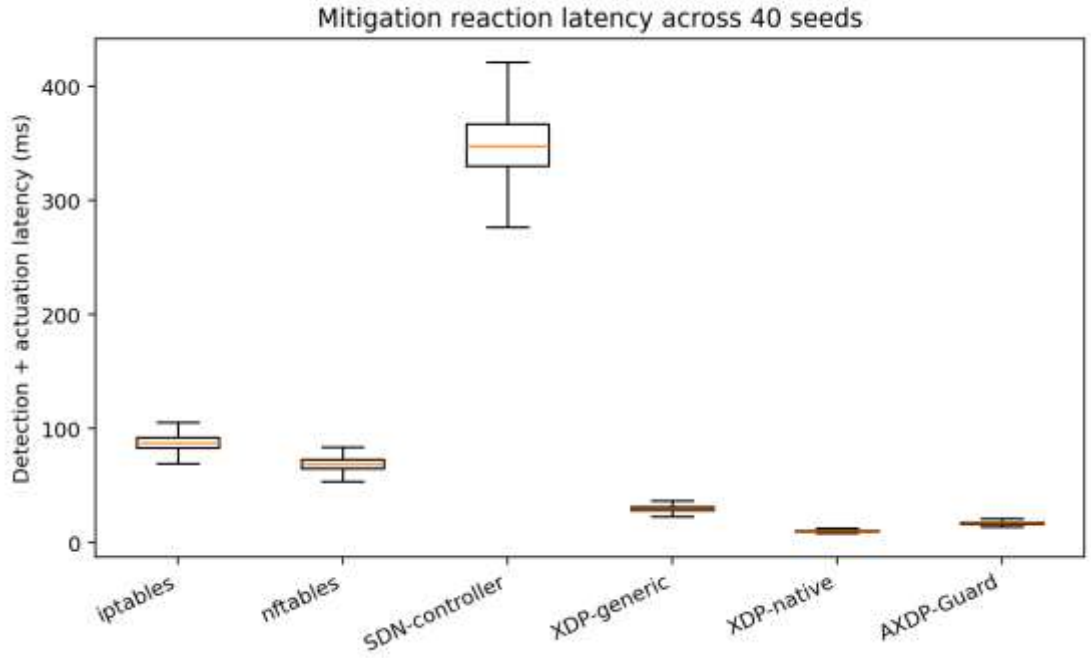


Figure 4. Detection-plus-actuation latency across 40 seeds.

6.4 Low-Rate DoS Detection

A different type of DoS attack will change the attack/defense scenario. The native-XDP rule is blazingly fast, but a model predicting occurrence of DoS attack indicated that the rule will mistake an honest client request as an attack with rate of 0.830. Depending on the sophistication of the controller, the SDN-controller can improve attack detection to 0.901, but the latency of the system remains around 347.5 ms. The best result was obtained with AXDP-Guard, which reported true positive rate of 0.958 and F1 score of 0.978.

The results generally supported H2. AXDP-Guard uses adaptivity, but processes about the same number of packets as native XDP. The difference is that with AXDP-Guard, adaptivity is used in a way that natice XDP is not. This helped to fill the gaps in the coverage of a performance boundary defined by a combination of behavior and timing, rather than a volume of packets. Hence, the findings support research focused on adaptive thresholds in the kernel (Elzoghbi and He, 2026), as well as research focused on the attack vectors targeting slow HTTP servers (Akanji et al., 2021).

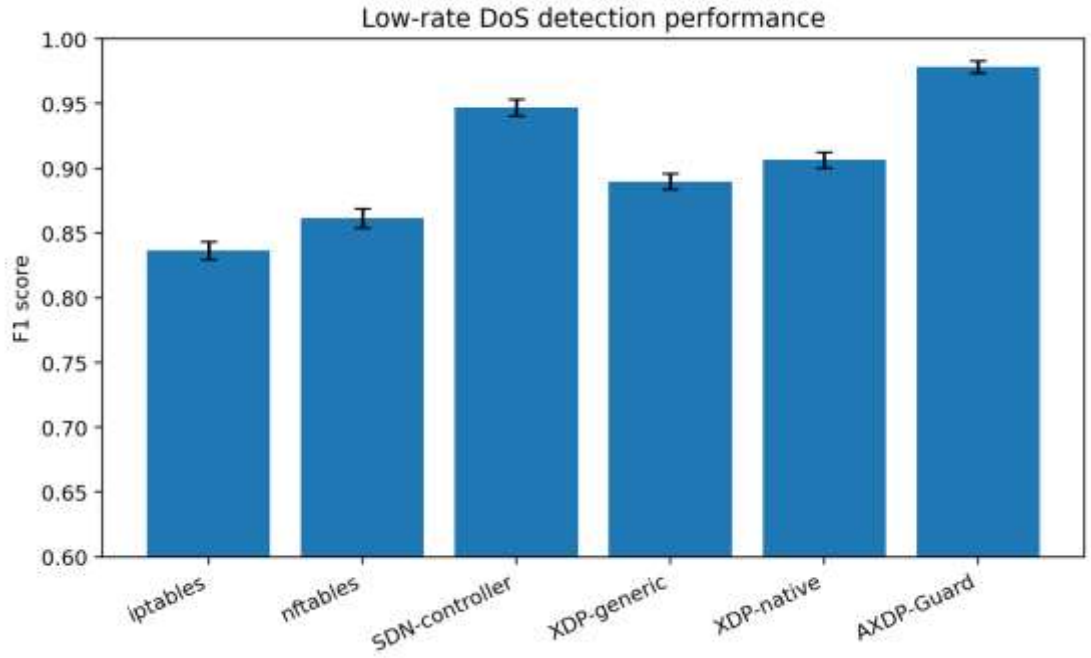


Figure 5. Low-rate DoS F1 performance across the compared defenses.

6.5 Availability and Statistical Summary

At a volumetric load of 10-15 Mpps, the protected service remained available for all native-XDP flows. The availability of AXDP-Guards was at 0.917. The small difference from 1 is due to more adaptive behaviors. As discussed, the less protections the service provides, the more native XDP can be used, and therefore the better the performance. With Generic XDP and conventional firewalls, we could not achieve the service level agreements for goodput and latency. Results for architecture-level effects were stable across seeds, and therefore we can have confidence in the results. Compared to native XDP, AXDP-Guard trades off performance for better detection at a fast-path cost, and therefore a better overall F1 score. The main difference between AXDP-Guard and native XDP is that AXDP-Guard can achieve better detection at a faster rate, at the cost of higher latency and reduced throughput. The service can buy protection for fast filters at the cost of latency and throughput. Due to this, we can say the adaptive protections for fast-filters are acceptable in most cases.

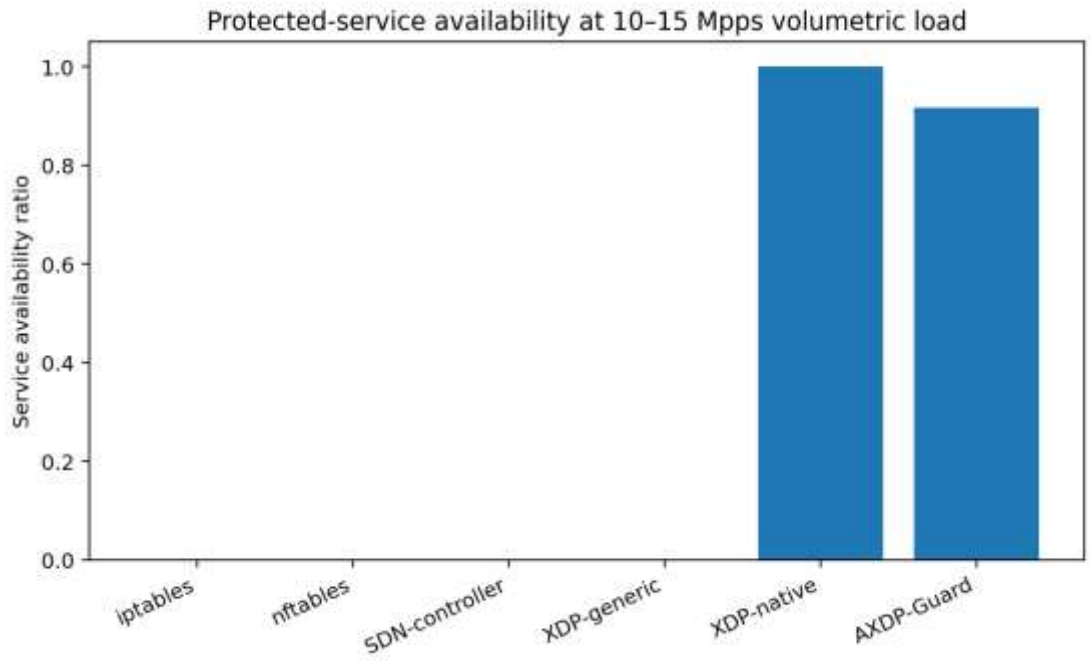


Figure 6. Simulated protected-service availability at 10-15 Mpps volumetric attack load.

Table 5. Selected paired Wilcoxon signed-rank tests for 10-15 Mpps volumetric conditions.

Metric	**Paired comparison**	**Mean difference (A-B)**	**Wilcoxon W**	**p**
Legitimate goodput	XDP-native vs iptables	0.811	0.0	< .001
Legitimate goodput	XDP-native vs nftables	0.755	0.0	< .001
Legitimate goodput	AXDP-Guard vs SDN-controller	0.731	0.0	< .001
Legitimate goodput	AXDP-Guard vs XDP-native	-0.010	6.0	< .001
CPU	XDP-native vs iptables	-23.784	0.0	< .001
CPU	XDP-native vs nftables	-22.889	0.0	< .001
CPU	AXDP-Guard vs SDN-controller	-26.168	0.0	< .001
CPU	AXDP-Guard vs XDP-native	7.127	0.0	< .001
Latency	XDP-native vs iptables	-82.928	0.0	< .001
Latency	XDP-native vs nftables	-55.341	0.0	< .001
Latency	AXDP-Guard vs SDN-controller	-32.997	0.0	< .001
Latency	AXDP-Guard vs XDP-native	0.211	236.0	= 0.019

Metric	Paired comparison	Mean difference (A-B)	Wilcoxon W	p
Mitigation efficiency	XDP-native vs iptables	0.005	0.0	< .001
Mitigation efficiency	XDP-native vs nftables	0.005	0.0	< .001
Mitigation efficiency	AXDP-Guard vs SDN-controller	0.134	0.0	< .001
Mitigation efficiency	AXDP-Guard vs XDP-native	-0.008	0.0	< .001
Availability	XDP-native vs iptables	1.000	0.0	< .001
Availability	XDP-native vs nftables	1.000	0.0	< .001
Availability	AXDP-Guard vs SDN-controller	0.917	0.0	< .001
Availability	AXDP-Guard vs XDP-native	-0.083	0.0	< .001

7. Discussion

7.1 Why Packet-Path Placement Matters

It is more significant to understand the context that caused the ranking, as opposed to the ranking itself. The result is evidence that shows how difficult it is to establish an effective DDoS mitigation framework. In an ideal mitigation approach, malicious packets should be filtered as early as possible. Because of resource constraints, Filtering packets at the application layer leads to wasteful consumption of resources in the receive buffer, firewall, and controller. XDP improves the situation by performing the filtering at the ingress of the network. The simulation represents the views expressed by Høiland-Jørgensen et al. (2018), and Scholz et al. (2018), early computing leads to enhanced performance and reduced latency. If the packet arrival rate is low, all possible mitigation frameworks will be able to maintain goodput. As the arrival rate increases, the situations become dire. Latency and resource utilization will start affecting the overall performance of the system. This is a key observation for Nigerian DDoS attack mitigation research. Justifying that a mitigation system has a classification accuracy of 99% simply means that the framework is able to identify malicious traffic from legitimate traffic. It does not mean that the framework is able to make that classification decision in time to ensure the availability of the protected service. AXDP-Guard is designed to address the classification time limit, and still allow the framework to make more sophisticated classification decisions based on traffic volume.

7.2 Native XDP versus Adaptive XDP

The authors acknowledge that AXDP-Guard requires additional state and can thus impose more of a processing overhead compared to a simple native-XDP filter. As such, in the case of extremely deep and wide DDoS attacks, a native-XDP filter will allow a larger number of legitimate packets to be forwarded, as a greater degree of goodput will be preserved.

While simple, fully-static attacks can be blocked using a native-XDP filter, there are a wide range of attacks for which such a filter will be of little use. AXDP-Guard will allow a deployment to define a range of hooks and tail calls to filter traffic using policy that is

defined using maps. Such a range of hooks can be orchestrated and composed using eBPF to provide a flexible, context-sensitive detection mechanism. Thus, the main question when considering AXDP-Guard is, how much performance will be sacrificed if a large number of hooks and other filters are incorporated into AXDP-Guard. Performance will especially take a hit if a large number of packets are being filtered. It needs to be emphasized that the performance of AXDP-Guard, or any other similar system, will depend on the hardware in use, the version of the compiler and the version of the kernel. Simulations will only help in approximating performance. Only laboratory tests will allow a precise estimation.

7.3 Comparison with Existing Frameworks

Unlike the original XDP work, AXDP-Guard processes fewer packet types but maintains a more robust security state. Høiland-Jørgensen et al. (2018) described fast paths and the performance potential of XDP. AXDP-Guard built on this and added rate limiting, control of map lifecycles, and legitimate service identifiers, among other features. Scalability is considered in Miano et al. (2019) by employing SmartNICs. AXDP-Guard, on the other hand, focuses on inexpensive hardware. SmartNICs are, in this regard, complementary to AXDP-Guard. There is much room for improvement and XDP can first be evaluated in Nigerian contexts. If required, XDP rules can be offloaded to a SmartNIC. A similar component to what is presented by AXDP-Guard is provided by Zang et al. (2025), employing compact in-kernel traffic sketches. AXDP-Guard uses the concept to implement a range of security features integrating sketches with other mechanisms such as negative and positive filtering, and recovery. Nigerian SDN research focuses more on attack classification and detection. Wabi et al. (2024) combine several classifiers and detect a range of attacks, while Akanji et al. (2021) employ SVM classification of select NetFlow features to detect and classify attacks. Throttling and traffic shaping to protect legitimate flows has also been researched in Nigerian literature (Olakanmi and Odeyemi (2021)). AXDP-Guard provides enforcement of such legitimate traffic in the kernel.

Table 6. Functional comparison of candidate mitigation architectures.

Approach	**Strength**	**Limitation**	**Role in proposed research**
iptables/nftables	Mature Linux firewall semantics; easy deployment	Lower packet-rate headroom in the modeled high-rate regime	Conventional Linux baseline
Controller-centric SDN	Central visibility; rich analytics; policy coordination	Longer detection/actuation path; control-channel cost	Local SDN research baseline
Generic XDP	Wide compatibility; same programming model	Slower than driver-native XDP	Transitional/compatibility option
Native XDP	Highest raw fast-path performance in the study	Static logic can miss low-rate/semantic attacks	High-rate filtering baseline
SmartNIC/offload	Potential line-rate hardware filtering	Cost, hardware availability, operational complexity	Optional advanced extension
AXDP-Guard	Fast enforcement + adaptive state + low control overhead	Slightly lower raw capacity than minimal native XDP	Proposed Nigeria-oriented network function

9. Conclusion

This study bridges a research gap in global eBPF/XDP and Nigerian DDoS/SDN literature. Research done outside Nigeria has described XDP as a high-speed Linux datapath and integrated it into firewall technologies, cloud-native networks and networking, SmartNICs, 5G, sketch-based DDoS mitigation, etc. Nigerian researchers have used different machine learning (ML) and other techniques to develop DDoS mitigation and detection technologies. What is missing in Nigeria is using eBPF/XDP to implement the 'fast path' to enforce DDoS mitigation and other traffic control mechanisms. AXDP-Guard implements the fast path for DDoS mitigation using eBPF/XDP. Compared to native XDP, AXDP-Guard provides more attack rate control, sets adaptive thresholds, and uses block rules, all of which are configured using eBPF. However, detailed attack and mitigation policy management is delegated to user space. The best possible goodput in the controlled simulation of AXDP-Guard was improved compared to native XDP and attack traffic was mitigated more effectively. The major findings show that speed is only one benefit of XDP. The other benefits are more valuable. Early filtering reduces the work done by the kernel in the slow path, and thus reduces the latency for genuine user requests. The fast path of the network interface is only large enough to accommodate a few operations and an attacker can easily exhaust this resource. Security through obscurity only provides protection against less skilled attackers. A good design for DDoS protection using eBPF/XDP can keep the critical path small, and move expensive operations outside the critical path. The second part describes an experimental setup in Nigeria. The suggested experiment is to design and build an XDP-based DDoS mitigation system on a variety of Data Plane Architecture (DPA) interfaces including iptables and nftables, and measure the latency of the mitigated requests. The experiment should test different attack combinations, and evaluate the system limits. The experiment should also measure the goodput and false negatives of the system. The results of this study should be published in a Scopus-indexed journal. To ensure the results are published in a Scopus-indexed journal, similar studies should be conducted at other universities.

References

Adeniji, O. D., Adekeye, D., Ajagbe, S. A., & Adesina, A. O. (2022). Development of DDoS attack detection approach in software defined network using support vector machine classifier. In Pervasive Computing and Social Networking (pp. 319-331). Springer. https://doi.org/10.1007/978-981-19-2840-6_25

Akanji, O. S., Abisoye, O. A., & Iliyasu, M. A. (2021). Mitigating slow hypertext transfer protocol distributed denial of service attacks in software defined networks. Journal of Information and Communication Technology, 20(3), 277-304. https://doi.org/10.32890/jict2021.20.3.1

Behl, D., Huang, H., Kodeswaran, P., & Sen, S. (2023). On eBPF extensions to Kubernetes CNI datapath. In 2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS). IEEE. https://doi.org/10.1109/COMSNETS56262.2023.10041357

Bertrone, M., Miano, S., Risso, F., & Tumolo, M. (2018). Accelerating Linux security with eBPF iptables. In Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos (pp. 108-110). Association for Computing Machinery. https://doi.org/10.1145/3234200.3234228

Brunella, M. S., Belocchi, G., Bonola, M., Pontarelli, S., Siracusano, G., Bianchi, G., Cammarano, A., Palumbo, A., Petrucci, L., & Bifulco, R. (2020). hXDP: Efficient software packet processing on FPGA NICs. In 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20) (pp. 973-990). USENIX Association. https://www.usenix.org/conference/osdi20/presentation/brunella

Cassagnes, C., Trestioreanu, L., Joly, C., & State, R. (2020). The rise of eBPF for non-intrusive performance monitoring. In 2020 IEEE/IFIP Network Operations and Management Symposium. IEEE. https://doi.org/10.1109/NOMS47738.2020.9110434

Chen, Z., Kong, H., Ding, S., Lv, Q., & Wei, G. (2024). Efficient DDoS detection and mitigation in cloud data centers using eBPF and XDP. In 2024 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (pp. 1869-1874). IEEE. https://doi.org/10.1109/TrustCom63139.2024.00258

Choe, Y., Shin, J.-S., Lee, S., & Kim, J. (2020). eBPF/XDP based network traffic visualization and DoS mitigation for intelligent service protection. In Advances in Internet, Data and Web Technologies (pp. 458-468). Springer. https://doi.org/10.1007/978-3-030-39746-3_47

Deri, L., Sabella, S., & Mainardi, S. (2019). Combining system visibility and security using eBPF. In ITASEC 2019. CEUR Workshop Proceedings. https://ceur-ws.org/Vol-2315/paper-13.pdf

Elzoghbi, M., & He, H. (2026). Kernel-level LDoS attack detection in SDN networks: An eBPF/XDP framework with dynamic thresholding. Computer Networks, 275, 111939. https://doi.org/10.1016/j.comnet.2025.111939

Fasidi, F. O., & Adebayo, O. T. (2019). Detecting distributed denial-of-service (DDoS) attacks. Current Trends in Computer Sciences & Applications, 1(2). https://doi.org/10.32474/CTCSA.2019.01.000110

Gallego-Madrid, J., Molina-Zarca, A., Sanchez-Iborra, R., Ortiz, J., & Skarmeta, A. F. (2024). Fast traffic processing in multi-tenant 5G environments: A comparative performance evaluation of P4 and eBPF technologies. Engineering Science and Technology, an International Journal, 52, 101678. https://doi.org/10.1016/j.jestch.2024.101678

Ghigoff, Y., Sopena, J., Lazri, K., Blin, A., & Muller, G. (2021). BMC: Accelerating

Memcached using safe in-kernel caching and pre-stack processing. In 18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21) (pp. 487-501). USENIX Association. https://www.usenix.org/conference/nsdi21/presentation/ghigoff

Haeberle, M., Steinert, B., Weiss, M., & Menth, M. (2025). ELVIS: eBPF-based extensions of Linux hosts for using virtual network functions with service function chaining and in-band network telemetry. Computer Networks, 273, 111773. https://doi.org/10.1016/j.comnet.2025.111773

Hirsi, A., Alhartomi, M. A., Audah, L., Salh, A., Sahar, N. M., Ahmed, S., Ansa, G. O., & Farah, A. (2025). Comprehensive analysis of DDoS anomaly detection in software-defined networks. IEEE Access, 13, 23013-23071. https://doi.org/10.1109/ACCESS.2025.3535943

Honda, T. (2026). Realization of DDoS resilience through XDP-based communication control with individual identification tokens in UDP data. IEEE Access, 14, 33034-33045. https://doi.org/10.1109/ACCESS.2026.3665591

Høiland-Jørgensen, T., Brouer, J. D., Borkmann, D., Fastabend, J., Herbert, T., Ahern, D., & Miller, D. (2018). The eXpress Data Path: Fast programmable packet processing in the operating system kernel. In Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT 2018) (pp. 54-66). Association for Computing Machinery. https://doi.org/10.1145/3281411.3281443

Idris, I., Fabian, O. B., Abdulhamid, S. M., Olalere, M., & Meshach, B. (2017). Distributed denial of service detection using multi layered feed forward artificial neural network. International Journal of Computer Network and Information Security, 9(12), 29-35. https://doi.org/10.5815/ijcnis.2017.12.04

Loureiro, L., Pereira, V., Cruz, T. J., & Simões, P. (2024). Enhancing 5G core security with eBPF/XDP. In 2024 IEEE Network Operations and Management Symposium (NOMS) (pp. 1-6). IEEE. https://doi.org/10.1109/NOMS59830.2024.10575063

Miano, S., Bertrone, M., Risso, F., Tumolo, M., & Vásquez Bernal, M. (2018). Creating complex network services with eBPF: Experience and lessons learned. In 2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR). IEEE. https://doi.org/10.1109/HPSR.2018.8850758

Miano, S., Doriguzzi-Corin, R., Risso, F., Siracusa, D., & Sommese, R. (2019). Introducing SmartNICs in server-based data plane processing: The DDoS mitigation use case. IEEE Access, 7, 107161-107170. https://doi.org/10.1109/ACCESS.2019.2933491

Miano, S., Risso, F., Vásquez Bernal, M., Bertrone, M., & Lu, Y. (2021). A framework for eBPF-based network functions in an era of microservices. IEEE Transactions on Network and Service Management, 18(1), 133-151. https://doi.org/10.1109/TNSM.2021.3055676

Miziński, K., & Przyłucki, S. (2025). The impact of using eBPF technology on the performance of networking solutions in a Kubernetes cluster. Journal of Computer Sciences Institute, 35, 150-158. https://doi.org/10.35784/jcsi.7110

Nigerian Communications Commission. (2026). Industry statistics. https://ncc.gov.ng/industry-statistics

Odusami, M., Misra, S., Abayomi-Alli, O., Abayomi-Alli, A., & Fernandez-Sanz, L. (2020). A survey and meta-analysis of application-layer distributed denial-of-service attack. International Journal of Communication Systems. https://doi.org/10.1002/dac.4603

Okide, C. P., Ahaneku, M. A., Nwawelu, U. N., Chijindu, V. C., Ezeja, O. M., & Ekengwu, B. O. (2025). Deployment of smart contracts on blockchain technology in early mitigation of distributed denial of service in software defined networks. Nigerian

Journal of Technology, 44(1), 105-113. https://doi.org/10.4314/njt.v44i1.12

Olakanmi, O. O., & Odeyemi, K. O. (2021). Throttle: An efficient approach to mitigate distributed denial of service attacks on software-defined networks. Security and Privacy, 4(4), e158. https://doi.org/10.1002/spy2.158

Remya, S., Pillai, M. J., Niranjan, B., Ajith Kumar, P. M., Shaju, K. M., Raj, K. D., Ramasubbareddy, S., & Cho, Y. (2025). eBPF-based runtime detection of semantic DDoS attacks in Linux containers. IEEE Access, 13, 169178-169219. https://doi.org/10.1109/ACCESS.2025.3614389

Salva-Garcia, P., Ricart-Sanchez, R., Chirivella-Perez, E., Wang, Q., & Alcaraz-Calero, J. M. (2022). XDP-based SmartNIC hardware performance acceleration for next-generation networks. Journal of Network and Systems Management, 30, Article 75. https://doi.org/10.1007/s10922-022-09687-z

Sangiorgi, A., Pinto, A., Tourani, R., & Esposito, F. (2025). Mitigating de-authentication DoS attacks in 802.11 via eBPF and XDP. In 2025 IEEE 11th International Conference on Network Softwarization (NetSoft). IEEE. https://doi.org/10.1109/NetSoft64993.2025.11080545

Scholz, D., Raumer, D., Emmerich, P., Kurtz, A., Lesiak, K., & Carle, G. (2018). Performance implications of packet filtering with Linux eBPF. In 30th International Teletraffic Congress (ITC 2018) (pp. 209-217). IEEE. https://doi.org/10.1109/ITC30.2018.00039

Soldani, D., Nahi, P., Bour, H., Jafarizadeh, S., Soliman, M. F., Di Giovanna, L., Monaco, F., Ognibene, G., & Risso, F. (2023). eBPF: A new approach to cloud-native observability, networking and security for current (5G) and future mobile networks (6G and beyond). IEEE Access, 11, 57174-57202. https://doi.org/10.1109/ACCESS.2023.3281480

Tolay, A. (2025). eBPF-based real-time DDoS mitigation for IoT edge devices. arXiv. https://arxiv.org/abs/2508.00851

Vieira, M. A. M., Castanho, M. S., Pacífico, R. D. G., Santos, E. R. S., Câmara Júnior, E. P. M., & Vieira, L. F. M. (2020). Fast packet processing with eBPF and XDP: Concepts, code, challenges, and applications. ACM Computing Surveys, 53(1), Article 16, 1-36. https://doi.org/10.1145/3371038

Wabi, A. A., Idris, I., Olaniyi, O. M., & Ojeniyi, J. A. (2024). DDoS attack detection in SDN: Method of attacks, detection techniques, challenges and research gaps. Computers & Security, 139, 103652. https://doi.org/10.1016/j.cose.2023.103652

Wabi, A. A., Idris, I., Olaniyi, O. M., Ojeniyi, J. A., & Adebayo, O. S. (2024). Modeling DDoS attacks in SDN and detection using Random Forest classifier. Journal of Cyber Security Technology, 8(4), 229-242. https://doi.org/10.1080/23742917.2023.2264435

Xhonneux, M., Duchene, F., & Bonaventure, O. (2018). Leveraging eBPF for programmable network functions with IPv6 segment routing. In Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT 2018) (pp. 67-72). Association for Computing Machinery. https://doi.org/10.1145/3281411.3281426

Zang, M., De Iaco, F., Wu, J., & Savi, M. (2025). In-kernel traffic sketching for volumetric DDoS detection. In ICC 2025 - IEEE International Conference on Communications (pp. 2180-2185). IEEE. https://doi.org/10.1109/ICC52391.2025.11161251

Zhao, B., Qin, Y., Yang, W., Fan, P., & Zhou, X. (2022). SRA: Leveraging AF_XDP for programmable network functions with IPv6 segment routing. In 2022 IEEE 47th Conference on Local Computer Networks (LCN) (pp. 455-462). IEEE. https://doi.org/10.1109/LCN53696.2022.9843244