Adaptive Mitigation of Amplification-Based DDoS Attacks in Programmable Data Planes

Simone Sampognaro[2], Marios Avgeris[1], Anestis Dalgkitsis[1], Paola Grosso[1]
[1] University of Amsterdam, LAB42, Science Park 900, 1098 XH Amsterdam, The Netherlands
[2] Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy
Email: simone.sampognaro@studenti.polito.it, {m.avgeris@uva.nl, a.dalgkitsis, p.grosso}@uva.nl

***Abstract*—Distributed Denial-of-Service (DDoS) attacks, particularly amplification-based variants, continue to threaten Internet availability by generating massive traffic volumes through protocol asymmetries. Conventional mitigation approaches based on static filtering or control-plane inspection are increasingly inadequate against dynamic attack behaviors. In response, this work investigates whether online learning-based mitigation can be executed directly in the programmable data plane using P4. We design an in-switch Q-learning agent that performs amplification-aware mitigation by updating its policy online from bidirectional flow telemetry, using traffic asymmetry features and a switch-level quality-of-service degradation metric, without external retraining or offline datasets. We implement the approach in P4 and evaluate it in an emulated environment with programmable software switches under synthetic amplification attack and legitimate traffic scenarios. Results indicate that the proposed system adapts to changing traffic conditions and reduces malicious amplification traffic while preserving legitimate service performance, demonstrating the potential of programmable data planes for autonomous DDoS mitigation.**

***Index Terms*—DDoS Mitigation, Programmable Data Planes, In-Network Machine Learning, Reinforcement Learning, P4**

I. Introduction

Distributed Denial-of-Service (DDoS) attacks continue to represent one of the most severe threats to Internet security and stability [1]. Their core objective is to overwhelm the resources of a target, whether computing power, memory, or bandwidth until legitimate users can no longer access services [2]. The open nature of the Internet makes such attacks particularly difficult to prevent, as attackers can generate or redirect malicious traffic from distributed sources with relative ease, exploiting available infrastructure and tools to disrupt even well-protected organizations [3].

The scale and intensity of DDoS attacks have grown significantly in recent years. In early 2025, tens of millions of attacks were reported, already surpassing total volumes from the previous year, with campaigns reaching multi-terabit-per-second levels and extreme cases exceeding 7 Tbps [4]–[6]. Institutional reports further confirm DDoS as the leading availability threat, impacting both enterprises and national infrastructures [7]. This escalation is driven by large-scale botnets, such as those comprising tens of thousands of compromised IoT devices, and increasingly by politically motivated campaigns targeting public-sector entities [8]. Collectively, these trends highlight the evolution of DDoS into a persistent, large-scale, and systemic threat, exposing the limitations of static mitigation and motivating adaptive, real-time defenses.

Recent work on DDoS detection has increasingly explored In-Network Machine Learning (IN-ML) using P4-enabled Programmable Data Planes (PDPs) to identify attacks directly within switches [3]. Allowing custom packet parsing, processing, and forwarding at line rate, P4 enables the integration of monitoring and security functions into the data plane, facilitating Tbps-scale mitigation while reducing reliance on CPU/GPU-based systems and limiting exposure of backend services. However, strict line-rate constraints impose significant limitations: only a restricted set of operations is supported, preventing the deployment of complex models. As a result, PDPs are typically limited to inference, while training remains offline or control-plane-based, and simplified feature extraction introduces inherent precision–recall trade-offs. Motivated by the above and in an effort to overcome the PDP limitations, we design an in-switch Q-learning agent that performs amplification-aware mitigation by updating its policy online. Our contribution is fourfold:

- We propose, to the best of our knowledge, the first approach that performs amplification DDoS mitigation entirely within the PDP using online Reinforcement Learning (RL), without reliance on control-plane support or offline datasets.
- We develop an RL agent that operates within the packet processing pipeline, using only features extractable at line rate, enabling scalable, low-latency mitigation directly in the forwarding path.
- We introduce a novel state formulation combining bidirectional traffic asymmetry metrics with a switch-level Quality-of-Service (QoS) degradation indicator, enabling online, context-aware mitigation decisions.
- We experimentally demonstrate the advantages of adaptive in-switch learning over threshold-based filtering, highlighting improved flexibility and effectiveness under dynamic traffic conditions.

The rest of the paper is structured as follows: Section II is a collection of related works to support the basis of our work. Section III provides an overview of the system model. Section IV presents in detail the proposed amplification-based DDoS attack adaptive mitigation solution. Section V showcases the experimental setup from both software and

hardware perspective. Section VI analyzes the experimental evaluation of the proposed approach under different attack scenarios. Finally, Section VII concludes this work.

II. Related Work

The emergence of PDPs has enabled a new paradigm in which machine learning functionality can be embedded directly within network devices. In-network Machine Learning (IN-ML) shifts computation from external servers into the forwarding path, reducing latency and enabling real-time responsiveness for tasks such as intrusion detection and congestion control [9]. This paradigm is particularly attractive for security applications, where early, line-rate decision making is critical.

A significant body of work has focused on deploying inference logic directly within programmable switches. P4-NIDS [9] demonstrates how decision-tree classifiers can be encoded into match–action pipelines, enabling inline intrusion detection using header-derived features and register-based state. Similarly, Paolini *et al.* [10] show that even deep neural networks can be mapped onto the PISA architecture by translating computations into staged table lookups and register operations, allowing per-packet inference at line rate. These works highlight the feasibility of embedding increasingly complex inference mechanisms within the data plane, but they remain limited to pre-trained, static models.

Beyond inference, only a limited number of works explore learning directly within the data plane. QCMP [11] is among the first to demonstrate RL training on programmable switches. It applies Q-learning to adaptive load balancing, updating path-selection policies at line rate based on queue-length feedback. By avoiding explicit modeling of network dynamics, QCMP shows that lightweight, incremental learning can operate under strict data-plane constraints. However, its focus is on traffic engineering rather than security.

In parallel, RL has been widely studied for DDoS mitigation, primarily in SDN-based architectures. Khozam *et al.* [12] propose a deep RL framework where an agent dynamically adjusts mitigation strategies based on flow-level statistics and QoS-aware rewards, with actions enforced via the SDN controller. Simpson *et al.* [13] argue that RL is well-suited for DDoS defense due to its ability to adapt to non-stationary traffic and limited labeled data, addressing key limitations of traditional anomaly detection systems. Similarly, Ganesan *et al.* [14] combine RL with programmable switches in SDN to mitigate flooding attacks while reducing controller overhead. Zhang *et al.* [15] further explore RL-based defenses for amplification attacks, where agents learn throttling policies based on server-specific and aggregate traffic load, enabling adaptive filtering of attack traffic.

Despite these advances, existing approaches exhibit key limitations. IN-ML solutions predominantly focus on inference rather than online adaptation, while RL-based mitigation systems typically rely on control-plane execution or offline training. Moreover, amplification-aware mitigation within the PDP remains largely unexplored. In response, this work proposes a fully in-network RL approach that operates entirely within the data plane, performs online policy updates at line rate, and explicitly targets amplification-based DDoS attacks using telemetry-derived features and QoS-aware feedback.

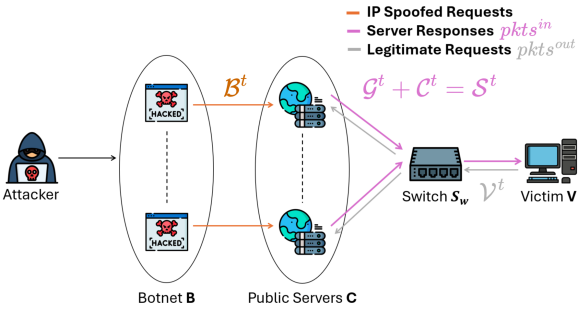

Fig. 1. System model of the amplification-based DDoS attack scenario.

III. System Model

In this section we model the network scenario subject to an amplification-based DDoS attack. Time is modeled as a discrete variable $t \in \mathbb{N}$, where each time step corresponds to a fixed observation window of duration T. The system consists of the following abstract components: (i) an attacker controlling a set of compromised hosts denoted as B (e.g., bots generating spoofed requests); (ii) a set of public servers exploited as reflectors denoted as C (e.g., DNS or NTP servers); (iii) a network switch forwarding traffic towards the victim S_{w} (a PDP switch); (iv) a victim host targeted by the attack denoted as V (the protected service endpoint), as depicted in Fig. 1. During each time window t, the victim generates a set of request packets that are forwarded by the switch toward one of the public servers. Let

$$\mathcal{V}^t = \{pkt_1^{out}, pkt_2^{out}, \ldots, pkt_{|\mathcal{V}^t|}^{out}\} \tag{1}$$

denote the set of outbound packets observed during time window t, and $|\mathcal{V}^t|$ is the number of packets in the set. As a consequence, the public servers generate response packets, forming the set

$$\mathcal{C}^t = \{pkt_1^{in}, pkt_2^{in}, \ldots, pkt_{|\mathcal{C}^t|}^{in}\}, \tag{2}$$

of inbound packets observed during time window t, and $|\mathcal{C}^t|$ is the number of packets in the set, which are forwarded by the switch toward the victim. During each time window, the switch observes a set of flows, denoted as

$$\mathcal{F}_{\mathcal{C}} = \{f_1, f_2, \ldots, f_k, \ldots, f_{|\mathcal{C}|}\}, \tag{3}$$

where each flow is identified by the 5-tuple

$$f_k = \langle IP_{src}, IP_{dst}, Protocol, Port_{src}, Port_{dst} \rangle. \tag{4}$$

While benign traffic may occasionally exhibit moderate request–response imbalance (e.g., due to DNSSEC or bulk replies), amplification attacks are characterized by persistent and significantly larger deviations that are sustained across multiple time windows. In contrast, under normal

request–response operation, the Bandwidth Amplification Factor, BAF (AF_{b}), i.e., the ratio between the size of the response sent by the amplifier and the size of the request generated by the attacker, and the Packet Amplification Factor, PAF (AF_{p}) that quantifies the degree of traffic amplification, as defined in [16], between any request pkt^{out} and its corresponding response pkt^{in} associated with the same flow f_{k} satisfy

$$AF_{\text{b}} = \frac{\sum_{pkt^{in} \in \mathcal{C}^t} \text{bytes}(pkt^{in})}{\sum_{pkt^{out} \in \mathcal{V}^t} \text{bytes}(pkt^{out})} \approx 1, \qquad AF_{\text{p}} = \frac{|\mathcal{C}^t|}{|\mathcal{V}^t|} \approx 1. \tag{5}$$

If an attack occurs during a time window, the botnet generates a set of spoofed request packets directed toward the public servers, with the destination being victim's IP address. Let

$$\mathcal{B}^t = \{pkt_1^{out}, pkt_2^{out}, \ldots, pkt_{|\mathcal{B}^t|}^{out}\}, \tag{6}$$

denote the set of spoofed requests observed during time window t, and $|\mathcal{B}^t|$ is the number of packets in the set. For each spoofed request $pkt^{out} \in \mathcal{B}^t$, the public servers generate a set of amplified response packets, denoted by $\mathcal{G}(pkt^{out})$. The overall set of amplified response packets observed during the same time window is therefore given by

$$\mathcal{G}^t = \sum_{pkt^{out} \in \mathcal{B}^t} \mathcal{G}(pkt^{out}) = \{pkt_1^{in}, pkt_2^{in}, \ldots, pkt_{|\mathcal{G}^t|}^{in}\}, \tag{7}$$

that is the set of inbound packets observed during time window t, and $|\mathcal{G}^t|$ is the number of packets in the set, which are forwarded by the switch toward the victim. The resulting inbound traffic may saturate the victim's network and processing resources, leading to severe service degradation or denial of service. In this case, BAF and PAF satisfy

$$AF_{\text{b}} = \frac{\sum_{pkt^{in} \in \mathcal{G}^t} \text{bytes}(pkt^{in})}{\sum_{pkt^{out} \in \mathcal{B}^t} \text{bytes}(pkt^{out})} \gg 1, \qquad AF_{\text{p}} = \frac{|\mathcal{G}^t|}{|\mathcal{B}^t|} \gg 1. \tag{8}$$

As demonstrated in [13], [15], the correspondence ratio between upstream and downstream traffic represents a highly informative feature for detecting malicious traffic. Traffic asymmetry constitutes a key behavioral indicator that helps distinguish legitimate traffic patterns, which typically exhibit balanced request-response exchanges, from illegitimate patterns generated during amplification attacks, where a limited volume of requests triggers a disproportionately large volume of responses. Motivated by these observations, we introduce a generalized *Asymmetry Index (AI)* that jointly accounts for both packet-level and bandwidth-level imbalance to have a protocol-agnostic representation of the degree of asymmetry observed during a time window t. Specifically, the index is defined as a function of AF_{p} and AF_{b},

$$AI_t = f(AF_{\text{b}}, AF_{\text{p}}), \tag{9}$$

where the function $f(\cdot)$ is designed to map traffic asymmetries to a normalized range $AI_t \in [0, 1]$. Both factors are required, as amplification may arise either from a single request triggering a disproportionately large response, or from a single request eliciting multiple response packets. Values of AI_t close to 0 indicate a strong asymmetry skewed toward the victim, where the outbound traffic from the victim to the servers (denoted as *out*) significantly exceeds the inbound traffic in the opposite direction (denoted as *in*). Conversely, values of AI_t close to 1 indicate asymmetry skewed toward the servers, where inbound traffic dominates outbound traffic. Values around 0.5 correspond to symmetric traffic, typically associated with benign request-response communication.

IV. PDP-enabled Q-Learning Mitigation of Amplification-Based DDoS Attacks

The objective of the mitigation system is to reduce traffic asymmetry induced by amplification-based DDoS attacks while preserving legitimate request–response communication to avoid service degradation. However, static, rule-based mechanisms in PDPs are insufficient, as effective mitigation requires reasoning over multiple time windows and capturing how actions influence future traffic dynamics and QoS, which is incompatible with the limited computation, memory, and locality of data-plane devices. This is particularly challenging for amplification attacks, where malicious traffic closely resembles benign flows and packet-level filtering is inadequate, motivating the use of aggregate features such as correspondence ratios. We adopt a value-based RL approach based on Q-learning, as it supports online learning with lightweight computations compatible with programmable data-plane constraints. Q-learning operates on a state–action–reward formulation; in particular, the programmable switch is modeled as a learning agent that observes traffic statistics on a per-flow basis, selects mitigation actions for each flow, and receives feedback as reward.

a) Asymmetry Index (AI): To quantify per-flow traffic imbalance, traffic volumes and packet counts are discretized into B bins, such that

$$\text{bin}(\cdot) \in \{0, \ldots, B-1\}. \tag{10}$$

The BAF and PAF for flow f_k are computed as

$$AF_{\text{b}}^{(k)} = h\big[\text{bin}(\text{bytes}(\mathcal{S}_k^t)) - \text{bin}(\text{bytes}(\mathcal{V}_k^t))\big], \tag{11}$$

$$AF_{\text{p}}^{(k)} = h\big[\text{bin}(|\mathcal{S}_k^t|) - \text{bin}(|\mathcal{V}_k^t|)\big], \tag{12}$$

where

$$h(x) = \frac{x}{B-1}, \tag{13}$$

and the function bytes$(\cdot)$ returns the corresponding traffic volume measured in bytes and $\mathcal{S}_k^t$ and $\mathcal{V}_k^t$ denote the sets of inbound and outbound packets associated with flow f_k during time window t, respectively. In particular, $\mathcal{S}_k^t$ is defined as

$$\mathcal{S}_k^t = \mathcal{C}_k^t \cup \mathcal{G}_k^t, \tag{14}$$

where $\mathcal{C}_k^t$ represents the legitimate response packets generated by the servers in reply to victim requests, and $\mathcal{G}_k^t$ the amplified response packets triggered by spoofed attacker requests.

This discretization is motivated by the constraints of programmable data-plane architectures, which support only a limited set of arithmetic operations and typically do not allow general-purpose division. As a result, continuous-valued ratios

are replaced by discrete representations that can be efficiently computed using match–action tables, where the output corresponding to a given pair of operands is precomputed and stored. The resulting Asymmetry Index for flow f_k at time window t is defined as

$$AI_t^{(k)} = 50 \cdot \left(1 + \left[w_{\mathrm{b}} AF_{\mathrm{b}}^{(k)} + w_{\mathrm{p}} AF_{\mathrm{p}}^{(k)}\right]\right), \qquad (15)$$

where the weighting coefficients satisfy $w_{\mathrm{b}} + w_{\mathrm{p}} = 1$ and $0 \leq w_{\mathrm{b}}, w_{\mathrm{p}} \leq 1$. Since both $AF_{\mathrm{b}}^{(k)}$ and $AF_{\mathrm{p}}^{(k)}$ take values in the interval $[-1, 1]$, the weighted sum in brackets is also bounded in $[-1, 1]$. The index is scaled in the range $[0, 100]$ to ensure compatibility with programmable data-plane arithmetic. Values of $AI_t^{(k)} \approx 50$ indicate symmetric traffic conditions, values close to 0 indicate victim-side traffic imbalance, and values close to 100 indicate server-side traffic imbalance.

b) State: At the end of each time window t, the agent observes an aggregated traffic asymmetry indicator and a service degradation signal resulting from the previously applied mitigation decisions. The state at time t is defined as

$$s_t = \left(\mathbf{AI}_{t-1}, \mathbf{a}_{t-1}, \mathrm{ServDegr}_t\right), \qquad (16)$$

where $\mathbf{AI}_{t-1} = (AI_{t-1}, AI_{t-2}, \ldots, AI_{t-H})$ is a finite history vector of past asymmetry index values, computed over the previous H time windows; $\mathbf{a}_{t-1} = (a_{t-1}, a_{t-2}, \ldots, a_{t-H})$ is the corresponding history of mitigation actions applied during those windows; $\mathrm{ServDegr}_t$ encodes whether mitigation decisions have negatively impacted legitimate traffic, due to excessive packet drops over consecutive windows. The state representation is extended to include both the asymmetry index and the action history, i.e., (AI_{t-1}, a_{t-1}), as the asymmetry value alone lacks absolute semantic meaning and depends on the policy under which it was generated. Associating each asymmetry observation with its corresponding action allows the agent to distinguish between benign traffic, attack-induced imbalance, and policy-induced effects, while also enabling correct reward attribution by avoiding penalization due to exogenous traffic dynamics. This contextual encoding improves learning stability and ensures causal consistency in the decision process. The history length H is treated as a design parameter, and by augmenting the state with this bounded memory, the Markov property is preserved, allowing the agent to capture short-term temporal dependencies under partial observability, as also adopted in [13].

c) Action: Mitigation decisions are applied independently on a per-flow basis. For each flow f_k, the action space is defined as

$$\mathcal{A} = \{\texttt{DROP}, \texttt{ALLOW}\}, \qquad (17)$$

where the selected action $a_t \in \mathcal{A}$ determines whether packets belonging to flow f_k are forwarded or filtered during the current time window. In particular, when the action `DROP` is selected, packets associated with flow f_k are dropped in both directions, that is for traffic flowing from the victim to the servers and from the servers to the victim.

d) Reward: The reward function is designed to drive the agent toward mitigation strategies that suppress attack-induced traffic asymmetry while avoiding unnecessary service disruption. The learning objective is therefore to promptly react to malicious imbalance and to prevent excessive degradation of legitimate traffic. At each t, the switch receives a scalar reward $r_t \in \{-1, 0, +1\}$, computed as the combination of an action-conditioned asymmetry component r_t^{AI} and a service degradation constraint explained in detail below.

The current asymmetry index AI_{t-1} is not evaluated in isolation. Its interpretation depends on the action applied in that time window a_{t-1}, since the same asymmetry value may correspond to balanced traffic, insufficient mitigation, or excessive dropping depending on whether traffic was previously allowed or dropped. When evaluating the action selected at time window $t-1$, the reward mechanism therefore considers the recent decisions and traffic history that led to the current traffic condition. In principle, the reward function could incorporate a longer temporal dependency and inspect multiple past decisions in order to more accurately attribute causality. However, for the sake of methodological clarity and to demonstrate feasibility in this implementation, the historical depth is limited to two decision windows ($H = 2$). This limited look-back is sufficient to capture corrective or persistent behaviors while maintaining a simple and interpretable formulation.

Accordingly, the asymmetry-based reward component r_t^{AI} is defined as a function of the pair (AI_{t-1}, a_{t-1}), optionally informed by the immediately preceding asymmetry observation AI_{t-2}. Let β_L, β_M^-, β_M^+, and β_H denote symbolic thresholds defining low, moderate, and high asymmetry regions, with

$$\beta_L < \beta_M^- < \beta_M^+ < \beta_H. \qquad (18)$$

The asymmetry-based reward is defined as:

$$r_t^{AI} = \begin{cases} 0, & \text{if } AI_{t-1} \in (\beta_M^-, \beta_M^+) \\ & \text{and } a_{t-1} == \texttt{ALLOW} \text{ and } AI_{t-2} \geq \beta_H, \\ +1, & \text{if } AI_{t-1} \in (\beta_M^-, \beta_M^+) \\ & \text{and } a_{t-1} == \texttt{ALLOW}, \\ 0, & \text{if } AI_{t-1} \leq \beta_L \\ & \text{and } a_{t-1} == \texttt{DROP} \text{ and } AI_{t-2} \geq \beta_H, \\ -1, & \text{if } AI_{t-1} \leq \beta_L \\ & \text{and } a_{t-1} == \texttt{DROP}, \\ 0, & \text{if } AI_{t-1} \geq \beta_H \\ & \text{and } a_{t-1} == \texttt{ALLOW} \text{ and } AI_{t-2} \leq \beta_M^+, \\ -1, & \text{if } AI_{t-1} \geq \beta_H \\ & \text{and } a_{t-1} == \texttt{ALLOW}, \\ 0, & \text{otherwise.} \end{cases} \qquad (19)$$

The reward function evaluates actions contextually based on recent asymmetry and policy history. The agent is rewarded when asymmetry and the previous action are consistent with *balanced* traffic, i.e., moderate asymmetry following an `ALLOW` decision, first pair of cases in Eq. (19). However, if this follows a previous high-asymmetry phase, the reward

is set to zero to avoid rewarding ALLOW decisions during ongoing attacks. The agent is penalized for *overreaction*, i.e., when a DROP decision leads to strong asymmetry toward the victim side, indicating unnecessary suppression of legitimate traffic, second pair of cases in Eq. (19). *Underreaction* occurs when an ALLOW decision leads to strong asymmetry toward the server side, indicating insufficient mitigation under likely attack conditions, third pair of cases in Eq. (19). Penalties are relaxed when prior conditions justified the action.

Finally, to balance security and service availability, the final reward incorporates the service degradation counter ServDegr_t. Service degradation is modeled as a cumulative counter that increases upon drop decisions and decreases when traffic is forwarded. If the cumulative degradation exceeds a predefined threshold X, the reward is adjusted as follows:

$$r_t = \begin{cases} -1, & \text{if ServDegr}_t > X \text{ and } r_t^{AI} \neq +1, \\ 0, & \text{if ServDegr}_t > X \text{ and } r_t^{AI} = +1, \\ r_t^{AI}, & \text{otherwise.} \end{cases} \quad (20)$$

An important assumption underlying this design is that amplification-based attacks do not persist longer than N consecutive time windows. The attacker model is simplified to enable controlled evaluation; however, more adaptive or heterogeneous attackers could impact the learned policy. Therefore, the service degradation threshold X is chosen such that $X \leq N$, thus this design ensures that the service degradation mechanism regulates unnecessary aggressiveness but does not interfere with legitimate defensive behavior during attack phases.

Q-learning learns an action-value function $Q(s, a)$, which estimates the expected cumulative discounted reward obtained by selecting action a in state s and following the learned policy thereafter. For each flow f_k, the Q-function is updated independently at time window t, after observing state $s_t^{(k)}$, selecting action $a_t^{(k)}$, and receiving reward $r_t^{(k)}$, the Q-value is updated according to

$$Q_{t-1}\left(s_{t-1}^{(k)}, a_{t-1}^{(k)}\right) = Q_{t-1}\left(s_{t-1}^{(k)}, a_{t-1}^{(k)}\right) + \alpha\left[r_t^{(k)} + \gamma \max_{a \in \mathcal{A}} Q_t\left(s_t^{(k)}, a\right) - Q_{t-1}\left(s_{t-1}^{(k)}, a_{t-1}^{(k)}\right)\right], \quad (21)$$

where $\alpha \in (0, 1]$ is the learning rate and $\gamma \in (0, 1)$ is the discount factor. The flow of the Q-learning process is described in detail in Algorithm 1.

V. Experimental Setup

Based on the network topology introduced in Fig. 1 in the previous section, we consider a simplified yet representative setup involving one NTP server, i.e., $|C| = 1$. The following assumptions are made: a single logical attacker entity is considered, modeling spoofed amplification traffic without explicitly simulating a distributed botnet and traffic decisions are taken at fixed intervals of duration $T = 10s$. Service degradation is modeled as a cumulative counter that increases by one unit for each DROP decision and decreases by one unit for each ALLOW decision. Furthermore, we assume that amplification-based attacks do not persist longer than $N = 8$ consecutive time windows. Finally, the asymmetry thresholds are set as $\beta_L = 0.35$, $\beta_M^- = 0.45$, $\beta_M^+ = 0.55$, and $\beta_H = 0.65$.

Algorithm 1 Q-Learning for Mitigation of Amplification-Based DDoS Attacks

Initialize $Q(s, a)$ arbitrarily
Set learning rate α, discount factor γ, exploration rate ϵ
Set window duration T and episode length to E windows
for each episode **do**
  Initialize environment and observe state s_0
  for $t = 0$ to $E - 1$ **do**
    Select action a_t using ϵ-greedy policy, per flow f_k
    Apply a_t for the entire time window t
    Measure traffic statistics over T, Eqs. (11), (12)
    Compute next state s_{t+1}, Eq. (16)
    Compute reward r_t, Eq. (20)
    Update $Q(s_t, a_t)$, Eq. (21)
    $s_t \leftarrow s_{t+1}$
  end for
end for

The data plane is implemented using BMv2 with the *simple_switch* target, a widely adopted reference platform for P4 experimentation [17], [18]. Based on the *v1model* architecture, it provides a full packet-processing pipeline with support for match-action tables, counters, meters, and registers, enabling accurate and reproducible prototyping of in-switch RL mechanisms. The source code of the prototype is publicly available [19]. The environment consists of a preconfigured virtual machine based on Ubuntu 24.04 Desktop (AMD64) that includes the main P4 software components compiled from source. The VM, equipped with 2vCPUs, 4GB of RAM and 60GB of storage, integrates the P4 compiler (p4c), the BMv2 software switch, the Mininet network emulator, and the P4Runtime control interface.

Experiments emulate amplification-based DDoS traffic using a reproducible, seed-controlled generator that alternates between *cyclic* (gradual escalation–plateau–decrease) and *burst* (sudden spikes) attack patterns, interleaved with silent phases, exposing the agent to heterogeneous temporal dynamics. Traffic is organized in windows with attacker states $\{SILENT, LOW, MEDIUM, HIGH\}$ and victim states $\{LOW, MEDIUM, HIGH\}$ following a probabilistic distribution $(0.3/0.4/0.3)$. Packet rates per window are $\{10, 30, 50\}$ for the victim and $\{0, 10, 30, 75\}$ for the attacker. Attack phases include silent (3–5 windows), cyclic escalation with a 3–4 window $HIGH$ plateau, and burst events (4–6 windows). Amplification factors (BAF/PAF) and packet characteristics (48 bytes fixed size) are grounded in empirical studies to ensure realism [16].

The Q-learning agent ($\alpha = 0.1, \gamma = 0.9$) is trained for 2500 windows ($\sim$3000 reachable states) and evaluated over 500 windows ($\sim 8.3h$), with a service degradation threshold $X = 8$ to penalize sustained attacks. Experiments span NTP scenarios by varying asymmetry weights w_b, w_p and exploration rate

ϵ, assessing sensitivity to feature emphasis and the role of exploration in avoiding bias toward default ALLOW actions and improving discrimination between benign and attack traffic.

VI. Results and Discussion

We evaluate the proposed mechanism under the three following scenarios with different $\{w_b, w_p, \epsilon\}$ configurations: 1) $\{0.5, 0.5, 0.1\}$, 2) $\{0.4, 0.6, 0.1\}$ and 3) $\{0.5, 0.5, 0.3\}$. Fig. 2 shows the evolution of the average reward across NTP scenarios, exhibiting a stable learning trend in all configurations. The reward starts slightly negative and gradually increases, stabilizing at small positive values. Scenarios with $\epsilon = 0.1$ achieve smoother convergence, while $\epsilon = 0.3$ introduces more oscillations due to increased exploration. Scenario 1 converges to a slightly lower reward than 2, as the service degradation penalty is triggered more frequently, leading to more penalized states despite correct mitigation behavior.

Fig. 3 illustrates the impact of the service degradation component on decision-making in scenario 1. Around time window 418, the agent selects ALLOW during a *HIGH*-intensity attack to prevent exceeding the degradation threshold, causing a temporary increase in the Asymmetry Index and a subsequent switch back to DROP action. A similar pattern occurs near window 474, where ALLOW is chosen under *MEDIUM*-intensity traffic to maintain acceptable service levels, allowing subsequent *LOW*-intensity traffic without additional mitigation. These cases highlight the agent's ability to balance mitigation and QoS constraints under dynamic conditions.

Fig. 4 summarizes detection performance across attacker profiles. For *SILENT* windows, the agent correctly selects ALLOW in approximately 89%–91% of cases, indicating low false positives. Detection improves with attack intensity, exceeding 70% for *MEDIUM* and *HIGH* profiles and reaching above 80% in the best cases. Scenario 1 achieves the highest *MEDIUM* detection rate (82.95%) while maintaining strong *HIGH* detection performance (79.75%), confirming the effectiveness of the learned policies. The lower detection rates for *LOW*-intensity attacks (approximately 30%–40% across scenarios) can be attributed to their limited traffic asymmetry, which often resembles benign fluctuations and therefore does not provide a strong signal for the agent to trigger mitigation. This conservative behavior reduces false positives and is desirable, as *LOW*-intensity activity may reflect probing phases; aggressive reactions risk disrupting legitimate traffic, whereas caution preserves service availability. The reward formulation penalizes unnecessary drops to preserve QoS, thereby biasing the policy toward ALLOW decisions in borderline conditions.

To provide a baseline, we compare our proposed approach with a static threshold-based filtering policy inspired by traditional firewall mechanisms, where mitigation is triggered when a monitored metric exceeds a predefined threshold [20]. In our case, the Asymmetry Index is used, and traffic is dropped in the subsequent window if $AI \geq \beta_H = 0.65$, otherwise allowed. This rule relies solely on instantaneous observations, without historical context or adaptation. Both approaches are evaluated under identical conditions using the same traffic

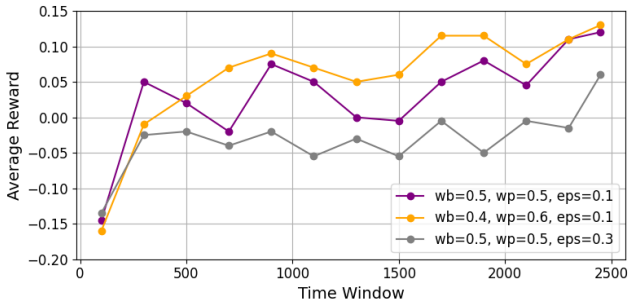

Fig. 2. Training reward under different parameter configurations for the NTP attack scenario, shown as the average over intervals of 200 windows.

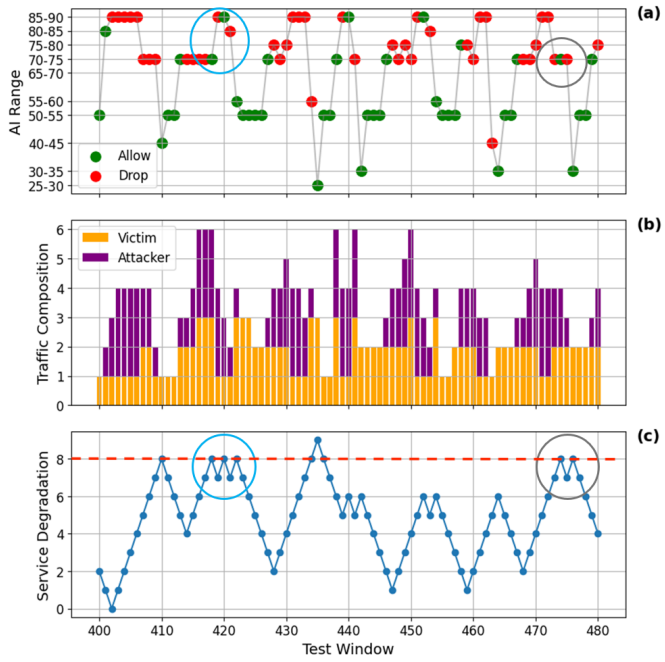

Fig. 3. Example agent decisions for scenario 1: (a) AI and selected action; (b) traffic composition per window; (c) resulting service degradation.

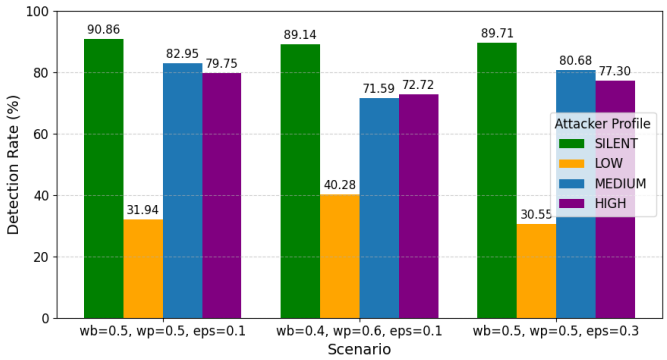

Fig. 4. Detection rate across scenarios for each attacker profile.

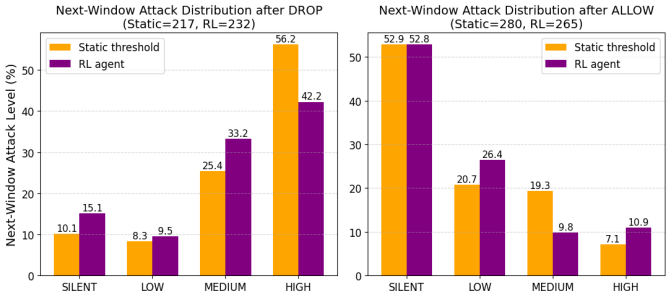

Fig. 5. Distribution of next-window attacker activity conditioned on current mitigation decisions, comparing the RL agent and a static threshold.

traces (scenario 1) and a 500-window test phase, allowing a fair comparison of their decision behavior and predictive capability, as illustrated in Fig. 5.

The results show that the static policy primarily reacts to high-intensity attacks, with `DROP` decisions followed by *HIGH* traffic in $\sim$56% of cases, whereas the RL agent distributes decisions more evenly, increasing intervention during *MEDIUM* phases ($\sim$33%) and enabling earlier mitigation. For `ALLOW` decisions, both methods preserve benign traffic (*SILENT* $\sim$53%), but the RL agent tolerates more *LOW* activity (26% vs. 21%) to avoid unnecessary QoS degradation, while reducing missed *MEDIUM* attacks by leveraging temporal context. In contrast, the threshold policy suffers from delayed reactions and traffic masking effects, allowing attacks to escalate before triggering mitigation and occasionally continuing mitigation after the attack subsides. Overall, the RL-based approach demonstrates more adaptive and context-aware behavior, particularly in scenarios where instantaneous asymmetry is insufficient to characterize evolving attacks.

VII. Conclusion and Future Work

This paper presents an adaptive approach for mitigating amplification-based DDoS attacks directly within PDPs using an in-switch Q-learning agent. The proposed system relies on line-rate telemetry and lightweight computations, and allows for online policy updates without reliance on control-plane support or offline training datasets. The integration of traffic asymmetry metrics with service degradation feedback allows the agent to make context-aware mitigation decisions under dynamic traffic conditions. Experimental results show that the agent adapts to heterogeneous attack patterns, maintains high detection performance across different traffic intensities, and improves decision behavior compared to static filtering by reducing unnecessary drops while preserving service quality.

Future work will focus on extending the action space beyond binary decisions, improving feature expressiveness within data-plane constraints, and validating the approach on hardware targets and more realistic traffic scenarios.

References

[1] Z. R. Alashhab, M. Anbar, M. M. Singh, I. H. Hasbullah, P. Jain, and T. A. Al-Amiedy, "Distributed denial of service attacks against cloud computing environment: Survey, issues, challenges and coherent taxonomy," *Applied Sciences*, vol. 12, no. 23, 2022.

[2] A. Singh and B. B. Gupta, "Distributed denial-of-service (ddos) attacks and defense mechanisms in various web-enabled computing platforms: Issues, challenges, and future research directions," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 18, no. 1, pp. 1–43, 2022.

[3] F. Musumeci, A. C. Fidanci, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-enabled ddos attacks detection in p4 programmable networks," *Journal of Network and Systems Management*, vol. 30, no. 1, p. 21, 2022.

[4] "Targeted by 20.5 million ddos attacks, up 358% year-over-year: Cloudflare's 2025 q1 ddos threat report." [Online]. Available: https://blog.cloudflare.com/ddos-threat-report-for-2025-q1/

[5] "Hyper-volumetric ddos attacks skyrocket: Cloudflare's 2025 q2 ddos threat report." [Online]. Available: https://blog.cloudflare.com/ddos-threat-report-for-2025-q2/

[6] "Defending the internet: how cloudflare blocked a monumental 7.3 tbps ddos attack." [Online]. Available: https://blog.cloudflare.com/defending-the-internet-how-cloudflare-blocked-a-monumental-7-3-tbps-ddos/

[7] ENISA, "Enisa threat landscape 2024," pp. 1–10, 2024. [Online]. Available: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024

[8] "Global operation targets noname057(16) pro-russian cybercrime network." [Online]. Available: https://www.europol.europa.eu/media-press/newsroom/news/global-operation-targets-noname05716-pro-russian-cybercrime-network

[9] Y. Chen, S. Layeghy, L. D. Manocchio, and M. Portmann, "P4-NIDS: high-performance network monitoring and intrusion detection in P4," *CoRR*, vol. abs/2411.17987, 2024.

[10] E. Paolini, L. D. Marinis, D. Scano, and F. Paolucci, "In-line any-depth deep neural networks using P4 switches," *IEEE Open J. Commun. Soc.*, vol. 5, pp. 3556–3567, 2024.

[11] C. Zheng, B. Rienecker, and N. Zilberman, "QCMP: load balancing via in-network reinforcement learning," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Future of Internet Routing & Addressing, FIRA@SIGCOMM 2023, New York, NY, USA, 10 September 2023*. ACM, 2023, pp. 35–40.

[12] S. Khozam, G. Blanc, S. Tixeuil, and E. Totel, "Ddos mitigation while preserving qos: A deep reinforcement learning-based approach," in *10th IEEE International Conference on Network Softwarization, NetSoft 2024, Saint Louis, MO, USA, June 24-28, 2024*. IEEE, 2024, pp. 369–374.

[13] K. A. Simpson, S. Rogers, and D. P. Pezaros, "Per-host ddos mitigation by direct-control reinforcement learning," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 1, pp. 103–117, 2020.

[14] A. Ganesan and K. Saraç, "In-network reinforcement learning for attack mitigation using programmable data plane in SDN," in *33rd International Conference on Computer Communications and Networks, ICCCN 2024, Kailua-Kona, HI, USA, July 29-31, 2024*. IEEE, 2024, pp. 1–9.

[15] Y. Zhang and Y. Cheng, "An amplification ddos attack defence mechanism using reinforcement learning," in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI 2019, Leicester, United Kingdom, August 19-23, 2019*. IEEE, 2019, pp. 634–639.

[16] C. Rossow, "Amplification hell: Revisiting network protocols for ddos abuse," in *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014.

[17] C. Zheng et al., "In-network machine learning using programmable network devices: A survey," *IEEE Commun. Surv. Tutorials*, vol. 26, no. 2, pp. 1171–1200, 2024.

[18] P. L. Consortium, "Behavioral model v2 documentation (bmv2)," 2024, accessed: January 2026. [Online]. Available: https://github.com/p4lang/behavioral-model

[19] S. Sampognaro, "In-switch rl ddos defense: Reinforcement learning for amplification-based ddos mitigation in programmable data planes," 2026, gitHub repository. [Online]. Available: https://github.com/SimoneSampognaro/in-switch-rl-ddos-defense

[20] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to ddos attack detection and response," in *Proceedings DARPA Information Survivability Conference and Exposition*, vol. 1, 2003, pp. 303–314 vol.1.