BUNGEE: An Adaptive Pushback Mechanism for DDoS Detection and Mitigation in P4 Data Planes

Libardo Andrey Quintero González, Lucas Castanheira, Jonatas Adilson Marques,
Alberto Schaeffer-Filho, and Luciano Paschoal Gaspary
Institute of Informatics, Federal University of Rio Grande do Sul
Porto Alegre, Brazil
{laqgonzalez, lbcastanheira, jonatas.marques, alberto, paschoal}@inf.ufrgs.br

***Abstract*—A DDoS attack aims for resource exhaustion and directly impacts the availability of servers in a network infrastructure. Although significant efforts have been made to detect and mitigate DDoS attacks in viable time, this type of attack remains one of the leading security concerns in networking. By leveraging data plane programmability, it becomes possible to implement novel security solutions that do not rely on coordination with external servers, keeping the detection and mitigation local to the data plane, potentially reducing delays and not being subject to usual communication bottlenecks. In this paper we present BUNGEE[1], an in-network, collaborative pushback mechanism for DDoS attack mitigation that runs entirely in the data plane. This mechanism is able to, locally at a given switch, identify suspect IP addresses (through the use of continuous IP entropy analysis) and propagate them to other switches. The different switches that are made aware of the suspects enforce a pushback strategy for repelling potential attacks. We implemented our solution using the P4 language. The results reveal that the identification process has high accuracy and that the pushback strategy is effective in minimizing strain to network resources.**

***Index Terms*—Programmable Data Plane, P4, DDoS, Mitigation, Attack Pushback.**

I. INTRODUCTION

Distributed Denial of Service (DDoS) attacks have increased in both volume and duration in the last years, and remain one of the leading security concerns in networking [1]. As recent reports show (*e.g.*, [2]–[4]), DDoS attacks can reach rates in the order of Tbps with the purpose of overloading servers or network links.

Motivation and Problem Definition. Significant efforts have been made recently to understand DDoS attacks [5], and several strategies for detecting [6], [7] and mitigating [8] these attacks have been proposed. However, finding the optimal place to deploy DDoS detection and mitigation mechanisms remains an intricate task. In general, accurate detection occurs near the victim, specifically at its immediately upstream device, since attack flows can be more easily outlined. The further away from the victim, the more dispersed the attack flows are, and obtaining a correct detection becomes more difficult. Analyzing the problem from a different perspective, when coarse-grained mitigation (*e.g.,* filtering) is deployed closer to the victim, we have a higher toll on legitimate flows and allow illegitimate ones to travel unabated for longer, further compromising the infrastructure. On the other hand, the placement of mitigation mechanisms near the attack source results in a (potentially) reduced number of impacted legitimate flows.

The P4 programming language [9] was recently introduced to enable the programming of the data plane. It allows the offloading of advanced and configurable packet processing functionalities onto network devices [10]. This makes it possible to implement new security solutions directly in the data plane, with no need to communicate with the control plane for decision-making. Recent investigations [11]–[13] use data plane programmability to collect and analyze network traffic statistics, enabling accurate and prompt detection of when an attack is ongoing. However, to the best of our knowledge, no efforts so far capitalized on P4 programmability to devise innovative and effective pushback-based mitigation solutions, whose deployment can be adaptive, without human intervention, to contain fluctuating DDoS attack campaigns.

Our Work. To bridge this gap, in this paper we propose BUNGEE, a novel fully in-network dynamic pushback mechanism to detect and mitigate DDoS attacks. The mechanism is entirely coordinated by the forwarding devices[2], thereby removing the control plane from the critical path and speeding up reactiveness for a prompt defense. In a nutshell, BUNGEE detects a DDoS attack through the entropy analysis of the addresses of incoming packets, since this type of attack is known to cause disturbances in the distribution of source and destination IP addresses. In the proposed pushback mechanism, a forwarding device located next to the victim is expected to alert upstream forwarding devices immediately after entropy disturbances are detected. These devices, in turn, will start to apply a filtering strategy to packets from suspect network flows. Further, these devices will also run the detection mechanism themselves and, if necessary, alert their upstream counterparts, thus effectively "pushing back" the effects of the attack. This process will be repeated in an attempt to confine the attack traffic to as close to the

[1]BUNGEE is named after *Bungee Jumping*, a sports activity that abstractly resembles our mechanism, in which a person leaps down from a high height, connected to an elastic cord. The jumper oscillates up and down until energy vanishes and stability is reached.

[2]We use the terms (*forwarding*) *device(s)* and *switch(es)* interchangeably throughout the manuscript.

978-3-903176-32-4

source as possible. Our goal is to ensure the availability of the victim and prevent unnecessary consumption of processing and network resources. As the attack ceases, the mechanism starts to gradually deactivate the filtering procedures in place.

Differently from existing data plane approaches for DDoS mitigation that require frequent communication with another network component (*e.g.,* external controller), the mechanism proposed in this paper runs entirely in the data plane, at line rate. This avoids dependence on permanent communication with the controller, which could cause congestion in the communication channel when an attack is in progress [14]. To demonstrate the concept and technical feasibility of the proposed approach, we carried out a comprehensive set of experiments. Results show that our mitigation mechanism is effective in several proposed attack scenarios, contributing to attenuating the impact on network resources with an overall small memory footprint.

Main Research Contributions. In summary, we make the following three main novel research contributions to the field.

- We devise a *network-wide communication strategy* for switch coordination using P4 data plane primitives.
- We design a *control-plane-free pushback mechanism*, which builds upon our communication strategy to implement a "quick-mounting defense barrier" against DDoS.
- We integrate an efficient *IP entropy anomaly detection approach* to the pushback mechanism to "make the barrier a moving one", avoiding wasting network resources.

Paper Organization. In Section II, we present the background for this work. In Section III, we discuss the state-of-the-art in pushback strategies and DDoS attack mitigation using programmable data planes. In Section IV, we introduce the proposed mechanism. In Section V, we detail our experimental evaluation. Finally, in Section VI, we present the concluding remarks and perspectives for future work.

II. Background

In this section, we briefly revisit the fundamental concepts about data plane programmability and the general background on DDoS attack mitigation strategies.

A. Programmable Data Planes

P4 [9] has emerged as a high-level language for programming the data plane, making networks more flexible and adaptive. It enables the implementation of new functions directly in the programmable data plane (PDP), including security functions for threat mitigation [10]. P4 was designed to describe packet processing capabilities to be executed by a programmable forwarding device. It incorporates a parser, which determines how to handle incoming packets. This parser supports the definition of headers (including those of new protocols) and how they must be processed. Extracted fields are passed to *match+action* tables that describe what actions may be applied to packets when a header field matches the tables. These actions can modify packet headers and determine if incoming packets must be forwarded, dropped, or replicated.

B. DDoS Packet Mitigation

There are four main mitigation techniques that can be used to protect a victim from a DDoS attack [15]. The first is *dropping* all packets coming from sources identified as malicious. The second is packet *filtering*, where just a fraction of the traffic identified as malicious is discarded. The advantage of this approach is that incorrectly identified traffic, *i.e.,* legitimate packets regarded as malicious, will not be completely blocked. The third technique is traffic *redirection*, and consists of forwarding suspicious traffic to a new IP address for further inspection. While it is a compelling approach, it imposes non-negligible overheads regarding forwarding device processing and network bandwidth. Finally, the fourth approach is *quarantine* or *traffic isolation*, where malicious traffic is (temporarily) confined to prevent network resources from being overwhelmed by the attack. This is essentially impractical because it demands large amounts of storage resources.

Given the trade-offs among the techniques above, and as we detail later starting in Section IV, our proposed solution employs packet filtering for attack mitigation. This method can be adequately parameterized to be more conservative or relaxed, depending on the services (and underlying infrastructure) to be guarded. This flexibility, together with a dynamic pushback approach, can yield a solution that is both very accurate and contributes to significant savings in (otherwise wasted) network resources.

III. Related Work

In this section, we discuss the recent literature beginning with general DDoS mitigation approaches that benefit from a programmable data plane design (P4). Right after, we review the main pushback strategies proposed within the domain of software-defined networks (OpenFlow).

A. General Attack Mitigation on PDPs

Innovative and effective attack mitigation solutions taking advantage of programmable data planes (*e.g.*, protocol-independent switch architecture, reconfigurable match-action tables, and P4) are still rare but are starting to thrive. Febro *et al.* [16] propose an approach in which an edge P4 forwarding device is programmed to keep a per-port record of the number of packets received per second. When this number is higher than a configured threshold, subsequent connection requests are automatically dropped by the forwarding device until no attack is detected for an operator-determined interval. The work is interesting in exploring P4 primitives, but its simplified design will penalize all incoming requests, whether legitimate or otherwise, by completely blocking ports (and not only suspicious flows).

Li *et al.* [17] implement a whitelist table that can be used for filtering off spoofed IPs. This table is formed by legitimate IP addresses that have previously established TCP connections with targets of interest. Due to the complexity and required resources (*e.g.*, memory) to keep track of each connection state, the creation of such a whitelist is performed

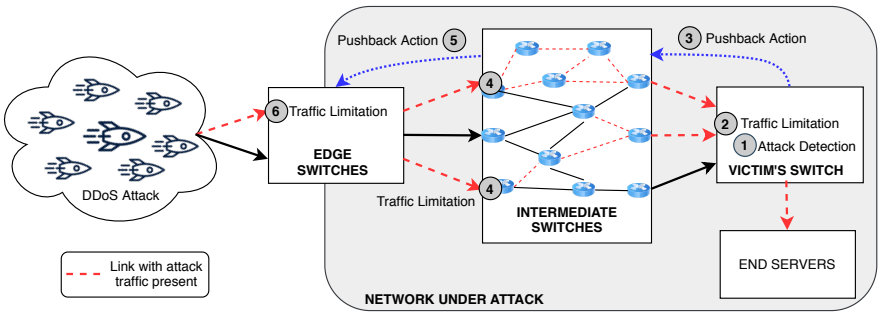

Fig. 1. Overview of the proposed mechanism considering an attack scenario.

on the control plane. To speed up the analysis of incoming packets, a shadow version of the whitelist is maintained on the data plane (and updated periodically by the external SDN controller). Hence, packets from IPs in the data plane whitelist are forwarded to their destination without further processing. Otherwise, they are sent to the control plane for analysis. However, this approach suffers from high communication overheads and memory utilization footprint.

B. Attack Pushback on SDN/OpenFlow Networks

If little has been proposed so far on general attack mitigation mechanisms based on programmable data planes, when it comes to pushback mechanisms, the landscape is even arider, with a few solutions based on SDN/OpenFlow (but not PDPs). As shown in [18], pushback is one of the key strategies used for mitigating threats in the network. To protect the victim, defensive measures are gradually deployed in the opposite direction of an attack. In this context, Zhang *et al.* [19] propose an exciting framework designed to assist the deployment of defenses at strategic locations in the path to a DDoS attack victim. The proposed approach focuses on determining optimal defense locations, given a particular attack scenario. It is, therefore, static and does not support rearrangements.

Moving to SDN/OpenFlow-based solutions, which have the potential to be more dynamic, Bülbül and Fischer [20] introduce a mitigation framework that uses pushback to configure OpenFlow filtering rules on devices belonging to the victim network. However, pushback actions are only carried out when a device capacity is already exceeded, with the same rule set being broadcasted indiscriminately to all upstream switches. Similarly, Pande *et al.* [21] propose a mechanism in which mitigation is performed by configuring SDN/OpenFlow rules to block packets at the edge upon the identification of an attacker from a different autonomous system.

As previously mentioned and evidenced by the discussed efforts, the limited existing body of knowledge typically relies on an external controller to decide on pushback actions, which invariably causes increased latency and additional communication overheads. As far as we are aware of, BUNGEE is the first P4-based mechanism to offload detection, mitigation, and pushback deployment coordination tasks entirely to the data plane.

IV. BUNGEE: A P4-BASED DDOS PUSHBACK MECHANISM

In this section, we introduce BUNGEE, our in-network, collaborative pushback mechanism for DDoS attack mitigation. In Subsection IV-A, we present an overview of its operation. In Subsection IV-B, we describe its foundations, and in Subsection IV-C, we detail its main components, their interactions, and how they are instantiated in P4.

A. Approach Overview

Our mechanism was designed to deal with volumetric DDoS attacks, where, in a coordinated way, multiple sources send a large amount of forged traffic to targets, depleting their processing and network resources. Figure 1 shows a network attack scenario and the general idea behind BUNGEE. It illustrates three different switch "levels" taken into account in our design. The first level refers to the *victim's* switch. It is where the victim of the attack is directly linked to and receives all attack traffic sent to it. Next, *intermediate* switches belong to the next level. These are forwarding devices located either on the same AS or any AS connecting the victim network to the attack source networks. Finally, *edge* switches are equipment to which the attack sources are immediately connected to.

BUNGEE supports *two operation modes*:

- *Intra-AS, corporate deployment:* subnetworks are spread across a metropolitan or regional area but under the management of a single administrative domain, facilitating deployment. In this scenario, it is possible to pushback a DDoS attack to as far as the domain border device.
- *Inter-AS deployment:* switch coordination can span over multiple cooperating ASs across the Internet. In this scenario, mutual agreements (like those required for peering) are necessary, but the pushback can go to as far as the edge switches of the source network(s) of an attack.

The first step of our proposed mitigation mechanism is – once a DDoS attack is detected (which will be described in the following subsection) on the victim's switch (Label 1 in Figure 1) – determining a set of IP addresses as candidate attack sources and starting to filter them locally (2). At this time, the neighbor intermediate switches are alerted about the ongoing attack to take the necessary actions, *i.e.,* a *pushback* action is executed (3). This communication is carried out

through *alarm* packets containing a list of suspect IP addresses. Right after a neighbor switch receives an alarm, it starts its own detection process. If it detects an ongoing attack, it starts filtering packets from informed suspects (4). Additionally, it notifies its upstream switches (5), which re-iterate through the described steps. If the neighbor switch does not detect the attack, we assume it is not going through this path, so no further action is required for this device. Ideally, the pushback mechanism is expected to end at the edge switches (6). In this case, malicious packets will be discarded as close as possible to their sources, thereby avoiding network resource exhaustion.

Although not shown in the figure, an intermediate/edge switch that starts filtering packets for suspect IP addresses also sends a *notification* message (containing IP addresses that the switch has started filtering) to its downstream switches. If a switch receives notifications from all of its upstream neighbors, it means it has successfully handed off filtering of an IP address to its upstream counterparts, and so it stops filtering packets from that IP, making room for new entries. Finally, as we will detail ahead, the proposed solution is adaptive to fluctuating attack dynamics. For example, when an ongoing attack ceases, our mechanism starts a "contraction" process, where filtering actions are gradually deactivated.

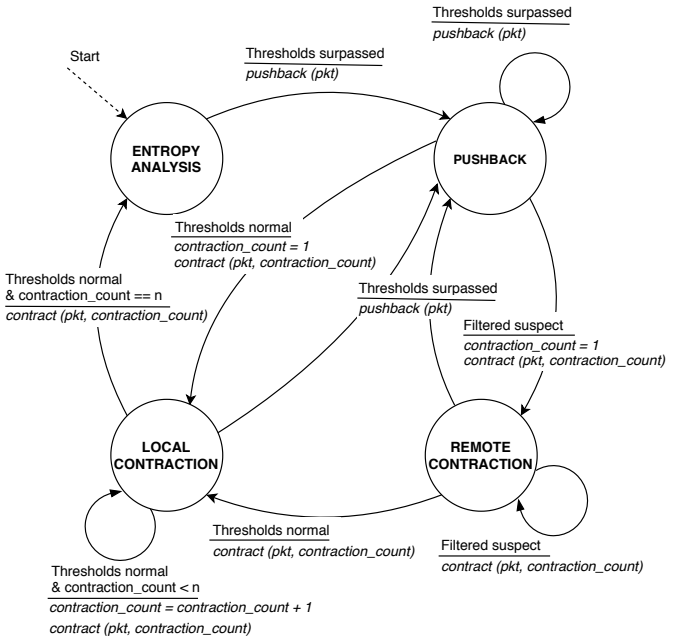

Fig. 2. Operation life cycle of a BUNGEE-enabled switch.

B. Mechanism Formalization

The finite-state machine (FSM) in Figure 2 formalizes the states in which a switch running BUNGEE may be at any point in its operation life-cycle. We focus here on the inter-switch communication processes. In Section IV-C, we will detail the fine-grained operation steps of the proposed mechanism. A forwarding device can detect disturbances in the network traffic via entropy analysis, as discussed below. After detecting an entropy alteration, a pushback action is executed to notify all upstream switches. As a monitoring window ends, the device continues calculating entropy for the following windows. After a no entropy disturbance window, the switch falls into a local contraction state, indicating that the attack is being contained. However, if an entropy disturbance is registered for a window while in this state, the switch performs a pushback action again. To avoid oscillating activation/deactivation of defensive measures, the switch waits for some entropy normality windows to inactivate the defenses (local contraction)[3]. Next, we discuss in detail the main aspects of this FSM.

IP Entropy Analysis. Before the pushback action can kick-off, BUNGEE needs first to detect an ongoing attack. To this end, we employ *IP entropy analysis* in the data plane, building on previous work [12]. The choice for entropy analysis is due to its recognized efficacy for anomaly detection and its simplicity, making it possible to be implemented in resource-constrained programmable devices. In information theory, the entropy of a variable is the level of information inherent in the variable's possible outcomes. It is appropriate to analyze the behavior of network indicators due to its high sensitivity in detecting abnormal network traffic changes. We make use of *source* and *destination IP entropy analysis* for attack detection.

A volumetric DDoS attack causes disturbances in the distribution of source and destination addresses. During a DDoS, the entropy of source addresses tends to increase due to the wide distribution of the attacking (potentially forged) IP addresses. Conversely, the entropy of destination addresses tends to decrease due to the high recurrence of the victim IP address. For discretization purposes, we divide incoming packets into windows of a fixed number of packets. In this context, Shannon's entropy offers the degree of randomness of a monitoring window *(X)*, and can be calculated, per source (H_{src}*(X)*) or destination (H_{dst}*(X)*) IP, as follows:

$$H(X) = \log_2(m) - \frac{1}{m}\sum_{x=0}^{N} f_x \log_2(f_x) \qquad (1)$$

where *m* corresponds to the number of packets in a window, and $f_0, f_1, \ldots, f_N$ represent the frequencies of each (source or destination) address in the window *X*. In case of a DDoS attack, the destination IP entropy tends to zero, H_{dst}*(X)* $\rightarrow 0$. In contrast, the source IP entropy tends to the maximum value, H_{src}*(X)* $\rightarrow MAX$. The entropy values calculated for each monitoring window are compared with thresholds set according to entropy measurements observed during the "normal" network operation. If these values are exceeded, BUNGEE will start the pushback process, as described below.

Pushback Process. The pushback process deploys defenses progressively at the upstream devices, shifting to as far as

[3]Note that state changes occur at different time frames since the attack is detected. It is identified in the monitoring window W_t. First-hop upstream switches perform entropy analysis in the next time window W_{t+1}. Consequently, edge switches will be warned in window W_{t+m}, where m corresponds to the number of intermediate switches.

Algorithm 1: Pushback

```
Input: Incoming packet completing monitoring
       window (pkt)
createAlarm(pkt);
SuspectIPs = {s_1,s_2,s_3...s_n}, vector of suspect IPs
for s_i in SuspectIPs do
    Alarm.append(s_i);
end
UpstreamSwitchList = {d_1,d_2,d_3...d_m}, vector of
 upstream switches
sendAlarm(UpstreamSwitchList);
```

Algorithm 2: Contract

```
Input: Incoming packet from suspect (pkt)
       Consecutive monitoring windows without
       detection (c)
if c == n then
    deactivate(defenses);
end
if pkt.SourceIP ∈ SuspectIPs then
    ApplyFilteringStrategy(pkt);
    createNotification(pkt);
    Notification.append(pkt.SourceIP);
    DownstreamSwitchList = {d_1,d_2,d_3...d_m}, vector
     of downstream switches
    sendNotification(DownstreamSwitchList);
end
```

possible from the victim network. After an entropy disturbance, the victim's switch will alert its first-hop upstream counterparts, sending them an *alarm packet* containing a list of suspect IP addresses. Upon receiving an alarm, upstream neighbors will begin calculating the entropy for the upcoming windows and filtering packets incoming from the suspect IPs.

Usually, the controller executes pushback by sending a particular packet to the switches. Due to limitations in programmable hardware, spontaneous packet generation is not possible. To address this limitation, we resort to the P4 `clone` primitive [22], which allows making a copy of a packet. Thus, pushback is executed by cloning the last packet completing the monitoring window, and using it for inter-switch communication. The cloned packet is transformed into an *alarm packet* by appending a custom header and the identified suspect IPs. Finally, the alarm packet is sent to upstream devices. Algorithm 1 summarizes the described pushback operation.

Local and Remote Contraction. So far, we have shown how pushback is adopted for the deployment of defenses. However, BUNGEE includes a reverse process to inactivate these defenses, which we refer to as *contraction*. This contraction process allows the optimal use of TCAM (Ternary Content-Addressable Memory) and SRAM (Static Random Access Memory) resources. As shown in the FSM in Figure 2, we consider two contraction types. *Local contraction* refers to deactivating defenses after an attack ceases. For this, after a pushback action, the switch keeps a registry of monitoring windows without entropy alteration (*contraction_count*). When the switch considers that the attack stopped for a certain number of windows (*n*), the defenses are deactivated, and the switch returns to the initial state.

During its operation, as the switch begins to filter off a suspect IP address, it notifies the downstream neighbor that filtering has been handed off and that it can stop filtering the suspect; we denote this notification process as *remote contraction*. As this process involves another device, it occurs in different windows, similarly to what happens with the pushback process. A switch begins to filter a suspect in window W_t and notifies the downstream device. At this point, the downstream device requires receiving the same notification from *all* (differently from the pushback action) its upstreams switches to execute the contraction process, *i.e.,* remove the IP address as a suspect. Thus, this process will occur in *up to* window W_{t+w}, where w corresponds to the number of time windows the switch takes to receive contraction notifications from all its upstream switches (in the best case $w = 1$, in the worst case it is a maximum operator-defined value). Algorithm 2 shows how the contraction process operates in an upstream switch after receiving an alarm packet.

C. Architectural Components

The main components that comprise the proposed mitigation mechanism are illustrated in Figure 3. These are expected to be executed on the victim's switch. Alternatively, intermediate and edge switches implement slightly different functionality compared to the victim's switch. Next, we detail the role and operation of each component, following their invocation order in the pipeline followed by ingress packets. Whenever the victim's switch and the intermediate and the edge switches behave differently, those differences will be made explicit.

Suspect List. This is the first component of the mitigation mechanism. It is responsible for examining the source IP address of all incoming packets and determining if they come from sources suspect of generating attack traffic. To this end, the component implements a data structure[4] that is consulted on a per-packet basis. If the source address of an incoming packet matches any entry on the suspect list, it is selected for filtering. Otherwise, it continues following the usual flow, to the *Suspect Identification* component.

Filtering. Packets that match the suspect list must be subjected to filtering. Different strategies can be implemented, ranging from dropping all suspect packets to just a fraction of them, according to a predetermined policy (*e.g.,* random, individual, route-based, score) [23]. In this work, we adopt a partial packet dropping procedure where we control the ratio of suspect packets that will be allowed to follow their way to the victim. Packets that are not dropped move forward in the pipeline to the *Attack Detection* component.

Suspect Identification. For the proper functioning of our mechanism, a list of suspects must be generated. As we

[4]This data structure is populated by the *Suspect Identification* component.

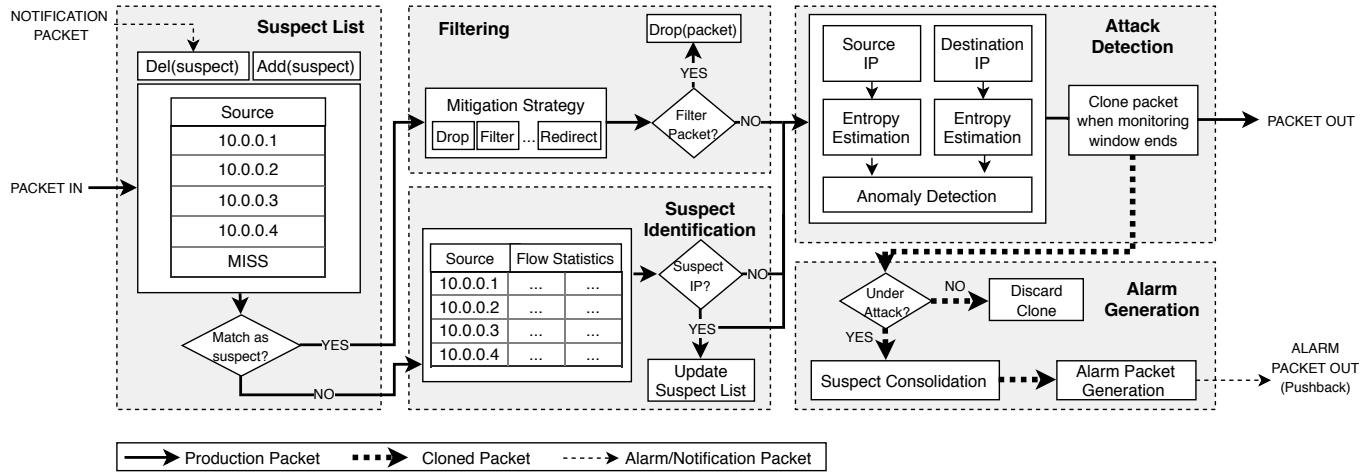

Fig. 3. DDoS attack detection and pushback mechanism implemented in the programmable switch.

mentioned earlier in the formalization section, we discretize traffic analysis using monitoring windows. During a window, the victim switch determines the source IP address of every incoming packet and updates counters associated with addresses. As shown in Figure 4, we resort to a data structure based on sketches (multiple hash tables) [24] to store the monitored statistics. At the end of a window, IP addresses whose frequency is higher than a determined threshold are considered suspects. These sources are sent to the *Suspect List* component.

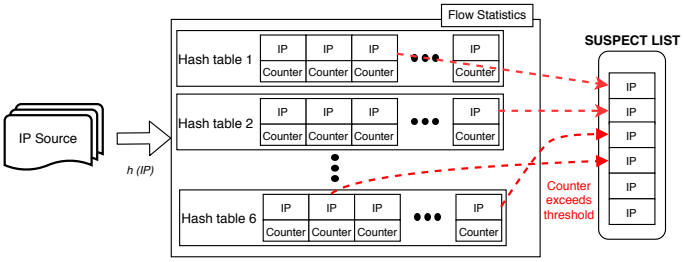

Fig. 4. Sketch-based data structure for suspect identification.

In an intermediate or edge switch, this process is slightly different, as this device only keeps a list of IP addresses to inspect, whose entries are included by a downstream switch (*i.e.,* the one(s) from which it receives alarms). In these devices, source IP addresses matching the inspection list are included in the suspect list.

Attack Detection. The previous components handle the construction/maintenance of a suspect list, as well as the execution of a filtering procedure on packets from suspect IPs. For the sake of triggering the pushback mechanism, we need to detect if an intrusion is taking place. This is the responsibility of the *Attack Detection* component. For intrusion detection, we resort to a robust entropy-based heuristic proposed in our previous work [12]. Additionally, the last packet within a monitoring window is *cloned* and used to trigger the consolidation of values; as such, this *cloned* packet will carry the consolidate values to the *Alarm Generation* component for analysis. A new monitoring window is initiated. All packets that reach this point leave the pipeline and are forwarded to their intended destination.

Alarm Generation. Consolidated entropy values are compared to values estimated for a "healthy" network. If the difference is higher than a determined threshold, the network is considered to be under attack and alarm packets are sent to the upstream switches. Each alarm packet carries the IP addresses considered suspects, consolidated from the *Suspect Identification* component. As previously mentioned, the implementation of this component in P4 is achieved with *cloning* and *recirculation* of packets.

For intermediate and edge switches, in addition to the alarm packets being sent upstream, the intrusion detected is also notified to the downstream switches. Further, an intermediate or edge switch that is upstream in relation to the victim and receives an alarm packet will verify if it can also detect the attack[5]. If so, packets arriving from suspect sources will be subjected to filtering at this upstream location, and downstream switches will be notified to stop filtering the address (thus effectively *"pushing back"* the attack).

V. Experimental Evaluation

We performed an experimental evaluation to demonstrate the technical feasibility of BUNGEE. Since we are interested in accuracy, resource utilization, reaction time and resource footprint analysis (and not on performance measurements), we carried out an emulation-based evaluation. Additional experiments on real equipment focusing, *e.g.,* on throughput, are left as future work.

[5]Attack detection thresholds in intermediate or edge switches are set up to a (configurable) fraction of those used by the victim's switch. The rationale behind this is that the farther from the victim a switch is, the less aggressively an ongoing attack will be perceived (or even missed completely), since the attack traffic is potentially coming from different sources (see Figure 1).

A. Experiment Setup

Our test infrastructure was instantiated on a virtual machine running Ubuntu 16.04 with four processors and 16 GB RAM. Figure 5 illustrates the instantiated network. It is comprised of a victim network, three attack source edge networks, and four intermediate networks. The forwarding devices depicted in the figure are those that support P4 and are configured to run our mechanism using the BMv2 software switch.

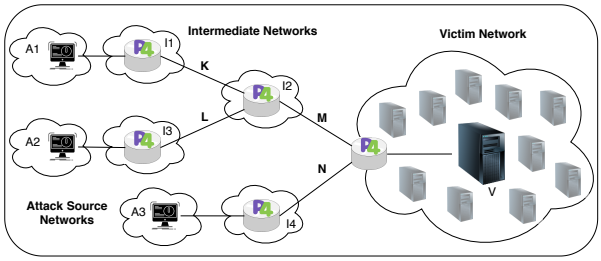

Fig. 5. Topology used in the experiments.

We conducted a sensitivity analysis to determine the parameters of the proposed mechanism. We report the results obtained for the best combination of parameter values, which were the following. The window size was set to 8,192 packets. The threshold to consider an IP address as a suspect was fixed to 1% of the window size (*i.e.,* 82 packets per window). Finally, the filtering component was configured to perform traffic throttling, allowing 30% of suspect packets to follow their way to the victim.

We used the CAIDA Anonymized Internet Traces 2016 dataset to represent legitimate traffic on the network. To perform the attack, we used the CAIDA DDoS Attack 2007 dataset, consisting of anonymized traffic traces from a DDoS attack of around one hour. We generated training and test workloads for each of the three attack sources. The training phase was composed of 150 windows containing legitimate traffic only. As for the test phase, the workload consisted of 300 windows, where the first 75 windows were formed by legitimate traffic, the following 150 windows encompassed a mix of authentic and malicious traffic, and the 75 last windows were, again, legitimate traffic.

In our experiments, three different scenarios were considered, ranging from a confined to a broad attack, as follows:

- *Scenario 1 – One attack source network:* hosts in the source network A1 send a proportion of 90% malicious-to-legitimate packets during the 150 attack windows, while hosts in the other two source networks transmit legitimate traffic all the time.
- *Scenario 2 – Two attack source networks:* hosts in source networks A1 and A2 send a proportion of 45% malicious-to-legitimate packets during the 150 attack windows, while the remaining source network sends legitimate traffic all the time.
- *Scenario 3 – Three attack source networks:* hosts in the three source networks, A1, A2, and A3, transmit each a proportion of 30% malicious-to-legitimate packets during the 150 attack windows.

We analyzed the following evaluation metrics: *detection accuracy*, *network utilization*, *reaction time*, and switch *memory footprint*. Accuracy was measured through *true positives* (ratio of packets coming from suspect IP addresses that were indeed dropped) and *false positives* (proportion of packets coming from legitimate IP addresses that were wrongly dropped). Network utilization was measured considering the amount of link bandwidth consumed by attack packets (without and with the proposed solution activated). Reaction time was calculated as the delay between the beginning of the attack and BUNGEE's filtering start. Finally, memory footprint was measured by calculating the amount of device memory that our mechanism requires to be implemented.

B. Accuracy

Figure 6 shows the accuracy of the proposed mechanism to detect DDoS attacks considering the three scenarios. As one can observe, the detection rate (TPR) is higher than 90% for all cases. Besides, the false-positive rates (FPR) are below 10% for all scenarios. The figure also reveals that the mechanism is nearly equally effective regardless of the origin (single x multiple sources) of the attack. Notice that these results are consonant with the performance of state-of-the-art approaches (*e.g.*, [16], [17]). As we will show next, the gains resulting from our proposal are not necessarily in higher accuracy (already at the top), but substantially lower usage of unnecessary network and processing resources.

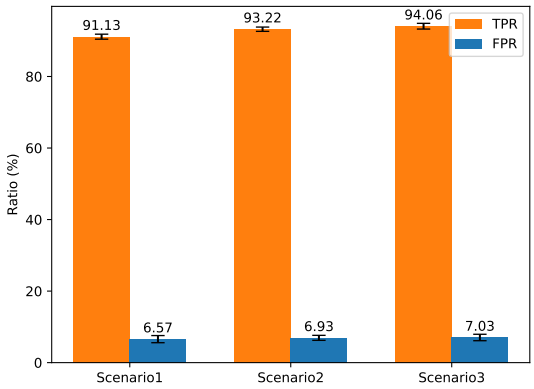

Fig. 6. Accuracy of the proposed mechanism according to True Positive Rate (TPR) and False Positive Rate (FPR).

C. Network Utilization

DDoS attacks cause link saturation. The attack packets end up occupying a substantial share of the links, sometimes exceeding their capacity. As a result, network resources are "wasted", and legitimate traffic may not reach its destination. Figure 7 illustrates the occupation of links K, L, M, N (and aggregated) of the network topology without and with our mechanism enabled. Remember that, in this evaluation, we allow 30% of suspect packets to follow their way to the victim. As one can observe, without the mechanism (orange), network resource consumption with attack traffic ranges from 15% to 25%. With our mechanism activated (blue), these numbers fall

to around 5%. Looking carefully at the aggregated plot, the difference between the orange and the blue areas represent the bandwidth savings obtained by our proposed mechanism, which near 65% with the given parameters.

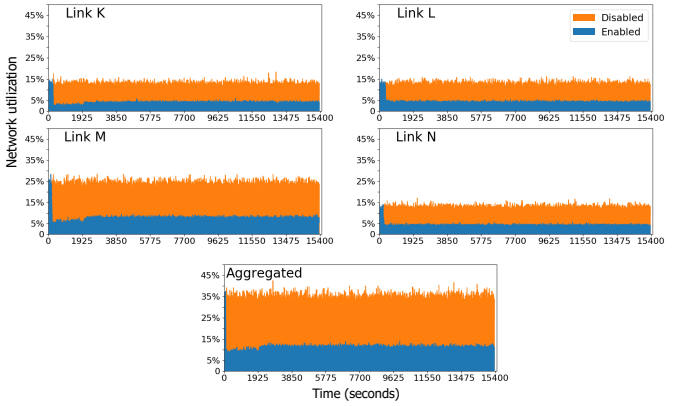

Fig. 7. Network utilization savings resulting from the proposed mechanism.

The mentioned parameters have different effects on BUNGEE: firstly, the monitoring window size affects the processing time to make decisions, changing the reaction time. Larger monitoring windows take longer to process. Secondly, the threshold used to determine if an IP is suspicious affects BUNGEE's accuracy. Finally, the probability of dropping packets in the filtering strategy will trade off overall system accuracy and network utilization.

D. Reaction Time

While packet processing delays are minimal (due to the architecture of a P4-enabled switch, designed to operate at line speed) and can be neglected, the monitoring window size plays an essential role in the time it takes for BUNGEE to activate its defensive actions. Table I shows the theoretical upper-bound for the reaction time of our mechanism as a function of window size. As one can observe, for a window of 8,192 packets, BUNGEE can detect a DDoS attack and initiate its mitigation procedures in as low as 1 millisecond (for a 1 Gbps link) and 0.1 milliseconds (10 Gbps). Even for larger window sizes, the reaction takes place in sub-second time frames. Such low values are achieved mainly due to the design of BUNGEE as a fully in-network mechanism, eliminating the periodic intervention of the control plane controller.

TABLE I
REACTION TIME TO ACTIVATE DEFENSIVE ACTIONS.

Window Size (Packets)	1Gbps (ms)	10 Gbps (ms)
8,192	1.114	0.111
32,768	4.416	0.441
131,072	17.825	1.782
262,144	35.651	3.565
524,288	71.303	7.130

E. Memory Footprint

Each forwarding device instantiates a set of data structures to handle the attack detection and the pushback mitigation mechanisms. For suspect identification, we devised a sketch-based structure [24] composed of six stages (*i.e.,* hash tables). Each hash table has 1,400 entries with 7 bytes each. This accounts for a total of 58,800 bytes. Given a monitoring window of size *ws* packets and a suspect threshold of *k* packets, *ws/k* represents the maximum number of suspects that can exist within a window. Considering our current instantiation of *ws* = 8,192 packets and *k* = 82 packets, we would need room for 100 suspects, while the data structure supports a considerably higher number of entries (8,400). As we illustrate in Table II, many other configurations for both *ws* and *k* would still be supported by the hash data structure (collision-free).

TABLE II
MAXIMUM NUMBER OF SUSPECTS FOR WINDOW SIZE AND THRESHOLD.

Window Size (Packets)	Suspect Threshold (Packets)	Max. Suspects
8,192	41	200
	82	100
	123	67
32,768	41	800
	82	400
	123	267
131,072	41	3,197
	82	1,599
	123	1,066
262,144	41	6,394
	82	3,197
	123	2,132
524,288	41	Not Supported
	82	6,394
	123	4,263

In addition to the data structure described above, each switch instantiates three vectors to maintain the inspection and suspect lists, as described in Section IV. Each entry of these structures occupies 4 bytes to store an IP address. Considering lists of 1,400 entries each, the total number of bytes needed for them is 16,800 bytes. Finally, there is the cost of executing the entropy-based detection mechanism described in previous work [12], which is of 39,040 bytes. Therefore, the total switch memory footprint is of 114,640 bytes, which is deemed very low, given that existing forwarding devices have available memory in the order of hundreds of MB.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed BUNGEE, a mechanism for DDoS detection and mitigation to be entirely executed on a programmable P4 data plane. Our main novel research contribution is the proposal of an adaptive pushback approach that, through entropy-based anomaly detection and inter-switch communication, allows for the optimal, dynamic deployment of filtering procedures. The results obtained reveal the mechanism is accurate and contributes to considerable savings of network resources. Moreover, the solution reacts in a fraction of the time typically demanded by its external CPU and SDN controller-based counterparts. Finally, it consumes a small amount of a forwarding device's memory, enabling other applications to co-exist on the device.

As future work, we plan to focus on refinements to the technique used for suspect identification, including devising dynamically-defined thresholds. In the longer term, we also plant to potentially resort to machine learning-based methods in order to perform suspect identification.

REFERENCES

[1] S. Dong, K. Abbas, and R. Jain, "A survey on distributed denial of service (DDoS) attacks in sdn and cloud computing environments," *IEEE Access*, vol. 7, pp. 80 813–80 828, 2019.

[2] N. A. ATLAS. (2019) 14th annual worldwide infrastructure security report (wisr). [Online]. Available: https://www.netscout.com/report/

[3] J. Porter. (2019) Telegram blames china for 'powerful ddos attack' during hong kong protests. [Online]. Available: https://www.theverge.com/2019/6/13/18677282/telegram-ddos-attack-china-hong-kong-protest-pavel-durov-state-actor-sized-cyberattack

[4] E. G. A. Kupreev, O. Badovskaya. (2020) Ddos attacks in q2 2020. [Online]. Available: https://securelist.com/ddos-attacks-in-q2-2020/98077/

[5] A. Bhardwaj, G. Subrahmanyam, V. Avasthi, H. Sastry, and S. Goundar, "Ddos attacks, new ddos taxonomy and mitigation solutions—a survey," in *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*. IEEE, 2016, pp. 793–798.

[6] P. Kamboj, M. C. Trivedi, V. K. Yadav, and V. K. Singh, "Detection techniques of ddos attacks: A survey," in *2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)*, Oct 2017, pp. 675–679.

[7] A. Santos da Silva, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho, "Atlantic: A framework for anomaly traffic detection, classification, and mitigation in sdn," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 27–35.

[8] N. Agrawal and S. Tapaswi, "Defense mechanisms against ddos attacks in a cloud computing environment: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2019.

[9] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2656877.2656890

[10] F. Paolucci, F. Civerchia, A. Sgambelluri, A. Giorgetti, F. Cugini, and P. Castoldi, "P4 edge node enabling stateful traffic engineering and cyber security," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 11, no. 1, pp. A84–A95, Jan 2019.

[11] R. Hwang, V. Nguyen, and P. Lin, "Statefit: A security framework for sdn programmable data plane model," in *2018 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN)*, Oct 2018, pp. 168–173.

[12] C. Lapolli, J. Adilson Marques, and L. P. Gaspary, "Offloading real-time ddos attack detection to programmable data planes," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, April 2019, pp. 19–27.

[13] M. Dimolianis, A. Pavlidis, and V. Maglaris, "A multi-feature ddos detection schema on p4 network hardware," in *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2020, pp. 1–6.

[14] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A survey on the security of stateful sdn data planes," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1701–1725, 2017.

[15] N. Z. Bawany, J. A. Shamsi, and K. Salah, "Ddos attack detection and mitigation using sdn: Methods, practices, and solutions," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 425–441, Feb 2017. [Online]. Available: https://doi.org/10.1007/s13369-017-2414-5

[16] A. Febro, H. Xiao, and J. Spring, "Distributed sip ddos defense with p4," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, 2019, pp. 1–8.

[17] G. Li, M. Zhang, C. Liu, X. Kong, A. Chen, G. Gu, and H. Duan, "Nethcf: Enabling line-rate and adaptive spoofed ip traffic filtering," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, 2019, pp. 1–12.

[18] R. K. Sanodiya, "Dos attacks: A simulation study," in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, 2017, pp. 2553–2558.

[19] M. Zhang, L. Shi, D. Sisodia, J. Li, and P. Reiher, "On multi-point, in-network filtering of distributed denial-of-service traffic," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, April 2019, pp. 180–188.

[20] N. S. Bülbül and M. Fischer, "Sdn/nfv-based ddos mitigation via pushback," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, June 2020, pp. 1–6.

[21] B. Pande, G. Bhagat, S. Priya, and H. Agrawal, "Detection and mitigation of ddos in sdn," in *2018 Eleventh International Conference on Contemporary Computing (IC3)*, Aug 2018, pp. 1–3.

[22] T. P. L. Consortium. (2020) P4_16 language specification. [Online]. Available: https://p4.org/p4-spec/docs/P4-16-v1.2.1.pdf

[23] K. Kalkan, G. Gür, and F. Alagöz, "Filtering-based defense mechanisms against ddos attacks: A survey," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2761–2773, Dec 2017.

[24] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, "Heavy-hitter detection entirely in the data plane," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 164–176. [Online]. Available: https://doi.org/10.1145/3050220.3063772