Towards a Unified In-Network DDoS Detection and Mitigation Strategy

Kurt Friday*, Elie Kfoury†, Elias Bou-Harb*, Jorge Crichigno†
*The Cyber Center For Security and Analytics, University of Texas at San Antonio, USA
†Integrated Information Technology, University of South Carolina, USA

***Abstract*—Distributed Denial of Service (DDoS) attacks have terrorized our networks for decades, and with attacks now reaching 1.7 Tbps, even the slightest latency in detection and subsequent remediation is enough to bring an entire network down. Though strides have been made to address such maliciousness within the context of Software Defined Networking (SDN), they have ultimately proven ineffective. Fortunately, P4 has recently emerged as a platform-agnostic language for programming the data plane and in turn allowing for customized protocols and packet processing.**

To this end, we propose a first-of-a-kind P4-based detection and mitigation scheme that will not only function as intended regardless of the size of the attack, but will also overcome the vulnerabilities of SDN that have characteristically been exploited by DDoS. Moreover, it successfully defends against the broad spectrum of currently relevant attacks while concurrently emphasizing the Quality of Service (QoS) of legitimate end-users and overall SDN functionality.

We demonstrate the effectiveness of the proposed scheme using a software programmable P4-switch, namely, the Behavorial Model version 2 (BMv2), showing its ability to withstand a variety of DDoS attacks in real-time via three use cases that can be generalized to most contemporary attack vectors. Specifically, the results substantiate that the mechanism herein is orders of magnitude faster than traditional polling techniques (e.g., NetFlow or sFlow) while minimizing the impact on benign traffic. We concur that the approach's design particularities facilitate seamless and scalable deployments in high-speed networks requiring line-rate functionality, in addition to being generic enough to be integrated into viable network topologies.

***Index Terms*—P4, Distributed Denial of Service, Data Plane, In-Network, Real-Time**

I. Introduction

While control plane strategies offer several security advantages over traditional network architectures, these relatively newfound technologies have a large undertaking with respect to DDoS. Indeed, this classification of misdemeanors has been compromising the functionality of our networks for many years; comprised of the notorious attacks on CNN, Yahoo, Ebay, and Amazon in 2000; the specific targeting of DNS servers in both 2002 and 2007; the assaults on Mastercard and Visa in 2010; the bringing down of the main website of the Malaysian Government for days in 2011; and the disconnecting of several highly sought after Internet services, e.g., Amazon and GitHub, in 2016. Moreover, not only have these threats demonstrated an uncanny ability to devastate today's networks [1–3], but they have also increased by 84% within the first quarter of 2019 alone [4]. Further, this excessive growth was also not only observed in the number of recorded DDoS incidents that persisted for upwards of an hour, but also in the average duration of time that these logged attacks were executed for. Undoubtedly, that combined with the record-breaking attack of 2018 which used memcache as a means of amplification in order to reach a shocking 1.7 Tbps necessitates the dire need for real-time detection followed by prompt mitigation. Though DDoS comes in many forms, we effectively perform a binary classification of such maliciousness for the sake of our approach herein, as subsequently detailed.

1) Stealthy Approaches: First and foremost, a sophisticated attack vector that has warranted substantial attention from both the research and industry communities alike is that of a stealthier category, often simply referred to as slow DDoS. While there are several variations such as R U Dead Yet? (R.U.D.Y.) [5] or Slowloris [6], they each ultimately endeavor to tie up the server's available connections to incite Denial of Service (DoS) to authentic clients. Furthermore, not only does slow DDoS utilize legitimate Transmission Control Protocol (TCP) behavior, but malicious packets are sent at a frequency similar in intensity to that of benign traffic and will thus reach the victim undetected by traditional anomaly and signature detection techniques [7].

2) Volumetric Attack Vectors: Similarly to that of the aforesaid stealthy approaches, volumetric DDoS varieties leverage valid protocols as well; in turn, detecting and properly mitigating such conventional tactics continues to persist as an incredibly challenging and elusive objective. Generally, these attacks are primarily associated with the aim of saturating bandwidth and flooding targets with attack traffic by way of User Datagram Protocol (UDP) or Internet Control Message Protocol (ICMP), aside from that of TCP employing mixtures of set flags within the headers of packets which do not correspond to any valid sessions on the server. Furthermore, malicious entities will use these protocols for adding a means of amplification to their attacks, allowing them to send requests that result in response traffic reaching magnitudes far exceeding that sent, and even leading to DDoS throughput of upwards of Tbps [8]. Moreover, because no authentic sessions are required to execute such exploits, adversaries typically evade detection via spoofing their source IP addresses [9, 10], on top of harnessing it for reflection purposes.

To this end, we endeavor to advance the state-of-the-art not only by way of delegating more responsibility to the data plane but also by making the following core contributions:

- Designing and developing a first-of-a-kind DDoS detection and mitigation engine that tackles a broad spectrum of attacks in real-time solely by analyzing one-way ingress traffic on the switch.
- Placing unparalleled emphasis on practicality and end-user QoS via its lightweight, switch-based methodology that dynamically adjusts to input parameters based off the unique constraints of the network in which it's implemented, and in turn possesses the ability to be dispensed in any network using any number of programmable switches.
- Nullifying all flow table and control channel saturation vulnerabilities by way of fixing both the amount of storage utilized on the switch and its communications with the control plane, irrespective of the rate of traffic traversing the device.
- Empirically evaluating the model's ability to counter slow DDoS attacks and varying volumetric types amid real attack scenarios resulting in 3.5% and 0% of legitimate clients experiencing negligible delays for the two aforementioned classes of attacks, respectively, while evading a DoS state in all instances. Further, the typical latencies observed in past approaches for complete slow DDoS mitigation are reduced from generally upwards of many seconds down to under two, as well as demonstrating a detection latency of below 0.25 seconds for SYN floods and circumventing such delays entirely among other volumetric attack vectors.
- Completely negating all TCP flooding attacks that strive to impersonate authentic sessions with the server solely in-network.
- Providing a blueprint for harnessing the abilities of programmable switches to provide enhanced network security measures that instrument line-rate packet analysis and offer accurate telemetry, in order to facilitate its widespread adoption as a highly effective means of safeguarding networks, given the untapped potential of P4 within SDN security provisioning and research.

The remainder of this paper is organized as follows. In Section II, we elaborate on recent works related to DDoS attack detection that rely upon polling techniques (e.g., NetFlow or sFlow), and detail all associated research gaps that were pinpointed in the process. To this extent, Section III articulates how the proposed approach endeavors to mitigate such shortcomings. Following the firm establishment of this work's objectives, Section IV introduces P4 and describes the significance of its incorporation for programming the data plane, and how said advantage was leveraged to facilitate the proposed in-network DDoS attack detection and mitigation scheme. Subsequently, Section V presents the evaluation methodology in the form of three use cases, and by way of the corresponding results, empirically validates the proposed strategy's ability to not only bridge the aforementioned gaps within the literature, but to do so in an efficient and practical manner. Lastly, Section VI concludes the paper with final remarks and aspirations going forward.

II. Related Work

Nearly all recent works that have pioneered SDN security [11] development are rooted in the OpenFlow protocol. One of the most noteworthy is undoubtedly AVANT-GUARD [12] which aspired to address control channel saturation attacks, though it necessitated an OpenFlow extension to allow for data plane rule modifications–a feature that P4 now does inherently. In the years to follow, an assortment of other anti-DDoS developments reliant upon OpenFlow have been made [13–16]. Further, with the prevailing concern of slow DDoS attacks targeting our networks, the aforementioned protocol was also harnessed in a plethora of other research endeavors dedicated to addressing this stealthy yet devastating avenue of maliciousness [7, 17].

As of late, more attention has been brought to OpenFlow's inability to control the data plane's internal behavior. To this end, programmable switches enable the customization of the data plane's behavior and access to real-time metadata (e.g., header fields and counters), and thus greatly promote customized network functionality and overhead reduction. That being said, research pertaining to programming said switches with P4 is substantially lacking within the security domain. Aside from heavy hitter detection [18, 19] which indirectly applies to volumetric attacks but generally fails to resolve the issue of source IP address spoofing which coincides with such flooding DDoS varieties, to date we have identified only the entropy-centric in-network flooding detection approach of Lapolli et al. [20] as being within the scope of that which we propose. That being said, their implementation is largely based on complex source code not feasible for real hardware implementations.

In a nutshell, each of these works offered valuable insight pertaining to the inner-workings of the aforementioned misdemeanors; however, despite the progress made, research gaps in desperate need of filling still persist, as summarized in Table I. As a result, such voids are adopted as motivations for the proposed approach herein.

III. Methodological Considerations

To emphasize the aim of this work to pervade the above-mentioned research gaps, we subsequently elaborate upon how these aspects were addressed to facilitate future in-network solutions.

A. Efficient Network Integration

The P4 programmable switch is deployed at the network's edge, which is in fact the only requirement for implementation being that the proposed approach endeavors to conduct one-way ingress traffic analysis prior to it interfacing with the network's internal devices. Furthermore, it precedes all network-specific switch applications, and in turn, the entirety of attack vectors aiming to exploit flow table and control channel saturation vulnerabilities are negated. As a result, network administrators can be confident that all succeeding

Challenges	Description
Lack of a unified detection paradigm	There is an evident need for an all-encompassing defense mechanism against the vast spectrum of DDoS classifications.
Source attribution	Given the notable overhead associated with attempting to attribute spoofed sources, the contemporary security landscape is in dire need of a more efficient alternative.
Innate SDN shortcomings	Flow table and control channel saturation attacks have yet to be eradicated without incurring considerable costs, and SDN's centralized control predisposes its control channel to becoming a bottleneck; thus, a more robust means of communication is warranted.
Arbitrary threshold and detection techniques	With the ever-evolving copious technologies and network architectures in mind, developing a scalable, portable, and resilient DDoS detection and mitigation scheme that isn't dependent up a given configuration is of the essence.
Explicit end-user QoS considerations	The collateral damage of DDoS with respect to benign traffic is exorbitant yet often acquires subpar attention among proposed defenses, and therefore thorough consideration is merited.

TABLE I: Existing SDN-based research gaps pertaining to DDoS

operations will be executed in a secure manner, on top of having a clear modular structure separate from typical network functionality for simplified management, provisioning, and maintenance.

B. End-User Quality of Service (QoS)

To address the key challenge of enhancing QoS for legitimate end-users in the midst of DDoS attacks, the methodology herein takes a number of unequivocal measures to minimize such collateral damage. As ingress traffic arrives at the switch, the P4 programmable parser segregates it by ICMP, UDP, and TCP so exploits can be resolved on the basis of the underlying protocol. Subsequently, stateful switch-based counting mechanisms are utilized to form dynamic distributions of current traffic patterns over 10 second moving averages [21] in order to decrease the probability of impeding the traversal of that from legitimate sources when an attack transpires. Note that said averages were limited to 10 seconds based upon the underlying notion that too extensive of such an interval may not sufficiently account for the most recently measured trends due to excessive smoothing, as well as being more computationally expensive. For traffic of the SYN request variety, signatures are tracked to promote source attribution and the ensuing blocking of the corresponding malicious packets amid spoofing to better ensure that requests originating from benign entities are not adversely affected.

With respect to the spectrum of suspicious traffic that may traverse the switch, dropping only occurs to remain strictly within the bounds of the given network's resources; in turn, not only is the impact on authentic nodes minimized, but the potential flash events or users with slow connections which may present themselves as volumetric or stealthy attacks, respectively, are also accounted for. Moreover, if an attack transpires, all mitigation measures are executed temporarily on an as-needed basis; therefore, the event that the attacking entities are merely legitimate users that were infected and temporarily leveraged by a botmaster [22] is effectively addressed. Lastly, in the unlikely scenario that a legitimate user was denied interaction with the network amid an attack, full access to its services will resume once the thresholds are satisfied by said mitigation effort, which in fact can equate to seemingly no packet loss from a TCP user's perspective when considering its automatic retransmission feature.

C. Administrator Input

Building upon the aim of assisting the proposed strategy's seamless integration into contemporary network environments consisting of copious architectures and associated hardware, the detection and mitigation approach allows for the passing of the network-specific parameters detailed in Table II, either upon initialization or during runtime via conveying them directly to the switch.

Parameters	Description
`syn_queue_size`	Acts as the maximum threshold for ingress SYN traffic prior to receiving its corresponding ACK
`timeout`	A unique timeout value implemented by the administrator on the server to enable in-network TCP connection management
`max_clients`	The number of threads allocated by the administrator on the server for concurrent TCP connections
`bandwidth_spec[0..3]`	Four administrator-defined acceptable throughputs (in Bps) that the switch will constrain ingress traffic as a whole to (mandatory), as well as individually (optional) that of ICMP, UDP, and TCP, respectively

TABLE II: Input parameters facilitating customization for a given network

D. High-Speed Performance

To promote the ever-increasing need for throughput and its associated processing, this research utilizes a switch-centered means of functioning which allows it to perform packet analysis within the data plane, without the need for controller intervention. Note that P4 programmable switches now have the capacity to process packets at rates of up to 12.8 Tbps which effectively renders alternative communications with the control plane a bottleneck. Moreover, match-action dependencies are minimized programmatically to leverage the full parallel-processing capability of the switch in order to maximize the number of operations that can be completed within one cycle.

E. SDN Attack-Resistance

Though past OpenFlow-based research devoted to remediating the vulnerabilities of SDN has demonstrated relative success, considerable network overhead and latency are typically incurred in the process [23], and ultimately it has yet to be fully eradicated. To address this problem, the proposed approach first prevents control channel saturation via maintaining a fixed control channel throughput irrespective

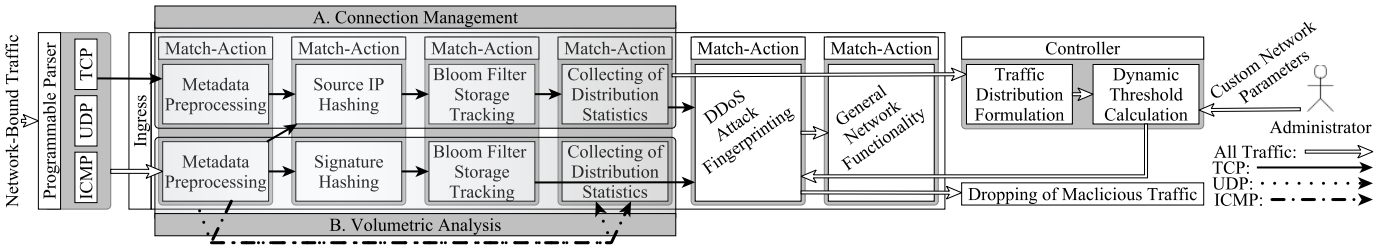

Fig. 1: Overview of the proposed methodology

of ingress traffic traversing the switch. Secondly, just as the rate of control channel communications remains static, so too does the amount of memory leveraged for the stateful logging of traffic-related data on the switch in order to nullify flow table saturation attack vectors. Specifically, all traffic-related information is stored within a predefined number of counters and registers, and it is precisely this data that is pulled by the controller at pre-established one second intervals. For example, the storing of such statistics relative to detecting TCP-based attacks is achieved via Bloom filters, or arrays of registers instantiated on the switch.

Bloom filters offer the distinct advantages of being storage efficient while simultaneously giving a desirable runtime of $O(1)$, with the one caveat being potential false positives. The probability of such undesirable events occurring is given by the following equation:

$$p = (1 - e^{-\frac{k}{m/n}})^k,$$

where p, k, m, and n are the probability of a false positive, number of hash functions used, amount of items in the filter, and total number of bits encompassed by the filter, respectively. Taking the Bloom filter employed for the fingerprinting of slow DDoS as an illustration, when applying one Cyclic Redundancy Check (CRC) 16-bit hash function in conjunction with the 65,536 cells both Bloom filters contain and letting $m = 256$, i.e., the arbitrary amount assigned to `max_clients` in the evaluation conducted in Section V-C, $p \approx 0.0038986$ is arrived at. Note that the methodology only maintains such stateful data for mere seconds; thus, the size of m will generally remain sufficiently negligible, and in turn so will p.

IV. Proposed Detection and Mitigation Design

A. Protocol Independent Packet Processors (P4)

P4 not only enables the programming of the data plane but also facilitates portable implementations over hardware targets, and therefore optimizes network integration efficiency. That being said, while there has been efforts devoted to advancing the P4 compiler and BMv2, the demonstration of its advantages and innovative potential for anomaly detection in practice has been scarcely reported by the scientific literature [24]. In turn, to address the aforementioned research gaps surrounding the current cybersecurity landscape with respect to DDoS and to facilitate future P4 development in this regard, we subsequently detail how P4's extensive capabilities were leveraged to perform per-packet analysis for real-time DDoS detection and mitigation within the switch itself.

B. P4-Based DDoS Abatement Overview

Following suit with the contemporary threat model laid out in Section I, the methodology is effectively compartmentalized into two distinct stages, namely, Connection Management and Volumetric Analysis, as can be seen in parts A and B of Fig. 1, respectively, and are elaborated upon next.

Connection Management. After a packet is identified as utilizing TCP, there are essentially two scenarios that must be accounted for: (1) excessive SYN requests and (2) consuming the server's available connections. Considering the inevitable spoofing of the source IP addresses that accompanies the former, in order to circumvent the aforesaid pitfalls associated with source attribution attempts, signature matching is employed by way of hashing the portions of ingress SYN packets that have the tendency to imply source configurations, as demonstrated in Fig. 2. The underlying notion is that botnet-orchestrated DDoS attacks, which are largely responsible for such coordinated maliciousness, will generally target specific vulnerabilities such as that encompassed by a given Operating System (OS) version that has yet to be patched appropriately, and consequently there exists a strong likelihood that the infected bots will induce the same signatures [25]. In turn, such attacks can be mitigated via one-way traffic analysis, which in turn circumvents any incurred latency resulting from more expensive two-way measures, e.g., IP traceback or SYN cookie techniques [12].

To this end, signature counts are accumulated in the data plane via the Bloom filter shown in Fig. 2, as well as the number of SYN requests per 0.25 second window noting that the width of this measurement window is rather arbitrary and can be scaled as an administrator deems fit. As a result, the controller can promptly calculate the 10 second moving average of SYN requests and proactively store said average plus two standard deviations on the switch within the DDoS Attack Fingerprinting match-action tables of Fig. 1.A to be used as a dynamic threshold, and thereby facilitating real-time detection, at line rate. If this threshold is surpassed, the disturbance is attributed to the maximum signature counts, of whose corresponding packets are dropped upon arrival at the switch as subsequently shown in Fig. 1. Note that the fallback threshold of `syn_queue_size` given in Table II is also

imposed on the switch to ensure the network will never reach a state of DoS.

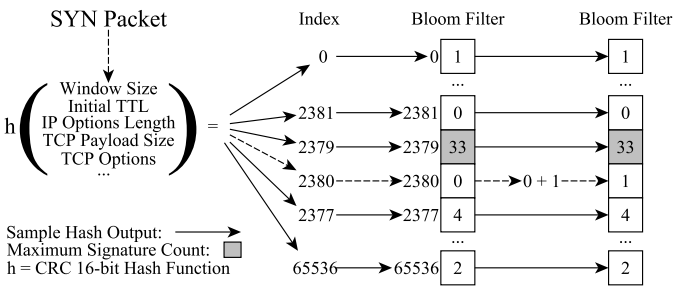

Fig. 2: The Bloom filter-based SYN request signature tracking mechanism residing within the data plane

Regarding the remaining TCP transmissions, and with the premise of slow DDoS aiming to occupy sessions on thread-based servers in mind, all active TCP sessions are statefully tracked on the switch by way of the Bloom filter shown in Fig. 3. As depicted, it is here that the switch stores the interarrival times of each connection, which is the fundamental characteristic for distinguishing such attacks. Moreover, the in-network tracking of the server's active sessions enables the switch to eradicate all other assortments of TCP flooding attacks encompassing packets not associated with any current valid connection.

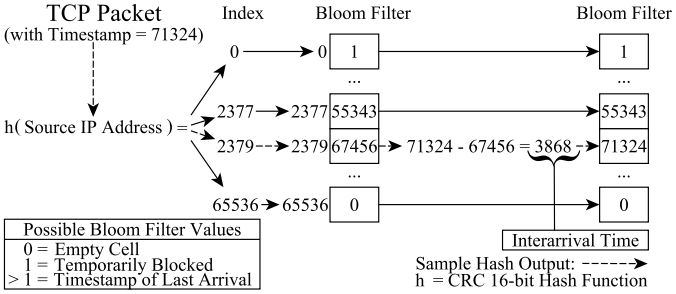

Fig. 3: The in-network management of the current TCP sessions by way of a Bloom filter

From a statistical perspective, we associate direct P4 counters with match-action table entries comprised by the Collecting of Distribution Statistics portion of the match-action pipeline in Fig. 1.A, which are in turn pulled by the controller to formulate a precise distribution of the current connections with respect to their transmission rates. The connection management component also maintains the maximum number of consecutive session establishments c_{est} without any existing sessions being closed over a one second window, which is taken as one standard deviation over a 10 second moving average of such establishments recorded on the switch. To this extent, c_{est} dynamically embodies the maximum burst of authentic connection requests that the switch should account for during any given time window. Therefore, applying said averages to a Probability Density Function (PDF) f of random variable X of interarrival times in seconds, we arrive at the following equation:

$$\int_{a}^{\infty} f(x)dx = \frac{c_{est}}{\texttt{max\_clients}}, \tag{1}$$

with a being the dynamic threshold and `max_clients` noted in Table II. As previously mentioned, because bursts of connection establishments within the small one second time window utilized is within the realm of possibilities, the proposed mechanism aims to preserve c_{est} available connections at all times by way of dropping packets corresponding to $x \geq a$ when necessary. Lastly, given the lack of precedent set within the context of slow DDoS detection, it can however be noted that such attacks exhibit $a \geq 10$ [5] [6], coupled with the fact that networks will typically implement `timeout`$>$ 10 seconds (detailed in Table II) where `timeout`$\in \mathbb{N}$ [7]; thus, the forcing of malicious connections to inevitably timeout will generally result in many more legitimate users' requests being denied due to a resultant mitigation latency equating to `timeout`. To address this challenge, the mitigation strategy proposed herein modifies the packet headers of identified adversarial ingress traffic on the switch via P4 to immediately cause the respective connections to terminate on the server.

Once the aforementioned data is garnered by the Collection of Distribution Statistics segment of the Connection Management component, it is pulled by the controller for Traffic Distribution Formulation. In practice, the controller obtains the switch's interarrival time counts as an array. Given the set

$$S_i = \{x \mid x \in \mathbb{Q} \wedge i * 0.01 \leq x < (i+1) * 0.01\},$$

we can define such an array $\boldsymbol{A}$ as follows:

$$\boldsymbol{A}[i] = \sum_{s \in |S_i|} 1, \forall i \in \mathbb{N}(i \leq 100 * \texttt{timeout}),$$

recalling that x is the interarrival time of a particular connection measured on the switch in seconds. With $\boldsymbol{A}$ encompassing discrete values, Eq. (1) is framed as a Probability Mass Function (PMF) on the controller via Algorithm 1 in order to compute the dynamic threshold a to be returned to the switch to enable the real-time in-network detection of slow DDoS. As it can be observed, Algorithm 1 sums over the probabilities of the recorded interarrival times in ascending order by way of pdf_est until $c_{est}/\texttt{max\_clients}$ is reached, with line 8 ensuring $\sum_{x \in \boldsymbol{A}} p(x) = 1$. Upon completion of the algorithm's iterations, $a = i$ * 0.01 is returned to the switch which corresponds to the aforementioned set S_i, and consequently any source whose transmission rates are encompassed by or exceed said set's range is deemed malicious.

Volumetric Analysis. Operating under the assumption of the proportionality of bandwidth consumption to resource depletion of the target, the primary objective of the Volumetric Analysis component in Fig. 1.B is to statefully record the bandwidth used by various applications and their respective transport protocols to promote the necessary mitigation measures amid an attack. In addition to monitoring the rates in bps allocated to each TCP connection, the switch also tracks the

Algorithm 1: Slow DDoS threshold calculation

Input: $\boldsymbol{A}, c_{est}, \texttt{max\_clients}$
Output: a

$i = 0$;
$dist_{sum} = 0$;
$pdf\_est = 0$;
for $x \in \boldsymbol{A}$ **do**
  $dist_{sum} = dist_{sum} + x$;
end
while $i < |\boldsymbol{A}|$ **do**
  $pdf\_est = pdf\_est + (\boldsymbol{A}[i]/dist_{sum})$;
  if $(1 - pdf\_est) \leq (c_{est}/\texttt{max\_clients})$ **then**
    return $i * 0.01$;
  end
  $i = i + 1$;
end

amount of bandwidth of `bandwidth_spec[1]` from Table II that is consumed by ICMP versus adopting the approach taken by many modern-day networks of simply blocking all ICMP traffic at the edge for security purposes. Specifically, the proposed strategy drops such ICMP traffic when the header field Type is equal to 0, 3, 4, 5, 8, and several others that have been deprecated. This process not only aids the mitigating of flooding attacks but also negates sub-classes of ICMP-related attacks and vulnerabilities [26]. Subsequently, the bandwidth associated with the remaining ICMP traffic is recorded and bounded to `bandwidth_spec[1]` on the switch.

In a manner similar to that exercised by the Connection Management engine of Fig. 1.A, UDP traffic traverses the constituent tables of the Volumetric Analysis component in Fig. 1.B to promptly arrive at a real-time distribution calculation; however, alternatively to tracking interarrival times, the bps per application layer protocol is recorded with corresponding 10 second moving average calculations. Specifically, the switch holds an individual dynamic threshold for each UDP-encapsulated protocol found by Kührer et al. [8] to be at the greatest risk for amplification exploitations, with an additional dynamic threshold accounting for the aggregation of the remainder of UDP-based protocols that do not provide any semblance of said amplification. Note that though UDP amplification attacks can not be concretely verified without Deep Packet Inspection (DPI), it is later empirically demonstrated that such exploits are successfully mitigated by the proposed approach. Moreover, by way of additionally analyzing the destination ports of incoming UDP traffic (versus the originating ports) on the switch in a comparable fashion and multiplying the respective UDP throughput by the associated amplification factor, the proposed strategy vanquishes attempts by adversaries to leverage the network's servers as a means of amplification towards other networks/assets as well.

V. Evaluation

With the methodology confronting the broad scope of respective attack vectors that are currently relevant, the proceeding three use cases effectively act as an overarching strategy of the approach's underlying mechanisms, and in turn facilitate the generalization of the results to the vast spectrum of DDoS as a whole. The experimental testbed was executed on Mininet in conjunction with BMv2 over Ubuntu 16.04.6 LTS, functioning with 16GB of memory and eight Intel Xeon Gold 6130 CPUs at 2.10GHz. As depicted in Fig. 4, to emulate the aforementioned scenarios, six clients were connected to a virtualized P4-programmed switch by way of Linux network namespaces, which in turn directed the corresponding traffic towards the target server. Additionally, all evaluation statistics were gathered on the switch itself via P4 and pulled by the controller for analysis.

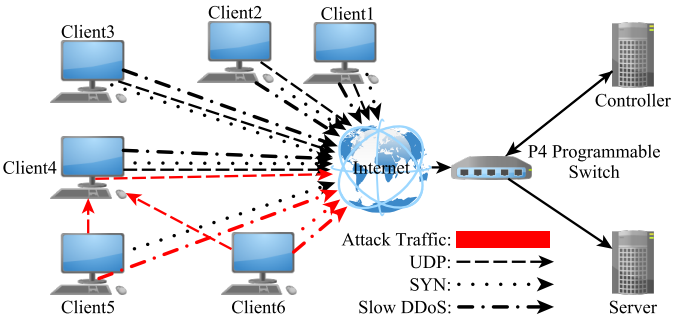

Fig. 4: The topology of the evaluation environment

A. Use Case 1: UDP Amplification

Considering that the strategy employed for detecting and mitigating volumetric ICMP attacks is effectively a subclass of that employed for UDP-based varieties, we thereby utilize a UDP flooding scenario leveraging DNS and NTP for attack traffic amplification as a representative sample of the effectiveness of both mechanisms. Further, to implement such a real-world scenario, Client4 in Fig. 4 was configured to emulate a resolver to consequently respond to both DNS and NTP requests of the corresponding source IP address (spoofed by the originating client to the server's IP) with a level of amplification in accordance with [8]. `bandwidth_spec[2]` noted in Table II was set to 300 Mbps, and the `hping3` Linux tool was used for traffic generation. Since the approach dynamically adjusts its thresholds based upon the traffic distributions it formulates in real-time from line-rate measurements, this use case was initiated by generating NTP, DNS, as well as other UDP traffic encapsulating randomized destination ports not associated with amplification, in order to establish a baseline of expected ingress traffic rates and protocols. Specifically, as shown in Fig. 4, Client1 and Client2 sent benign DNS and NTP traffic, respectively, at approximately 714 datagrams/sec to the server. To additionally represent a network environment encompassing an assortment of UDP protocols not associated with any amplification, Client3 transmitted such legitimate traffic at a rate of 2,856 datagrams/sec, doubling that of the UDP protocols exhibiting amplification to better account for networks servicing a variety of UDP protocols. An arbitrary 10 seconds was deemed sufficient to establish said baseline,

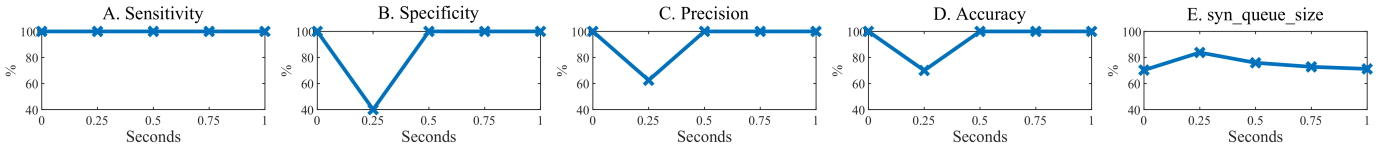

Fig. 5: SYN flooding mitigation results

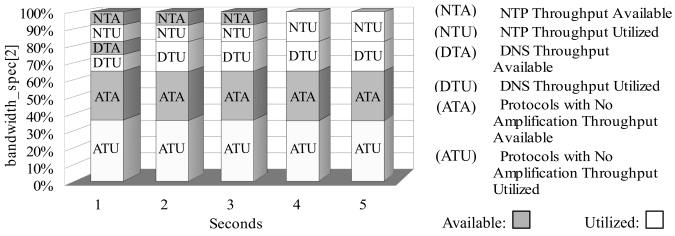

Fig. 6: UDP Amplification attack mitigation results

at which point Client5 began flooding Client4 with DNS traffic spoofed with the source IP address of the server shown at second 1 of Fig. 6. By second 2 of Fig. 6, it can be observed that while the available DNS throughput had been fully consumed, neither NTP nor any of the UDP protocols which do not exhibit amplification were affected by the attack. In fact, both of these groupings were able to increase their rates amid the DNS amplification with no degradation in service or packet loss. In the midst of said attack, Client6 launched another at second 3 of the use case utilizing NTP as a means of amplification which was reflected off Client4 towards the server. Analogously to when the network was subjected to only DNS amplification, the remaining protocols (those not corresponding to any amplification in this instance) experienced no latency or packet drops while the network was under two separate amplification attacks and were able to effectively double their transmission rates collectively to 5712 datagrams/sec towards the server unimpeded, which was the allotted share of `bandwidth_spec[2]` defined in Table II that was dynamically established prior to the network being attacked.

B. Use Case 2: SYN Flooding

A SYN flooding attack was chosen for Use Case 2 given the underlying mechanisms of signature hashing and tracking via a Bloom filter that the strategy employs in such a scenario. For this use case, a `syn_queue_size` value of 1024 was passed to the virtual programmable switch, and `hping3` was again used to generate SYN requests to the server. To emulate traffic that the network would typically observe, Clients 1 through 5 each transmitted 600 SYN requests per second to the server with subsequent ACKs. Following the topology set forth in Fig. 4, at an arbitrary 10 seconds later Client6 began targeting the server with malicious SYN requests peaking at approximately 2000 packets/sec as depicted at the 0 second mark in Fig. 5. Given that all traffic is originating from the exact same OS environment, the configurations of each of the six client machines were modified to better mimic the diversity that would be observed in real-world settings in order to effectively evaluate the signature attribution mechanism.

To empirically determine such an approach's effectiveness, a binary classification of the generated traffic was applied to the use case's results, with the positive class encompassing benign traffic and the negative class pertaining to that with malicious intent. In turn, the results portrayed in Fig. 5 utilize both the true positives (TP), or correctly classified legitimate requests, and false negatives (FN) which are those that were incorrectly identified as being part of the attack. Additionally, the malicious requests properly fingerprinted were deemed true negatives (TN) and those which were wrongly classified as benign as false positives (FP). From the aforementioned performance metrics, $sensitivity$, $specificity$, $precision$, and $accuracy$ are derived by way of the equations $TP/(TP+FN)$, $TN/(TN+FP)$, $TP/(TP+FP)$, and $(TP+TN)/(TP+TN+FP+FN)$, and are depicted in parts A, B, C, and D of Fig. 5, respectively. As given by the $sensitivity$, all of the legitimate requests were treated as such throughout the duration of the attack. With SYN flood initiating at time 0, it can be additionally observed that the proposed approach demonstrated a dip in performance which peaked at roughly the 0.25 second mark and can be attributed to the length of time needed to observe the signature deviation corresponding to the attacking machine; however, by 0.5 seconds, a flawless performance was given by all metrics. Furthermore, all benign requests were serviced without any observed latency throughout the course of the use case, and as shown in Fig. 5.E, the SYN queue never exceeded its conservative `syn_queue_size` imposed for evaluation purposes. In fact, the SYN flooding packets were effectively fingerprinted and blocked entirely by 0.5 seconds, which ultimately only resulted in the occupancy of the SYN queue increasing by approximately 13%. It is also worth mentioning that due to the proactive design of the detection and mitigation engine coupled with its calculated dynamic thresholds that are exercised on a per-packet basis, the 0.25 sec time window taken to depict the results is due to how often the control plane outputted the switch-based packet statistics, and in all actuality it does not speak to the speed of the approach. In fact, in a manner analogous to that which was conducted in Use Case 1 and as articulated in Fig. 1, the proactive calculation of the dynamic threshold imposed directly on the switch is conducted independently of the attack fingerprinting segment of the match-action pipeline which is executed at line rate, on a per-packet basis.

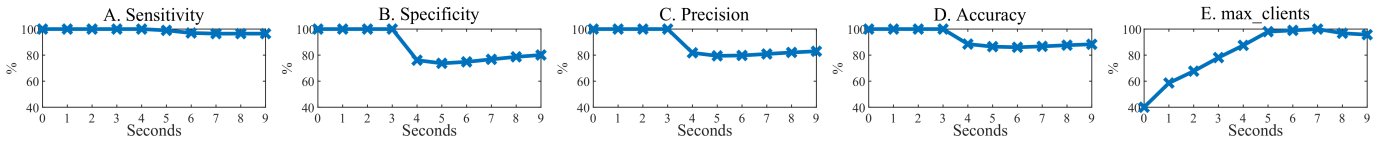

Fig. 7: Slow DDoS attack mitigation performance

C. Use Case 3: Slow DDoS

Given that the administrator-specified `max_clients` parameter tailors the proposed approach to a specific network, it was opted to take a more conservative evaluation scenario (`max_clients` $= 256$) to better put the scheme's full capabilities to the test given the smaller margin for error. With regards to the generated traffic, it is first notable to recall that slow DDoS attacks thrive off interarrival times that are near but not exceeding `timeout` in order to conserve the adversary's resources while simultaneously keeping the server's connections busy; thus, the transmission rates of the adversarial connections were increased accordingly for this use case.

In particular, considering that the interarrival times of all the traffic directed towards a server can effectively be divided into three transmission speed groupings encapsulating legitimate users whose connections support typical transport rates, those subjected to environments inducing more pronounced latency (e.g., traffic originating from more sparsely-connected regions), and that from malicious entities which inherently coincide with slower interarrival times, $t_{b\_fast} \in \mathbb{Q}(0.00 < t_{b\_fast} \leq 1.00)$, $t_{b\_slow} \in \mathbb{Q}(1.00 < t_{b\_slow} \leq 2.50)$, and $t_m \in \mathbb{Q}(1.75 < t_m \leq 5.00)$ were consequently applied, respectively. Further, t_{b_fast}, t_{b_slow}, and t_m were generated in a randomized fashion over their respective intervals in order to better circumvent any distribution imbalances of interarrival times within each, i.e., $t_{b\_fast} \sim U(0.01, 1.00)$, $t_{b\_slow} \sim U(1.01, 2.50)$, and $t_m \sim U(1.75, 5.00)$. Note that the aforementioned groupings result in the set $S_{intersect} = \{t_{intersect} \mid t_{intersect} = t_{b\_slow} \wedge t_{intersect} = t_m\}$, and therefore $t_{intersect} \sim U(1.75, 2.50)$, i.e., immensely hindering the accurate fingerprinting of slow DDoS traffic by way of interarrival times being that $|S_{intersect}|$ is much greater in this use case than would be observed in typical real-world scenarios given the lack of distinction between t_{b_slow} and t_m.

Once more harnessing `hping3` for traffic generation and following the blueprint laid out in Fig. 4, the first of the connections to be established were that associated with t_{b_fast}. With 50 connections each originating from Clients 1 through 3, 150 resultant threads, or 58.59% of the available connections, thereby became occupied as depicted at second 1 of Fig. 7.E. Following this first wave of transmissions, 50 additional threads were then consumed by Client4 as given by 78.13% of `max_clients` shown to be expended at second 3 of Fig. 7.E. In order to create a realistic traffic pattern representing the modest number of legitimate sources being restricted to slower connection speeds, the interarrival times corresponding to the aforementioned 50 sessions were encompassed by t_{b_slow}. At approximately second 4, Clients 5 and 6 then began generating slow DDoS traffic leveraging the interarrival times encapsulated by t_m with the aim of overtaking 128 connections each. This resulted in 22.27% of `max_clients` initially being exhausted at second 5 by the attack as displayed in Fig. 7.E, and ultimately increasing to 24.61% two seconds later. At this juncture, the proposed approach had successfully fingerprinted and subsequently denied 193, or 75.39% of the malicious sources at the switch. Furthermore, 11 more had been identified to drop by second 10, which was in fact the dynamic value calculated for c_{est} in this use case, thus leaving a minimum of c_{est} threads open on the server for expected incoming legitimate connections.

The performance metrics utilized in Section V-B were once again harnessed to empirically evaluate the approach's aptitude for slow DDoS detection and mitigation. As exhibited in Fig. 7.A, the proportion of the legitimate sessions that were correctly identified as such was impeccable, never extending below 96.5% and thereby reaffirming the aforementioned prioritization of QoS for benign clients. Alternatively, it can be observed by way of the $specificity$, $precision$, and $accuracy$ in Fig. 7 that the lowest effectiveness was given at the 5 second mark when the attack traffic first reached its peak and in turn had not been fully fingerprinted yet. Nevertheless, the model's performance steadily climbed form that point until reaching its ultimate goal of freeing c_{est} threads on the server. To this extent, achieving this aim necessitated an $accuracy$ of 88.32% which was the maximum that could have been attained given the considerable magnitude of $S_{intersect}$ employed in this use case. Moreover, this gives credence to the underlying objective of the proposed approach of always striving to maximize its $sensitivity$ while simultaneously endeavoring to reach a sufficient $specificity$ and $precision$, and by default $accuracy$, such that $c_{tot} - c_{av} \leq c_{est}$, given the collateral damage of DDoS attacks pertaining to end-user QoS inevitably follows an $|S_{intersect}| > 0$.

It should also be stated that `max_clients` was never exceeded throughout the duration of this use case, and only 7 of the initial 200 benign connections temporarily lost the rights to their given thread due to the the attack as a direct result of $|S_{intersect}|$. With regards to the mitigation effort, it only took two seconds after exhausting the server's threads for the Connection Management component of Fig. 1.A to open up c_{est} connections for new users, with eight of such connections ($72.\overline{72}\%$ of c_{est}) being available at one second after the attack's onset; thus, the resultant automatic retransmissions of the aforementioned 7 legitimate clients would allow them to

immediately be serviced again with negligible latency being their transmission rates are a subset of $S_{intersect}$, i.e., virtually indistinguishable from the mitigation delay.

VI. Concluding Remarks and Future Direction

The leveraging of programmable switches by way of P4 was examined in order to target the extensive assortment of DDoS attacks that persist within the modern-day networking landscape while simultaneously circumventing the associated vulnerabilities of SDN. By way of three use cases, the effectiveness of the proposed approach was demonstrated amid both volumetric and slow DDoS attack vectors, as well as validating its explicit emphasis on end-user QoS and ease of deployment. In particular, it was empirically demonstrated that UDP-based exploits could be constrained to a dynamically allocated bandwidth coinciding with the given UDP protocol being leveraged by the attacker (e.g., NTP, DNS, etc.), at line rate, with no degradation of QoS to legitimate users utilizing any of the other application layer or transport-specific protocols. Additionally, all strains of TCP flooding were either shown to be rendered ineffective (SYN flooding) with no delays to benign requests, or negated entirely (all other volumetric TCP-based attacks). Lastly, the proposed strategy's ability to successfully counter slow DDoS was confirmed; the server never entered a state of DoS and services to authentic clients were reestablished prior to one second of the attack reaching its peak, followed by a full mitigation within two seconds. This is in contrast to the current state-of-the-art's timeout-based alternatives ranging to upwards of 10 seconds. Moreover, only 3.5% of legitimate users endeavoring to connect to the server suffered a generally unnoticeable delay.

For future work, to further build upon the foundations laid herein, we aim to deploy the methodology on real hardware utilizing `P4Runtime` amid an actual network to put the aforesaid findings to a more comprehensive test. To this extent, a broad set of DDoS attack tools will be instrumented to exercise the full capacity of a programmable switch to safeguard networks amid high rates of malicious traffic.

VII. Acknowledgment

This work was supported by the U.S. National Science Foundation, awards 1907821 and 1925484.

References

[1] E. Bou-Harb, E. I. Kaisar, and M. Austin, "On the impact of empirical attack models targeting marine transportation," in *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. IEEE, 2017, pp. 200–205.

[2] S. Torabi, E. Bou-Harb, C. Assi, M. Galluscio, A. Boukhtouta, and M. Debbabi, "Inferring, characterizing, and investigating internet-scale malicious iot device activities: A network telescope perspective," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 562–573.

[3] E. Bou-Harb, "A brief survey of security approaches for cyber-physical systems," in *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2016, pp. 1–5.

[4] G. Kupreev, Badovskaya, *DDoS attacks in Q1 2019*, 2019 (accessed May 20, 2019), https://securelist.com/ddos-report-q1-2019/90792/.

[5] DanielRTeixeira, "R.u.d.y." Dec 2016. [Online]. Available: https://github.com/DanielRTeixeira/R.U.D.Y.

[6] Gkbrk, "Slowloris." [Online]. Available: https://github.com/gkbrk/slowloris/blob/master/slowloris.py

[7] K. Hong, Y. Kim, H. Choi, and J. Park, "Sdn-assisted slow http ddos attack defense method," *IEEE Communications Letters*, vol. 22, no. 4, pp. 688–691, 2017.

[8] M. Kührer, T. Hupperich, C. Rossow, and T. Holz, "Exit from hell? reducing the impact of amplification ddos attacks," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 111–125.

[9] E. Bou-Harb, M. Debbabi, and C. Assi, "Behavioral analytics for inferring large-scale orchestrated probing events," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2014, pp. 506–511.

[10] E. Bou-Harb, W. Lucia, N. Forti, S. Weerakkody, N. Ghani, and B. Sinopoli, "Cyber meets control: A novel federated approach for resilient cps leveraging real cyber threat intelligence," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 198–204, 2017.

[11] J. Crichigno, E. Bou-Harb, and N. Ghani, "A comprehensive tutorial on science dmz," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 2041–2078, 2018.

[12] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 413–424.

[13] T. Lukaseder, K. Stölzle, S. Kleber, B. Erb, and F. Kargl, "An sdn-based approach for defending against reflective ddos attacks," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. IEEE, 2018, pp. 299–302.

[14] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Yau, and J. Wu, "Realtime ddos defense using cots sdn switches via adaptive correlation analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, pp. 1838–1853, 2018.

[15] B. H. Lawal and A. Nuray, "Real-time detection and mitigation of distributed denial of service (ddos) attacks in software defined networking (sdn)," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2018, pp. 1–4.

[16] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic ddos defense," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 817–832.

[17] P. Xiao, Z. Li, H. Qi, W. Qu, and H. Yu, "An efficient ddos detection with bloom filter in sdn," in *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, 2016, pp. 1–6.

[18] R. Harrison, Q. Cai, A. Gupta, and J. Rexford, "Network-wide heavy hitter detection with commodity switches," in *Proceedings of the Symposium on SDN Research*, 2018, pp. 1–7.

[19] 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*, 2017, pp. 164–176.

[20] Â. C. Lapolli, J. A. 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)*. IEEE, 2019, pp. 19–27.

[21] C. Fachkha, E. Bou-Harb, and M. Debbabi, "On the inference and prediction of ddos campaigns," *Wireless Communications and Mobile Computing*, vol. 15, no. 6, pp. 1066–1078, 2015.

[22] E. Bou-Harb, M. Debbabi, and C. Assi, "A systematic approach for detecting and clustering distributed cyber scanning," *Computer Networks*, vol. 57, no. 18, pp. 3826–3839, 2013.

[23] G. Shang, P. Zhe, X. Bin, H. Aiqun, and R. Kui, "Flooddefender: Protecting data and control plane resources under sdn-aimed dos attacks," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

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

[25] M. S. Pour, A. Mangino, K. Friday, M. Rathbun, E. Bou-Harb, F. Iqbal, K. Shaban, and A. Erradi, "Data-driven curation, learning and analysis for inferring evolving iot botnets in the wild," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*. ACM, 2019, p. 6.

[26] Z. Trabelsi, S. Zeidan, and K. Hayawi, "Denial of firewalling attacks (dof): The case study of the emerging blacknurse attack," *IEEE Access*, vol. 7, pp. 61 596–61 609, 2019.