Hocarest: Host Cardinality Estimation On-Switch using Randomized Algorithm and TCAM

Timon Krack and Martina Zitterbart
Institute of Telematics, Karlsruhe Institute of Technology (KIT)
{timon.krack,martina.zitterbart}@kit.edu

***Abstract*—Monitoring the properties of network traffic, such as the number of different hosts actively sending traffic (host cardinality), is used to observe the behavior of a network and detect threats such as Distributed Denial of Service (DDoS) attacks. However, increasing data rates pose a scalability problem due to the packet processing overhead and excessive memory requirements. In this paper, we propose Hocarest, a novel randomized algorithm that efficiently estimates the host cardinality of the ingress traffic on a switch using its Ternary Content Addressable Memory (TCAM). Hocarest writes a constant number of flow rules to the TCAM. The rules probabilistically match IP source addresses. An estimation of the host cardinality is obtained solely from the match statistics tracked by the switch. No traffic needs to be mirrored and processed externally, and the memory requirements are independent of the traffic volume. We analyze the accuracy of the estimations using real-world traffic and a hardware testbed. Our results demonstrate that Hocarest provides accurate and scalable cardinality estimations with less than 5% variance when using 4000 TCAM rules.**

***Index Terms*—Traffic Monitoring, Network Security, Host Cardinality, Flow Cardinality, TCAM, Randomized Algorithms**

I. Introduction

The number of different hosts that send at least one packet ("host cardinality") or different flows ("flow cardinality") within a predefined monitoring interval is a key metric for network monitoring. Tracking them continuously over time enables multiple applications, including the detection of DDoS attacks by observing sudden increases in the host or flow cardinality [1], [2]. However, counting the exact cardinality of hosts suffers from a fundamental scalability problem. State keeping memory for every distinct host previously seen is required in order to determine for each next host if it is a duplicate or new. In high-volume traffic environments, such as ISPs or IXPs, the memory requirements become infeasible.

This paper proposes *Hocarest*, a novel randomized algorithm that accurately estimates the host cardinality by leveraging the properties of TCAM, which is commonly used by switches in high-volume traffic environments. Hocarest can also estimate the flow cardinality with an equivalent strategy, which is discussed at the end of the paper.

An architectural overview of Hocarest is shown in Fig. 1. The rule generator generates and installs a predefined and constant number of flow rules on the switch ①. The flow rules match multiple, random addresses that allow a subsequent cardinality estimation. After a monitoring interval (a few seconds) expires ②, the flow statistics for the flow rules are

ISBN 978-3-903176-72-0

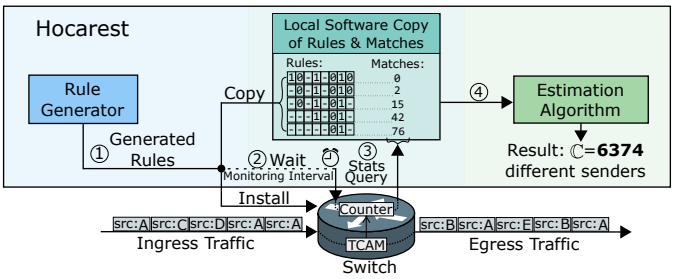

Fig. 1. Architectural overview of Hocarest

queried from the switch ③, including either the number of packets or bytes matched for each rule. The host cardinality is estimated from these statistics only ④, no traffic needs to be transferred or otherwise processed and no per-flow memory allocation is made. The estimation is based on the probabilistic distribution of matched or non-matched rules. This process is repeated for each monitoring interval to obtain a time series of host cardinality estimations, which is the output of Hocarest.

The estimation strategy of Hocarest is related to the HyperLogLog (HLL) [3] cardinality estimation algorithm, as they both estimate the same distribution. However, HLL needs to apply a uniform hash function to each input and count bits in the resulting hash, which is not possible on switches without a programmable data plane (non-programmable switches). Opposed to HLL-based host or flow cardinality estimation approaches [4], [5], Hocarest does not require a programmable switch. It can be used on any switch with a TCAM and the ability to count the number of matched packets per flow rule. These requirements are satisfied by typical SDN-capable switches. All required resources, like the TCAM capacity and the statistics data transferred out of the switch, are independent of the traffic volume, making Hocarest highly scalable.

This paper focuses on presenting the architecture and estimation algorithm of Hocarest, and evaluating the approach using real-world network traffic and real hardware. Our key contributions include:

- Novel approach to host cardinality estimation using only TCAM and match statistics, running on non-programmable switches without external packet processing
- Detailed analysis of the algorithm's properties and optimizations to save TCAM capacity
- Comprehensive performance evaluation of Hocarest in a testbed environment and in different configurations
- Demonstration of using Hocarest to detect a DDoS attack

II. Related Work

Cardinality estimation is a common task in many domains, especially in the context of databases [6]. A widely used algorithm for this purpose is HyperLogLog (HLL) [3], [7]. It works by first evaluating a uniform and collision-resistant hash function for each element. Equal elements produce the same hash value. HLL counts the number of leading zeros in each hashed value. The more different elements get hashed (and thus, more different hashes are produced), the higher the probability that hash values with many leading zeros occur. The cardinality estimation is based on the maximum number of leading zeros observed, thus eliminating the need to allocate a growing amount of memory to track the previously seen elements. Hocarest uses a similar fundamental approach for its estimation, however does not need to compute a hash function over every packet and instead uses TCAM.

Non-programmable switches that do not execute customizable code for each packet cannot implement HLL. However, programmable switches can evaluate a hash function for each packet and count the number of leading zeros. HLL was implemented for programmable switches using the P4 programming language to estimate the host cardinality [4] and flow cardinality [5]. A drawback to using HLL is the computational overhead of repeatedly evaluating the hash function over each packet.

Other projects have focused on obtaining cardinality estimations in network traffic through custom data structures [8], which also rely on a similar bit-counting strategy of hashed outputs like HLL. Compared to the computation of exact solutions, the memory requirements are significantly reduced. Cardinality estimation is also performed on a per-flow basis, using a technique to share registers between multiple flows to reduce the memory requirements [9] or by using a virtual HLL sketch that enables fast queries in online operation [10].

Host and flow cardinality estimation is often used for the detection of DDoS attacks. For instance, the HLL sketch [4] implemented using the P4 programming language is used for detecting DDoS attacks. CARBINE [1] implements an improved version of HLL on a monitoring server that has a smaller error on average than HLL. The cardinality estimations are used to detect DDoS attacks at different attack volumes.

III. Hocarest Cardinality Estimation

This section first explains the basic approach of Hocarest and its TCAM rule structure to obtain a rough cardinality estimation. Then, multiple refinements are presented to reduce the estimation variance and the TCAM space consumption, to correct for estimation bias and to enable the extraction of additional information from the monitored data.

A. Basic Architecture and Operation

Hocarest runs on an external server and controls one or multiple switches on which the cardinality estimations are performed. For instance, Hocarest can be implemented as an SDN app running on the SDN controller. An overview of the approach is illustrated in Fig. 1.

1) Workflow in a Nutshell: Ingress traffic enters the switch, and the goal of Hocarest is to repeatedly estimate the host cardinality $\mathbb{C}$ of these incoming packets over time, for each consecutive monitoring interval. At the beginning of each monitoring interval, the rule generator generates a constant number of rules, which are written to the TCAM of the switch. The rules from the previous monitoring interval are overwritten. Hocarest then waits for the duration of a monitoring interval, during which incoming packets are matched with the rules in the TCAM. The switch counts the number of matches for each rule. After the monitoring interval has expired, the match counters for the TCAM rules are retrieved using a statistics query. Based on these match counts, $\mathbb{C}$ is estimated and appended to a time series of cardinality estimations. The cycle then repeats, and a new estimation is performed for the next monitoring interval.

2) Rule Generation and Characteristics: The rules generated by the rule generator have a specific structure that enables the estimation of $\mathbb{C}$. They match packets by their IP source address. The first rule generated matches a single, uniformly randomly chosen IP source address. Then, for each next rule, one additional bit of the address is randomly chosen to be replaced by the "don't care" (−) state. Let d be the number of "don't care" bits in a rule. As d increases by one with each consecutive rule, the number of IP source addresses that match with the bitmask of the rule doubles to 2^d. When d reaches 32, all the bits are in the "don't care" state and the rule matches all possible IP source addresses.

Rules with larger d receive a lower priority, ensuring that rules matching fewer addresses are always preferred. For any specific IP source address, the probability that the last rule (i.e., the rule with $d = 32$) gets matched is 50%: even though the rule matches all addresses, the superior rule with $d = 31$ has a higher priority and is preferred if the single bit in its bitmask that is not "don't care" matches. One of the more superior rules could also match. With the same argument, the probability that packets from any specific IP source address are matched by superior rules halves with each additional bit that is not "don't care".

If there are many different IP source addresses contained in the ingress traffic (i.e., the host cardinality $\mathbb{C}$ is high), the probability increases that at least one of these source addresses randomly matches with one of the rules with a small d. Similarly, it is very unlikely that a rule with a small d receives a match if the host cardinality is low. To derive an estimation for $\mathbb{C}$, Hocarest finds the rule with the lowest $d = d_{min}$ that has *at least one match*. Hocarest estimates the host cardinality as $\mathbb{C} = 2^{32-d_{min}} \cdot \beta$, where β is a bias correction term that is described later.

3) Illustrative Example: In Fig. 2, rules generated by the rule generator are shown in the TCAM, consisting of IP source address match fields and priorities, and the match counters tracked by the switch. The match counts for this example are based on the beginning of a MAWI traffic capture [11]. The randomly chosen initial IP address constructing the address fields is 193.24.221.44. It forms the rule with the highest

priority (32). For the rule with the second-highest priority (31), the bit index 27 is randomly chosen to be replaced with the "don't care" state, causing it to match with two IP source addresses: 193.24.221.44 and 204.24.221.44. The matching addresses double with each rule, until the last rule with priority 0, which matches all addresses.

The rough estimation of $\mathbb{C}$ is determined by the strictest rule that received at least one match, which in this case, has $d_{min} = 19$ "don't care" bits. Therefore, the host cardinality estimation is $\mathbb{C} = 2^{32-d_{min}} = 2^{32-19} = 8192$. The traffic used to create the example consisted of 6935 different IP source addresses. While the rough estimation is at a similar magnitude, it is still quite far off. The reason for the discrepancy is that the estimation has an extreme variance. Only powers of two can be estimated, and it is possible that occasionally there are outliers which can massively skew the estimation. For instance, even though unlikely, if the actual host cardinality is low, but one IP source address randomly matches one of the rules with few "don't care" bits, the estimation overshoots by a large factor. Therefore, the estimations would only be useful if collected repeatedly over longer periods of time and averaged to compensate the outliers. To obtain useful estimations within short time periods, the variance needs to be significantly reduced.

B. Reducing Estimation Variance

To reduce the variance of the estimation, this section describes changes to the TCAM entries that enable multiple small host cardinality estimations for individual parts of the ingress traffic, instead of having just a single overall estimation. The partial estimations still have the same high variance like before, however, by averaging them, the variance of the overall estimation is reduced significantly. The goal is to split the ingress traffic by IP source addresses into s subsets $S_0, S_1, ...S_{s-1}$, such that packets with the same source IP address are always contained in the same subset. The host cardinality for each subset should thus be (approximately) equal and is estimated individually by a corresponding set of rules (*ruleset*) in the TCAM, which has the same structure as described above. Each subset receives and estimates $\frac{1}{s}$ of the source addresses. The overall host cardinality estimation is $\mathbb{C} = s \cdot \text{avg}(S_0, ..., S_{s-1})$. By averaging the high-variance estimations of the individual subsets, the variance of the overall estimation is reduced.

A natural way of splitting the packets by their source address into s subsets is to use the modulo operator, assigning the packet to the subset with the index equal to the result of the packet's source address modulo s. Since Hocarest aims to work on non-programmable switches, it is not possible to actively make a modulo calculation for each packet. However, if s is equal to a power of two (let $s = 2^b$), the *last b bits* of the source address are already equal to the result of the source address modulo s. This fact can be used to make the TCAM decide the subset assignment of packets automatically and without additional overhead. Fig. 3 shows an example of the TCAM contents for $b = 2$ bits, $s = 2^2 = 4$ subsets. The $s = 4$ different rulesets are

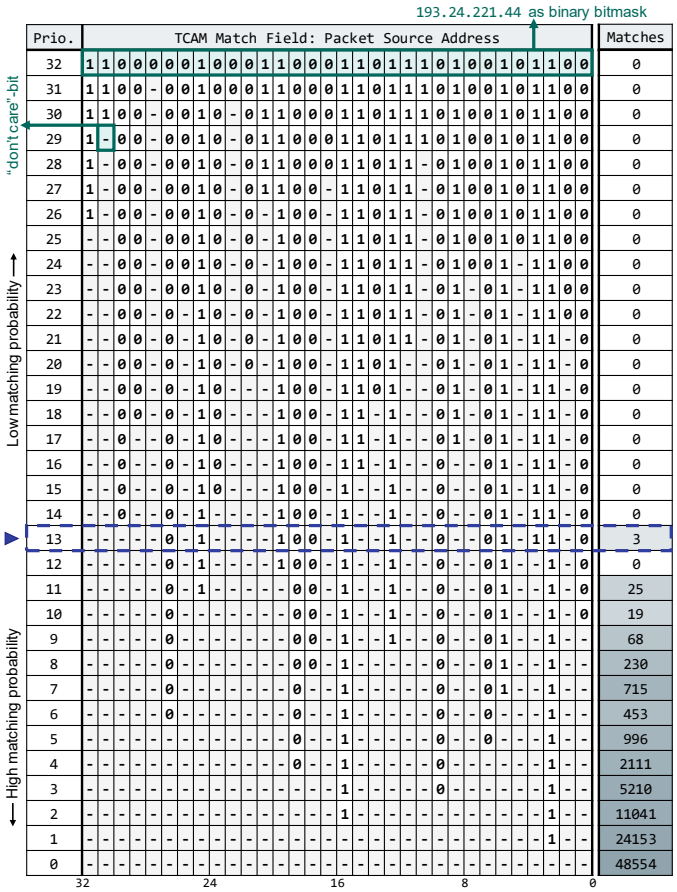

Fig. 2. Example ruleset generated by Hocarest (priority and IP source address match field) and resulting match counts for MAWI traffic [11]

marked, each containing the structure described in Sec. III-A to make the host cardinality estimation, but starting with different random addresses and replacing different random bits with "don't care" in each consecutive rule. The majority of rules in each ruleset are omitted in the figure due to restricted space. In each ruleset, all rules have a fixed value for their last $b = 2$ bits in the packet source address match field, which is equal to the set index. For instance, the first set has `00`, the second set `01`, and so on. When the TCAM matches a packet, only the rules from the ruleset with the index equal to the packet source address modulo s can match. For all other rulesets, the last b bits of the packet source address do not match in at least one position. Within the correct ruleset, the packet matches one of the rules according to their priority. When the monitoring interval has elapsed and the match count statistics are queried from the switch, the host cardinality gets estimated for each of the s rulesets and is averaged. The average is computed as the *harmonic mean*, which is most suitable for averaging the exponentially distributed estimators [3]. Since the TCAM checks all rules in parallel, there is no additional packet processing overhead when using more rulesets, but adding more rulesets results in an increased consumption of TCAM space.

C. Reducing TCAM Space Consumption

When a ruleset is installed in the TCAM and matches packets, most of its rules remain unused with a high probability. For instance, it is exceedingly unlikely that a packet matches one of the top rules where the bitmask has few "don't care" bits. Similarly, unless there is close to no traffic flowing through the switch, the bottom rules of the set with bitmasks that consist almost completely of "don't care" bits probably always have at least one match. To reduce the TCAM space consumption, some of the rules at the top and bottom of the ruleset are not actually written to the switches TCAM, but Hocarest just saves them internally for the subsequent cardinality estimation. Omitted rules from the top of the ruleset are assumed to have 0 matches, and omitted rules from the bottom are assumed to have ∞ matches. To decide which rules are written to the switch, let w be a *window size* and $\mathbb{C}_{prev}$ the cardinality estimation of the previous monitoring interval. The window of rules that get written to TCAM is placed in each ruleset around the rule that has closest to $log_2(\mathbb{C}_{prev})$ non-"don't care" bits ("window center"). This corresponds to the average rule with the least "don't care" bits that still received at least one match. The rules surrounding the window center will be the most relevant to the estimation, while rules further above rarely receive matches and rules further below are almost always matched. Besides the rule at the window center, only the w rules above and below the window center ($2w + 1$ in total) are written to TCAM for each ruleset. Initially, $\mathbb{C}_{prev}$ is initialized with 0, potentially causing the first estimation to undershoot.

D. Bias Correction Term

In the previous sections, bias correction was left out for simplicity, which is caught up in this section. The described cardinality estimation approach within each ruleset has a systematic multiplicative bias, which is caused by the exponential distribution of the estimations within each ruleset. To remedy this, the bias correction term β is used. Since the same type of bias arises with HyperLogLog, the mathematical aspects are already analyzed in detail in the corresponding work, which also derives the compensating bias correction term [7]. With s still being the number of rulesets and d_{min} the number of "don't care" bits in the strictest rule that received at least one match, the cardinality for a ruleset is estimated as

$$\mathbb{C} = 2^{32-d_{min}} \cdot \beta = 2^{32-d_{min}} \cdot \left(s \cdot \int_0^\infty log_2(\frac{2+x}{1+x})^s \, dx \right)^{-1}$$

Since the bias correction term only depends on s, it can be pre-computed for all relevant values of s and looked up in a table as required.

E. Estimation of Cardinality with Minimum Rate

Besides estimating the overall host or flow cardinality, Hocarest can also estimate the number of hosts or flows sending with at least a minimum sending rate of r, measured in packets or bytes per monitoring interval. The resulting estimations are referred to as $\mathbb{C}_{\geq r}$. This information can be extracted with the existing setup and does not require any additional TCAM

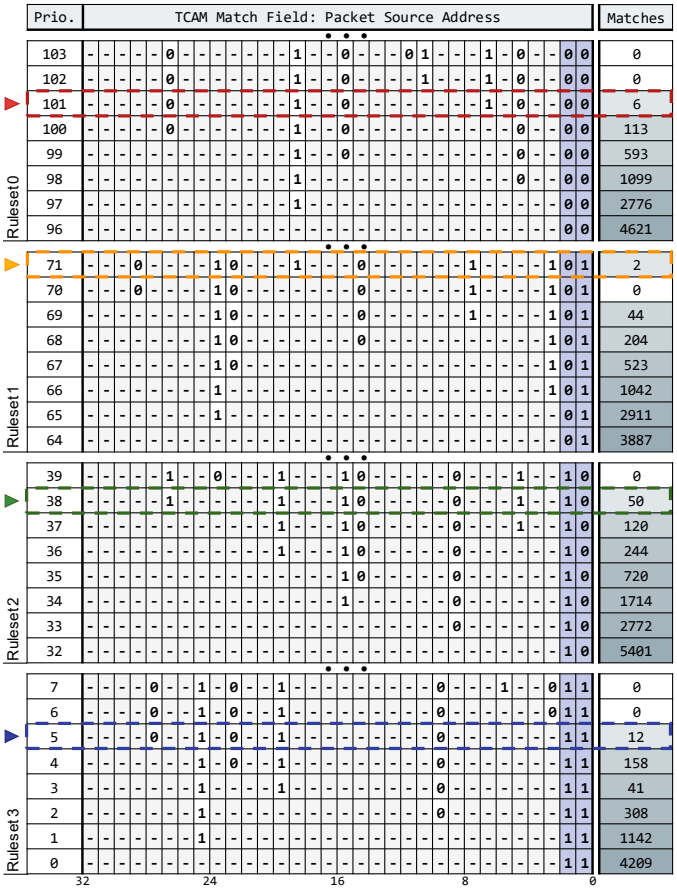

Fig. 3. Example TCAM contents for subset estimations to reduce variance. $b = 2$ bits, $s = 4$ rulesets

space or modifications, and provides useful additional insight for detecting certain attacks or scans by comparing changes in the number of slow/fast sending hosts or flows to the changes in the overall estimation. Also, the number of single-packet transfers (e.g. SYN-Floods, Portscans) can be calculated with $\mathbb{C}_{\geq 1} - \mathbb{C}_{\geq 2}$. The strategy works equivalently for both host and flow cardinality, but this section only refers to host cardinality for simplicity. First, the normal host cardinality estimation is performed to obtain $\mathbb{C}$. The host cardinality estimation is repeated to estimate the number of hosts sending with a minimum rate of r: Given a ruleset, for each rule, the expected number of hosts that contributed to that rule's match count is calculated. If there are s sets overall, each set received $\frac{\mathbb{C}}{s}$ hosts. As discussed in Sec. III-A, the last rule in a ruleset (i.e., the rule with most "don't care" bits) matches 50% of the hosts, halving with each superior rule. For instance, the second-to-last rule is expected to have matched $\frac{\mathbb{C}}{s} \cdot 25\%$ hosts. The match count of each rule is divided with the expected number of matched hosts to obtain the average sending rate of the hosts that matched with the corresponding rule. The estimation algorithm is executed again, but now for each set, bases the estimation on the rule with the minimum number of "don't care" bits that has an average host sending rate of at least r. Cardinality estimations for a minimum sending rate will be demonstrated in the evaluation section.

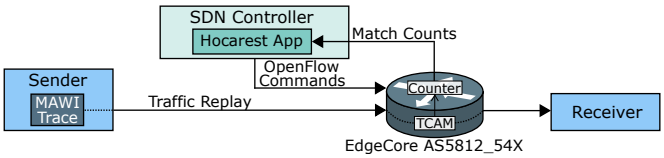

Fig. 4. Overview of Hocarest evaluation setup

IV. Evaluation Methodology and Datasets

Hocarest is evaluated on real hardware and using authentic traffic. For experiments that benefit from multiple and reproducible iterations with arbitrary TCAM space, a simulated environment is used. This section explains the evaluation setup and specifies the datasets used during evaluation.

A. Setup and Methodology

For evaluating Hocarest, an EdgeCore AS5812_54X hardware switch is used. An overview of the evaluation setup is shown in Fig. 4. The switch is connected to two servers. One server sends traffic, replayed from an authentic traffic dataset, to the other server via the switch. For this evaluation, Hocarest is implemented as an SDN app using OpenFlow [14] to control and query the SDN-capable testbed switch. The SDN controller runs the Hocarest app to estimate the host cardinality of the traffic, by generating and installing rule sets on the switch and reading the statistics. As the switch is an older and comparatively weak model, the TCAM capacity limits experiments to ≈ 1000 TCAM entries. This is significantly less than modern switches, which have in the range of tens to hundreds of thousands of TCAM entries. The evaluation will show feasible results even for the limited TCAM availability, and it is expected that Hocarest would perform even better on switches with a larger TCAM size.

Some experiments require more TCAM space than available in the testbed switch. Even though a switch is faster at processing traffic, a simulation on a server can support an arbitrary amount of TCAM rules. Therefore, a simulation of Hocarest is implemented ("Hocarest-Sim") in addition to the SDN app. It directly parses packets from a dataset to simulate and count how many matches each rule would receive. Hocarest-Sim stores a list with the TCAM rules generated by the rule generator. Unlike real TCAM, the rules cannot be checked in parallel, and instead need to be compared consecutively in the order of their priority for each packet, until a matching rule is found. Therefore, the simulation becomes slower for more rulesets, but does not need to run in real time. In the first experiment, the results from the hardware and Hocarest-Sim implementations are compared on the same input to verify that they are similar, and that the simulation can substitute the switch in experiments where it is beneficial.

B. Datasets

All experiments use an authentic traffic dataset provided by the WIDE MAWI archive [11], which captures real-world traffic from a backbone switch. The traffic capture from 10/01/2022 was chosen, as according to the Fukuda Lab [13] analysis it contains a network scan. For verification, all experiments have been repeated on the randomly chosen capture from 11/19/2024 [12] and yielded equivalent results. For an experiment showing the detection of a DDoS attack, traffic is used from the CAIDA dataset [15], which captured a real ICMP flood DDoS attack.

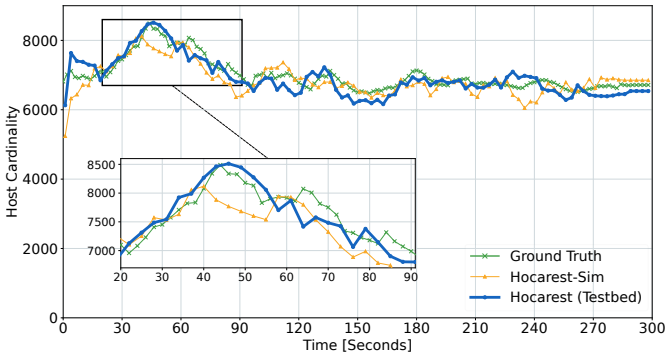

Fig. 5. Host cardinality over time estimated by Hocarest in a simulation and using the testbed switch, compared to the ground truth. MAWI dataset [11].

V. Evaluation Results

This section presents the evaluation results for the accuracy of the cardinality estimations from Hocarest in different configurations, the effectiveness of more rulesets and the sliding window optimization, and the detection of DDoS attacks.

A. Cardinality Estimation Accuracy

This experiment compares the time series of host cardinalities created by Hocarest on the hardware switch and Hocarest-Sim to the ground truth. The latter was determined by parsing the traffic dataset and using a hash table to track the exact number of different source addresses. To approximate the host cardinality, the hardware and simulation both use $s = 64$ rulesets and a sliding window size of $w = 5$, resulting in a TCAM consumption of $(2w + 1) \cdot s = 704$ rules. The MAWI traffic capture [11] is replayed and sent through the switch running Hocarest. The host cardinality is estimated for every monitoring interval of 2 seconds.

The resulting time series are shown in Fig. 5, showing host cardinalities (y-axis) over time (x-axis). The three lines show the ground truth cardinality (green), the estimations from Hocarest-Sim (yellow), and the estimations obtained from the hardware testbed (blue). Both estimations remain close to the ground truth values throughout, and sometimes deviate a few percentage points but catch up a few seconds later. The results from using hardware and the simulation are also very similar. Since different random rulesets are generated each time, it is expected that the deviations of hardware and simulation are not at the same time or in the same direction. It does not appear that the hardware implementation has larger or more deviations compared to Hocarest-Sim. At the beginning, the hardware and simulation estimations both undershoot the true host cardinality value. The reason for this effect is that the sliding window of rules which get installed is not correctly centered at the

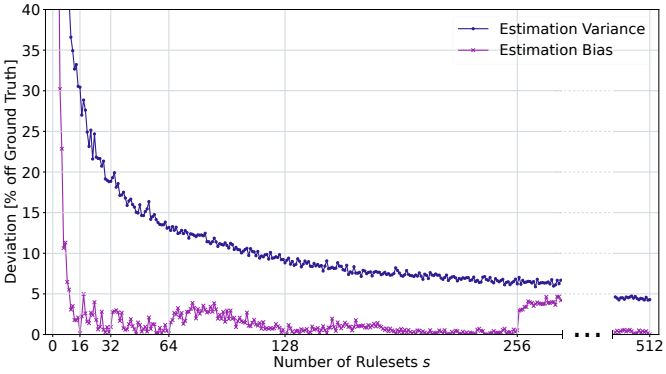

Fig. 6. Estimation variance and bias (lower is better) for different amounts of rulesets. 500 estimation iterations per datapoint. MAWI dataset [11].

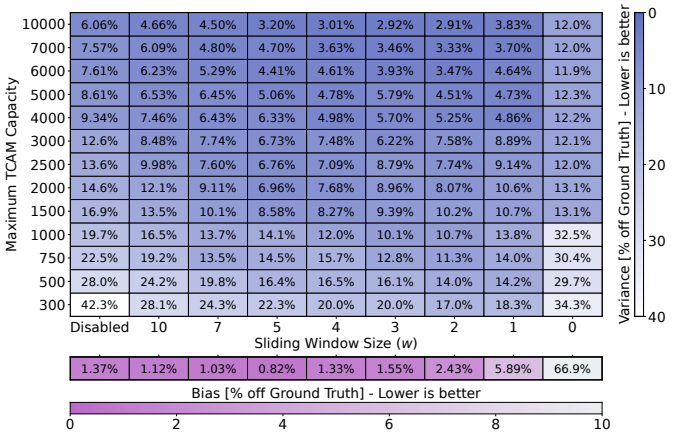

Fig. 7. Estimation variance for different sliding window sizes and TCAM capacities. 500 estimation iterations per value field. MAWI dataset [11].

beginning, because there is no previous estimation available based on which the center of the window can be determined. If the window starts too low, potential matches of rules with few "don't care" bits are lost because they are above the window, causing the estimation to be too small. The problem only occurs during the first few estimations until the window is adjusted accordingly.

Findings: The produced estimations are within a reasonable margin even with less than 1000 TCAM rules. Hocarest-Sim obtains similar results as the testbed setup. With more TCAM space, it is expected that the variance of the estimations further decreases as more rulesets can be used. This cannot be evaluated with the hardware switch due to its limited TCAM capacity, but is shown using Hocarest-Sim.

B. Rulesets Reducing Variance

In this experiment, the deviations of Hocarest's estimations are quantified and examined for different rule set counts s. Since the TCAM capacity of the available testbed switch is exceeded for larger s, this experiment uses Hocarest-Sim, which is not limited in the number of rulesets. The accuracy of the estimated host cardinality $\mathbb{C}$ is assessed using two metrics: Variance and Bias. Given the ground truth time series of host cardinalities $(g_1, ..., g_n)$ and the estimated values $(e_1, ..., e_n)$, the deviations d_i at each time step are calculated as $d_i = g_i - e_i$. The bias is the average of these deviations, and a bias of 0 means that the estimator is unbiased and converges to the exact ground truth value if averaged over enough iterations. The variance in the deviations quantifies how often and by how much individual estimations are expected to be offset.

The bias and variance of the Hocarest algorithm are calculated for each s in $[1, 512]$. The sliding window is disabled to ensure the evaluated estimation accuracy is only depending on s. Therefore, $32 \cdot s$ TCAM rules get utilized. The same MAWI traffic dataset as before is used. The results are shown in Fig. 6. The x-axis shows the number of rulesets used, and the y-axis the variance (blue) and bias (purple), converted to percentages. If less than 16 rulesets are used, the variance is very high ($\geq 30\%$). With more rulesets, the variance decreases exponentially and approaches $\approx 5\%$ for ≥ 256 rulesets. The bias is only high if < 8 rulesets are used, it otherwise mostly remains close to 0%. Since the bias is low, a rolling average over the consecutive estimations converges to the correct value and further reduces the variance compared to the more volatile individual estimations considered in this experiment.

Even though the bias remains mostly low, it increases after each power of two that s surpasses and then decreases again. For exact powers of two, the bias is the lowest. The reason for this effect is that if s is slightly higher than a power of two, the number of bits b used to distribute IP addresses to the sets must be increased, losing the information from the unused sets. For instance, for $s = 257$, $b = 9$ needs to be used, allowing for a theoretical 512 rulesets. Addresses that would fall into sets $258 - 512$ according to their last 9 bits are not matched, and information is lost. The cardinality estimation is extrapolated to correct for the missing rulesets. However, this is not completely accurate unless the distribution of addresses to the sets is perfectly uniform, which is not the case for real-world traffic. To avoid a bias in the estimation, s should not be chosen slightly higher than a power of two. Instead, for $s > 64$, the next lower exact power of two should be chosen, preventing the increase in estimation bias while only slightly increasing the variance, even if this means the allocated TCAM remains partially unutilized. For $s < 64$, the reduction of the variance is more important.

Findings: Based on the results of this experiment, the subsequent experiments will use the following rule: given an available TCAM capacity, if the maximum possible s is larger than 64, s is rounded down to the nearest power of two if it is less than half-way to the next higher power of two. Overall, the experiment showed that more rulesets exponentially decrease the estimation variance to $\approx 6\%$ for 256 rulesets.

C. Trade-off of TCAM-Saving Sliding Window

The optimization of limiting the rules that are actually written to the TCAM to a sliding window that is dynamically adjusted based on the previous estimation was explained in Sec. III-C. Depending on the window size w, only $2w+1$ rules per ruleset

are active, saving TCAM capacity and allowing more rulesets to fit within the same space. This section evaluates the approach by comparing the estimation variance and bias for varying w with different capacity limitations of the TCAM. The number of rulesets s is chosen according to the rule from Sec. V-B, depending on the maximum TCAM space and the number of rules per ruleset that need to be written in TCAM.

The results are shown in Fig. 7 with w on the x-axis and the maximum TCAM capacity on the y-axis. The upper part shows the estimation variance (blue), the lower part the estimation bias (purple). The bias is shown as the average over all TCAM capacities, as it primarily depends on the window size. In the leftmost column, the sliding window strategy is disabled for comparison, i.e., the whole ruleset is written to TCAM. For $w = 0$ in the rightmost column, only a single rule (window center) for each ruleset is written to TCAM. The variance significantly reduces across all maximum TCAM capacities if the sliding window is enabled and decreased to $w \approx 4$, while the bias remains low ($\approx 1\%$) throughout. For smaller w, more rulesets can fit within the same maximum TCAM capacity, which reduces the variance. However, when decreasing w below 3, the estimations begin to get worse. The bias increases, because an increasing number of rules that would have received matches are not written in the TCAM since the window is too small. This missing information skews the estimation.

Findings: Using the sliding window to reduce the TCAM consumption per ruleset and therefore enabling more rulesets to fit within the same space works as intended. The estimation variance can be significantly reduced. With a TCAM size of 5000 rules, enabling the sliding window with $w = 4$ reduces the variance by 44%, from 8.61% to 4.78%. However, w should not be reduced below 4 to prevent an increased estimation bias.

D. Estimation with Minimum Sending Rate

Besides estimating the overall host cardinality, Hocarest can also estimate $\mathbb{C}_{\geq r}$, the number of senders that are sending packets at a rate of at least r packets per monitoring interval. The approach has been explained in Sec. III-E and is evaluated in this section. Since no further TCAM rules or resources are required to obtain these additional estimations, the same testbed configuration as previously is used with $s = 64$ rulesets, a sliding window size of $w = 5$ and MAWI traffic. For this experiment, $\mathbb{C}_{\geq 1}, \mathbb{C}_{\geq 2}, \mathbb{C}_{\geq 5}$, and $\mathbb{C}_{\geq 10}$ were determined, and are shown in Fig. 8. The x-axis shows the time in the capture, and the y-axis the estimated host cardinality.

The cardinalities at different minimum sending rates get estimated at a similar accuracy as the overall cardinality in Sec. V-A. As the host cardinality is less volatile over time for higher minimum sending rates, the estimation even appears to be more accurate. The estimation for $\mathbb{C}_{\geq 1}$ rises significantly between 30 and 90 seconds, while $\mathbb{C}_{\geq 2}$ does not. According to the dataset analysis by Fukuda Lab [13], the dataset used in this experiment contains a network scan, which explains the unexpected increase in just single-packet transfers without an increase in the estimations for higher rates.

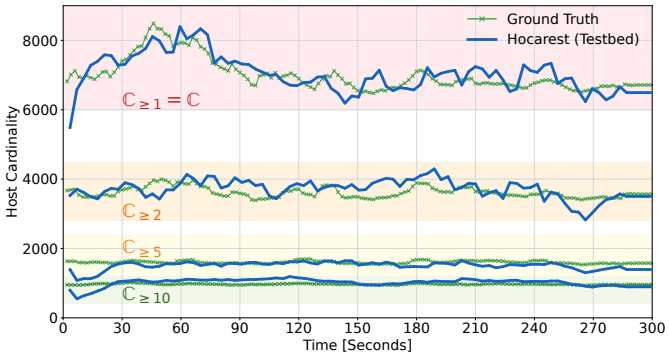

Fig. 8. Host cardinality for minimum rates over time estimated by Hocarest using the testbed switch, compared to the ground truth. MAWI dataset [11].

Findings: Hocarest can estimate the host cardinality at different minimum sending rates without the need for additional TCAM rules. More information about the traffic can be revealed by comparing the cardinalities at different rates, such as $\mathbb{C}_{\geq 1}$ and $\mathbb{C}_{\geq 2}$ for the number of single-packet transfers. For instance, the detection of SYN-flood DDoS attacks greatly benefits from this knowledge, as the number of single-packet transfers drastically increases when the attackers spoof each packet with randomized source IP addresses.

E. Hocarest with DDoS Attack Traffic

An important application of host cardinality estimations is the detection of DDoS attacks. This experiment demonstrates the detection of a DDoS attack on the testbed switch by using Hocarest. The same MAWI traffic [11] as before is used, however this time, after 120 seconds, the CAIDA DDoS attack dataset [15] is also replayed by the sender and sent through the testbed switch for 2 minutes (until 240 seconds). The same configuration of $s = 64$ rulesets and a sliding window size of $w = 5$ is used. The results are shown in Fig. 9, the x-axis is the time, the left y-axis shows the unique senders and the right x-axis shows the overall traffic volume for reference.

Findings: The increase in host cardinality as soon as the attack begins is clearly visible. The overall traffic volume itself does not increase noticeably during the attack, as the background traffic is much stronger than the attack. For automatic detection of the DDoS attack, the first derivative of the host cardinality curve can be computed and increases that are over a preconfigured threshold trigger the detection. In this case, the attack would have been detected within 2 seconds (duration of the monitoring interval). The previous experiments showed a variance of $\approx 5\%$ in the cardinality estimation. False positive detections due to outliers can be reduced by requiring two consecutive estimations to be elevated if the increase is only slightly above the detection threshold, at the trade-off of delaying the detection by an additional monitoring interval. It should be noted that like with other DDoS detection strategies, unexpected traffic events, like flash crowds, may produce similar effects that can cause a misclassification as a DDoS attack. Subsequent (e.g., machine-learning based) detection systems that investigate traffic deemed suspicious on a finer level help to prevent false alarms.

VI. Further Aspects and Extensions

This section discusses additional aspects important for the hardware implementation of Hocarest and for handling IPv6 traffic. Extensions to the approach are described, including the estimation of the flow cardinality instead of the host cardinality, and merging results over multiple switches.

A. Practical Considerations

This section discusses practical challenges from using Hocarest on real hardware and with real-world traffic.

Uniformity of IP-Address Bits: The estimation algorithm assumes that the individual bits of the IP addresses are (at least approximately) uniformly distributed. However, in the real world, this is not the case for the most significant bits of IP addresses. To prevent an impact to the estimation accuracy, when generating a ruleset, the most significant bits receive a higher probability of being replaced with "don't care" first. The first rules of a ruleset practically never get matched (and are not even written to the TCAM when the optimization from Sec. III-C is used), leaving the actual estimation to the rules matching some of the least significant bits of addresses, which are approximately uniformly distributed.

Rule Installation Synchronization: The time difference between submitting the control commands to the switch to write the rulesets to TCAM and the completion of the operation may be large enough to disturb the estimation for short monitoring intervals. Different switch models offer several synchronization primitives. For instance, for OpenFlow, sending an `OFPT_BARRIER` request causes the switch to notify the sender after the current command queue has been cleared. Only then the monitoring interval is started. Additionally, unless negligible due to proximity, the round trip time between the switch and the controlling server needs to be considered.

Non-Atomic Rule Installation: Depending on the specific switch model, when adding multiple TCAM entries at once, the switch may not treat the addition as an atomic operation. Instead, it may steadily add the entries. This could cause some rules or whole rulesets to receive matches before others, skewing the estimation noticeably if a short monitoring interval ($1-2$ seconds) is chosen. Newer OpenFlow versions (≥ 1.4) offer bundle-operations, which ensure atomicity. If the switch does not support atomic operations, the problem can be solved by waiting until the switch has completed writing to the TCAM, and immediately reading the rule statistics. They may already indicate matches for the rules written first. These matches are subtracted from the regular stats query at the end of the monitoring interval before the estimation algorithm is executed.

Handling of IPv6: When Hocarest is applied to IPv6 instead of IPv4, the fundamental approach remains unchanged. However, since IPv6 addresses are much wider (128 bits), the TCAM space optimization strategy of Sec. III-C is required to fit the rulesets within a reasonable portion of the TCAM. Additionally, an automatic analysis of the bit distributions should be performed: Before starting with estimations, 256 rules are added for a few seconds, two for each bit, counting the number of `0` and `1` occurrences. Afterward, the match statistics are read. Only the bits that are closest to a uniform distribution should be used, all others are always "don't care". This ensures that the estimation is not skewed by biased bits that may be significantly more common in IPv6 traffic compared to IPv4.

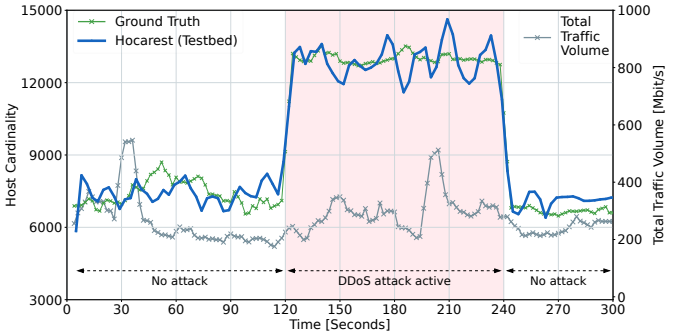

Fig. 9. Host cardinality over time estimated by Hocarest using the testbed switch, with a DDoS attack starting at 2 minutes. MAWI/CAIDA [11], [15].

B. Merging and Global View

Hocarest can run on multiple switches, and results can be merged between the instances to obtain a global cardinality estimation of the total traffic that passed through all participating switches. Additionally, the global estimation is not skewed if the same traffic passes through multiple of the participating switches. The switches need to share a common RNG seed and need to be sufficiently synchronized to generate and write the same rulesets to their TCAM. After the monitoring interval expired, the match statistics are collected at some central server, which is responsible for determining the global estimation. For each ruleset and each rule, it takes the maximum value from the different results of the switches. Intuitively, if a packet p matches the rule with the least "don't care" bits in a ruleset, that rule will be decisive for the cardinality estimation of the ruleset. If p then flows through another switch that also runs Hocarest, the same rule of the same ruleset is matched. The maximum over both matches is still 1, which deduplicated p. If two different packets match the same rule on different switches, the maximum is also 1, which is also correct. If there is a host or flow sending more than one packet during the monitoring interval (via the same switch), the largest count is chosen. While the overall cardinality estimation should always remain accurate, the estimations for a minimum sending rate may diverge if it is not guaranteed that the same host or flow is always transported via the same switch.

To evaluate the merging of estimations using traffic with partially repeated IP addresses, the first and second minute of the MAWI dataset is used simultaneously as two separate traffic streams. Hocarest-Sim uses $s = 64, w = 5$ for both instances. The deviations compared to the ground truth of the merged estimations are similar to the previous experiments. For instance, for the first monitoring interval (2 seconds), the actual host cardinality of the two streams combined is 11915, and the estimation is 11350 (-4.7%).

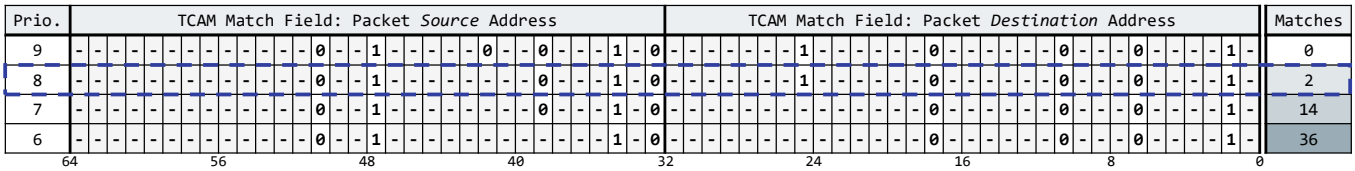

Fig. 10. Example of the rule structure for estimating the flow cardinality

C. Estimation of Flows instead of Senders

The previous sections focused on estimating the host cardinality, i.e., the number of different senders in the packet stream. The same approach can be used to estimate the flow cardinality instead, which may be a more useful metric when using Hocarest to detect network scans. To estimate the number of active flows, the rules in the TCAM not only use the IP source address match field, but also the IP destination address match field. In the case of IPv4 addresses, both have a width of 32 bits, but are treated by Hocarest as a single 64-bit field over which estimations are made. An example is shown in Fig. 10, with the TCAM match fields for source and destination addresses in binary. The estimation rule structure stretches across both match fields, and now estimates the cardinality of different (source, destination) IP address pairs. Optionally, the layer 4 ports could be added as well. By using the TCAM space optimization strategy from Sec. III-C, the required TCAM capacity does not increase when measuring the flow cardinality compared to the host cardinality.

D. Further Use Cases

Besides the detection of DDoS attacks, Hocarest can also be used to detect network scans, especially with the ability to estimate the cardinality for different minimum sending rates. Even though the type and origin of a scan cannot be determined, subsequent systems can be triggered to investigate only the traffic flagged as suspicious. Due to the mergability of results across an arbitrary number of switches, cardinality estimations for a whole network can be obtained for monitoring, statistical and planning purposes without the need to mirror and analyze traffic externally.

VII. Conclusion

This paper presented Hocarest, a host cardinality estimation algorithm that runs on non-programmable switches and leverages the properties of TCAM. By writing rules in the TCAM that probabilistically match addresses, the host cardinality is estimated using the rule's match counters tracked by the switch. A larger allocation of TCAM space makes the estimation more accurate, a variance of less than 5% is achieved with 4000 TCAM rules. Hocarest is evaluated on a testbed switch to obtain host cardinality estimations for real-world backbone traffic. The host cardinality at a minimum sending rate is also determined. The estimations can be used for threat detection, such as DDoS attacks and network scans. The TCAM consumption and amount of data that needs to be transferred out of the switch is independent of the traffic volume.

The main benefit of Hocarest is the ability to obtain these cardinality estimations directly on non-programmable switches, which has not been previously achieved. Future work aims to extend the DDoS detection approach by additionally finding the attack's destination IP address using a refined rule structure.

Acknowledgements: This work was supported by the bwNET2.0 project, funded by the Ministry of Science, Research and the Arts Baden-Württemberg (MWK). This work was supported by the Helmholtz Association (HGF) through the Kastel Security Research Labs (POF structure 46.23.01: Methods for Engineering Secure Systems)

References

[1] D. Ding. "CARBINE: Exploring Additional Properties of HyperLogLog for Secure and Robust Flow Cardinality Estimation". In: IEEE INFOCOM, Vancouver, BC, Canada. May 2024. https://doi.org/10.1109/INFOCOM52122.2024.10621185

[2] Y. Du, H. Huang, YE. Sun, A. Liu, G. Gao, and B. Zhang. "Online High-Cardinality Flow Detection over Big Network Data Stream". DASFAA 2021, Springer. https://doi.org/10.1007/978-3-030-73194-6_28

[3] S. Heule, M. Nunkesser, and A. Hall. "HyperLogLog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm". ACM, 16th EDBT, pp. 683-692, 2013. https://doi.org/10.1145/2452376.2452456

[4] V. Clemens, L. Schulz, M. Gartner and D. Hausheer. "DDoS Detection in P4 Using HyperLogLog and CountMin Sketches". IEEE/IFIP Network Operations and Management Symposium, 2023.

[5] D. Ding, M. Savi, F. Pederzolli and D. Siracusa. "Design and Development of Network Monitoring Strategies in P4-enabled Programmable Switches". IEEE/IFIP NOMS 2022. https://doi.org/10.1109/NOMS54207.2022.9789848

[6] L. Woltmann, C. Hartmann, M. Thiele, D. Habich and W. Lehner. "Cardinality estimation with local deep learning models". In: Processdings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, NY, USA 2019. https://doi.org/10.1145/3329859.3329875

[7] P. Flajolet, E. Fusy, O. Gandouet, and F. Meunier. "HyperLogLog: The analysis of a near-optimal cardinality estimation algorithm". Discrete Mathematics & Theorecial Computer Science, 2012. https://doi.org/10.46298/dmtcs.3545

[8] K. Ishibashi, T. Mori, R. Kawahara, Y. Hirokawa, A. Kobayashi, and K. Yamamoto. "Estimating Top N Hosts in Cardinality Using Small Memory Resources". In: 22nd International Conference on Data Engineering Workshops (ICDWE). https://doi.org/10.1109/ICDEW.2006.56

[9] Q. Xiao, S. Chen, Y. Zhou, M. Chen, J. Luo and T. Li. "Cardinality Estimation for Elephant Flows: A Compact Solution Based on Virtual Register Sharing". In: IEEE/ACM Transactions on Networking, vol. 25, no. 6, pp. 3738-3752, 2017. https://doi.org/10.1109/TNET.2017.2753842

[10] Q. Xiao, Y. Cao and S. Chen. "Accurate and O(1)-Time Query of Per-Flow Cardinality in High-Speed Networks". IEEE/ACM Transactions on Networking 2023, vol. 31. https://doi.org/10.1109/TNET.2023.3268980

[11] WIDE Project. "MAWI Working Group Traffic Archive" 2022/10/01 https://mawi.wide.ad.jp/mawi/samplepoint-F/2022/202210011400.html

[12] WIDE Project. "MAWI Working Group Traffic Archive" 2024/11/19 https://mawi.wide.ad.jp/mawi/samplepoint-F/2024/202411091400.html

[13] R. Fontugne, P. Borgnat, P. Abry and K. Fukuda. "MAWILab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking". In: ACM CoNEXT 2010. Philadelphia, PA. December 2010. https://doi.org/10.1145/1921168.1921179

[14] ONF, OpenFlow Switch Specification, Version 1.5.1. March 2015.

[15] Caida.org The CAIDA UCSD DDoS Attack 2007 Dataset. 2007. https://www.caida.org/catalog/datasets/ddos-20070804_dataset