Design and Development of Network Monitoring Strategies in P4-enabled Programmable Switches

Damu Ding[1], Marco Savi[2], Federico Pederzolli[3], and Domenico Siracusa[3]

[1]University of Oxford, Oxford, United Kingdom
[2]Department of Informatics, Systems and Communication, University of Milano-Bicocca, Milano, Italy
[3]Fondazione Bruno Kessler, Trento, Italy
damu.ding@eng.ox.ac.uk, marco.savi@unimib.it,{fpederzolli, dsiracusa}@fbk.eu

Abstract*—Network monitoring is of paramount importance for effective network management: it allows to constantly observe a network's behavior to ensure it is working as intended, and can trigger both automated and manual remediation procedures in case of failures and anomalies. Software-Defined Networking (SDN) decouples the control plane of network infrastructure from its data plane to perform centralized control on the multiple switches in a network. In this context, the responsibility of switches is only to forward packets according to the instructions provided by a controller. The lack of programmability in the data plane of SDNs prompted the advent of data-plane programmable switches, which allow developers to customize the data-plane pipeline (e.g. match-action tables) by using a domain specific language named P4, and implement novel programs and protocols operating at wire speed directly in the switches. This unlocks the possibility to offload some monitoring tasks to the programmable data plane, and to perform fine-grained monitoring at very high packet processing speeds. Given the central importance of this topic, the principal goal of this thesis is to enable a wide range of monitoring tasks in data-plane programmable switches, with a focus on the ones equipped with programmable Application-Specific Integrated Circuits (ASICs). To achieve this goal, this thesis makes three main contributions: *(i.)* We enhance P4-supported data plane programmability for network monitoring; *(ii.)* We design and develop several network monitoring tasks in programmable data planes; *(iii.)* We combine multiple tasks in a single commodity switch to collect various metrics for different monitoring purposes. Our evaluations show that our solutions can be exploited by network administrators, operators and security engineers to better track and understand the current network status, and thus prevent infrastructure and service failures.

I. Introduction

Network monitoring is of primary importance: it is the main enabler of various network management and security tasks, ranging from accounting [1][2] to traffic engineering [3][4], anomaly detection [5], distributed denial-of-service (DDoS) detection [6], superspreader detection [7], and scans detection [8][9], among others. With the advent of Software-Defined Networking (SDN), the significance of network monitoring has certainly increased. This is because SDN, with its idea of a (logically) centralized control plane, allows an easier coupling of network management with real-time network status observation. As a result, SDN has been seen as the answer to many of the limitations of legacy networks' control and management [10][11][12]. However, such a noble intent has been limited by SDN's current predominant incarnation, the OpenFlow (OF) protocol. Indeed, current OpenFlow APIs are ill-suited for monitoring large network data streams and cannot

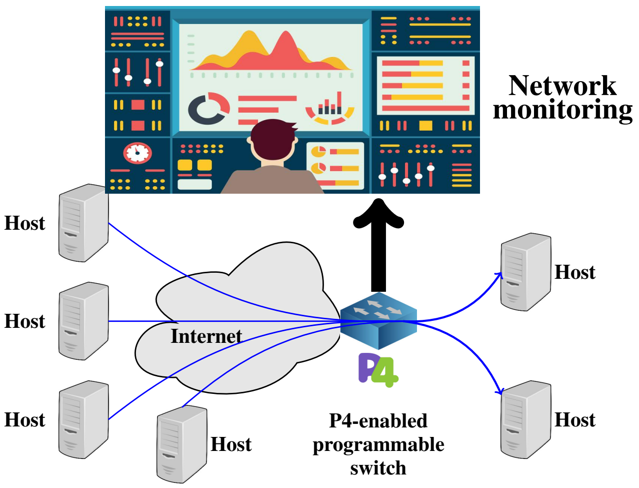

Fig. 1. Deployment scenario example

provide accurate data-plane measurements: the main mechanism exposes simple per-port and per-flow[1] counters naturally available in the switches [13]. An application running on top of the controller can periodically poll each counter using the standard OF APIs and then react accordingly, instantiating appropriate rule changes. As a consequence, OF suffers from two important limitations: *(i)* the controller has to monitor all flows in the network and *(ii)* as the data plane exposes just simple counters, the controller needs to do all the processing to determine the current network state. This also implies that OF-enabled devices are only able to collect raw and fine-grained flow statistics, which have to be sent to a monitoring collector causing significant communication overhead for monitoring purposes. This drawback is well-know for legacy devices as well (e.g. SNMP- and sFlow-supporting equipment) [14]. Another limitation of this approach is the large anomaly detection latency; monitoring intervals are mostly greater than one minute [15]), and this imposes strict time limits on the analysis done by the collector.

Lately, the advent of the so-called data-plane *programmable switches* (e.g. P4-enabled switches [16]) has introduced the possibility to program the data-plane pipeline with advanced functionalities, enabling the possibility to implement more

[1]Flow is a sequence of packets with the same key, such as source IP, destination IP, or any other combination of fields in the packet's headers.

refined monitoring solutions directly in the switch fabric while still guaranteeing line-speed packet processing. Such an innovative technology has attracted a growing number of researchers and practitioners that in turn have proposed many different solutions to enhance SDN capabilities in the context of network monitoring [17][18][19][20][21]. As a result, the prospect of realizing fine-grained network monitoring (see Fig. 1) by analyzing the elaborated and exposed information from the programmable switches in a network, has attracted a lot of interests [19][22][23]. For instance, memory-efficient data structures, such as *sketches* [24][25], have been proven to be implementable in programmable switches to reduce redundant monitoring information.

A. Challenges in the adoption of programmable data planes for network monitoring enhancement

There are many challenges when network monitoring comes to programmable data planes, among which the most important are: *(1)* dealing with limited on-board computational and memory resources in programmable switches; *(2)* designing and implementing monitoring tasks (e.g. for elephant flow detection) executable by programmable switches; *(3)* exploiting the limited hardware resources for a wide range of concurrent monitoring tasks to maximize monitoring visibility. In detail:

Challenge 1: overcoming limited programmability and resources of programmable switches. To assure line-rate packet processing, domain-specific languages such as P4 [16] and POF [26] do not implement many common operations/instructions that, in well-known programming languages (e.g. C/C++), are usually embodied or made available by standard libraries. For instance, (unbounded) loops are not allowed. This is because the switch has a strict and finite time limit for processing a single packet without impacting the queuing time of others. For a similar reason, in a hardware switch, such as an ASIC, the same metadata (i.e., the fields in the packet header) cannot be used more than once. Moreover, the memory of programmable switches is only a few tens of MB large, which means that memory efficiency must be ensured as well. In summary, to enable new functionalities in programmable switches, a good design considering both computational and memory limitations is necessary.

Challenge 2: design and development of monitoring solutions in programmable data planes. A wide range of network monitoring tasks can be executed in the monitoring servers, but not all of them can be offloaded to the programmable switches. To enjoy the benefits of programmable switches, the first step is to understand which tasks are possible and useful to move to the programmable data plane. Afterwards, it is important to investigate which compact data structures (e.g. sketches) or actions (e.g. sample the packets) are suitable for specific monitoring tasks. Finally, the resource limitations of programmable switch described above should not be ignored to make the tasks executable in the data plane.

Challenge 3: combination of multiple monitoring tasks. Due to non-negligible cost of programmable switches (e.g., a Tofino switch [27] may cost thousands of dollars) and increasing throughput requirements of modern networks, if it is made possible to bundle several tasks in a single switch and deploy it in a strategic position in the network, this will significantly reduce the required hardware-upgrade budget. Thus, another challenge is how to combine multiple tasks in a programmable switch to perform high-speed monitoring. In this case, not only the resource limitation in the switch but also the resource allocation for different tasks need to be considered.

B. Goals, research questions, and approaches

The main goal of the thesis is to design and integrate a wide variety of novel and practical monitoring tasks, implement them in P4-enabled programmable switches, and assess their performance. To achieve it, the following research questions (RQs) have been formulated:

- **RQ1:** Why is network monitoring so important in modern telecommunication networks and what are the benefits to implement it in programmable data planes?
- **RQ2:** How to improve P4-enabled data plane programmability with novel functionalities?
- **RQ3:** What are the most important network monitoring tasks that can be offloaded to programmable switches and to what extent?
- **RQ4:** How to coordinate multiple monitoring tasks in a single commodity programmable hardware switch while overcoming existing resource limitations?

To answer these research questions, we started by simulating multiple existing monitoring strategies to understand their behaviour and performance. We then designed some novel strategies, which massively benefit by an offloading of part of their logic to the programmable data plane, and implemented them in P4. Afterwards, the P4 program was compiled and installed in P4-enabled simulated switches. We conducted experiments with simulated switches in an emulated environment (Mininet [28]) to better understand how those strategies impact on the workflow of programmable switches. Finally, by taking into consideration stricter hardware computational and resource constraints, we implemented a subset of our monitoring solutions in a programmable switch equipped with Tofino ASIC, proving that our proposed approaches can be deployed in real P4-programmable networks.

C. Contributions to challenges

The main contribution of this thesis is a detailed study on how to enhance network monitoring by exploiting data plane programmability in the context of Software-Defined Networks. With respect to the three challenges described in Section I-A, we made the following contributions:

Contribution to Challenge 1. We proposed new algorithms to approximate some *arithmetic operations* (i.e., logarithm and exponential function computation) in the programmable switches that are not supported by default in P4, thus enhancing data plane programmability capabilities for network monitoring (or possibly other) purposes.

Contribution to Challenge 2. We studied and developed five different monitoring tasks (partially) executable by programmable data planes: *(i.) heavy-hitter detection* to detect heavy flows with large packet counts; *(ii.) flow cardinality* *estimation* to estimate the number of distinct flows in the network; *(iii.) network traffic entropy estimation* to track the flow distribution; *(iv.) total traffic volume estimation* to know how many packets are flowing in the network; and *(v.) volumetric DDoS attack detection* to detect potential volumetric DDoS attacks by tracking entropy or flow cardinality sudden variations. The aim of this contribution is to offload, as much as possible, these monitoring tasks to the switch's data plane. The best case-scenario happens when the task can be executed entirely in the programmable data plane, thus enabling *in-network monitoring*.

Contribution to Challenge 3. We revisited our designed tasks and proposed a new way to combine them into a single commodity hardware switch. The switch is used only to store flow and packet statistics, while the controller is responsible to compute/estimate various monitoring metrics. In this way, all five monitoring tasks mentioned above can be executed by the same programmable switch to perform high speed monitoring, while the controller can guarantee high accuracy on the estimation of monitoring metrics to diagnose performance and security issues.

II. Thesis overview

The PhD thesis subject of this dissertation paper is available at [35]. Fig. 2 shows a schematic overview of the work. Each block represents a chapter in the thesis, and the scheme report the relationship between chapters as well. For each block, the scheme also records the conference or journal publication reporting the work done. Moreover, the scheme also specifies how the proposed approaches have been tested, including simulations in Python, emulations in Mininet, and experiments in a commodity programmable switch. In the following, we provide a brief summary of each technical chapter of the thesis.

A. Estimation of logarithmic and exponential functions in P4 [29]

In this chapter we take a first step to investigate P4 and report its limitations. We then show how we overcame some of those limitations to estimate logarithms and exponential functions with a given precision by only using the few arithmetic operations available in P4 and without consuming any ternary content-addressable memory (TCAM) resources [36]. This enhancement is leveraged by several monitoring tasks so that they can be implemented in the data plane.

B. Network-wide heavy-hitter detection robust to partial deployment [30][31]

Heavy hitters are often identified as those flows that carry more than a fraction of the overall number of packets in the network. An alternative definition identifies as heavy hitters the *top-k* flows by size (i.e., top-k heavy hitters). Many network management applications can benefit from finding the set of flows that significantly contribute to the traffic carried on a link, e.g. to relieve link congestion [4], to plan network capacity [3], or to detect network anomalies and attacks [37].

In this chapter, we adopt the former definition of heavy hitter. We first use *Count-min Sketch* [25] to filter the flows with relatively large packet counts in a given time interval. At the end of the time interval, the controller retrieves the *local* heavy flows from all programmable switches in the network. Finally, by combining the information from multiple switches, it is able to identify global (also called *network-wide*) heavy hitters. Our network-wide heavy-hitter detection is robust to partial deployments of programmable switches. We thus also propose an incremental deployment strategy that has been designed to ensure that deployed switches have visibility over the largest number of distinct flows. The results show that when only a limited number of programmable switches is deployed, our network-wide heavy-hitter detection strategy outperforms an existing approach in terms of detection accuracy, memory occupation and communication overhead.

C. Flow cardinality estimation [32]

Flow cardinality estimation is the task of determining the number of distinct flows in a stream of packets [38]. In the domain of online traffic monitoring of high-speed networks, cardinality estimation can be used to detect traffic anomalies, such as network IP/port scan and DDoS attacks. Moreover, such an estimation can also be used to monitor the number of active connections on a network link. However, as pointed out in [39], cardinality estimation over large data sets presents a challenge in terms of computational resources and memory: due to this, a non-negligible fraction of packets may not be processed and the information they carry may be lost.

In this chapter, we describe and adopt the compact *LogLog* data structure [40] to estimate the flow cardinality of large packet streams in P4. With respect to the state-of-the-art approaches, our flow cardinality estimator can guarantee good accuracy while ensuring small memory usage in the switch.

D. Network traffic entropy estimation [29][32]

Network traffic entropy is a metric that gives an indication of the traffic flow distribution in the network [29]. According to the definition of *Shannon entropy* [41], the traffic entropy reaches 0 when all packets belong to the same flow, while it reaches its maximum value when each flow carries the same number of packets. Tracking the entropy helps spot performance and security issues, and can be adopted for congestion control [42], load balancing [43], port-scan detection [44][45], DDoS attacks detection [46][47] and worm detection [48].

We designed a time interval-based entropy estimation strategy relying on the estimations proposed in Section II-A. A prototype has been implemented in the P4 behavioral model (BMv2) and has been proven to be fully executable in a P4 emulated environment. The accuracy is comparable to that of an existing state-of-the-art solution [46], but our approach does not require the usage of TCAM, which is a scarce resource.

E. Network-wide total traffic volume estimation [33]

The ability to precisely estimate the total traffic volume [22] (i.e., number of distinct packets flowing in the network), and the related number of distinct flows and average flow size (i.e., average number of packets per flow) is necessary to support a broad range of tasks, especially concerning network-wide monitoring. For instance, as already mentioned, network-wide

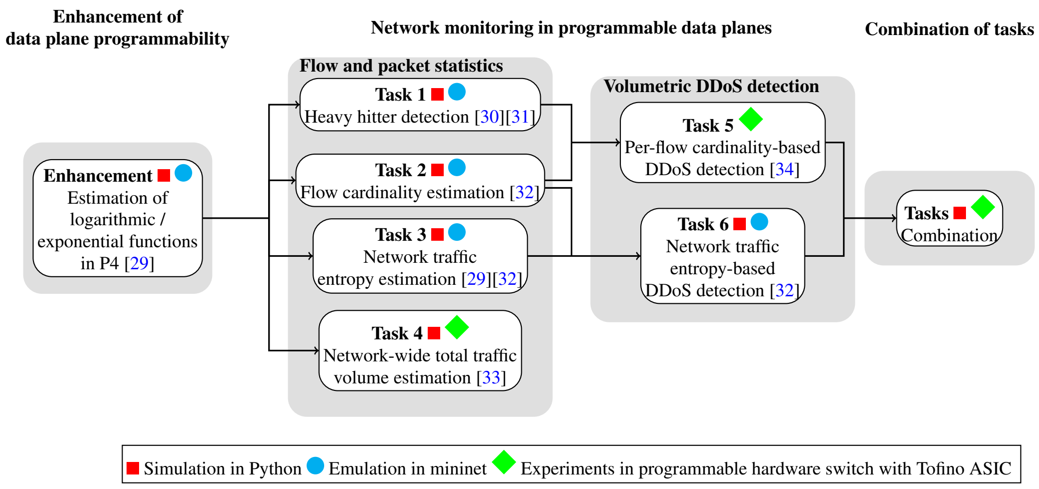
Enhancement of data plane programmability
Network monitoring in programmable data planes
Combination of tasks
Flow and packet statistics
Volumetric DDoS detection
Enhancement 🟥 ●
Estimation of logarithmic / exponential functions in P4 [29]
Task 1 🟥 ●
Heavy hitter detection [30][31]
Task 2 🟥 ●
Flow cardinality estimation [32]
Task 3 🟥 ●
Network traffic entropy estimation [29][32]
Task 4 🟥 ◆
Network-wide total traffic volume estimation [33]
Task 5 ◆
Per-flow cardinality-based DDoS detection [34]
Task 6 🟥 ●
Network traffic entropy-based DDoS detection [32]
Tasks 🟥 ◆
Combination
🟥 Simulation in Python ● Emulation in mininet ◆ Experiments in programmable hardware switch with Tofino ASIC

Fig. 2. Scheme of the thesis

heavy-hitters (see Section II-B) are identified as a function of the total traffic volume.

This chapter presents a novel method that exploits data-plane programmable switches to estimate number of flows, average flow size and traffic volume in a given time interval, and that removes some unrealistic assumptions from the state of the art [22]. In fact, most network-wide monitoring systems assume that each packet is monitored and counted by only a single programmable switch on its path through the network, which either limits routing decision or requires coordination among switches. In networks where packets traverse (and are counted by) multiple switches, the proposed strategies are not accurate anymore: this problem is named *packet double counting*. We solved it by exploiting the peculiarities of a compact data structure called *HyperLogLog* (HLL) [49], which makes it possible to compute network-wide statistics over the *union* of multiple HLL registers. We implemented our strategy in a P4-based commodity switch equipped with Tofino ASIC.

F. Per-destination flow cardinality-based DDoS detection [34]

A volumetric DDoS attack can be identified by looking at many different metrics, such as at a significant decrease of the normalized entropy of distinct destination IP addresses observed in the network [50][51][52], or at an increased number of packets coming from different source IP addresses towards specific destination hosts (i.e., per-destination flow cardinality) [53][17][20]. Note that entropy-based DDoS detection can only detect DDoS attacks, while flow cardinality-based DDoS detection is also able to identify the DDoS victim(s), which allows operators to mitigate the impact on the targeted nodes as soon as an attack is detected.

In this chapter, we first introduce *BACON Sketch*, a memory-efficient data structure to estimate per-destination flow cardinality, which combines a Count-min Sketch with a set of *bitmap* data structures, and theoretically analyze its error bounds. We then propose a simple in-network DDoS victim identification strategy that relies on BACON Sketch to detect the destination hosts for which the number of incoming connections exceeds a pre-defined threshold. We successfully implemented our strategy in a programmable switch equipped with a Tofino ASIC while overcoming the limitations imposed by the hardware.

G. Network traffic entropy-based DDoS detection [32]

The goal of this chapter is similar to that of Section II-F, i.e., efficiently detect volumetric DDoS attacks. We tried to solve this problem from a different perspective, that is, by evaluating variations on the observed normalized network traffic entropy.

We leveraged the achievements and findings on per-destination flow cardinality estimation (Section II-C) and on network traffic entropy estimation (Section II-D) to define an entropy-based volumetric DDoS detection strategy that is able to track the normalized entropy of distinct destination IP addresses. In fact, during a volumetric DDoS attack, the observed normalized entropy of distinct destination IP addresses significantly decreases in comparison to previous time windows. This phenomenon drove the design of our novel approach, which is implementable in P4 and fully executable by emulated programmable data planes. Our work outperforms an existing entropy-based DDoS detection solution [46] in terms of detection accuracy, especially in the case of internal botnet DDoS attacks, while implementing a simpler logic.

H. Combination of multiple tasks in a single commodity switch

This chapter is entirely new and has not been presented and published anywhere else yet. Unlike most of the previous chapters, where experiments have been firstly conducted in the Mininet-based emulated environment, here we combine and implement several monitoring tasks directly in Tofino ASIC. The motivation is demonstrating that our proposals can be practically deployed and exploited together in carrier-grade network devices.

We first revisited the designed tasks described in the previous chapters, and sought the possibility to migrate them all together into a single hardware switch. With this in mind, we discovered that implementing all of them entirely in the switch's data plane is not possible due to both resource and P4 language limitations. We therefore propose an alternative solution, where only summarized flow statistics are tracked in the data plane (thanks to a *Count Sketch* and a *HyperLogLog* data structure), while complex tasks leveraging those statistics are executed by the SDN controller. The collected flow statistics can help the SDN controller understand the overall network status by estimating metrics such as flow variance, flow cardinality, entropy estimation, etc. These metrics can then further be used by the controller to detect anomalies, such as heavy hitters or volumetric DDoS attacks.

III. Revisiting the research questions

The outcomes of this PhD thesis have helped us answer the research questions formulated in Section I-B, even though we are fully aware that there is still a lot of work to do.

Giving an answer to **RQ1** is what has driven our work and its motivation in nearly every chapter of this thesis. Network monitoring is the main enabler of many network management related tasks, ranging from accounting, traffic engineering, anomaly detection, DDoS detection, superspreader detection, scans detection and so on. With the advent of data plane programmability as a way to make Software-Defined Networks more programmable and flexible, some monitoring functionalities can be offloaded to the switch's data plane. This can help diagnose performance and security issues without the need of control plane intervention, thus reducing detection latencies and data/control plane communication overhead, and while processing packets at line rate.

With respect to **RQ2**, we discovered that some mathematical operations, useful for network monitoring purposes, are missing in P4. Among them, logarithm and exponential function estimations, as well as division, are the most important as they can be used to enable a wide set of monitoring tasks to be executed in the data plane. With respect to state-of-the-art solutions, our estimations incur a slightly higher packet processing time but require only P4-supported operations and no TCAM consumption.

When it comes to **RQ3**, we focused on five different monitoring tasks: heavy-hitter detection, flow cardinality estimation, network traffic entropy estimation, total traffic volume estimation, and volumetric DDoS detection. The main objective, also given the answer to RQ1, has been offloading as much as possible these functionalities into the switch's data plane. For some of the tasks we were able to fully offload them, thus enabling in-network monitoring. We first designed a network-wide heavy hitter detection robust to partial deployment of programmable switches in ISP networks. The switches only report flows with large packet counts to the controller for further network-wide investigations. Cardinality estimation, entropy estimation, and entropy-based DDoS detection strategies have been implemented in P4 and can be fully executed in the data plane of an emulated switch. However, we failed to migrate them to the programmable data plane of a commodity hardware switch due to the strict hardware resource constraints. Hence, thanks to the gained experience, we started working on solutions taking hardware limitations into consideration by design. This led to our proposed strategies for traffic volume estimation and flow cardinality-based volumetric DDoS detection, which can be executed by the commodity hardware switch at our disposal.

Finally, focusing on **RQ4**, we realized that implementing many different monitoring tasks in the same hardware switch is not currently feasible. Even though this limitation may be solved by next-generation data plane programmable hardware, we proposed an alternative to be adopted in the short term: only summarized and relevant flow and packet statistics are collected in the data plane and sent to the SDN controller, which is then in charge of executing a wide range of monitoring tasks using such information. Even though this is a workaround, it is a first step to offload part of computation needed by multiple monitoring tasks to the data plane.

IV. Conclusion

The complexity of carrying out high-speed monitoring in today's computer networks hinders a prompt detection of network anomalies and failures. Exploiting programmable data planes for network monitoring purposes has become an appealing solution, as monitoring data can be collected at line rate while ensuring fast packet processing. The contribution of this thesis is thus to deeply investigate the opportunities arising in this area.

We proposed different approaches and evaluated them by also comparing with the state of the art. Our results show that (partially) offloading monitoring operations to the programmable data plane is both feasible and beneficial, especially to reduce the amount of data that needs to be forwarded to a collector. Some contributions of this thesis have been carried on within the GN4-3 project, whose goal is to upgrade the pan-European GÉANT [54] network. We believe that our efforts may be useful to move a step forward towards a better network management of next generation high-speed telecommunication networks.

Acknowledgement

The research leading to these results has received funding from the EC within the H2020 Research and Innovation program, Grant Agreement No. 856726 (GN4-3 project).

References

[1] N. Duffield, C. Lund, and M. Thorup, "Charging from Sampled Network Usage," in *ACM Internet Measurement Workshop (IMW)*, 2001.

[2] C. Estan and G. Varghese, "New Directions in Traffic Measurement and Accounting," in *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2002.

[3] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving Traffic Demands for Operational IP Networks: Methodology and Experience," *IEEE/ACM Transactions On Networking*, vol. 9, no. 3, pp. 265–279, 2001.

[4] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine Grained Traffic Engineering for Data Centers," in *ACM COnference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2011.

[5] A. Lakhina, M. Crovella, and C. Diot, "Mining Anomalies Using Traffic Feature Distributions," in *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, 2005.

[6] C. Wang, T. T. Miu, X. Luo, and J. Wang, "SkyShield: a Sketch-based Defense System Against Application Layer DDoS Attacks," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 559–573, 2018.

[7] Y. Liu, W. Chen, and Y. Guan, "Identifying High-cardinality Hosts from Network-wide Traffic Measurements," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 5, pp. 547–558, 2016.

[8] Y. Xie, V. Sekar, D. A. Maltz, M. K. Reiter, and H. Zhang, "Worm Origin Identification Using Random Moonwalks," in *IEEE Security and Privacy (SP)*, 2005.
[9] S. Venkataraman, D. Song, P. B. Gibbons, and A. Blum, "New Streaming Algorithms for Fast Detection of Superspreaders," in *Network and Distributed System Security Symposium (NDSS)*, 2005.
[10] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a Globally-deployed Software Defined Wan," in *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2013.
[11] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central Control Over Distributed Routing," in *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2015.
[12] A. Gupta, R. MacDavid, R. Birkner, M. Canini, N. Feamster, J. Rexford, and L. Vanbever, "An Industrial-scale Software Defined Internet Exchange Point," in *USENIX Networked Systems Design and Implementation (NSDI)*, 2016.
[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM Computer Communication Review*, vol. 38, no. 2, 2008.
[14] N. L. Van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network Monitoring in OpenFlow Software-defined Networks," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2014.
[15] "NetFlow," http://www.cisco.com/warp/public/732/Tech/netflow.
[16] P. Bosshart, D. Daly, G. Gibb *et al.*, "P4: Programming Protocol-independent Packet Processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.
[17] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman, "One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon," in *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2016.
[18] Y. Li, R. Miao, C. Kim, and M. Yu, "FlowRadar: A Better NetFlow for Data Centers," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2016.
[19] T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig, "Elastic Sketch: Adaptive and Fast Network-wide Measurements," in *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2018.
[20] Q. Huang, X. Jin, P. P. Lee, R. Li, L. Tang, Y.-C. Chen, and G. Zhang, "SketchVisor: Robust Network Measurement for Software Packet Processing," in *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2017.
[21] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, "Heavy-hitter Detection Entirely in the Data Plane," in *ACM Symposium on SDN Research (SOSR)*, 2017.
[22] R. Ben-Basat, G. Einziger, S. L. Feibish, J. Moraney, and D. Raz, "Network-wide Routing-oblivious Heavy Hitters," in *IEEE/ACM Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2018.
[23] R. Harrison, Q. Cai, A. Gupta, and J. Rexford, "Network-wide Heavy Hitter Detection with Commodity Switches," in *ACM Symposium on SDN Research (SOSR)*, 2018.
[24] T. Yang, J. Gong, H. Zhang, L. Zou, L. Shi, and X. Li, "Heavyguardian: Separate and Guard Hot Items in Data Streams," in *ACM International Conference on Knowledge Discovery & Data Mining (SIGKDD)*, 2018.
[25] G. Cormode, "Count-min sketch," in *Springer Encyclopedia of Database Systems*, pp. 511-516, 2009.
[26] S. Li, D. Hu, W. Fang, S. Ma, C. Chen, H. Huang, and Z. Zhu, "Protocol Oblivious Forwarding (POF): Software-defined Networking with Enhanced Programmability," *IEEE Network*, vol. 31, no. 2, pp. 58–66, 2017.
[27] "Intel Tofino," https://www.intel.it/content/www/it/it/products/network-io/programmable-ethernet-switch/tofino-series.html.
[28] "Mininet," http://mininet.org/.
[29] D. Ding, M. Savi, and D. Siracusa, "Estimating Logarithmic and Exponential Functions to Track Network Traffic Entropy in P4," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2020.
[30] D. Ding, M. Savi, G. Antichi, and D. Siracusa, "Incremental Deployment of Programmable Switches for Network-wide Heavy-hitter Detection," in *IEEE Conference on Network Softwarization (NetSoft)*, 2019.
[31] D. Ding, M. Savi, G. Antichi, and D. Siracusa, "An Incrementally-deployable P4-enabled Architecture for Network-wide Heavy-hitter Detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 75–88, 2020.
[32] D. Ding, M. Savi, and D. Siracusa, "Tracking Normalized Network Traffic Entropy to Detect DDoS Attacks in P4," *IEEE Transactions on Dependable and Secure Computing*, 2021.
[33] D. Ding, M. Savi, F. Pederzolli, and D. Siracusa, "INVEST: Flow-based Traffic Volume Estimation in Data-plane Programmable Networks," in *IFIP Networking Conference*, 2021.
[34] D. Ding, M. Savi, F. Pederzolli, M. Campanella, and D. Siracusa, "In-network Volumetric DDoS Victim Identification Using Programmable Commodity Switches," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1191–1202, 2021.
[35] D. Ding, "Design and Development of Network Monitoring Strategies in P4-Enabled Programmable Switches," *PhD Thesis*, 2021. [Online]. Available: https://dingdamu.github.io/papers/PhD_thesis_DAMU.pdf
[36] N. K. Sharma, A. Kaufmann, T. Anderson, A. Krishnamurthy, J. Nelson, and S. Peter, "Evaluating the Power of Flexible Packet Processing for Network Resource Allocation," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017.
[37] A. Lakhina, M. Crovella, and C. Diot, "Characterization of Network-wide Anomalies in Traffic Flows," in *ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2004.
[38] Z. Zhou and B. Hajek, "Per-flow Cardinality Estimation Based On Virtual LogLog Sketching," in *Annual Conference on Information Sciences and Systems (CISS)*, 2019.
[39] S. Heule, M. Nunkesser, and A. Hall, "HyperLogLog in Practice: Algorithmic Engineering of a State of the Art Cardinality Estimation Algorithm," in *International Conference on Extending Database Technology (EDBT)*, 2013.
[40] M. Durand and P. Flajolet, "LogLog Counting of Large Cardinalities," in *European Symposium on Algorithms*, 2003.
[41] C. E. Shannon, "A Mathematical Theory of Communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.
[42] L. Jiang, D. Shah, J. Shin, and J. Walrand, "Distributed Random Access Algorithm: Scheduling and Congestion Control," *IEEE Transactions on Information Theory*, vol. 56, no. 12, pp. 6182–6207, 2010.
[43] K. Wu, L. Chen, S. Ye, and Y. Li, "A Load Balancing Algorithm Based on the Variation Trend of Entropy in Homogeneous Cluster," *International journal of Grid and Distributed Computing*, vol. 7, no. 2, pp. 11–20, 2014.
[44] Y. Gu, A. McCallum, and D. Towsley, "Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation," in *ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2005.
[45] P. Bereziński, M. Szpyrka, B. Jasiul, and M. Mazur, "Network Anomaly Detection Using Parameterized Entropy," in *IFIP International Conference on Computer Information Systems and Industrial Management (CISIM)*, 2015.
[46] A. C. Lapolli, J. A. Marques, and L. P. Gaspary, "Offloading Real-time DDoS Attack Detection to Programmable Data Planes," in *IEEE/IFIP Symposium on Integrated Network and Service Management (IM)*, 2019.
[47] X. Ma and Y. Chen, "DDoS Detection Method Based on Chaos Analysis of Network Traffic Entropy," *IEEE Communications Letters*, vol. 18, no. 1, pp. 114–117, 2013.
[48] A. Wagner and B. Plattner, "Entropy Based Worm and Anomaly Detection in Fast IP Networks," in *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE)*, 2005.
[49] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier, "HyperLogLog: The Analysis of a Near-optimal Cardinality Estimation Algorithm," in *Discrete Mathematics and Theoretical Computer Science*, 2007.
[50] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an Effective and Scalable Anomaly Detection and Mitigation Mechanism on SDN Environments," *Computer Networks*, vol. 62, pp. 122–136, 2014.
[51] K. Kalkan, L. Altay, G. Gür, and F. Alagöz, "JESS: Joint Entropy-based DDoS Defense Scheme in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2358–2372, 2018.
[52] R. Wang, Z. Jia, and L. Ju, "An Entropy-based Distributed DDoS Detection Mechanism in Software-defined Networking," in *IEEE Trustcom/BigDataSE/ISPA*, 2015.
[53] M. Yu, L. Jose, and R. Miao, "Software Defined Traffic Measurement with OpenSketch," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2013.
[54] "GÉANT Network," http://https://www.geant.org/.