
Article:
Liatifis, A., Sarigiannidis, P., Argyriou, V. et al. (2023) Advancing SDN from OpenFlow to P4: a survey. ACM Computing Surveys, 55 (9). pp. 1-37. ISSN: 0360-0300

https://doi.org/10.1145/3556973

Advancing SDN from OpenFlow to P4: A Survey

ATHANASIOS LIATIFIS and PANAGIOTIS SARIGIANNIDIS, University of Western Macedonia, Greece
VASILEIOS ARGYRIOU, Kingston University, United Kingdom
THOMAS LAGKAS, International Hellenic University, Greece

Software-defined Networking (SDN) marked the beginning of a new era in the field of networking by decoupling the control and forwarding processes through the OpenFlow protocol. The Next Generation SDN is defined by Open Interfaces and full programmability of the data plane. P4 is a domain-specific language that fulfills these requirements and has known wide adoption over recent years from Academia and Industry. This work is an extensive survey of the P4 language covering domains of application, a detailed overview of the language, and future directions.

CCS Concepts: • **Networks** → **Programmable networks;**

Additional Key Words and Phrases: SDN, P4, Next Generation SDN, programmable networks

ACM Reference format:
Athanasios Liatifis, Panagiotis Sarigiannidis, Vasileios Argyriou, and Thomas Lagkas. 2023. Advancing SDN from OpenFlow to P4: A Survey. *ACM Comput. Surv.* 55, 9, Article 186 (January 2023), 37 pages.
https://doi.org/10.1145/3556973

1 INTRODUCTION

Modern Computer Networks are characterized by a multitude of network devices (routers, firewalls, load-balancers, proxies, etc.) provided by numerous networking vendors. These devices operate on proprietary software, offer limited configure abilities and require certain level of expertise. On top of that it is difficult to enrich these devices with new features due to hardware limitations and throughput constraints.

The advent of **Software-defined Networking (SDN)** paradigm has set the stage for a new era in the networking industry and academia. The vision of SDN is to offer a fully programmable software-driven network supporting custom and in-house solutions. Decoupling the control plane (i.e., the control mechanisms of the network) from the data plane (i.e., the forwarding mechanisms of the network) SDN has simplified and enabled a multitude of new possibilities. The SDN Controller is a logically centralized entity that is tasked with orchestrating the entire network following

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 957406 (TERMINET).
Authors' addresses: A. Liatifis and P. Sarigiannidis, University of Western Macedonia, Karamanli & Ligeris, Kozani, Greece, 50100; emails: {aliatifis, psarigiannidis}@uowm.gr; V. Argyriou, Kingston University, London, United Kingdom, KT1 2EE; email: vasileios.argyriou@kingston.ac.uk; T. Lagkas, International Hellenic University, Agios Loukas, Kavala Campus, Greece, 65404; email: tlagkas@cs.ihu.gr.

0360-0300/2023/01-ART186
https://doi.org/10.1145/3556973

Table 1. Contributions of This Work

Survey	Technologies similar to P4 presented	Overview of the P4 language provided	Taxonomy of research fields that use P4 provided	Challenges and limitations P4 faces as a programming language presented	In depth discussion and future directions presented
[32]	No	Partially	No	No	No
[77]	Yes	Yes	Partially	No	Yes
[162]	Yes	No	No	No	No
[168]	Yes	Partially	No	No	No
[59]	No	Yes	Yes	Yes	Partially
[84]	No	Partially	No	No	No
[79]	No	Yes	Partially	Yes	Yes
This work	Yes	Yes	Yes	Yes	Yes

the policies defined by the administrator. On top of the SDN controller, intelligent applications can be developed that instruct the controller on how to operate the network forming an Application plane. The SDN controller acts as an abstraction mechanism between these two planes using well defined and documented **Application Programming Interfaces (APIs)**.

To this day, though, the vision of SDN has yet to be fulfilled. OpenFlow [112], one of the most widely used southbound communication protocols in today's Software-defined network deployments, is far from realizing this vision [7]. P4 [14] is a **Domain-specific Language (DSL)** for **Programmable Data Planes (PDPs)** that has attracted the attention of the academia and industry rapidly. This survey article aims to offer the reader an overview of the P4 language, by presenting the language in-depth, its application in various fields of networking, and the deployment environments it is present in. Finally, it presents potential new areas of deployment and future directions.

1.1 Motivation

Numerous surveys related to SDN have been published over the years focusing on different aspects of the SDN paradigm. Some aim to provide a comprehensive overview [88, 149, 162], others focus more on the OpenFlow protocol [15, 96], one of the most widely deployed protocols found in SDN environments, while some research involves other SDN solutions [12, 52]. All previous efforts pay little attention to P4 despite the fact P4 has known great success in academia and industry. References [32, 77, 84] give and overview of data plane programmability and refer to P4. Moreover, Two survey works so far are dedicated to P4 [59, 79], though they are not complete. Both do not include related technologies and differences over P4 and lack future directions. Vieira et al. [168] present an extensive work focused on eBPF and XDP in a manner similar to our contribution. To summarise our contribution Table 1 presents research related to our work.

1.2 Contribution of This Work

The contribution of this work can be summarised as followed:

- we present various data plane programming solutions,
- we present the P4 programming language, a domain-specific language for expressing data plane behavior,
- we provide a taxonomy of various fields of networking where P4 is actively used,
- we present the challenges P4 as a programming language faces,
- we provide future directions related to new domains of applications and to P4 as a programming language.

1.3 Methodology, Bibliography, and Article Structure

The bibliographic search consisted of three phases: *Phase 1* included the bibliography search using popular search engines for researchers, specifically Google Scholar and Scopus. *Phase 2* consisted of abstract reading and evaluation of each scientific paper. During this process, more than half of our database was eliminated and the output result summed to a total of 320 manuscripts. Finally, *Phase 3* included an exhaustive reading of the literature and its grouping in research fields. The result is a collection of 150 scientific manuscripts that are included in this survey work.

We begin by giving a background on network programmability in Section 2. We present the events that lead to the design of solutions such as P4. We also provide a list of solutions similar to P4 in the same section. The supplementary material presents the P4 programming language, a domain-specific language for programmable data planes. An outline of the language is given in this section. In Section 3 the networking domains where P4 is used is presented. We tried our best to provide the reader with a taxonomy that is meaningful, highlights the benefits P4 offers, and has low repetition of information. Section 4 research related to P4 as a programming language is presented. As a programming language P4 incorporates both the benefits and concerns of a programming language. An analysis of our findings is discussed in Section 6 and future directions are presented. Finally, Section 7 includes some final words regarding P4 language and programmable networks.

2 PROGRAMMABLE NETWORKS

2.1 Evolution of Networks

The first computer networks were simple when compared to modern deployments. Emphasis was given to the ability of each network element to work independently for long time frames due to little inter-connectivity and high failure rates, introducing complex protocols and management techniques. The rapid onset of Cloud technologies lead to highly interconnected networks for fast connectivity, though fundamentally networks remained the same.

The first generation of SDN decoupled the data plane and the control plane, and moved the control plane in a logically centralized entity called the controller. The controller is responsible for the overall management of the underlying network. Additionally, the communication between the data plane and the control plane uses a well-defined API. One such API, and the most associated term to SDN, is the Openflow protocol.

The OpenFlow protocol is a southbound API used for communication between the controller and the data plane devices. It defines a series of messages in which the controller can enforce a desired policy to the network. The main limitation of OpenFlow is the fixed set of header fields supported. Each new version of the protocol should first be approved by the **Open Networking Foundation (ONF)** and then be implemented by hardware manufactures. Although more flexibility is available to the network operator in contrast to traditional networking, it is still limited by a fixed set of features of the OpenFlow protocol. Lately progress regarding OpenFlow has stopped in favor of the P4 project [134].

The next generation SDN introduces several new features and possibilities. First, hardware independence sets the stage for a common abstraction layer. The hardware details do not intervene with the desired functionality of the switch. Additionally, the programmability now follows a top-down approach, where the operator specifies the desired behavior and the network is responsible for implementing it without requiring detailed instructions on how to. Instead of closed and proprietary solutions, **Next Generation SDN (NG-SDN)** embraces open interfaces and whitebox hardware. Figure 1 depicts a visual representation of the differences between networking.

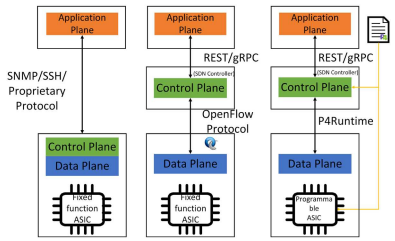

Fig. 1. Evolution of networks: left image depicts a traditional networking element, middle image depicts an OpenFlow-based network element and right image depicts a P4-based networking element.

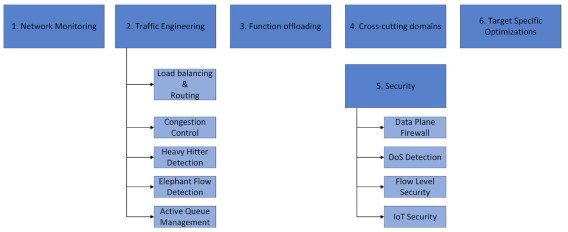

Fig. 2. Domains of application.

Table 2. Data Plane Programming Solutions

	P4	eBPF/XDP	DPDK	POF	ODP	Click	PX
Abstraction level	High	Low	Low	Low	High	High	High
Loops	Not allowed	Allowed	Allowed	Not allowed	Unknown	Allowed	Not Allowed
Pipeline forwarding	Match-Action	N/A	N/A	Match-Action	Match	N/A	Match-Action
Syntax	C-like	C restricted	C restricted	N/A	C restricted	C++	C++

2.2 DataPlane DSLs

In this subsection, we provide a list of Data Plane programming solutions. These solutions offer similar capabilities to P4 but also luck a few. We avoided including derivatives of these solutions, since the main contribution of this survey is the P4 language and not the available Data Plane programming solutions. Table 2 illustrates frameworks and DSLs for data plane programming.

Click [86] was one of the first programmable data plane solutions. It is a modular toolkit for building routers. The user builds the router by connecting several basic components, called elements, in a graph structure. The connections between the components indicate the processing pipeline of a packet.

Protocol Oblivious Forwarding (POF) [157] first appeared in 2013 with the intention of a protocol-agnostic data plane. A POF switch understands no protocol format. Instead, the parsing process of a packet is directed by the controller through <offset, length> keys, and match+action tables. Packet manipulation is possible through bitwise operations and simple mathematical instructions. Furthermore, a POF switch can keep track of flow status through the flow metadata, a data structure that is associated with each network flow where various information can be stored. Additionally, POF allows sharing of common instructions on match+action tables, thus achieving more efficient memory utilization. Counters are also present in POF and offer functionality similar to their OpenFlow counterparts with the distinction that these counters are shared resources that can be allocated to flows by the network developer as they see fit.

Data Plane Development Kit (DPDK) [36] was initially developed by Intel but has since moved under the umbrella of the Linux Foundation. DPDK aims to offer an easy-to-use framework for developing data plane applications for a wide variety of targets including x86_64 **Central Processing Units (CPUs)**, Smart **Network Interface Cards (NICs)**, and **Network Processing Units (NPUs)**. Network administrators can implement their own network protocol stacks without the need for specialized hardware. DPDK applications are essentially C programs that run on user space and reserve all necessary resources in advance.

XDP [65] and eBPF [111] are software frameworks that when combined offer high performance networking and a highly programmable data plane expressed in C language. The **eXpress Data**

Path (XDP) framework provides access to the lowest layer of the Linux networking stack and is executed in the earliest possible stage of the packet processing pipeline. **Extended Berkeley Packet Filter (eBPF)** is small virtual machine structure in the Linux Kernel that allows the execution of C code in a secured and isolated environment. **eXpress Data Path (XDP)** is a module that offers network programmability to the lowest possible layer of the Linux networking stack.

Open Data Plane (ODP) [126] is an open source project for development of data plane applications through a common API in C language. ODP separates the data plane from the hardware heterogeneity through the implementation layer. The implementation layer is target-specific and optimized implementation of these API. The developer develops platform independent applications using these APIs and a compiler maps these high-level programs to the multiple targets.

PX language is a domain-specific DSL designed by Xilinx to program FPGA boards [16]. PX follows an object oriented C++ like style to define the packet processing pipeline. Using PX, the network operator can define packet parsing, header fields update, and deparsing.

3 P4 DOMAINS OF APPLICATION

The simplicity of P4 might arise the question of what can someone do with such a language. Networking devices though implement relatively simple operations on packets, for example, TTL values are decreased, IP address might change due to NAT and **Cyclic Redundancy Check (CRC)** checks are recalculated. P4 is by far capable to fulfill these requirements and also offer additional functionality. The literature in this survey has been categorised in six main categories with many sub-categories as shown in Figure 2. All domains of application make use of all available features of the language.

The parser and the deparser are two important features of the language. If used properly, then one can define custom protocols that meet the requirements and constraints of many environments.

In-band Network Telemetry (INT) [50, 71] is one example of P4 potential to disrupt the networking ecosystem offering new functionalities. Using INT network statistics are appended to an application packet when traversing the network. Before reaching its final destination the INT metadata are stripped from the packet and are delivered to the controller or any other device while the packet remained unchanged from the applications point of view. Sketch-based methods, however, attempt to maintain an estimate of the real state using internal counters and hash-arrays [56].

P4 has made possible another set of applications. Using the offered tools of P4 tasks that were traditionally run in middleboxes or expensive equipment can now be offloaded to the data plane. Certain **Network Functions (NFs)** can be implemented directly in the data plane, whereas a programmable data plane can also assist higher-level applications.

3.1 Network Monitoring

Network Monitoring refers to processes of statistics collections or perform certain actions when an event is triggered. The monitoring process is a crucial part of modern networks. An efficient monitoring solution can minimize communication overhead, leading to less computational tasks for the controller, and help security tools identify anomalies faster and more accurately.

Thanks to P4, the development of high precision monitoring algorithms is made possible. The ability to piggyback metadata and statistic collected from data plane devices to packets is proven quite useful. Moreover, using efficiently the available memory, it is possible to identify abnormalities, monitor overall performance of keep track of paths on a packet level with little performance loss. Table 3 contains a summary of the surveyed work. Niou et al. [123] developed a flexible **Mutlilayer In-band Network Telemetry (ML-INT)** system for IP-Over-Optical networks.

Table 3. Network Monitoring Solutions

Solution	Monitoring Technique	FPGA	Software	ASIC	Simulation	Validation
Niou et al. [123]	INT	No	No	Tofino, NFP-4000 NPU	No	Measure INT header overhead and the effectiveness of the sampling method
Hanf et al. [56]	Sketch	No	No	Tofino	No	Measure memory throughput and compare them against similar solutions
SWAP [54]	–	No	BMv2	No	No	The authors measure the relative error SWAP introduces
Pereira et al. [133]	–	No	BMv2	No	No	Measure latency, throughput, estimations error against similar approaches
TurboFlow [155]	–	No	No	Tofino, NFP-4000	No	Cost estimation of monitoring, overhead introduced on various load scenarios
Elastic Sketch [177]	Sketch	Stratix V	OvS	Tofino	No	Compare to state-of-the-art solution, measure error rate, accuracy and time delays
BitMatrix [110]	Sketch	No	BMv2	No	No	Measure the accuracy of BitMatrix approximation
BurstRadar [74]	–	No	No	Tofino	No	Compare agains INT and Oracle, measure RAM utilization
Gent et al. [47]	INT	No	BMV2	No	No	Compare estimated congestion with actual value
UniROPE [46]	INT-like	No	PISCES	No	No	Evaluate on multiple topologies, compare against CherryPick
Khan et al. [83]	INT	No	No	No	No	–
InOpt [10]	INT	NetFPGA	No	No	No	Measure latency and delays introduced
Choi et al. [26]	INT	No	No	No	No	–
FlowSpy [51]	–	No	BMv2	No	No	Measure the mean deviation and completeness rate for various evaluation metrics
Wang et al. [169]	INT	No	No	Tofino	No	Measure the reduction rate of INT reports in various threshold values
P4-InTel [21]	INT	No	No	BMv2	No	Throughput variation when the number of switches in a path increases
FS-INT [159]	INT	No	No	No	Yes	Compare the INT header in varaying path lengths using FS-INT and alternative methods
TBSW [55]	–	No	BMv2*	No	No	Accuracy loss in each stage and resource consumption
Elastic Trie [94]	–	Xilinx Virtex, Xilinx UltrScale+	No	No	No	Measure accuracy in detecting variations in traffic patterns
SpreadSketch [160]	Sketch	No	No	Tofino	No	Measure preceision, recall, f1 score and relative error against state-of-the-art solutions
FEAL [139]	–	No	BMv2	No	No	Measure precision, recall
*Flow [156]	–	No	No	Tofino	No	Resource utilization and throughput
Laraba et al. [98]	–	No	BMv2	No	No	Fair Bandwidth sharing and switch processing time
Hyun et al. [70]	INT	No	BMv2	No	No	CPU usage, processing time of INT data and bandwidth overhead of INT appended data

* Entries with * are not clearly stated in the text document.

ML-INT selects a small number of packets to insert INT headers. Additionally, to minimize the overhead of the INT data, INT headers of the involved packets contain part of the statistical data. To have meaningful data the disaggregated data need to be collected before being presented. To solve this problem authors developed a parser able to analyze the INT packets and extract the required information. To evaluate the ML-INT system the authors experimented on a real world small scale IP-over-Optical network.

Hanf et al. [56] developed the interleaved sketch, a network telemetry system that is decentralized across all switches. The main idea of their system is the existence of two sketch pipelines that are interchanged between the control and the data plane to avoid delays caused by flushing the data. Using P4, the authors were able to express the decentralized nature of their approach and interchange the available sketches automatically in line-rate. SpiderMon [170] is a network-wide diagnosis system designed for data center networks. The key idea of SpiderMon is to maintain per flow telemetry data on switches for short time periods and upon detection of unusual behavior a debugging mechanism is triggered to identify the root cause of performance degradation. Prototype implementation on BMv2 software switch showed that SpiderMon can quickly identify the problem with minimal overhead.

Sliding Window Algorithm for Packets (SWAP) [54] is a counting mechanism designed for statistics counting using P4 registers and metadata. SWAP operates on every single packet with no performance penalty thanks to P4 requiring few hardware resources. Authors of Reference [166] develop an event detection framework for INT data and an XDP module able to parse the INT data and forward them to Kafka for further analysis. The framework is also able to modify its detection mechanisms, through the SDN controller, by analyzing the INT collected data. Evaluation results show that the proposed framework can easily scale to hundreds of INT monitored flows.

Authors of Reference [133] propose a secure sketch-based monitoring algorithm that runs on programmable switches. Sketch-based techniques rely on aggregated data statistics of packet instead of sampling them. The algorithm can prevent an attacker from performing various attacks that can falsify the collected data. Keys are used during the hashing processes and they are periodically changed. This leads to unpredictable behavior of the algorithm from the hackers point of view.

TurboFlow [155] is a flow record generator tool that can operate on high traffic rates with no need of sampling and produce feature-rich Flow records the same time. To achieve fine-grained measurements, TurboFlow uses microflows that are essentially summaries of flows for a small time windows embedded in table pipeline using P4. These micro flows are later aggregated and forwarded to the control plane. Yang et al. [177] developed Elastic Sketch, a network monitoring method that avoids the introduction of additional load to the network by adapting its behavior in three traffic characteristics, namely, bandwidth, packet rate, and network flow size. Elastic Sketch consists of two parts: (1) the heavy-part where Elephant flows data reside and (2) the light-part where mouse flows data reside. This separation helps to quickly identify elephant flows in the network. Additionally, an eviction mechanism (called ostracism) is introduced. This mechanism will move a flow from the heavy-part to the light-part if certain conditions are met.

Martins et al. [110] propose propose a new skteching structure called BitMAtrix suitable for multi-tenant networks. BitMatrix combines bitmaps and counter-arrays defined and operating in the switch memory using P4. More specifically BitMatrix uses bitmaps ability to efficiently monitor packet arrivals using hashing techniques and also uses counter-arrays ability to count bytes. The end result is able to measure traffic information on networks with multiple tenants. The authors evaluate BitMAtrix on an emulated network using Mininet, BMv2, and an external server that acted as the collector of the data.

Dealing with microbursts BurstRadar [74] is a monitoring tool that operates in the data plane and collects telemetry information of microbursts flows only. BurstRadar consists of three modules. First, the snapshot algorithm designates if a packet belongs to a micro burst and marks it using P4 primitives. Second, for each packet, a courier packet is sent to monitoring servers for further analysis by cloning and inserting it to the pipeline anew. Finally, a ring buffer is used as a temporary storage space for all marked packets.

Gent et al. [47] developed P4-based monitoring and scheduling architecture leveraging INT. The proposed architecture consists of two modules: the monitoring module responsible for collecting information from each switch and the network-management module responsible for traffic forwarding, traffic scheduling, and additional tools provisioning. To efficiently capture information from the switches the monitoring module leverages In-band Network Telemetry. The traffic-management module takes advantage of the information offered by the monitoring module to schedule traffic through the available paths. To evaluate their proposed architecture the authors develop a fat-tree topology using Mininet and BMv2.

Universal and Robust in-band PackEt (UniROPE) [46] is a path tracing approach for SDN-networks that supports arbitrary topologies. UniROPE uses In-band Network Telemetry to efficiently encode information on a packet and uses one of the two path tracing algorithms developed by the authors based on flow characteristics. Results from tests run on Mininet showed that UniROPE is effective in path reconstruction with small overhead and is an ideal solution for debugging and management tasks.

Authors of Reference [83] propose a simple query language to assist network diagnostics using In-band Network Telemetry. Queries are written in a topology-agnostic format and sent to the Query Controller, an entity responsible for communicating with the network. The Query Controller is assisted by the Network Controller to retrieve the actual network topology and sent the appropriate commands to the P4 switches. Once all the data retrieved from the network, a reply returned to the administrator. The primary objective of this approach is to reduce the amount of traffic collected without reducing global accuracy. To achieve this the switches make use of INT framework.

Bahmare et al. [10] developed InOpt, a VNF chain monitoring system that uses heuristics to minimize the overhead introduced by the INT collection process. InOpt determines the locations

of INT sources and sinks to collect the required statistics and constructs a path to collect these statistics.

Authors of Reference [26] propose a verification transverse methodology based on metric dynamic logic using INT framework. The proposed methodology performs real time collection of network statistics, verifies the SLA requirements are met and performs correction actions in case they are not met.

FlowSpy [51] is a monitoring framework that attempts to distribute the computational load of monitoring evenly in the data plane. FlowSpy formulates the assignment problem as an Integer Linear Problem to find the optimal assignment of tasks per switch. Using P4 the control plane can dynamically decide the behaviour of each node without the need of updating the data plane behaviour of individual devices. The evaluation of FlowSpy was done using Mininet and BmV2 reference software switch and the results indicate that FlowSpy is able to load balance the tasks better than traditional approaches.

Wang et al. [169] developed an INT system able to monitor the rules a flow triggers that utilizes the available bandwidth efficiently. More specifically the proposed system assigns an id to each rule and using INT collects these values. Moreover, the authors designed a traffic reduction scheme, since the computational overhead introduced by the method overwhelmed the performance of a typical server.

P4-InTel [21] is a framework for monitoring in-network computational tasks using INT. **Capacity and Available Bandwidth Estimation method (CAPEST)** [76] is a passive method for estimation of available bandwidth based on packet dispersion. To collect all the required data from the data plane CAPEST uses INT.

Authors of Reference [159] propose a flexible sampling-based INT mechanism to overcome large INT headers that introduce additional overhead. **Flexible Sampling INT (FS-INT)** uses two sampling methods, the first is the rate-based method where one in every R packets contains an INT header and the event-based method where each intermediate switches decide when to append INT data or not to each individual packet.

Time-based Sliding Window (TBSW) [55] is a novel measurement algorithm designed to fully exploit the programmable data plane. TBSW measures every packet over a specified time-based sliding window in line-rate speed using . The authors implement TBSw on a Barefoot Tofino ASIC chip to demonstrate its feasibility.

In Reference [94] authors propose a push-based network monitoring method using a new data structure called **Elastic Trie (ET)**. Taking into consideration the hardware limitations authors implemented ET via a P4 program that utilizes hardware registers and built functions of the evaluated target. Using ET the authors can detect hierarchical heavy hitters or superspreaders. To evaluate their proposed monitoring method the authors implement it on two different FPGA boards.

SpreadSketch [160] is an invertible sketch data structure for detecting superspreaders. SpreadSketch has low memory usage and utilizes standard P4 operations and primitives enabling easy deployment on multiple hardware and software targets with a simple backend compilation. **Framework for Efficient Anomaly Localization (FEAL)** [139] is a monitoring framework that use source routing when probes are routed over the network. The framework consists of two modules, the probing cycle module, responsible for placing the minimal number of monitors that cover the network, and the anomaly detection module, responsible for identifying link anomalies using statistical methods.

In References [97, 98] Laraba et al. make use of **Extended Finite-state Machines (EFSMs)** to model stateful monitoring of protocol abuse and enforcing security actions in the data plane. An EFSM is a generalisation of FSM able to use variables and actions and support complex operations without resulting in a large number of states. The proposed system, first, models a protocol into an

EFSM, then translates the EFSM model into P4 primitives and finally, translates the P4 code into target-specific binary. To validate the proposed system the authors model **Explicit Congestion Control (ECN)** and run several tests using BMv2 switch to measure throughput gain, processing time and memory usage.

Hyun et al. [70] proposed a network architecture able to perform closed-loop management. The proposed architecture consists of four planes, namely, (a) the programmable Data plane able to support **In-band Network Telemetry (INT)**, (b) the control plane responsible for deploying the INT functionality to the Data Plane and, (c) the Management Plane where the INT data are collected, transformed, analyzed and finally sent to the control and Knowledge planes, and (d) the Knowledge Plane that utilized ML algorithms onto the INT data and extracts knowledge, which is fed back to the network through the control plane. The authors evaluate their system through a Mininet testbed with BMv2 P4 reference switches and ONOS as the controller of their choice. Performance metrics for their evaluation included CPU usage, the processing time of INT data, and the bandwidth overhead introduced by the INT headers. Their evaluation results reveal that INT header can significantly increase a packet's size and analysis of such data is a computationally intensive task.

3.2 Applications in Traffic Engineering

Traffic Engineering (TE) [6], according to the **Internet Engineering Task Force (IETF)**, is a method of optimizing the overall performance of a computer network through dynamic adjustments. Intelligent algorithms that utilize data collected from network devices classify traffic, make decisions regarding next hops of packets to avoid congestion or quickly reroute traffic to meet **Quality-of-Service (QoS)** requirements.

Offering the ability to express the behaviour of a switch through a common API (i.e., P4), network operators and researchers have introduced several new algorithms and methods. A key aspect of every approach is the efficient utilization of the available resources. It is evident that a good understanding of the features offered by each target is important to achieve high speed networking. Table 4 offers an overview of the surveyed work regarding TE solutions that leverage P4's features.

3.2.1 Load Balancing and Routing. Load-balancing is the process of distributing traffic accross multiple communication paths. The goal of load-balancing is the best possible utilization of network resources in a network. Olteanu et al. [125] developed a stateless load-balancer called Beamer that is able to overcome common limitations of traditional stateful load balancers. Incoming traffic is hashed based on a set of header fields extracted from packet headers using P4 and sent to one bucket. Each bucket is assigned to one server, and one server can have multiple buckets. Additionally, the server has knowledge of the buckets' assignment. To avoid potential connection reset due to server number change, a daisy chain mechanism is implemented. If a packet that is not part of an active connection triggers the daisy chain process which forwards the packet to the proper destination.

Ye et al. [180] developed **weighted ECMP (wECMP)** a variation of ECMP that takes into consideration the path utilization for the selection of the next hop. More specifically each leaf switch stores utilization information of all paths to other leaf switches in a table and uses these information to calculate a weight. wECMP also uses INT to transfer congestion information instead of probes. After evaluating wECMP's performance, the authors conclude that it is an effective load balancing technique.

SHELL [136] is a stateless application-agnostic and load aware load balancer. Using consistent hashing each connection is mapped to an application and the packets are delivered using segment routing. The server agent is responsible for accepting a new flow or redirect it to another application. SHELL can achieve performance similar to other proposed methods. Additionally, the

Table 4. Traffic Engineering Solutions

TE Subdomain	Solution	FPGA	Software	ASIC	Simulation	Evaluation
Routing & Load Balancing	BEAMER [125]	NetFPGA	BMv2	No	No	Twice the performance of Google Maglev
	wECMP [180]	No	BMv2	No	No	Addresses limitations of HULA and CONGA
	SHELL [136]	NetFPGA	No	No	No	Packet Throughput and latency
	Kawaguchi et al. [80]	No	BMv2*	No	No	Throughput when congestion is identified
	GRED [175]	No	BMv2*	No	No	evaluate how network size impacts routing stretch
	DASH [68]	No	BMv2	No	ns-3	Compared against HULA, ECMP
	HULA [78]	No	No	No	ns-3	Addresses limitations of ECMP and CONGA
	MP-HULA [9]	No	No	No	ns-2	Overcomes limitation of HULA
	Paolucci et al. [130]	No	BMv2	No	No	Masure latency and scalability of the system
	Olteanu et al. [125]	No	BMv2	No	No	Measure the amount of dropped connections compare Beamer with state-of-the-art software solution
	Miguel et al. [114]	No	BMv2	No	No	Compare throughput and latency against an ethernet switch
	Bhat et al. [11]	No	BMv2	No	No	Measure dowload and throughput values against baseline scenatio
	Contra [67]	No	BMv2	No	ns-2	Compare against HULA
	QROUTE [165]	No	BMv2	No	No	Tets included large topologies and multiple applications, multiple QoS metrics were measured during the tests.
	Foes et al. [45]	No	BMv2	No	No	Comparison scenarios against MPLS equivalent implementation
	Luiz R. Madureira et al. [109]	No	BMv2	No	No	Compare payload of IoTP against a baseline scenario
Failure Detection	Blink [66]	No	BMv2	Tofino	No	Measure time frame required to restore connectivity
	Wharf [49]	Xilinx ZU9EG	No	Tofino	ns-3	Latency and throughput were measured
	Hirata et al. [64]	No	BMv2*	No	No	Measure throughput of multiple flows when a link fails
	Qu et al. [138]	No	No	No	Tofino	Measure the effectiveness of sQR in concealing the link failure from the end hosts
Congestion Control	P4QCN [48]	No	BMv2*	No	No	Test scenarios against TCP and UDP protocols
	AAW [72]	No	BMv2*	No	No	Compare AAW against HSTCP and Reno, throughput, goodput and retransmission rate
	He et al. [63]	No	No	No	No	Measure mean throughput against various transmission rates
	Turkovic et al. [163]	No	BMv2	Netronome SmartNIC	No	Measure average delay, jitter and packet loss of network flows
	ABC [113]	No	BMv2	No	No	Measure goodput in cases where ABC is present against case where it is not
	Kfoury et al. [82]	No	BMv2	No	No	Measure overall throughput of TCP sessions
	Shahzad et al. [153]	No	BMv2*	No	No	Comparison scenarios against plain TCP congestion mechanism and ECN-based TCP mechanism
	Tokmakov et al. [161]	No	BMv2	No	No	Compare against other scheduling mechanisms
	QoSTCP [22]	No	BMv2	No	No	Comparison tests against ECN and DTCP
Heavy-hitter Detection	Lin et al. [101]	No	No	Tofino	No	Compare performance against HashPipe
	Harrison at al. [58]	No	No	Tofino	No	Measure sensitivity
	Damu et al. [35]	No	BMv2	No	No	Measure detection accuracy, communication overhead and memory consumption
Elephant Flow Detection	IDEAFIX [30]	No	BMv2	No	No	Compare against an Openflow/Sflow wtestbed, accuracy, reaction time, excess data exchange adn monitoring traffic volue were measured
	Hashflow [189]	No	BMv2	No	No	Compare HashFlow against mutliple alternative solution, throughput and network flow coverage
Active Queue Management	P4-CoDel [90]	No	BMv2	No	No	Measure delay and throughput from ingress to egress
	ConQuest [23]	No	No	Tofino	No	Measure the precision and recall of contributing flows to the queue

*Entries with * are not clearly stated in the text document.

proposed hashing method results in few network-flows drops, due to rehashing, even in worst-case scenarios.

Authors of Reference [80] formulate the load balancing of data plane traffic as an **Unsplittable flow Edge Load factor Balancing (UELB)** problem and solve it using linear programming. The proposed solution performs well on small scale networks, though cannot scale to large network topologies due to data plane to control plane communication overhead.

Greedy Routing for Edge Data (GRED) [175] is an efficient data placement and retrieval algorithm for edge computing environments. CRED utilizes the SDN paradigm to load balance and minimize route paths in edge environments. Experimental results show that GRED has superior performance in contrast to similar solutions like Chord.

Data-plane Adaptive Spliting with Hash threshold (DASH) [68] is a partitioning mechanism for Weighted Cost MultiPath load balancing. DASH partitions the hash space in regions and based on the computed hash value a path is chosen. Additionally, these regions are dynamically adjusted to achieve more efficient load balancing with less modifications of tables. Evaluation results on BMv2 switch indicated that DASH performs better compared to other load balancing techniques.

HULA [78] is a distance vector and memory efficient load-balancing algorithm, designed to take advantage of programmable switches. A switch running HULA maintains information about the best next hop only, and special probes are exchanged between switches. An enhancement of HULA, called MP-HULA [9] generalization of HULA that keeps track of k best paths. Paolucci

et al. [130] demonstrated how P4 stateful constructs can be leveraged to perform traffic shaping in optical networks, an important task in Traffic Engineering. Stateful networking refers to the ability of nodes to maintain information regarding the state of a flow (e.g., if TCP handshake is realised). However, stateless networking nodes do not store any information. By defining threshold values the data plane is able to offload traffic to other ports with no intervention of the Controller. Additionally, the authors demonstrated how P4 can be used to implement security functions directly in the data plane.

Miguel et al. [114] propose a **Named Data Networking (NDN)** router for P4 switches. NDN is an alternative networking architecture where a data-centric approach is favored instead of a host-centric. In a NDN network routing is achieved using a named string instead of the traditional IP address. The proposed design uses match-action tables to map names to interfaces. NDN names are split in substring and hashes for each one are calculated and stored in hashtrays, data structures used to store the computed hashes of substrings in an NDN name. The appropriate egress interface is indicated using hashtrays and the longest prefix match. The proposed scheme is implemented in BMv2 software switch.

Bhat et al. [11] propose a hybrid OpenFlow-P4 architecture for traffic routing in SDN-WAN. P4 is used to transform ingress and egress packets of an OpenFlow core network. In Reference [137], a hybrid multipath routing strategy was developed thanks to the programmability offered by the P4 language. Using P4, switches are programmed for fast decision making regarding the next hop, while the control plane can periodically update network global state and make changes that affect switches decisions.

Contra [67] is a system designed for performance-aware routing. Unlike previous solutions Contra can operate on any network topology and adapts to traffic changes. Network administrator express desired policies and Contra compiler undertakes the task to generate switch-local P4 programs. The data plane, as a whole implements a distant vector protocol that forwards traffic based on the current network state and the constraints the administrator provided.

QROUTE [165] is an efficient QoS scheme for SDN overlay networks able to satisfy many constraints. QOSROUTE use a DAG-based routing algorithm and always keeps a secondary DAG for resilience.

Foes et al. [45] propose the **Programmable Labels (ProgLab)**, a novel approach to support traffic differentiation with QoS. ProgLab is based on a minimalist model of the **Residue-defined Networking Architecture (RDNA)**. In RDNA core nodes forward packets based on modulo operations. This results in tableless nodes at the core network and simplifies the routing process in the data plane. The label is embedded in the packet similar to the Multi-Protocol Label Switching approach. Finally, in Reference [109] the authors present the **Internet of Things Protocol (IoTP)**, a Layer 2 communication protocol for IoT-enabled programmable data planes. IoTP is capable of performing aggregations on IoT data and achieve high efficiency of network resources.

3.2.2 Failure Detection. Blink [66] is a failure detection system that operates in the data plane without the intervention of the control plane. Blink leverages the mechanisms of the TCP protocol to quickly identify failures in the data plane. More specifically Blink selects the most suitable network flows for the monitoring process.

Wharf [49] is a Link Layer **Forward Error Correction (FEC)** mechanism able to infer failing links. By continually polling port statistics in the network Wharf can identify the faulty links and when possible perform frame classification on incoming network traffic to the network. To evaluate their proposed mechanism the authors implemented Wharf on a Xilinx Zynq board. The results of their experiments show that Wharf was able to benefit TCP protocol to achieve higher throughput.

Authors of Reference [64] install multiple routing configurations on the data plane to achieve high availability in the data plane. The proposed technique offer fast recovery in case of a link-failure, whereas in Reference [138] the authors developed a Shared Queue Mechanism that caches packets for a specified amount of time and sends them over alternative paths in case of a link failure detection. Since hardware switches often do not provide flexible queuing methods this is a challenging task to accomplish.

3.2.3 Congestion Control. Yan et al. [48] proposed a new protocol based on **Quantizied Congestion Notification (QCN)** protocol called P4QCN. P4QCN is compatible with IP networks and mitigates many problems of the **Priority Flow Control (PFC)** mechanism used in QCN. The authors tested P4QCN using Mininet and BMv2 software switch. Using INT authors in Reference [72] developed an Explicit Congestion Control mechanism named **Adjusting Advertised Windows (AAW)**. AWW achieved 5%–10% higher throughput in contrast to HSTCP and RENO.

Authors of Reference [63] take into consideration the constraints of the hardware switches and implement a traffic rate limiter. The proposed system is implemented using two known methods, namely, token bucket and policing in case of excessive traffic.

Authors of Reference [163] developed a congestion avoidance method suited for programmable data planes using P4. The data plane keeps track of critical flows on a network and applies preconfigured actions in case of congestion. More specifically, each switch is configured with a disjoint predefined set of paths and based on collected statistics it chooses the appropriate path. In case no path is available a special packet is sent to the previous switch to inform about potential congestion. To evaluate their proposed system authors implement it on Mininet network emulator and on a Netronome Agilio CX SmartNIC.

Activity-base Congestion (ABC) [113] management QoS mechanism in P4. According to authors the implementation of ABC in hardware targets is a feasible task but requires many extern objects. In Reference [82], authors propose a scheme that dynamically adjusts TCP pacing rates. The switches store the network state and notify end hosts to update their pacing rate when the network state is altered. To achieve this though the authors inserted a custom header after the IP header. Shahzad et al. [153] propose an enhanced Explicit Congestion Control Mechanism using P4 language. P4 switches store in their registers ECN information and when congestion is detected the ECN-capable header is altered to notify the sender of congestion. The proposed scheme is able to respond faster compared to default ECN mechanism.

Authors of Reference [161] developed a novel traffic management algorithm that combines the Rate-Limited Strict Priority and Deficit round Robin policies and results in a latency aware and quite fair scheduling approach for virtualized environments like Data Centers. QoSTCP [22] is able to slightly adjust the TCP window value to avoid unnecessary drops. To achieve this, two kinds of threshold values are measure for each flow: the **peak threshold rate (PTR)** and the **Marking Threshold Rate (MTR)**. The first one indicate an absolute limit above which any packet will be dropped by the switch, while the alter indicates a warning.

3.2.4 Heavy-hitter Detection. Heavy-hitter traffic is characterised as traffic that consumes a lot of bandwidth for a short period of time often remaining undetected by conventional solutions. Authors of Reference [101] modify the HashPipe algorithm, an algorithm for heavy-hitter detection, to meets the requirements and the constraints of a Tofino hardware switch.

Utilizing P4 registers authors of Reference [58] developed an efficient network-wide algorithm for detection of heavy-hitter flows. A central entity called the coordinator sets threshold values on the switches for a specific set of flows. The switches are configured via P4 to count incoming traffic for these flows and report their values upon reaching or surpassing this threshold value to

the coordinator. To minimize inaccuracies a global threshold is defined and upon reaching it the coordinator explicitly requests current counter values.

Damu et al. [35] propose a new algorithm for heavy-hitter detection in wide area networks, like **Internet Service Provider (ISP)** networks, since the current solutions were not able to achieve high enough accuracy. To evaluate their proposed algorithm the authors run simulations and developed it using P4 and Mininet.

3.2.5 Elephant Flow Detection. Elephant Flows refer to network flows that consume significant portion of bandwidth. In contrast mice flows are flows that are characterised by a small number of packets and consequently bandwidth. Identifying these flow early lead to higher overall networking resources utilisation.

Using P4 registers IDEAFIX [30] can quickly identify elephant flow in the data plane with little intervention by the controller. Instead of sampling traffic in regular intervals and balancing between CPU usage and accuracy, IDEAFIX uses in-memory hash tables in combination with bloom filters classifiers to identify elephant flows. The authors test IDEAFIX with other known sampling techniques and conclude that IDEAFIX is able to outperform them using less memory.

Hashflow [189] is flow collection tool that maintains accurate records for elephant flows and summarized records for mice flows. Additionally, HashFlow applies a novel collision resolution record strategy using hash tables and a promotion strategy for elephant and mice flows. Evaluation results show that HashFlow performance is superior to similar methods.

3.2.6 Active Queue Management. **Active Queue Management (AQM)** is the process of smart and efficient usage of the internal packet queue. AQM can benefit other TE applications, though given the fact that most switches have a black-box AQM module some workarounds have been proposed over the years. Kundel et al. [90] implement **Controlled Delay (CoDel)** AQM algorithm in P4 to address the bufferbloat problem. CoDel reference algorithm as described in RFC 8287 is translated to P4. To evaluate the P4 program the authors develop a Mininet testbed and measure the delay between hosts. The P4 CoDel was able to keep delay below the target value set by the authors. ConQuest [23] is a data structure that estimates flows size in a queue and is able to estimate which flows contribute the most in queue delays. ConQuest is meant to be used as the basis for **Active Queue Measurement (AQM)** schemes according to authors. *Flow [156] is a switch accelerated framework that offers efficient and dynamic measurement of packets. Additionally, *Flow supports concurrent measurements and dynamic queries by lifting selected operations from the data plane and implement them in software.

3.3 Function Offloading

The programmability of the data plane has set the stage for new applications. Certain applications like load-balancers are now easier to offload to the data plane thanks stateful data structures (e.g., registers) offered. While existing solutions like OpenState [12] offer similar capabilities, they inherit all the limitation of OpenFlow. Offering highly programmable data plane also enables offloading of application directly to it. Before the advent of P4, VNFs run on dedicated hardware devices or traditional servers with high overall coast and low throughput. Molero et al. [116] noted that modern hardware solutions in the networking industry are able to perform complex operations without sacrificing throughput and are highly programmable. The aforementioned factors have enabled certain control plane tasks to drift to the data plane introducing benefits and new challenges. They conclude that future network designs are highly possible to be influenced by this shift. Table 5 contains information about the work surveyed in the following subsections. In Reference [39], authors developed a logically centralized yet physically distributed wireless access

Table 5. VNF Offloading

Category	Solution	FPGA	Software	ASIC	Simulation	Evaluation
VNF Offloading	Engelhard et al. [39]	No	BMv2	No	No	Measure throughput of the proposed solution against traditional approaches
	HyMoS [3]	No	No	NFP-4000	No	Measure latency against load and packet size
	DPPx [127]	NetFPGA	No	No	No	Response time distribution of
	Ripple [172]	No	No	Yes*	No	Measure the receiving throughput rate during reconfiguration process
	P4NFV [115]	No	No	Netronome Agilio4000 CX	No	Measure resource consumption and compare against UNO
	P4-BNG [91]	NetFPGA	No	No	No	Measure latency and throughput, compare against traditional approach
	Moro et al. [119]	No	BMv2	No	No	Measure the effectiveness of decomposition during the arrival of multiple requests
	FOP4 [117, 118]	No	BMv2	No	No	Measure initialization delay and consumption delay
	P4SC [24, 187]	No	BMv2	No	No	Efficiency of service chain creation
	INI [107]	No	No	Tofino	No	minimize delay and congestion when processing data

*Entries with * are not clearly stated in the text document.

system that introduces network slicing in Wireless Sensor Networks. Since wireless protocols differ from the wired ones, P4 language is used to bridge this gap. The system is able to run Network Functions and scale accordingly to meet the current requirements.

Hybrid Modular Switch (HyMoS) [3] aims to meet the requirements of the the modern NFV applications by utilizing software and hardware components. More specifically the CPU acts as the scheduler orchestrating the memory. P4-enabled Smart **Network Interface Cards (NICs)** are used to perform lookup operations and perform ingress and egress processing, since they are optimized for this kind of operations. Finally, PCI-e interface is used as the communication bus for its high throughput capacity.

In Reference [127], authors propose the **Data Plane Programmability and Exposure framework (DPPx)**, a P4-based framework that enhances NFV services. DPPx overcomes many limitations of the traditional deployment methods of VNFs and offloads them to the data plane achieving line-rate processing. Ripple [172] is a reconfigurable runtime framework for VNFs that avoids recompilation of P4 programs and thus offers near zero downtime. P4NFV [115] is a framework that offers a unified view of a P4-based data plane to the SDN controller and aims to ease the configuration and management of the data plane. Additionally, P4NFV performs several optimizations using **Mixed Integer Linear Programming Optimization (MILP)** methods.

Authors of Reference [91] analyze the requirements of a BNG environment and design a P4-based data plane for such environments. They further extend their work by implementing the data plane on an FPGA board [92] and achieve better coverage of QoS requirements. Authors of Reference [119] propose a framework that decomposes NFs and offloads them partially to the data plane. The data plane can accommodate different programmable elements both software and hardware that support different protocols and frameworks. An orchestrator receives as input the network topology with its constraints, the NFs alongside with their traffic constrains and uses mixed linear programming to find an optimal decomposition of them to maximize the traffic offloaded to the hardware elements of the network.

Function Prototyping with P4 (FOP4) [117, 118] is an extension of containernet [135]. It is designed for rapid prototyping of heterogeneous scenarios where offloading of task is assigned to the data plane. Zhang et al. developed P4SC [24, 187] a P4-based framework for developing Service Function chains that run directly in the data plane. P4SC offers a set of primitives to ease the process of defining SFCs and all the necessary tools to translate these primitives in P4 code. Additionally, the authors developed a **Longest Common Sequence (LCS)** algorithm to combine multiple SFCs in one P4 program without causing any discrepancies and a management tool for automation and ease of management. The authors evaluated P4SC with six real-word scenarios. Finally, authors of Reference [107] present a new architecture, called **Intra-network Inference (INI)**, that combines SDN and NFV technologies to minimize delay and congestion when processing data.

Table 6. Cross-cutting Domains

Category	Solution	FPGA	Software	ASIC	Simulation	Evaluation
In-network Computations	Sakakibara et al. [144]	NetFPGA	No	No	No	Measure latency, throughput during blockchain operations
	$\lambda - NIC$ [28]	No	No	Netronome Agilio	No	Multiple scnarios where latency was measured
	P4DNS [171]	NetFPGA	No	No	No	Latency and throughput during DNS queries were measaured
	ARGUS [27]	No	No	Netronome Agilio CX	No	Measure perfromance improvement in replication processes
Industrial Control Systems	FastReact [167]	No	BMv2	No	No	Delay for failover detection was measured
	Rüth et al. [143]	No	XDP-based	No	No	Visual comparison of diagrams representing the error margin prove the effectiveness of the system
Internet Of Things	BLESS [164]	No	No	No	No	measure the average round-trip delay against a traditional system
Next Generation Cellular Networks	Ricart-Sanchez et al. [142]	NetFPGA	No	No	No	Measure the effectiveness of isolation during congestion
	TurboEPC [151]	No	No	Netronome Agilio card	No	Measure the throughput scalability when the number of user increases and the performance benefits control plane is introduced to
	Aghdai et al. [2]	No	No	Netronome NFP-4000	No	Measure latency of end-to-end communication
	SMARTHO [128]	No	BMv2	No	No	compare SMARTHO's handover latency against the traditional approach
Traffic Generators	Hyper-Gen [173]	No	No	Tofino	No	Measure the packet generation rate, accuracy and precision
	P4STA [93]	No	BMv2	Tofino, Netronome Agilio	No	Average latency of P4STA platform on various targts was measured
	P4TrafficTool [73]	No	No	No	No	measure the amount of generated plugin code

* Entries with * are not clearly stated in the text document.

3.4 Cross-cutting Domains

P4 has also proven useful in accelerating or assisting other applications. Programmable targets can partially help other services or act as a unifying factor between multiple technologies. Such examples include network-assisted computations, Internet of Things protocols, and traffic generators, to name a few. Table 6 describes the surveyed work.

3.4.1 In-network Computations. In Reference [144], the performance of blockchain-based application was accelerated using the programmable plane P4 offers. More specifically the authors developed a system that support basic blockhain commands in NetFPGA SUME board. The end result is considerably faster and on typical blockchain operations.

lambda-NIC [28] is a framework for serverless computing using smart NICs. Using the Finite-state Machine of the P4 parser *lambda*-NI can identify the lambda actions a packet should pass through and thanks to the high number of processing units it can achieve high processing throughput.

P4DNS [171] is an in-network DNS server that introduces significant performance gains when compared to traditional DNS services. ARGUS [27] is a system that improves the performance of replication protocols by caching clients requests using the processing power of Smart NICs instead of x86 servers. ARGUS caches client requests, thus offering immediate responses by removing the communication delays of servers. The preliminary evaluation of ARGUS shows that it offers higher throughput and achieves lower latency compared to similar solutions.

3.4.2 Industrial Control Systems. FastReact [167] is a tool fashioned for **Industrial Control Networks (ICN)** that offers in-network monitoring, caching and control of actuators. The control plane pushed the logic to the P4 switches via the controller and install the appropriate match/action table entries. Since P4 is a domain-specific language and lacks some features present in traditional programming languages the controller translates the ICN program login into a P4 equivalent program that utilizes registers using **Conjunctive Normal Form (CNF)**. FastReact also passively monitors the traffic exchanged between the actuators and the industrial controllers. Assuming there is continuous communication between them if a certain time frame passes with no reply from one of the sides the switch forward traffic to backup entities ensuring the industrial system is up all the time. Finally, FastReact uses switches as caching memory for sensor data, the sensor value along with a timestamp are stored and are updated frequently. To evaluate FastReact, the authors create two network topologies using the CORE software tool. The first network is a traditional one, while the other is a P4-enabled one that implements FastReact. The results show that

FastReact can significantly reduce response time and offer enhanced resilience. For future work, the authors plan to evaluate FastReact's performance on NetFPGA cards.

Rüth et al. [143] developed an in-**Network Control System (NCS)** that performs control computations directly in the data plane. P4 programs are written and pushed to switches to offload control function in the data plane. Utilizing the data plane any delays and jitter are eliminated. To evaluate their proposed system authors use to emulate a network using Mininet and simulate a pendulum system. The goal for the pendulum is to reach in equilibrium and the results indicate this approach is one worthy of further investigating.

3.4.3 Internet of Things. **Bluetooth Low Energy Service Switch (BLESS)** [164] is a software switch based on PISCES able to forward BLE data packets through IP networks, bypassing the physical limits of the BLE protocol. BLESS has three components that offer the above functionalities. First, the connectivity modules receive and transmit all the Bluetooth traffic between the peers. Second, the packet-switching module responsible for enforcing the forwarding policy of the controller. Finally, the IP connectivity module offers connectivity beyond the physical limits of the BLE. The authors present and evaluate a prototype implementation in PISCES and P4.

3.4.4 Next Generation Cellular Networks. In Reference [142], a novel a slicing framework is developed for 5G MEC environments able to meet the strcit requirements set by the 5G standard. The proposed system use 6-tuple list of header fields to determine the output queue and enforce QoS policies. The implementation of this system was done in NetFPGA SUME FPGA board. TurboEPC [151] is a redesign of the mobile packet core. It offloads tasks to the data plane to reduce communication overheads and achieve higher overall throughput.

Aghdai et al. [2] propose a P4-based **Edge Gateway (EGW)** for MEC environments in Mobile Networks and more specifically in LTE ones. EGW is backward compatible with protocols used in LTE networks thanks to the P4 language. In essence, EGW consists of four modules: (1) the L3-switches module, (2) the Load-balancer module, (3) the Service Offloader module, and (4) the S1AP Processor module. The L3-switches perform L3 routing and choose the appropriate egress port. The Load Balancer modules choose the appropriate application by performing the ECMP load balancing method. The Service Offloader handles S1 messages. Finally, the S1AP Processor is a control plane application that populates tables of the switches. The authors implement EGW as a P4 application using a Netronome NFP4000 card. EGW was incorporated on the LTE network with no major modification required. Also, EGW was able to reduce the core network load by offloading traffic to MEC applications in the edge.

Authors of Reference [128] propose a Smart Handover system for fixed-path mobile devices called SMARTHO. More specifically, SMARTHO operates on a Central Unit level, spoofing user traffic and proactively allocating resources. Authors validate SMARTHO using Mininet emulator and BMv2 software switch. Their results show that the proposed approach has several benefits over traditional.

3.4.5 Traffic Generators. In Reference [173], HyperGen was developed, a paket generator that uses a the P4 ASIC of a switch or a smartNIC to generate traffic in the Tbps scale. P4STA [93] can reduce the overall cost of traffic load generators by offloading the processes of traffic generation and timestamping to the data plane. P4STA is available on many P4 targets. Finally, P4TrafficTool [73] is a tool designed specifically to generate plugin code for many common traffic generators and analyzers. P4TrafficTool uses the intermediate P4 code to identify headers or defined new ones prioritizing builtin structures.

Table 7. Security Solutions

Category	Solution	FPGA	Software	ASIC	Simulation	Evaluation
Data plane Firewall	Paolucci [129]	No	BMv2	No	No	measure overall latency introduced by the proposed solution
	StateFit [69]	No	BMv2	No	No	measure consistancy and latency of updates to the ONOS SDN controller
	CoFilter [18]	No	No	Tofino	Yes	Compare latency of Colfilter against OVS-ConnTrack
	Ricart-Sanchez [140, 141]	NetFPGA	No	No	No	Latency and throughput depending on the number of users
	P4ID [99]	No	BMv2	No	No	measure percentage of traffic not redirected to IDS
	Almaini et al. [4]	No	BMv2	No	No	Throughput and latency
	FrameRTP4 [13]	No	No	No	No	efficiency of hardware resources utlization in rules produced
DoS Detection	TDoSD@DP [40]	No	BMv2*	No	No	CPU utlization was compared against a baseline solution
	Lapolli et al. [95]	No	BMv2	No	No	Entropy estimation accuracy and detection accuracy were measured, comparison was made with a baseline solution
	P4knocking [182]	No	BMv2	No	No	evaluate various deployments and compare them in terms of usability
	Kuka et al. [89]	Xilinx Virtex UltraScale+, Intel Stratix 10	No	No	No	Throughput on 2 FPGA targets was measured
	Dimolianis et al. [34]	No	No	Netronome Agilio CX	No	Accuracy and Forwarding capacity were measured
	Friday et al. [44]	No	No	No	No	Sensitivity, precision and accuracy were amongst the evaluation metrics
	Alsadi et al. [5]	No	BMv2	No	No	measure link delay and flow detection rate of the proposed solution
	POSEIDON [188]	No	No	Tofino	No	measure effectiveness against high volume attacks and ability to adapt in various attack patterns
Flow Security	LANIM [104]	No	No	BMv2	No	System performance under attack was measured and the results were compared against traditional solutions
	SPINE [33]	No	No	BMv2	No	–
	P4NIS [103]	No	No	BMv2	No	Probability of eavesdropping, throughput and network RTT were measrued during the expiriments
	Lin et al. [102]	No	No	Tofino	No	Encoding and Decoding speeds were measured
	P4-MacSec [61]	No	BMv2	NetFPGA SUME	No	Googput, RTT depending on the number of hops
	P4-IPSec [60]	No	No	No	No	–
	Zhu et al. [193]	No	BMv2	No	No	measure encryption/decryption time against traditional solutions
	Yazdinejad [179]	Xilinx Zynq-7000	No	No	No	measure detection accuracy and hardware resources consumption
IoT Security	POISE [120]	No	No	No	No	Overhead introduced by the POISE system
	NETHCF [100]	No	No	Tofino	No	measure resource utilization and latency

*Entries with * are not clearly stated in the text document.

3.5 Security

Securing a computer network is a challenge for organisations of any size. SDN as a management approach enhances security policy enforcement though the unified control plane. Using programming language like P4 can further assist enforcing policies or detect anomalies with zero latency in line-rate. In this section applications of P4 related to security are presented. Table 7 list security solution that utilize a P4 data plane to achieve their goal.

3.5.1 Data Plane Firewall. In Reference [129], P4 was used for Deep Packet Inspection and packet processing in line-rate speeds. In particular the authors propose a P4-enabled node able to perform traffic offloading and operate as a DDoS firewall in packet-over-optical networks. To validate their system authors implemented it both in BMv2 software switch and in an FPGA board. Authors of Reference [121] study popular spoofing attacks and their countermeasure mechanisms. Afterwards, implementations of these countermeasure mechanisms are realised using P4. Evaluation results using Mininet and BMv2 software switch in terms of throughput prove the efficiency of data plane offloading for security solutions.

StateFit [69] is a security framework able to detect and filter out malicious traffic at the data plane. StateFit consists of two parts: (1) the StateFit App, an ONOS application acting as the managing component of StaFit, and (2) the StateFit Interpreter, which is responsible for translating policies originating from (1) and applying them to data plane devices. Evaluation results using Mininet and BMv2 show that StateFit can push rules timely with no disturbances in network's operation.

CoFilter [18] is a hybrid stateful packet filter that uses hash tables to store data. Hash collisions are calculated by a server in the control plane. Doing so, CoFilter can lower the communication load between the Network Elements and control plane. To evaluate their proposed solution two tests were conducted. The first measured the offloading capacity of the proposed system and the second measured the latency between server and switch.

In References [140, 141] authors propose a 5G hardware firewall able to meet the KPIs of 5G networks. The proposed firewall is implemented using the Xilinx NetFPGA board and programmed

using P4. The prototype testbed was empirically tested with generated pcap files that contained malicious traffic. The proposed firewall was able to block traffic before entering the core network and can achieve high throughput. Finally, the authors show that the proposed system can be deployed in real testbeds. P4ID [99] is a P4-based intrusion detection system that consists of rule generator that translates known IDS rule sets in P4. This approach results in high packet processing directly in the data plane.

Akmaini et al. [4] developed two lightweight authentication mechanisms and delegate them to data plane devices. The first method is a port-knocking mechanism based on FSMs while the second one is a one-time password. The controller can dynamically change the port sequence by issuing an update on switch tables and registers values. Evaluation results indicate that the proposed solution has no significant negative impact.

Bonfin et al. [13] developed FrameRTP4, a P4-based framework for real-time detection and mitigation of 5G **Network Slicing (NS)** scenarios offering efficient and scalable wildcard ACL rules and a monitoring system to reduce overhead. Moreover, it delivers an orchestrator to control switches enabling lifecycle management of NSs. Taking into account the complexity of 5G environments, the orchestrator exposes an interface to developers and operators for easier deployment of new features and management. Since data plane devices have limited resources at their disposal and must operate at line rate, FrameRTP4 uses **Bloom Filters (BFs)** to quickly identify attacks and generates the ACL rules dynamically for efficient use of the available memory. The detection rate of BFs and the compression ratio of wildcard rules were measured and analysed during their evaluation. Results indicate that FrameRTP4 is a feasible and extensible solution suitable for 5G environments.

3.5.2 DoS Detection. TDoSD@DP [40] is a tool developed to mitigate Telephony DoS attacks at the edge of the network, thus preventing needless resource utilization. TDoSD@DP enforces limits on each port regarding the maximum number of allowed SIP sessions. Performing **Deep Packet Inspection (DPI)** on SIP protocol headers, the switch extracts related fields and counts the number of active sessions. Each session start message increases the session counter by one and each session termination message decreases the session counter by one. For their evaluation, the authors compared their proposed system against a legacy system. The proposed system protects efficiently by dropping the excess DoS traffic. Additionally, the SIP server shows no CPU spikes, thus it can service other SIP clients without disruptions.

Lapolli et al. [95] developed an in-network DDoS attack detection mechanism. The proposed system is developed using P4 and is able to filter traffic without affecting overall network performance. The evaluation of the proposed mechanism included CAIDA data set traces. In Reference [182], authors investigate the benefits of offloading port knocking mechanism to the data plane and its constraints on various levels of offloading. In particular, the authors present four different offloading variations, two full data plane offloading mechanisms and two hybrid control plane and data plane offloading mechanisms. An important design aspect of the proposed methods is the indexing of IP addresses. Fully offloaded methods offer higher autonomy of data plane devices, since they operate on the data plane, at the costs non optimal resource utilization or multiple collisions. However, hybrid methods offer more flexibility at cost of dependency over the control plane. The authors offers a detailed comparison study of the proposed methods and conclude that offloaded port knocking is effective though data plane constraints need to be taken into consideration.

In Reference [89], the authors design a hardware-based firewall using P4 and Xilinx Virtex UltraScale+ FPGA board able to counter amplification attacks. They compare its throughput with a hand crafted VHDL.

In Reference [34], a **Distributed Denial of Service (DDoS)** detection system is developed that operates in the data plane. More specifically the proposed system inspects network traffic and

computes some metrics, evaluates these metric to detect potential malicious traffic and, last, informs external systems in case a malicious traffic is detected. To evaluate the proposed system, a Netronome SmartNIC is used. Moreover, Friday et al. [44] developed a novel DDoS attack detection and mitigation scheme able to identify attacks of various sizes. The proposed scheme is capable of overcoming known SDN vulnerabilities caused by DDoS attacks.

Alsadi et al. [5] present a security monitoring solution for programmable data plane that consists of three modules. The first module is responsible to perform real time deep packet inspection, the second one is responsible to measure latency using probes, and the third module is responsible to monitor flow behaviour without risking the control plane's stability. Similarly Zhang et al. [188] developed POSEIDON, a DDoS mitigation system that makes efficient use of programmable data plane high throughput capabilities. POSEIDON provides a modular developer-friendly policy expression language similar to NetCore [150]. To make efficient use of the data plane the authors developed an orchestrating component responsible to translate and partition the policy to the most suitable switches and commodity servers using integer linear programming to maximize the overall throughput. To validate their design, the authors implemented POSEIDON in an actual Tofino P4 ASIC switch and DPDK for commodity servers. The extended suite of DDoS attacks POSEIDON undergo proves its efficiency compared to traditional methods in terms of throughput and power consumption.

3.5.3 Flow Security. Liu et al. in Reference [104] propose a new mechanism of eavesdropping attack identification and prevention called LANIM. An attacker has to bypass three defence lines before being able to successfully land an attack. First, LANIM uses minimum risk Machine Learning algorithms to identify abnormal network behavior and take actions. Second, traffic is encrypted using a novel stasteful encryption scheme that involves hosts timestamps in the selection of an encryption policy from a pool of available ones. Lastly, the programmable network devices various policies. P4 was chosen for its ability to store information using the registers data structure. To evaluate LANIM authors rely on network entropy and their results show that LANIM can be deployed on various networks, adapts easily to various security scenarios and enhances the transport security.

Surveillance Protection in the Network Elements (SPINE) [33] is a traffic encryption approach targeted for Large networks and does not require the collaboration of the end hosts, instead only the two end **Autonomous Systems (ASs)** need to collaborate between each-other. SPINE encrypts IP and TCP packet headers before leaving the initial AS and decrypts them in the last AS before being deliverd to the end host. To demonstrate SPINE's effectiveness authors implement it on a P4 PISA hardware switches and the results they concluded to show that SPINE can easily be deployed in real world networks.

P4NIS [103] is a three-layer defence system against eavesdropping attacks. The first layer leverages the security of the existing encryption schemes, the second layer encrypts a subset of header fields using various encryption methods, and the final layer forwards traffic using various forwarding techniques and algorithms making it difficult for the attacker to capture all the traffic of a flow.

Lin et al. [102] developed a permutation algorithm that make us of P4's ability to manipulate packets to enhance the security of 5G and IoT applications. Thanks to line-rate processing offered by the switches the performance penalties are minuscule compared to software-based solutions.

P4-MacSec [61] is an implementation of the IEEE MacSec specification in a P4-based data planes. P4-MacSec architecture consists of a two-tier control plane, one controller that has a global view of the network and switch-local controllers. Additionally, authors propose a secure link discovery mechanism that uses this two-tier controller architecture. The evaluation process was done using BMv2 software switch. Attempts were made to realize P4-MacSec on a NetFPGA SUME FPGA

board too. P4-IPSec [60] is the first attempt to integrate IPSec VPN tunnels and an SDN-enabled data plane. To realize though Ipsec function extern P4 objects are required. Extern objects though often lead to portability issues when deployed on different targets.

Zhu et al. [193] design a dynamic multi-path and multi-protocol communication mechanism that encrypts traffic as they traverse data plane devices. The proposed mechanism consists of four modules. First, the information collection module is responsible the collect and analysing user traffic. Second, the use requirement storage module is responsible for sore encryption requirements of user flows. Third, the encryption and transmission decision module is tasked with choosing the most appropriate encryption schemes using fuzzy logic and constructing the path a network flow will follow. Finally, the mapping module acts as a database storing all necessary information. To evaluate the proposed mechanisms authors implemented them in the ONOS controller and validated in using the BMv2 target switch. The authors conclude that the proposed mechanism is resource-efficient and requires less time than traditional schemes.

3.5.4 IoT Security. **Programmable In-network security (POISE)** [120] is a security solution for custom made devices. Consisting of three modules, namely, the client module, which collects device-specific data and encapsulated them in special packets, the POISE compiler receives a policy configuration program and outputs a P4 program able to enforce these policies and parse the packets correctly, and finally the POISE runtime environment, which acts as the controller of the network. POISE enforces policies in the data plane ensuring a more robust security model but requires client side modifications to function properly.

NETHCF [100] is an in-network filtering system for spoofed traffic that take full advantage of programmable data planes. It overcomes the limitations of traditional **Hop Count Filtering (HCF)** techniques by utilizing the data plane as a cache for line-rate processing of traffic while the control plane is considered as an HCF mirror handling non-heavy-hitter network flows. Moreover, to make use of the limited memory of programmable switches the authors suggest an **IP hop count design (IP2HC)** leveraging binary trees. To evaluate NETHCF a testbed consisting of a Tofino hardware switch and two servers using CAIDA traffic. The results demonstrate NETHCF's ability to process flows in line rate and maintain heavy-hitter statistics in the data plane.

3.6 Target-specific Optimizations

As a high-level DSL P4 can be compiled in multiple target-specific configuration formats. Each target though may differ considerably compared to other targets. Target-specific compilers produce optimised solutions, though for optimal data plane utilization knowledge of target-specific details is needed.

Cabal et al. [17] developed a novel packet parser design for FPGA targets able to achieve high raw throughput by expoiting the enormous processing parallelism offered by FPGA boards. The authors divide the bus in logical areas called regions where one is each able to store a packet. This way more packets can be processed in a single cycle. Additionally, using pipeline stages the underlying hardware is used more efficiently. The P4 code is transformed into a **Parse Graph Representation (PGR)** and topological ordering is applied. Finally, the VHDL code is generated and is pushed to the FPGA board. The design in validated through two FPGA boards scenarios and the results show 1 Tbps of raw throughput and significantly higher throughput in more complex protocol stacks.

Cheng-Hung et al. [62] proposed an alternative method of rule removal in programmable switches. In contrast to OpenFlow switches where two timeout counters are utilized for this operation, P4 switches can automatically remove entries by inspecting the payload. Regarding the TCP the switch removes a flow entry when the session termination flags are existent. For UDP connections and half-closed TCP sessions, an idle timeout counter is used. The evaluation results show better TCAM utilization and less overhead for the controller.

Yazdinejad et al. [178] propose a new highly flexible architecture for programmable network devices that support all operations a switch is supposed to offer and a two step preprocessing framework suitable for FPGA boards. The proposed architecture promises to alleviate the designer of the **Hardware Description Languages (HDL)** details and accelerate the development process. **Recursive InterNetwork Architecture (RINA)** is an alternative network architecture based on **Inter-Process Communication (IPC)**. Authors of Reference [41] developed a RINA compatible Interior router using BMv2 software switch and P4.

COIN [176] is a data indexing mechanism for edge servers that is implemented in P4. The control plain maintains a two-dimensional space for associating the P4 switches with the cached data. When a query is related to cached data an immediate response is sent.

4 P4 AS A DATA PLANE LANGUAGE

Introducing programmability through a DSL raises many benefits and concerns as well. The developer can be more expressive when using a programming language instead of a fixed functionality interface. Though the developer is a human being and risk of bugs are always existent. This Sections devoted to techniques and tools used to debug, optimize and extend the P4 as a programming language.

4.1 Code Validation

Netdiff [37] is a framework built on top of Synmet symbolic execution engine that performs symbolic execution of two data plane configurations expressed in SEFL language and decides if they are equivalent or not. This process can act as a validation mechanism. Netdiff was able to identify many bugs in various data plane configurations.

KeySight [174, 191] is a three-module system P4 Programmable Data Plane that aims to minimize postcard packet processing. Using Packet Equivalence Class abstraction KeySight forwards and processes postcard packets when necessary. To evaluate KeySight, authors deployed it on a Tofino ASIC and on a smartNIC.

Freire et al. [42] proposed a simple yet powerful assertion language called ASSERT-P4 built on top of the KLEE tool and the P4 reference compiler able to identify bugs. Provided a P4 annotated program ASSERT-P4 produces a C-based model and performs symbolic execution to identify any assertion failures. To validate their design 4 P4 application from recent literature, the results of which show that ASSERT-P4 can identify bugs in common P4 applications in a reasonable time even though the assertion time grows exponentially with the of the tables and the number of assertion inserted to the P4 program.

Kohler et al. [87] developed an in-network complex event processing system called P4CEP using P4. Instead of performing these tasks on dedicated servers, P4CEP performs them directly in the data plane. P4CEP offers a compiler that integrates a user-provided P4 program written in any version of P4, adds its P4 code additions, and generates a final P4 target program. Additionally, P4CEP runtime offers the operator a control plane interface. Complex Event Processing operations are expressed using a rule-based specification language. Their evaluation was performed using a Netronome Agilio Smart NIC card and BMv2 P4 reference switch. Their results show that P4CEP achieves good performance. Finally, the authors present the limitations of P4 regarding performing in-network computations.

Vera [158] is a verification tool that performs symbolic execution on P4 programs to identify bugs. P4 code is translated to Symnet's SEFL language and is symbolically evaluated by the Symnet software tool. An innovative contribution the Vera tool introduces is that it goes a step further and extends its evaluation process on table entries (tables rules), structures that are dynamically populated by the control plane.

P4Box [122] is a P4 verification system that leverages the benefits dynamic enforcement and deploys run-time monitors in the data plane. These monitors are essentially P4 constructs that do not modify the intended behavior of the P4 code and introduce insignificant delay to the pipeline. Developers can use these constructs to verify the behavior of their P4 program.

SAFEP4 [38] is a type-safe version of P4 that eliminates many errors P4 is prone to. SAFEP4 tracks header dependencies and use many domain-specific optimizations to minimize the amount of bugs.

P4RL [154] is a system that adopts Reinforcement Learning to verify switches at runtime. More specifically P4RL use a reinforcement learning assisted fuzzy testing method to verify the switch behavior and uses a query language to specify the desired behavior of the switch. Tests on existing P4 application show the superior performance of P4RL against baseline approaches.

Freire et al. [43] explore the many C-like features of P4 to produce a C equivalent programs and then perform symbolic analysis to find violations and possible security concerns.

4.2 Code Optimization

As in all modern programming languages, the compiler undertakes the optimization of the code for the respective architecture. As a programming language P4 is able to exploit the knowledge of all these years. MATReduce framework [25] intends to produce a more efficient P4 pipeline by eliminating duplicate match operations. MATReduce is of two modules, the preprocessor module and the runtime-management module. The preprocessor is responsible for identifying dependencies between MATs, the desing of an improved compound MAT and finally the generation of an information file that is feeded to the second module. The runtime-management module is responsible for transforming user rules into the compound equivalent commands. For each rule the runtime-management modules considers the information-file to generate and unambiguous equivallent rule. Evaluation of the proposed system is done in both software and hardware implementation and the results show that the compound pipeline is more efficient.

pcube [152] is a preprocessor that offers primitives for easier development of P4 applications. pcude primitives include loops, min, and max operations. To evaluate pcube the authors implement two applications. In both applications, the lines of code and the correctness of the developed applications were measured. The results indicate that pcude is capable of reducing the overall size of the code.

P4LLVM [31] is an LLVM-based compiler that performs further optimizations in contrast to the default P4 compiler. The intermediate state is translated to LLVM-IR and target-independent optimization operations are performed. The results demonstrate that LLVM's minor optimizations lead to more optimized output when compared to P4 default compiler output. P4HL [57] a set of tools that accelerate the development of P4 programs. P4HL consists of two components, namely, E-Domino a high-level language, the P4HLC the compiler that translates E-Domino code in P4.

B-cache [186] is a behavior caching framework that supports stateful as well as stateless behaviors. B-Cache reduces the processing time of packets that traverse through complex pipelines by caching the behavior of the pipeline in a single cache MAT. The authors present optimized methods for cache MAT entries generation, take into consideration the limited memory offered by the data plane elements and device a method to keep in sync the expected output of the pipeline with the cache MAT. The authors claim that B-Cache can minimize the delay by 50% and increase throughput by 200% on BMv2 software switch.

Patra et al. [132] developed a compiler system called **Multi-Architecture Compiler System for Abstract Dataplanes (MACSAD)**. MACSAD combines the P4 language with **OpenData-Plane (ODP)** API to offer a high-level and cross-platform compiler. The MACSAD compiler consists of three components: (1) the auxiliary fronted, (2) the auxiliary back end, and (3) the core

compiler. MACSAD was extensively evaluated with different target platforms and under several processing pipelines. The performance and the scalability of it were extensively measured too. The results show that MACSAD performs match state of the art solutions. Authors of Reference [181] develop a compositional abstraction model using Boolean formulas. The proposed system includes a compiler that translates the P4 input program in boolean formulas. PRIME [131] is an interpreter that parses and merges P4 programs in a sequence specified by the administrator. A verification phase is also added to avoid any unnecessary loops.

4.3 Testing and Debugging

Troubleshooting a network is a difficult and error prone task. When including programmability to the network, this task may turn into a more complex problem. The research community has proposed several debugging tools and solutions to mitigate this problem. PFPSim [1] is a simulator for programmable data planes that uses SystemC framework to build a model of the target hardware and software modules. PFPSim links the P4 program to this model and associates the control plane model to the hardware. Finally, a traffic generator feeds the model while the PFPSim collects the events that are generated.

P4pktgen [124] is a tool that automatically generates test cases for P4 programs using the BMv2 target's json configuration file as input. Using **satisfiability modulo theories (SMT)** solvers and theory of bit vectors p4pktgen attempts to find a packet that satisfies all these constrains. Although p4pktgen support a subset of P4 language specification it proves that P4 programs introduce bugs like conventional programs written in languages like C.

NS4 [8] is a P4-based simulator that aims to bridge the gap between simulation and emulation tools. NS4 translates P4 code into discrete events and loads them in their respected modules. Doing so NS4 avoids the limitation of an emulation tool. To evaluate NS4 authors create a flat tree topology, simulate SilkRoad, and afterward compare it with a Software Load Balanced developed in ns-3 simulator. The final results show the NS4 is less computational heavy than its ns-3.

P4Tester [190] is an efficient testing system designed for programmable data planes able to find bugs in run time rules faults. To achieve this P4Tester proposes a new intermediate representation of the P4 program based on the Binary Decision Diagram and offers a source routing forwarding probing model. Similarly P4DB [192] is a debugger designed for P4-enabled networks. Using P4DB operators have access to a primitive set of commands similar to ones that offer traditional tools like GNU Debugger.

4.4 P4 Target Optimizations

A high-level language as P4 aims to abstract away target dependant details and offer a generic means to express the desired behaviour of a device. In the end, though the target-specific result s what matters the most. An improperly utilised target may lead to diminished performance. Smart and efficient allocation of P4 data structures to physical hardware components is important to gain the most of the data plane devices. Taking into consideration target-specific features is equally important.

Silva et al. [146] proposed the **Heterogeneous Data Plane (HDP)** platform, a platform that according to authors can further extend the programmable of data planes. HDP combines traditional x86 servers, FPGA boards, and Smart NICs and aims to eliminate hardware limitations by combining the aforementioned P4 targets effectively and efficiently.

Authors of Reference [81] propose a new architecture for mapping P4 match/actions operation to FPGA boards utilizing DCFL algorithm. The architecture is further optimized by introducing memory duplication for more memory accesses per clock cycle and TCAM rule offloading for rules that degrade the performance of the CDFL output.

Authors of Reference [19] present a P4 to VHDL translation framework. During the compilation process a set of optimized templates are used to translate the P4 code and output result is further optimized by the compiler using a pre-built evaluation library.

4.5 Virutalization of Data Plane

The purpose of the virtualization is the logical partitioning of resources. These virtualized resources are then lent to other users and utilized as seen fit by them. Many virtualization solution have been proposed by the research community over the past years.

Hyper4 [53] is a virtualization solution for programmable data planes that supports functions such as slicing and virtual networks. On a high level, a compiler translates a P4 input program into table entries that emulate the behavior of the original P4 program. Hyper4 though lacks support of P4 native actions and is slower that native solutions. MPVisor [184] is a Network Functions hypervisor able to decouple the tables NFs use and rearrange them in such a way that the underlying hardware is optimally utilized.

HyperVDP [183, 185] is a P4 hypervisor able to fully virtualize the underlying PDP while being resource-efficient and achieves high throughput. HyperVDP offers a dynamic compiler responsible for carrying out various novel techniques to optimize the output. Moe specifically the compiler receives multiple P4 programs as inputs and produces a unified output for multiple targets. To validate its performance, the authors implemented two prototypes, one based on BMv2 and one based on DPDK. VirtP4 [147, 148] is a P4 virtualization architecture that offers parallel execution of independent switches. A proof-of-concept implementation was done using NetFPGA SUME board where the performance exhibits that of similar solutions. P4MT [29] is an attempt to introduce multitenancy in the control plane of the P4Rutime. An experimental international network, called i-P4EN, demonstrated its feasibility.

5 DISCUSSION

It should be clear by now that P4 language has a wide range of applications and has gained a lot popularity over recent years. One survey article is unable to cover all these fields in detail, since its domains of application span across many fields of networking. In the field of Traffic Engineering, P4 is used for its ability to manipulate packet headers. INT is a heavily used concept that allows the transfer of network metadata across the networks using the user-generated traffic. Many new algorithms have been proposed over the years like HULA to load-balance the traffic between links, or detect link failures.

The security domain also benefits greatly by using P4. Hardware-based firewall were black box devices with little configuration options. Programmable switches can be tasked with firewall responsibilities and be highly flexible the same time. Taking advantage of the real time and enriched statistics novel AI-based solutions can be designed and offloaded directly to the data plane operating in line-rate speed. Moreover, the flexible programmable devices can easily be reconfigured to offer upgraded security services or different ones depending on the needs of each organization.

One other important category is task offloading to the data plane. VNFs were traditionally run on dedicated server, though this process was under utilizing the resources and resulted in low throughput. Programmable hardware can implement a wide variety of applications without the penalties of commodity server. Tasks that were not thought of also arose. For example, traffic generators receive a small payload and amplify it to produce artificial traffic with a fraction of the cost of dedicated traffic generators.

As a programming language, P4 has both benefits and challenges. Flexibility and expressiveness are amongst the most important benefits, while verification and bug identification are amongst

the most unwanted features. Especially for bugs, their identification in the early stages of developments can prevent network outages or unexpected behaviors.

6 FUTURE DIRECTIONS

In recent years, P4 has met the acceptance of both academia and industry. This trend is very likely to continue for the following years with more vendors, network providers and researchers embracing what P4 has to offer. The increasing number of interested parties will inevitably lead to new domains where P4 will be used. Moreover, these new domains will also lead to needs that the current specification does not cover. In the following subsections, future directions are presented in the form of new application domains and challenges.

6.1 Integrating In-network Computing to Data Plane Processes

In-network computing is a direct result of the programmability introduced to the data plane through languages like P4. As discussed in previous sections (Section 3.4.1) in-network computing enables data plane devices to partially or fully perform computations as the packets traverse the programmable fabric. This results in to assuming programmable devices as computing nodes with special capabilities like **Tensor Proccessing Units (TPUs)** [75]. Data plane devices though have a primary function of forwarding traffic but the opportunity of performing computations on the fly is a promising feature for future offloading techniques. To better exploit data plane offloading existing resource allocation algorithms should be developed and existing ones properly adjusted to into consideration the twofold nature of the data plane (traffic forwarding and data computations). Regarding traffic under QoS restriction, new routing algorithms should be developed to efficiently guide traffic through the appropriate nodes performing computations without interrupting the normal operation of the network.

6.2 Multi-protocol Stack Deployments

Multiple internet architectures have been proposed over the years promising to replace IP/TCP (e.g., RINA, NDN). The transition from one architecture to another is a challenging task though. The need for coexistence between multiple architectures will eventually emerge, since each one has benefits and limitations compared to others. Offering a programmable fabric P4 can help telecommunication providers implement multiple architectures and offer them as a service to stakeholders. Based on individual organisation needs one or many internet architectures may seem fit.

6.3 End-to-End Fine-grained Monitoring

The next generation of cellular networks (B5G/6G) [20, 108] aim for seamless connectivity, higher density than previous generations, and a more robust network relying heavily on AI to perform several operations. To achieve the mentioned above goals 6G will leverage the available aerial space through **Unmanned Aerial Vehicles (UAVs)** [145] and **Low Earth Orbit (LEO)** Satellites [85]. Programmable data planes can boost the accuracy in measuring delays, offer fast-failover mechanisms and even implement solutions previsously not posible to do so [105, 106]. Current solutions though are not designed to exploit the rich statistics offered by P4-based data planes. It is evident that efforts should be made in standardising the monitoring mechanisms with respect to metrics monitored and the structure of information.

6.4 Verification of Data Plane Behaviour

Introducing such a level programmability to the data plane in the form of abstractions introduces a series of challenges as well. One of the most important on of these challenges is the verification of a

P4 program. As P4-based deployments will gradually replace traditional ones and OpenFlow-based ones they will become more complex, a need to verify their proper behaviour will arise. Thankfully P4 in its latest version is not a Turing complete language, but this statement may not hold true for future versions of the language or data plane programmability in general. Moreover, transpilation of one language to another is still facing many limitations due to lack of tools and proper validation methods. The need of such tools will arise in the near future taking into consideration the rapid evolution of network programming DSLs.

6.5 Generalisation of Network Programming

P4 is a DSL for network programming. Despite that fact though, P4 is not capable of carrying out any network related computation. For example, the language cannot perform deep packet inspection in its current version ($P4_{16}$). Future services and infrastructure deployments expect a network that can accommodate multi-criteria needs (e.g., deep packet inspection, bizzare flow-prioritization polines, etc.) Future versions of the language should take into consideration current limitations and needs of the community. Offering control through P4 to scheduling and queuing policies of switches can further assist the design and development of custom solutions. Additionally, introducing loops to the language specification can help in many protocols (e.g., MPLS).

6.6 Requirements of New Hardware Devices

Programmable devices should offer a high level of flexibility and programmability to the developers without sacrificing throughput. Software targets are easy to extend and deploy, a statement not true for hardware ones. Hardware targets often lack many features due to ASIC design constrains. Hardware design often ends up in a compromise between set of features and processing speed. Future hardware designs should take into consideration limitations of existing solutions (e.g., inability to manipulate certain pipeline stages) without sacrificing throughput. Moreover, all hardware targets available today offer Tbps scale throughput. To boost the adoption of P4 in smaller scale networks 10 and 1 Gpbs variants of these ASICs should be designed.

7 CONCLUSION

In this survey, we presented the P4 programming language, a high-level domain-specific programming language for programmable networks. We begun by providing a historical overview of the Computer Network evolution dating back to their first appearance until today. This made clear that programmable networks are the next logical evolution of computer networks.

An overview of the P4 language followed. First, the two major versions of the language were presented, $P4_{14}$ and $P4_{16}$, with the latter being the latest. Emphasis was given to the latest version of the language, since it emended on many design choices of the previous version. The basic building blocks, alongside the various additional tasks that are often omitted but are equally important, were presented. The applications of P4 span across many sub-fields of computer networks. Traffic load-balancing and congestion control are only two of many examples in the domain of Traffic Engineering. Flexible hardware-based firewall systems are also a good example of P4's application in the domain of security.

P4 is also used to perform network-assisted tasks and in-network tasks. For example, Network Functions, like application-level load-balancers were originally executed in commodity servers with low-performance gains relative to cost. Programmable networks alleviate the server from these highly specialized and application-specific tasks. Finally, the concerns around P4 as a programming language were presented. Like conventional languages, P4 requires the debugging and optimization of the final binary file.

REFERENCES

[1] S. Abdi, U. Aftab, G. Bailey, B. Boughzala, F. Dewal, S. Parsazad, and E. Tremblay. 2016. PFPSim: A programmable forwarding plane simulator. In *Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'16)*. ACM/IEEE, 55–60. https://doi.org/10.1145/2881025.2881029

[2] A. Aghdai, M. Huang, D. Dai, Y. Xu, and J. Chao. 2018. Transparent edge gateway for mobile networks. In *Proceedings of the IEEE 26th International Conference on Network Protocols (ICNP'18)*. IEEE, 412–417. https://doi.org/10.1109/ICNP.2018.00057

[3] A. Aghdai, Y. Xu, and H. J. Chao. 2017. Design of a hybrid modular switch. In *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN'17)*. IEEE, 1–6. https://doi.org/10.1109/NFV-SDN.2017.8169825

[4] Amar Almaini, Ahmed Al-Dubai, Imed Romdhani, Martin Schramm, and Ayoub Alsarhan. 2021. Lightweight edge authentication for software defined networks. *Computing* 103, 2 (Feb. 2021), 291–311. https://doi.org/10.1007/s00607-020-00835-4

[5] Amir Alsadi, Davide Berardi, Franco Callegati, Andrea Melis, and Marco Prandini. 2021. A security monitoring architecture based on data plane programmability. In *Proceedings of the Joint European Conference on Networks and Communications and 6G Summit (EuCNC/6G'21)*. IEEE, 389–394. https://doi.org/10.1109/EuCNC/6GSummit51104.2021.9482549

[6] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. 2002. *Overview and Principles of Internet Traffic Engineering*. Technical Report RFC3272. RFC Editor. RFC3272 pages. https://doi.org/10.17487/rfc3272

[7] A. Azzouni, N. T. Mai Trang, R. Boutaba, and G. Pujolle. 2017. Limitations of openflow topology discovery protocol. In *Proceedings of the 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net'17)*. IEEE, 1–3. https://doi.org/10.1109/MedHocNet.2017.8001642

[8] Jiasong Bai, Jun Bi, Peng Kuang, Chengze Fan, Yu Zhou, and Cheng Zhang. 2018. NS4: Enabling programmable data plane simulation. In *Proceedings of the Symposium on SDN Research (SOSR'18)*. Association for Computing Machinery, New York, NY, Article 12, 7 pages. https://doi.org/10.1145/3185467.3185470

[9] Cristian Hernandez Benet, Andreas J. Kassler, Theophilus Benson, and Gergely Pongracz. 2018. MP-HULA: Multipath transport aware load balancing using programmable data planes. In *Proceedings of the Morning Workshop on In-network Computing (NetCompute'18)*. Association for Computing Machinery, New York, NY, 7–13. https://doi.org/10.1145/3229591.3229596

[10] D. Bhamare, A. Kassler, J. Vestin, M. A. Khoshkholghi, and J. Taheri. 2019. IntOpt: In-band network telemetry optimization for NFV service chain monitoring. In *Proceedings of the IEEE International Conference on Communications (ICC'19)*. IEEE, 1–7. https://doi.org/10.1109/ICC.2019.8761722

[11] D. Bhat, J. Anderson, P. Ruth, M. Zink, and K. Keahey. 2019. Application-based QoE support with P4 and OpenFlow. In *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'19)*. IEEE, 817–823. https://doi.org/10.1109/INFCOMW.2019.8845180

[12] Giuseppe Bianchi, Marco Bonola, Antonio Capone, and Carmelo Cascone. 2014. OpenState: Programming platform-independent stateful openflow applications inside the switch. *SIGCOMM Comput. Commun. Rev.* 44, 2 (Apr. 2014), 44–51. https://doi.org/10.1145/2602204.2602211

[13] Michel Bonfim, Marcelo Santos, Kelvin Dias, and Stênio Fernandes. 2020. A real-time attack defense framework for 5G network slicing. *Software: Pract. Exper.* 50, 7 (2020), 1228–1257. https://doi.org/10.1002/spe.2800 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2800

[14] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.* 44, 3 (July 2014), 87–95. https://doi.org/10.1145/2656877.2656890

[15] Wolfgang Braun and Michael Menth. 2014. Software-defined networking using OpenFlow: Protocols, applications and architectural design choices. *Future Internet* 6, 2 (2014), 302–336. https://doi.org/10.3390/fi6020302

[16] Gordon Brebner and Weirong Jiang. 2014. High-speed packet processing using reconfigurable computing. *IEEE Micro* 34, 1 (2014), 8–18. https://doi.org/10.1109/MM.2014.19

[17] Jakub Cabal, Pavel Benáček, Lukáš Kekely, Michal Kekely, Viktor Puš, and Jan Kořenek. 2018. Configurable FPGA packet parser for terabit networks with guaranteed wire-speed throughput. In *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'18)*. Association for Computing Machinery, New York, NY, 249–258. https://doi.org/10.1145/3174243.3174250

[18] Jiamin Cao, Jun Bi, Yu Zhou, and Cheng Zhang. 2018. CoFilter: A high-performance switch-assisted stateful packet filter. In *Proceedings of the ACM SIGCOMM Conference on Posters and Demos (SIGCOMM'18)*. Association for Computing Machinery, New York, NY, 9–11. https://doi.org/10.1145/3234200.3234251

[19] Z. Cao, H. Su, Q. Yang, J. Shen, M. Wen, and C. Zhang. 2020. P4 to FPGA-A fast approach for generating efficient network processors. *IEEE Access* 8 (2020), 23440–23456. https://doi.org/10.1109/ACCESS.2020.2970683

[20] David Carrascal, Elisa Rojas, Joaquin Alvarez-Horcajo, Diego Lopez-Pajares, and Isaías Martínez-Yelmo. 2020. Analysis of P4 and XDP for IoT programmability in 6G and beyond. *IoT* 1, 2 (2020), 605–622. https://doi.org/10.3390/iot1020031

[21] Lucas Castanheira, Alberto Schaeffer-Filho, and Theophilus A. Benson. 2019. P4-InTel: Bridging the gap between ICF diagnosis and functionality. In *Proceedings of the 1st ACM CoNEXT Workshop on Emerging In-network Computing Paradigms (ENCP'19)*. Association for Computing Machinery, New York, NY, 21–26. https://doi.org/10.1145/3359993.3366648

[22] C. Chen, H. Fang, and M. S. Iqbal. 2020. QoSTCP: Provide consistent rate guarantees to TCP flows in software defined networks. In *Proceedings of the IEEE International Conference on Communications (ICC'20)*. IEEE, 1–6. https://doi.org/10.1109/ICC40277.2020.9148715

[23] Xiaoqi Chen, Shir Landau Feibish, Yaron Koral, Jennifer Rexford, Ori Rottenstreich, Steven A. Monetti, and Tzuu-Yi Wang. 2019. Fine-grained queue measurement in the data plane. In *Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies (CoNEXT'19)*. Association for Computing Machinery, New York, NY, 15–29. https://doi.org/10.1145/3359989.3365408

[24] X. Chen, D. Zhang, X. Wang, K. Zhu, and H. Zhou. 2019. P4SC: Towards high-performance service function chain implementation on the P4-capable device. In *Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM'19)*. IEEE, Arlington, VA, 1–9.

[25] X. Chen, D. Zhang, and H. Zhou. 2018. MATReduce: Towards high-performance P4 pipeline by reducing duplicate match operations. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'18)*. IEEE, 1–7. https://doi.org/10.1109/GLOCOM.2018.8647320

[26] N. Choi, L. Jagadeesan, Y. Jin, N. N. Mohanasamy, M. R. Rahman, K. Sabnani, and M. Thottan. 2019. Run-time performance monitoring, verification, and healing of end-to-end services. In *Proceedings of the IEEE Conference on Network Softwarization (NetSoft'19)*. IEEE, Paris, France, 30–35. https://doi.org/10.1109/NETSOFT.2019.8806660

[27] Sean Choi, Seo Jin Park, Muhammad Shahbaz, Balaji Prabhakar, and Mendel Rosenblum. 2019. Toward scalable replication systems with predictable tails using programmable data planes. In *Proceedings of the 3rd Asia-Pacific Workshop on Networking 2019 (APNet'19)*. Association for Computing Machinery, New York, NY, 78–84. https://doi.org/10.1145/3343180.3343181

[28] Sean Choi, Muhammad Shahbaz, Balaji Prabhakar, and Mendel Rosenblum. 2019. λ-NIC: Interactive serverless compute on SmartNICs. In *Proceedings of the ACM SIGCOMM Conference Posters and Demos (SIGCOMM'19)*. Association for Computing Machinery, New York, NY, 151–152. https://doi.org/10.1145/3342280.3342341

[29] B. Chung, C. Tseng, J. H. Chen, and J. Mambretti. 2019. P4MT: Multi-tenant support prototype for international P4 testbed. In *Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'19)*. ACM/IEEE, Cambridge, UK, 1–2. https://doi.org/10.1109/ANCS.2019.8901869

[30] M. V. B. da Silva, A. S. Jacobs, R. J. Pfitscher, and L. Z. Granville. 2018. IDEAFIX: Identifying elephant flows in P4-based IXP networks. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–6. https://doi.org/10.1109/GLOCOM.2018.8647685

[31] T. K. Dangeti, V. Keerthy Soundararajan, and R. Upadrasta. 2018. P4LLVM: An LLVM-based P4 compiler. In *Proceedings of the IEEE 26th International Conference on Network Protocols (ICNP'18)*. IEEE, 424–429. https://doi.org/10.1109/ICNP.2018.00059

[32] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti. 2017. A survey on the security of stateful SDN data planes. *IEEE Commun. Surveys Tutor.* 19, 3 (2017), 1701–1725. https://doi.org/10.1109/COMST.2017.2689819

[33] Trisha Datta, Nick Feamster, Jennifer Rexford, and Liang Wang. 2019. SPINE: Surveillance protection in the network elements. In *Proceedings of the 9th USENIX Workshop on Free and Open Communications on the Internet (FOCI'19)*. USENIX Association. Retrieved from https://www.usenix.org/conference/foci19/presentation/datta.

[34] M. Dimolianis, A. Pavlidis, and V. Maglaris. 2020. A multi-feature DDoS detection schema on P4 network hardware. In *Proceedings of the 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN'20)*. IEEE, Paris, France, 1–6. https://doi.org/10.1109/ICIN48450.2020.9059327

[35] D. Ding, M. Savi, G. Antichi, and D. Siracusa. 2020. An incrementally-deployable P4-enabled architecture for network-wide heavy-hitter detection. *IEEE Trans. Netw. Service Manage.* 17, 1 (2020), 75–88. https://doi.org/10.1109/TNSM.2020.2968979

[36] DPDK. 2022. Data Plane Development Kit. Retrieved from https://www.dpdk.org/.

[37] Dragos Dumitrescu, Radu Stoenescu, Matei Popovici, Lorina Negreanu, and Costin Raiciu. 2019. Dataplane equivalence and its applications. In *Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI'19)*. USENIX Association, 683–698. Retrieved from https://www.usenix.org/conference/nsdi19/presentation/dumitrescu.

[38] Matthias Eichholz, E. Campbell, Nate Foster, G. Salvaneschi, and M. Mezini. 2019. How to avoid making a billion-dollar mistake: Type-safe data plane programming with SafeP4. Retrieved from https://arxiv.org/abs/1906.07223.

[39] P. Engelhard, A. Zachlod, J. Schulz-Zander, and S. Du. 2019. Toward scalable and virtualized massive wireless sensor networks. In *Proceedings of the International Conference on Networked Systems (NetSys'19)*. IEEE, München, Germany, 1–6. https://doi.org/10.1109/NetSys.2019.8854518

[40] A. Febro, H. Xiao, and J. Spring. 2018. Telephony denial of service defense at data plane (TDoSD@DP). In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–6. https://doi.org/10.1109/NOMS.2018.8406281

[41] Carolina Fernández, Sergio Giménez, Eduard Grasa, and Steve Bunch. 2020. A P4-enabled RINA interior router for software-defined data centers. *Computers* 9, 3 (2020). https://doi.org/10.3390/computers9030070

[42] Lucas Freire, Miguel Neves, Lucas Leal, Kirill Levchenko, Alberto Schaeffer-Filho, and Marinho Barcellos. 2018. Uncovering bugs in P4 programs with assertion-based verification. In *Proceedings of the Symposium on SDN Research (SOSR'18)*. Association for Computing Machinery, New York, NY, Article 4, 7 pages. https://doi.org/10.1145/3185467.3185499

[43] Lucas Freire, Miguel Neves, Alberto Schaeffer-Filho, and Marinho Barcellos. 2017. POSTER: Finding vulnerabilities in P4 programs with assertion-based verification. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*. Association for Computing Machinery, New York, NY, 2495–2497. https://doi.org/10.1145/3133956.3138837

[44] K. Friday, E. Kfoury, E. Bou-Harb, and J. Crichigno. 2020. Towards a unified in-network DDoS detection and mitigation strategy. In *Proceedings of the 6th IEEE Conference on Network Softwarization (NetSoft'20)*. IEEE, 218–226. https://doi.org/10.1109/NetSoft48620.2020.9165336

[45] Wallas Froes, Lucas Santos, Leobino N. Sampaio, Magnos Martinello, Alextian Liberato, and Rodolfo S. Villaca. 2020. ProgLab: Programmable labels for QoS provisioning on software defined networks. *Comput. Commun.* 161 (2020), 99–108. https://doi.org/10.1016/j.comcom.2020.07.026

[46] Yi Gao, Yuan Jing, and Wei Dong. 2018. UniROPE: Universal and robust packet trajectory tracing for software-defined networks. *IEEE/ACM Trans. Netw.* 26, 6 (Dec. 2018), 2515–2527. https://doi.org/10.1109/TNET.2018.2871213

[47] Junjie Geng, Jinyao Yan, Yangbiao Ren, and Yuan Zhang. 2018. Design and implementation of network monitoring and scheduling architecture based on P4. In *Proceedings of the 2nd International Conference on Computer Science and Application Engineering (CSAE'18)*. Association for Computing Machinery, New York, NY, Article 182, 6 pages. https://doi.org/10.1145/3207677.3278059

[48] Junjie Geng, Jinyao Yan, and Yuan Zhang. 2019. P4QCN: Congestion control using P4-capable device in data center networks. *Electronics* 8, 3 (2019). https://doi.org/10.3390/electronics8030280

[49] Hans Giesen, Lei Shi, John Sonchack, Anirudh Chelluri, Nishanth Prabhu, Nik Sultana, Latha Kant, Anthony J McAuley, Alexander Poylisher, André DeHon, and Boon Thau Loo. 2018. In-network computing to the rescue of faulty links. In *Proceedings of the Morning Workshop on In-network Computing (NetCompute'18)*. Association for Computing Machinery, New York, NY, 1–6. https://doi.org/10.1145/3229591.3229595

[50] The P4.org Applications Working Group. 2021. In-band Network Telemetry (INT) Dataplane Specification. Retrieved from https://github.com/p4lang/p4-applications/blob/master/docs/INT_v2_1.pdf.

[51] B. Guan and S. Shen. 2019. FlowSpy: An efficient network monitoring framework using P4 in software-defined networks. In *Proceedings of the IEEE 90th Vehicular Technology Conference (VTC'19)*. IEEE, 1–5. https://doi.org/10.1109/VTCFall.2019.8891487

[52] Joel M. Halpern, Robert Haas, Avri Doria, Ligang Dong, Weiming Wang, Hormuzd M. Khosravi, Jamal Hadi Salim, and Ram Gopal. 2010. Forwarding and Control Element Separation (ForCES) Protocol Specification. RFC 5810. (Mar. 2010). https://doi.org/10.17487/RFC5810

[53] David Hancock and Jacobus van der Merwe. 2016. HyPer4: Using P4 to virtualize the programmable data plane. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies (CoNEXT'16)*. Association for Computing Machinery, New York, NY, 35–49. https://doi.org/10.1145/2999572.2999607

[54] Zijun Hang, Yang Shi, Mei Wen, Wei Quan, and Chunyuan Zhang. 2019. SWAP: A sliding window algorithm for in-network packet measurement. In *Proceedings of the 3rd International Conference on High Performance Compilation, Computing and Communications (HP3C'19)*. Association for Computing Machinery, New York, NY, 84–89. https://doi.org/10.1145/3318265.3318280

[55] Z. Hang, Y. Shi, M. Wen, and C. Zhang. 2019. TBSW: Time-based sliding window algorithm for network traffic measurement. In *Proceedings of the IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS'19)*. IEEE, 1305–1310. https://doi.org/10.1109/HPCC/SmartCity/DSS.2019.00182

[56] Z. Hang, M. Wen, Y. Shi, and C. Zhang. 2019. Interleaved sketch: Toward consistent network telemetry for commodity programmable switches. *IEEE Access* 7 (2019), 146745–146758. https://doi.org/10.1109/ACCESS.2019.2946704

[57] Zijun Hang, Mei Wen, Yang Shi, and Chunyuan Zhang. 2019. Programming protocol-independent packet processors high-level programming (P4HLP): Towards unified high-level programming for a commodity programmable switch. *Electronics* 8, 9 (2019). https://doi.org/10.3390/electronics8090958

[58] Rob Harrison, Qizhe Cai, Arpit Gupta, and Jennifer Rexford. 2018. Network-wide heavy-hitter detection with commodity switches. In *Proceedings of the Symposium on SDN Research (SOSR'18)*. Association for Computing Machinery, New York, NY, Article 8, 7 pages. https://doi.org/10.1145/3185467.3185476

[59] Frederik Hauser, Marco Häberle, Daniel Merling, Steffen Lindner, Vladimir Gurevich, Florian Zeiger, Reinhard Frank, and Michael Menth. 2021. A Survey on Data Plane Programming with P4: Fundamentals, Advances, and Applied Research. Retrieved from https://arxiv.org/abs/2101.10632.

[60] F. Hauser, M. Häberle, M. Schmidt, and M. Menth. 2020. P4-IPsec: Site-to-site and host-to-site VPN with IPsec in P4-based SDN. *IEEE Access* 8 (2020), 139567–139586. https://doi.org/10.1109/ACCESS.2020.3012738

[61] F. Hauser, M. Schmidt, M. Häberle, and M. Menth. 2020. P4-MACsec: Dynamic topology monitoring and data layer protection with MACsec in P4-Based SDN. *IEEE Access* 8 (2020), 58845–58858. https://doi.org/10.1109/ACCESS.2020.2982859

[62] Cheng-Hung He, Brian Y. Chang, Suchandra Chakraborty, Chien Chen, and Li Chun Wang. 2018. A zero flow entry expiration timeout P4 switch. In *Proceedings of the Symposium on SDN Research (SOSR'18)*. Association for Computing Machinery, New York, NY, Article 19, 2 pages. https://doi.org/10.1145/3185467.3190785

[63] Yongchao He and Wenfei Wu. 2019. Fully functional rate limiter design on programmable hardware switches. In *Proceedings of the ACM SIGCOMM Conference Posters and Demos (SIGCOMM'19)*. Association for Computing Machinery, New York, NY, 159–160. https://doi.org/10.1145/3342280.3342344

[64] K. Hirata and T. Tachibana. 2019. Implementation of multiple routing configurations on software-defined networks with P4. In *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, Lanzhou, China, 13–16. https://doi.org/10.1109/APSIPAASC47483.2019.9023230

[65] Toke Høiland-Jørgensen, Jesper Dangaard Brouer, Daniel Borkmann, John Fastabend, Tom Herbert, David Ahern, and David Miller. 2018. The EXpress data path: Fast programmable packet processing in the operating system kernel. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT'18)*. Association for Computing Machinery, New York, NY, 54–66. https://doi.org/10.1145/3281411.3281443

[66] Thomas Holterbach, Edgar Costa Molero, Maria Apostolaki, Alberto Dainotti, Stefano Vissicchio, and Laurent Vanbever. 2019. Blink: Fast connectivity recovery entirely in the data plane. In *Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. USENIX Association, 161–176. Retrieved from https://www.usenix.org/conference/nsdi19/presentation/holterbach.

[67] Kuo-Feng Hsu, Ryan Beckett, Ang Chen, Jennifer Rexford, and David Walker. 2020. Contra: A programmable system for performance-aware routing. In *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI'20)*. USENIX Association, 701–721. Retrieved from https://www.usenix.org/conference/nsdi20/presentation/hsu.

[68] Kuo-Feng Hsu, Praveen Tammana, Ryan Beckett, Ang Chen, Jennifer Rexford, and David Walker. 2020. Adaptive weighted traffic splitting in programmable data planes. In *Proceedings of the Symposium on SDN Research (SOSR'20)*. Association for Computing Machinery, New York, NY, 103–109. https://doi.org/10.1145/3373360.3380841

[69] R. Hwang, V. Nguyen, and P. Lin. 2018. StateFit: A security framework for SDN programmable data plane model. In *Proceedings of the 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN'18)*. IEEE, 168–173. https://doi.org/10.1109/I-SPAN.2018.00035

[70] J. Hyun, N. Van Tu, and J. W. Hong. 2018. Towards knowledge-defined networking using in-band network telemetry. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'18)*. IEEE, 1–7. https://doi.org/10.1109/NOMS.2018.8406169

[71] Vimalkumar Jeyakumar, Mohammad Alizadeh, Yilong Geng, Changhoon Kim, and David Mazières. 2014. Millions of little minions: Using packets for low latency network programming and visibility. *SIGCOMM Comput. Commun. Rev.* 44, 4 (Aug. 2014), 3–14. https://doi.org/10.1145/2740070.2626292

[72] J. Jiang and Y. Zhang. 2019. An accurate congestion control mechanism in programmable network. In *Proceedings of the IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC'19)*. Las Vegas, NV, IEEE, 0673–0677. https://doi.org/10.1109/CCWC.2019.8666502

[73] Deepanshu Jindal, Raj Joshi, and Ben Leong. 2019. P4TrafficTool: Automated code generation for P4 traffic generators and analyzers. In *Proceedings of the ACM Symposium on SDN Research (SOSR'19)*. Association for Computing Machinery, New York, NY, 152–153. https://doi.org/10.1145/3314148.3318047

[74] Raj Joshi, Ting Qu, Mun Choon Chan, Ben Leong, and Boon Thau Loo. 2018. BurstRadar: Practical real-time microburst monitoring for datacenter networks. In *Proceedings of the 9th Asia-Pacific Workshop on Systems (APSys'18)*. Association for Computing Machinery, New York, NY, Article 8, 8 pages. https://doi.org/10.1145/3265723.3265731

[75] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-datacenter performance analysis of a tensor processing unit. *SIGARCH Comput. Archit. News* 45, 2 (June 2017), 1–12. https://doi.org/10.1145/3140659.3080246

[76] N. S. Kagami, R. I. T. da Costa Filho, and L. P. Gaspary. 2020. CAPEST: Offloading network capacity and available bandwidth estimation to programmable data planes. *IEEE Trans. Netw. Service Manage.* 17, 1 (2020), 175–189. https://doi.org/10.1109/TNSM.2019.2934316

[77] E. Kaljic, A. Maric, P. Njemcevic, and M. Hadzialic. 2019. A survey on data plane flexibility and programmability in software-defined networking. *IEEE Access* 7 (2019), 47804–47840. https://doi.org/10.1109/ACCESS.2019.2910140

[78] Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, and Jennifer Rexford. 2016. HULA: Scalable load balancing using programmable data planes. In *Proceedings of the Symposium on SDN Research (SOSR'16)*. Association for Computing Machinery, New York, NY, Article 10, 12 pages. https://doi.org/10.1145/2890955.2890968

[79] Sukhveer Kaur, Krishan Kumar, and Naveen Aggarwal. 2021. A review on P4-Programmable data planes: Architecture, research efforts, and future directions. *Comput. Commun.* 170 (2021), 109–129. https://doi.org/10.1016/j.comcom.2021.01.027

[80] E. Kawaguchi, H. Kasuga, and N. Shinomiya. 2019. Unsplittable flow edge load factor balancing in SDN using P4 runtime. In *Proceedings of the 29th International Telecommunication Networks and Applications Conference (ITNAC'19)*. IEEE, 1–6. https://doi.org/10.1109/ITNAC46935.2019.9077984

[81] M. Kekely and J. Korenek. 2017. Mapping of P4 match action tables to FPGA. In *Proceedings of the 27th International Conference on Field Programmable Logic and Applications (FPL'17)*. IEEE, 1–2. https://doi.org/10.23919/FPL.2017.8056768

[82] E. F. Kfoury, J. Crichigno, E. Bou-Harb, D. Khoury, and G. Srivastava. 2019. Enabling TCP pacing using programmable data plane switches. In *Proceedings of the 42nd International Conference on Telecommunications and Signal Processing (TSP'19)*. IEEE, 273–277. https://doi.org/10.1109/TSP.2019.8768888

[83] Jehandad Khan and Peter Athanas. 2018. Query language for large-scale P4 network debugging. In *Proceedings of the Symposium on Architectures for Networking and Communications Systems (ANCS'18)*. Association for Computing Machinery, New York, NY, 162–164. https://doi.org/10.1145/3230718.3232108

[84] Sajad Khorsandroo, Adrián Gallego Sánchez, Ali Saman Tosun, JM Arco, and Roberto Doriguzzi-Corin. 2021. Hybrid SDN evolution: A comprehensive survey of the state-of-the-art. *Comput. Netw.* 192 (2021), 107981. https://doi.org/10.1016/j.comnet.2021.107981

[85] O. Kodheli, E. Lagunas, N. Maturo, S. K. Sharma, B. Shankar, J. F. M. Montoya, J. C. M. Duncan, D. Spano, S. Chatzinotas, S. Kisseleff, J. Querol, L. Lei, T. X. Vu, and G. Goussetis. 2021. Satellite communications in the new space era: A survey and future challenges. *IEEE Commun. Surveys Tutor.* 23, 1 (2021), 70–109. https://doi.org/10.1109/COMST.2020.3028247

[86] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. 2000. The click modular router. *ACM Trans. Comput. Syst.* 18, 3 (Aug. 2000), 263–297. https://doi.org/10.1145/354871.354874

[87] Thomas Kohler, Ruben Mayer, Frank Dürr, Marius Maaß, Sukanya Bhowmik, and Kurt Rothermel. 2018. P4CEP: Towards in-network complex event processing. In *Proceedings of the Morning Workshop on In-network Computing (NetCompute'18)*. Association for Computing Machinery, New York, NY, 33–38. https://doi.org/10.1145/3229591.3229593

[88] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. 2015. Software-defined networking: A comprehensive survey. *Proc. IEEE* 103, 1 (2015), 14–76. https://doi.org/10.1109/JPROC.2014.2371999

[89] M. Kuka, K. Vojanec, J. Kučera, and P. Benáček. 2019. Accelerated DDoS attacks mitigation using programmable data plane. In *Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'19)*. ACM/IEEE, 1–3. https://doi.org/10.1109/ANCS.2019.8901882

[90] R. Kundel, J. Blendin, T. Viernickel, B. Koldehofe, and R. Steinmetz. 2018. P4-CoDel: Active queue management in programmable data planes. In *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN'18)*. IEEE, 1–4. https://doi.org/10.1109/NFV-SDN.2018.8725736

[91] R. Kundel, L. Nobach, J. Blendin, H. Kolbe, G. Schyguda, V. Gurevich, B. Koldehofe, and R. Steinmetz. 2019. P4-BNG: Central office network functions on programmable packet pipelines. In *Proceedings of the 15th International*

Conference on Network and Service Management (CNSM'19). IEEE, 1–9. https://doi.org/10.23919/CNSM46954.2019.9012666

[92] Ralf Kundel, Leonhard Nobach, Jeremias Blendin, Wilfried Maas, Andreas Zimber, Hans-Joerg Kolbe, Georg Schyguda, Vladimir Gurevich, Rhaban Hark, Boris Koldehofe, and Ralf Steinmetz. 2021. OpenBNG: Central office network functions on programmable data plane hardware. *Int. J. Netw. Manage.* 31, 1 (2021), e2134. https://doi.org/10.1002/nem.2134 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nem.2134 e2134 nem.2134.

[93] R. Kundel, F. Siegmund, J. Blendin, A. Rizk, and B. Koldehofe. 2020. P4STA: High performance packet timestamping with programmable packet processors. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–9. https://doi.org/10.1109/NOMS47738.2020.9110290

[94] Jan Kučera, Diana Andreea Popescu, Han Wang, Andrew Moore, Jan Kořenek, and Gianni Antichi. 2020. Enabling event-triggered data plane monitoring. In *Proceedings of the Symposium on SDN Research (SOSR'20)*. Association for Computing Machinery, New York, NY, 14–26. https://doi.org/10.1145/3373360.3380830

[95] Â. C. Lapolli, J. Adilson Marques, and L. P. Gaspary. 2019. Offloading real-time DDoS attack detection to programmable data planes. In *Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM'19)*. IEEE, Arlington, VA, 19–27.

[96] A. Lara, A. Kolasani, and B. Ramamurthy. 2014. Network innovation using OpenFlow: A survey. *IEEE Commun. Surveys Tutor.* 16, 1 (2014), 493–512. https://doi.org/10.1109/SURV.2013.081313.00105

[97] Abir Laraba, Jérôme François, Shihabur Rahman Chowdhury, Isabelle Chrisment, and Raouf Boutaba. 2021. Mitigating TCP protocol misuse with programmable data planes. *IEEE Trans. Netw. Service Manage.* 18, 1 (2021), 760–774. https://doi.org/10.1109/TNSM.2021.3054528

[98] Abir Laraba, Jérôme François, Isabelle Chrisment, Shihabur Rahman Chowdhury, and Raouf Boutaba. 2020. Defeating protocol abuse with P4: Application to explicit congestion notification. In *Proceedings of the IFIP Networking Conference (Networking)*. IEEE, 431–439.

[99] B. Lewis, M. Broadbent, and N. Race. 2019. P4ID: P4 enhanced intrusion detection. In *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN'19)*. IEEE, Dallas, Texas, 1–4. https://doi.org/10.1109/NFV-SDN47374.2019.9040044

[100] Guanyu Li, Menghao Zhang, Chang Liu, Xiao Kong, Ang Chen, Guofei Gu, and Haixin Duan. 2019. NETHCF: Enabling line-rate and adaptive spoofed IP traffic filtering. In *Proceedings of the IEEE 27th International Conference on Network Protocols (ICNP'19)*. IEEE, 1–12. https://doi.org/10.1109/ICNP.2019.8888057

[101] Y. Lin, C. Huang, and S. Tsai. 2019. SDN soft computing application for detecting heavy hitters. *IEEE Trans. Industr. Inform.* 15, 10 (2019), 5690–5699. https://doi.org/10.1109/TII.2019.2909933

[102] Y. Lin, T. Huang, and S. Tsai. 2019. Enhancing 5G/IoT transport security through content permutation. *IEEE Access* 7 (2019), 94293–94299. https://doi.org/10.1109/ACCESS.2019.2926479

[103] G. Liu, W. Quan, N. Cheng, N. Lu, H. Zhang, and X. Shen. 2020. P4NIS: Improving network immunity against eavesdropping with programmable data planes. In *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM'20)*. IEEE, 91–96. https://doi.org/10.1109/INFOCOMWKSHPS50562.2020.9162975

[104] M. Liu, D. Gao, G. Liu, J. He, L. Jin, C. Zhou, and F. Yang. 2019. Learning-based adaptive network immune mechanism to defense eavesdropping attacks. *IEEE Access* 7 (2019), 182814–182826. https://doi.org/10.1109/ACCESS.2019.2956805

[105] Diego Lopez-Pajares, Joaquin Alvarez-Horcajo, Elisa Rojas, Juan A. Carral, and Isaias Martinez-Yelmo. 2020. One-shot multiple disjoint path discovery protocol (1S-MDP). *IEEE Commun. Lett.* 24, 8 (2020), 1660–1663. https://doi.org/10.1109/LCOMM.2020.2990885

[106] Diego Lopez-Pajares, Elisa Rojas, Juan A. Carral, Isaias Martinez-Yelmo, and Joaquin Alvarez-Horcajo. 2021. The disjoint multipath challenge: Multiple disjoint paths guaranteeing scalability. *IEEE Access* 9 (2021), 74422–74436. https://doi.org/10.1109/ACCESS.2021.3080931

[107] Y. Lu and K. C. Lin. 2019. Enabling inference inside software switches. In *Proceedings of the 20th Asia-Pacific Network Operations and Management Symposium (APNOMS'19)*. ACM/IEEE, 1–4. https://doi.org/10.23919/APNOMS.2019.8893042

[108] Yang Lu and Xianrong Zheng. 2020. 6G: A survey on technologies, scenarios, challenges, and the related issues. *J. Industr. Info. Integr.* 19 (2020), 100158. https://doi.org/10.1016/j.jii.2020.100158

[109] André Luiz R. Madureira, Francisco Renato C. Araújo, and Leobino N. Sampaio. 2020. On supporting IoT data aggregation through programmable data planes. *Comput. Netw.* 177 (2020), 107330. https://doi.org/10.1016/j.comnet.2020.107330

[110] R. F. T. Martins, F. L. Verdi, R. Villaça, and L. F. U. Garcia. 2018. Using probabilistic data structures for monitoring of multi-tenant P4-based networks. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC'18)*. IEEE, 00204–00207. https://doi.org/10.1109/ISCC.2018.8538352

[111] Steven McCanne and Van Jacobson. 1993. The BSD packet filter: A new architecture for user-level packet capture. In *Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings (USENIX'93)*. USENIX Association, 2.

[112] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.* 38, 2 (Mar. 2008), 69–74. https://doi.org/10.1145/1355734.1355746

[113] Michael Menth, Habib Mostafaei, Daniel Merling, and Marco Häberle. 2019. Implementation and evaluation of activity-based congestion management using P4 (P4-ABC). *Future Internet* 11, 7 (2019). https://doi.org/10.3390/fi11070159

[114] R. Miguel, S. Signorello, and F. M. V. Ramos. 2018. Named data networking with programmable switches. In *Proceedings of the IEEE 26th International Conference on Network Protocols (ICNP'18)*. IEEE, 400–405. https://doi.org/10.1109/ICNP.2018.00055

[115] A. Mohammadkhan, S. Panda, S. G. Kulkarni, K. K. Ramakrishnan, and L. N. Bhuyan. 2019. P4NFV: P4 enabled NFV systems with SmartNICs. In *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN'19)*. IEEE, 1–7. https://doi.org/10.1109/NFV-SDN47374.2019.9040000

[116] Edgar Costa Molero, Stefano Vissicchio, and Laurent Vanbever. 2018. Hardware-accelerated network control planes. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks (HotNets'18)*. Association for Computing Machinery, New York, NY, 120–126. https://doi.org/10.1145/3286062.3286080

[117] D. Moro, M. Peuster, H. Karl, and A. Capone. 2019. Demonstrating FOP4: A flexible platform to prototype NFV offloading scenarios. In *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN'19)*. IEEE, 1–2. https://doi.org/10.1109/NFV-SDN47374.2019.9040056

[118] D. Moro, M. Peuster, H. Karl, and A. Capone. 2019. FOP4: Function offloading prototyping in heterogeneous and programmable network scenarios. In *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN'19)*. IEEE, 1–6. https://doi.org/10.1109/NFV-SDN47374.2019.9040052

[119] D. Moro, G. Verticale, and A. Capone. 2020. A framework for network function decomposition and deployment. In *Proceedings of the 16th International Conference on the Design of Reliable Communication Networks (DRCN'20)*. IEEE, 1–6. https://doi.org/10.1109/DRCN48652.2020.1570613823

[120] Adam Morrison, Lei Xue, Ang Chen, and Xiapu Luo. 2018. Enforcing context-aware BYOD policies with in-network security. In *Proceedings of the 10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud'18)*. USENIX Association. Retrieved from https://www.usenix.org/conference/hotcloud18/presentation/morrison.

[121] Niranjhana Narayanan, Ganesh C. Sankaran, and Krishna M. Sivalingam. 2019. Mitigation of security attacks in the SDN data plane using P4-enabled switches. In *Proceedings of the IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS'19)*. IEEE, 1–6. https://doi.org/10.1109/ANTS47819.2019.9118071

[122] M. Neves, B. Huffaker, K. Levchenko, and M. Barcellos. 2019. Dynamic property enforcement in programmable data planes. In *Proceedings of the IFIP Networking Conference (IFIP'19)*. IEEE, 1–9. https://doi.org/10.23919/IFIPNetworking.2019.8816830

[123] B. Niu, J. Kong, S. Tang, Y. Li, and Z. Zhu. 2019. Visualize your IP-over-optical network in realtime: A P4-based flexible multilayer in-band network telemetry (ML-INT) system. *IEEE Access* 7 (2019), 82413–82423. https://doi.org/10.1109/ACCESS.2019.2924332

[124] Andres Nötzli, Jehandad Khan, Andy Fingerhut, Clark Barrett, and Peter Athanas. 2018. P4pktgen: Automated test case generation for P4 programs. In *Proceedings of the Symposium on SDN Research (SOSR'18)*. Association for Computing Machinery, New York, NY, Article 5, 7 pages. https://doi.org/10.1145/3185467.3185497

[125] Vladimir Olteanu, Alexandru Agache, Andrei Voinescu, and Costin Raiciu. 2018. Stateless datacenter load-balancing with beamer. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI'18)*. USENIX Association, 125–139. Retrieved from https://www.usenix.org/conference/nsdi18/presentation/olteanu.

[126] OpenDataPlane. 2022. OpenDataPlane. Retrieved from https://opendataplane.org/.

[127] T. Osiński, H. Tarasiuk, L. Rajewski, and E. Kowalczyk. 2019. DPPx: A P4-based data plane programmability and exposure framework to enhance NFV services. In *Proceedings of the IEEE Conference on Network Softwarization (NetSoft'19)*. IEEE, 296–300. https://doi.org/10.1109/NETSOFT.2019.8806625

[128] P. Palagummi and K. M. Sivalingam. 2018. SMARTHO: A network initiated handover in NG-RAN using P4-based switches. In *Proceedings of the 14th International Conference on Network and Service Management (CNSM'18)*. IEEE, Rome, Italy, 338–342.

[129] F. Paolucci, F. Civerchia, A. Sgambelluri, A. Giorgetti, F. Cugini, and P. Castoldi. 2019. P4 edge node enabling stateful traffic engineering and cyber security. *IEEE/OSA J. Optic. Commun. Netw.* 11, 1 (2019), A84–A95.

[130] F. Paolucci, F. Cugini, and P. Castoldi. 2018. P4-based multi-layer traffic engineering encompassing cyber security. In *Proceedings of the Optical Fiber Communications Conference and Exposition (OFC'18)*. IEEE, 1–3.

[131] R. Parizotto, L. Castanheira, F. Bonetti, A. Santos, and A. Schaeffer-Filho. 2020. PRIME: Programming in-network modular extensions. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'20)*. IEEE, 1–9. https://doi.org/10.1109/NOMS47738.2020.9110355

[132] P. G. K. Patra, F. E. R. Cesen, J. S. Mejia, D. L. Feferman, L. Csikor, C. E. Rothenberg, and G. Pongracz. 2018. Toward a sweet spot of data plane programmability, portability, and performance: On the scalability of multi-architecture P4 pipelines. *IEEE J. Select. Areas Commun.* 36, 12 (2018), 2603–2611. https://doi.org/10.1109/JSAC.2018.2871288

[133] F. Pereira, N. Neves, and F. M. V. Ramos. 2017. Secure network monitoring using programmable data planes. In *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN'17)*. IEEE, 286–291. https://doi.org/10.1109/NFV-SDN.2017.8169867

[134] Larry Peterson. 2021. Retrieved from https://opennetworking.org/news-and-events/blog/openflow-catalyst-that-kickstarted-the-sdn-transformation/.

[135] M. Peuster, H. Karl, and S. van Rossem. 2016. MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments. In *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN'16)*. IEEE, 148–153. https://doi.org/10.1109/NFV-SDN.2016.7919490

[136] B. Pit-Claudel, Y. Desmouceaux, P. Pfister, M. Townsley, and T. Clausen. 2018. Stateless load-aware load balancing in P4. In *Proceedings of the IEEE 26th International Conference on Network Protocols (ICNP'18)*. IEEE, 418–423. https://doi.org/10.1109/ICNP.2018.00058

[137] M. Pizzutti and A. E. Schaeffer-Filho. 2019. Adaptive multipath routing based on hybrid data and control plane operation. In *Proceedings of the IEEE Conference on Computer Communications*. IEEE, 730–738. https://doi.org/10.1109/INFOCOM.2019.8737398

[138] T. Qu, R. Joshi, M. Chan, B. Leong, D. Guo, and Zhong Liu. 2019. SQR: In-network packet loss recovery from link failures for highly reliable datacenter networks. In *Proceedings of the IEEE 27th International Conference on Network Protocols (ICNP'19)*. 1–12.

[139] M. Rahali, J. Sanner, and G. Rubino. 2020. FEAL: A source routing Framework for Efficient Anomaly Localization. In *Proceedings of the IEEE International Conference on Communications (ICC'20)*. IEEE, 1–7. https://doi.org/10.1109/ICC40277.2020.9148725

[140] R. Ricart-Sanchez, P. Malagon, J. M. Alcaraz-Calero, and Q. Wang. 2018. Hardware-accelerated firewall for 5G mobile networks. In *Proceedings of the IEEE 26th International Conference on Network Protocols (ICNP'18)*. IEEE, 446–447. https://doi.org/10.1109/ICNP.2018.00066

[141] R. Ricart-Sanchez, P. Malagon, J. M. Alcaraz-Calero, and Q. Wang. 2019. NetFPGA-based firewall solution for 5G multi-tenant architectures. In *Proceedings of the IEEE International Conference on Edge Computing (EDGE'19)*. IEEE, 132–136. https://doi.org/10.1109/EDGE.2019.00037

[142] R. Ricart-Sanchez, P. Malagon, J. M. Alcaraz-Calero, and Q. Wang. 2019. P4-netfpga-based network slicing solution for 5G MEC architectures. In *Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'19)*. IEEE/ACM, Cambridge, UK, 1–2. https://doi.org/10.1109/ANCS.2019.8901889

[143] Jan Rüth, René Glebke, Klaus Wehrle, Vedad Causevic, and Sandra Hirche. 2018. Towards in-network industrial feedback control. In *Proceedings of the Morning Workshop on In-network Computing (NetCompute'18)*. Association for Computing Machinery, New York, NY, 14–19. https://doi.org/10.1145/3229591.3229592

[144] Y. Sakakibara, Y. Tokusashi, S. Morishima, and H. Matsutani. 2018. Accelerating blockchain transfer system using FPGA-based NIC. In *Proceedings of the IEEE International Conference on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom'18)*. IEEE, 171–178. https://doi.org/10.1109/BDCloud.2018.00037

[145] Omar Sami Oubbati, Mohammed Atiquzzaman, Tariq Ahamed Ahanger, and Atef Ibrahim. 2020. Softwarization of UAV networks: A survey of applications and future trends. *IEEE Access* 8 (2020), 98073–98125. https://doi.org/10.1109/ACCESS.2020.2994494

[146] J. Santiago da Silva, T. Stimpfling, T. Luinaud, B. Fradj, and B. Boughzala. 2018. One for all, all for one: A heterogeneous data plane for flexible P4 processing. In *Proceedings of the IEEE 26th International Conference on Network Protocols (ICNP'18)*. IEEE, Cambridge, United Kingdom, 440–441. https://doi.org/10.1109/ICNP.2018.00063

[147] Mateus Saquetti, Guilherme Bueno, Weverton Cordeiro, and José Rodrigo Azambuja. 2019. Hard virtualization of P4-based switches with VirtP4. In *Proceedings of the ACM SIGCOMM Conference Posters and Demos (SIGCOMM'19)*. Association for Computing Machinery, New York, NY, 80–81. https://doi.org/10.1145/3342280.3342314

[148] M. Saquetti, G. Bueno, W. Cordeiro, and J. R. Azambuja. 2019. VirtP4: An architecture for P4 virtualization. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW'19)*. IEEE, 75–78. https://doi.org/10.1109/IPDPSW.2019.00021

[149] Surbhi Saraswat, Vishal Agarwal, Hari Prabhat Gupta, Rahul Mishra, Ashish Gupta, and Tanima Dutta. 2019. Challenges and solutions in software defined networking: A survey. *J. Netw. Comput. Appl.* 141 (2019), 23–58. https://doi.org/10.1016/j.jnca.2019.04.020

[150] Cole Schlesinger, Michael Greenberg, and David Walker. 2014. Concurrent NetCore: From policies to pipelines. In *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming (ICFP'14)*. Association for Computing Machinery, New York, NY, 11–24. https://doi.org/10.1145/2628136.2628157

[151] Rinku Shah, Vikas Kumar, Mythili Vutukuru, and Purushottam Kulkarni. 2020. TurboEPC: Leveraging dataplane programmability to accelerate the mobile packet core. In *Proceedings of the Symposium on SDN Research (SOSR'20)*. Association for Computing Machinery, New York, NY, 83–95. https://doi.org/10.1145/3373360.3380839

[152] R. Shah, A. Shirke, A. Trehan, M. Vutukuru, and P. Kulkarni. 2018. Pcube: Primitives for network data plane programming. In *Proceedings of the IEEE 26th International Conference on Network Protocols (ICNP'18)*. IEEE, 430–435. https://doi.org/10.1109/ICNP.2018.00060

[153] S. Shahzad, E. Jung, J. Chung, and R. Kettimuthu. 2020. Enhanced explicit congestion notification (EECN) in TCP with P4 programming. In *Proceedings of the International Conference on Green and Human Information Technology (ICGHIT'20)*. IEEE, 35–40. https://doi.org/10.1109/ICGHIT49656.2020.00015

[154] Apoorv Shukla, Kevin Nico Hudemann, Artur Hecker, and Stefan Schmid. 2019. Runtime verification of P4 switches with reinforcement learning. In *Proceedings of the Workshop on Network Meets AI & ML (NetAI'19)*. Association for Computing Machinery, New York, NY, 1–7. https://doi.org/10.1145/3341216.3342206

[155] John Sonchack, Adam J. Aviv, Eric Keller, and Jonathan M. Smith. 2018. Turboflow: Information rich flow record generation on commodity switches. In *Proceedings of the 13th EuroSys Conference (EuroSys'18)*. Association for Computing Machinery, New York, NY, Article 11, 16 pages. https://doi.org/10.1145/3190508.3190558

[156] John Sonchack, Oliver Michel, Adam J. Aviv, Eric Keller, and Jonathan M. Smith. 2018. Scaling hardware accelerated network monitoring to concurrent and dynamic queries with *flow. In *Proceedings of the USENIX Annual Technical Conference (USENIX ATC'18)*. USENIX Association, 823–835. Retrieved from https://www.usenix.org/conference/atc18/presentation/sonchack.

[157] Haoyu Song. 2013. Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'13)*. Association for Computing Machinery, New York, NY, 127–132. https://doi.org/10.1145/2491185.2491190

[158] Radu Stoenescu, Dragos Dumitrescu, Matei Popovici, Lorina Negreanu, and Costin Raiciu. 2018. Debugging P4 programs with vera. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'18)*. Association for Computing Machinery, New York, NY, 518–532. https://doi.org/10.1145/3230543.3230548

[159] Dongeun Suh, Seokwon Jang, Sol Han, Sangheon Pack, and Xiaofei Wang. 2020. Flexible sampling-based in-band network telemetry in programmable data plane. *ICT Express* 6, 1 (2020), 62–65. https://doi.org/10.1016/j.icte.2019.08.005

[160] L. Tang, Q. Huang, and P. P. C. Lee. 2020. SpreadSketch: Toward invertible and network-wide detection of superspreaders. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'20)*. IEEE, 1608–1617. https://doi.org/10.1109/INFOCOM41043.2020.9155541

[161] K. Tokmakov, M. Sarker, J. Domaschka, and S. Wesner. 2019. A case for data centre traffic management on software programmable ethernet switches. In *Proceedings of the IEEE 8th International Conference on Cloud Networking (CloudNet'19)*. IEEE, 1–6. https://doi.org/10.1109/CloudNet47604.2019.9064114

[162] C. Trois, M. D. Del Fabro, L. C. E. de Bona, and M. Martinello. 2016. A survey on SDN programming languages: Toward a taxonomy. *IEEE Commun. Surveys Tutor.* 18, 4 (2016), 2687–2712. https://doi.org/10.1109/COMST.2016.2553778

[163] Belma Turkovic, Fernando Kuipers, Niels van Adrichem, and Koen Langendoen. 2018. Fast network congestion detection and avoidance using P4. In *Proceedings of the Workshop on Networking for Emerging Applications and Technologies (NEAT'18)*. Association for Computing Machinery, New York, NY, 45–51. https://doi.org/10.1145/3229574.3229581

[164] M. Uddin, S. Mukherjee, H. Chang, and T. V. Lakshman. 2017. SDN-based service automation for IoT. In *Proceedings of the IEEE 25th International Conference on Network Protocols (ICNP'17)*. IEEE, 1–10. https://doi.org/10.1109/ICNP.2017.8117555

[165] N. Varyani, Z. Zhang, and D. Dai. 2020. QROUTE: An efficient quality of service (QoS) routing scheme for software-defined overlay networks. *IEEE Access* 8 (2020), 104109–104126. https://doi.org/10.1109/ACCESS.2020.2995558

[166] J. Vestin, A. Kassler, D. Bhamare, K. Grinnemo, J. Andersson, and G. Pongracz. 2019. Programmable event detection for in-band network telemetry. In *Proceedings of the IEEE 8th International Conference on Cloud Networking (CloudNet'19)*. IEEE, 1–6. https://doi.org/10.1109/CloudNet47604.2019.9064137

[167] J. Vestin, A. Kassler, and J. Åkerberg. 2018. FastReact: In-network control and caching for industrial control networks using programmable data planes. In *Proceedings of the IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA'18)*, Vol. 1. IEEE, 219–226. https://doi.org/10.1109/ETFA.2018.8502456

[168] Marcos A. M. Vieira, Matheus S. Castanho, Racyus D. G. Pacífico, Elerson R. S. Santos, Eduardo P. M. Câmara Júnior, and Luiz F. M. Vieira. 2020. Fast packet processing with EBPF and XDP: Concepts, code, challenges, and applications. *ACM Comput. Surv.* 53, 1, Article 16 (Feb. 2020), 36 pages. https://doi.org/10.1145/3371038

[169] S. Wang, Y. Chen, J. Li, H. Hu, J. Tsai, and Y. Lin. 2019. A bandwidth-efficient INT system for tracking the rules matched by the packets of a flow. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'19)*. IEEE, 1–6. https://doi.org/10.1109/GLOBECOM38437.2019.9013581

[170] Weitao Wang, Praveen Tammana, Ang Chen, and T. S. Eugene Ng. 2020. Grasp the root causes in the data plane: Diagnosing latency problems with SpiderMon. In *Proceedings of the Symposium on SDN Research (SOSR'20)*. Association for Computing Machinery, New York, NY, 55–61. https://doi.org/10.1145/3373360.3380835

[171] J. Woodruff, M. Ramanujam, and N. Zilberman. 2019. P4DNS: In-network DNS. In *Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'19)*. ACM/IEEE, 1–6. https://doi.org/10.1109/ANCS.2019.8901896

[172] X. Wu, P. Li, T. Miskell, L. Wang, Y. Luo, and X. Jiang. 2019. Ripple: An efficient runtime reconfigurable P4 data plane for multicore systems. In *Proceedings of the International Conference on Networking and Network Applications (NaNA'19)*. IEEE, 142–148. https://doi.org/10.1109/NaNA.2019.00034

[173] Zhaowei Xi, Yu Zhou, Dai Zhang, Jinqiu Wang, Sun Chen, Yangyang Wang, Xinrui Li, HaoMing Wang, and Jianping Wu. 2019. HyperGen: High-performance flexible packet generator using programmable switching ASIC. In *Proceedings of the ACM SIGCOMM Conference Posters and Demos (SIGCOMM'19)*. Association for Computing Machinery, New York, NY, 42–44. https://doi.org/10.1145/3342280.3342301

[174] Zhaoyue Xia, Jun Bi, Yu Zhou, and Cheng Zhang. 2018. KeySight: A scalable troubleshooting platform based on network telemetry. In *Proceedings of the Symposium on SDN Research (SOSR'18)*. Association for Computing Machinery, New York, NY, Article 20, 2 pages. https://doi.org/10.1145/3185467.3190787

[175] J. Xie, C. Qian, D. Guo, X. Li, S. Shi, and H. Chen. 2019. Efficient data placement and retrieval services in edge computing. In *Proceedings of the IEEE 39th International Conference on Distributed Computing Systems (ICDCS'19)*. IEEE, 1029–1039. https://doi.org/10.1109/ICDCS.2019.00106

[176] J. Xie, C. Qian, D. Guo, M. Wang, S. Shi, and H. Chen. 2019. Efficient indexing mechanism for unstructured data sharing systems in edge computing. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'19)*. IEEE, 820–828. https://doi.org/10.1109/INFOCOM.2019.8737617

[177] Tong Yang, Jie Jiang, Peng Liu, Qun Huang, Junzhi Gong, Yang Zhou, Rui Miao, Xiaoming Li, and Steve Uhlig. 2018. Elastic sketch: Adaptive and fast network-wide measurements. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'18)*. Association for Computing Machinery, 561–575. https://doi.org/10.1145/3230543.3230544

[178] Abbas Yazdinejad, Reza M. Parizi, Ali Bohlooli, Ali Dehghantanha, and Kim-Kwang Raymond Choo. 2020. A high-performance framework for a network programmable packet processor using P4 and FPGA. *J. Netw. Comput. Appl.* 156 (2020), 102564. https://doi.org/10.1016/j.jnca.2020.102564

[179] Abbas Yazdinejad, Reza M. Parizi, Ali Dehghantanha, and Kim-Kwang Raymond Choo. 2020. P4-to-blockchain: A secure blockchain-enabled packet parser for software defined networking. *Comput. Secur.* 88 (2020), 101629. https://doi.org/10.1016/j.cose.2019.101629

[180] J. Ye, C. Chen, and Y. Huang Chu. 2018. A weighted ECMP load balancing scheme for data centers using P4 switches. In *Proceedings of the IEEE 7th International Conference on Cloud Networking (CloudNet'18)*. IEEE, 1–4. https://doi.org/10.1109/CloudNet.2018.8549549

[181] Farnaz Yousefi, Anubhavnidhi Abhashkumar, Kausik Subramanian, Kartik Hans, Soudeh Ghorbani, and Aditya Akella. 2020. Liveness verification of stateful network functions. In *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI'20)*. USENIX Association, 257–272. Retrieved from https://www.usenix.org/conference/nsdi20/presentation/yousefi.

[182] E. O. Zaballa, D. Franco, Z. Zhou, and M. S. Berger. 2020. P4Knocking: Offloading host-based firewall functionalities to the network. In *Proceedings of the 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN'20)*. IEEE, 7–12. https://doi.org/10.1109/ICIN48450.2020.9059298

[183] C. Zhang, J. Bi, Y. Zhou, A. B. Dogar, and J. Wu. 2017. HyperV: A high performance hypervisor for virtualization of the programmable data plane. In *Proceedings of the 26th International Conference on Computer Communication and Networks (ICCCN'17)*. IEEE, 1–9. https://doi.org/10.1109/ICCCN.2017.8038396

[184] Cheng Zhang, Jun Bi, Yu Zhou, Abdul Basit Dogar, and Jianping Wu. 2017. MPVisor: A modular programmable data plane hypervisor. In *Proceedings of the Symposium on SDN Research (SOSR'17)*. Association for Computing Machinery, New York, NY, 179–180. https://doi.org/10.1145/3050220.3060600

[185] C. Zhang, J. Bi, Y. Zhou, and J. Wu. 2019. HyperVDP: High-performance virtualization of the programmable data plane. *IEEE J. Selec. Areas Commun.* 37, 3 (2019), 556–569. https://doi.org/10.1109/JSAC.2019.2894308

[186] C. Zhang, J. Bi, Y. Zhou, K. Zhang, and Z. Ma. 2018. B-Cache: A behavior-level caching framework for the programmable data plane. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC'18)*. IEEE, 00084–00090. https://doi.org/10.1109/ISCC.2018.8538450

[187] D. Zhang, X. Chen, Q. Huang, X. Hong, C. Wu, H. Zhou, Y. Yang, H. Liu, and Y. Chen. 2019. P4SC: A high performance and flexible framework for service function chain. *IEEE Access* 7 (2019), 160982–160997. https://doi.org/10.1109/ACCESS.2019.2950446

[188] Menghao Zhang, G. Li, Shicheng Wang, Chang Liu, Ang Chen, Hongxin Hu, Guofei Gu, Qi Li, Mingwei Xu, and Jianping Wu. 2020. Poseidon: Mitigating volumetric DDoS attacks with programmable switches. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'20)*. The Internet Society.

[189] Z. Zhao, X. Shi, X. Yin, Z. Wang, and Q. Li. 2019. HashFlow for better flow record collection. In *Proceedings of the IEEE 39th International Conference on Distributed Computing Systems (ICDCS'19)*. IEEE, 1416–1425. https://doi.org/10.1109/ICDCS.2019.00141

[190] Yu Zhou, Jun Bi, Yunsenxiao Lin, Yangyang Wang, Dai Zhang, Zhaowei Xi, Jiamin Cao, and Chen Sun. 2019. P4Tester: Efficient runtime rule fault detection for programmable data planes. In *Proceedings of the International Symposium on Quality of Service (IWQoS'19)*. Association for Computing Machinery, New York, NY, Article 5, 10 pages. https://doi.org/10.1145/3326285.3329040

[191] Y. Zhou, J. Bi, T. Yang, K. Gao, C. Zhang, J. Cao, and Y. Wang. 2018. KeySight: Troubleshooting programmable switches via scalable high-coverage behavior tracking. In *Proceedings of the IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 291–301. https://doi.org/10.1109/ICNP.2018.00045

[192] Y. Zhou, J. Bi, C. Zhang, B. Liu, Z. Li, Y. Wang, and M. Yu. 2019. P4DB: On-the-fly debugging for programmable data planes. *IEEE/ACM Trans. Netw.* 27, 4 (2019), 1714–1727. https://doi.org/10.1109/TNET.2019.2927110

[193] Shaolong Zhu, Du Chen, Meiyi Yang, and Xuening Shang. 2021. Dynamic multi-path and multi-protocol encrypted communication mechanism. In *Proceedings of the IEEE 13th International Conference on Computer Research and Development (ICCRD'21)*. IEEE, 58–62. https://doi.org/10.1109/ICCRD51685.2021.9386407

Received 6 July 2021; revised 17 July 2022; accepted 8 August 2022