Unlocking the Potential of P4: An Examination of General Performance Gains (extended version)

Rúben Silva, Bruno Neiva, Pedro António, Marco Pereira, Rúben Cerqueira
Universidade do Minho
Braga, Portugal

Paulo Carvalho, Solange Rito Lima
Centro Algoritmi, Univ. do Minho
Braga, Portugal

João Marco C. Silva
INESC TEC, Universidade do Minho
Braga, Portugal

ABSTRACT

This paper presents a survey of recent proposals for general performance enhancement in the P4 programming language. The survey covers a wide range of proposals, including those related to packet processing, buffer management, and load balancing. We provide an overview of the main ideas and contributions of each proposal and evaluate their potential impact on performance according to the tests provided. Our analysis shows that there are several promising proposals that have the potential to improve the performance of P4-based systems significantly. However, more work is needed to fully realize these potential gains and to understand the trade-offs between different proposals. To conclude, the gaps in current research as well as possible directions that P4 could take are offered. Through these contributions, the survey aims to serve as a valuable resource for researchers and practitioners interested in P4 performance optimization.

KEYWORDS

P4, Network Performance, Hardware Programming, Buffer Management, Load Balancing, Offloading, Routing and Packet Forwarding, Video Playing, Software Defined Networks, Flow Scalability, Rule Responsiveness, Pipeline Mapping

ACM Reference Format:
Rúben Silva, Bruno Neiva, Pedro António, Marco Pereira, Rúben Cerqueira, Paulo Carvalho, Solange Rito Lima, and João Marco C. Silva. 2023. Unlocking the Potential of P4: An Examination of General Performance Gains (extended version). In *Proceedings of* . ACM, New York, NY, USA, 13 pages.

1 INTRODUCTION

From more than two decades now, networking technology has become a mainstay in every household and evolved to the point where it can now be accessed from each of our pockets [44], but it also has deeply rooted core concepts that are very slow to change [10]. These barriers have been something that was accepted for decades as a necessity, but have started to fall with the advent of Software Defined Networks [21, 55], as the control plane is finally decoupled from the data plane. The control plane is responsible for managing how data packets are forwarded, where the data plane performs the actual forwarding. SDNs allow for programmable switches where the control plane actually resides outside of the device. It can be managed to a fine degree where it frees the network from conventional routing algorithms and forwarding rules.

Openflow [41, 49], maintained by the ONF (Open Networking Foundation), is the main protocol used in SDNs, having experienced widely adoption ever since it was made public in 2011. As powerful as OpenFlow might be, it is still bound by the manufacturer's design for its data plane functions. This means it remains limited to a set of protocols that was made available, as well as the forwarding functions that were implemented when the device was created. The design cycle for switch ASICs can span several years, meaning that changes to any of these features could take just as long to be fully implemented [11].

While SDNs certainly grant a much larger degree of freedom in how network devices behave by making the control plane programmable, there is now a demand for a data plane that can be similarly configured to a fine degree [13].

Naturally, this led to the development of technology that would allow for programmable forwarding. The main driving force in this regard is known as Programming Protocol-independent Packet Processors (P4 for short), and confers the ability for the programmer to fully define the behaviour of a P4-programmable switch [13]. Despite being a relative newcomer to the networking scene (first appearing in 2013), it has since gained a large amount of acceptance by many established companies such as Intel [53], along with other companies like Barefoot Networks [2], Juniper [26] and Cisco Systems [4].

With P4-programmable hardware being more and more available as well as being affordable, companies are looking to expand and adopt the technology, but may have some questions as to how it can benefit and improve their usual workloads. Given the limited literature available on these topics, this paper will focus on how P4 improves upon the traditional architecture and SDN approaches, as well as build upon the work that has been done in cataloguing and referencing P4's advances in this regard.

For the writing of this paper, over 100 papers on the P4 programming language and traditional hardware approaches for networking excellence were assessed. Through this survey, we identified and selected 40 of the most important papers that compared the use of P4 and traditional hardware in networking and offered significant proof that P4 can be a superior option when compared with traditional hardware for a large amount of use cases. These papers cover a wide range of topics related to P4, including its flexibility, programmability, efficiency, performance, and security compared to traditional hardware. Our selection of these papers represents a comprehensive overview of the benefits of using P4 over traditional hardware in networking.

By providing a comprehensive overview of the advantages of using P4 over traditional hardware, we hope to further encourage the adoption of P4 in the industry and to facilitate continued research and development in this important and rapidly growing area.

Although this paper focuses only on the advancements made in one year, the number of articles and quality of results produced during this time was too high for a new survey to not be a requirement. Due to P4 being a very new and still developing language, every month at least two or three relevant papers related to performance are published. Their content may be focused on offloading some of the work to the data plane using P4 or resort to the usage of P4 to be able to supply a new feature that will provide flexibility in areas like buffer management as this paper explores further along. The purpose of this paper is to light the most recent advancements when it comes to being able to transmit information as quickly and efficiently as possible.

The paper is structured in the following way: it begins with an analysis of previously published surveys on p4, followed by a comparison between P4-Enabled Switches and Traditional Fixed-Function Switches, before getting to the main section, titled "Results". This chapter includes sub-sections on every area of performance in which the papers have been grouped. Each of these sub-sections are further broken down and cover different aspects of the conducted research.

Lastly, under Future Directions, potential future research and developments in P4 are analysed and discussed.

The areas of influence in which this paper focuses have been selected by doing an analysis over all the gathered papers. The purpose of this analysis was to not only group every paper found according to their respective area, but also assess how much of an impact it made in the industry. Some areas exhibited a decidedly larger leap over what was available the prior year. The final selection stands as: Buffer Management, Latency, Load Balancing, Offloading, Routing, Packet Forwarding, Video Playing SDN, Testing, Flow Scalability and Rule Responsiveness, as well as Pipeline Mapping.

2 BRIEF SUMMARY OF PREVIOUS P4 SURVEYS

Surveys can be used to gather information on how the language is being used, what features are important to users, and what types of applications are being developed with P4, as well as any challenges or limitations users face. These surveys provide valuable insights into the P4 ecosystem and guide its development.

Altangerel and M'at'e[9] discuss several contributions to P4 evolution related to In-band Network Telemetry (INT) and In-Network Computing. They cover optimization of telemetry data by reducing the number of monitored network elements, not including the INT header in all flow packets, sampling packets for monitoring, and using thresholds to identify in-band monitored flow. Additionally, the survey also states that P4-based INT does not support sampling and adding an INT header to all incoming packets can cause high network overhead in a large scale network. Through an intelligent trigger mechanism for fault detection that offloads event detection from monitoring servers to in-network P4 applications, it succeeds in reducing network overhead and load on monitoring servers, and combining monitoring with machine learning methods referred to as knowledge plane or knowledge-defined network. As a final point in INT, it states that the applications with regard to congestion control and advanced routing based on INT should be efficient, and proposes different INT characteristics such as link load based, rate-based and queuing and processing delay based congestion control.

Altangerel and M'at'e[9] also discuss In-Network Computing, and how the advent of flexible programmable networking devices has made it possible to perform additional computation on network devices without reducing packet processing rates. It presents the concept of data aggregation functionality being performed on the network path, reducing traffic load and making it simpler and more efficient for simple arithmetic logic operations. A prototype solution called DAIET is also referenced, which is built in P4 and is an in-network aggregation solution for machine learning and graph analytics applications, while being generic enough to be used in various partition/aggregation data center applications.

Furthermore, the paper discusses the possibility of offloading most of the tasks on a control plane to a data plane and how it can accelerate the control plane, while mentioning that some tasks are not optimal to run on data plane, as they require a lot of resources.

The idea of creating in-network caching is similarly explored, detailing how the programmable data plane provides new opportunities to create in-network caching, in an effort to further reduce latency. A key-value store data structure is used to build the database in the cache because it is generic and widely used by applications.

Lastly, it explains how the advent of a programmable data plane has made it possible to develop a customizable load balancing mechanism on the data plane, and presents Hop-by-hop Utilization-aware Load balancing Architecture (HULA) [28]. HULA was programmed in P4 as the first load-balancing mechanism explicitly designed for the programmable switch architecture, and it is scalable and congestion-aware. This paper doesn't contain any testing references, which is one of the problems we try to address, by presenting the testing results of optimizations researched by comparing those results with the current state-of-the-art results.

Kfoury et al. [30]'s study provides an overview of P4 programmable data plane switches, its potential use cases and its current state of development. The authors present a taxonomy of P4 switches, which includes different types of P4 switches based on their architecture and the level of programmability. They also discuss various applications of P4 switches and cover the use of P4 in different domains such as data centers and edge networks. Additionally, the paper addresses the challenges that need to be addressed for the widespread adoption of P4 switches, such as scalability, security, and compatibility with existing networks. The authors also provide an overview of the future trends in the field of P4 switches, including the integration of P4 with other technologies such as Network Function Virtualization (NFV). The paper then compares P4 with other programmable data plane technologies such as OpenFlow[49] and Netronome[3], which provides a clear understanding of the advantages and disadvantages of P4.

In the following section, we will be detailing each of the segments that constitute the paper, as well as the structure that was followed for their respective ordering.

3 COMPARISON TO TRADITIONAL FIXED-FUNCTION SWITCHES

Through being a high-level programming language, P4 allows network engineers to create custom packet processing pipelines in a way that is similar to how software developers write code. This is both intuitive, as well as efficient, as it is not a disruption to the engineer's workflow.

P4 enables network engineers to define and manipulate packet headers in a way that is abstracted from the underlying hardware, allowing code to be compiled and run on a variety of different types of hardware, including switches, routers, and network interface cards (NICs). Network engineers may also create custom match-action tables that can be used to perform a wide range of tasks such as load balancing, traffic management, and security functions while also enabling the creation of custom header fields, adding new functionality to existing network devices [46].

Flexibility is perhaps P4's largest advantage. P4 code can be written to run on a variety of different types of hardware, allowing network engineers to create custom solutions that are tailored to their specific needs, rather than being limited to the pre-defined functionality of fixed-function devices.

Fixed-function devices, on the other hand, are pre-configured to perform specific functions and cannot be programmed or configured to perform different tasks. They are also usually less expensive than programmable devices. Examples of fixed-function devices include basic network switches and routers, as well as specialized appliances such as firewalls and intrusion detection systems.

Overall, P4 provides a high level of programmability and flexibility for network devices, allowing network engineers to create custom packet processing pipelines and add new functionality to existing devices. Fixed-function devices, on the other hand, while simpler and faster, are much less flexible and can only perform pre-defined functions.

4 REFERENCE TABLE

Paper	Area	Parameters
[47]	Buffer Management	$\overline{RTT}$, FCT, $\overline{F}$, $\overline{\rho}$
[33]	Buffer Management	$\overline{RTT}$
[34]	Buffer Management	$\overline{RTT}$
[39]	Latency	Latency
[38]	Latency	Latency
[17]	Latency	Latency
[14]	Load Balancing, Routing	$\overline{FCT}$
[40]	Load Balancing	SRAM
[32]	Load Balancing	Throughput
[20]	Offloading	Throughput
[57]	Offloading	Update Time
[7]	Offloading	Pacekt Loss
[40]	Routing	Packet Loss
[36]	Packet Forwarding	$\overline{FCT}$, Jitter
[56]	Testing	Throughput, Error rate
[25]	Video Playing SDN	Response Time
[22]	Flow Scalability, Rule Responsiveness	Response Time
[37]	Pipeline Mapping	Portability
[16]	Congestion Control	Congestion Control
[19]	Congestion Control	AQM, TCP CC
[24]	Congestion Control	FCT, TDP
[35]	Congestion Control	Memory Optimiziation
[8]	Security	Security
[50]	Security	Security
[31]	INT	INT
[48]	INT	INT
[27]	Machine-Learning	Traffic Volume
[43]	Machine-Learning	QoS
[52]	Quality of Service	Latency
[15]	Quality of Service	Bandwidth Management
[54]	Quality of Service	QoS
[45]	Serverless	Response Time
[42]	Visual Cloud Computing	Packet Loss

5 RESULTS

Over this chapter, we will be detailing the different areas where we feel that P4 has seen the largest improvement over previously published works. Where applicable, a comparison will be drawn between the described solution and traditional hardware as well as Software Defined Networks. Each section is the result of the advancement of various scientific papers in the area, so some of the improvements will need to implement several technologies in order to achieve the proposed results.

5.1 Buffer Management

Buffer management is the process of managing the memory buffer in a network device to ensure efficient use of resources and to prevent congestion [1]. Various topics can be grouped into this category, one of them being the setting of the buffer size. The papers referenced and explored in this section try to dynamically calculate the router's buffer size and manage the queue by basing it on the network's conditions. The conditions are passively measured by tapping on the router's ports, the traffic is then forwarded to a programmable switch. This switch tracks the number of persistent over time flows and computes the respective RTTs. With this information, the router's buffer size is then appropriately adapted.

5.1.1 Dynamic Router's Buffer Size. For a long time, the buffer size of the router was configured as being equal to the capacity of the port multiplied by the average round trip time, this represents the Bandwidth Delay Product (BDP). Eventually, it was demonstrated that the buffer size could be reduced to $BDP/\sqrt{N}$, with N being the number of persistent over time flows traversing the port.

In order to be able to dynamically calculate the buffer size[29], a tap is inserted in the data link, doing passive traffic collection to a P4 switch, allowing for metrics computation and the buffer size calculation to take place, in the data plane and control plane respectively. This switch uses a management link to be able to update the legacy router's buffer size.

The first scenario created for testing uses the default buffer size on the router, while the second uses the programmable P4 switch to measure and modify the buffer size of the router. Apart from the difference described, both scenarios have identical conditions in terms of environment, as well as workload.

Three major experiments were made, the first one evaluates the performance of the buffer when using multiple long flows with some propagation delays, the second evaluates the performance of short flows when having to share the bottleneck with long flows and finally, the third one evaluates the performance when using long flows with different propagation delays.

Experiment 1. The first experiment compares the solution with buffer modification to the solution without buffer modification. The terms of comparison are as follows: link utilization percentage, usage fairness index and average round trip time. Variables for artificial delay were added, as well as a Congestion Control Algorithm (CCA) and a number of long flows.

The results obtained in the first experiment show that regardless of the CCA used, the solution with buffer modification produces better and more consistent results for both the link usage fairness index and the average round trip time. The link utilization is slightly better without buffer modification as it will always be 100% in usage due to the initial given value.

Experiment 2. The second experiment compares the Cumulative Distribution Functions (CDF) for each of the CCAs used considering both RTT and FCT with and without the buffer modification. The results confirm that, for both CCAs, the CDF will reach above 0.9 before 50 milliseconds for RTT and before 1 second for FCT, meaning that more than 90% of all the collected values for RTT and FCT are smaller than the values referred. On the other hand, without buffer modification, the CDFs reach 0.9 closer to 200 milliseconds for RTT and closer to 2.5 seconds for the FCT [].

Experiment 3. This experiment consists of 100 long flows, divided into four groups of 25 flows each. Each group starts three minutes after the other. The improvements here can be specifically seen in the RTT of each group of flows, which is close to the propagation delay. The increase in packet loss when new flows join produces a big impact. This occurs because the buffer size is being adjusted while the buffer is already holding packets.

5.1.2 Active Queue Management (AQM). AQM is a technique used to control the amount of data queued in the buffers of network devices, such as routers and switches. The goal of AQM is to prevent network congestion and to reduce the delay and packet loss that can occur when buffers become full [18]. Having the router's buffer size fixed represents a compromise between latency and bandwidth: a larger buffer causes Bufferbloat, which is a phenomenon that happens when the buffer is constantly filled, leading to latency-related problems; a smaller buffer causes bandwidth problems due to too many packet drops.

In order to solve these problems, several AQM algorithms were implemented to improve the control of queues, namely Random Early Detection (RED), Controlled Delay (CoDel), and Proportional Integral Controller Enhanced (PIE). RED associates a large average queue length with congestion and notifies end hosts by randomly dropping packets based on a drop probability. CoDel uses four stateful variables - dropping, count, last count, and drop next - to determine when received packets will be dropped, using a 5ms threshold for the observed queuing delay. PIE calculates a drop probability, similar to CoDel, based on the queue latency, and using the drop probability randomly drops packets at enqueuing.

The works [34] and [33] provide more information and comparative results on the implemented AQM algorithms.

5.2 Latency

Latency, defined as the time required for a packet to travel from its source to its destination, is a significant factor impacting performance. The implementation of P4 technology has enabled the creation of more efficient devices, tailored to the specific conditions of the network, thus reducing the overall latency of packets [23].

In this area, the works Mafioletti et al.[38] and Mafioletti et al.[39] provide approaches to reduce latency in PONs (Passive Optical Networks), which are the most cost-effective approach for delivering broadband. The works try to upgrade PONs upstream scheduling when using a DBA (Dynamic Bandwidth Assignment) mechanism, which introduces a very high latency due to working with a report/grant mechanism.

5.2.1 Splitting DBA allocation into two parts. Mafioletti et al.[38] proposes splitting DBA allocation into two parts. The first part works as a normal DBA, however, part of the bandwidth map should be left unallocated. The second part of this proposal is called Fast Intercept and runs in P4 hardware, spoofing upstream low latency requests until the next bandwidth map arrives. Then all requests are sent together, saving latency.

The results obtained during testing show how the proposed method reduces significantly the total time. The measured latency using a normal PON was 113.9ms, whereas the Fast Intercept mechanism results achieved a combined latency of 3.17 microseconds, showing a much shorter latency time when compared to the normal approach.

5.2.2 Dual-DBA allocation using a P4-enabled scheduler. Mafioletti et al.[39] proposes a solution to provide better latency when the low latency requirement comes from the application. For this, a P4 embedded Network Function (eNF) on a Netronome SmartNIC was created. Through the processing of BWMAPs and DBRus data structures, it analyses the incoming data on the downstream requests and updates the stored data accordingly.

While a standard DBA mechanism achieved an average latency of 374.5 μs and 418.5 μs, for an OEM and a virtual implementation of a PON DBA, respectively, the average latency obtained when splitting the phases into CPU and SmartNIC was 237.5 μs. This represents a latency reduction of 37% and 43% respectively, making the programmable approach a much better solution when performance is critical.

The Tofino P4 programmable ASIC hardware's latency performance is examined in this study [17]. In order to forecast latency based on parser, deparser, and control block functions, the authors conducted a profile study and constructed a regression model. PLP was the most accurate machine learning method for forecasting latency across these blocks at both 10G and 100G speeds, according to their tests of other techniques. The PLP model proved to be useful in real-world circumstances by achieving excellent accuracy (over 98%) even for P4 features that had not yet been seen. The study also revealed an interesting detail - the observed non-linear latency behavior at 10G speeds seems to be caused by the switch's gearbox and not the P4 program itself. This emphasizes how crucial it is to take the network infrastructure as a whole into account when assessing latency performance. The 100G findings, on the other hand, showed a linear trend, indicating a higher fit for applications requiring delay.

5.3 Load Balancing

Load balancing is a crucial technique for enhancing the performance of a system by efficiently distributing network traffic across multiple resources. The primary goal of load balancing is to prevent the overloading of any single resource, which can lead to delays or system failures [5].

In traditional networks, load balancing is typically achieved through software solutions. However, with the advent of SDNs, the control plane and data plane are separated, making it easier to program the network and providing greater flexibility in managing network resources. This separation also enables more granular control over the network, allowing for dynamic and efficient load balancing.

In SDN, load balancing is achieved through the use of SDN controllers, which are responsible for managing the flow of traffic through the network. Administrators can easily monitor network traffic and make adjustments to the load-balancing configuration as needed to ensure that the network remains stable and responsive. This, in turn, leads to improved system performance and reduced risk of network congestion or downtime [51].

In this area, the work in [14] proposes a routing method for multi-rooted tree topologies, mainly fat tree topologies.

5.3.1 Congestion-aware multi-path label switching (CAMP). The proposed load balancing algorithm in [14] mainly focuses on dealing with elephant flows (substantial continuous flows). The edge switches in CAMP run an elephant flow detection algorithm, which allows for routing the received elephant flows toward the least congested path. A P4 program describes the processing logic for switches in CAMP.

The testing compares three load balancing methods, the proposed algorithm CAMP, and two state-of-the-art algorithms - Equal-Cost Multi-Path routing (ECMP) and HULA. When comparing the average FCT (Flow Completion Time), CAMP provides 35% worse times than ECMP on mice flows (short flows). However, on both combined and elephant flows, the main concern of the work, a large discrepancy can be observed: both HULA and CAMP obtain significantly better times (50% the time of ECMP). CAMP provides 15% smaller times when compared to HULA regarding mice flows.

Based on paper [40], and from a P4 standpoint, SilkRoad stands out as a ground-breaking invention that replaces conventional Server Load Balancers (SLBs) while significantly lowering SRAM consumption through the use of programmable switching ASICs. Large-scale load balancing paradigms have changed significantly as a result of its capacity to support numerous connections and preserve Packet Completion Control (PCC) in the face of dynamic modifications to Destination IP (DIP) pools. In addition, SilkRoad's ability to do complete line-rate load balancing with little processing delay raises the possibility of improved performance isolation in comparison to traditional SLBs. However, there are significant drawbacks to traditional methods like Chord, which are based on Distributed Hash Tables (DHTs), including longer routing pathways and more storage cost. With GRED, these drawbacks are addressed, and a viable solution is shown that maintains better load balancing performance while using less memory and shorter routing pathways. Together, these developments highlight how load balancing technologies are developing, with stateless implementations and programmable ASIC-based systems providing interesting directions for further study and exploration.

The researchers in the subsequent study [32] used P4, a data plane programming language, to develop the DLBA method for testing. They used Mininet and OpenFlow switches to construct an experimental network architecture. The P4 software was compiled and interpreted using P4c and PI, and the SDN controller utilized was ONOS. GNUPLOT was used to show the results, and Wireshark was used to confirm them. The performance of the P4-based DLBA was significantly better than that of the current techniques. The throughput increased dramatically, maintaining a range of 550 to 840 Megabytes per second. DLBA had significantly lower packet loss rates (3.1%) than WRR and Hash, which were 14.48% and 17.42%, respectively. Ultimately, DLBA was able to maintain a bigger congestion window size, suggesting that it can manage network traffic more effectively and perhaps resulting in increased throughput and decreased congestion.

5.4 Offloading

Offloading is a rather extensive area, covering different means of processing/forwarding data. P4 devices' flexibility due to programmability now enables offloading into networking devices in an efficient and timely manner [47].

Security assurance through an IDS has been a staple for many years, but with increased density of traffic, the practice quickly becomes a large deterrent to a no congested network [6]. NFVs were then used to try to improve the situation, which they remarkably did, though this approach is still not ideal as NFVs will also reach a point of congestion due to the high overhead of having to identify unmarked traffic.

This is exactly the area where P4 aims to make a difference. Through faster feature extraction at the hardware level and subsequent usage of a detection application, P4 manages to produce much better results than just through the use of a state-of-the-art IDS, such as Suricata [20]. By leaving the feature extraction at the hardware level, it is now possible to send much fewer data to be assessed through the network, thus significantly decreasing network congestion. Data throughput with a fixed function device will cap at around 80 Tbps, whereas P4 devices are able to process close to 100 Tbps, even during the most processing-intensive tasks, such as metadata extraction. This represents a 25% increase in throughput, which is often the bottleneck.

Regarding the accuracy of the ML model both for Suricata IDS and P4 offloading approach, by being able to be continuously updated and deployed, the model used by the P4 approach naturally vastly outshines a fixed function device that relies on updates, such as Suricata. This results in not only being able to detect attacks that are not in the current version of software implemented in the fixed function device but also have a 5% higher rate of True Positives/Negatives as well as a 4% lower rate of False Positives/Negatives for the tested environment and workload [].

Another important example of Offloading for reducing network congestion is P4Update[57]. One of the most intensive tasks networking devices usually undertake has to do with network updates, which traditionally are all assessed at the control plane. With P4 making the data plane programmable, one of the major applications of this kind of technology is in offloading the network updates to hardware. Results show that P4Update will generally be a faster alternative to the state-of-the-art network update software by a margin of over 25%. The testing runs very different flows, demonstrating that this approach is overall faster without being completely outshone by a traditional offering in any given scenario.

In other hand, TOS-P4, a unique approach for P4-assisted task offloading, improves task offloading in Fog-based IoT networks. When compared to traditional techniques, it reduces task waiting time by almost six times by intelligent allocation based on Fog server characteristics. In addition, TOS-P4 constantly adjusts to changes in server load, enhancing offloading techniques for enhanced network efficiency [7].

From many others, a distinct example of Offloading in P4 is the User Plane Function offloading to enhance 5G Mobile Edge Computing, which while currently still in development, offers great insights as to how offloading can be used for new technologies [28].

5.5 Routing

Routing refers to the process of determining the path that network traffic should take to reach its destination. This process is typically performed by routers, which are devices that connect different networks together and forward traffic between them. Routing can be done using various protocols, such as the Internet Protocol (IP) and the Border Gateway Protocol (BGP), and has received a large amount of academic attention with regard to its implementation in fixed-function devices. Given the freedom that P4 confers, the possibility of programming new routing protocols becomes a reality, and routing is certainly one of the most researched areas in P4.

Many breakthroughs have been made with regard to routing using P4. By allowing for any given header to be defined at a hardware level, devices are offered unprecedented speeds when it comes to assessing and forwarding traffic. This in turn leads to advancements in areas where speed has been a marked bottleneck to the industry, as is the case of datacenter routing. When the load on the network climbs to over 50%, even the most widely used protocol for datacenter routing, ECMP, struggles and starts taking a much longer amount of time to complete its flows.

A P4 alternative had been found back in 2016 with HULA [28], which greatly improved flow completion times at a higher network load, and is now surpassed by CAMP [14]. CAMP is a two-phased system, which starts with an ECMP-like approach, only now done at a hardware level. If an elephant flow is detected, then it is routed with congestion awareness, thus improving upon earlier offerings, as was previously shown. As the hash collision probability climbs, ECMP degrades at a much faster rate than both P4 approaches, and overall CAMP is slightly faster than HULA, proving that there is still room for growth and more performing approaches.

The authors of the following paper [40], describe that to solve network issues, several rerouting techniques make use of customizable data planes. They carried out a thorough analysis to gain a better understanding of various routing options, and the following findings were obtained: Rerouting traffic by altering packet headers, whereas P4Neighbor incorporates other pathways into packet headers. Through caching and parallel search techniques, SQR and PURR streamline port selection and avoid packet loss, thereby optimizing Fast Reroute (FRR). Programmable switches are used by proactive solutions such as P4-Protect and reactive mechanisms such as Wharf to mitigate faults and protect flow. Proactive load balancing is provided by Silkroad, while reactive load balancing is provided by W-ECMP. In the meanwhile, topology changes must be manually implemented by GRED and UELB as they depend on the control plane for topology-aware forwarding.

5.6 Packet Forwarding

P4 provides programmers with a detailed capacity for tailoring devices' functions, including multilayer policies for packet forwarding. This has leveraged comprehensive research on how packet forwarding performance can be enhanced with the support of P4. Typically, recent proposals take advantage of the flexibility of defining optimised processing for packets that match an entry in the match-action table at the data plane level.

An important field addressed through packet forwarding in P4 is early detection and answer to microbursts. Notably, the Real-Time In-Network Microburst Mitigation on Programmable Switch (RIMM) [36] detects microbursts by analysing the egress queue's depth and diverting part of the packets to other programmable switches to prevent more microbursts along the whole path. This leads to faster flow completion times on average.

RIMM achieves effective results during background flows. Still, the most significant difference comes into play during bursty flows, where the average completion time is markedly lower than traditional forwarding is capable of. These results in Flow Completion

Times (FCT) that are, on average, twice as fast as basic forwarding would allow for while remaining significantly faster on background flows.

5.7 Video Playing SDN

Various networks present distinctive challenges, which require adjusting their components to enhance performance, particularly concerning video playback. The increasing number of video streaming services demands networks that can promptly and efficiently deliver packets, minimizing jitter and latency, both of which are critical determinants of video quality. Consequently, software-defined networking (SDN), with a focus on the P4 programming language, has been suggested as a solution to address this issue.

In [25], the authors use a P4 behavioural model switch to dynamically filter critical packets in devices' priority queues toward reducing the end-to-end delay of video traffic in SDN environments. To evaluate the proposed strategy, they used two approaches, i.e., *highest priority* and *lowest priority*. For the highest priority approach, the quality of service was assessed by analysing the network reaction to changes in response time through a variable end-to-end delay of beacon packets. The achieved results are:

- For a 1-second end-to-end delay, the average response time was 121.4 seconds.
- For a 20-second end-to-end delay, the average response time increased to 223.2 seconds.
- For a 50-second end-to-end delay, the average response time further increased to 313.1 seconds.
- For a 100-second end-to-end delay, the average response time was 567.5 seconds.

These results indicate that as the end-to-end delay increases, there is a corresponding degradation in network performance. The video transmission begins to deteriorate at a 50-second end-to-end delay, and at a 100-second delay, the video can no longer be reproduced.

The lowest priority approach results in the infeasibility of video streaming due to the substantial bandwidth allocation to traffic produced by intermediate switches. As a consequence, video packets are discarded instead of being forwarded.

5.8 Testing

Advancements in technology have resulted in increasingly rigorous requirements for network testers. Consequently, constant development and adaptation have been necessary to find better solutions to cope with highly demanding networks. With the introduction of programmable data planes utilising the P4 programming language, network testers now possess the necessary tools and flexibility to satisfy these requirements. Programmable switches represent an optimal testing target due to their flexible and reconfigurable packet processing capacity coupled with multi-Tbps bandwidth. Additionally, their straightforward and deterministic pipeline processing model and specialised hardware accelerators provide high accuracy for testing tasks.

In an effort to increase flexibility in testing, a tool called HyperTester has been developed by [56]. It consists of a network testing tool seeking to balance flexibility, high accuracy, large-scale scenarios, and low-cost operation, which was challenging to achieve simultaneously without the recent programmable switches.

The presented findings indicate that HyperTester exhibits linear scalability concerning the number of ports and can achieve a maximum total throughput of 1.6 terabits per second with 16 active ports. Additionally, HyperTester distinguishes itself from other similar software solutions. The comparison was conducted taking into consideration three technologies, namely HyperTester (HT), MoonGen (MG), and TRex (TR). It focuses on the achieved throughput while accounting for variations in packet size and line rates of 10Gbps and 40Gbps. The results revealed that MG and TR could not generate small-sized packets to utilise the entire line rate due to CPU processing capacity limitations. At the same time, TR failed to saturate the higher line rate wire due to inefficient implementation. In contrast, HyperTester was able to generate packets of arbitrary size at both line rates. To fully utilise the throughput potential of MG and TR, packet sizes of approximately 256 bytes were necessary.

5.9 Flow Scalability and Rule Responsiveness

The Software-Defined Networking (SDN) concept involves separating the control plane from the data plane in forwarding devices, which enables programmability in both the control and data planes. The P4 programming language leverages such potential by providing data plane programmability to enhance forwarding performance. A study evaluating this concept has been conducted and published by Harkous et al. [22].

The authors assess the performance implications of P4 using two metrics: Flow Scalability and Rule Responsiveness. Specifically, Flow Scalability is defined as the capability of a system to maintain its performance level when accommodating an increasing number of devices or users. As more concurrent flows are added, the device's capacity for each starts dwindling, delaying communications. Fixed-function devices do not currently have a way to deal with this issue. Conversely, Rule Responsiveness is defined as the duration required for a packet processing pipeline to respond to control plane commands and is one of the main problems OpenFlow faces. As the number of rules of an SDN device increases, so does the time it takes to match a packet with its corresponding rule, becoming much slower than a fixed-function device when processing these packets.

The experimental component used three P4 programmable packet processors: Netronome SmartNIC, NetFPGA-SUME and T4P4S. The results indicate that the NetFPGA-SUME and Netronome SmartNIC could handle more flows without negatively impacting packet processing latency, unlike the T4P4S software-based switch. The rule update responsiveness analysis also revealed that the devices responded differently to control plane commands. However, the response time was always in the range of milliseconds, which is three orders of magnitude larger than data plane processing. Furthermore, it was observed that preprocessing of control plane commands, which occurs in the device's provided toolchains, contributes significantly to the response time, highlighting the importance of optimising their design.

5.10 Pipeline Mapping

Although SDN has enhanced network flexibility, due to low-cost fixed-function networking hardware limitations, creating portable control plane software with current SDN standards requires accepting limited functionality and navigating inconsistent implementations. Conversely, using more powerful hardware may still be too expensive for most enterprises [12].

To address this issue, Shoehorn was developed [37]. It allows for creation of portable SDN control-plane software suitable for low-cost hardware. To achieve the desired effect, the software defines a virtual pipeline in P4 for the target hardware. Once this pipeline is in place and operational, the Shoehorn finds mappings between the physical and virtual pipelines and matches the respective tables to maximise performance and portability. This mapping allows the Shoehorn to understand the relationship between the different components of the network and how they interact with each other. Then, it uses the mapping information to match the respective tables between the physical and virtual pipelines.

5.11 Congestion Control

Congestion control in the Internet faces challenges due to high-speed links, diverse traffic, limited buffer sizes, and varied goals. Goals include fair resource sharing, delay reduction, throughput maximization, and solving the Incast challenge. Using hardware-supported priority classes and data-plane programmability of P4 switches, the authors of [16] describe a novel network-assisted congestion feedback (NCF) technique. Instead of marking packets for congestion like ECN does when congestion occurs at a switch, they produce NACKs from trimmed packets and transmit them straight to the sender. This method, which deviates from NDP, speeds up and shortens sender reaction times by enabling effective NACK creation utilizing P4 within the data plane. Through eschewing the receiver in the feedback loop and arranging NACKs in high-priority control queues, they accelerate alerts of congestion, helping to mitigate Incast issues and promote equitable bandwidth distribution. NACKs can be dropped in times of extreme congestion, and there is very little overhead when they are not aggregated.

According to [19], the control capabilities and improved visualization of P4 greatly benefit TCP in congestion control and Active Queue Management (AQM) algorithms. P4-programmable devices handle packets at line rate, making it possible to precisely construct congestion control algorithms. It is possible to design custom congestion management algorithms to meet certain requirements, such as latency reduction and TCP flow dynamics isolation. Sub-millisecond responsiveness is attained by telemetry-based congestion control, making it appropriate for high-speed networks. While more recent techniques try to decrease this expense, INT-based schemes still lower payload size. Interoperability with legacy systems and incremental implementation should be the main areas of future effort. Although traffic isolation methods have the potential to reduce congestion, the different congestion control algorithms used by competing flows may cause problems with fairness. By improving TCP congestion feedback using customized AQM algorithms, bufferbloat can be effectively addressed and TCP performance measures such as fairness, link usage, and Flow Completion Time (FCT) can be improved. By freeing up CPU time for difficult tasks, P4-programmable NICs improve server performance while saving resources and money for cloud operators. They provide low overhead and great performance, which are essential for effective data center operations. However, the increasing discrepancy between CPU capability and network bandwidth makes it difficult to guarantee good TCP behavior. Subsequent studies ought to tackle the security consequences associated with circumventing kernel features. To diagnose and locate errors related to TCP performance, network measurements and monitoring are essential. Prior studies have attempted to reduce overhead, evaluate the seriousness of the issue, and pinpoint failure sites. Future developments should broaden monitoring capabilities, decrease controller intervention, improve issue evaluation accuracy, decrease storage requirements, and interface with legacy networks. Application-layer inspection, traffic metering, traffic separation, and bandwidth management are a few examples of traffic policing and management strategies that are used in response to shortcomings in Quality of Service (QoS).

In other hand, according to paper [24], The Dynamic Longer Stay Less Priority (D-LSLP) algorithm was created with the purpose of reducing the flow completion time (FCT) in data center networks, especially for brief flows. Demotion thresholds are adjusted using P4 switches in accordance with the current queue occupancy, enabling short flows to finish swiftly in queues with higher priorities. Without requiring human input, it automatically adjusts to traffic patterns, decreasing tail drop and enhancing perform. The goal of D-LSLP is to reduce FCT for short flows. In response to the current queue occupancy, D-LSLP dynamically modifies the demotion threshold for flows' packets that arrive in strict priority queues. In order to prevent queue overflow, it also addresses the tail drop problem related to PIAS, LSLP, and AuTO by examining queue occupancy. In the experiment conducted in the study,D-LSLP's functionality on a small test bed after implementing it in the P4 programmable switches. They compared the outcomes with those of LSLP and DCTCP, demonstrating that when an alternate workload is applied, D-LSLP performs better than LSLP. They intend to employ artificial intelligence in the future to determine more ideal threshold values, both minimum and maximum.

The next paper's authors [35], examine novel techniques made possible by P4 for identifying and controlling elephant and heavy-hitter flows, which are essential for maximizing the use of network resources. Researchers have modified methods such as HashPipe for heavy-hitter identification to operate with the Tofino hardware switches, which are common in contemporary networks. A network-wide technique has been developed that makes use of P4 registers to allow a central coordinator to define switch thresholds for certain flows. By monitoring incoming traffic and reporting values to the coordinator when thresholds are achieved, these switches—which are set via P4—ensure effective detection while reducing errors through global threshold management. Similar to this, P4-enabled technologies have become effective instruments for elephant flow detection. Unlike other solutions, IDEAFIX uses P4 registers to quickly detect elephant fluxes in the data plane, reducing the need for controller intervention. IDEAFIX optimizes memory utilization while achieving accurate identification with the use of bloom filters and in-memory hash tables. Comparison with conventional sampling methods shows that IDEAFIX performs better and requires less memory. Another noteworthy tool is hashflow, which provides full flow management by precisely logging mouse flows and summarizing data for elephant flows. With the use of hash tables and creative collision prevention techniques, Hashflow guarantees accurate flow control. The evaluation findings show that hashflow performs better than comparable techniques, demonstrating its effectiveness in P4-based network setups.

5.12 Security

Based on [8], rapid development of new applications and protocols at all levels of the protocol stack is becoming more and more necessary as the Internet of Things, cloud computing, data centers, and 5G networks come into being. P4 is essential to online security because it makes it possible for programmable network devices to effectively deploy strong protections against a range of cyberthreats.

- Spoofing: When applied uniformly to every router, for basic router-level filtering, P4 enables the creation of programmable data plane logic that includes methods like RPF and ingress/egress filtering—offers strong defense against spoofing attacks. On the other hand, distributed anti-spoofing techniques such as SPM provide higher detection efficiency but require more frequent distribution of keys and switch resources.
- DDoS: Most DDoS detection methods concentrate on volumetric attacks. While other research projects, like POSEIDON, need external hardware to conduct computation-intensive tasks, a few recent studies, shows that it is possible to detect a wide range of threats using simply programmable switches such a P4. Infrastructures benefit from approaches that don't rely on external hardware because they have cheap capital costs and offer line rate performance.
- Network Verification: Traditional methods of network testing are not enough for programmable networks because of their dynamic and adaptable nature. Present methods concentrate on checking P4 programs and compilers to identify as many defects as possible through the use of tools like SMT solver, dynamic and static verification, symbolic model checker, etc.
- Privacy and anonymity: Existing methods for network anonymization systems work quickly to establish anonymity using things like onepad encryption, source address rewriting and normalization, etc. The operator can configure the anonymization policy through an interface offered by ONTAS, an early P4-based anonymization technique. More newer techniques, like PANEL and SPINE, require an independent system or trusted first hop in order to achieve anonymization up to the TCP/UDP layers.
- Cryptography and security protocols: Early data plane cryptography systems offloaded sophisticated computations to an external server or the switch's local CPU, adding to the latency. Recent research, however, has shown that recirculation and resubmission are frequently employed in P4 to simulate loops (rounds). In a similar vein, content permutation can be achieved by splitting the payload and rearranging it during several phases. Some strategies, like the one in, concentrate on speeding up cryptographic operations and refer to them as P4 externs; these might be useful for businesses that are willing to forgo performance in lieu of security.
- Firewalls: P4 implements packet filtering, port knocking, and 5G firewall defenses to provide high-performance, low-cost firewalls. It also makes it possible to create bespoke firmware according to rules that the user specifies, increasing the range of options for firewall design to include unconventional header fields.

With source address verification and anomaly detection, SECAP Switch is a P4-based security solution that thwarts topology poisoning attempts. In [50], highlights the effectiveness of an address-based LFA defense that doesn't require a vetting period in order to thwart relay-type assaults. It does, however, recognize that other protections, such as variance-based detection, are necessary, particularly for edge circumstances. Constraints such as the absence of looping and recursive function support make it difficult to implement these solutions in P4, but there are viable workarounds, like the statistical computations shown by Gao et al., which open up new directions for investigation.

Examining security in programmable data planes with a P4 language emphasis. The background of P4, the specifications for programmable switches, and related works are all covered in this examination. In addition, a major objective-based taxonomy of P4 security implementations is presented.

5.13 Inband Network Telemetry

It is difficult for legacy telemetry tools and protocols to record microbursts and offer precise telemetry measurements. To overcome these constraints, in-band network telemetry, or INT, was created, which enables data plane engineers to precisely query the internal state of switches. After being embedded into packets, telemetry data is sent to a high-performance collector where it can be analyzed and used for tasks like updating control plane table entries. P4 is an essential component of this process since it offers a programming language that allows network devices to declare how they will process packets. P4 enables the deployment of INT and other advanced network features by enabling the flexible and programmable alteration of packet headers and payloads.Furthermore, P4 helps to address the overhead-accuracy trade-off by facilitating the construction of INT variations that minimize telemetry traffic overhead and maximize performance. Furthermore, P4 can be used to speed up INT collectors, allowing them to effectively manage high traffic volumes up to thousands of packets per second (Kpps). Further developments in this field might look into ways to make INT even better, like reducing packet headers, increasing visibility and coverage, improving telemetry data, and making deployment easier. P4 will still be essential in enabling programmable data plane capabilities [31].

According to [48]:

- Network Performance Telemetry: P4 makes it possible to put advanced telemetry techniques like CAPEST into practice. CAPEST uses INT to measure network performance indicators including available bandwidth and latency precisely, improving measurement accuracy and reducing intrusion.

- Network Monitoring: P4-based products, such as FlowStalker, show how effective P4 is in enabling thorough network monitoring without compromising speed by effectively capturing and processing flow packets to provide real-time monitoring capabilities.
- Congestion Control:P4 technology enables switches to directly integrate important network data into packets, improving visibility for end-host TCP algorithms, and increasing decision accuracy. It also makes congestion control methods like HINT easier to design.
- Network Security: As demonstrated by SINT, which uses P4-based telemetry processing to reduce weaknesses in INT and provide secure and accurate network measurements, P4's programmability improves security measures in networks.
- Routing: Scalable intent-based networking is made possible by P4-driven designs such as the Intent Plane, which convert high-level intents into P4 programs. This illustrates how P4 enables programmable and flexible routing solutions that maximize network performance.

5.14 Machine-Learning

Developing algorithms for machine learning enables computers to learn from data and make judgments without the need for explicit programming. Specialized programming languages like P4 are used to program network hardware like switches and routers. When combined, they present chances to improve network capabilities through the direct application of machine learning techniques to network management, allowing for real-time decision-making and dynamic traffic optimization.

The following paper [27], based on traffic volume and duration, the study suggests a flow classification scheme for data center networks that differentiates between long-lived and short-lived flows. It uses machine learning techniques developed in P4 language to handle various flow classes and perform early classification. The contribution uses Decision Tree ML approach for efficiency in a P4 context and provides an enhanced hash-and-store algorithm for classification. When compared to current approaches, evaluation utilizing data generated by simulation demonstrates improved classification accuracy and three times faster classification speeds. The paper presents SMASH, a machine learning (ML) inference based early-stage categorization method for data center network (DCN) flows within switches. It shows that, although SMASH accomplishes classification as early as 3 MB of flow size, it can categorize three categories of flows with equivalent accuracy to threshold-based techniques. The integration of machine learning inference into switches improves its early flow classification performance.

Using P4-enabled data plane programmability, based on [43], features extraction at the data plane to significantly reduce detection latency in ML-assisted DDoS attack detection framework for SDN networks. In order to achieve this, was compared the accuracy and computational time of various ML classifiers and was implemented the algorithms in a real-time scenario where the P4 switch provides the ML classifiers with three different types of data: packet mirroring, header mirroring, and P4-metadata extraction. According to numerical results, attack detection accuracy can typically be carried out with accuracy higher than 98% and in less than 1 ms if P4 is utilized for feature extraction.

5.15 Quality of Service

According to [52], by directly integrating maximum latency limitations into packet headers, P4 enables dynamic QoS adaption. In order to ensure that packets fulfill their deadlines without overloading the network, switches can now decrease delay allowances based on actual network conditions. Their method combines the creation of a unique link-latency estimation protocol (LLEP) enabling real-time updates on latency information within switches with QoS-based packet forwarding, where switches prioritize packets based on predetermined latency deadlines. In order to guarantee effective handling and prompt packet delivery throughout the network, they also propose Cooperative Active Queue Management (AQM), a technique that classifies packets according to their deadline needs.

Based on [15], In order to maximize bandwidth utilization, they describe a bandwidth management strategy for P4 programmable switches that permits non-guaranteed traffic to use unallocated bandwidth. Their architecture is not limited by set actions or packet processing constraints, in contrast to OpenFlow-based techniques. By restricting output traffic and putting priority forwarding in place, they guarantee the minimum amount of bandwidth during traffic congestion. Their studies on a production network show that both guaranteed and limited bandwidth can be enforced effectively, and that the use of priority queues affects how well bandwidth management works.

The authors of the following paper [54] presents P4SQA, which uses P4 switches to manage queues and classify in-network traffic in order to improve QoS in data center networks. With the use of a binarized neural network, it transfers neural computation to the data plane for effective traffic classification. It also incorporates PCoDeL for efficient queue management, which lowers network latency and raises service standards.

5.16 Serverless

This study [45] showcased a solution that extends AWS services for serverless function deployment and operation, facilitating the transfer of computing tasks to edge settings. This method may be applied to serverless settings and other providers. Two compute task migration strategies were shown, and warm-up capabilities were included to remove function cold start delays. Significant reductions in memory consumption and latency spikes during migration were seen in evaluations utilizing an automotive remote control application. The P4 Offload Controller (P4OC) demonstrated exceptional efficacy by minimizing communication overhead and function warm-up, resulting in no additional delay. Subsequent enhancements encompass reducing offload controller response times via increased frequency of CPU assessments and effective metrics acquisition. Performance could be improved by expanding the P4OC to accommodate additional edge nodes and a variety of metrics. Further configuration optimization may be possible through active delay discovery between edge nodes and an upper-level placement optimization entity. All things considered, their study shows how serverless operations may be extended to edge settings, enhancing performance and opening the door for further developments in serverless edge computing.

5.17 Visual Cloud Computing

The configurable data planes of P4 are utilized in the following research [42] to improve video streaming in Visual Cloud Computing applications. Utilizing P4's features and a customized header, they are able to avoid bottlenecks and route traffic more efficiently. P4 offers useful information on timestamps, queue occupancy, and routing paths through its in-band telemetry. Their video streaming testbed on the FABRIC infrastructure shows less packet loss and increased throughput. In order to provide a baseline, the researchers first measure the transmission of video data over a regular network without P4. After that, they installed a testbed on the P4-programmable switches that are part of the FABRIC infrastructure. P4 programs are particularly created and put into these switches to control the flow of video data. The efficiency of P4 in lowering packet loss and increasing throughput for video streaming may be evaluated by the researchers by contrasting the performance of video transmission with and without P4 in the network. Their findings demonstrated notable advancements with P4. As seen by the table, P4 eliminated packet loss to 0% in both 400 Mbps and 800 Mbps congestion situations (as opposed to substantial packet loss in the absence of P4). Additionally, P4 boosted throughput by 85% and 122% for the 400 Mbps and 800 Mbps congestion scenarios, respectively, as indicated by average packets per second (PPS). Lastly, P4 dramatically lowered the maximum delay (delta) that video packets encountered.

6 CONCLUSIONS AND FUTURE DIRECTIONS

P4 has had a very large impact ever since its inception, and it is our belief that it will continue to do so. As the technology matures, more advancements will be found, and as it becomes more economically available, adoption rates will similarly grow.

One of the key areas of future research in P4 is the application of machine learning techniques to optimize P4 programs. While some initial work has been done in this area, there is still much room for improvement in terms of developing more sophisticated algorithms and models that can effectively learn from network data and improve the performance of P4 programs. Language maturity will certainly be a large factor in developing machine learning models that will help further improving networking.

Integration of P4 with other network technologies SDN and NFV is similarly still in its infancy. Some research has already been made in this regard, but there is a lack of solutions that incorporate these technologies together. There is a growing need to understand how P4 can be effectively combined to create even more flexible and efficient network systems.

The past few years have seen a massive growth of new and different types of networks such as data center networks, WANs, and edge networks. Investigating the unique characteristics and requirements of these different types of networks, and how P4 can be tailored to meet them, could similarly lead to significant advances in the field as P4 advances in these fields are still relatively sparse.

Security is becoming an increasingly important concern in P4 networks, and more research is needed to understand how P4 can be used to secure networks against various types of attacks as well improve upon current methods. Investigating new security mechanisms and techniques that can be implemented in P4, while evaluating their effectiveness, would be a valuable contribution to the field.

Lastly, P4 is being used for programmable data planes for 5G and future networks, but there is still much to be done in terms of understanding the trade-offs and opportunities that these programmable data planes bring to network operators and service providers. 5G is still very new and the full potential of using P4 alongside it has not yet been realized. As 5G becomes the standard and widely adopted, many uses of P4 will undoubtedly be found, making its flexibility crucial not only for current technology but also ensuring it can adapt to any future developments.

ACKNOWLEDGMENTS

This work is financed by National Funds through the FCT - Fundação para a Ciência e a Tecnologia within the project FLEXCOMM, EXPL/CCI-INF/1543/2021, and by FCT, within the R&D Units Project Scope: UIDB/00319/2020.

REFERENCES

[1] [n. d.]. Buffer management. https://www.sciencedirect.com/topics/computer-science/buffer-management
[2] [n. d.]. Explore the power of Intel® intelligent fabric processors. https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch.html
[3] [n. d.]. Resources. https://www.netronome.com/document-library/
[4] 2021. Solutions - cisco silicon one Q211 and Q211L processors data sheet. https://www.cisco.com/c/en/us/solutions/collateral/silicon-one/datasheet-c78-744834.html
[5] 2022. What is load balancing? how load balancers work. https://www.nginx.com/resources/glossary/load-balancing/
[6] Zeeshan Ahmad, Adnan Shahid Khan, Cheah Shiang, and Farhan Ahmad. 2021. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies* 32 (01 2021). https://doi.org/10.1002/ett.4150
[7] Oğuzhan Akyıldız, İbrahim Kök, Feyza Yıldırım Okay, and Suat Özdemir. 2023. A P4-assisted task offloading scheme for Fog networks: An intelligent transportation system scenario. *Internet of Things* 22 (2023), 100695. https://doi.org/10.1016/j.iot.2023.100695
[8] Ali AlSabeh, Joseph Khoury, Elie Kfoury, Jorge Crichigno, and Elias Bou-Harb. 2022. A survey on security applications of P4 programmable switches and a STRIDE-based vulnerability assessment. *Computer Networks* 207 (2022), 108800. https://doi.org/10.1016/j.comnet.2022.108800
[9] Gereltsetseg Altangerel and Tejfel M'at'e. 2021. Survey on some optimization possibilities for data plane applications. *ArXiv* abs/2201.11516 (2021).
[10] Mostafa Ammar. 2017. ex uno pluria: The Service-Infrastructure Cycle, Ossification, and the Fragmentation of the Internet. *ACM SIGCOMM Computer Communication Review* 48 (12 2017). https://doi.org/10.1145/3211852.3211861
[11] Sumit Badotra. 2017. A Review Paper on Software Defined Networking. *International Journal of Advanced Computer Research* 8 (03 2017).
[12] Sumit Badotra. 2017. A Review Paper on Software Defined Networking. *International Journal of Advanced Computer Research* 8 (03 2017).
[13] 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 (jul 2014), 87–95. https://doi.org/10.1145/2656877.2656890
[14] Yeim-Kuan Chang, Hung-Yen Wang, and Yu-Hsiang Lin. 2021. A Congestion Aware Multi-Path Label Switching in Data Centers Using Programmable Switches. In *2021 IEEE International Conference on Networking, Architecture and Storage (NAS)*. 1–8. https://doi.org/10.1109/NAS51552.2021.9605422
[15] Yan-Wei Chen, Li-Hsing Yen, Wei-Cheng Wang, Cheng-An Chuang, Yu-Shen Liu, and Chien-Chao Tseng. 2019. P4-Enabled Bandwidth Management. In *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 1–5.

https://doi.org/10.23919/APNOMS.2019.8892909

[16] Anja Feldmann, Balakrishnan Chandrasekaran, Seifeddine Fathalli, and Emilia N. Weyulu. 2020. P4-enabled Network-assisted Congestion Feedback: A Case for NACKs. In *Proceedings of the 2019 Workshop on Buffer Sizing* (Palo Alto, CA, USA) *(BS '19)*. Association for Computing Machinery, New York, NY, USA, Article 3, 7 pages. https://doi.org/10.1145/3375235.3375238

[17] David Franco, Eder Ollora Zaballa, Mingyuan Zang, Asier Atutxa, Jorge Sasiain, Aleksander Pruski, Elisa Rojas, Marivi Higuero, and Eduardo Jacob. 2024. A comprehensive latency profiling study of the Tofino P4 programmable ASIC-based hardware. *Computer Communications* 218 (2024), 14–30. https://doi.org/10.1016/j.comcom.2024.01.010

[18] Ahammed G.F.Ali and Reshma Banu. 2010. Analyzing the Performance of Active Queue Management Algorithms. *International journal of Computer Networks and Communications* 2 (03 2010). https://doi.org/10.5121/ijcnc.2010.2201

[19] Jose Gomez, Elie F. Kfoury, Jorge Crichigno, and Gautam Srivastava. 2022. A survey on TCP enhancements using P4-programmable devices. *Computer Networks* 212 (2022), 109030. https://doi.org/10.1016/j.comnet.2022.109030

[20] Nicholas Gray, Katharina Dietz, Michael Seufert, and Tobias Hossfeld. 2021. High Performance Network Metadata Extraction Using P4 for ML-based Intrusion Detection Systems. In *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*. 1–7. https://doi.org/10.1109/HPSR52026.2021.9481849

[21] Akram Hakiri, Aniruddha Gokhale, Pascal Berthou, Douglas C. Schmidt, and Thierry Gayraud. 2014. Software-Defined Networking: Challenges and research opportunities for Future Internet. *Computer Networks* 75 (2014), 453–471. https://doi.org/10.1016/j.comnet.2014.10.015

[22] Hasanin Harkous, Mu He, Michael Jarschel, Rastin Pries, Ehab Mansour, and Wolfgang Kellerer. 2021. Performance Study of P4 Programmable Devices: Flow Scalability and Rule Update Responsiveness. In *2021 IFIP Networking Conference (IFIP Networking)*. 1–6. https://doi.org/10.23919/IFIPNetworking52078.2021.9472782

[23] Keqiang He, Junaid Khalid, Aaron Gember-Jacobson, Sourav Das, Chaithan Prakash, Aditya Akella, Li Erran Li, and Marina Thottan. 2015. Measuring Control Plane Latency in SDN-Enabled Switches. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research* (Santa Clara, California) *(SOSR '15)*. Association for Computing Machinery, New York, NY, USA, Article 25, 6 pages. https://doi.org/10.1145/2774993.2775069

[24] Muhammad Shahid Iqbal and Chien Chen. 2024. Instant queue occupancy used for automatic traffic scheduling in data center networks. *Computer Networks* 244 (2024), 110346. https://doi.org/10.1016/j.comnet.2024.110346

[25] Calin-Marian Iurian, Robert Botez, Iustin-Alexandru Ivanciu, and Virgil Dobrota. 2022. Video Streaming Evaluation Using Priority Queuing in P4 Programmable Networks. In *2022 21st RoEduNet Conference: Networking in Education and Research (RoEduNet)*. 1–5. https://doi.org/10.1109/RoEduNet57163.2022.9921096

[26] JuniperNetworks. [n. d.]. Juniper's P4 efforts - Sandesh Kumar Sodhi, Juniper Networks. https://junipernetworks.lookbookhq.com/depaulpresentations/JRENS-p4-efforts

[27] Radhakrishna Kamath and Krishna M Sivalingam. 2021. Machine Learning based Flow Classification in DCNs using P4 Switches. In *2021 International Conference on Computer Communications and Networks (ICCCN)*. 1–10. https://doi.org/10.1109/ICCCN52240.2021.9522272

[28] 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* (Santa Clara, CA, USA) *(SOSR '16)*. Association for Computing Machinery, New York, NY, USA, Article 10, 12 pages. https://doi.org/10.1145/2890955.2890968

[29] Elie Kfoury, Jorge Crichigno, Elias Bou-Harb, and Gautam Srivastava. 2021. Dynamic Router's Buffer Sizing using Passive Measurements and P4 Programmable Switches. In *2021 IEEE Global Communications Conference (GLOBECOM)*. 01–06. https://doi.org/10.1109/GLOBECOM46510.2021.9685160

[30] Elie F. Kfoury, Jorge Crichigno, and Elias Bou-Harb. 2021. An Exhaustive Survey on P4 Programmable Data Plane Switches: Taxonomy, Applications, Challenges, and Future Trends. *IEEE Access* 9 (2021), 87094–87155. https://doi.org/10.1109/ACCESS.2021.3086704

[31] Elie F. Kfoury, Jorge Crichigno, and Elias Bou-Harb. 2021. An Exhaustive Survey on P4 Programmable Data Plane Switches: Taxonomy, Applications, Challenges, and Future Trends. *CoRR* abs/2102.00643 (2021). arXiv:2102.00643 https://arxiv.org/abs/2102.00643

[32] Manasa Kulkarni, Bhargavi Goswami, and Joy Paulose. 2024. Efficient Traffic Distribution: P4 language based Dynamic Load Balancing Algorithm in SDN. https://doi.org/10.21203/rs.3.rs-3988061/v1

[33] Ralf Kundel, Amr Rizk, Jeremias Blendin, Boris Koldehofe, Rhaban Hark, and Ralf Steinmetz. 2021. P4-CoDel: Experiences on Programmable Data Plane Hardware. In *ICC 2021 - IEEE International Conference on Communications*. 1–6. https://doi.org/10.1109/ICC42927.2021.9500943

[34] Ike Kunze, Moritz Gunz, David Saam, Klaus Wehrle, and Jan Rüth. 2021. Tofino + P4: A Strong Compound for AQM on High-Speed Networks?. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. 72–80.

[35] 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 (jan 2023), 37 pages. https://doi.org/10.1145/3556973

[36] Yu-Jie Lin, Chi-Hsiang Hung, and Charles H.-P. Wen. 2022. Real-Time In-Network Microburst Mitigation on Programmable Switch. *IEEE Access* 10 (2022), 2446–2456. https://doi.org/10.1109/ACCESS.2021.3139642

[37] Christopher Lorier, Matthew Luckie, Marinho Barcellos, and Richard Nelson. 2022. Shoehorn: Towards Portable P4 for Low Cost Hardware. In *2022 IFIP Networking Conference (IFIP Networking)*. 1–9. https://doi.org/10.23919/IFIPNetworking55013.2022.9829819

[38] Diego Rossi Mafioletti, Frank Slyne, Robin Giller, Michael O'Hanlon, David Coyle, Brendan Ryan, and Marco Ruffini. 2022. Demonstration of a low latency bandwidth allocation mechanism for mission critical applications in virtual PONs with P4 programmable hardware. In *2022 Optical Fiber Communications Conference and Exhibition (OFC)*. 1–3.

[39] Diego Rossi Mafioletti, Frank Slyne, Robin Giller, Michael O'Hanlon, Brendan Ryan, and Marco Ruffini. 2021. A Novel low-latency DBA for Virtualised PON implemented through P4 In-Network Processing. In *2021 Optical Fiber Communications Conference and Exhibition (OFC)*. 1–3.

[40] Ali Mazloum, Elie Kfoury, Jose Gomez, and Jorge Crichigno. 2023. A Survey on Rerouting Techniques with P4 Programmable Data Plane Switches. *Computer Networks* 230 (2023), 109795. https://doi.org/10.1016/j.comnet.2023.109795

[41] 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

[42] Alicia Esquivel Morel, Prasad Calyam, Chengyi Qu, Durbek Gafurov, Cong Wang, Komal Thareja, Anirban Mandal, Eric Lyons, Michael Zink, George Papadimitriou, and Ewa Deelman. 2023. Network Services Management using Programmable Data Planes for Visual Cloud Computing. In *2023 International Conference on Computing, Networking and Communications (ICNC)*. 130–136. https://doi.org/10.1109/ICNC57223.2023.10074183

[43] Francesco Musumeci, Valentina Ionata, Francesco Paolucci, Filippo Cugini, and Massimo Tornatore. 2020. Machine-learning-assisted DDoS attack detection with P4 language. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 1–6. https://doi.org/10.1109/ICC40277.2020.9149043

[44] M.W. Oliphant. 1999. The mobile phone meets the Internet. *IEEE Spectrum* 36, 8 (1999), 20–28. https://doi.org/10.1109/6.780995

[45] István Pelle, Francesco Paolucci, Balázs Sonkoly, and Filippo Cugini. 2023. P4-assisted seamless migration of serverless applications towards the edge continuum. *Future Gener. Comput. Syst.* 146 (2023), 122–138. https://api.semanticscholar.org/CorpusID:258253771

[46] Alexandru Seibulescu and Mario Baldi. 2020. Leveraging P4 Flexibility to Expose Target-Specific Features. In *Proceedings of the 3rd P4 Workshop in Europe* (Barcelona, Spain) *(EuroP4'20)*. Association for Computing Machinery, New York, NY, USA, 36–42. https://doi.org/10.1145/3426744.3431326

[47] Maryam Sheikh Sofla, Mostafa Haghi Kashani, Ebrahim Mahdipour, and Reza Faghih Mirzaee. 2022. Towards Effective Offloading Mechanisms in Fog Computing. *Multimedia Tools Appl.* 81, 2 (jan 2022), 1997–2042. https://doi.org/10.1007/s11042-021-11423-9

[48] Amit Kumar Singh. 2024. In-band Network Telemetry with Programmable Data Plane Research Review: Challenges and Future Directions. *Authorea* (02 February 2024).

[49] Timon Sloane. 2013. OpenFlow. https://opennetworking.org/sdn-resources/customer-case-studies/openflow/

[50] Dylan Smyth, Sandra Scott-Hayward, Victor Cionca, Sean McSweeney, and Donna O'Shea. 2023. SECAP Switch—Defeating Topology Poisoning Attacks Using P4 Data Planes. *Journal of Network and Systems Management* 31, 1 (2023), 28. https://doi.org/10.1007/s10922-022-09714-z

[51] Hadar Sufiev, Yoram Haddad, Leonid Barenboim, and José Soler. 2019. Dynamic SDN Controller Load Balancing. *Future Internet* 11, 3 (2019). https://doi.org/10.3390/fi11030075

[52] Belma Turkovic, Soovam Biswal, Abhishek Vijay, Antonia Hüfner, and Fernando Kuipers. 2021. P4QoS: QoS-based Packet Processing with P4. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. 216–220. https://doi.org/10.1109/NetSoft51509.2021.9492539

[53] Intel Vladimir Gurevich. 2021. Intel highlights P4 research with over 60 papers published by the members of Intel® Connectivity Research Program. https://opennetworking.org/news-and-events/blog/intel-highlights-p4-research-with-over-60-papers-published-by-the-members-of-intel-connectivity-research-program/

[54] Qianqian Wu, Qiang Liu, Zequn Jia, Ning Xin, and Te Chen. 2023. P4SQA: A P4 Switch-Based QoS Assurance Mechanism for SDN. *IEEE Transactions on Network and Service Management* 20, 4 (2023), 4875–4886. https://doi.org/10.1109/TNSM.2023.3280913

[55] Wenfeng Xia, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, and Haiyong Xie. 2015. A Survey on Software-Defined Networking. *IEEE Communications Surveys & Tutorials* 17, 1 (2015), 27–51. https://doi.org/10.1109/COMST.2014.2330903

[56] Dai Zhang, Yu Zhou, Zhaowei Xi, Yangyang Wang, Mingwei Xu, and Jianping Wu. 2021. HyperTester: High-Performance Network Testing Driven by Programmable Switches. *IEEE/ACM Transactions on Networking* 29, 5 (2021), 2005–2018. https://doi.org/10.1109/TNET.2021.3077652

[57] Zikai Zhou, Mu He, Wolfgang Kellerer, Andreas Blenk, and Klaus-Tycho Foerster. 2021. P4Update: Fast and Locally Verifiable Consistent Network Updates in the P4 Data Plane. In *Proceedings of the 17th International Conference on Emerging Networking Experiments and Technologies* (Virtual Event, Germany) *(CoNEXT '21)*. Association for Computing Machinery, New York, NY, USA, 175–190. https://doi.org/10.1145/3485983.3494845