Enhancing 5G SDN/NFV Edge with P4 Data Plane Programmability

F. Paolucci
CNIT, 56124 Pisa, Italy

F. Cugini
CNIT, 56124 Pisa, Italy

P. Castoldi
Scuola Superiore Sant'Anna, 56124 Pisa, Italy

T. Osiński
Orange Labs & Warsaw University of Technology,
Warsaw, Poland

***Abstract*—5G Networks revolution will be enabled by deep integration of Software Defined Networking (SDN) and Network Function Virtualization (NFV) to support multi-tenancy, per-user and per-application quality of service and experience. However, full softwarization and current SDN platforms may not be able to sustain the complexity and the heterogeneity of different requirements, e.g. strict latency, jitter, high precision traffic and advanced monitoring. For such services, SDN/NFV needs to be boosted not only considering orchestration and control plane, but also data plane programmability. In this paper, the potential of the P4 language is illustrated with the aim to show its disruptive novel functionalities at the data plane level currently not available in a SDN/NFV network, opening the way to new orchestration frameworks and enabling a novel autonomic and flexible network at the edge. Use cases, assessments and softwarized performance results are proposed and discussed in the edge and IoT scenario, targeting advanced traffic engineering, cyber security, multi-tenancy, 5G offloading, and telemetry, to demonstrate the feasibility of such approach.**

1 INTRODUCTION

■ **THE ADVENT** of beyond 5G Networks is driving relevant enhancements in all the communication and IT segments to convey the requirements of Software Defined Networking (SDN) and Network Function Virtualization (NFV) [1]. Networks will be SDN-configurable and orchestrated to support multi-tenant environments by means of network slicing and fully virtualized service applications. These network functions will be automatically deployed as service chains to offer an unprecedented quality of transmission, service, and experience with yet unreached flexibility and cost reduction. Softwarization is one of the key drivers of this novel service-oriented framework [1]. SDN control

plane is foreseen as the candidate solution to support automated network configurability by means of standard API (e.g., OpenFlow, Netconf/YANG), separating and abstracting the data plane from the orchestration and control layer. However, current SDN platforms will not be able to sustain the complexity disclosed by heterogeneous softwarized networks, such as next generation disaggregated metro networks connecting data centers with 5G fronthaul, edge/fog nodes supporting IoT gateways serving massive sensors and devices. In such scenarios, the current SDN control lacks advanced and automatic forwarding capabilities, per-packet treatment and manipulation functions, high precision traffic monitoring/telemetry, and services with extreme requirements (e.g., zero jitter, bounded low latency). So far, a limited set of operation have been demanded to the SDN controller. However, this introduces undesirable delay and scalability issues, preventing its practical deployment as network bitrates increase.

To overcome such limitations, a new layer of softwarization has been introduced, providing the capability to program the switch data plane through high-level API and languages. The P4 language is emerging as the largest and most considered initiative [2]. Originally designed for intra-data center scenarios, P4 is rapidly attracting attention for advanced SDN/NFV softwarization solutions. Among the different languages, platforms and frameworks proposed for data plane programmability, including stateful functions, such as OpenState [3], RMT [4] and FAST [5], the P4 framework is gaining consensus and support by means of many vendors (e.g., Intel/Barefoot, Xilinx, Mellanox, Broadcom) and open source initiatives (e.g., ONOS and OpenDayLight).

In this paper, we first summarize the P4 technology and capabilities. Then, also resorting to our previous works on P4 [6][7], we present and discuss a number of use cases and novel opportunities enabled by the P4 language implemented in 5G edge/fog nodes, including dynamic traffic control for load balancing and offloading, latency-bounded service implementation, and novel stateful operations for embedded cyber security functions. Finally, we present an experimental assessment of selected use cases, providing novel P4 containerized software switch performance evaluation with different platforms in the edge/IoT context and a critical discussion on its benefits, applicability and open issues.

2 P4 DATA PLANE PROGRAMMABILITY

The P4 language was introduced to define and program the data plane pipelines and functions of an SDN switch [2]. P4 is a high level and platform-agnostic language. Data plane programmability includes, besides a custom pipeline (sequence of SDN match-action tables), configurable dataset (packet header, metadata), workflows (control), functions (actions) and stateful objects, allowing complex packet manipulation and processing procedures currently not available in standard SDN switches and that would require the intervention of the SDN controller. Packet headers may be defined simply by describing their structure in terms of fields and bytes, allowing existing (e.g., IP, Ethernet, TCP) and novel/experimental header parsers. The general scheme of a P4 switch follows the Portable Switch Architecture (PSA) framework and is shown in Fig. 1. The packet received by the input interface, is passed to the parser module, in charge to detect and dissect the supported protocols stack headers (e.g., Ethernet, MPLS, IP, TCP). The packet is then passed to the Ingress pipeline, triggering actions on the packet based on match conditions imposed by the flow tables definition. Moreover, the pipeline structure is not rigid (i.e., static sequence of tables). In fact, P4 allows a control section able to impose jump/goto/exclude conditions to the pipeline table enforcement and the order of execution. The ingress pipeline, besides all possible conditional actions and manipulations on the packet, is responsible for computing the target output interface and send the packet to the queue/buffers module. Finally, before retransmission, a further Egress pipeline is exploited for specific post-forwarding operations (e.g., multicasting).

The P4 code is first compiled by the P4 compiler, generating a JSON descriptor, that is processed by the data plane runtime system (provided by the switch manufacturer) in order to generate the modules and the structures inside the device (either software or hardware). Moreover, the JSON is used to provide the switch control plane runtime (and optionally the SDN controller) with the functional structure of the switch in terms of flow tables (names, values, matches). This

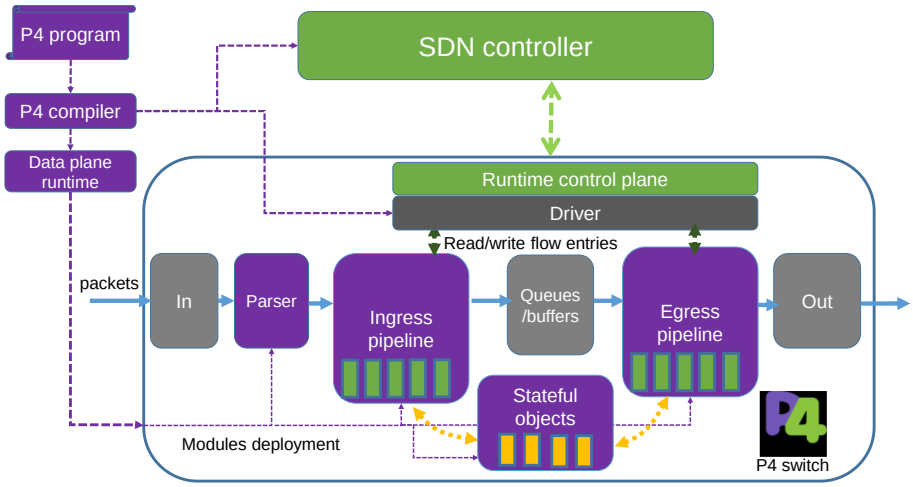

Figure 1. P4 switch: PSA architecture, P4 compilation and relationship with the SDN controller.

way, the device exposes the desired custom pipeline to the SDN controller, allowing flow entry modifications at runtime by means of the SDN API (e.g., OpenFlow, Thrift, CLI).

With respect to standard SDN switches using fixed pipeline and based on OpenFlow or Netconf/YANG, the P4 potentials are numerous. First, the custom parser section allows to define proprietary or novel protocol headers, which may be used by other P4 switches aware of such headers. Second, the custom pipelines and the control allows complex operations on the fields and values of the packets and on flow control and forwarding functions, including packet cloning, replication and recirculation in the pipeline. Third, the availability of custom packet metadata (fields and values defined in P4 used as additional per-packet parameters) allows to perform per-packet actions. Fourth, the stateful objects allow to define finite state machines inside the switch and describe complex states over which performing further forwarding selections or packet manipulations.

One interesting feature provided by P4 is the independency with respect to the data plane platform. The same P4 program can be applied on a different device, e.g. a software switch running on general purpose hardware (also as virtual machine or virtual container), a bare metal switch or an FPGA equipped with the P4 driver framework.

3 BENEFITS OF DATA PLANE PROGRAMMABILITY FOR THE EDGE/FOG

Fig. 2 shows a general functional architecture of a fog/edge node equipped with P4 programmable capabilities. The node interconnects several heterogeneous network segments, including other fog metro areas, 5G front-haul and IoT gateways, as well as metro-core infrastructures (e.g., towards remote Data Centers). The node includes compute and storage resources for multi-tenant and secure edge computing applications [1], which require complex and careful traffic forwarding treatments based on the requirements of each traffic flow and service. This node may be employed also in different locations of the 5G network to perform specific function offloading (see sec. 3.E). As described in the following, P4 has the potential to provide innovative forwarding functionalities opening the way to novel edge capabilities and softwarization solutions.

A) TRAFFIC ENGINEERING

Current Edge-Cloud and Fog infrastructures data pipelining is subject to quasi-static policies for traffic forwarding, defined either by distributed control plane or enforced by an SDN Controller. For example, if dynamic traffic conditions determine congestions and service degradation, the OpenFlow switch has no means to dynamically adapt forwarding decisions. Indeed, it operates on each packet according to flow rules, with no stateful capabilities. In case of congestion, the switch can only exchange relevant statistics and wait for the Controller to modify match conditions and flow actions. However, this procedure is slow and may introduce scalability issues at the controller.

Although P4 supports a limited traffic congestion control (i.e., priority change), the P4-based stateful

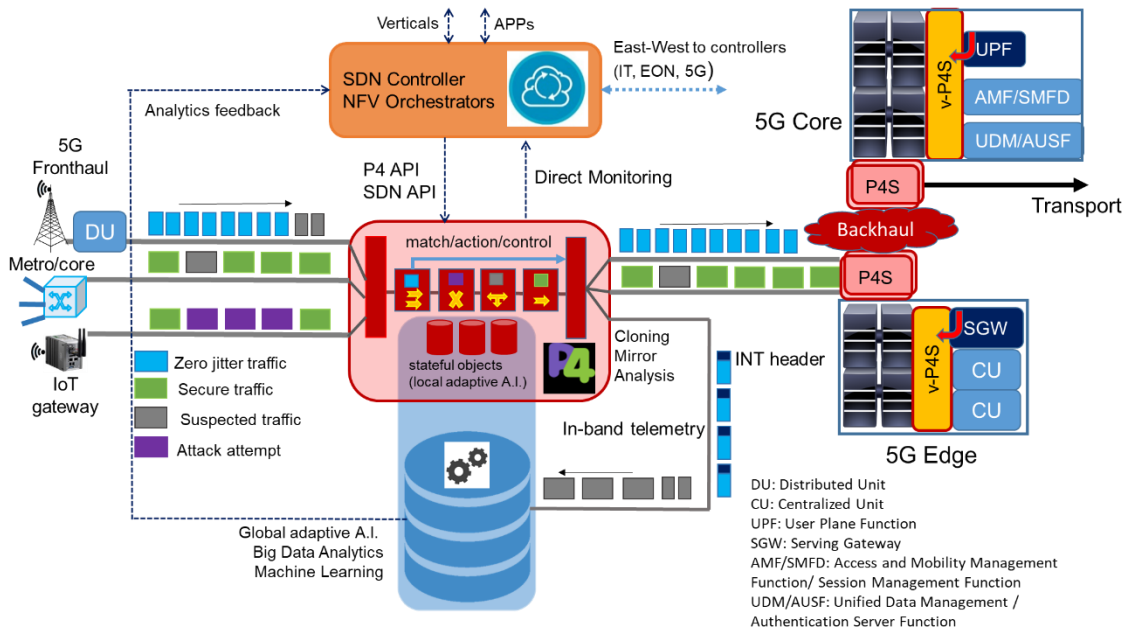

Figure 2. P4 switch interfacing edge/fog computing resources and utilization in the 5G landscape.

programmability can be exploited to dynamically modify match conditions and forwarding actions according to specific values of stateful parameters, such as meters or registers accounting for traffic rates or flow matches occurred during a specific time interval. This way, the SDN Controller can pre-instruct the switch with several forwarding options (e.g., finite state machines), eliminating the need to interact with the SDN Controller when critical network events occur (e.g. congestions). This opens the way to novel dynamic traffic engineering solutions, preserving centralized control while avoiding scalability issues at the Controller.

B) IN-BAND TELEMETRY FOR LATENCY-CRITICAL SERVICES

Traditionally, QoS is applied on a per-flow basis by configuring high scheduling priority to high class services. However, all high-class packets compete for the same queuing resources. Overall, unequal treatment among services of same class may be experienced as well as relevant latency variations over time (i.e., jitter) among packets of the same service.

In this context, the P4 language, despite not supporting yet a full queue scheduling programmability, may provide per-packet QoS with latency/jitter performance guaranteed exploiting in-band telemetry (INT). In particular, a specifically designed INT header [8] can be added/modified to all high-class packets at each traversed node, reporting the time spent in the outgoing queues across the network. This way, by analyzing the INT data on cumulated delay, indirect dynamic packet scheduling may be implemented (i.e., dynamic per-packet classification and priority enforcement), minimizing jitter and maximum experienced end-to-end latency. In addition, INT statistics could be exploited to derive long term statistics of metro network latency/service performance, potentially leading to global network re-optimizations to be enforced by the SDN Controller. Finally, such unprecedented amount of monitoring/telemetry and logging data may be the input of A.I.-based and machine learning aided adaptive strategies (see Fig. 2) to boost overall network performance [8].

C) SLICING AND MULTI-TENANCY

The emerging 5G market will rely on data collected from heterogeneous devices, vehicles and sensors. While 5G nodes and IoT gateways are managed by the infrastructure owner (e.g., the Operator or the city municipality), IoT data are typically handled and

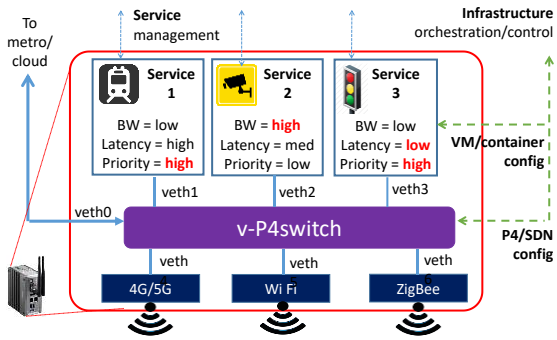

Figure 3. Virtual P4 switch enabling acceleration of cached services in edge and IoT gateway.

managed by different service entities and tenants (e.g., the vehicle service company, the company in charge of traffic lights, etc.), which would highly benefit from virtualized architectures with native slicing functionalities for VNF. Each service or vertical requires a different packet treatment to assure, e.g, secure real-time control to tune traffic light settings, support autonomous driving and electric vehicles, operate on corporate or home building automations, provide security through CCTV, and improve environmental sustainability in many ways. In this multi-tenant context, P4-based edge/fog nodes have the potential to provide per-tenant P4 virtualized functions. This way, each tenant is able to operate on a slice of network resources, being able to cope with smart services operated over a dedicated network platform with pre-defined and programmed performance features in terms of sustainable bandwidth, latency, jitter, priority, availability and security.

Fig. 3 shows an example of applicability of a virtual P4 switch integrated with an IoT gateway for smart grid applications. In this scenario the network infrastructure owner (e.g., the municipality) is able to host services of different providers (e.g., railway information system, road traffic management, CCTV system). A virtual P4 switch connects the virtual network functions of the different providers, implementing traffic segregation, slicing, dedicated QoS priority and security functions, switching traffic to the target gateway wireless interface, instantiating each service with its own requirements.

D) CYBER-SECURITY

Current stand-alone firewalls are exposed to internet with no other network barriers to stem attack/threats, which typically take advantage of security breaks at the edges (e.g., IoT devices). Moreover, firewalls are decoupled from switching elements, implemented over ad hoc, expensive hardware. OpenFlow-based firewalls have represented an interesting novelty in the firewall's approach [9, 10]. However, despite the cost benefits of exploiting bare metal hardware, the lack of stateful capabilities and the need to always delegate decisions to the slow SDN Controller have limited their utilization.

Thanks to the P4 native Deep Packet Inspection (DPI) capabilities to process the full packet header stack (i.e., including transport, session and application layers) combined with stateful functions, cyber security can be effectively implemented in each network switching element, including edge nodes and IoT gateways. For example, many cyber-attacks (e.g., Mirai IoT DDoS) leverage on address/port scan. Each packet alone, in stateless firewall implementations, would be simply considered as related to a genuine attempt to perform remote access. However, thanks to the stateful capabilities of the P4 node, by correlating the sequence of packets/scan, the attack can be detected and blocked already at network edges as well as at intermediate network nodes, successfully avoiding that such threats reach a DC gateway, and freeing resources to focus on more complex attacks. Moreover, by relying on telemetry and feature-based selected mirroring, correlations among selected threats affecting different nodes can lead to the creation of a distributed barrier against cyber-attacks [11].

E) 5G VNF OFFLOADING

As depicted in Fig. 2, the 5G Mobile Edge Computing (MEC) employs a number of specific VNF to realize 5G functional split between Distributed Units (DU) located in the fronthaul, Central Units (CU) at the edge and evolved packet core (EPC) functions. Some VNF may exploit P4 to be offloaded in an already existing software switch without resorting to dedicated virtual machines or additional and costly bare metal hardware [12], thanks to P4 programmability. For example, in the context of Serving Gateway (SGW) functions, the User Plane Function (UPF) VNF is responsible for GPRS tunneling protocol (GTP) encapsulation and decapsulation, traffic steering to single or multiple UPF nodes. Such functions are offloadable to virtual P4 switches. A single P4 code may be employed to perform GTP encapsulation, decapsulation, inter-UPF swapping and related steering of the traffic. Moreover,

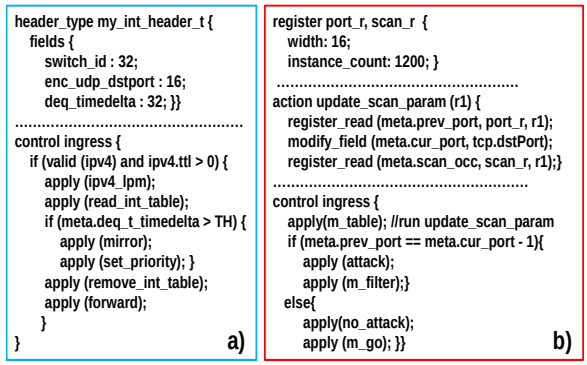

Figure 4. P4 codes excerpts: (a) in-band telemetry with proactive latency optimizator; (b) DDoS port scan mitigator.

P4 may be employed in an integrated way in both physical and virtual 5G network switches. With reference to Fig. 2, the combination of physical P4 top of the rack switches (P4S) and virtual intra-rack and intra-server switches (v-P4S) provides full SDN programmability of network functions for improved monitoring and tuning of 5G services under the same extended SDN control, drastically improving dynamic network performance and service reconfigurability.

4 P4 USE CASES

The five aforementioned use cases have been implemented and validated. For space reasons, we further detail two of them, including key code excerpts to show P4 compactness and suitability for IoT and virtual environments.

A) TELEMETRY

In a SDN network comprising P4-based edge, fog and aggregation nodes, INT header can be added to selected high priority flows to carry metadata as the time spent in queue at each node. Such metadata is normally not available in traditional switching solutions, but it has the potential to significantly improve the way QoS is implemented to low-latency services [7].

Each traversed node, from the edge to the fog/aggregation up to the central office or cloud can update/append its own metadata and elaborate on the cumulated received value to take specific forwarding actions. Then, according to latency thresholds, scheduling priority could be dynamically modified on a per-packet basis, overall reducing the maximum experienced latency and jitter to time-sensitive flows. In addition, also selected mirroring could be implemented to forward selected INT statistics to external processing modules, e.g. implementing A.I.-based long term elaborations for global traffic engineering re-optimizations.

Fig. 4(a) shows an excerpt of the P4 code utilized to implement telemetry and dynamic low-latency QoS enforcement. First, the code defines a new packet header *my_int_header_t* that will be used to dissect the telemetry header. The header encloses a switch id, a UDP port target and a differential timestamp. This way the switch is able to detect, dissect and store the telemetry header values. Once performed preliminary packet processing, the ingress control section checks whether the differential timestamp carried out by the header is above a TH threshold. In the case of excessive jitter, the packet is sent to a P4-level modified pipeline branch, where packet priority is increased (through table *set_priority*) and packet is cloned towards a mirror interface for external processing (through table mirror). Standard pipeline is exploited for all the packets by implementing the telemetry header extraction (table *remove_int_table*) and the forward operation (table *forward*).

B) ONLINE DDoS MITIGATION

An implementation of DDoS mitigation inside a P4 switch has been shown in [6], where a NetFPGA board was programmed with a P4 code exploiting stateful objects. With respect to stateless firewalls, the P4 switch processing is able to store and correlate protocol field values belonging to the same sessions, thus identifying suspected flows (e.g., port scan behavior) that may be dynamically blocked. However, it may not be sufficient to detect complex distributed attacks, simultaneously affecting multiple fog/edge nodes. Selected mirroring is then introduced to enable an external monitor to correlate packets traversing different edge nodes and potentially part of a more complex attack.

Fig. 4(b) shows an extract of P4 code exploiting novel stateful objects. For selected sessions, a register stores the last TCP port *port_r* and the number of related scan occurrences scan_r. The ingress control applies the *update_scan_param* action through the *m_table* table,

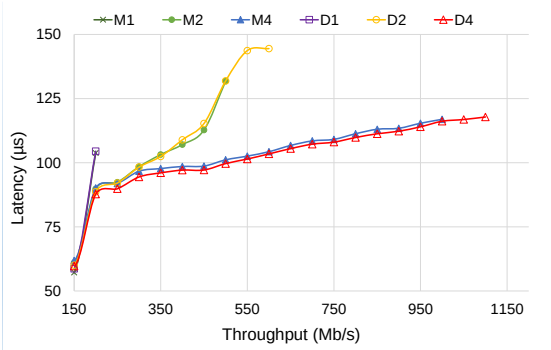

Figure 5. BMV2 DDoS mitigator latency and throughput: main (M) and docker (D) instances

storing the TCP port of the current session packet and retrieving the relative session stateful values as packet metadata. The scan condition (i.e., an incremental port pattern) determines a pipeline switch where, upon a given threshold, the packet is dropped, or subject to selected counter-actions, such as mirroring to a monitoring element in charge of intelligent correlations.

5 VIRTUALIZED P4: PERFORMANCE AND DISCUSSION

In order to show the P4 potentials in terms of performance in virtualized scenarios, the most common software switch employing the P4 language, the Behavioral Model version 2 (BMv2)[13], was employed implementing the P4 DDoS mitigator program (around 100 lines of code) over a standard Linux box server (Intel Xeon CPU E5-2620 6-core 2.10GHz, 16GB RAM). In the *Main* (M) configuration the switch runs on the PC physical instance, while in *Docker* (D), runs as a virtual container. The switch is loaded with 1000 flow entries and attached to 10 Gigabit Ethernet optical interfaces I1 and I2. TCP traffic (frame length 1500 byte) generated by the Spirent N2U Traffic Generator and Analyzer is injected in I1 and received by I2 crossing the BMv2 instance.
Fig. 5 shows the throughput and the latency sustained by the switch as a function of the number of utilized cores (i.e., 1, 2 and 4). The maximum sustainable throughput is evaluated by considering the maximum tolerated packet loss ratio set to 10^{-4} and is reflected in the range of each plot. Results show that, using 4 cores, the switch is capable of sustaining more than 1Gb/s traffic while applying the DDoS mitigator, introducing an average latency below 150 µs. The utilization of a reduced number of cores impacts the maximum throughput linearly (1 core achieves around 200Mb/s, 2 core around 500Mb/s) and the latency (reaching 105µs and 140µs approaching the maximum throughput, respectively). This is because the software switch process always runs 4 parallel threads. A significant result is derived by comparing the main and the virtual container instances. In fact, the docker instance does not impact negatively on networking performance. Indeed, a slight improvement was observed due to a better thread scheduling among the available cores, which limits the 100% CPU load events that are at the basis of the BMV2 packet drop statistics. In light of this, the docker version achieves an improvement of 100-150 Mb/s of additional sustainable throughput. Moreover, docker assures no latency degradation, rather showing a slight improvement with a higher number of available cores serving BMV2 threads. Such effect was analyzed in the literature [14], confirming that containers are particularly suitable for virtualizing P4 stateful switches.

A) P4 HIGH PERFORMANCE SWITCHES

In order to prove that the virtualized P4 application can run at the 10G+ speed the DDoS mitigator was also tested with P4rt-OVS, a programmable virtual switch derived from the Open Virtual Switch (OVS) implementation [15]. The test setup was the same as for BMv2, but the P4 program was adapted to make it compatible with the architecture model of P4rt-OVS.

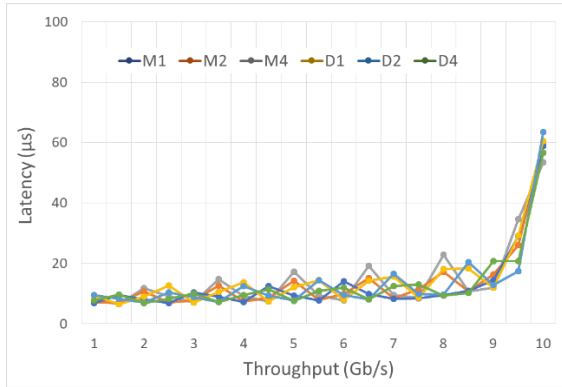

Figure 6. P4rt-OVS DDoS mitigator latency and throughput: main (M) and Docker (D) instances

Fig. 6 shows the throughput and latency observed for P4rt-OVS. Due to the fact P4rt-OVS is based on DPDK to poll packets from an interface, the measured latency

is comparable for every test scenario. The latency behavior is similar to results obtained with FPGAs, with stable floor not exceeding 20μs (using FPGA the floor was around 5us) under 90% interface nominal speed utilization [6]. However, similarly to BMv2, the latency observed for Docker was slightly lower than for physical machine, thus confirming the effectiveness of containerized solutions. It is worth to note that the test session successfully achieved the link speed (10G). Other tests confirmed wire speed performance at even 40G for flows having a packet size of 1500 bytes.

B) APPLICABILITY AND OPEN ISSUES

General purpose hardware equipped with P4 software switches is able to support stateful DPI with no packet loss. As results suggest, the P4 virtualized solution is effective and achievable at the 10 Gigabit/s scale and beyond for P4rt-OVS with latency ranges (i.e., 20μs) typical of legacy hardware switches, with the advantage to be deployed, migrated and attached on-demand to the network resorting to a container-based orchestrator (e.g., Kubernetes) and a P4-enabled SDN controller (e.g., ONOS). This is particularly promising for large-scale applications running at the 5G edge (e.g., smart city apps, video recognition, sensors for automotive), where selected traffic load (e.g., IoT gateway tenant, vertical industry slice) may be processed with zero-cost open-source P4 virtual function. Improved software switch results (e.g., throughput) are achievable by using hardware acceleration. However, the P4 software switch/container applicability may not always be directly extendible to central cloud systems (e.g., large DC) and the core network. In fact, P4 and hardware-based solutions are already in place to sustain 100Gb/s and beyond throughput exploiting programmable bare metal switches and FPGA. The same DDoS mitigator program running on a Xilinx FPGA performs around 5 μs latency with extremely stable performance [7], however at the expense of an increased hardware equipment cost.

Open issues and applicability extensions are related to the development and the success of the P4 language. Novel P4 features such as packet generation, direct and fine queue management for traffic congestion control, extension and improved management of stateful objects, novel primitive action definitions, flexible definitions of the PSA internal architecture, may pave the way for a wider and potentially massive adoption in edge/fog and industrial networks. In particular, referring to next generation industrial networks and the advent of 6G, the availability of high precision queue management and deterministic latency bound behavior, such as in Time Sensitive Networks (TSN), directly mapped as performance functions in both P4 language and related virtual switch platform environments, will probably play a significant step towards a fully converged, programmable and autonomic high performance softwarized networking.

6 CONCLUSIONS

This paper showed the significant potential of P4-based data plane programmability that will enable flexible and advanced programmable functions in a SDN/NFV edge scenario serving IoT, 5G and metro networks in a multi-tenancy environment. Novel P4 programs enforcing stateful capabilities will open the way to the development of novel services and applications with stringent requirements in terms of latency, QoS and autonomic reactions upon complex network events directly at the data plane. Assessments of use cases and results show the applicability of P4 switches even as virtual container in the future 5G edge nodes, demonstrating the flexibility and the effectiveness of such technology to boost, integrate and complete the SDN/NFV paradigm.

ACKNOWLEDGMENT

This project has received funding from the ECSEL Joint Undertaking (JU) BRAINE Project under grant agreement No 876967. The JU receives support from the European Union's Horizon 2020 research and innovation programme and from MIUR (Italy).

■ REFERENCES

1. K. Samdanis and T. Taleb. "The Road beyond 5G: A Vision and Insight of the Key Technologies." IEEE Network 34.2 (2020): 135-141.
2. P. Bosshart, et al., "P4: Programming protocol-independent packet processors," SIGCOMM Conf. 2014.
3. G. Bianchi, M. Bonola, A. Capone, and C. Cascone, "OpenState: Programming platform-independent stateful OpenFlow applications inside the switch," ACM SIGCOMM Conf. 2014.

4. P. Bosshart et al., “Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN,” ACM SIGCOMM Conf., 2013.
5. M. Moshref et al., “Flowlevel state transition as a new switch primitive for SDN,” in Proc. 3rd Workshop Hot Topics Softw. Defined Netw. (HotSDN), Chicago, IL, USA, 2014, pp. 61–66.
6. F. Paolucci, F. Civerchia, A. Sgambelluri, A. Giorgetti, F. Cugini, and P. Castoldi, "P4 Edge Node Enabling Stateful Traffic Engineering and Cyber Security" J. Opt. Commun. Netw. 11, A84-A95 (2019).
7. F. Cugini, P. Gunning, F. Paolucci, P. Castoldi, and A. Lord, "P4 In-Band Telemetry (INT) for Latency-aware VNF in Metro Networks," OFC 2019, OSA, M3Z.6.
8. J. Hyun et al., "Towards knowledge-defined networking using in-band network telemetry," NOMS Conf. 2018.
9. Nagai R et al "Design and Implementation of an OpenFlow-Based TCP Syn Flood Mitigation" MobileCloud Conf. ‘18.
10. T. Dargahi, "A Survey on the Security of Stateful SDN Data Planes", IEEE Commun. Surv. & Tut.,19 (3), 2017.
11. F. Musumeci et al, “Machine-learning-assisted DDoS attack detection with P4 language”, ICC 2020.
12. C. Shen et al, “A Programmable and FPGA-accelerated GTP Offloading Engine for Mobile Edge Computing in 5G Networks”, IEEE INFOCOM 2019.
13. “BMV2, Behavioral Model version 2”, https://github.com/p4lang/behavioral-model.
14. J. Struye et al., "Assessing the value of containers for NFVs: A detailed network performance study," in Proc. CNSM 2017.
15. T. Osiński et al., “P4rt-OVS: Programming Protocol-Independent Runtime Extensions for Open vSwitch using P4,” in Proc. IFIP Networking 2020.

Francesco Paolucci received the M.S. degree in Telecommunications Engineering from the University of Pisa, and the Ph.D. degree from Scuola Superiore Sant’Anna, Pisa, Italy, in 2009. In 2008 he was granted a Research Merit Scholarship at the Istitut National de le Recherche Scientifique (INRS), Canada. Currently, he is Senior Researcher at CNIT, Pisa, Italy. His main research interests are in the field of advanced SDN network control plane, network fault tolerance, inter-domain traffic engineering, data plane programmability, security and confidentiality. He is co-author and contributor of 2 IETF Internet Drafts, more than 160 international publications and filed 4 patents. He is co-author of the “5G-PPP Phase 1 Security Landscape” released by the 5G-PPP Security Working Group.

Filippo Cugini received the M.S. degree in Telecommunication Engineering from the University of Parma, Italy. Since 2001, he has been with the National Laboratory of Photonic Networks, Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), Pisa, Italy. His main research interests include theoretical and experimental studies in the field of communications and networking. In particular, the focus is on Ethernet, GMPLS and PCE protocols and architectures, software defined networking (SDN), Segment Routing, OpenFlow and P4. He is an IEEE member and he is co-author of 14 patents and more than 200 international publications.

Piero Castoldi (PhD) has been Professor at Scuola Superiore Sant’Anna, Pisa, Italy since 2001. He spent abroad at Princeton University (USA) overall about two years in 1996, 1997, 1999, 2000. He is currently Leader of the “Networks and Services” research area at Scuola Superiore Sant’Anna. His research interests cover telecommunications networks and system both wired and wireless, and more recently reliability, switching paradigms and control of optical networks, including application-network cooperation mechanisms for cloud networking. He is an IEEE Senior Member and he is author of more than 400 international publications.

Tomasz Osiński works as an R&D Expert at Orange Labs Poland. He received a B.Sc. degree and a M.Sc, degree in telecommunications from Warsaw University of Technology, in 2016 and 2018. His main research areas are Network Functions Virtualization and Software-Defined Networking including cloud-native VNF’s design, virtualization performance, network orchestration and programmable data plane. Currently, he works for his PhD focusing on leveraging programmable data plane and the P4 language to accelerate cloud-native network functions.