The use of P4 for 5G networks

MC Nkosi, AA Lysko
Meraka Institute, Council for Scientific and Industrial Research (CSIR),
Advanced Networks and Architecture systems Research group,
Meiring Naude Road, Lynnwood,
0001, Pretoria, South Africa
E-mail: {*mnkosi2, alysko*}*@csir.co.za*

***Abstract*—5G networks are not only the next mobile generational networks but, the future networks of "*everything*" which are expected to offer new services and business opportunities on universal level through improved performance, and flexibility. These networks will combine various access technologies such as wireless, mobile, satellite, and fixed optical access networks to offer reliable performance and improve network coverage. SDN and NFV are the key enabling technologies for 5G networks. This paper describes use Programmable Protocol-independence Processors (P4) as the main language for programming the forwarding plane in SDN and NFV.**

Keywords—SDN;NFV;P4;OpenFlow;5G.

I. INTRODUCTION

Fifth Generation (5G) is the next mobile generation network that enables innovation and evolution just beyond mobile internet. 5G promises to integrate telecom and information technology into a universal infrastructure by connecting mobile and fixed access networks [1]. 5G combines networking, computing and storage resources into a flexible programmable joined infrastructure. The 5G networking is based on Software Defined Networking (SDN) and Network Function Virtualisation (NFV) technologies.

SDN is about separating the control plane from the data plane and making the control plane open and programmable. SDN is largely based on the OpenFlow protocol which is a standard for remotely programming the forwarding plane of the network devices. The OpenFlow protocol overtly specifies protocol headers on which it operates and these headers have been growing to account for new functionality, with each new version of OpenFlow [2]-[3]. The increase of headers has endorsed the complexity of the OpenFlow specification without offering flexibility to add new headers.

NFV is about moving the data plane from physical hardware to virtual machines. NFV is based on the use of hypervisors which employ virtual switches to sends packets between virtual machines [4]. The virtual switch is usually thousands or more lines of code and thus, changing the switch may require network protocol design, developing, testing and maintaining a large code. This complexity may ultimately reduce the efficiency of new feature implementation and increase cost of performance of software switches [5].

This work analyses the Programming Protocol-Independent Packet Processor (P4) with regards to:

- Use of P4 alongside OpenFlow to enable flexibility and protocol-independence without being tied to any specific network protocol in SDN. Most importantly, the use of P4 alongside OpenFlow for SDN in 5G networks is highlighted.
- The use of P4 as a domain-specific language for protocol independent software switches in NFV to enable the separation of custom protocol implementations from switch code in order to avoid the complexity that comes with protocol "tied" software switches. In particular, Open virtual Switch (*OvS*), which is based on OpenFlow and is widely used in NFV, is compared to P4 based software switches.

The rest of this paper is organized as follows: section two provides an overview on OpenFlow, section three describes P4 as used in SDN, and section four compares the OpenFlow *OvS* switch to P4 switch for NFV. Section five concludes the paper.

II. OPENFLOW IN SDN

OpenFlow (OF) is a widely adopted and standardized protocol used in SDN to enable the programming capability of the forwarding devices in SDN's data plane. As depicted in Fig.1, the OF based SDN network is made up of an OF SDN controller, which controls the forwarding devices through OF protocol. OF was introduced to disaggregate firmware from hardware, so as to use open source firmware which promotes more control over features and innovation. OF was founded by Open Networking Foundation (ONF) which is a non-profit consortium that promotes the adoption of SDN.

OF is based on match and action dataflow concept which, is implemented and fixed into OF switches. Using the match/action structure, as summarized in Fig 2, each incoming packet in the ingress port is matched against a set of group tables. Each flow entry in a group table consists of a match field, priority, counters, instruction, timeouts, cookies, and flags. When a match is found, associated actions are executed. Otherwise a packet may be dropped. The matching is based on the OF header fields such as, to mention a few, MAC address, TCP/UDP port number, IP address and etc. [6].

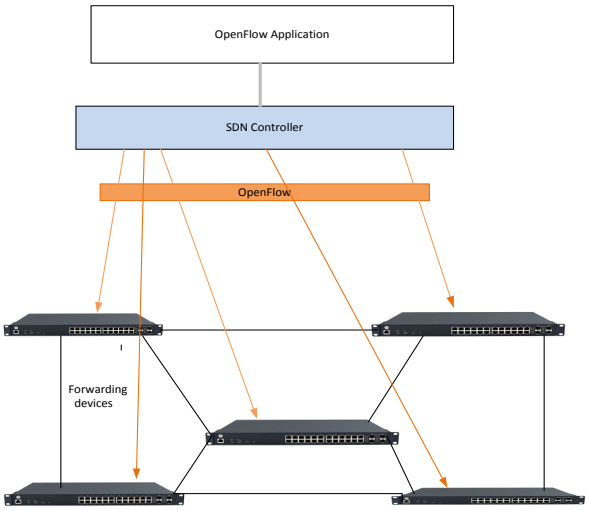

Figure 1: OpenFlow components

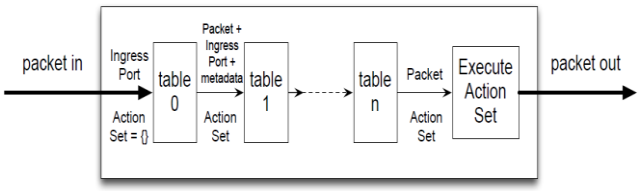

Figure 2: OF packet process pipeline [6]

Introduction of new features (for example, support for MPLS, IPv6) in OF brought about a dramatic increase of header fields in later versions, for example, from just 12 header fields in OF 1.0 to 41 header fields in OF 1.4. The increase in header fields complicated the OF specification [3].

Although OF has introduced programmability of the data plane, it does not support custom protocols , the data plane is limited to OF existing header fields, and Switches are tied to protocol headers. Therefore, OF SDN networks cannot adapt to network demands as required. With 5G networks expected to be a mix of everything, it is necessary for SDN to be flexible enough to succumb to user demands and support the expected evolution of the data plane. The next section describes P4 as an alternative to OF for SDN.

III. P4 IN SDN

P4 is a domain specific programming language designed for the data plane [7]. With P4, the SDN controller is given full capability to tell a switch how to process packets instead of being limited to a fixed switch design. P4 is designed with the objective to support reconfigurability, protocol and target independence. With reconfigurability, the controller is able to re-define packet processing in the fields. Protocol independence ensures that the switch is not tied to any protocol. With target independence, the controller does not need to know the details and specification of the switches in the data plane.

Work that proposes protocol independence in SDN exists [8] – [11]. Cohn et al [8] introduced abstract forwarding for OF but did not include compiler processing. Protocol –oblivious forwarding was introduced by Song [9] but was focused on network processors rather than data plane. Raju et al [10] introduced NOSIX which, defines the flexibility of match/action packet processing but does not include protocol independence. Barbette et al [11] focused on the click modular router which also defines the flexibility of match/action packet processing. However, the click modular router does not support different hardware switches. This paper, recommends P4 because it supports both protocol independence and flexibility of packet processing for the SDN data plane.

A. P4: Abstract Forwarding Model

P4 provides a joined way to build networks using a mixture of programmable and fixed-function switches in SDN and allows for easier introduction of new features and protocols in software than new hardware. P4 is a program that configures the forwarding behavior of the data plane. As compared to OF switches, which are hard wired protocol header switches that assume packets in a specific format, P4 switches come with a compiler installed. The compiler translates programs into target-specific representation. As in fig.3, P4 abstract forwarding model employs configure and populate operations. The configuration operation determines the processing of packets and protocols supported in a switch. P4 is used to dynamically program a parser graph whose state may be stored in Ternary Content address memory (TCAM).

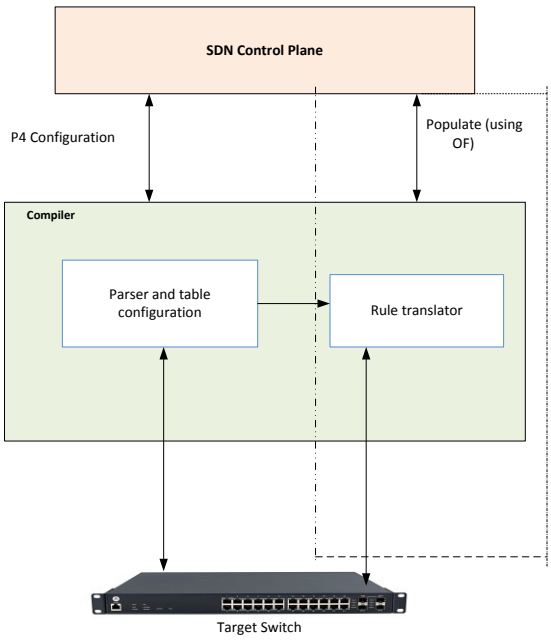

Figure 3: P4 abstraction model

P4 also configure the match/action instructions together with header fields to be processed. The populate operation adds and deletes entries of the match/action tables that were configured during the configuration operation. The populate operation can be performed using the normal OF. The P4 abstract model separates populate and configuration operations to support the reconfiguration of a switch without any disruption in the data plane.

As compared to OF, P4 offers flexible data plane which makes it easier to support introduction of new protocols and features with ease. Table 1 summarizes the main differences between OF and P4 in SDN. The next section describes the use of P4 in Network Function Virtualization (NFV).

IV. P4 IN NFV

As one of the key enabling technologies for 5G, NFV enables the implementation of software-based network functions on commodity hardware by using softwarization techniques. NFV relies heavily on hypervisors which, use software switches to move packets to and from virtual machines. To support new protocol headers and to improve debugging features, the software switches require continuous upgrading and customization [12]. Since software switches are based on large code, changing the code requires an extensive understanding of network protocol design which may be complex. In this section, we look at Open virtual Switch (*OvS*) as an OF-based virtual switch in NFV and describe benefits of using P4 in software switches.

A. Open vSwitch

OvS is an OF-based software switch widely used in virtual environments. It employs necessary protocols such as, just to mention few, Ethernet, VXLAN, and NVGRE for virtual platforms [12]. As an SDN and OF switch, *OvS* relies on the controller to define its behavior. As described above, OF defines behavior of a switch based on a series of match/action rules. However, the behavior defined based on OF match/action rules tends to limit the performance of the switch because, each packet that enters the switch has to go through a couple of match/action tables which needs a general packet classification. Therefore, *OvS* relies on cache to achieve better forwarding performance [13].

B. Use of P4 in Software switches

To define a new approach to perform software upgrade and customization with great ease and flexibility, P4 program specifies packet headers to parse and the structure of match/action. The underlying software substrate is a generic engine augmented to parse, match and executes instructions as the program specifies. The P4 Protocol independence offers the following benefits [7]: (1) ease of adding new features: new standard or private protocols can be easily added to a P4 program, compiled and deployed quickly; (2) removing a standard protocol header: removing an unused protocol is as simple as removing unused portion of program code; (3) addition of visibility: addition of new protocols and actions to collect the state of the switch so as to understand network behavior and operation conditions.

Table 1: OF vs. P4

OpenFlow	**P4**
Assumes a fixed parser	Supports programmable parser to allow new headers to be defined
Assumes the match/action stages in series	The match/action can be in parallel or series
Actions are constrained by a fixed switch design	Actions are composed from protocol-independent primitives supported by the switch

Choi et al [14] developed a software switch which was derived from *OvS*. In their work, they compared P4 based software switch with an OF-based *OvS* in terms of complexity, forwarding performance. Their P4 based switch performed better in performance and the code is 40 times shorter as compared to that OF *OvS*.

V. CONCLUSION

SDN and NFV are the key enabling technologies for development of 5G networks. SDN is based on OpenFlow as the standard protocol for defining the behavior of the underlying forwarding devices. However, the standard protocol brings about limitation, such as protocol-dependence, inflexibility in introducing new features, to forwarding devices. In this paper, the use of P4 as a high level programming language for SDN forwarding plane and Software switches for NFV platforms was described. The benefits of using P4 were also highlighted, so as to ensure that flexibility in SDN data plane and NFV platforms is achieved.

For future work, P4 with be explored further with regards to reliability of the SDN data plane.

REFERENCES

[1] 5G PPP SN Working Group, "Vision on Software Networks and 5G," *5G-PPP Initiat.*, vol. 2017, no. January, pp. 1–38, 2017.

[2] Han, S., Jang, K., Panda, A., Palkar, S., Han, D. and Ratnasamy, S., 2015. Softnic: A software nic to augment hardware. *Dept. EECS, Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2015-155*.

[3] Thyagaturu, A.S., Mercian, A., McGarry, M.P., Reisslein, M. and Kellerer, W., 2016. Software defined optical networks (SDONs): A comprehensive survey. *IEEE Communications Surveys & Tutorials*, *18*(4), pp.2738-2786.

[4] Palkar, S., Lan, C., Han, S., Jang, K., Panda, A., Ratnasamy, S., Rizzo, L. and Shenker, S., 2015, October. E2: a framework for NFV applications. In *Proceedings of the 25th Symposium on Operating Systems Principles* (pp. 121-136). ACM.

[5] Perino, D., Gallo, M., Laufer, R., Houidi, Z.B. and Pianese, F., 2016, April. A programmable data plane for heterogeneous nfv platforms. In *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on* (pp. 77-82). IEEE.

[6] Version, O.S.S., 2015. 1.5. 0 (Protocol version 0x06). *Open Networking Foundation*.

[7] Sivaraman, A., Kim, C., Krishnamoorthy, R., Dixit, A. and Budiu, M., 2015, June. Dc. p4: Programming the forwarding plane of a data-center

switch. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research* (p. 2). ACM.

[8] Yadav N, Cohn D. OpenFlow Primitive Set. Google Inc., External Version: 0.1, Created Jul. 2011 Jul.

[9] Song, H., 2013, August. Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking* (pp. 127-132). ACM.

[10] Yu, M., Wundsam, A. and Raju, M., 2014. NOSIX: A lightweight portability layer for the SDN OS. *ACM SIGCOMM Computer Communication Review*, *44*(2), pp.28-35.

[11] Barbette, T., Soldani, C. and Mathy, L., 2015, May. Fast userspace packet processing. In *Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for networking and communications systems* (pp. 5-16). IEEE Computer Society.

[12] Rosa RV, Rothenberg CE. Taking Open vSwitch to the Gym: An Automated Benchmarking Approach. InIV Workshop pre IETF/IRTF 2017 Jul.

[13] Singh T, Jain V, Babu GS. VXLAN and EVPN for data center network transformation. InComputing, Communication and Networking Technologies (ICCCNT), 2017 8th International Conference on 2017 Jul 3 (pp. 1-6). IEEE.

[14] Cidon, E., Choi, S., Katti, S. and McKeown, N., 2017, August. AppSwitch: Application-layer Load Balancing within a Software Switch. In *Proceedings of the First Asia-Pacific Workshop on Networking* (pp. 64-70). ACM.