Proceedings of the 10th International Conference on Applied Informatics
Eger, Hungary, January 30–February 1, 2017. pp. 263–270
doi: 10.14794/ICAI.10.2017.263

Programmable Data Planes: A survey

Mohammed Fekhreddine Seridi, Sándor Laki

Eötvös Loránd University, Faculty of Informatics
Department of Information Systems
{seridi,lakis}@inf.elte.hu

Abstract

Networking is getting more complex because networks are rigid and don't permit innovation. Software Defined Network SDN gave more freedom for the network evolution, since it separates the Control plane and the data plane, and create new layering in the network. In SDN the controller is programmable whereas the data plane is not and this causes some limitation in the functionality and complexity. As a solution for this issue the Data Plane Programmability (DPP) was suggested. Some important DPP technologies such as OpenFlow-Protocol independent (OF-PI) and Programming Protocol independent packet processing (P4) and Protocol Obvious forwarding (POF) are good choices to improve the packet processing and networking in general.

Keywords: SDN, Protocol independent, programmable data planes, Packet forwarding

MSC: A68M10

1. Introduction

Networking has been known as a complex task, from handling a lot of protocols to configure the interfaces of many devices and ensure a reliable routing protocol. All these tasks are maintained in the same time by network engineers and administrators. The task become harder in case of implementing new protocol or to edit the existing one to satisfy a user need. This complexity is due to the fact that the network is one rigid stack; although the OSI model gives a good layering, many functions in networking are hard to perform such as: implementing new protocols, controlling the network and even monitoring and measuring some parameters. To solve this problem, the separation between the control and the forwarding in the devices was suggested. This separation led to a centralized control and decoupled forwarding plane. And this gave the network more flexibility by creating a logical layering. This basic separation referred to Software Defined Networking (SDN) [1][2][3]. where a programmable control specifies the forwarding behavior

of the data plane through APIs (e.g.: OpenFlow [1]). Another problem is that the bottom-up design due to the fixed-function ASIC in the data plane that imposes some limitations because of the fixed hardware. At this stage, comes the idea of programmable data planes. For this goal some NPU, CPU and FPGA based hardware were used in addition to Protocol Independent Switch Architecture (PISA)[2]. In this issue some initiative such as: protocol independent Forwarding (PIF), Protocol Obvious Forwarding (POF), OpenFlwo Protocol Independent OF-PI and the widely used and the more dominant nowadays the Protocol Independent Packet Processing (P4) language[4].

In this paper, we survey the before mentioned works done on data plane programmability, because it is important to have a clear vision about this topic especially that the surveyed items are interrelated and have some similarities. In next section we will mention the important points in SDN since it is the general paradigm that leads to the DP programmability. In section 3 we will talk about each DPP and its technologies. Section 4 concludes this paper.

2. Background

Communication networks suffer from the problem that they find difficulties in evolution. In case of implementing new features in the network or improve some functionalities[5]; a change in the hardware is not avoidable[4]. This type of networks is called hardware centric networks. A common example here is the IPv6 protocol implementation it took decades since it was suggested and it is not practically in use. To avoid this type of problems a new architecture considering the network devices should be introduced. It should be a top-down design[6], in which the user needs and the application layer specifies the way the forwarding devices work. So it is clear that the reason of this architecture is to provide more flexibility and permit innovation to enter to networking and better controlling and monitoring the network performance. This is thanks to two things. First, the separation of the forwarding plane (data plane) and the control plane. The second is the programmability of the control plane and hence the name Software defined networking.

2.1. SDN architecture

Basically, the SDN architecture consists of three main layers; the application layer, the control layer and forwarding layer as shown in figure (1).

2.1.1. Application Layer

The application layer is in the top of the SDN architecture. It performs networking tasks such as: network management, traffic engineering, measurement and monitoring, security, manage middle-boxes and virtualization. It uses the services provided by the controller.

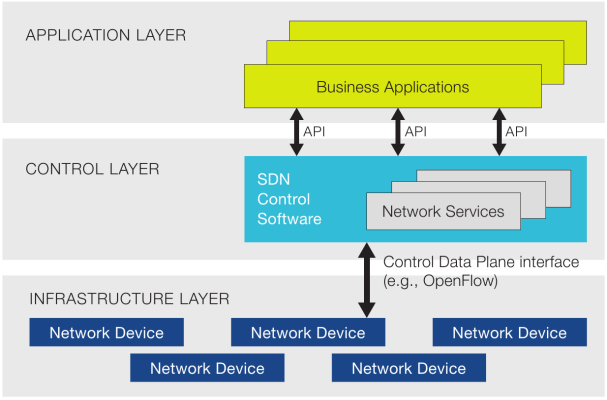

Figure 1: SDN architecture [2]

2.1.2. Control Plane

The control plane is centralized (figure 2) and communicates to deferent switches or the so called forwarding elements through APIs such as OpenFlow and ForCES a comparison between the two is found here[7] The control-switch connection is known as the southbound interface [1].

2.1.3. Data Plane

In SDN and most southbound APIs the data plane is considered as a set of fixed flow tables where rules, such as the Openflow rules, are executed and packets are processed and forwarded. There are some software switch implementation e.g.: OpenVSwitch [8] CPQD ofsoftswitch13 [8] both designed for OpenFlow.

2.2. Remarks about SDN

SDN strongly relied on OpenFlow. In fact, OpenFlow rules are more complicated than the traditional IP routers [1]. Increasing the rule set leads to more complexity [4]. The solution for this issue was to have more flexible data plane to reduce the complexity and have a top-down architecture. For deep information about SDN these references [1] [2] [3] are detailed surveys about the topic.

3. Data plane programing technologies

The data plane programmability aims to create a tool that will be used to tell the switch how to process the coming packets. one way is to make the switch behavior controllable and configurable and create an abstraction of the hardware with the behavior of the switch. Another alternative is use a high-level programing language (HLL) such as P4[4]. One of the drawbacks of OpenFlow is the increasing

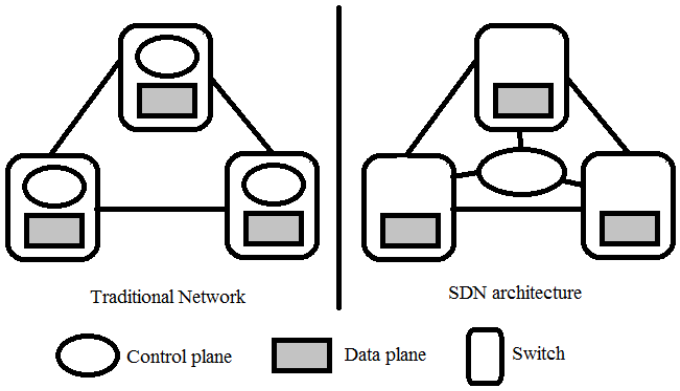

Figure 2: Hardware use in traditional and SDN architectures

number of protocol that leads to more complexity. An idea is to make the data plane protocol independent and this is a common key in the data plane programing techniques.

3.1. Protocol Obvious Forwarding POF

POF was suggested in [9] it aims to make the forwarding elements white boxes because the black box networking model make many difficulties and reduce the chances of innovation for the network administrators.
POF relies on Flow Instruction Set FIS to make the data plane more flexible. Also POF has software and hardware prototype that can be found here[10] [11].
POF focus on what the primitive instruction should be. Whereas P4 focus on how to express the packet processing methods in a program and compile it using the POF-FIS.
The main important concepts of POF are as follow[10]:
Matching Fields: They are defined by the tuple <offset, length> where the offset is the number of bits to be escaped from the start location of the packet. And the length is the number of bits of that matching field.
POF-FIS: Are the operations to be made on the packet such as generic adding, modifying, deleting fields. And other complex instructions. This instruction set use the match fields to determine which instruction to be executed.
Flow tables: Are tables where each match field is stored and attached to instructions form the FIS. POF has four types of tables: Masked-Match (MM), Longest-Prefix-Match (LPM), Extract-Match (EM) tables and Direct Table (DT). DT table contains only instructions. These tables create a pipeline that make the network programmable and flexible.
Metadata Memory: Is the memory in which information about the packet is kept. It is generated by the table processing the packet and will be used by the next tables in the pipeline.

3.2. OpenFlow Protocol independent OF-PI

OF-PI is an integration of P4 and POF into OpenFlow[12]. OF-PI refers to system in which P4 or POF program and compiler are used to configure the switch and set its behavior. In the run time OpenFlow protocol is used for the south bound communication and OpenFlow rules are kept in this communication. OF-PI main contribution is the addition of Protocol independent Layer(figure 3) that will interpret any protocol (new or existing) and make the forwarding plane capable of processing the packets with these protocols.

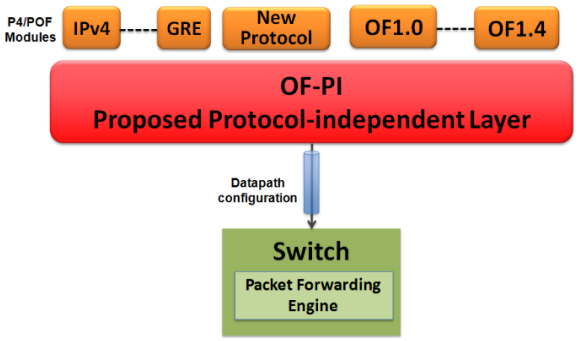

Figure 3: OF-PI additional Protocol independent Layer[12]

OF-PI has three principals [12]:
OF-PI should be protocol neutral.
OF-PI should help create a software development ecosystem.
OF-PI supports existing OpenFlow specifications.

3.3. P4: programming Protocol Independent Packet Processing:

P4 is a high-level and domain specific language suggested in [4]. Later the P4 consortium [13] was created with contribution of leading companies in telecommunication industry and academic institutions. P4 had a large interest among researchers and there have been many implementations and demos [14]. There are two versions of P4, $P4_{14}$ and $P4_{16}$ the main deference is in the syntax and some additional predefined libraries [15] in this section we will present the general architecture model and some syntax example of $P4_{14}$ specification V1.0.4 to find more about the previous versions see [15]. P4 is used to program the data plane; as any programming language, it has a compiler. The compilers create a High-level Intermediate representation (HLIR) and output a code for example C code that will be executed in the hardware by a Network Operating System. For deep information about P4 compilers check [16][17].

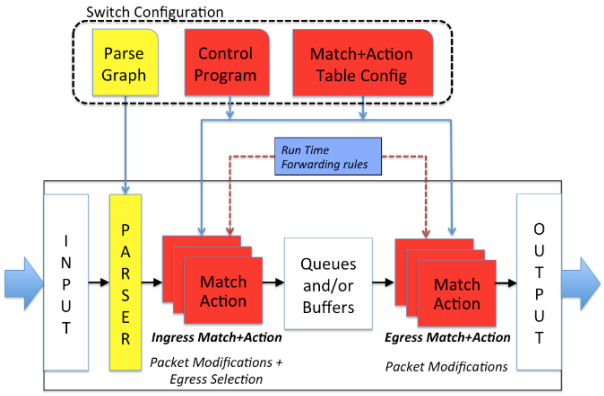

Figure 4: P4 abstract forwarding model [15]

3.3.1. P4 architecture model

The structure of P4 program is as follow[15]:
1 Header type declaration: in this part the headers used in the packets are declared. The header declaration consists of the fields in the header and their length in bits. The order of the fields is important. The header length should be a multiple of 8 bits. Also some metadata can be declared in the same way, or use standard metadata in the program. After declaring the header types, header instances are created.
2 The parser: defines how to interpret the received packet. By looking at some field of the previous headers. For example, if the packet start with a Ethernet header the next bytes are interpreted depending on the value of Ethernet type field in the Ethernet header. This make the P4 protocol independent since there is no fixed structure for the headers in the packet.
3 Table definition: In P4 table is a tool to list the possible actions to be performed on the packet and the reading based on which the actions are selected. In addition to the size of the table.
4 The actions: are the operations to be made on the packet headers and fields. There are primitive actions such as addition and subtraction, modify field, delete field, add new field. And compound actions which are combination of a set of primitive actions.
5 Control: flow and pipeline layout: this part describes in which order the deferent tables are implemented and executed. And the way in which the arriving packets will flow in the pipeline after the parser.
In P4 we distinguish between two stages configuring time and run time. In configuring time the P4 program is compiled and executed in the switch; and in the run time, the forwarding rules are implemented in the look up tables. Figure 4 shows the abstraction of a P4 programmable switch.

P4 is a target independent since the program is general and don't look to the

details of the running hardware and the compiler take care about how to execute the p4 program in a specific hardware. In ELTE P4 compiler [p4cop] a Hardware Abstraction Library HAL was added so that the output of the compiler will fit the required hardware. Run time Reconfigurability in P4 programmable forwarding devices is possible because any change in the network (configuration or topology) can be handled by the controller by just updating the look up table and keeping the forwarding device working as it was programed previously.

3.4. Discussion

POF was the first introduction of the protocol independent concept and the first initiative towards DPP, because of how it works since it gives a way how to parse packets and process them with flow tables implemented in a pipeline architecture. P4 has the same goals and offers the same results and add the feature of programing language that can be used to communicate to the forwarding devices and describe the packet processing function. P4 is evolving and more features are added to this DSL. OF-PI uses either POF or P4 and add a protocol independent layer and keep the OpenFlow southbound communication. Using P4 it is hard to implement traffic management.

A controller that can support these data plane programmability schemes is needed. That is why in many cases the OpenFlow southbound communication is kept even if these technologies are considered a developed version of OpenFlow.

Actually, there is no specific technology named PIF. The term PIF is used to refer to the concept behind OF-PI and P4. There are other initiatives such as Flare the deep programmable network project [18]. They not discussed in this works because we want to give the main ideas behind DPP. OpenFlow protocol is not discussed here because it is considered as a communication protocol between the Controller and the Data Plane and not a DPP technology.Also the hardware architecture is an important point in DPP that we will cover in later works.

4. Conclusion

SDN paradigm make a revolution in networking because it brings some core concepts to the field. An important one is the decoupling of the control and data planes; this offer more flexibility, abstraction and layering. To make the network field and industry successful we must make the possibility of innovation by supporting the data plane programmability idea. DPP gives independence from the underlying hardware because it supports the modularity of the networking elements and it makes the packet processing and data flow independent from the protocols used. Compared to the software industry, networking is in it is early stages and DPP is the best way to rapidly increase its development.

References

[1] Bruno Nunes Astuto, Marc Mendon ca, Xuan Nam Nguyen, Katia Obraczka, Thierry Turletti A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *Communications Surveys and Tutorials IEEE Communications Society* 2014, 16 (3), pp. 1617–1634.

[2] Hamid Farhady, HyunYong Lee , Akihiro Nakao, Software-Defined Networking: A survey, *Elsevier Computer Networks.*, Vol. 81 (2015), pp. 79–95 .

[3] Rahim Masoudi and Ali Ghaffari Software defined networks: A survey, *Journal of Network and Computer Applications.*, http://dx.doi.org/10.1016/j.jnca.2016.03.016

[4] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, P4: programming protocol-independent packet processors, *SIGCOMM Comput. Commun. Rev.*,vol. 44, no. 3, pp. 87-–95, 2014.

[5] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz, Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN, *SIGCOMM* ,pp. 99-–110, 2013

[6] Open Networking Foundation, "ONF SDN Evolution ver 1.0 Sep. 2016"

[7] T. Tsou, X. Shi, J. Huang, Z. Wang, X. Yin,Analysis of Comparisons between OpenFlow and Forces., https://tools.ietf.org/html/draft-wang-forces-compare-openflow-forces-01

[8] OpenVswitch website http://openvswitch.org/

[9] Shengru Li, Daoyun Hu, Wenjian Fang, Zuqing Zhu Protocol Oblivious Forwarding: Unleash the Power of SDN through a Future-Proof Forwarding Plane *SIGCOMM HotSDN Workshop.*, Aug. 2013.

[10] H. Song, Source Routing with Protocol-oblivious Forwarding (POF) to Enable Efficient e-Health Data Transfers *Communications (ICC), 2016 IEEE International Conference.*, Jun. 2016.

[11] POF website http://www.poforwarding.org

[12] OFPI ProtocolIndependent OpenFlow v.1.0 ONF Sept. 2014.

[13] The P4 language consortium website http://www.p4.org/

[14] P4 talks, demos and tutorials ACM-SIGCOMM 2016 http://p4.org/p4/p4-papers-talks-demos-tutorial-labs-acm-sigcomm-2016-florianopolis-brazil/

[15] $P4_{16}$ specification V1.0.0 P4 consortium https://p4lang.github.io/p4-spec

[16] ELTE P4 Compiler https://github.com/P4ELTE/t4p4s

[17] P4 Consortium compiler https://github.com/p4lang/p4c

[18] Akihiro Nakao FLARE: Open Deeply Programmable Switch *The 16th GENI Engineering Conference.*, 2012 http://www.ieice.org/~nv/nvs2013/nvs3-is6-nakao.pdf