Next-Generation SDN and Fog Computing: A New Paradigm for SDN-Based Edge Computing

Eder Ollora Zaballa
DTU Fotonik, Technical University of Denmark, Lyngby, Denmark
eoza@fotonik.dtu.dk

David Franco
Department of Communications Engineering, University of the Basque Country UPV/EHU, Bilbao, Spain
david.franco@ehu.eus

Marina Aguado
Department of Communications Engineering, University of the Basque Country UPV/EHU, Bilbao, Spain
marina.aguado@ehu.eus

Michael Stübert Berger
DTU Fotonik, Technical University of Denmark, Lyngby, Denmark
msbe@fotonik.dtu.dk

Abstract

In the last few years, we have been able to see how terms like Mobile Edge Computing, Cloudlets, and Fog computing have arisen as concepts that reach a level of popularity to express computing towards network Edge. Shifting some processing tasks from the Cloud to the Edge brings challenges to the table that might have been non-considered before in next-generation Software-Defined Networking (SDN). Efficient routing mechanisms, Edge Computing, and SDN applications are challenging to deploy as controllers are expected to have different distributions. In particular, with the advances of SDN and the P4 language, there are new opportunities and challenges that next-generation SDN has for Fog computing. The development of new pipelines along with the progress regarding control-to-data plane programming protocols can also promote data and control plane function offloading. We propose a new mechanism of deploying SDN control planes both locally and remotely to attend different challenges. We encourage researchers to develop new ways to functionally deploying Fog and Cloud control planes that let cross-layer planes interact by deploying specific control and data plane applications. With our proposal, the control and data plane distribution can provide a lower response time for locally deployed applications (local control plane). Besides, it can still be beneficial for a centralized and remotely placed control plane, for applications such as path computation within the same network and between separated networks (remote control plane).

2012 ACM Subject Classification Networks → Programmable networks

Keywords and phrases SDN, P4, P4Runtime, control planes, Fog, Edge

Digital Object Identifier 10.4230/OASIcs.Fog-IoT.2020.9

Funding This research has been supported by the Spanish Ministry of Science, Innovation, and Universities within the project TEC2017-87061-C3-2-R (CIENCIA/AEI/FEDER, UE).
David Franco: Supported by aforementioned project.
Marina Aguado: Supported by aforementioned project.

1 Introduction

The concept of Cloud Computing has turned into a revolution for computer science engineering, becoming essential when designing modern communication networks. With the advances in computational architectures and distributed computing, other approaches like Mobile Edge Computing, Fog Computing, and Cloudlet have arisen as relevant terms to describe new data processing procedures that bring computation closer to the clients.

The next-generation requirements created by offloading processing to the Edge need a way to be addressed. In recent years, Software-Defined Networking (SDN) has seen an evolution to develop full network programmability. The requirements imposed by Edge-based data processing can be addressed by SDN. We can see in Figure 2 how SDN networks have evolved, starting from traditional forwarding devices, to OpenFlow-based network and then to next-generation SDN networks based on P4 (data plane programming language) [2] [3] and P4Runtime (runtime protocol to control P4-defined switches) [4].

In particular, P4 is a language to define the data plane behavior for forwarding devices. As we can see in Figure 1, we can define how packets are parsed (custom headers), how packets are treated (custom tables and actions at ingress or egress stages of a pipeline) and how packets are sent again into the network (custom deparser, add or remove headers). We can also define how to count for packets/bytes, define network meters or be able to program our third-party external functions (also known as *externs*) that can be integrated into the data plane. Depending on the P4 programmable hardware, packets can also be treated differently in queues, being able to assign a particular priority.

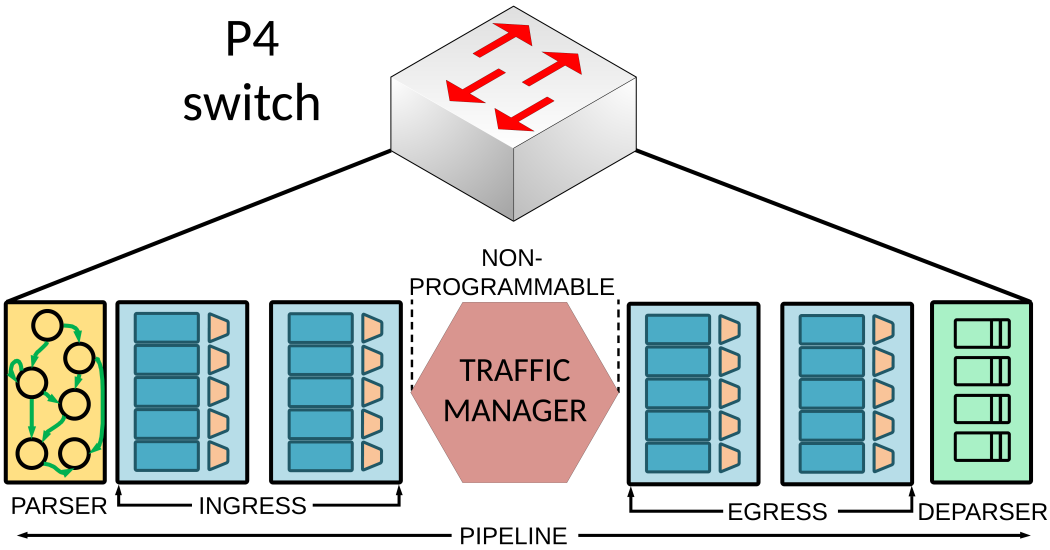

Figure 1 P4 switch pipeline definition by V1model architecture.

Thanks to the advent of data plane programming in the last few years, we can extend the functionality of networking devices. For instance, they are capable of performing computational operations in the data plane, at line speed. This concept is known as in-network computing, and it has several classes of applications such as network functions, caching or data aggregation. This computational approach yields new control-to-data plane protocols (e.g. P4Runtime), which opens new opportunities to define the control plane of next-generation SDN networks. Fog networks can especially benefit from these as control

planes can work both locally and remotely at the same time for the same data planes. Therefore, this paper describes several novel approaches to locally and remotely deploy SDN control to benefit Fog computing networks. For instance, a local control plane can address requests that require quick responses (e.g. Edge Computation), while a remote controller is responsible for making decisions that demand a global view of the network (e.g. path computation).

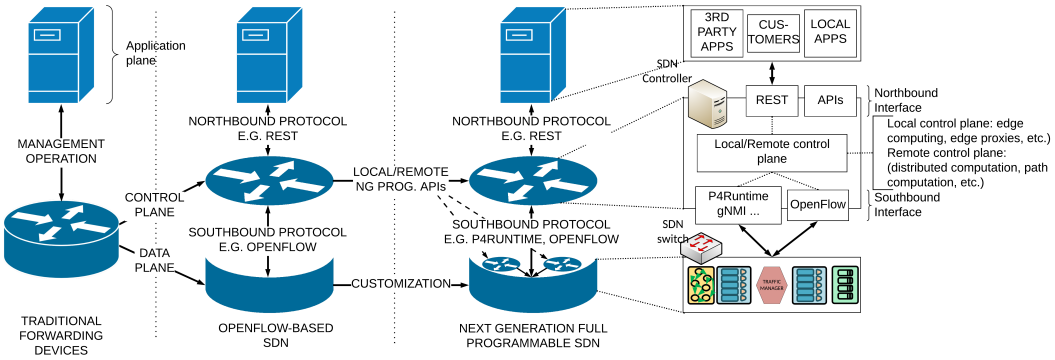

Figure 2 Evolution of SDN from traditional and unified control and data plane devices towards first generation OpenFlow-enabled SDN and current full programmable control and data planes.

Apart from the details we have offered about SDN and P4, we offer further details about the related work in Section II. We also provide further details on how exactly P4 and P4Runtime can be beneficial for Fog computing in Section III. Section IV describes a novel paradigm in which SDN network devices can be used for edge computing, specifically accounting for distributed control planes that are integrated in a cross-layer way. Finally, Section V offers a conclusion for the content explained in this paper.

2 Related Work

In this section, we will try to offer a variety of work-related to SDN and Fog computing. We have tried to focus on different use cases using SDN although most of the work done in the past years focuses on surveys that relate SDN to Edge/Fog computing [1] [10] [8]. There is not as much work in the practical and implementation side done with SDN and Edge/Fog compared to surveys, however, this section addresses a variety of topics inside Edge/Fog in which SDN has been interesting for.

The work we followed in [1] offers a wide variety of use cases for SDN within edge computing. The authors mention several of the key areas where SDN can benefit Edge networks using some of the key papers of the work. As they only briefly mention P4, we can demonstrate that the main areas of the paper have not deeply considered P4 or P4Runtime. However, some of the main areas are Service-Centric Implementation, Adaptability, Interoperability, High Resolution, and Effective control, etc. The role of SDN in these areas encompasses several control plane applications support, runtime service reconfiguration or mobility management support. The authors still point out that future research is going in the direction of improving network virtualization, enhancing north and southbound protocols to support future services and improve scalability and reliability.

Authors in [10] also briefly go through some of the key areas for SDN controllers in Fog computing. Authors mention that controllers should perform actions within data traffic management, resilience, and Fog orchestration. Time sensitivity is also mentioned as a key area for SDN controllers to manage when dealing proactively or reactively with table entries.

As mentioned in [6], authors show how to integrate SDN and Cloud-RAN within 5G to globally allocate resources for VANETs. The goal for SDN within the Fog network remains unclear but authors seem to integrate SDN controllers hierarchically on top of Fog computing BBU controllers to manage Fog orchestration and resources apart from doing regular network management tasks.

Specifically focusing on P4, authors in [9] explain how to integrate P4 into a multi-layer edge scenario. The authors propose 3 cases in which dynamic Traffic Engineering (TE) is covered and also cybersecurity which is addressed through P4-based solutions. The authors test their ideas in virtual switches like BMv2 and physical switches like the NetFGPA. Their tests report successful dynamic TE and cybersecurity mitigation without controller intervention. However, the authors still do not look into the same topics as we do in this paper for cross-layer multi-SDN control planes.

3 P4 and P4Runtime as a tool for fog computing

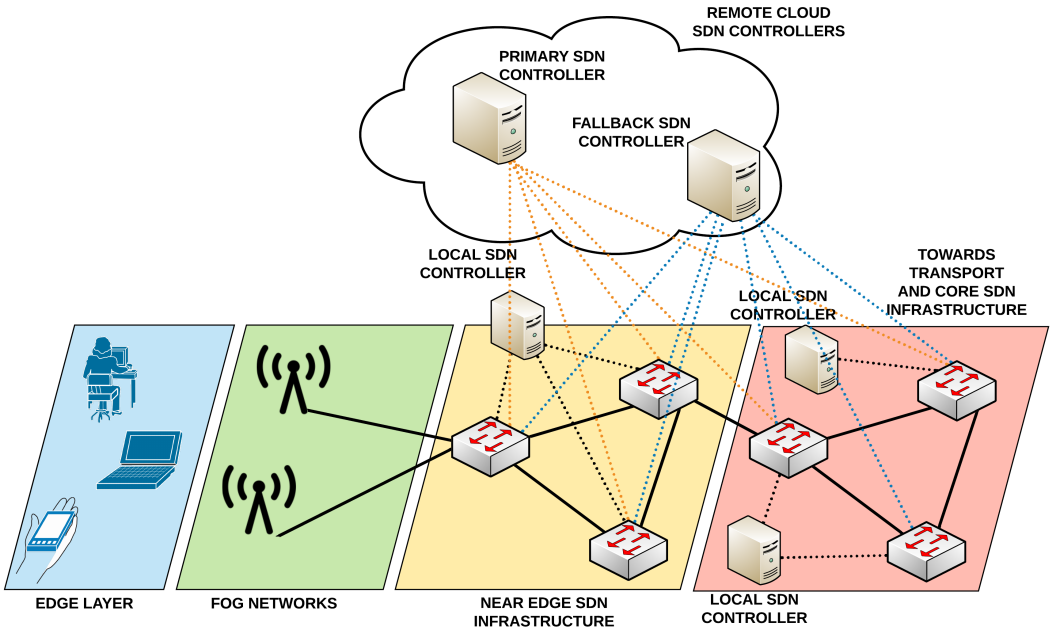

Figure 3 Local and remote control planes managing the data plane from Fog network and Core network based programmable data planes.

In 2008 an article describing a protocol called OpenFlow (OF) [7] gained enormous traction. The authors describe a protocol to encourage self-programmed control planes to program switches' forwarding tables. Switches would support a set of protocols (Ethernet, IPv4, TCP, etc.) and forward traffic depending on rules installed by SDN controllers. The supported matching engine has evolved to support new header fields (i.e. more protocols) as OpenFlow versions got developed. For instance, the OpenFlow version 1.5.1 supports over 41 fields. However, the researchers decided to develop new data and control plane programmability approaches. In 2014, the P4 language emerged to overcome some of the limitations that OpenFlow could have imposed. P4 encourages protocol evolution, enables faster design and better development cycle and provides new data plane monitoring techniques.

With P4, developers can define which headers a switch can understand, how to parse packets and customize matching criteria and actions. Moreover, as P4 is a programming language to describe the behaviour of the data plane, the entire processing takes place in the ASIC.

The limitations found in legacy SDN do not only refer to the data plane. The use of OpenFlow also imposes specific architectures that controllers need to follow. For instance, two different controllers can not control the same OpenFlow data plane, unless a layer between control and data plane is included [11]. Therefore, the support for multiple controllers over the same pipeline is not natively included in OpenFlow while it is in P4Runtime. This is further explained in the next sections.

While Fog computing has not been the main field for SDN study we believe that P4 and P4Runtime have strong arguments to be included at the Edge too. This section aims to strengthen the use of P4 within 5 main areas that make it relevant to be considered as the driver for Edge/Fog computing:

- Customized protocols: Within the core use case of P4 it is the definition of standardized and custom protocols. The definition of the headers, tables, and actions in P4 provides the means to build custom protocols.
- Protocol translation: As a consequence of being able to program which headers are parsed and which actions are executed, the pipeline can also enable and disable headers on demand (e.g. Tunneling packets between from Fog/Edge environments and also towards core).
- Monitoring: Having a custom data plane enables new telemetry protocols (e.g. In-band telemetry (INT)) to monitor the behavior and performance of packets within P4-programmable devices. Besides being able to create, modify and forward telemetry packets, the devices need to be able to monitor the data plane at the edge. To support this use case, data planes generally offer information via metadata available while packets traverse the data plane. This can be a beneficial use case to monitor packets at the edge, exporting information to network managers.
- Data plane partition: Being able to create tables (define maximum entry size, keys to match, actions and parameters, etc.) also brings new ways to distribute data plane functionality. Custom data planes can bring new ways of dealing with packets within the pipeline. For instance, one table can be used to learn new hosts and prevent spoofing while other tables can perform L2/L3 forwarding. This concept is tightly related to *controller partition*, explained in the next section.
- Edge cache: One of the areas with the biggest potential is using programmable switches as network caches. This functionality has already been studied in other publications [5] and its potential has already been demonstrated.
- Packet aggregation/disaggregation: Ongoing researcher explores the possibilities to accumulate data between a large number of devices (e.g. IoT devices [12]) and aggregate data to send it to the Cloud (and then back). The process followed here encourages the functional split between Edge and Cloud, by offloading some tasks at switches before forwarding traffic data to the Cloud.
- Control plane localization: With P4runtime the control plane for networks at the Edge and networks towards core networks can share the same controller. Indeed, they can also hold local controllers for fast tasks that independently work from centralized controllers.

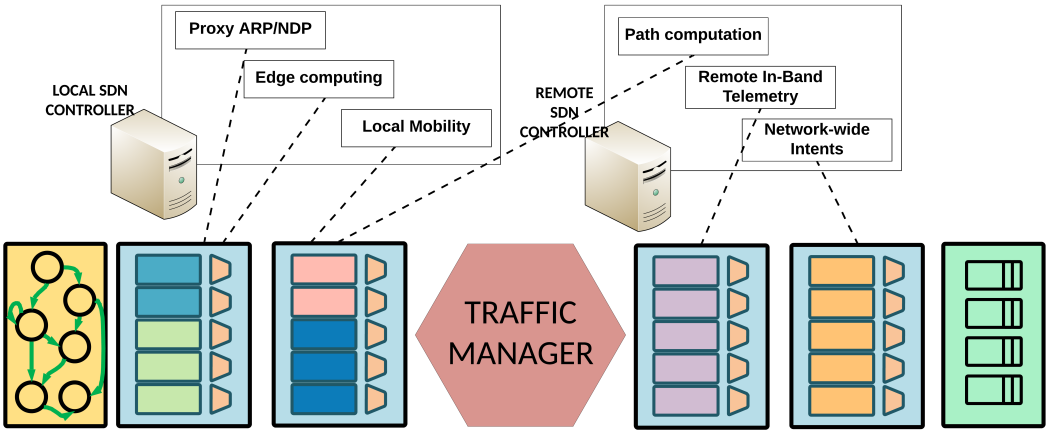

Figure 4 Local and remote control planes managing different match-action parts of a P4 pipeline.

4 A novel paradigm for Fog Computing

In this part of the paper, we will review how new SDN paradigm controllers can be beneficial for Edge/Fog networks. The controllers can be organized in particular ways to bring some functionalities closer to the Edge and offload others to controllers that are centralized in the Cloud. Edge caching for SDN networks (e.g. answering queries withing control plane within P4 switches) is a particularly interesting use case. Other functions like applications that deal with traffic forwarding can work centralized in the network to provide a better path computation calculation. In this section, we address two of the aforementioned areas that are data/control plane partition and Edge caching. In the next paragraphs, we explain how next-generation SDN control planes can be integrated and organized.

In Figure 3 we can observe how we integrate multi-layer SDN control planes. We focus on integrating SDN control planes within the SDN data plane that belongs closest to the network Edge. In this way, we can be sure that locally deployed control planes can manage any functionality that needs faster processing. As shown in Figure 3, we can observe both locally and remotely deployed control planes that are managing P4-based data planes both at the Edge and towards transport networks. With this approach, we enable both locally deployed control plane attend a few functionalities. The same switches also support remotely deployed primary and fallback control planes to attend actions that are delay tolerant but can benefit from a centralized control plane.

While Figure 3 shows how SDN controllers are organized within different network layers, we also want to show how different applications attend functionalities within the same switch. Figure 4 shows how locally and remotely deployed control planes manage their resources (match action units). For instance, as Figure 4 shows, SDN controllers generally act as ARP/NDP proxies, which requires the controller to answer the particular messages. In our example, we decide to move this app to a locally deployed control plane and let it answer messages as requested. Latest OpenFlow-based data planes (e.g. version 1.3 and on) can hold table rules that answer ARP messages too. However, centralized control planes need to attend new ARP requests and prepare the table entries. This now is expected to improve as locally deployed control planes deal with these requests instead of forwarding them to centralized control planes. On the other side, network-wide intents benefit from centralized control planes. To enhance traffic optimization, path computation algorithms benefit from network-wide information instead of locally deployed control planes that exchange routing information.

5 Conclusion

In this paper, we focus on bringing SDN but specifically P4/P4Runtime to Fog networks. While bringing processing to the Edge has many advantages, the challenges that arise need to be considered. To address some of the network challenges, we believe that P4 as a data plane programming language and P4Runtime as a control-to-data plane protocol can benefit next-generation network requirements. Particularly, we propose a new way of deploying SDN control planes that manage both SDN data planes that belong to Fog networks and also SDN data planes that are integrated close to the core network. In this way, SDN controllers can manage cross-layer SDN data planes to offload some functionalities to the Edge (e.g. proxying, caching, small scale auditing, etc.) and also delay tolerant application (e.g. path computation, network wide monitoring etc.). We propose this data plane programming model to serve as the first step into future work to evaluate how next-generation SDN data planes can be distributed and organized to maximize workload for full programmability of next-generation SDN networks. We can demonstrate that this new cross-layer SDN network management is new to Edge/Fog networks and that it can enable a performance improvement of upcoming networks.

References

1. Ahmet Cihat Baktir, Atay Ozgovde, and Cem Ersoy. How can edge computing benefit from software-defined networking: A survey, use cases & future directions. *IEEE Communications Surveys & Tutorials*, PP:1–1, June 2017. doi:10.1109/COMST.2017.2717482.
2. Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, July 2014. doi:10.1145/2656877.2656890.
3. P4 Language Consortium. P4. https://p4.org/. (Accessed on 02/27/2020).
4. The P4.org API Working Group. Specification documents for the P4Runtime control-plane API. https://github.com/p4lang/p4runtime, 2020.
5. Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soulé, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. Netcache: Balancing key-value stores with fast in-network caching. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 121–136, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3132747.3132764.
6. A. A. Khan, M. Abolhasan, and W. Ni. 5g next generation vanets using sdn and fog computing framework. In *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6, January 2018. doi:10.1109/CCNC.2018.8319192.
7. Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008. doi:10.1145/1355734.1355746.
8. Jéferson Campos Nobre, Allan M. de Souza, Denis Rosário, Cristiano Both, Leandro A. Villas, Eduardo Cerqueira, Torsten Braun, and Mario Gerla. Vehicular software-defined networking and fog computing: Integration and design principles. *Ad Hoc Networks*, 82:172–181, 2019. doi:10.1016/j.adhoc.2018.07.016.
9. F. Paolucci, F. Civerchia, A. Sgambelluri, A. Giorgetti, F. Cugini, and P. Castoldi. P4 edge node enabling stateful traffic engineering and cyber security. *IEEE/OSA Journal of Optical Communications and Networking*, 11(1):A84–A95, January 2019. doi:10.1364/JOCN.11.000A84.

10 J. Pushpa and Pethuru Raj. *Performance Enhancement of Fog Computing Using SDN and NFV Technologies*, pages 107–130. Springer International Publishing, Cham, 2018. doi:10.1007/978-3-319-94890-4_6.

11 Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru Parulkar. Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, 1:132, 2009.

12 Shie-Yuan Wang, Chia-Ming Wu, Yi-Bing Lin, and Ching-Chun Huang. High-speed data-plane packet aggregation and disaggregation by p4 switches. *Journal of Network and Computer Applications*, 142:98–110, 2019. doi:10.1016/j.jnca.2019.05.008.