Does it Live up to the Hype: Extensibility of the $P4_{16}$ Compiler for New Targets

Funmilayo Olaiya
Cheriton School of Computer Science
University of Waterloo
folaiya@uwaterloo.ca

1 INTRODUCTION

Computer networks are notoriously complex to manage, especially because a network architecture involves a variety of equipment types, including routers, switches, and other devices [3].

These devices may continue to multiply inside a network topology and eventually become very challenging to control, maintain, and properly work with. Even if they all communicate with one another, since many of these devices are primarily made of hardware and operate independently, it still shows that trying to upgrade them, especially switches in particular, can be a challenging task.

The networking community is now placing a lot of emphasis on SDNs in an effort to reduce the complexity that comes with many network architectures. Despite appearing to have emerged relatively suddenly [3], SDNs as a concept have actually been developing for many years [4].

One of the reasons why SDNs are the preferred option is that, for instance; inside a network device, let's say a switch - the control plane and data plane are separated, whereas in other traditional networks, the control plane and data plane are integrated. The pathways leading to the data plane must now be configured by the control plane [4]. This direct approach cannot possibly encourage scalability.

This also leads to another aspect that makes SDN a much more attractive option which is that, the fact that there is a centralised controller and that networks can be much more programmable with the correct resources because the control plane and data plane are now separated.

And in this research, we'll concentrate our attention on a tool called the $P4_{16}$ compiler, which is a backend compiler that supports and compiles the P_4 programming language and helps in the programming of various network devices, and also encourages the programmability of network switches.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. . . $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 PROBLEM STATEMENT

The $P4_{16}$ *programming language* [2] is a paper that was published in 2017, and in the paper, the authors try to address the drawbacks of an earlier work titled P_4*: Programming Protocol-Independent Packet Processors* [1] that was published in 2014.

Some of the interesting design objectives that $P4_{16}$ aimed to achieve include *developing a statically-typed language/compiler in C*, *making the new language more expressive than* $P4_{14}$, *making it simple and full of reusable keywords*, and *supporting a wide range of targets that may have various architectures or capabilities* [2].

And in this research, we are going to investigate if $P4_{16}$ can genuinely support a large number of targets *(or any target)* as the authors promise it can.
Does it live up to the hype in reality?

3 CONTEXT AND MOTIVATION

Every network expert in the community looks at ways to make networks more programmable through software and less complex through hardware. And that is why many ideas and tools originate, grow, are wanted, and so on. One of the ambitious network tools developed to make sure networks are simple to program is $P4_{16}$ as a programming language and compiler.

And as a result of all these actions, these tools need to be improving daily; the more advanced they are, the more beneficial they are for usage on networks.

The primary driving force behind this research is to further our understanding of the tool by delving deeply into it. We are doing this because we need to see more of the tool and because we think that whatever we learn will significantly advance the development of the $P4_{16}$ compiler and the field of programmable networks as a whole.

4 METHODOLOGY

The approach is straightforward; first, we must delve deeply into the $P4_{16}$ compiler's backend architecture and comprehend its IR (Intermediate Representation) in great detail. Second, we attempt to build a different target (a backend system) to evaluate the compiler's support for it.

We need to confirm whether the claim that $P4_{16}$ can support numerous and unforeseen targets is valid.

5 RELATED WORK

Since we are attempting to estimate the long-term design goals of the $P4_{16}$ compiler [2], the majority of our research will be focused on this paper.

REFERENCES

[1] Pat Bosshart et al. "P4: Programming Protocol-Independent Packet Processors". In: *SIGCOMM Comput. Commun. Rev.* 44.3 (July 2014), pp. 87–95. ISSN: 0146-4833. DOI: 10.1145/2656877.2656890. URL: https://doi.org/10.1145/2656877.2656890.

[2] Mihai Budiu and Chris Dodd. "The P416 Programming Language". In: *SIGOPS Oper. Syst. Rev.* 51.1 (Sept. 2017), pp. 5–14. ISSN: 0163-5980. DOI: 10.1145/3139645.3139648. URL: https://doi.org/10.1145/3139645.3139648.

[3] Nick Feamster, Jennifer Rexford, and Ellen Zegura. "The Road to SDN: An Intellectual History of Programmable Networks". In: *SIGCOMM Comput. Commun. Rev.* 44.2 (Apr. 2014), pp. 87–98. ISSN: 0146-4833. DOI: 10.1145/2602204.2602219. URL: https://doi.org/10.1145/2602204.2602219.

[4] Sakir Sezer et al. "Are we ready for SDN? Implementation challenges for software-defined networks". In: *IEEE Communications Magazine* 51.7 (2013), pp. 36–43. DOI: 10.1109/MCOM.2013.6553676.