To be decided

Anonymous Author(s)

ABSTRACT

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Anonymous Author(s). 2019. To be decided. In *Proceedings of CoNEXT '19.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

« SDN separates control and data planes of networks and allows to change the behavior of packet processing device, depending on data plane of the devices. CP of the network function can configure its data plane objects at run-time or compile time to achieve desired packet processing behavior or modify it. This horizontal separation of network's control and data planes provides flexibility to control and manage network functions. Adding into that, P4, a data plane programming language, allows to describe data plane part of packet processing logic required to realize a network function. It exposes control plane APIs to program the data plane objects, thereby enabling network programmability along with easy control and management. »

In programmable networks paradigm, programmers need to describe data and control plane logic of network functions as different programs and more often in different languages. This, by design split of logic and execution control flow across multiple heterogeneous programs, increases development, test and deployment complexity of network functions. Also, it requires novel mechanisms to build complex network functions by reusing independently developed and tested data and control plane code. However, P4 mandates to write a monolithic data plane program and carefully configure the data plane objects to build a application processing various protocols and performing multiple network functions. For example, switch.p4 [1] has control blocks defined to processes different protocol headers and network functions(e.g., l2 switching, l3 routing etc.,). But, the control blocks globally share different types of metadata structures and parsed headers. Without understanding implementation details of the control blocks and entire program, reuse of the code is difficult due to lack of clear interface and abstraction. Existing paradigm requires modularity that allow programmers to expose interface to reuse code written to process packet at any granularity of functionalities while abstracting away the implementation details.

Let's consider a simple scenario. A program, l3.p4, (defines a callable entity that) parses IPv4 header from packets, performs longest-prefix match and determines next hop. Moreover, it decrements the ttl field and deparse the packet. Another program, l2.p4, processes the same packet and takes the next hop as input argument, parses Ethernet header, matches on next-hop and modifies ethernet addresses. Finally, it deparses the packet and sends on appropriate port. In this example, l3.p4 is not generating a functionally correct packet to forward on wire. However, it can be reused with different layer-2 forwarding mechanism or even with MPLS and create functionally correct packet to forward on wire. Such fine-grained packet processing modules enable code reuse and modular control over data plane objects, thereby facilitating incremental development of network functions.

To enable modular Earlier work on virtualization of data planes, HyPer4 [2], HyperV [3], lacks inter-module communication mechanism(e.g., next hop in above example) except via packets. Encapsulating customized headers inside the packet may allow such communication, but that would require to know implementation details of deparser in one module to write complimentary parser in other and vice versa. In case of virtualization, minimal composable unit is a data plane program of a network function(e.g., switching, routing etc.,). However, a network function can be enriched and incrementally built by reusing code protocol specific code.

P4Visor [4]

It does not allow reuse of the network functionality at arbitrary point in control execution of

network function level modularity, where processed packet are functionally valid packet to transfer on wire.

CoNEXT '19, December 9-12, 2019, Orlando, Florida, USA

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

REFERENCES

[1] P4 Language Consortium. 2013. The switch.p4 program describes a data plane of an L2/L3 switch. Retrieved June 5, 2019 from https://github.com/p4lang/switch/tree/master/p4src

[2] David Hancock and Jacobus van der Merwe. 2016. HyPer4: Using P4 to Virtualize the Programmable Data Plane. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies (CoNEXT '16)*. ACM, New York, NY, USA, 35–49. https://doi.org/10.1145/2999572.2999607

[3] C. Zhang, J. Bi, Y. Zhou, A. B. Dogar, and J. Wu. 2017. HyperV: A High Performance Hypervisor for Virtualization of the Programmable Data Plane. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. 1–9. https://doi.org/10.1109/ICCCN.2017.8038396

[4] Peng Zheng, Theophilus Benson, and Chengchen Hu. 2018. P4Visor: Lightweight Virtualization and Composition Primitives for Building and Testing Modular Programs. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '18)*. ACM, New York, NY, USA, 98–111. https://doi.org/10.1145/3281411.3281436