NETRONOME

WHITE PAPER

P4: Driving Innovation in Server-Based Networking

RESEARCHERS ARE INNOVATING USING P4 IN SERVER-BASED NETWORKING SYSTEMS WITH NOVEL APPROACHES TO OFFLOADING SERVERS AND REALIZING NEW FUNCTIONALITY.

INTRODUCTION

Programming Protocol-independent Packet Processors, or P4, is a new high-level programming language for software-defined networks [1, 2]. It is intended to describe the behavior of the data plane of any system or appliance that forwards, modifies or inspects network traffic. P4 started as an innovation for the network core [3], but researchers focused on server-based networking recognized its value to applications in their areas of interest. Smart NICs, now being deployed into data centers [4] are a potential vehicle to exploit P4 in server-based networking applications. Researchers are innovating using P4 in server-based networking systems with novel approaches to offloading servers and realizing new functionality.

This white paper is an introduction to recent research in P4 for server-based networking.

ABOUT P4

The P4 language [5] is based on the match-action flow paradigm used in OpenFlow [6]. In this approach, actions are used to change packet contents when they match specific keys. The language's constructs focus on enabling users to efficiently describe match-action flows.

P4 is meant to describe or specify the behavior of the data plane, but not how the data plane is actually implemented. Depending on the flexibility and complexity of the system or appliance, the data plane can run the gamut from fixed-function ASICs (such as those used in Layer 2 switches) all the way up to fully programmable, general-purpose CPUs found in white box implementations of routers, web proxies and firewalls.

One of the primary goals of P4 is to keep the language hardware-independent (or target-independent). When the designers of P4 state that it is a programming language intended for packet processors, which packet processing device are they referring to? Hardwired ASICs, FPGAs or CISC CPUs? The answer is: all of the above, provided they can interpret P4.

The job of interpreting P4 behavioral code and generating a lower-level, device-specific implementation falls to manufacturers of the target systems. For example, Netronome's Agilio® Smart NICs have programmable data planes. Hardware and software from Netronome make it possible to conceive, develop and debug P4 code on these Smart NICs [7]. P4 can similarly be compiled onto architectures ranging from a TCP/IP stack on an x86, through to full-custom ASICs, using open-source or custom-tools from manufacturers [8-12].

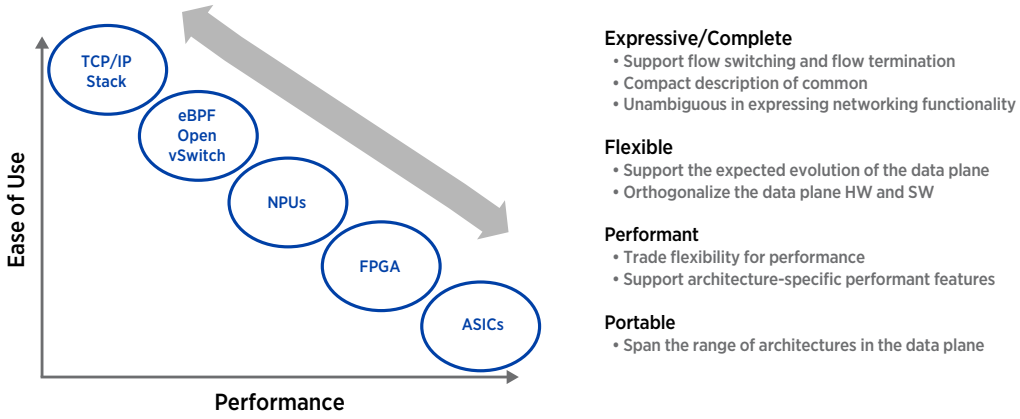

FIGURE 1 – P4 goals for hardware independence

P4 — OFFLOAD AND SYSTEM DESIGN

Many networking systems use special-purpose hardware to accelerate networking functions to meet line-rate and functionality requirements and to offload the host. The P4 language itself assumes that some actions are better done using externs, treated as black boxes, implemented using accelerators or special-purpose external hardware [5].

Highly programmable accelerators composed of a large number of programmable cores such as in network processors, or logic gates as in FPGAs, offer the potential to combine performance with the flexibility required in practical networks. Their use has been hampered because developers typically find accelerators hard to program. A developer has to manually manage functions that are automated by the operating system or other services in general-purpose systems. A developer has to implement two classes of functions when programming accelerators:

- Packet physical management: Logic to transport a packet from interface to interface and give it access to system resources, i.e. operating-system like functions.
- Packet function processing: Logic to alter the contents of the packet (protocol headers) and route it to the appropriate interface, i.e. actual application logic.

Network system developers are generally interested in the second class of functions. In systems with accelerators, they are required to implement the first type to achieve their objectives.

A language like P4 enables an interesting approach in systems with programmable accelerators. Physical packet management can be decoupled from functional packet management. A portion of the programmable logic in the accelerator can be set aside as a resource island, a sandbox. The accelerator vendor can now offer a toolchain to map the P4 code to the sandbox. The accelerator vendor can write performant software to use the non-sandbox resources on the accelerator to manage packets. The management software presents packets to the sandbox through logical interfaces. Network developers can write code in P4 to describe the logical network functionality required in the accelerator.

With this approach, network function developers can focus on what is relevant to them, packet function processing. Netronome uses the sandbox approach in its P4 SDK and toolchain. Additionally, with Netronome's P4 SDK developers can implement P4 external functions in C. This allows developers to realize features, especially stateful features, that are not possible with P4. Figure 3 shows the process with which developers create, compile and download P4 and C code to the Netronome SmartNIC. Figure 3 shows how P4/C code is inserted into a resource island.

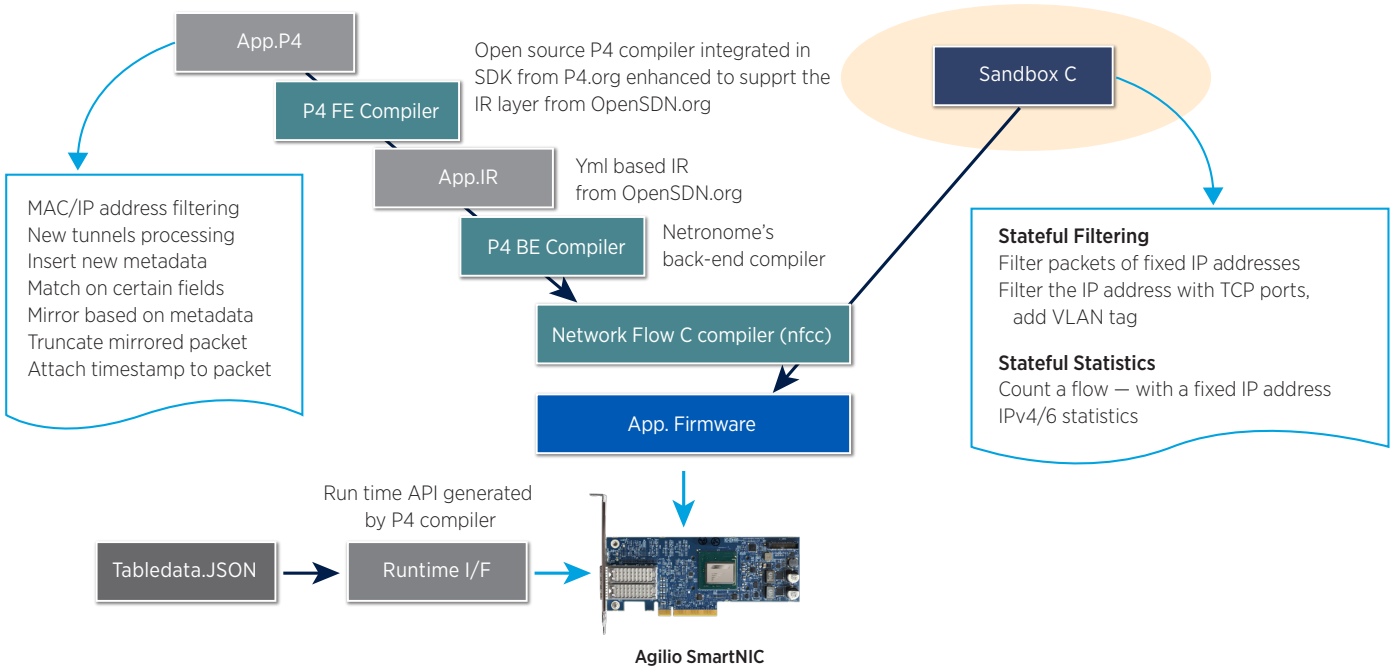

FIGURE 2 – The P4 Compilation Process

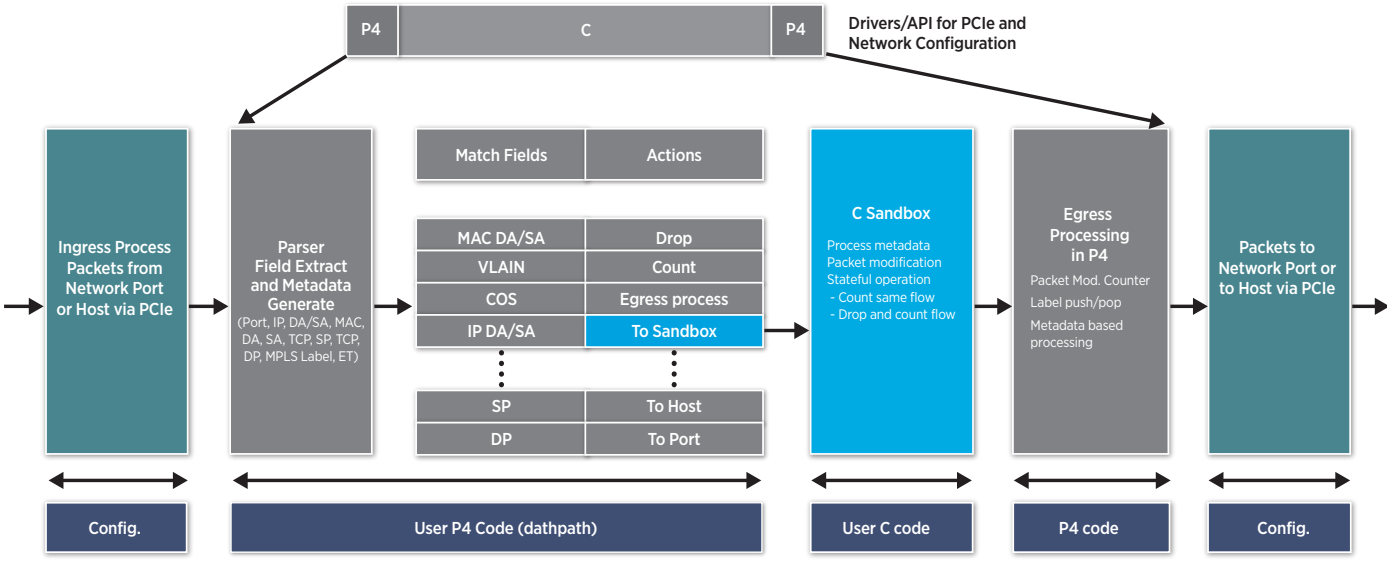

FIGURE 3 – Mapping P4/C code to a resource island

P4 AND SERVER-BASED NETWORKING

THE ABILITY TO EXPERIMENT WITH NEW PROTOCOLS HAS GIVEN DATA PLANE RESEARCHERS THE ABILITY TO INVESTIGATE AND PROTOTYPE AN EXTRAORDINARY RANGE OF NEW NETWORK FUNCTIONS ON PRODUCTION NETWORKING HARDWARE.

P4's development focus was and continues to be the switching data plane in the network core. However, P4 has shown surprising growth in server-based networking applications.

Most network data planes perform three basic operations: packet parsing, match/action operations and packet reassembly. P4 provides coding constructs that make describing these operations easy to understand. P4 programs have to describe the expected contents of the packets processed as well as the way in which they are modified. P4 has the ability to define not just standard packet header structures the parser will extract, but also entirely new protocols as seen in Figure 4.

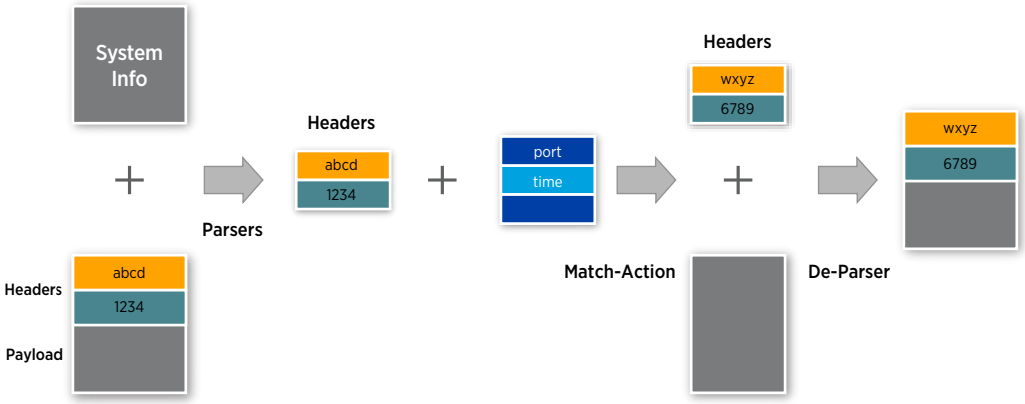

FIGURE 4 – Header parsing in P4

Developers in P4 can experiment in software-defined networks (SDN) with entirely new protocols at any layer of the network stack. This is a contrast with prior SDN efforts such as OpenFlow, which were built on existing networking standards. Parsing Ethernet, IP or TCP headers using OpenFlow is not a problem, as OpenFlow supports those protocols. Alternatively, if you need to parse a newer protocol, such as an NVGRE packet, you will have to wait until OpenFlow adds support for it.

The ability to experiment with new protocols has given data plane researchers the ability to investigate and prototype an extraordinary range of new network functions on production networking hardware. More importantly, with the island approach, they can focus their energies on the specific idea of interest while producing results in real networks. Secondly, with code in C, they are able to close functional gaps in P4. Netronome supports research and development in P4 (and other data plane acceleration technologies) through the community portal Open-NFP. We briefly discuss some of the projects in P4 proposed by the community on Open-NFP [13].

An obvious focus area for researchers is using P4 to drive new approaches to realizing networking systems on commodity off-the-shelf servers, both in bare-metal applications and in virtual applications in network functions virtualization(NFV) systems [14-16]. In bare-metal applications, researchers have proposed using P4 to provide multicast for network media streaming with unicast semantics, using P4 to implement information-centric networking concepts, using P4 to offload hosts and implement 5G network protocols on SmartNICs, and using P4

as a mechanism for OpenFlow controllers to push security policies to NICs and switches. In virtual applications, researchers are using P4 as a mechanism to offload Virtual Network Functions (VNFs) on hosts in NFV systems and/or to enhance VNF functionality.

The second area of interest has been protocols to implement new functionality [17, 18]. One interesting area is using a NIC to offload the host to better use host resources or improve application performance. Examples of this include offloading consensus protocol services to the network from hosts in distributed systems and enabling applications to directly receive network data by bypassing the network stack in distributed applications. Researchers have also proposed using P4 to program the data plane in Open Virtual Switch (OVS) [11].

Researchers are also planning to use the ability to define custom protocols on a NIC to investigate new approaches to security and monitoring [19-21]. These include new connection authentication protocols, new methods to authenticate and isolate flows securely, and protocol-based approaches to monitoring network performance in real time.

CONCLUSION

Looking forward, more change and growth are likely. Researchers can participate in and contribute to this effort in several ways. The P4 organization holds developer days twice a year and a research conference [22] once a year. The organization also drives language and architecture development through working groups. For example, the next version of P4, P4-16, is on the horizon. The P4 language repo[23] hosts a lot of example code for applications and networking functions. Discounted hardware to use with P4, a development SDK and learning material, including several complete P4 and P4/C projects [24] and labs, are available to researchers and at Open-NFP.

References

1. The P4 Language Consortium, **www.p4.org**
2. P. Bosshart, et al, "P4: Programming Protocol-Independent Packet Processors" ACM Sigcomm Computer Communications Review (CCR). Volume 44, Issue #3 (July 2014)
3. A. Sivaraman. "DC. p4: programming the forwarding plane of a data-center switch." Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research. ACM, 2015.
4. D. Firestone, "SmartNIC: Accelerating Azure's Network with FPGAs on OCS servers", Open Compute Project, **http://files.opencompute.org/oc/public.php?service=files&t=5803e581b55e90e-51669410559b91169&download&path=//SmartNIC%20OCP%202016.pdf**
5. The P4 language specification, **http://p4.org/wp-content/uploads/2016/11/p4-spec-latest.pdf**
6. The Open Networking Foundation, OpenFlow, **https://www.opennetworking.org/sdn-resources/openflow**
7. J. Tönsing, "P4/PIF + C Programmable Intelligent NICs: Requirements and Implementation Notes", P4 Workshop, 2015, **http://schd.ws/hosted_files/2ndp4workshop2015/74/Netronome,%20P4%20Workshop%20Nov%2018%202015.pdf**
8. G. Brebner, "P4 for an FPGA Target", P4 Workshop, 2016, **http://sched.co/3ZQA**
9. J. Fastabend, P4 on the Edge, P4 Workshop, 2016, **http://sched.co/6ouC**
10. P. Li and Y. Luo. 2016. P4GPU: Accelerate Packet Processing of a P4 Program with a CPU-GPU Heterogeneous Architecture. In Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems (ANCS '16). ACM, New York, NY, USA, 125-126.

11. M. Shahbaz et al, "PISCES: A Programmable, Protocol-Independent Software Switch", In Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference (SIGCOMM '16). ACM, New York, NY, USA, 525-538.
12. M. Budiu, "Compiling P4 to EBPF", **https://github.com/iovisor/bcc/tree/master/src/cc/frontends/p4**
13. Research Projects on Open-NFP, **www.open-nfp.org/projects**
14. A. Azgin, R. Ravindran, G.Q.Wang "pit/LESS: Stateless Forwarding in Content Centric Networks", IEEE Globecom, 2016.
15. M. Moradi, W. Wu, Li Erran Li. Z. M. Mao, "SoftMoW: Recursive and Reconfigurable Cellular WAN Architecture", In Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference (SIGCOMM '16). ACM, New York, NY, USA.
16. D. Hancock and J. van der Merwe. "HyPer4: Using P4 to Virtualize the Programmable Data Plane." Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies. ACM, 2016.
17. H-T. Dang,et al., "Network Hardware-Accelerated Consensus", USI Technical Report 2016-03, May 2016.
18. A. Kaufmann et al, "High Performance Packet Processing with FlexNIC", In 21st International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Atlanta, GA, USA, April 2016.
19. R. van der Pol, et al. "Assessment of SDN technology for an easy-to-use VPN service." Future Generation Computer Systems 56 (2016): 295-302.
20. J. Sonchack et al. "Enabling Practical Software-defined Networking Security Applications with OFX." Network and Distributed System Security, 2016
21. S. Song and T. Choi, "P4 In-band Network Telemetry Use Cases - Seeing Trees and Leaves to Learn About Forests", P4 Workshop, 2016.
22. P4 Workshop 2016, **https://2016p4workshop.sched.com/**
23. The P4 language repo, **https://github.com/p4lang**
24. The Open-NFP repo, **https://github.com/open-nfpsw**

NETRONOME

Netronome Systems, Inc.
2903 Bunker Hill Lane, Suite 150 Santa Clara, CA 95054
Tel: 408.496.0022 | Fax: 408.586.0002
www.netronome.com