NETRONOME

WHITE PAPER

P4 Data Plane Programming for Server-Based Networking Applications

THE PERFORMANCE OF A P4 PROGRAM IS DETERMINED MORE BY HOW WELL THE NETWORKING SYSTEM IS OPTIMIZED FOR NETWORKING APPLICATIONS OF INTEREST THAN BY OPTIMIZATIONS FOR PROGRAMMING LANGUAGE STRUCTURE

CONTENTS

INTRODUCTION

P4 is a relatively new domain-specific language developed to program the data plane in networking devices used in software-defined networking (SDN). Given the domain-specific application, it is tempting to believe a networking system can be "optimized for P4." P4 is a tool to program the data plane in a networking system; it is not meant to define how a data plane in a networking system must be implemented. It is expected that systems that utilize P4 programs to define a data plane also provide support for functions not supported in P4, in the form of "black box" externs. Finally, the performance of a P4 program is determined more by how well the networking system is optimized for networking applications of interest than by optimizations for programming language structure. The networking applications of interest described in this paper are those that run in data center servers (referred to as server-based networking applications); for example, a virtual switch (vSwitch), virtual router (vRouter), virtual network functions (VNFs) or other forms of data planes that implement network virtualization, distributed security and telemetry.

P4 SUCCESS FACTORS

The manufacturer of a target device is expected to provide a tool chain to map P4 programs to its target hardware architectures. When evaluating implementation of a P4-based data plane on a target for a specific set of applications, three factors are of particular interest:

- **Functional range:** The functional range of P4-based programs supported on the target architecture determines the range of applications that can be developed. For example, in one case the functional range of a target may be limited to stateless packet processing only based on layer 2 (L2) and layer 3 (L3) headers. Any functionality executable on the target can be expressed entirely in P4. However, a more complex target may support a functional range that includes flexible stateful payload processing. The full range of functionality possible on this target cannot be implemented solely with the P4 language as it is defined today. Netronome's Agilio SmartNICs enable programmers to efficiently implement flexible stateful payload processing functionality by extending P4 with programmable extern functions. Programmable externs extend the concept of fixed-function externs envisioned in P4.
- **Performance:** The performance of P4-based programs not only depend on the target hardware architecture but also on how the code is executed in the silicon, and whether program is instruction cycle-intensive or memory access-intensive. The latter is more prevalent in server-based networking applications. Netronome's tool chain produces performant P4 code by enabling more instruction cycles per dollar and watt therefore minimizing the cost of data movement through efficient memory access.
- **Productivity:** The productivity of the tool chain is related to the extent of working code that developers can produce within a given time period. Netronome's tool chain offers a full-featured integrated development environment (IDE) with debugging, editing and performance analysis support to promote developer productivity. Furthermore, the ability to augment P4 code with programmable extern functions implemented in C (or other languages in the future) enables development of more sophisticated server-based networking applications at a faster pace.

SMARTNICS: IDEAL FOR SERVER-BASED NETWORKING APPLICATIONS WITH P4

PROGRAMMABLE EXTERNS EXTEND THE CONCEPT OF FIXED-FUNCTION EXTERNS ENVISIONED IN P4

P4 was proposed as a solution to address the limitations of data plane flexibility in network switches that supported OpenFlow standards at various levels. Consistent with its OpenFlow heritage, the original P4 paper proposing this new data plane language focused on its value in L2/L3 network switching systems such as top of rack (TOR) switches. A relatively new organization, P4.org, was formed to more formally develop the language. P4.org introduced the first P4 specification in 2014, now referred to as the P4-14 specification.

Almost immediately after its inception, networking researchers and developers (such as those collaborating in Open-NFP.org) realized that P4 was not just a mechanism to program the data plane. They found use of P4 as a tool to describe the data plane in any networking device, not just switching systems at the core of the network. The functional range of server-based networking applications differ from those in a networking switch such as a TOR switch. Though the developers of the language recognized the diversity in networking applications, the constructs in the actual language – in both the P4-14 specification and even in the more recent P4-16 specification – are oriented to switching in the core as the primary application.

The P4 language assumes network function processing is flow based. While this abstraction is also applicable to server-based networking applications implemented in SmartNICs, it is not sufficient. Many server-based networking applications terminate flows and in many cases

send the packet payloads to the host. Flow termination requires the following attributes not supported in P4:

- **Payload access:** The P4 language specification assumes all processing is limited to the packet headers. Some server-based networking applications such as security applications require deep payload inspection. Others, including tunneling and link aggregation protocols, may demand that a packet's payload be aggregated/disaggregated across multiple packets.
- **Deep state per flow:** The ability to maintain state per flow is limited to registers and counters in the P4 language specification. Most server-based networking applications require much larger state per flow with more complex time-based rate attributes.
- **Other miscellaneous functions:** Many server-based networking applications require miscellaneous functions such as timeouts, packet origination and saturation arithmetic that are not easily implementable using the P4 language specification.

The P4 language specification recognizes that system vendors may implement functionality not realizable in P4 in logic outside the P4 program. The specification assumes that a P4 program will interact with fixed-function logic blocks for functions that cannot be implemented using the P4 language. This approach is suitable for high throughput systems such as the multi-terabit network switches available in the market today. A fixed-function logic approach is unsuitable, in many cases, for server-based networking applications because they have very different attributes such as:

- **Workload diversity is higher:** Networking at the core primarily involves routing packets based on various header attributes. Server-based networking applications involve more complex functions than this.
- **More workload churn:** Networking functions change more frequently because the applications on the servers change frequently.

One option for eliminating the fixed-function logic approach is to implement the logic in software using server CPU cores. However, implementing such networking functions takes away significant valuable CPU resources from revenue-generating applications. SmartNICs are a logical choice to execute P4-based networking applications, including externs. It is easier and more practical to implement the wide and complex set of networking functions required by externs in software executing in a SmartNIC. At the same time, the implementation can be achieved within the price, power, throughput and latency constraints required in server applications.

Netronome's Agilio SmartNICs are designed to efficiently support P4 programs for server-based networking applications. We address each of the three aspects discussed above - *functional range, performance, and productivity* - in more detail below.

FUNCTIONAL RANGE WITH NETRONOME SMARTNICS

Netronome's P4 IDE provides full support for all the features in the P4-14 specification. Netronome's IDE will support the P4-16 specification soon. To address the functional gaps in the P4 language and enable developers to create function-rich programs for server-based networking applications, Netronome provides an innovative option. Programmers can devel-

op custom functions in C to access packet payloads. These functions can then be accessed as extern functions by the P4 program. This integrated P4 with C approach offers several advantages:

- Developers can implement arbitrary state update and payload processing functions. For example, Netronome IDE users have used C to capture timestamps, examine payloads, maintain connection state, and implement other network functions..
- Soft externs can be both developed and updated rapidly as the networking applications on the server change.

Combining P4 with callable C functions enables developers to create more efficient implementations. As an example, the Count-min sketch is a well-known probabilistic data structure to identify heavy-hitters in streaming data. Researchers from the University of Massachusetts, Lowell implemented an algorithm for the Count-min sketch structure as a server-based networking application on Agilio SmartNICs to process streaming data.

The researchers implemented the algorithm in three ways: (1) Entirely in P4; (2) P4 augmented with C without multi-threading correctness; (3) P4 augmented with C and locks to guarantee multi-threading correctness. The table below compares the number of instruction cycles (in micro-engine (ME) cycles on the Network Flow Processors (NFP) in the Agilio SmartNIC) per data packet across the three options. Both C-augmented implementations, including the one that offers multi-threaded accuracy, consume fewer cycles per packet than the P4-only implementation. Readers are referred to the **presentation** from researchers at the UMass Lowell for more detail.

Implementation	P4 only	P4 & C	P4 & C with Lock
Number of Micro Engine (ME) cycles	7798	7396	7441

***TABLE 1** – ME cycles used in the Count-min application.*

The P4-only implementation resulted in the highest latency, about 6 percent higher than the P4 and C implementation, and 5 percent higher than the P4 and C with Lock implementation

P4-BASED DATA PLANE PERFORMANCE ON 40GBE AGILIO SMARTNIC

Since the P4 specification originally focused on network switch systems, the first version used an OpenFlow-like, map-action pipeline as a reference architecture. However, given the wide range of networking systems (networking switches and SmartNICs alike) for which applications have been developed in P4, the P4-14 specification also had a novel and explicit design goal for execution of a P4 program to be architecture-independent. The thrust of the P4 language specification being focused on network switch systems was weakened somewhat in the recent P4-16 specification. Here is a relevant excerpt from the P4-16 spec: “While initially P4 was designed for programming network switches, its scope has been broadened to cover a large variety of packet processing systems.”

The P4 specifications expect that a P4 program is not executed directly on a target architecture, but is somehow compiled onto the target architecture. Another relevant excerpt from the P4-14 spec: “P4 focuses on the specification of the parser, match + action tables and the

control flow through the pipelines. Programmers control this by writing a P4 program which specifies the switch configuration …. A machine that can run a P4 program is called target (sic). Although a target may directly execute a P4 program, it is assumed in this document that the program is compiled into a suitable configuration for the target."

Netronome's IDE tool chain compiles P4 programs to dedicated processing resources in the Agilio SmartNIC. The tool chain maps the P4 reference architecture and custom C functions to dedicated micro engines (MEs) on the NFP in the Agilio SmartNIC. Please see figure 1 below. With this approach, developers can create performant programs.

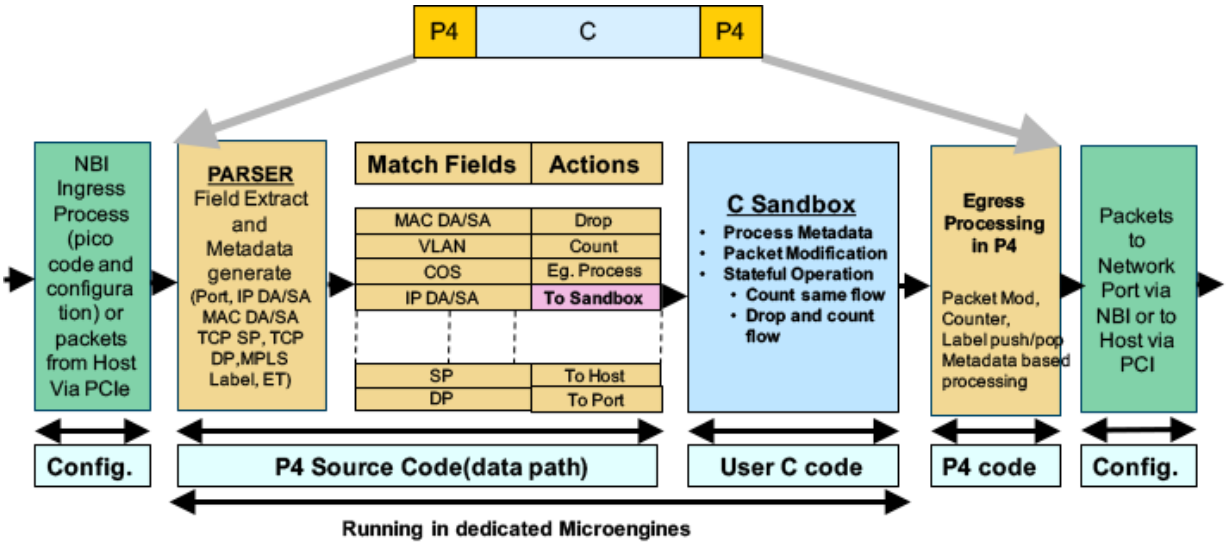

FIGURE 1 – P4 and C data plane programs running in NFP MEs in an Agilio SmartNIC

We implemented five common server-based networking application examples, each on UDP and TCP, in P4 to demonstrate the performance achievable with P4 programs on Agilio SmartNICs. This is shown in Table 2 below:

App Number/Name	Description
1	Stateless ACLs based on 5-tuple
1-Meter	Stateless ACLs based on 5-tuple + metering
2	Stateless ACLs based on 8-tuple
3.1	Stateless ACLs based on 5-tuple + Set Destination IP
3.2	Stateless ACLs based on 8-tuple + Set Destination IP
4.1	Stateless ACLs based on 5-tuple + Decrement TTL + Set Dest MAC
4.2	Stateless ACLs based on 5-tuple + Decrement TTL + Set Dest MAC

***TABLE 2** – Server-based networking application examples*

Figure 2 below shows the test system to measure latency and throughput for these example networking applications on a 40GbE Agilio SmartNIC. For each combination of protocol and frame size, we ran the tests with varying number of flows.

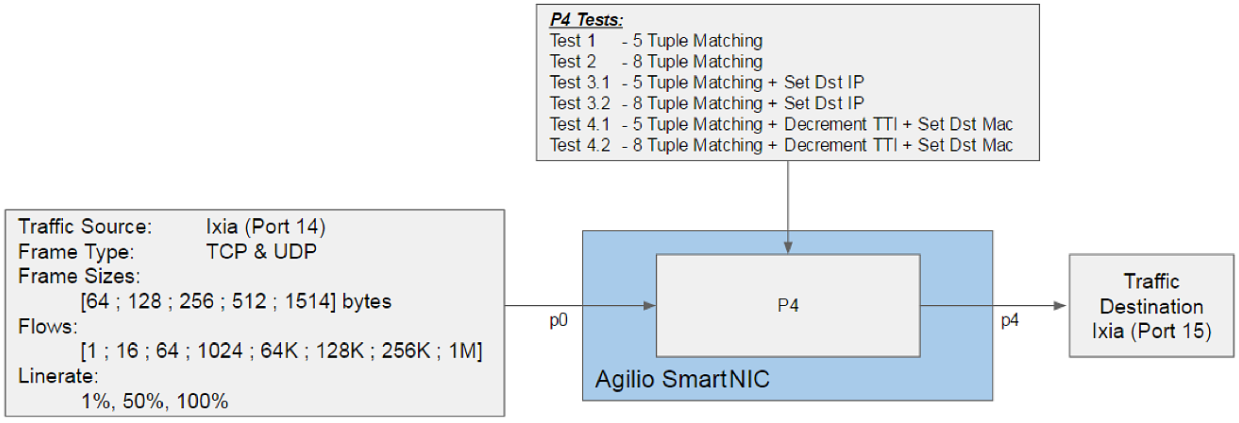

FIGURE 2 – Test system to measure latency and throughput of P4-based data plane implemented in the Agilio SmartNIC

The graphs in Figures 3a, 3b and 3c below show the mean throughput in millions of packets per second (Mpps), Gigabits per second (Gb/s) and latency in microseconds achieved respectively with each of these networking applications across the different flow counts. Application numbers/names shown in Table 2 are used for convenience.

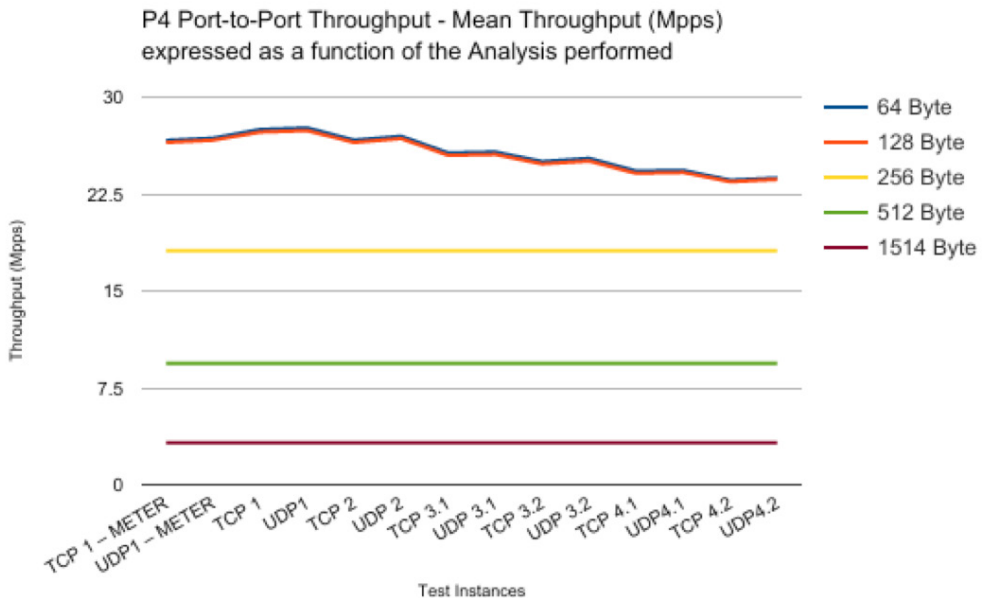

FIGURE 3a – Throughput in Mpps achieved with various P4 data plane applications on a 40GbE Agilio SmartNIC

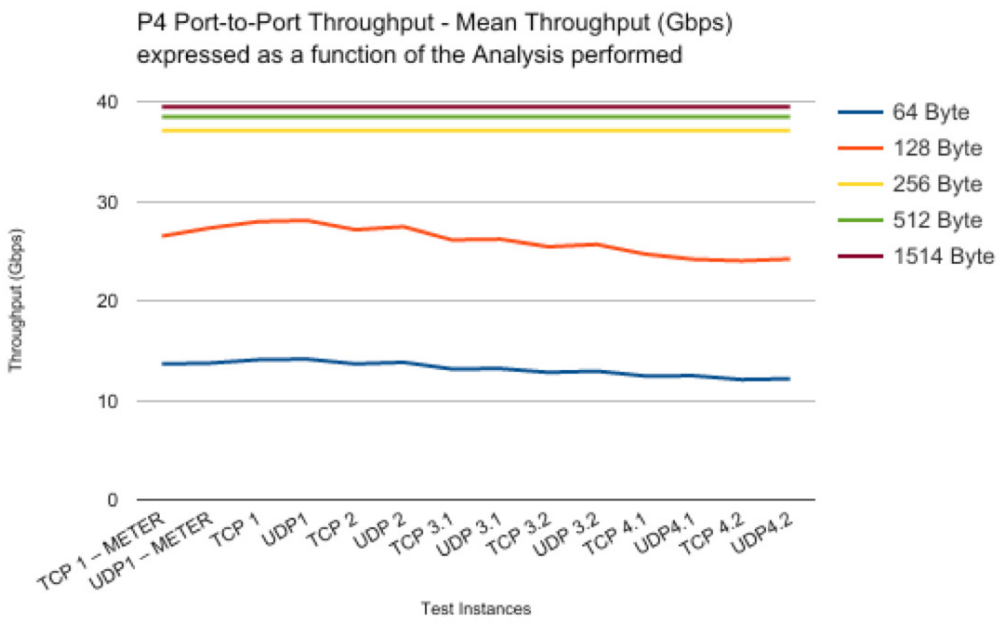

FIGURE 3b – Throughput in Gb/s achieved with various P4 data plane applications on a 40GbE Agilio SmartNIC

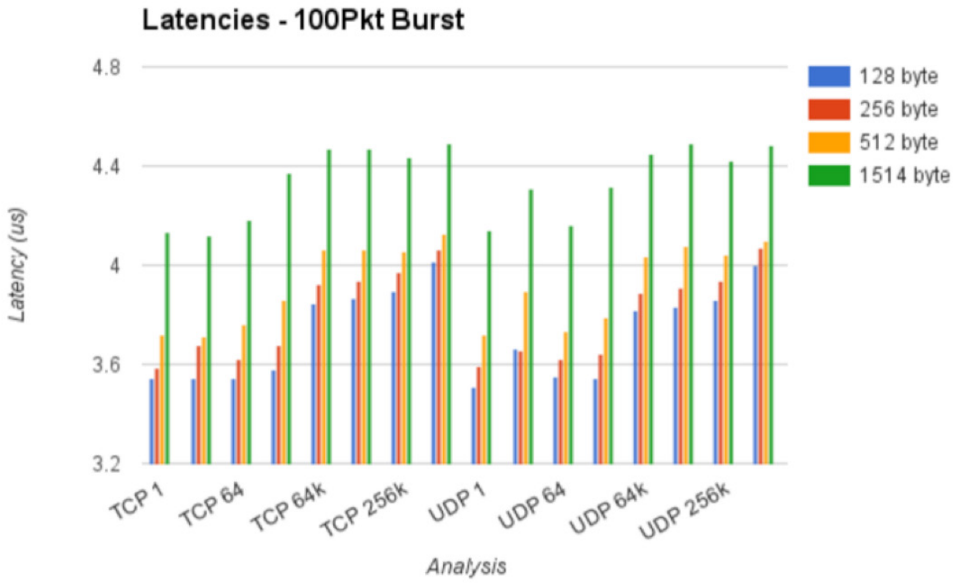

FIGURE 3c – Latency in microseconds achieved with various P4 data plane applications on a 40GbE Agilio SmartNIC

For each of the server-based networking application examples implemented in Agilio SmartNICs using P4, the following observations are noteworthy:

- For small packet sizes, the P4 programs achieve a mean throughput of almost 30 million packets per second.
- P4 programs achieve full 40GbE line rate for packet sizes of 256 Bytes and higher.
- These graphs show average results across multiple flow sizes for each combination of packet size, test type and protocol.

The results are consistent with the performance measured with Netronome's commercial data plane offload products such as **Agilio OVS** and **Agilio vRouter**.

INTRODUCTION TO THE NETRONOME P4 AND C IDE/SDK

Netronome offers developers a full featured integrated development environment (IDE) and SDK to develop P4 programs and augment them with C programs when needed. The IDE extends tools to develop and debug P4 programs similar to those available for other programming languages. Programs are then compiled to the NFP using a compiler as shown in figure 4a and 4b.

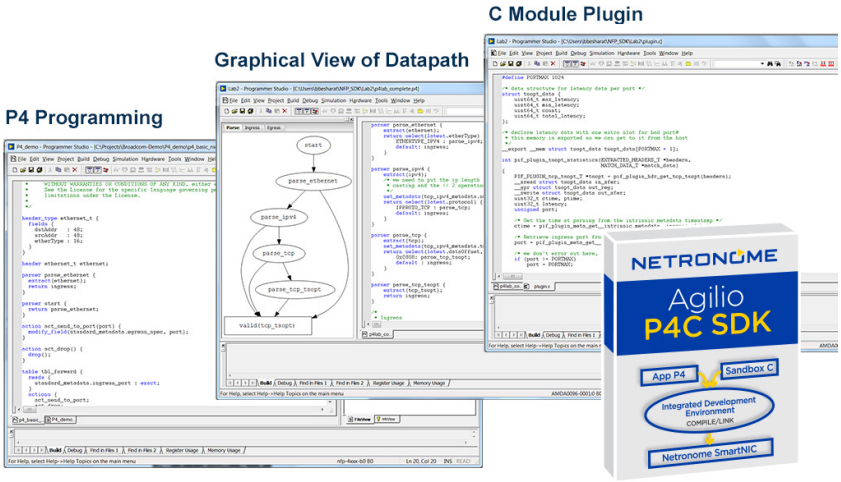

FIGURE 4a – Screenshots of the Netronome P4 and C IDE/SDK

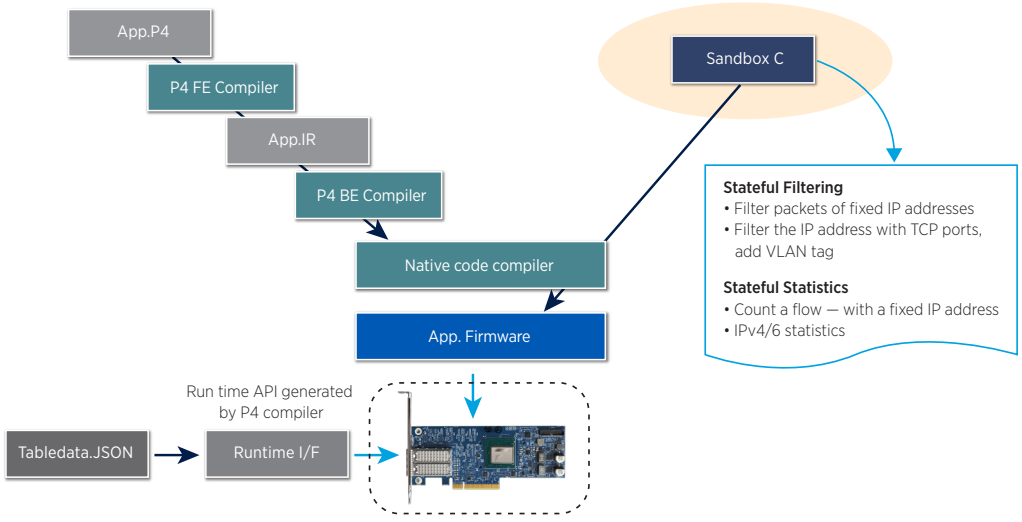

FIGURE 4b – Compiling P4 and C programs into the Agilio SmartNIC

This tool chain is the first production-quality P4 and C IDE/SDK available in the industry and is targeted for P4 developers building server-based networking applications. Netronome offers the IDE for research and development support through the Open-NFP **research portal**. More than a hundred developers from almost 40 universities and companies use the Netronome IDE and Agilio SmartNICs to develop P4 programs, a measure of the IDE's effectiveness in increasing developer productivity.

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