THE WORLD'S FASTEST & MOST PROGRAMMABLE NETWORKS

BAREFOOT NETWORKS

SUMMARY

In this whitepaper, we describe a new approach to building and operating networks.

Today, the fastest networks in the world – for example, those interconnecting servers in big data-centers – are built using *fixed-function* Ethernet switches. It's not that the owners of these networks don't want programmability. Quite the contrary, they dearly want programmability. In fact, many mega data center owners have teams of programmers eager to improve the way packets are processed in their networks. They just can't afford it. Because to get programmability, they would have to give up performance. Programmable switches have historically been unacceptably slow -- between 1/10 and 1/100th the performance of fixed-function switches.

Recent advances in networking research show a path to having both performance and programmability [1]. This research has revealed an instruction-set for processing packets, which in turn lays the foundation for building high-performance domain-specific processors for networking.

In this whitepaper we describe the PISA™ (Protocol Independent Switch Architecture), a new paradigm for processing packets at the highest speeds, under full programmatic control of the user. PISA upends conventional wisdom, showing that users can program the network for themselves, without any degradation in performance, using an open-source programming language.

We show that programmable network systems will not only outperform fixed-function devices, but will lead to a Cambrian explosion of innovation in networking. Data-center networks can now evolve at the same pace as the applications running on their servers. And the insight gleaned from network behavior can be captured in software, and transported across switches, and delivered at industry record breaking performance, thereby becoming a source of competitive advantage for the network owner.

BACKGROUND

Over the past 20 years, infrastructure in the data center and the enterprise have undergone fundamental transformations. We have seen the rise of virtualization, containers, distributed applications, cloud computing, clustering, and much more.

But Ethernet networking has remained stagnant. The speeds and feeds got faster (a *lot* faster). Yet the techniques and methodologies companies use to build, manage, and debug networks are largely the same as those used in 1996.

This is perhaps best illustrated by a story told by a Wall Street investment bank when one of its data centers suffered an outage. With lost revenue measured in millions of dollars per minute, an emergency team was assembled, with experts in *compute*, *storage* and *networking*. The compute team came armed with reams of logs, showing how and when the applications failed, and had already written experiments to reproduce and isolate the error, along with candidate prototype programs to workaround the failure. The storage team was similarly equipped, showing which file system logs were affected, and already progressing with workaround programs. The networking team only had *ping* and *traceroute*.

All the networking team had were two tools invented over twenty years ago to merely test end-to-end connectivity. Neither tool could reveal problems with the switches, the congestion experienced by individual packets, or provide any means to create experiments to identify, quarantine and resolve the problem. Whether or not the problem was in the network, the network team would be blamed since they were unable to demonstrate otherwise.

Simply put, compared to other areas of IT the network is stuck in the dark ages. It is no wonder that financial data centers feel the need to install a *second* monitoring network, more expensive than the first, whose sole job in life is to capture, filter and log the packets in the primary network. It is a sorry state of affairs when network owners can not trust the network to provide visibility into its operation and the means to remediate failures.

One might reasonably expect the new era of Software-Defined Networking (SDN) to come to our rescue. After all, SDN decouples the control plane from the forwarding plane, handing control of the software to those who own and operate networks. Surely they can modify the software quickly to track down bugs and fix them? Unfortunately, this is only true if the bugs are in the control-plane software. If finding and resolving an error requires changing the way packets are processed, or adding probes to data-plane packets, identifying congested links, or generating test packets at high rate to test many different paths, then changing the control plane software is not enough. After all, what is a network but a means to intelligently forward packets between end-points, processing them as they pass through. If we can not change the way packets are processed, then we can not really program the behavior of thenetwork.

WHY PROGRAMMABLE NETWORKS?

A programmable forwarding plane, where users can quickly deploy tests and probes, can reduce the time to recover from an outage. In one interesting example, packets were being dropped due to congestion in a switch once every minute. The intermittency of the failure made it hard to pin-point which application and cross-traffic was causing the problem. In a fixed switch environment debugging this problem is nearly impossible. However, using programmable switches, a programmer added a feature to their switch that inserts the current occupancy of the packet buffers to every passing packet, immediately identifying the rogue application clogging the network. Simple operations like these, which are only understood by those who build and operate networks, are best programmed in the field, as needed, with the full knowledge of how the network is used.

Ability to program the network allows costly and redundant monitoring networks to be eliminated. Monitoring networks can be eliminated because the network can now monitor itself.

Programmable switches allow us to eliminate other expensive, redundant equipment too. For example, big data-centers today commonly deploy expensive middleboxes – load-balancers, address translators, and DDoS detection or mitigation devices – to process packets arriving from the outside world or the untrusted portions of the data centers. A recent study showed that a typical enterprise has more middleboxes than routers, and devotes significantly more resources to buying and maintaining them. Yet when we look closer, we see that each box is essentially a packet processor; it "matches" on headers, performs "actions" based on those matches, and then forwards or filters packets. The reason these boxes are so expensive is because a commercial box must serve the needs of many networks; but its role in any one network is quite limited. If instead, the network owner can program these functions into their network themselves, by exploiting local knowledge of the precise functionality they need, then they can effectively fold the middleboxes into their existing network, for free.

In a particularly telling example, one of the largest data center operator currently uses a complex Network Function Virtualization (NFV) cluster of thousands of servers to load-balance incoming packets across web servers. By programming the top-of-rack switches to spread traffic, keeping track of which servers are up and running, they plan to repurpose thousands of servers to run revenue-generating applications instead.

WHO IS BAREFOOT NETWORKS?

Barefoot is a team of visionaries, experienced technologists and engineers who have created a blueprint for designing and operating the world's fastest and most programmable networks.

Barefoot Networks is on a mission to change networking. Our goal is to make programming your network as simple as programming a CPU. We believe that when the network is fully programmable—that is, both the control plane and data plane are under the control of the end user— the networking industry will enjoy the same innovative explosion as we have seen in software.

To do this, we must build the world's fastest and most agile networks. By building fully programmable hardware, along with the compilers, debuggers and development environments to write programs that work, we enable our users to write solutions rapidly, and to innovate broadly. In short, Barefoot builds the world's fastest switches that are also fully programmable, ensuring the network can adapt to meet the emerging needs of applications.

BUT AREN'T NETWORK SWITCHES ALREADY PROGRAMMABLE?

Yes, slow networks are programmable. NPUs and FPGAs exist and they are flexible. They can modify networking protocols and edit packets and provide a modest level of programmability for those who know how to write microcode or RTL. But they are 1/100th the performance of fixed-function ASICs and as a result they have only found homes in places where performance is not important.

If you want ASIC-like speeds there has been nothing available that is user and field programmable.

WHAT THE INDUSTRY NEEDS IS PROGRAMMABILITY & PERFORMANCE

Barefoot Networks solves this problem by giving the networking industry the best of both worlds: Programmability all the way down to data plane protocols, at industry best speeds.

Not only did Barefoot create the first programmable switch silicon that performs at better than ASIC speeds, we also made programming it easy and openly available.

Working with Google, Intel, Microsoft, Princeton and Stanford, we created a language for programming networks **[2]**. We then open sourced it. P4 - **P**rogramming **P**rotocol-independent **P**acket **P**rocessors – www.p4.org exists now as an independent entity to develop a rich open source ecosystem. P4.org is an open source consortium with over 40 member companies, working together to put an end to the tyranny of closed, fixed-function networks.

Unlike human languages, a programming language such as P4 can *unambiguously* specify the behavior of network forwarding behavior. A programmer can describe the behavior once, then compile the program to run on a variety of different platforms. Portability across a variety of hardware and software switches, makes it much more likely they will correctly interoperate, reducing the time spent testing and debugging. It also makes possible consistent behavior across a whole product line - a network equipment manufacturer can develop a standard P4 program (e.g. my_company_fwding.p4) that is used as the basis for a variety of products, ensuring they are consistent, even if the hardware comes from different sources. Enhancements can be made to the basic program by adding in libraries of P4 code representing other protocols. A common program naturally focusses verification on the program (even allowing provably correct behavior), rather than on the hardware, allowing one verification team to debug code for a broad set of products, without needing a dedicated microcode team for each generation of hardware. P4 offers a programming abstraction that is familiar to network owners (network developers and operators), rather than one familiar to hardware engineers (such as microcode, Verilog or VHDL).

Fundamentally, P4 provides network equipment manufacturers and network owners with the means to *differentiate*. They can make their products, or their entire networks, better than their competitors. One public cloud provider can compete with another by making their network more reliable, less congested, more secure (by isolating traffic better between customers) or simply bigger and faster.

But, there's perhaps an even bigger benefit than just differentiation: Network equipment manufacturers and owners can *exclusively own* their P4 code; they do not need to share it with anyone. We believe this simple and obvious principle -- "you own the code you write" -- is a key enabler of the flourishing software industry, and we believe P4 and programmable network devices will bring the same vibrant ecosystem to the networking industry.

This is in stark contrast to today - if you want to add a new feature to a network you have to share the feature with your chip vendor, and see it appear in your competitors' networks too, defeating the purpose of differentiation. Not surprisingly, equipment manufacturers have been reluctant to add new features; it takes several years before it is added, and then the competitive advantage is short-lived. This has led some to develop their own fixed-function ASICs, to maintain a lead. With P4, this is no longer necessary. Just as we do not share our Java programs with CPU vendors, there is no need to share P4 programs with chip vendors anymore, and there is no need to build special purpose chips merely to keep ahead.

What we are doing isn't magic; it's just very hard. Programmability has long existed in *compute, graphics,* and *digital signal processing,* enabling a thriving community of developers to innovate quickly and write end-user centric solutions.

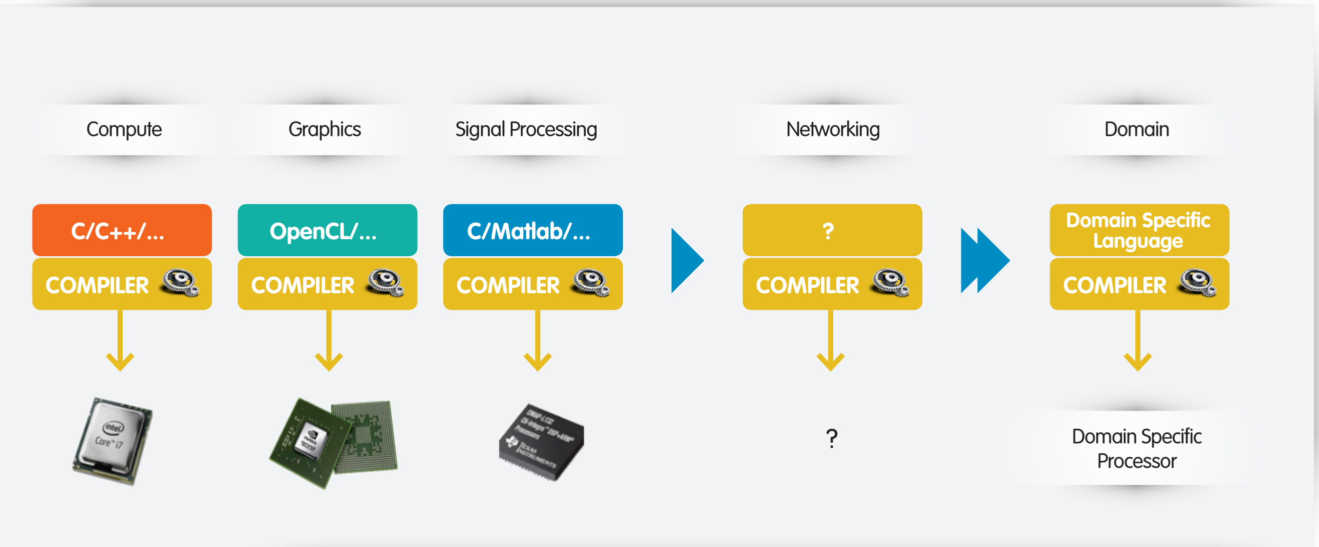

FIGURE 1 Opportunity to "compile" the networks

In the past, signal processing systems were hard-coded in dedicated fixed-function hardware. Many of us remember heated debates about whether or not programmable signal processors could match the performance of fixed-function hardware, for video codecs and base-stations. Then someone figured out a primitive instruction set for signal processing, along with a data model suitable for a variety of different applications. The Digital Signal Processor (DSP) was born, and nowadays no-one builds fixed-function signal processors. Instead, we describe a video codec in a high level language (like C or Matlab), which is then compiled down to run on the DSP chip. The DSP chip knows nothing of video coding; it has the basic primitive operations to allow programs to be compiled and run fast.

Likewise for GPUs, which burst onto the scene, replacing the fixed-function graphics cards built for workstations. Like DSPs, GPUs are domain-specific processors, designed with a primitive instruction set optimized for graphics. Games programmers write 3D games in a high level domain-specific language, such as OpenCL or CUDA, which is compiled to run on the GPU. The GPU is agnostic to, and unaware of, how games are written; it merely contains the primitive operations needed to execute a huge variety of programs.

More recently, the TPU has emerged as a domain-specific processor for machine learning. Once again, the processor does not know about specific machine learning applications; it is designed to execute a carefully optimized instruction set, and has a data model well suited to neural networks. Programs are compiled down to run on the TPU, which was designed to run machine learning programs much faster than regular CPUs.

Barefoot is simply bringing the same, long-overdue benefits to networking. The PISA architecture does for networking what the DSP did for signal processing, the GPU did for graphics and the TPU is doing for machine learning. It is putting full control in the hands of the network owner.

To do this, we took a step back and identified a small, primitive instruction set for processing packets - about 11 instructions in all - and a very uniform pipeline of programmable stages to process packet headers in rapid succession. Programs are written in a high level domain specific language (P4), then compiled down by the Barefoot Capilano compiler, and optimized to run at full line-rate on the PISA device.

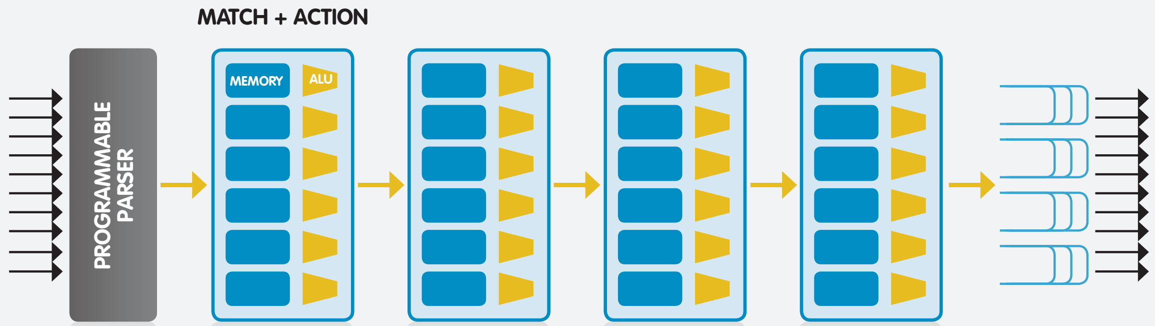

FIGURE 2 The PISA switch architecture

By so doing, Barefoot aims to make networks more agile, flexible, modular and less expensive.

INTRODUCING THE BAREFOOT NETWORKS SOLUTION

The Barefoot Networks solution is comprised of three parts: **FIRST: TOFINO™**, the world's fastest switch silicon, that happens also to be programmable. **SECOND: P4**, the open source programming language used to write programs for Tofino. And **THIRD: CAPILANO**, the compilers and development tools needed to compile and debug programs to run on Tofino. Let's look at these in a bit more detail.

1. BAREFOOT'S TOFINO IS A 6.5TB/S (65 X 100GE OR 260 X 25GE) FULLY USER-PROGRAMMABLE SWITCH

Proprietary homegrown designs by data-center owners

Open-source designs (e.g. Open Compute Project, OCP)

Via supported switch products from existing NEMs (Network Equipment Manufacturers)

Via "whitebox" switches from ODMs

2. THE P4 PROGRAMMING LANGUAGE

First created by Barefoot, Google, Intel, Microsoft, Stanford and Princeton.

P4 allows programmers to unambiguously specify forwarding behavior.

Allows P4 programs to be compiled and run on a variety of hardware and software targets, including CPUs, hypervisor switches and FPGAs.

Now developed by the independent open-source P4.org community.

P4.org has over 40 members.

2. THE BAREFOOT CAPILANO SOFTWARE SUITE: ALL THE SOFTWARE NEEDED TO WRITE P4 PROGRAMS, AND TO HAVE THEM RUN SUCCESSFULLY ON TOFINO, INCLUDING:

P4 Compiler, Debugger and Integrated Development Environment (IDE)

P4 Simulation Environment

Open APIs/SDK to allow others to publish open-source code connecting to Tofino.

Reference P4 programs illustrating new applications to run in the network.

Integration with 7 different open-source network operating systems including OpenSwitch (Hewlett Packard Enterprise), SONiC (Microsoft Azure), and FBOSS (Facebook).

WHAT ARE THE CASES FOR P4 + CAPILANO + TOFINO ?

Barefoot customers have been writing P4 programs for their own use cases since the middle of 2015. So far, use cases (open-source and proprietary) fall into five categories:

1 RAPIDLY PROTOTYPING AND DEPLOYING NEW PROTOCOLS

The standard bodies keep producing new encapsulation protocols to support network virtualization. GENEVE, NSH, VxLAN GPE are recent examples. Meanwhile, large data centers have started to introduce their own custom protocols to isolate one tenant from another, while allowing interconnection of tenants with their applications. Each custom protocol is designed to give competitive advantage, and is based on intimate local expertize built up from years running big networks at scale. For example, a large online service provider recently extended a standard encapsulation protocol to accelerate load-balancing, by caching decisions made when a new connection begins. Others are designing custom congestion control mechanisms, OAM, discovery, and high availability protocols.

2 REMOVING UNUSED PROTOCOLS, SIMPLIFYING AND STREAMLINING THE NETWORK

While there are hundreds of protocols in use around the world, any one data-center supports only three or four. The problem is, each data center uses a different set. Therefore, fixed function switches must support the superset. A data center finds that precious switch resources are hard-coded to protocols they do not use. Imagine a DC network operator reducing the size of the L2 forwarding table and repurposing the memory to increase the L3 IP routing tables instead. With Tofino, for example, a DC can easily increase the capacity of an IP routing table from 300K to 1.2M, allowing them to build even bigger networks and address many more servers. Throwing out unused protocols also means less to go wrong; data-center owners report outages caused by protocols they do not even use but were hard-coded into their switches! Because they do not use them, they have no expertize to debug them. With P4, you only include the protocols you need, focusing precious tables as-needed, simplifying the switch and making outages less likely.

3 ENSURING COMPLETE VISIBILITY OF THE NETWORK AND HOW IT PROCESSES EVERY PACKET

Network monitoring is becoming a huge application for programmable networking. First, introducing better monitoring features can be done in hours, rather than the years needed to change forwarding logic. Second, no chip or equipment vendor can possibly know what to monitor better than the network operators themselves. With P4, Capilano and Tofino network operators can quickly add powerful monitoring, analysis, and diagnostics features for themselves, in the field - and our users have already started to do so. One very popular technique, made possible by P4-capable switches, is called "In-band Network Telemetry" (INT) **[3]**. In a nutshell, the network operator decides exactly what information she wants to observe: For example, the precise latency taken by a packet through each switch is passes through, or the other packets it shared a queue with, the version of the software, the table entries it matched on. Every data packet can be recruited as a probe, without creating any new traffic. Such unprecedented visibility is made possible by placing programmability in the operator's hands. And of course, a baseline implementation is already available as the open-source "INT.p4" program. Programmers are already looking at how to fully-automate data collection and remediation, as a step towards making self-managed networks.

4 INTEGRATING MIDDLEBOX FUNCTIONS INTO EVERY SWITCH

Many network operators use only a tiny fraction of the features of their middleboxes. With local expertize, they can program the features they need directly into their network, eliminating huge numbers of expensive middleboxes. Our customers are already programming the Barefoot Tofino switch to provide middlebox functions that scale as the network grows, at no additional cost. IN most cases, the middlebox functions operate much faster than before, because they run on Tofino at full line-rate, rather than on a conventional CPU. In one scenario, Layer-4 load-balancing functionality was integrated into the switch, maintaining the connection state for up to tens of millions of connections with one hundred thousand servers. The DIP pools can be adjusted - smaller or larger - without disrupting existing connections. All of this was achieved by writing a few hundred lines of P4 code. The same approach can be used to integrate other middlebox functions like firewalls, intrusion detection systems, address & port translators, traffic de-duplicators, etc. We anticipate a revolution in the way middleboxes are folded into the network.

5 IMPLEMENTING PART OF DISTRIBUTED APPLICATIONS DIRECTLY IN THE NETWORK

A big data center runs many huge distributed applications; and also has a network with tens of thousands of switches. It is interesting to ask if the switches can accelerate distributed applications, offloading the servers. Recently, researchers demonstrated how the Paxos consensus protocol can be added to the network by implementing a portion of it in P4, and added to switches. This led to many of orders of magnitude acceleration of distributed applications. Other have built new key-value management services directly into the network data plane. We anticipate many new fast in-network services to be seamlessly integrated into networks, for free.

This is only the beginning - the tip of the iceberg - of how our customers will use our technology. Our goal is to put the means to differentiate into their hands, so they can create the ideas for the networks they know best. P4, Capilano and Tofino start a revolution in networking by empowering customers, researchers and the open networking community to innovate for themselves.

IS BAREFOOT A SYSTEMS SOFTWARE CHIP COMPANY?

In the past, networking systems were vertically integrated and proprietary, with closed software created and controlled by equipment vendors. But the networking industry is changing rapidly, with disaggregation blurring the lines between chip companies, software companies and systems companies. Today, network operators can choose from among several avenues. Some build their own network systems from the ground up, using merchant silicon in custom switches, running proprietary switch operating systems and homegrown control software. Others buy "whiteboxes" from ODMs, running open-source software. And many others build networks from switches delivered and supported by networking equipment vendors, such as Cisco, Arista, HPE, Dell and Juniper.

Barefoot is agnostic as to how its technology is provided to the end user. Barefoot sells switch hardware - and provides the software to program it - to equipment vendors, to ODMs and also directly to large data-centers.

Barefoot's primary interaction with the customer is through *software*. Our goal is to help as many people as possible write P4 programs to run in their switch hardware. They declare the behavior they want, then compile and run it on Tofino. To this extent, some consider us a software company; indeed our culture is very much like software companies, and a majority of our R&D staff are software developers. We also build chips, because no-one else builds programmable switch silicon that is close to the performance we need. Therefore, we have deep expertize in chip design, from the architecture down to the physical design. To this extent, we are a chip company. Finally, we build high quality reference systems, of sufficient quality to run in data-centers and to be manufactured by ODMs. This means we are sometimes considered to be a systems company too!

We tend to think that Barefoot is to the networking industry what nVIDIA is to the graphics industry. Like us, their primary interaction with customers is in software. But their revenue is mostly from chips.

At the end of the day, networking systems are heading towards being built from three layers: A P4-programmable Tofino switch at the bottom, with a Linux-based operating system above, with proprietary control plane applications on top. Barefoot's value proposition is to provide the fastest, most programmable, P4-optimized switch hardware for the entire industry.

CONCLUSION

Barefoot brings the twin pillars--performance and programmability--together for the first time in the history of networking. The combination of the Tofino programmable switch chip, the P4 programming language and Capilano tool set are revolutionary--best in industry performance with the first ever true switch programmability.

While the specific benefits of performance and programmability are broad, we believe an often overlooked effect will turn out to have the largest impact.

Performance plus programmability changes the ownership of networking insight

In the old world, network insight was the domain of chip companies or system companies. And they charged very high prices to lend you some.

In the new world, the owner of a network can write a program to optimize the behavior of their network running their applications. They do not need to share this with a system or with a chip vendor. This insight, captured in a P4 program, becomes their IP. They can then take this IP and run it on switches from different vendors, or across generations from the same vendor, or even buy white boxes and run it there. Suddenly, because it is captured in P4 software, network insight is portable.

Barefoot's Tofino switch chip, P4 programming language, and Capilano tools, provide the ability for networking professionals to capture, encode, and transport networking insight--thereby creating a new and defensible source of competitive advantage for their organizations.

REFERENCES

[1] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz. *Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN.* In ACM SIGCOMM, pages 99–110. ACM, Aug. 2013.

[2] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker. P4: *Programming protocol-independent packet processors.* ACM SIGCOMM Computer Communications Review, July 2014.

[3] Barefoot Networks: Changhoon Kim, Parag Bhide, Ed Doe, VMware:Mukesh Hira, Bruce Davie, Arista: Hugh Holbrook, Dell:Anoop Ghanwani, Intel:Dan Daly. *In-band Network Telemetry.* P4 Language Consortium.