P4 Workshop, 2023

The Power of Fully-Specified Data Planes

Rob Sherwood
NEX Cloud Networking Group, CTO

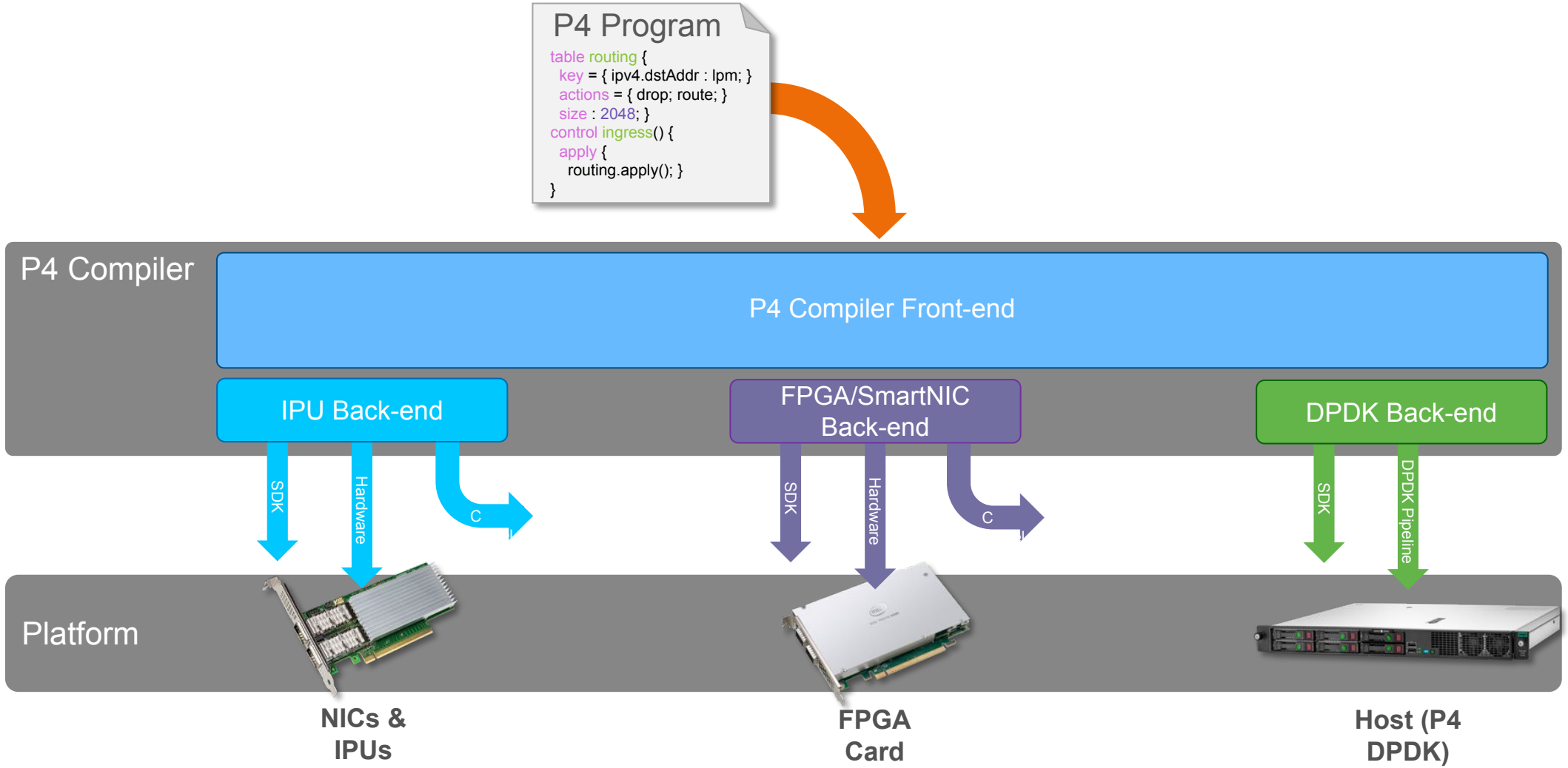
Talk Outline

- About Me
- The P4 We Know: Automated Dataplane Configuration
- Networks are complex under specification, surprises
- P4 provides a machine read-able data plane specification
  - Simplifies testing
  - Simplifies hardware qualification
  - Simplifies ***software*** evolution
  - Formal language removes unspecified behavior
- Guesses at the future of P4
  - The delta between “mostly” vs. “fully” specified

About Me

- Current: CTO of Intel's NEX Cloud Networking Group
- Past lives:
  - Cloud Service Provider: Meta- FBOSS and Network Foundation Teams
  - Startup: Big Switch Networks – CTO
  - Academia: Stanford Clean Slate Lab – helped launch SDN + OpenFlow
  - ISPs: Deustche Telekom, AT&T (intern)
- A lot of time ***manually programming*** data planes
  - Steep learning curve for each new device
  - Highly Complex Code a little, test a little "tweak and pray"
  - Many war stories from ugly surprises
- This sounds like a job for… <*queue superhero music*>

Intel Vision: Common Programming Model with P4

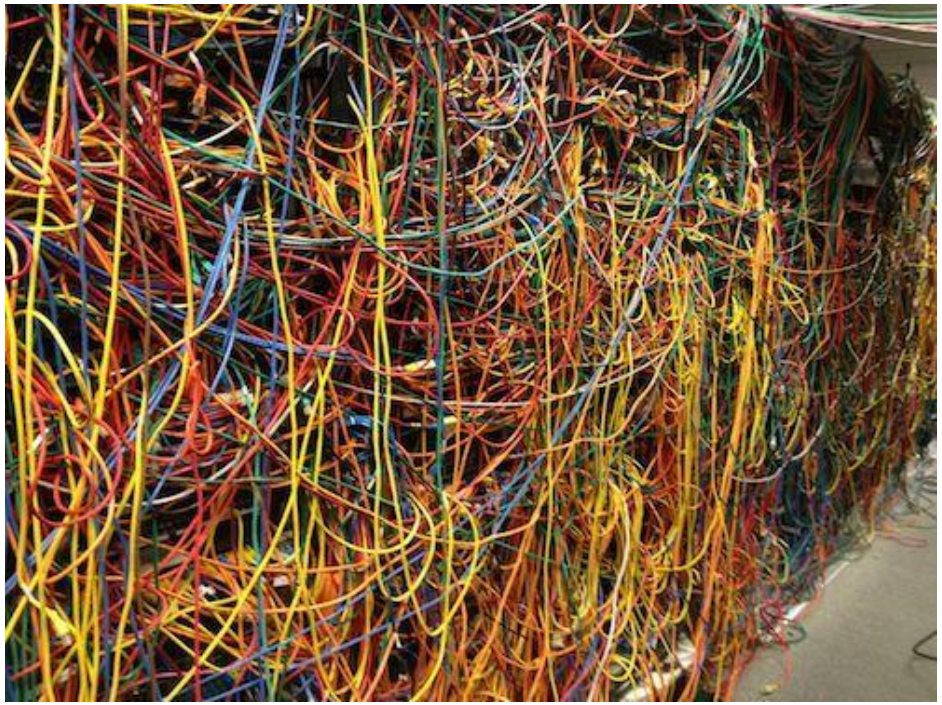


P4 – Automated Data Plane Configuration!

- The Good (We all know)
  1. Describe desired dataplane in p4
  2. Ask a compiler to map to hardware
- Work TODO (We already know)
  - Many packet ASICs do not have a simple/machine-readable resource description
  - Hard to write a good P4 compiler
    - A “No” result may not mean “No”
  - Most network changes don’t require a new data plane configuration
    - My experience: every ~3-6 months

Bigger, Related Problem: Networks are Complex

- Unclear feature interaction, implementations
  - Causes unspecified behavior
- Forces “bug level” interoperability
  - Manual, surprise-prone effort
- Any change to the network may have surprising effects
  - Again, “tweak and pray”
- How do we tame this complexity?
  - An “every change” problem (hourly/daily)



Scott Shenker, 2011 – "Networking Needs Abstractions"

- First Open Network Summit
- "Networking is a big bag of protocols"
- Key point: by decoupling architectural components, we enable independent innovation
- P4 (mostly) describes the dataplane
  - ***P4 is also an abstraction language!***
  - Partitions the control from dataplane
  - Allows control plane and dataplane innovations to evolve in parallel

"The Future of Networking, and the Past of Protocols - Scott Shenker"

Search terms: "Shenker ONS 2011"

Claim:

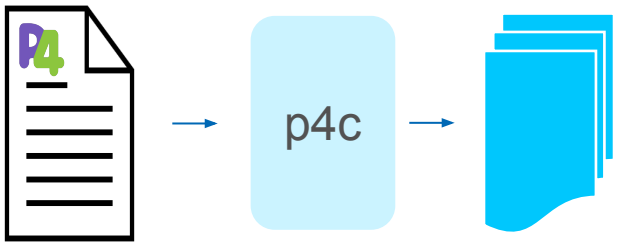
P4 was created to solve:

automatic data plane configuration

but the bigger, long-term contribution may be:

fully-specified data plane language.

Original Purpose



Long-Term Contribution

Talk Outline

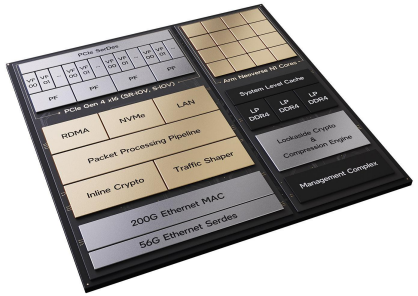
- About Me
- The P4 We Know: Dataplane Configuration Requests
- Networks are complex under specification
- P4 provides a machine read-able data plane specification
  - Simplifies testing
  - Simplifies hardware qualification
  - Simplifies ***software*** evolution
  - Formal language removes unspecified behavior
- Guesses at the future of P4
  - The delta between “mostly” vs. “fully” specified

P4 Simplifies Testing

- Control Plane Testing
  - P4 is a “mock layer” for the data plane
  - Easy to remove physical data plane and swap in, e.g., ***P4 DPDK***
    - https://github.com/p4lang/p4-dpdk-target

- Data plane + P4 tool chain Testing
  - P4 is a signal of intent allows creation of a “test oracle”
  - Autogenerate packets + table entries as tests, ala ***p4testgen***
    - https://opennetworking.org/wp-content/uploads/2022/05/Fabian-Ruffy-Final-Slide-Deck.pdf
  - Academic paper under submission
  - Found real bugs in production stacks; adding support in Intel products

P4 for Hardware Qualification



- Question: *“Can software X be ported to/supported on hardware Y?”*
- Historic answer: attempt complex port, if it takes too long or hits roadblock, declare “no”, else “yes” (assuming good test coverage)
  - Bad: expensive and slow
  - Worse: correctness not guaranteed (either ‘yes’ or ‘no’ cases)!
- Better answer: Write software X to a p4-abstracted data plane!
  - Simplify the question to: *“Can we map X.p4 to hardware Y?”*
  - Easy case: use a compiler to compute/verify the mapping
  - Reasonable case: manual mapping via p4 is easier than full software port
  - Both faster/simpler and more likely to be correct
- Evidence with both ***IPU E2000*** and **FPGA-based** platforms

P4 for Software Evolution

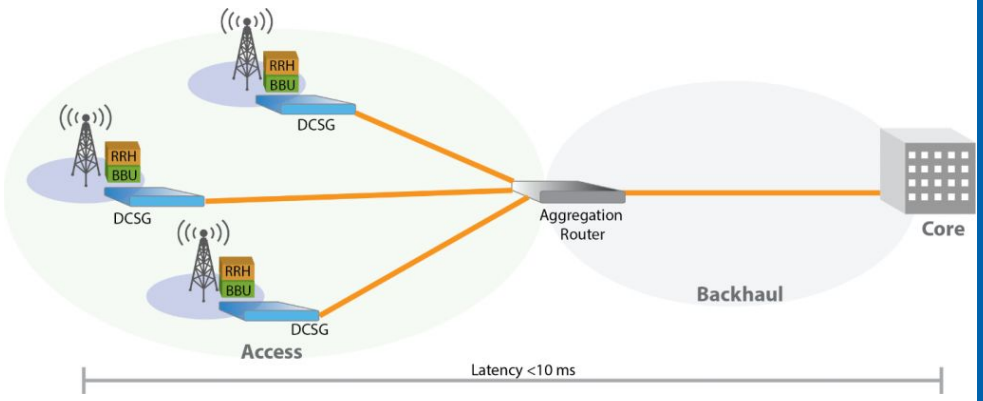
- Add “Support for P4” to a software data plane
  - Instant, well-understood data plane evolution mechanism
  - Clearer semantics for hardware-offload, if available

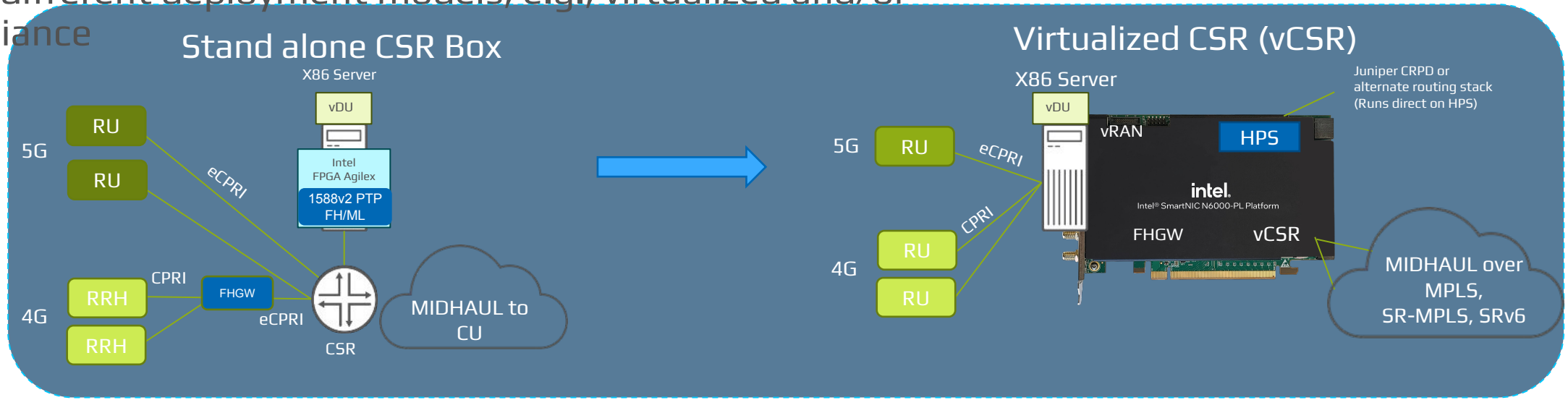
- Example #1: Linux Kernel: Traffic Classification (TC) System
  - First set of ***p4tc*** classifier patches posted for review in Feb 2023
  - Enhancements based on feedback are in the works
  - p4tc Home https://www.p4tc.dev/
- Example #2: P4 for Kubernetes
  - https://ipdk.io/documentation/Recipes/PaaSOffloadKubernetes/
  - https://github.com/ipdk-io/k8s-infra-offload

Example #3: Software + Hardware with P4

Cell Site Router & virtual Cell Site Router

- A cell site router aggregates traffic from Baseband Units and then backhauls it to the core
- 5G creates new use cases require new needs at cell site: traffic prioritization, timing and synchronization, security, etc.
- P4 enables rapid solution development, feature velocity, and different deployment models, e.g., virtualized and/or appliance





P4 for Formal Behavior Specification

- Question #1: Packets are spread across ***N*** LAG ports and ***M*** ECMP next-hops at the same time: what is the distribution?
  - Min(N,M) buckets? Max(N,M)? N*M buckets? Something else?
- Question #2: If you combine eVPN over MPLS-TP, which layer handles broadcast traffic?
- Meta point: *have all the interactions between all features been fully thought through?*
- Meta answer: No – but wouldn't be nice if, e.g., RFC's were written with a machine-parsable language (p4) to clarify this?

Talk Outline

- About Me
- The P4 We Know: Dataplane Configuration Requests
- Networks are complex under specification
- P4 provides a machine read-able data plane specification
  - Simplifies testing
  - Simplifies hardware qualification
  - Simplifies ***software*** evolution
  - Formal language removes unspecified behavior
- Guesses at the future of P4
  - The delta between “mostly” vs. “fully” specified

Future: Towards a “Fully-Specified” Data plane

- P4testgen had to manually code “p4 architecture” in C++
  - E.g., Portable {Switch,NIC} Architecture implicit behaviors
  - Currently no machine-readable definition of a p4 architecture
- Currently no mechanism to model the MMU/Traffic Manager
  - E.g., define queue properties, queue monitoring, drop paths, etc.
- These are less important for “P4: Automatic Dataplane Configuration”
- Will become important for “P4: Data plane specification language”
- Wistful: A “higher-level than RTL” machine parsable description of hardware resources would be nice/improve compiler quality…

Conclusion

- Intel continues to make significant investments in P4
  - Across many product: IPU, FPGA, x86, etc.
  - Across many domains: cloud, edge, RAN, etc.
- P4 continues to evolve
  - “Automatic dataplane configuration” still an important use-case
  - New use-cases emerging around P4 for data plane abstraction
  - Fully-specified data planes can tame network complexity
- Will continue to lead in new directions for P4
  - State of the art testing
  - Hardware qualification
  - Software evolution


intel