Exposing Data Plane Programmability on Turn-Key Network Devices
Opportunities, challenges, and options

Mario Baldi
Programmable Switch Deployment Flavors

Whitebox  Turn-key  Hybrid
Whitebox Deployment

- Maximum flexibility
- Maximum disruption/risk
- Significant barrier
  - Who can code in P4 today?

NOS (e.g., SONiC)

Remote controller/NOS (e.g., ONOS)

Platform vendor (Cisco)
Chip vendor (Barefoot)
Customer/open source

Programmable chip
Turn-key Deployment

- Deployment as usual
  - Familiar features and interfaces
- Resource optimization
- Future proof
- Feature agility
- Streaming telemetry

- No flexibility
  - No custom features and protocol support

Platform vendor (Cisco)
Chip vendor (Barefoot)
Customer/open source

Profiles
profile1.p4
profile2.p4
profile3.p4

Programmable chip

© 2018 Cisco and/or its affiliates. All rights reserved. Cisco Public
Open Platform

- Deployment as usual
  - Familiar features and interfaces
- Resource optimization
- Future proof
- Feature agility
- Streaming telemetry

- No flexibility
  - No custom features and protocol support

Chip vendor (Barefoot)
Customer/open source

Same as Turn-key

NOS (e.g. SONiC)

SAI

switch.p4

© 2018 Cisco and/or its affiliates. All rights reserved. Cisco Public
Hybrid Deployment

- Best of breed
- Deployment as usual
  - Familiar features and interfaces
- And also flexibility

Without the added risk!
Hybrid Deployment Challenges

Do not break what works

- Vendor data plane code is well tested
- ... and we don’t want to need regression testing

Don’t want to show, don’t want to see

- Vendor code and custom code may be confidential
- Not practical to familiarize with a lot of vendor code to just write a few lines

Resource availability

- Still “limited” on current chips

Data/control plane dependence

- Net OS should keep working
- Net OS should not be aware of custom data plane functions
In a nutshell

P4 and its ecosystem were not designed for *incremental programming*

Single programmer

Single source code

Single control plane
Possible Options

- Platform vendor (Cisco)
- Chip vendor (Barefoot)
- Customer/open source
Physical Separation

Tofino

vendor

customer

Net OS

P4Runtime

vendor.p4

cu.p4

Net OS

P4Runtime

vendor.p4

cu.p4

Custom App

© 2018 Cisco and/or its affiliates. All rights reserved. Cisco Public
Incremental Programming Workflow

- **Cu.c**: Must come from chip vendor
- **Cu.p4**: Other pipelines pre-loaded with vendor.bin
- **Cu.exe**: Loaded on dedicated customer pipeline
- **P4 Compiler**: Favorite SDE
- **Net OS API**: PD-API.o
- **Cu.bin**: NXOS
Software Solution

Platform vendor (Cisco)
Chip vendor (Barefoot)
Customer/open source
What about the challenges we mentioned earlier?

- Modify the language
- Offer a programming environment
Language Design Working Group

Modularity can help with incremental programming

- Sub-working group to introduce modularity in P4
  - March 2018

- Started focusing on polymorphism
  - Generic data type
  - Generic function type

- Intent to focus on modularity for incremental programming
daPIPE
Data Plane Incremental Programming Environment

Identify constraints on new code

Impose those constraints on the program
Support developers and streamline their task (while enforcing needed constraints)
Thank you

Any questions?