P4 - What We Can Expect from Switching System Vendors
Opportunities, Tools, and Benefits

Mario Baldi
Politecnico di Torino
Develops, sells, supports full systems

- Hardware
  - (ASIC)
  - Platform
- Data plane functions
- Operating system
- Control applications
Why focusing on switching system vendors?

- Device users are accustomed to their products
  - Familiar CLIs and APIs
  - Documentation
  - Technical support SLAs
- Turn-key switches might provide a smooth adoption path for P4
  - Less disruptive change
  - Less risk
  - With some exceptions (e.g., hyperscalers)

It is key that switching system vendors benefit from P4

... so that they’ll embrace it
Opportunities
How do we go through it?

How a switching system vendor can take advantage of a P4 programmable switching ASIC in their system
Accelerate Releases

Shorten the time between regular releases that contain new data plane features and bug fixes

- Be current
- Beat competitors
- Ride the wave of new technologies when they are hot
- Keep customers engaged
Optimize Resources and Scale

Enable (dynamic) feature selection

• By Vendor
• By Customer

Scale

- Address new use cases
- Address specific markets
- Acquire new customers
Unique Features Not Available in ASICs

• Unusual, possibly application dependent functions
  • Innovative load balancing algorithms
  • Big data and machine learning support
  • Deduplication algorithm in data broker

• As well as less fancy, but nevertheless important ones
  • Multicast-to-unicast translation

Unique advantage over competitors

- Improve sales
- Attract new customers
- Enter new markets
Customer Requested Features

• Examples/use cases
  • Existing protocols not yet widely adopted
    • MAP-T, SRv6, BIER
  • New protocols just standardized or not yet stable or specific techniques
    • Multicast address translation

Differentiation from competitors

Improve sales

Solidify relationship with existing customers
Customer or Third Party Development

Enable customers to implement their own features on the switch, while taking advantage of existing pre-packaged features

• Examples/use cases
  • Proprietary techniques and protocols
  • Timestamp-based switching (Fox Advanced Technologies)
  • Channel stuffing (DISA SDN RFI MAC0098)

Differentiation from competitors

Improve sales

Attract new customers
Value Brought by Data Plane Programmability

Shorter time to market
- Accelerated releases

Reduced investment/commitment/cost required for the hardware implementation of a feature
- Unique and customer requested features

Flexible feature support
- Optimize resources

New “feature”
- Custom and third party development

Goal: maximize the benefits

Is this reduction significant?
Let’s look at the process

Product management

ASIC design and fabrication

Platform

Control plane software

Quality assurance

Technical support

Roughly a 2-3 year span

A new ASIC every 2-3 years

18-24 months
With a programmable data plane chip

Product management

P4 Development

Platform

Control plane software

Quality assurance

Technical support

Roughly a 2-3 year span

18-24 weeks

Quite good reduction

No more “time discretization”
But we can do much better ...

... with process changes

- Product management can be more lighthearted
- Development can be organized around smaller releases
  - Small number of features
  - Shorter cycles
... and with some technical changes

Very modular software

Heavy reliance on testing automation
In the meantime ...

- Implementation of data plane function and fast implementation of control plane
  - Not tightly integrated with NOS
  - Possibly application running on it
  - Not dependent on the “normal” release cycle
  - Possibly using solutions for customer/third party programming
    - E.g., daPIPE

- Users start field trials/sales force proposes the solution

- Improvements are made based on results of field trials

- Confidence is gained on the market opportunity

- If feedback is positive, move to full integration with NOS
Optimize Resources and Scale
Challenges

Each profile needs the NOS to work with it

- Development
- Testing

Current process generates a huge number of different branches

Probably no profile is a perfect fit
What can we do?

Features a la carte

- Modular/composable NOS
- Some level of automated regression testing
Customer or Third Party Development
Challenges

Do not break what works

- Vendor data plane code is well tested
- ... and we don’t want to need (very comprehensive) 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
**daPIPE: DData Plane Incremental Programming Environment**

**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**
- NXOS should keep working
- NXOS should not be aware of custom data plane functions

**Identify constraints on new code**

**Enforce those constraints on the program**
Control Plane

- Cisco Apps
  - BGP
  - OSPF
- Customer Apps
  - Cfg
  - Ctrl plane

- SW (mostly) control plane
- HW data plane

- Infrastructure
- HAL
- Cisco.p4
- Cu.p4

- Guest Shell (container)
- NXOS
- Controlled data plane API access
- APIs generated by compiling P4
- Programmable ASIC
Components of the Solution

- daPIPE Graphical User Interface
- Nexus 34180YC
- daPIPE build environment
- Control program
- PD API
Main window
Existing header view
Adding RTP parser
Define control flow
Compile and upload to a switch
This is “simple” …

composable
data plane modules

are the next big challenge
Thank You