



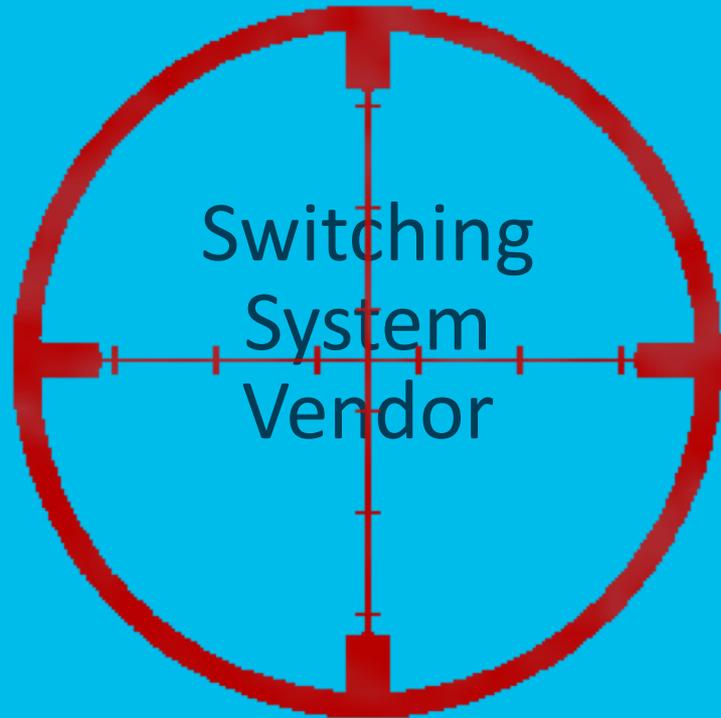
# P4 - What We Can Expect from Switching System Vendors

Opportunities, Tools, and Benefits



**Mario Baldi**  
Politecnico di Torino





Develops, sells, supports  
full systems

- Hardware
  - (ASIC)
  - Platform
- Data plane functions
- Operating system
- Control applications

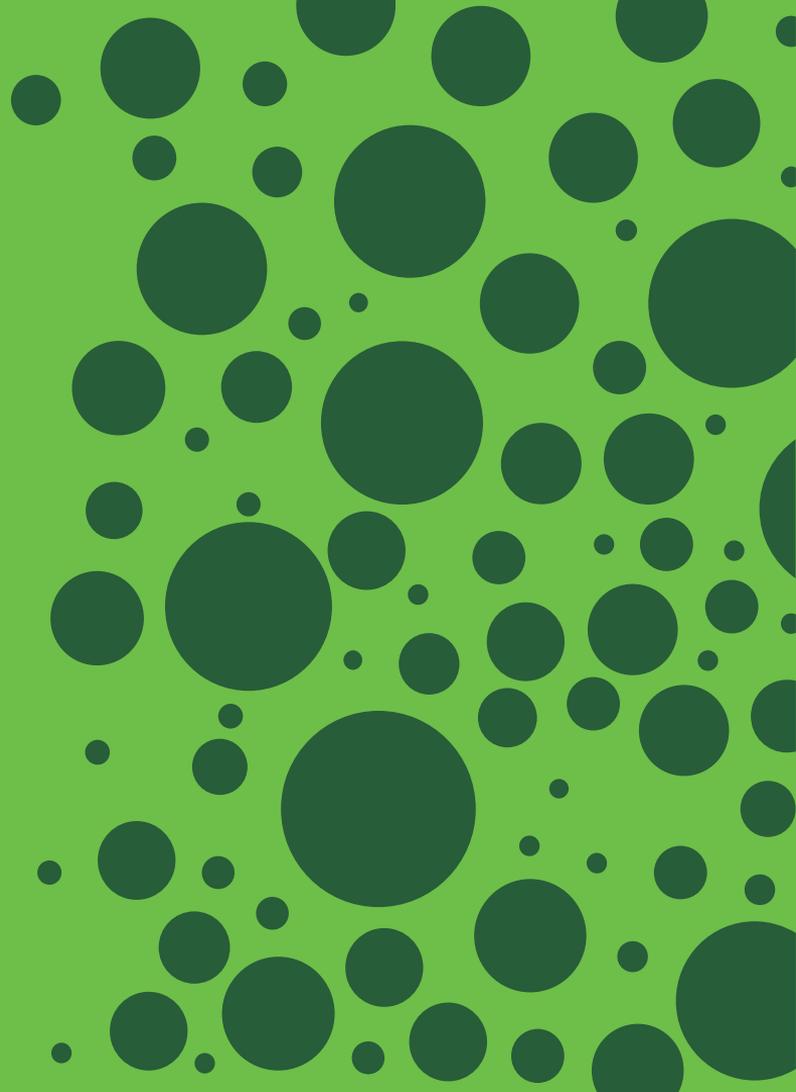
# Why focusing on switching system vendors?

- 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

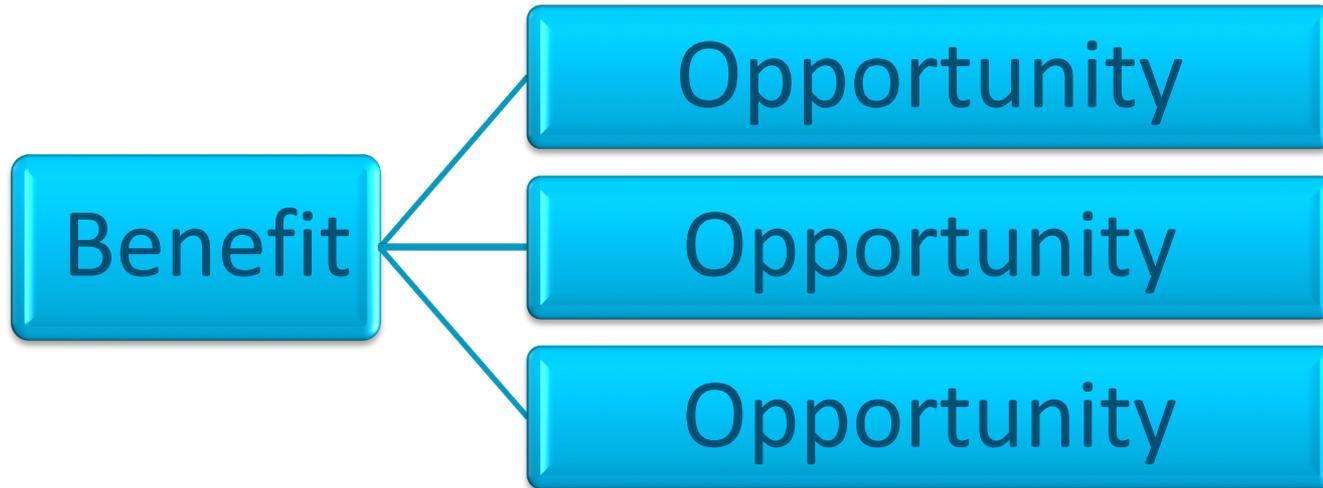
... so that they'll embrace it

Opportunities



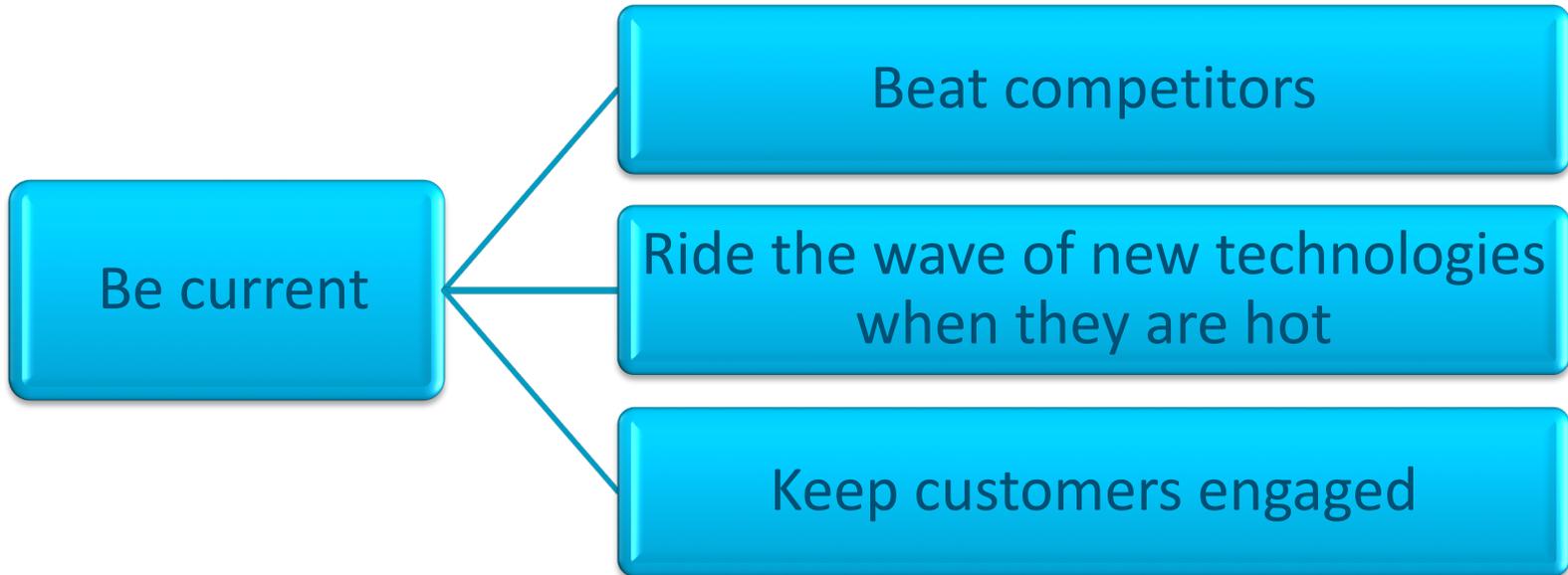
# How do we go through it?

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



# Accelerate Releases

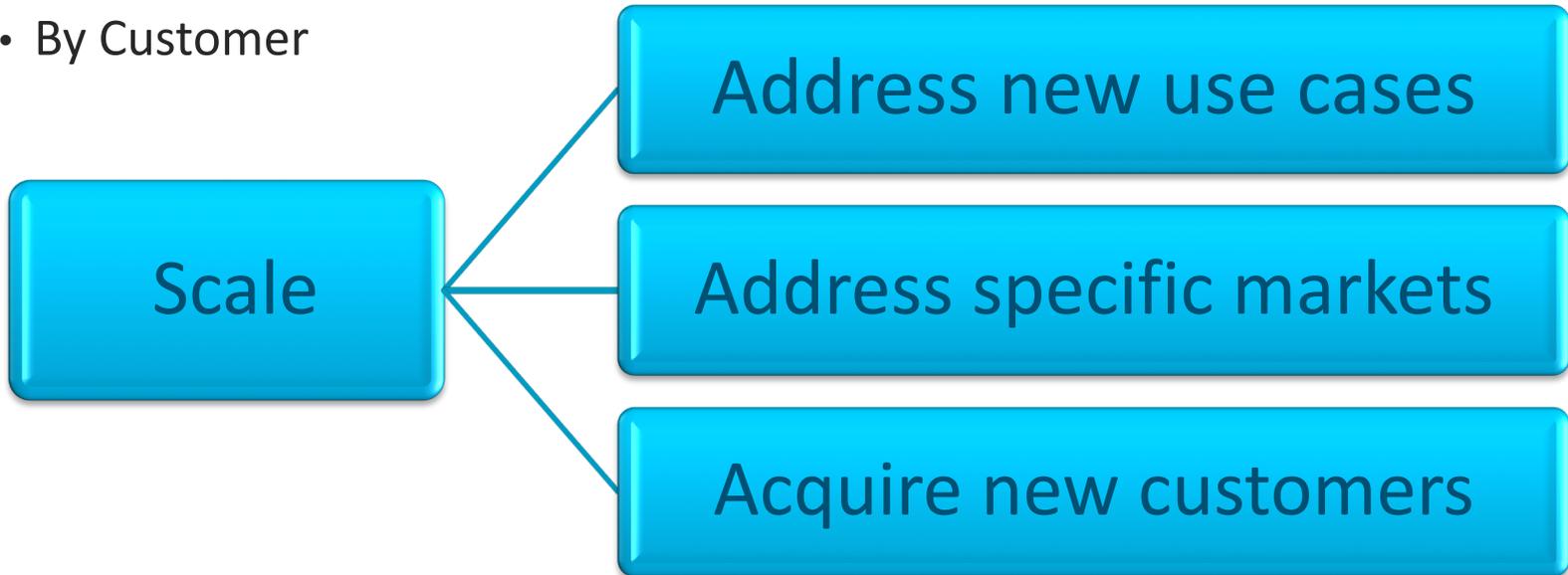
Shorten the time between regular releases that contain  
new data plane features and bug fixes



# Optimize Resources and Scale

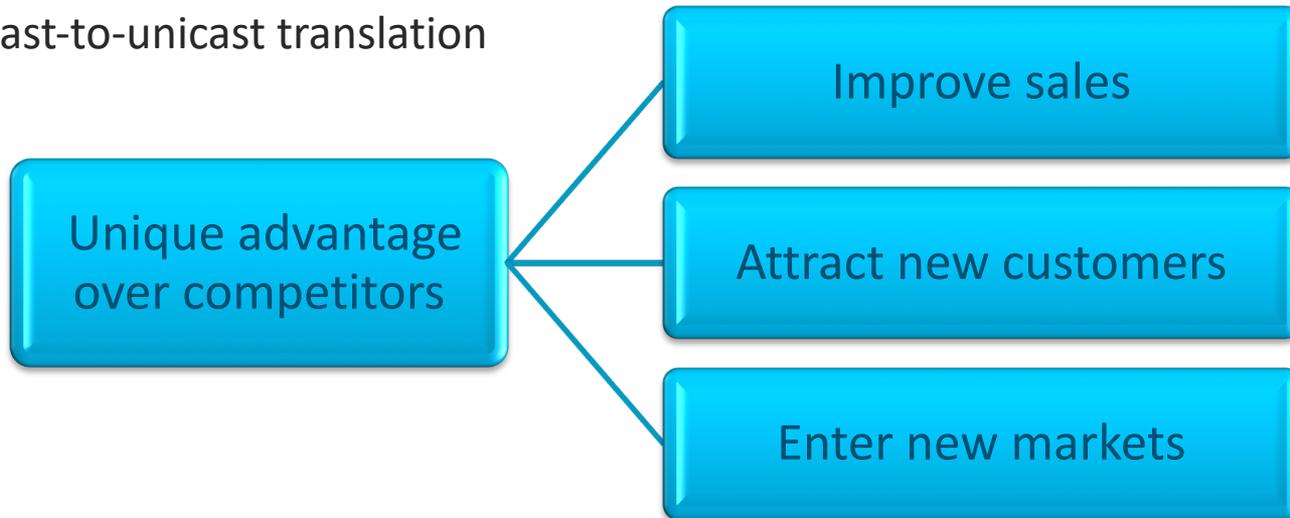
Enable (dynamic) feature selection

- By Vendor
- By Customer



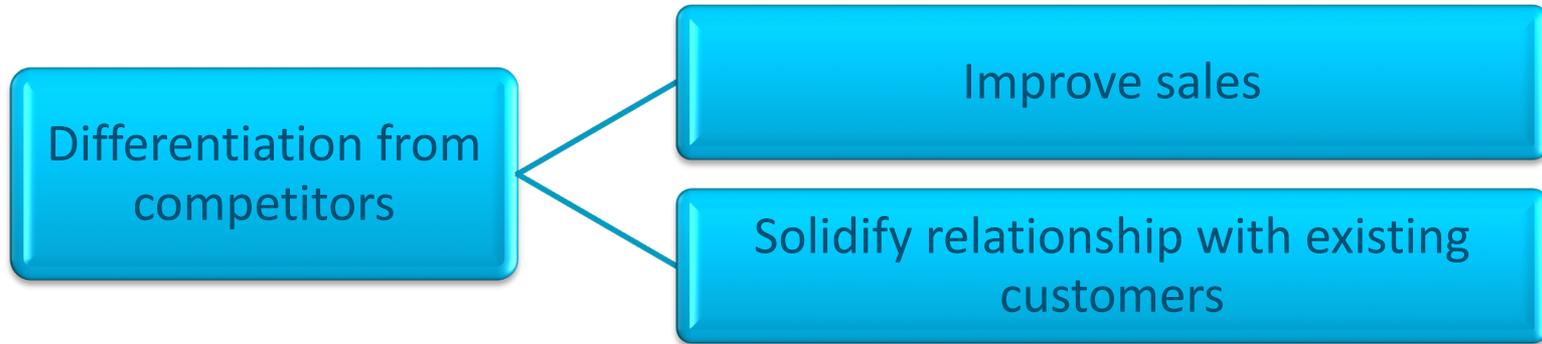
# Unique Features Not Available in ASICs

- 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



# Customer Requested Features

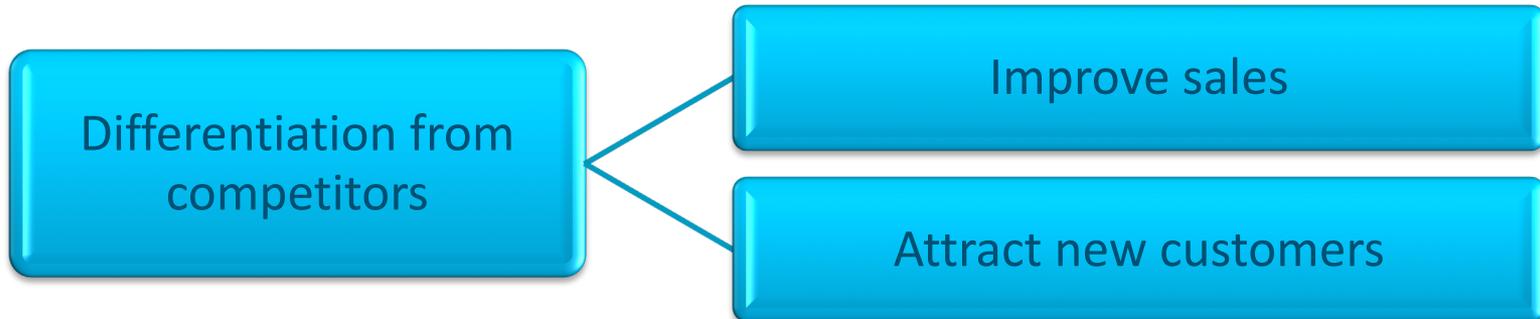
- Examples/use cases
  - Existing protocols not yet widely adopted
    - MAP-T, SRv6, BIER
  - New protocols just standardized or not yet stable or specific techniques
    - Multicast address translation



# 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)



# 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

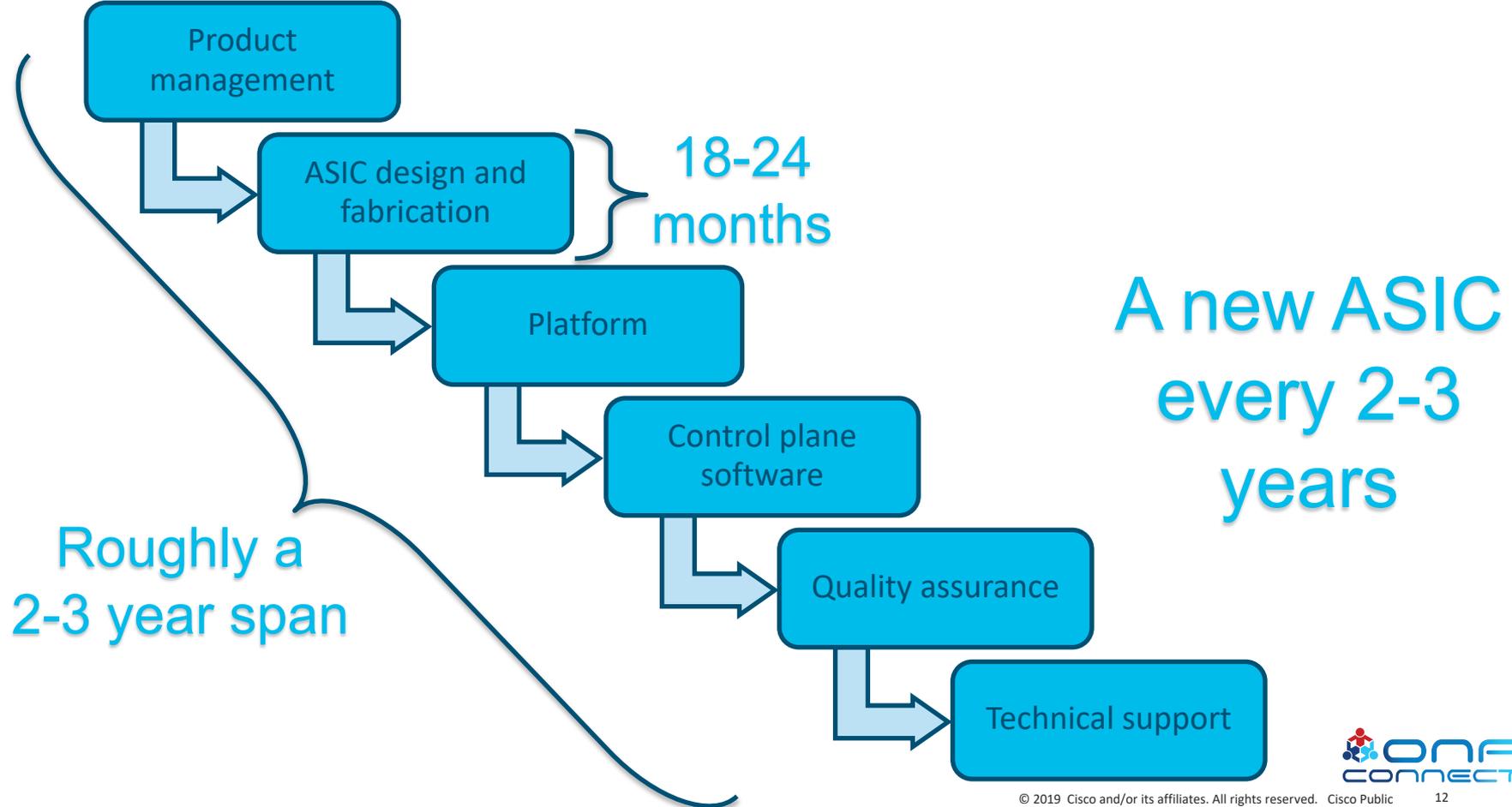
New "feature"

- Custom and third party development

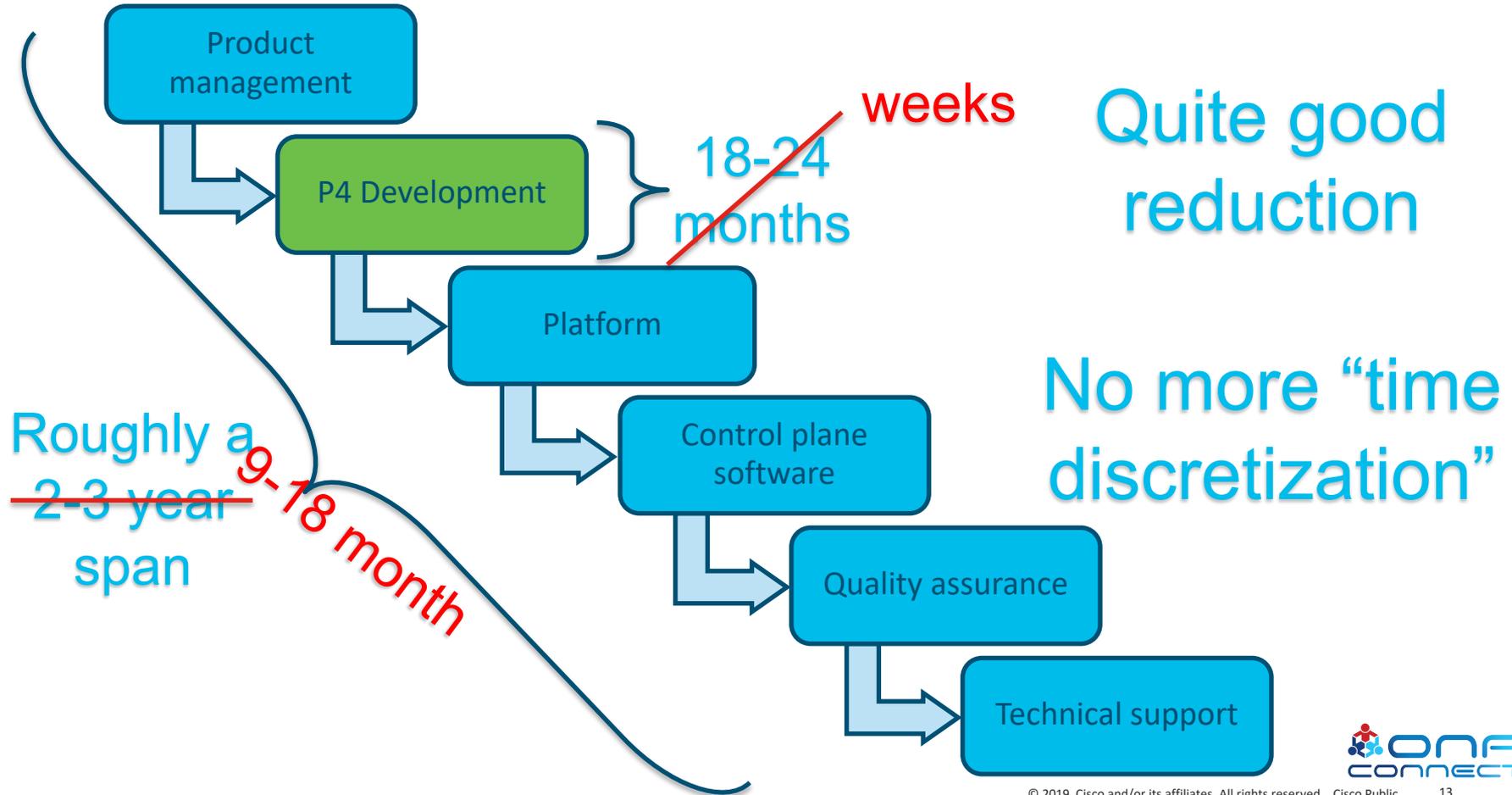
Is this reduction significant?

Goal: maximize the benefits

# Let's look at the process



# With a programmable data plane chip



# 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

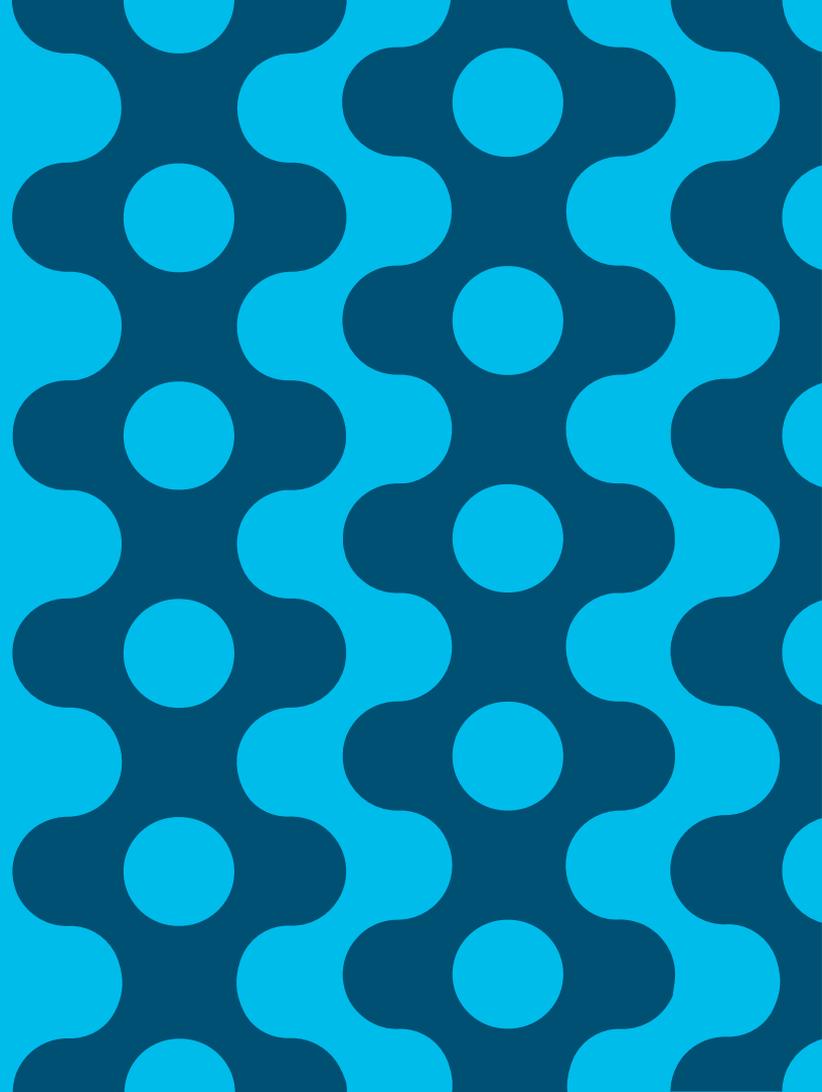


Heavy reliance on testing automation

# 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

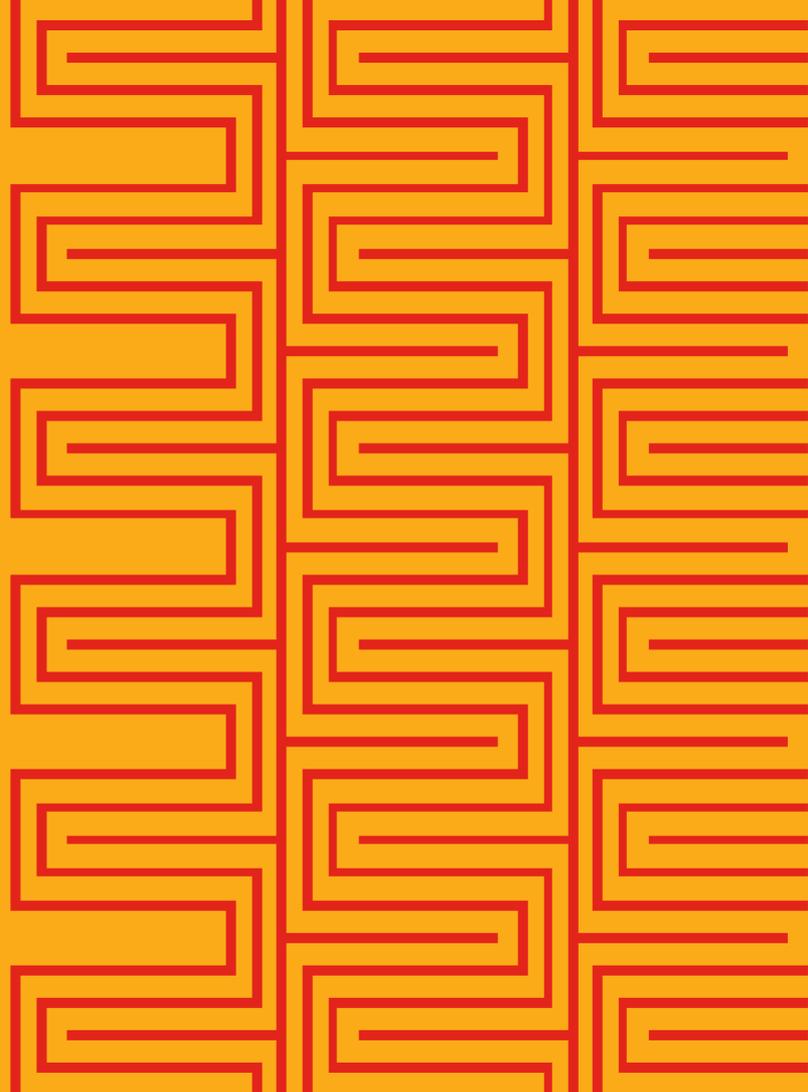


Modular/composable NOS



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: DAta Plane Incremental Programming Environment

Identify constraints  
on new code

Enforce those  
constraints on the  
program

## 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

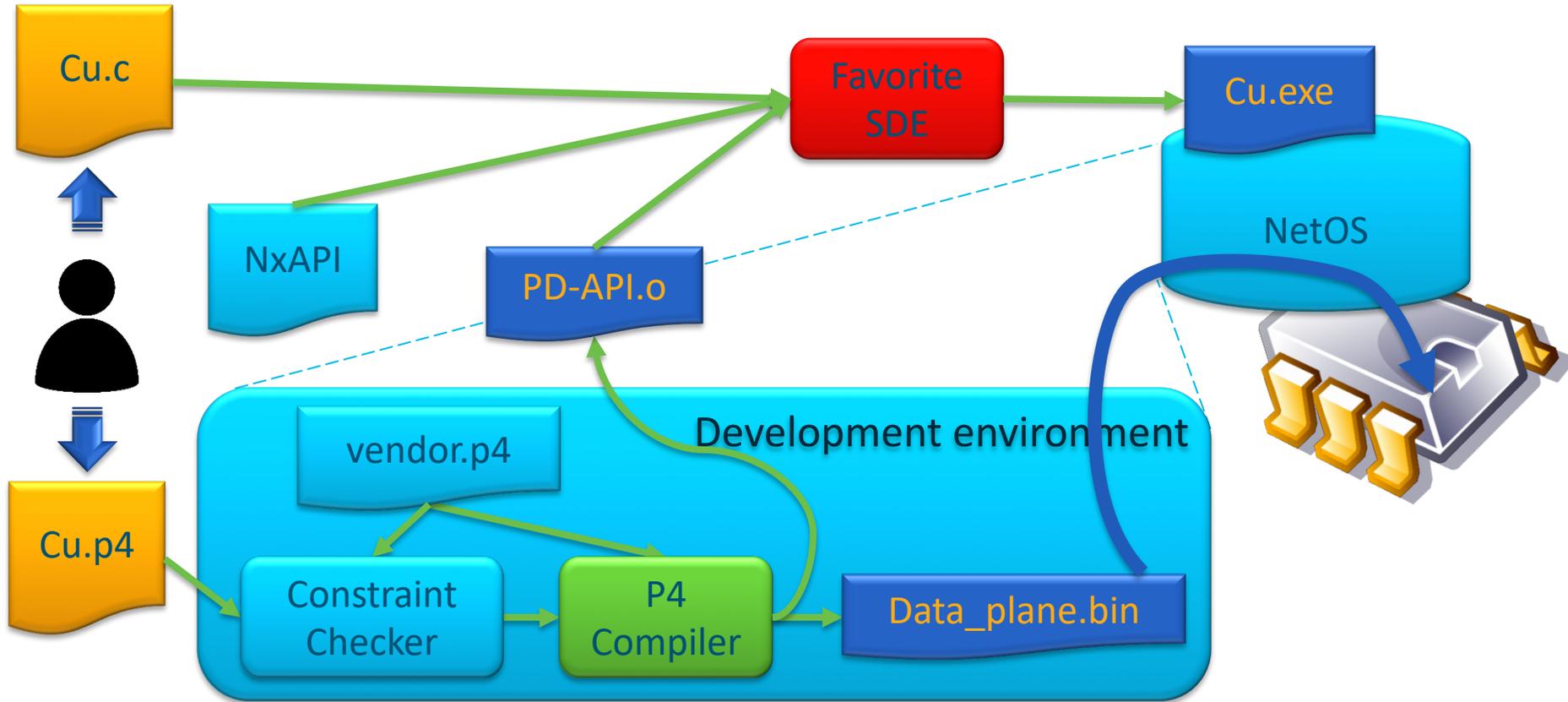
- Still "limited" on current chips

### Data/control plane dependence

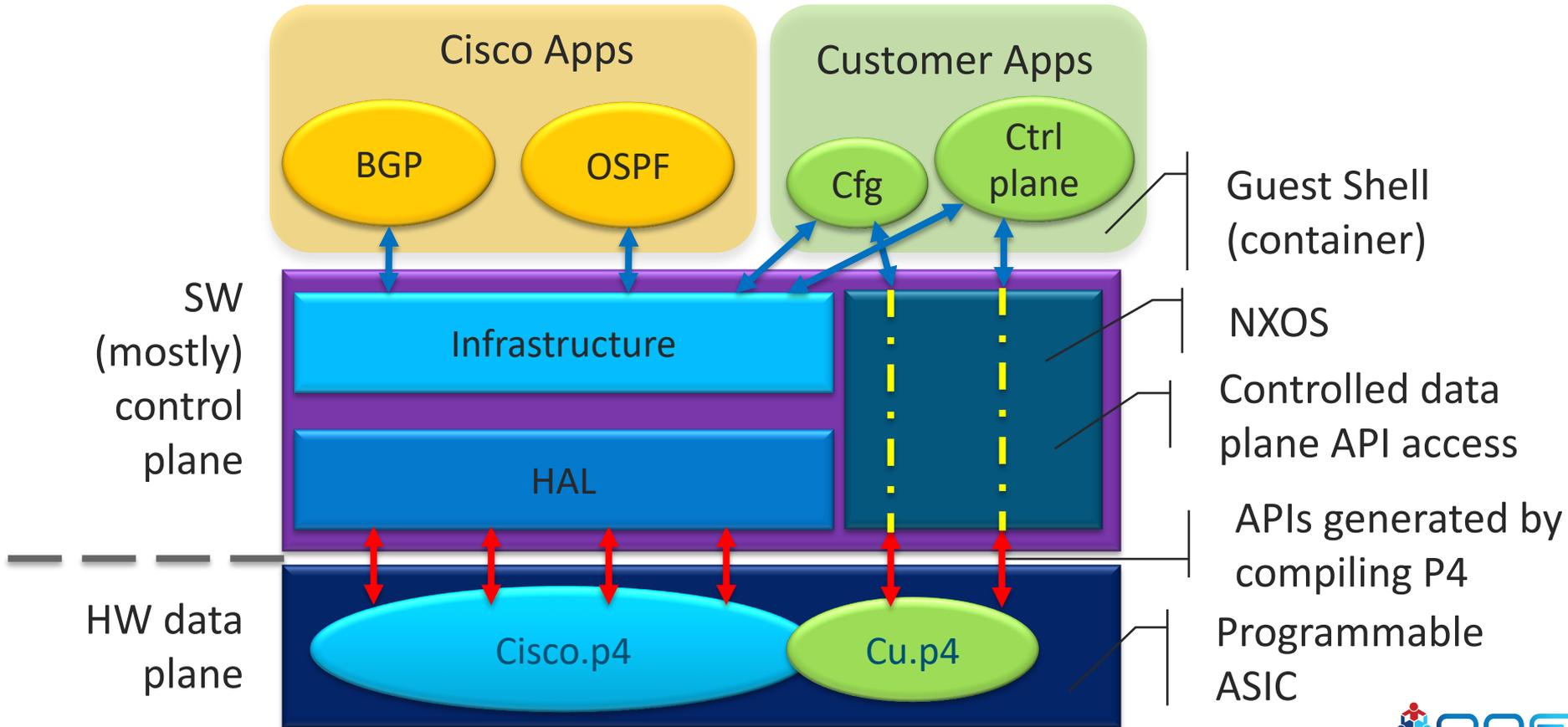
- NXOS should keep working
- NXOS should not be aware of custom data plane functions

© 2018 Cisco and/or its affiliates. All rights reserved. Cisco Public 33

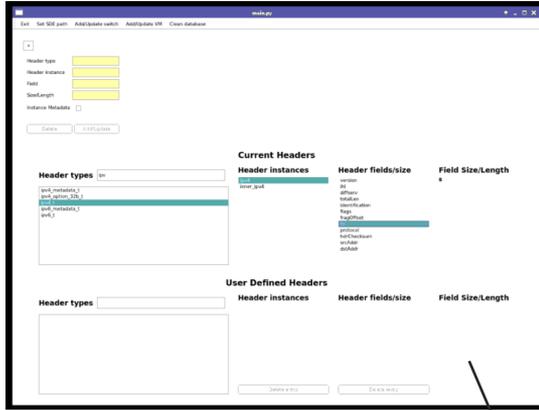
# Customer Programming Workflow



# Control Plane

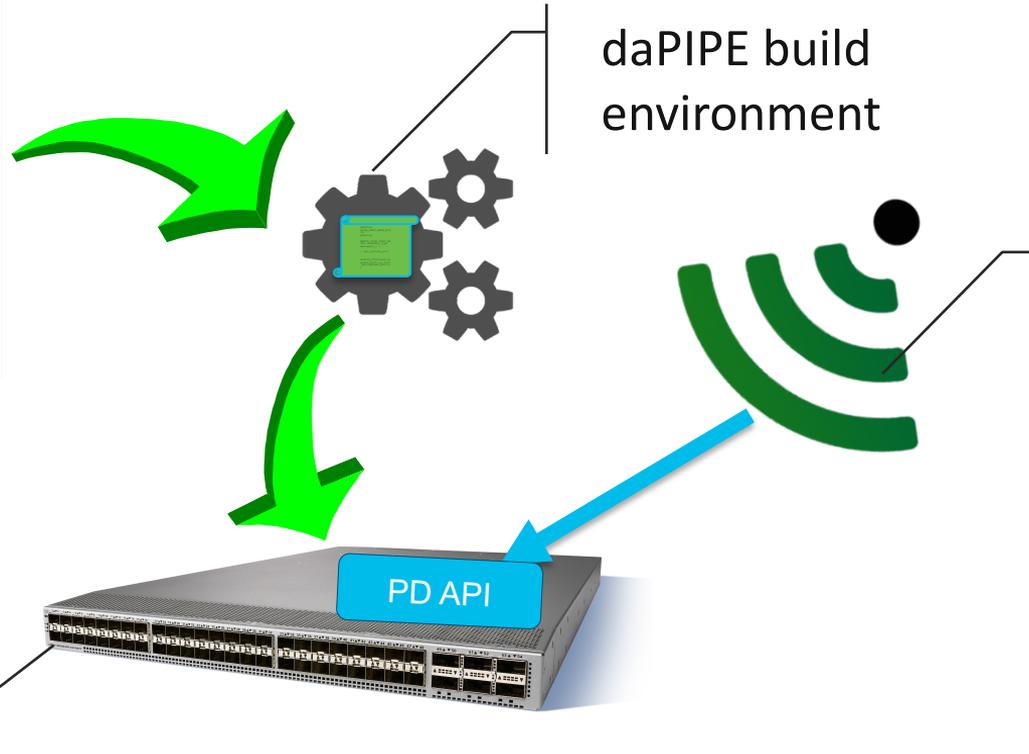


# Components of the Solution



daPIPE Graphical User Interface

Nexus 34180YC



daPIPE build environment

Control plane

# Main window



# Existing header view

main.py

Exit Set SOE path Add/Update switch Add/Update VM Clean database

Header type

Header instance

Field

Size/Length

Instance Metadata

Delete Add/Update

### Current Headers

Header types	Header instances	Header fields/size	Field Size/Length
<input type="text" value="ipv"/> ipv4_metadata_t ipv4_option_32b_t <b>ipv4_t</b> ipv6_metadata_t ipv6_t	<b>ipv4</b> inner_ipv4	version ihl diffserv totallLen identification flags fragOffset <b>ttl</b> protocol hdrChecksum srcAddr dstAddr	8

### User Defined Headers

Header types	Header instances	Header fields/size	Field Size/Length
<input type="text"/>			

Delete entry Delete entry

# Adding RTP parser

main.py

Exit Set SDE path Add/Update switch Add/Update VM Clean database Save Open

## Current Parsers

Selected parser: parse\_udp Selected hook point:

Parsers	Hook points	Potential Hook points
<input type="text"/> <ul style="list-style-type: none"><li>parse_set_prio_high</li><li>parse_set_prio_med</li><li>parse_sflow</li><li>parse_snap_header</li><li>parse_tcp</li><li><b>parse_udp</b></li><li>parse_vlan</li><li>parse_vxlan</li><li>start</li></ul> USER PARSERS:	udp.dstPort	

Parser name:   
Hook value:   
 Set as default

```
extract(rtp);  
return ingress;
```

Current parser:  
Current hook point:

## Available Headers

Header types	Header instances	Header fields/size	Field Size/Length
<input type="text"/> <ul style="list-style-type: none"><li>rtp_t</li><li>acl_metadata_t</li><li>egress_intrinsic_metadata_for_mirror_buffer_t</li><li>egress_intrinsic_metadata_for_output_port_t</li><li>egress_intrinsic_metadata_from_parser_aux_t</li><li>egress_intrinsic_metadata_t</li><li>egress_metadata_t</li><li>erspan_header_t3_t</li><li>ethernet_t</li><li>fabric_header_cpu_t</li></ul>			

Disclaimer

# Define control flow

Exit New Save Open Set SDE path Add/Update switch Add/Update SDE build env Clean database View PD-API

◀ Cisco Ingress Pipeline beginning  
Cisco Ingress Pipeline end  
Cisco Egress Pipeline beginning  
Cisco Egress Pipeline end  
New Control

Delete control Add to/Update pipeline

Disable Cisco Ingress pipeline Disable Cisco Egress pipeline

Controls

egress  
ingress  
USER Controls:

Tables

acl\_stats  
adjust\_lkp\_fields  
bd\_flood  
capture\_tstamp  
compute\_ipv4\_hashes  
compute\_ipv6\_hashes  
compute\_non\_ip\_hashes  
compute\_other\_hashes  
cpu\_packet\_transform  
dmac  
drop\_stats  
ecmp\_group  
egress\_bd\_map  
egress\_bd\_stats  
egress\_outer\_bd\_map  
egress\_port\_mapping  
egress\_system\_acl  
egress\_vlan\_xlate  
egress\_vni  
fabric\_ingress\_dst\_lkp

◀

blackbox name

blackbox Code

Add/Update pragmas

Delete Add/Update

Blackbox

Register

Update daPIPE Save PD-API bind Disclaimer

# Compile and upload to a switch

The screenshot shows the 'daPIPE - Compiling unit' web interface. The title bar includes 'daPIPE - Compiling unit' and window control icons. The menu bar contains: Exit, New, Save, Open, Set SDE path, Add/Update switch, Add/Update SDE build env, Clean database, and View PD-API. The main content area features a left sidebar with a back arrow icon. The form fields are as follows:

- Switch address:** A dropdown menu with 'admin@192.29.148.85' selected.
- Remote compilation:** A checked checkbox.
- Remote IP address:** A text input field containing '10.3.4.1'.
- Username:** A text input field containing 'mario'.
- Password:** A text input field with masked characters (dots).
- Remote BF-SDE path:** A text input field containing '/home/mario/bf-sde-8.2.2'.
- Use Pre-packaged SDE build Env:** An unchecked checkbox.

Below the form fields is a button labeled 'Open your P4 program'. At the bottom of the form area are four buttons: 'Compile', 'Compile & Upload', 'Restore', and 'Upload'. At the very bottom of the interface are two buttons: 'Save last compiled program' and 'Load a program'. The footer contains links for 'Update daPIPE', 'Save PD-API bind', and 'Disclaimer'.

This is “simple” ...

composable  
data plane modules

are the next big challenge



Thank You