daPIPE
Data Plane Incremental Programming Environment

Mario Baldi
Data Center Switching Group
Programmable switches:

What are deployment options?
Whitebox Deployment

- Maximum flexibility
- Maximum disruption/risk/work

NOS (e.g., Cumulus)

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

PD API/P4Runtime

customer.p4

Programmable chip

Platform vendor (Cisco)
Chip vendor (Barefoot)
Customer/open source
Turn-key Deployment

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

- No flexibility
  - No custom feature and protocol support

Profiles

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

Net OS

profile1.p4
profile2.p4
profile3.p4

Programmable chip
Hybrid Deployment

- Best of breed
- Deployment as usual
  - Familiar features and interfaces
- Minimum development effort
  - Leverage existing functions in building new features

Minimize disruption and risk!
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, we need an explicit effort to support **Incremental Programming**
How can we address these challenges?

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

Impose those constraints on custom code
Customer Programming Workflow

- Cu.c
- NetOS API
- PD-API.o
- vendor.p4
- Constraint Checker
- P4 Compiler
- Data_plane.bin
- Cu.exe
- Favorite SDE
- NetOS

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

Data Plane Incremental Programming Environment
Support developers and streamline their task (while enforcing constraints)
Components of the Solution

- daPIPE Graphical User Interface
- Nexus 34180YC
- daPIPE build environment
- Control program

```
#define FLOW_PORT_HASH_WID TH
#define EMPTY_FLOW_PORT_EN TRY 0

header_type
metadata_t {
    set_active_port
    modify_field(md.in(gress_port,ig_intr_md.ingress_port));
}
```
Nexus 3400 Programmable Switch Family

- Based on Tofino 1 by Barefoot
- 1.8/6.4 Tb/s aggregated switching capacity
- Flexible port configuration and multiple profiles for addressing different feature and scale requirements
- Inband Network Telemetry (INT) support

48p 10/25Gb/s SFP + 6p 40/100Gb/s QSFP
Nexus 34180YC

64p 40/100Gb/s QSFP
Nexus 3464C
daPIPE in Action
Sample Use Case
Fox Networks Advanced Technology Group

https://github.com/FOXNEOAdvancedTechnology/ts_switching_P4
Specification

• A switch shall forward packets based on the **RTP timestamp** they contain

• If sent to 239.1.1.1, change destination address to 239.3.3.3 when **RTP timestamp** is
  • Between 0 and 2
  • Between from 5 and F

• If sent to 239.2.2.2, change destination address to 239.3.3.3 when **RTP timestamp** is
  • Between 3 and 4
Development Workflow

• Browse available (stock) metadata
• Define custom headers and metadata
• Specify parser(s) and their hook(s) in existing (stock) parsers
• Define custom tables and actions
• Specify control flow
• Compile and load on chip
• Develop control plane functionalities
Main window

daPIPE v1.2.1

Disclaimer

By using this environment you are acknowledging that the available functions are limited and offered on an experimental basis. These work-in-progress features are not eligible for Cisco TAC support at this point. Please reach out to your Cisco representative for limited support offered by the product team. This environment and related documentation are confidential property of Cisco System and are provided to you under the terms of a Non-Disclosure Agreement (NDA).

Add Header  Add Action/Table
Add Parser  Add Control

Compile

Disclaimer
Existing header view
Adding RTP header
Adding RTP parser
New parser added
Resulting Parsing Code

```c
header_type ethernet_t {
    fields {
        dstAddr : 48;
        srcAddr : 48;
        etherType : 16;
    }
}
header ethernet_t ethernet;
...
header_type rtp_t {
    fields {
        version : 2;
        padding : 1;
        sequence_number : 16;
        timestamp : 32;
        SSRC : 32;
    }
}
header rtp_t rtp;
...
```

```c
parser parse_ethernet {
    extract(ethernet);
    return select(latest.etherType)
    {
        ETHERTYPE_IPV4 : parse_ipv4;
        default: ingress;
    }
}
parser parse_udp {
    extract(udp);
    return parse_rtp;
}
...
parser parse_rtp {
    extract(rtp);
    return ingress;
}
...
```
Add action
Adding a table
New table available
Define control flow
Compile and upload to switch
Control Plane

- Cisco Apps
  - BGP
  - OSPF

- Customer Apps
  - Cfg
  - Ctrl plane

- SW (mostly) control plane
- HW data plane

- Infrastructure
- HAL

- Cisco.p4
- Cu.p4

- APIs generated by compiling P4
- Programmable ASIC
- NXOS
- Guest Shell (container)

Controlled data plane API access
In summary

• daPIPE enables incremental programming
  • Cisco NX34xxx so far
  • Not platform specific
    • Any platform, any NetOS

• Developer can focus just on new features
  • Does not need to work on common features
  • Can leverage existing functions

• No need to deal with the complexity of stock P4 code

• Constrained changes ensure stock feature and NetOS integrity

• It does not address any possible use case, but it addresses many
Interested in giving it a try?

Get in touch with me (mariobal@cisco.com) ...

... and be willing to deal with the imperfections of something new