Programmable switches: What are deployment options?
Whitebox Deployment

- Maximum flexibility
- Maximum disruption/risk/work

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

NOS (e.g., Cumulus)

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

Programmable chip

PD API/P4Runtime

customer.p4

PD API/P4Runtime

PD API/P4Runtime

customer.p4

customer.p4
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

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

Profiles
- profile1.p4
- profile2.p4
- profile3.p4

Programmable chip

Net OS
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**
- **Favorite SDE**
- **Cu.exe**
- **NetOS**
- **Data_plane.bin**

---

© 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**

**Graphical User Interface**

**Nexus 34180YC**

**daPIPE build environment**

**Control program**
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
Sample Usecase
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
Existing header view

<table>
<thead>
<tr>
<th>Current Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header types</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>ipv4</td>
</tr>
<tr>
<td>ipv4v2_metadata</td>
</tr>
<tr>
<td>ipv6v2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Defined Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header types</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Current Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header fields/size</td>
</tr>
<tr>
<td>Field Size/Length</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Defined Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database entry</td>
</tr>
<tr>
<td>Delete entry</td>
</tr>
</tbody>
</table>
Adding RTP header
Adding RTP parser
New parser added
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;

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

SW (mostly) control plane

HW data plane

Cisco Apps

Customer Apps

Control Plane

Cisco.p4

Cu.p4

Infrastructure

HAL

Ctrl plane

BGP

OSPF

Cfg

Guest Shell (container)

NXOS

Controlled data plane API access

APIs generated by compiling P4

Programmable ASIC

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