Exposing Data Plane Programmability

Challenges and Options

Mario Baldi
Context

- A vendor building a switch
  - With a programmable data plane chip (e.g., Barefoot Tofino),
  - Running an operating system
  - Giving customers the possibility of adding features to the data plane
    - In addition to the ones provided by the vendor

- Constraints
  - Customers should be able to develop and test their data plane feature independently
    - No intervention by the vendor
    - No changes to the operating system
  - Customers should not have access to the vendor data plane and operating system code
Architecture

- **Vendor Apps**
  - BGP
  - OSPF

- **Customer Apps**
  - Ctrl plane
  - Cfg

- **Infrastructure**
  - HAL

- **HW data plane**
  - Vendor.p4
  - Cust.p4

- **SW (mostly) control plane**

- **Tofino**

- **Network OS**

- **Guest Shell (container)**

- **APIs generated by compiling P4**
Challenges

- P4 Compilation
  - HW linker not available
    - Not clear whether on Barefoot/P4.org roadmap and if so when

- Network OS-related
  - Provide access to customer P4-related APIs, while limiting access to vendor P4-related APIs

- Development and testing environment
  - Customer should be able to develop and test code interacting with vendor P4 features and Network OS APIs

- Regression testing for Vendor.p4 and Network OS stock features/functionalities
  - Made available to customers (e.g., through a web service)
P4 Compilation-related Issues

- Current compiler operates on a single .p4 program
- P4 language does not support modularity
  - Compiler must work on the whole code when compiling
  - Developer must know/understand the whole code
    - Vendor-Customer IPR contamination
    - Breadth of knowledge
- Adding new code might affect functionality of existing code
  - No compiler nor hardware support for programming interfaces or protection
Responsibilities

- **Vendor**
  - Develop a universal Network OS API for applications accessing any API generated by compiling customer P4 code

- **Customer**
  - Write P4 code for their custom functionalities
  - Write Network OS applications for configuration and control plane

- **???**
  - A compiler/SDE/linker that allows to “link” newly compiled P4 code (e.g. Cu.p4) to pre-compiled P4 code (e.g., Vendor.obj)
  - Environment that supports incremental development and checks that functionality and API of pre-existing code does not get altered/affected
    - Software
    - Hardware (with some support software)
Dedicated Customer Pipelines

Physical separation of vendor and customer functionality within a single chip

- N pipelines of Tofino, connected to MACs, run vendor P4
- M pipelines of Tofino run customer P4
- Packets can move internally or externally between pipelines
- Vendor.p4 and Cu.p4 can be compiled separately

- Minor modification to compiler and loader of binary onto the pipelines
- Long latency and lower throughput/port count
Customer Programming Workflow

Cu.c → Favorite SDE → Cu.exe

NetOS API
Possibly integrated in Network OS

Cu.p4 → P4 Compiler → Cu.bin

Other pipelines pre-loaded with Vendor.bin

Loaded on dedicated customer pipeline
Vendor and customer P4 code execute on the same pipeline

A development environment must be provided to support code writing under such conditions

Available P4 testing environment can be used as-is
Customer Hardware Programming Workflow

Cu.c → Favorite SDE → Cu.exe

Cu.p4 → NetOS API → NetOS

P4 Compiler → "Linker" → Cu.hw

Vendor.obj → "Linker" → Cu.hw

Cu.obj → "Linker" → Cu.hw
Network OS API can be used by customer application(s) to access data related to vendor P4 (stock) functionality

- Pass-through Network OS API enables customer application(s) to manipulate data plane tables added by customer P4
  - Network OS prevents direct access to vendor P4 tables
- Network OS offers opaque access to custom data plane tables and telemetry data through CLI
  - Similar to Tofino CLI
We might be able to run a separate operating system for the customer pipeline
  - Does it make things simpler or more complex?
  - Could be a different Network OS

We need also testing environment
  - BMv2 as well as the Tofino register-level model cannot be used in this case