OpenFlow enables control plane programmability...

... but programming a Controller App with OF is hard

- Mismatch Between Application Expectations and Reality
  - Expectations:
    - Homogeneous forwarding model
    - Sufficiently large flow tables
    - Predictable feature set and performance
    - Switch state known / deltas efficiently reconcilable
    - Support for fail-over
  - Reality: Switch landscape is highly heterogeneous
    - Data Plane:
      - Hardware vs. software
      - Supported matches + actions
      - Table count and sizes
    - Control Plane:
      - Rule updates (consistency, churn rate)
      - Counters
      - OpenFlow version + vendor extensions
  - Reality (II): OpenFlow primitives do not match expectations
    - With switch-side flow-expirations, flow table state is unknown
    - Barrier semantics switch dependent
    - No efficient reconciliation of changes after disconnect
  - Tradeoff between portability and performance?
  - So far: Onix, POX, Frenetic, ...
    - Manage the entire network
    - Provide a simplified network-wide programming model, controller distribution, consistent updates, composability,...
    - This requires making assumptions ➞ optimize for a particular programming model
    - All have to be adapted for each individual switch [class]
    ➞ Duplication of effort

A Missing Piece in the Stack?

- A common API that provides the common primitives
  - local - controlling one switch
  - ubiquitous - services (almost) everyone needs
  - expressive - does not constrain the OpenFlow programming model
- Principles
  - Applications expose expectations to the switch
  - Vendors provide switch drivers in the controller

Architecture

Core Concepts

- VFT: Virtualized Flow Tables
  - Created by the Application
  - Full Feature Set
  - No resource constraints
  - Per table consistent updates
  - Annotations describe application expectations
- Switch drivers
  - Map the Virtual Flow Table to the switch reality
  - Can virtualize resource constraints
    - E.g., rule paging to map 50k rules to 2k table entries
  - Optimize for switch specifics
    - E.g., knowledge of exact BARRIER Semantics

Benefits

- Decouples
  - Rule update semantics and mechanism at the switches
  - Application-specific and switch-specific performance optimizations
- Enable protocol innovations by the vendors, e.g.,
  - built-in transactions for updates
  - efficient ruleset reconciliation after disconnect
- Annotations
  - provide a knob to choose between portability and performance

OpenFlow Protocol

Controller

Forwarding Table (TCAM)

CLI

Web-IP

SNMP

Control OS

OS Kernel

Driver Interface

VFT:

Virtualized Flow Tables

Table 0

Table 1

Table n

Action Set

Applications

Rule updates
with transactions

Events, counters

Virtual Flow Tables for a single switch

Rule mgt

Updates

Perf. Opt

Switch Driver for HP

Switch Driver for OVS

HP Switch

firmware

OVS

Server CPU

TCAM

SRAM

Userspace

Kernel

Building block for higher level controller frameworks

Enables direct, portable development of low-level apps

JVM

MySQL

simple.c

POSIX

OS Kernel

ONIX

Frenetic

SimpleApp

vs.

NOSIX

Driver Interface

HP

OVS

...