P4 in ONOS and CORD

Andrea Campanella
ONF
P4 Day, Politecnico of Milan, 2/2/18
Slides
What is ONOS?

Open Network Operating System (ONOS) is an open source SDN network operating system. Our mission is to enable Service Providers to build real SDN/NFV Solutions.

Java

Open Source: Apache 2.0

Github: https://github.com/opennetworkinglab/onos

Quarterly Releases, Magpie (1.12.0) - released 2017-12
P4, P4Runtime

Concepts and workflow
P4, the ONOS developer perspective...

**Fixed-function dataplane**

- **ONOS**: Hard to code, debug, manage!
- **A** → **COMPLEX** → **PIPELINE** → **THAT** → **DOES** → **EVERYTHING** → **...POORLY**

**Programmable dataplane**

- **ONOS**: Easy to code debug, manage!
- **my_pipeline.p4** → **P4Runtime** → **P4 program**
- **ONOS**: Easy to code debug, manage!
- **JUST** → **WHAT**
- **1** → **NEED**
P4Runtime

- Framework for **runtime control** of P4 devices
  - Open-source: [https://github.com/p4lang/PI](https://github.com/p4lang/PI)
- Targeted for **remote controllers**
  - Protobuf + gRPC implementation
- **P4 program-independent**
  - API doesn’t change with the P4 program
- Device does not need to be fully programmable
  - Can be used on fixed-function devices
- Enables **field-reconfigurability**
  - Ability to push new P4 program to the device
- Defines **interaction with the entities** of a P4 program
  - Tables, counters, meters, externs, etc.
- **Bidirectional** stream channel
  - For packet-ins/outs, mastership updates, etc.
P4 support in ONOS
Challenge

- How can we control and configure P4-enabled devices?
- ONOS initially designed around OpenFlow fixed-function dataplane
  - NB abstractions morphed around OpenFlow (e.g. same match/actions)
  - Immutable pipeline
- With P4…
  - Generalized forwarding abstraction (e.g. arbitrary match/actions)
  - Mutable pipeline (devices can support different pipelines in time)
Architecture overview

Pipeline-agnostic applications

PD APIs
- Flow Rule
- Flow Objectives
- Intents

Events
- Packet, Topology, etc.

Pipeline-aware application

PI APIs
- Pipeline-specific entities

Device
- Tofino, BMv2, etc.

PD-to-PI translation serv.
- (flow rule, groups, etc.)

PI models
- (table, match, actions, groups, counters, etc.)

Pipeconf Store

PI Framework

Core

Other drivers
- Other drivers

Default drivers
- Tofino
- BMv2
- P4Runtime
- gNMI

Driver

Protocol

Other protocols
- gRPC

Other protocols
- P4info, bin, JSON

Device
- (Tofino, BMv2, etc.)

Other protocols

Other protocols
Deploy a P4-defined Pipeline

Store and deploy P4-defined pipelines

Pipeconf: data and code necessary to let ONOS understand a P4 program

1. Pipeline entities description (i.e. parsed P4 programs)
2. Device and control-protocol specific data: Target dependent compiled P4 program binaries (e.g. for Tofino, BMv2, etc.), P4Info
3. Pipeline-specific code (e.g. Pipeliner)

Pipeconf is distributed → available to all ONOS components
Device discovery and Pipeline Deploy

After Successful Handshake the Device

- Has P4 defined pipeline
- Is installed is known to ONOS
- Is ready for Entity manipulation and traffic flow

→ From ONOS APIs to PI entities such as Table entries, counters, packets I/O, etc.
→ Install, Remove, Modify PI entities on P4 defined Pipelines with P4Runtime
Flows, Group, Meters, Packet I/O

From Northbound ONOS API to P4Runtime proto message through translation we support:

- Flow Rules (Table Entries)
- Groups (Action Profiles)
- Meters
- Packet I/O

Full management and handling of traffic flow in the network
Mastership Arbitration

ONOS is also capable of performing Mastership Arbitration on P4 Runtime Devices

Controller redundancy and failover
Takeaway

Today ONOS offers capabilities to:

- **Store** and **deploy** P4-defined pipelines (PI Pipeconf Service)
- **Control** pipeline entities (standard ONOS NB APIs)
  - Table entries, counters, packets I/O, etc.
- Integrate P4 devices into **heterogenous networks** and them through the same high-level APIs
- Control both HW and SW

While maintaining **high availability, scalability and performance** key characteristics to the ONOS platform
L123 Demo
Google tor.p4

Edgecore Wedge100BF-32X
Edgecore Wedge100BF-65X

P4 Runtime

https://youtu.be/BE_y-Sz0WnQ
Current Efforts

- P4 Fabric
- SPGW VNF offloading to HW
  - Serving Gateway
  - Packet Data Network Gateway
- INT (in-band Network telemetry)
- Exploratory work around Dynamic program loading
  - Swap a program on a device at runtime
fabric.p4: P4-based CORD Fabric

● **Goal:** bring more heterogeneity in the CORD fabric with P4 silicon
  ○ e.g. Barefoot Tofino, Cavium XPliant, Mellanox

● **P4-based underlay**
  ○ Design a P4 pipeline (fabric.p4) that is functionally equivalent to the OF-DPA one but much simpler
  ○ Use fabric.p4 as a drop-in replacement for the current Trellis underlay
    ■ Do not change the ONOS application programming the pipeline

● **Long-term - offload x86 processing to fabric**
  ○ P4-based overlay, i.e. move VXLAN handling from OVS to the ASIC
  ○ CORD VNFs offloading (will come to this later)
fabric.p4: where we are today

- Prototype P4 code and ONOS driver for fabric.p4 (Pipeliner)
  - Under onos/pipelines/fabric

```
In port + VLAN filtering table

Forwarding classifier

Bridging  MPLS  IPv4 unicast routing  IPv4 multicast routing  IPv6 unicast routing  IPv6 multicast routing

ACL

Next id mapping

Unicast  Hashed (WCMP)  Broadcast
```
Current Support

Fabric.p4 supports:

- Bridging
- Unicast
- ARP
- ACL

→ can replace and/or co-exist with the OFDPA based Underlay Fabric in CORD and in general be used in Data-Center deployments
VNF Offloading
spgw.p4
VNF offloading

- Programmable data planes offer great degree of flexibility beyond plumbing

<table>
<thead>
<tr>
<th>Progr. ASIC capabilities</th>
<th>VNF building blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary header parsing/deparsing</td>
<td>Domain specific encap/decap (e.g. PPPoE termination, GTP, etc.)</td>
</tr>
<tr>
<td>Stateful memories</td>
<td>TCP connection tracking (L4 load balancing, NAT, firewall, etc.)</td>
</tr>
<tr>
<td>Computational capabilities</td>
<td>Billing</td>
</tr>
</tbody>
</table>

- Benefits
  - Scalability - VNFs executed at wire speed
  - Low latency and jitter - avoid non-determinism of x86 processing
Serving Gateway (SGW)
- UE handovers with neighboring eNodeB's
- data transfer in terms of all packets across user plane.

Packet Gateway (PGW)
- connectivity from the UE to external packet Data network PDN (e.g. Internet)
- point of entry or exit of any traffic for the UE.

User Equipment  Antenna
E.g. Phone

VNF on x86
SPGW user plane functionality

- **GPRS Tunneling** Protocol (GTP)
- **Encapsulation/decapsulation** of uplink traffic to the Internet (Figure 2) and downlink traffic to the User Equipment (UE)
- **Bearers mapping**
- **Traffic filtering, gating**
- **Charging**
Advantages of SPGW.p4

By moving the SPGW functionality from VNF to HW we obtained:
  - increased performance
  - reduced latency and jitter

→ all on programmable and commodity hardware switches, readily available and cheap compared to dedicated S/PGW hardware.
→ no need for additional x86 servers.
→ updatability and functionality optimization in the future at wire speed
INT
In Band Network Telemetry
In Band Network Telemetry

- Enhanced **network visibility** with In-band Network Telemetry
- Provides a framework for **collecting and reporting network state by the data plane with no added packets** (e.g. Switch ID, timestamp, Ingress/Egress Port ID Link Utilization, Hop Latency, Egress Queue Occupancy, Egress Queue Congestion Status)
High level architecture

External Collector

Aggregated INT metadata

Metadata (e.g., Topology, switch info)

INT metadata

OPNFV Barometer -Prometheus

INT Monitoring Application (Configuration, Visualization, Interface)

PD APIs
Flow Rule
Flow Objectives
Intents

Events
(Packet, Topology, etc.)

PI APIs
Pipeline-specific entities

INT-capable Pipeconf (.oar)

PD-to-PI translation serv.
(flow rule, groups, etc.)

PI models
(table, match, actions, groups, counters, etc.)

Pipeconf Store

Core

Driver

Protocol

- Install INT monitoring flow rules
- Specify metadata types

Device (Tofino, BMv2, etc.)

External Collector

Metadata (e.g., Topology, switch info)

INT metadata

INT Monitoring Application (Configuration, Visualization, Interface)

OPNFV Barometer -Prometheus

INT-capable Pipeconf (.oar)
INT Benefits

Examples of INT insights:

- **Path calculation**: network elements visited by the packet
- **Latency/Delay**: time spent in queues and buffers per device
- **Rules and groups**: rules it matched upon at every device along the way.
- Custom topology and network management thanks to custom headers.

Everything with **no added overhead** in the network

• Machine learning and algorithms to optimize in real time the network
• Real time **anomaly detection** and pin-point

Network Debugging made easy
Configuration

gNMI

Generic Network Management Interface
Evolving control and configuration

control

configuration

ONOS

control

config

P4Runtime

gNMI

P4Runtime

OpenConfig

ONOS

P4
gNMI: configuration

- RPCs and behaviors for managing state on a device
- built on the open source gRPC framework (gRPC ⊂ gNMI)
- Port description, port statistics, manage LEDs, etc.
- P4.org API WG suggests using existing OpenConfig Yang based data models → not reinventing the wheel
  - Currently OpenConfig Yang models are supported in ONOS
- gNMI → support from BMv2, PI (switch-side server for runtime control)
P4 Brigade

1. Looking for people with knowledge of both P4 and ONOS

2. Work on
   a. P4 Runtime enhancements (groups, meters, device mastership)
   b. gNMI
   c. Use cases (fabric.p4, INT, VNF HW offloading)
   d. Dynamic program swapping

3. To learn more:
   a. Wiki - [https://wiki.onosproject.org/display/ONOS/P4+brigade](https://wiki.onosproject.org/display/ONOS/P4+brigade)

4. Mail at
   a. [brigade-p4@onosproject.org](mailto:brigade-p4@onosproject.org)
By email: andrea@opennetworking.org